Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομέας: Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής
Διπλωματικη Εργασια

Δυναμικός Προγραμματισμός Διεργασιών σε Περιβάλλοντα Knative

Εκπονήτρια:
Διαμαντή Χριστίνα

Επιβλέπων:
Συμεών Παπαβασιλείου
Καθηγητής & Διευθυντής
του Εργαστηρίου NETMODE

Η παρούσα διπλωματική εργασία εκπονήθηκε και γράφτηκε στο

Εργαστήριο Διαχείρισης Δικτύων και Βέλτιστου Σχεδιασμού (NETMODE)

Αθήνα, Οκτώβριος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομέας: Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής

Διπλωματικη Εργασια

Δυναμικός Προγραμματισμός Διεργασιών σε Περιβάλλοντα Knative

Εκπονήτρια: Διαμαντή Χριστίνα

Επιβλέπων Καθηγητής: Συμεών Παπαβασιλείου, Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Οκτωβρίου 2024
Τριμελής Εξεταστική Επιτροπή:

Σ. Παπαβασιλείου, Καθηγητής Ε.Μ.Π.	Ι. Ρουσσάκη, Καθηγήτρια Ε.Μ.Π.	Β. Καρυώτης, Καθηγητής Ι.Π.

Ημερομηνία Έγκρισης: Αθήνα, Οκτώβρης 2024

.....
Διαμαντή Χριστίνα
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Sector: Communications, Electronics & Information Systems
DIPLOMA THESIS

Design and Implementation of a Dynamic Load Balancer Based on AIMD for Knative

---

Author:
Diamanti Christina

Supervisor:
Symeon Papavassiliou
Professor & Director of the
NETMODE Lab

This thesis was conducted and written at the

Network Management and Optimal Design Laboratory (NETMODE)

Athens, October 2024

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την πρόκληση του δυναμικού χρονοπρογραμματισμού διεργασιών και της εξισορρόπησης φορτίου σε περιβάλλοντα χωρίς διακομιστές, συγκεκριμένα στα πλαίσια του Knative. Καθώς οι σύγχρονες cloud-native εφαρμογές αντιμετωπίζουν κυμαινόμενα φορτία εργασίας, η διατήρηση της βέλτιστης απόδοσης και της αποδοτικής χρήσης πόρων καθίσταται κρίσιμη, ιδίως σε περιβάλλοντα κατανεμημένου και περιφερειακού υπολογισμού. Η εργασία που παρουσιάζεται εδώ προτείνει τον σχεδιασμό και την υλοποίηση ενός προσαρμοσμένου εξισορροπιστή φορτίου, βασισμένου στον αλγόριθμο Additive Increase Multiplicative Decrease (AIMD), ο οποίος ενσωματώνεται στο οικοσύστημα Knative. Αυτός ο εξισορροπιστής φορτίου προσαρμόζει δυναμικά τους ρυθμούς αποδοχής των εισερχόμενων αιτημάτων, διασφαλίζοντας την αποτελεσματική κατανομή της κίνησης σε πολλαπλές υπηρεσίες.

Η λύση αξιοποιεί τόσο το Knative Pod Autoscaler (KPA) όσο και το Horizontal Pod Autoscaler (HPA) για τη διαχείριση της κλιμάκωσης των πόρων, σε απόκριση σε πραγματικού χρόνου μοτίβα κίνησης. Η ενσωμάτωση του RabbitMQ για τη διαχείριση των μηνυμάτων και του Redis για τη διαχείριση της κατάστασης επιτρέπει την απρόσκοπτη επικοινωνία μεταξύ των ελεγκτών αποδοχής και των υπηρεσιών κατανάλωσης. Πραγματοποιήθηκαν εκτεταμένες δοκιμές και αξιολογήσεις επιδόσεων υπό διάφορες συνθήκες φόρτου, συμπεριλαμβανομένων σταθερών και εκρηκτικών μοτίβων κίνησης. Τα αποτελέσματα καταδεικνύουν ότι η προσέγγιση που βασίζεται στο AIMD εξισορροπεί αποτελεσματικά την κυκλοφορία, αποτρέπει τη συμφόρηση στις ουρές αναμονής και κλιμακώνει δυναμικά τους πόρους, οδηγώντας σε βελτιωμένη απόδοση του συστήματος.

Εκτός από την αντιμετώπιση του βασικού ζητήματος της εξισορρόπησης του φορτίου, η διατριβή εξετάζει τη δυνατότητα ενσωμάτωσης πιο εξελιγμένων μηχανισμών αυτόματης κλιμάκωσης. Η μελλοντική εργασία περιλαμβάνει την υιοθέτηση μοντέλων πρόβλεψης φόρτου εργασίας και σκιαγράφησης πόρων, βασισμένων στη μηχανική μάθηση, για την περαιτέρω βελτιστοποίηση των αποφάσεων κλιμάκωσης. Η προτεινόμενη λύση προσφέρει μια ευέλικτη και κλιμακούμενη αρχιτεκτονική για τη δυναμική διαχείριση πόρων σε περιβάλλοντα χωρίς διακομιστές, συμβάλλοντας στον ευρύτερο τομέα του cloud-native και του edge computing.

Λέξεις Κλειδιά

εξισορρόπηση φορτίου, αυτόματη κλιμάκωση, υπολογιστικό νέφος, Knative, AIMD , microservices, διαχείριση πόρων, cloud-native αρχιτεκτονικές, αρχιτεκτονική serverless

Abstract

The present thesis deals with the challenge of the dynami- scheduling of processes and load balancing in a process environments without servers, specifically in the context of Knative. modern cloud-native applications are faced with fluctuating workloads and workloads, maintaining optimal performance and efficient resource utilization becomes critical, especially in distributed and peripheral environments. computing. The work presented here proposes the design and implementation of a custom load balancer based on the Additive Increase Multiplicative Decrease (AIMD) algorithm, which integrates the integrated into the Knative ecosystem. This load balancer adap- dynamically adjusts the acceptance rates of incoming requests, ensuring that ensuring efficient distribution of traffic across multiple services. The solution leverages both the Knative Pod Autoscaler (KPA) and the Horizontal Pod Autoscaler (HPA) to manage the scaling of resources, from in response to real-time traffic patterns. The integration of RabbitMQ for message management and Redis for managing the message flow and the allows for seamless communication between the admission controllers and the and consumption services.Extensive testing and performance evaluations were carried out under various load conditions, including steady and explosive driving patterns. The results demonstrate that the AIMD-based approach effectively balances traffic, prevents congestion in queues and dynamically scales resources, leading to improved system performance.

In addition to addressing the key issue of load balancing, the thesis explores the possibility of incorporating more sophisticated automatic scaling mechanisms. Future work includes the adoption of workload prediction and resource profiling models based on machine learning to further optimize scaling decisions. The proposed solution offers a flexible and scalable architecture for dynamic resource management in serverless environments, contributing to the broader field of cloud-native and edge computing.

Keywords

load balancing, autoscaling, cloud computing, Knative, AIMD, microservices, resource management, cloud-native architectures, serverless architecture

Περιεχόμενα

Κατάλογος σχημάτων

Λίστα Συντομογραφιών

VMs	**V**irtual **M**achines
OS	**O**perational **S**ystem
IaaS	**I**nfrastructure as a **S**ervice
PaaS	**P**latform as a **S**ervice
SaaS	**S**oftware as a **S**ervice
BaaS	**B**ackend as a **S**ervice
FaaS	**F**unction as a **S**ervice
EDA	**E**vent **D**riven **A**rchitecture
QoS	**Q**uality of **S**ervice
AIMD	**A**dditive **I**ncrease **M**ultiplicative **D**ecrease
k8s	**K**ubernetes
HPA	**H**orizontal **P**od **A**utoscaler
VPA	**V**ertical **P**od **A**utoscaler
CRD	**C**ustom **R**esource **D**efinition
KPA	**K**native **P**od **A**utoscaler
CI/CD	**C**ontinuous **I**ntegration **C**ontinuous **D**eployment
AMQP	**A**dvanced **M**essage **Q**ueueing **P**rotocol

Στην οικογένεια μου...

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή στην Υπολογιστική Ακμής και στο Μοντέλο Χωρίς Διακομιστές

Κατά τα μέσα της δεκαετίας του 2000, σημειώθηκαν αρκετές εξελίξεις που έθεσαν τις βάσεις για την επακόλουθη εμφάνιση της τεχνολογίας υπολογιστών χωρίς διακομιστή (serverless computing).

Η πρώτη βασική πρόοδος ήταν αυτή της ανάπτυξης του υπολογιστικού νέφους. Το υπολογιστικό νέφος επέτρεψε την παροχή υπολογιστικής ισχύος κατά παραγγελία και σε πολύ μεγαλύτερη κλίμακα, απαλλάσσοντας τους προγραμματιστές και τις εταιρείες από την ανάγκη για τοπικό εξοπλισμο (hardware).

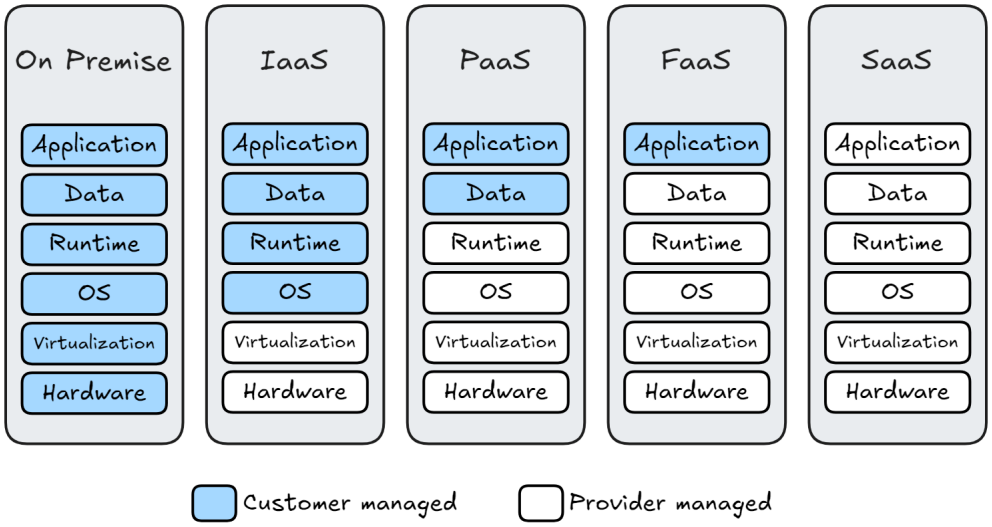
Η επόμενη, αξιοσημείωτη, βασική πρόοδος ήταν η ανάπτυξη του υπολογισμού βάσει συμβάντων (**events**), η οποία επέτρεψε στους προγραμματιστές να εκτελούν κώδικα σε απόκριση σε συμβάντα που δρουν σαν εναύσματα (**triggers**). Αυτή η εξέλιξη άνοιξε το δρόμο στους προγραμματιστές για τη δημιουργία πιο ευέλικτων και επεκτάσιμων εφαρμογών που μπορούσαν να προσαρμόζονται αυτόματα για να ανταποκρίνονται στις μεταβαλλόμενες απαιτήσεις.

Η υπολογιστική χωρίς διακομιστή (**serverless computing**) αναδύθηκε ως μοντέλο βασισμένο στο υπολογιστικό νέφος (**cloud computing**), το οποίο προέκυψε ως φυσική συνέχεια της εξέλιξης των τεχνολογιών νέφους και της ανόδου του μοντέλου υπολογισμού με γνώμονα τα συμβάντα (**event-driven computing model**.

1.1.1 Cloud Computing: το θεμέλιο της εξέλιξης

Το υπολογιστικό νέφος προέκυψε από την αξιοποίηση των αρχών της εικονικοποίησης (**virtualization)**. Η τεχνολογία της εικονικοποίησης επέτρεψε σε πολλαπλές απομονωμένες εικονικές μηχανές (**Virtual Machines** - VMs) να λειτουργούν με κοινή χρήση του ίδιου φυσικού υλικού, επιτρέποντας έτσι την αποδοτικότερη χρήση των πόρων του διακομιστή. Το υπολογιστικό νέφος επέκτεινε τα οφέλη της εικονικοποίησης με την αφαίρεση όλου του υποκείμενου υλικού και την παροχή πρόσβασης σε υπολογιστικούς πόρους σε ένα μοντέλο προσανατολισμένο στις υπηρεσίες. Τώρα οι διακομιστές, η αποθήκευση και η δικτύωση παρέχονται ως βοηθητικό πρόγραμμα. Μέχρι να κάνει την εμφάνισή του το serverless computing, οι κύριες κατηγορίες υπηρεσιών cloud ήταν: **Infrastructure as a Service (ή IaaS), Platform as a Service (PaaS) και Software as a Service (SaaS)**.

Με το IaaS οι εικονικοποιημένοι υπολογιστικοί πόροι προσφέρονται ως υπηρεσία, αυτό σημαίνει ότι οι χρήστες μπορούν να νοικιάζουν VM και αποθηκευτικό χώρο κατά παραγγελία και να πληρώνουν για ό,τι χρησιμοποιούν. Το PaaS βασίζεται στο IaaS παρέχοντας, όχι μόνο τις VMs, αλλά και τα λειτουργικά

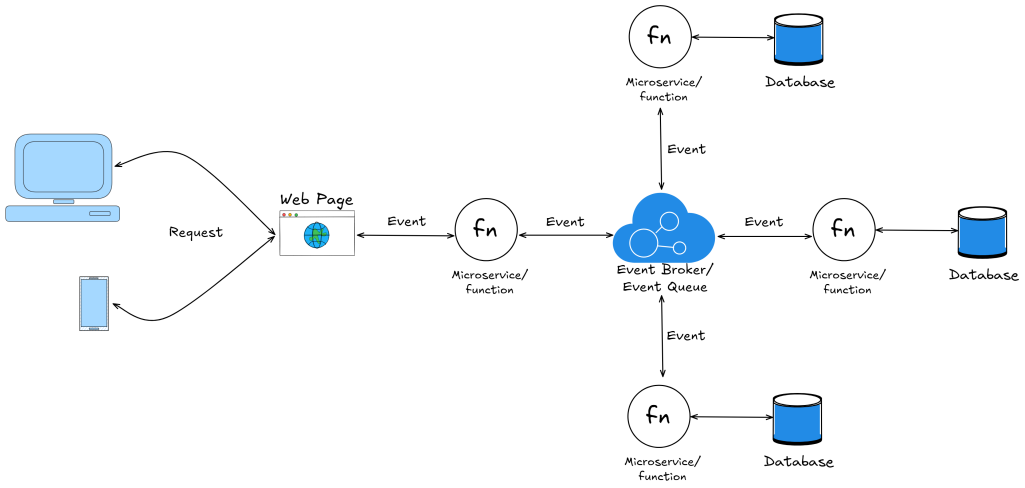


Σχημα 1.1: Η εξέλιξη των υπηρεσιών Cloud

συστήματα, τα πλαίσια ανάπτυξης και τις βάσεις δεδομένων που χρειάζεται κάποιος για να δημιουργήσει και να αναπτύξει τις εφαρμογές του. Με αυτόν τον τρόπο το επίπεδο υποδομής αφηρείται ακόμη περισσότερο, αλλά οι προγραμματιστές εξακολουθούν να είναι υπεύθυνοι για ορισμένα καθήκοντα διαχείρισης και διαμόρφωσης. Τέλος, το SaaS αποκρύπτει πλήρως την υποκείμενη υποδομή, καθώς παρέχει ολόκληρες εφαρμογές μέσω του νέφους χωρίς να χρειάζεται τοπική εγκατάσταση ή συντήρηση. Οι εφαρμογές διαχειρίζονται πλήρως από τον πάροχο υπηρεσιών και οι περιπτώσεις χρήσης και ο έλεγχος του χρήστη επί του λογισμικού είναι περιορισμένες.

Καθώς εξελίσσονταν οι υπηρεσίες νέφους, μια άλλη έννοια άρχισε να κερδίζει επίσης την προβολή, η έννοια της αρχιτεκτονικής καθοδηγούμενης από συμβάντα (**Event-Driven Architecture- EDA**). Η EDA είναι ένα παράδειγμα σχεδιασμού όπου οι εφαρμογές σχεδιάζονται έτσι ώστε να ανταποκρίνονται σε γεγονότα, όπως αλλαγές σε δεδομένα ή καταστάσεις του συστήματος, σε πραγματικό χρόνο. Αυτή η προσέγγιση επιτρέπει την αυτόματη κλιμάκωση με βάση τον όγκο των συμβάντων, η οποία από μόνη της επιτρέπει συστήματα υψηλής αποσύνδεσης. Με άλλα λόγια, η αρχιτεκτονική με γνώμονα τα συμβάντα έγινε γρήγορα ένας βασικός παράγοντας για τη δημιουργία εφαρμογών που μπορούν να αντιδρούν δυναμικά στη ζήτηση χωρίς την ανάγκη συνεχούς χειροκίνητης παρέμβασης.

Είναι σαφές ότι καθεμία από τις προαναφερθείσες βασικές υπηρεσίες νέφους αφηρεί περαιτέρω τη διαχείριση της υποδομής, καθώς το βάρος της διαχείρισης των πόρων και της διαμόρφωσης μετατοπίζεται προοδευτικά από τον χρήστη στον πάροχο. Παράλληλα, η άνοδος της EDA επέτρεψε στις εφαρμογές να ανταποκρίνονται δυναμικά σε γεγονότα πραγματικού χρόνου. Το επόμενο λογικό βήμα σε αυτή την τάση προς την αφαίρεση, την ευκολία χρήσης και την αυτοματοποίηση ήταν η εμφάνιση του **Serverless Computing**.

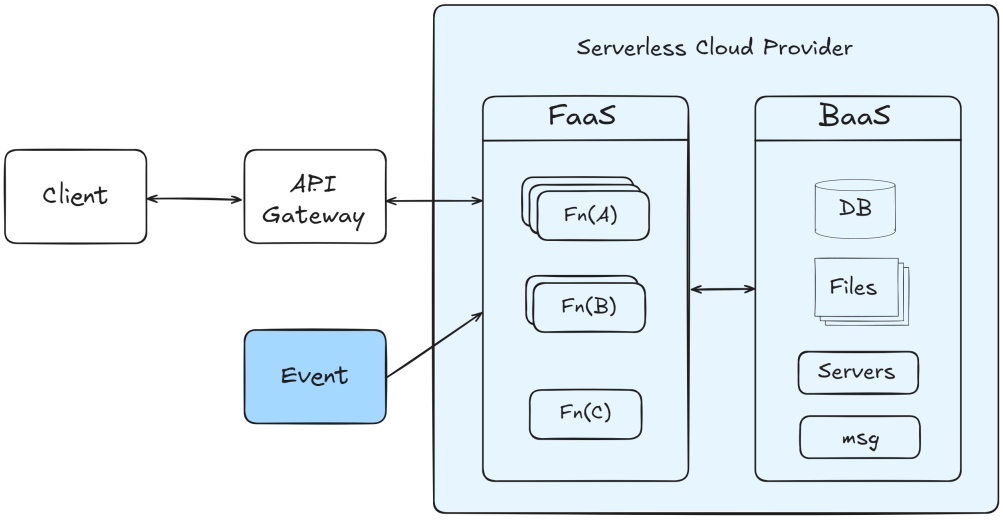


Σχημα 1.2: Παράδειγμα εφαρμογής καθοδηγούμενης από συμβάντα

1.1.2 Serverless Computing: Το επόμενο βήμα στην απλοποίηση των υποδομών cloud

Το μοντέλο υπολογιστικού νέφους χωρίς διακομιστή (Serverless Cloud Computing) αναπτύχθηκε για να επιλύσει τις πολυπλοκότητες στη διαχείριση των παραδοσιακών υπηρεσιών νέφους, δηλαδή τη διασφάλιση της διαθεσιμότητας των υπηρεσιών, την εξισορρόπηση του φορτίου μεταξύ των πόρων, τη διαχείριση της αυτόματης κλιμάκωσης, τη διατήρηση της ασφάλειας και την παρακολούθηση του συνολικού συστήματος.

Τα βασικά στοιχεία που παρέχονται από το Serverless είναι **Backend as Service (ή BaaS)** και **Function as a Service (FaaS)**. Οι όροι Serverless και FaaS συγχέονται συχνά λανθασμένα , αλλά στην πραγματικότητα το FaaS είναι μόνο ένα υποσύνολο του Serverless. Το FaaS αποτελεί όμως τον πυρήνα των serverless εφαρμογών, επιτρέποντας στους χρήστες να επικεντρωθούν στη λογική των stateless και event-driven λειτουργιών τους. Το BaaS μπορεί να εξυπηρετεί το FaaS για να κάνει την ανάπτυξη και τη συντήρηση του backend ευκολότερη για τους προγραμματιστές, με υπηρεσίες όπως η αποθήκευση στο cloud και οι ειδοποιήσεις.



Σχημα 1.3: Η Serverless Αρχιτεκτονική

Ένα από τα κύρια πλεονεκτήματα του serverless computing είναι ότι επιτρέπει στους προγραμματιστές να ανεβάζουν απλώς τον κώδικά τους (γραμμένο σε οποιαδήποτε γλώσσα προγραμματισμού) στην πλατφόρμα χωρίς να χρειάζεται να ανησυχούν για πολύπλοκες ρυθμίσεις περιβάλλοντος. Έτσι, η διαδικασία DevOps απλοποιείται και οι προγραμματιστές μπορούν να επικεντρωθούν στη συγγραφή κώδικα και όχι στη διαχείριση της υποκείμενης υποδομής.

Ένα άλλο βασικό χαρακτηριστικό, που κάνει το serverless να ξεχωρίζει, είναι η αυτόματη κλιμάκωση (**autoscaling**). Οι λειτουργίες που αναπτύσσονται σε πλατφόρμες serverless μπορούν να κλιμακωθούν αυτόματα είτε οριζόντια είτε κάθετα, όπως κρίνεται απαραίτητο. Όταν μια λειτουργία κλιμακώνεται κάθετα, παρέχονται περισσότεροι πόροι (μνήμη, cpu κ.λπ.) για την εκτέλεση της λειτουργίας. Όταν μια συνάρτηση κλιμακώνεται οριζόντια, αναπτύσσονται αυτόματα περισσότερες περιπτώσεις της ίδιας συνάρτησης για να μοιραστεί το φορτίο πιο αποτελεσματικά.

Το Serverless παρέχει 100% αποτελεσματικότητα στη χρήση των πόρων, καθώς οι πόροι παρέχονται μόνο κατά την άφιξη συμβάντων και απελευθερώνονται μετά την εκτέλεση (εάν δεν έχουν φθάσει άλλα συμβάντα). Εκτός από την εξοικονόμηση πόρων αυτό το μοντέλο εγγυάται επίσης ότι πληρώνετε μόνο για τους πόρους που χρησιμοποιείτε και ο χρήστης δεν χρειάζεται να νοικιάζει και να πληρώνει για πόρους που μένουν αδρανείς για μεγάλα χρονικά διαστήματα. Αξίζει να σημειωθεί όμως ότι αυτό μπορεί να σημαίνει ότι έχουμε υψηλότερο κόστος για εφαρμογές που εκτελούνται για μεγάλο χρονικό διάστημα.

Η δυνατότητα αυτόματης κλιμάκωσης προς το μηδέν (**scale-to-zero**) είναι ένα από τα κύρια πλεονεκτήματα του serverless μοντέλου, αλλά αποτελεί επίσης αιτία για ένα σημαντικό μειονέκτημα. Όταν μια συνάρτηση κλιμακώνεται σε μηδενικά instances, μπορεί να προκαλέσει το φαινόμενο της «ψυχρής εκκίνησης» (**cold start**), όπου η συνάρτηση χρειάζεται περισσότερο χρόνο για να εκκινήσει όταν φτάσει ένα νέο συμβάν, οδηγώντας έτσι σε αυξημένη καθυστέρηση στην επεξεργασία. Αυτό μπορεί να επηρεάσει αρνητικά την αποδοτικότητα και την απόδοση μιας εφαρμογής, ιδιαίτερα σε περιβάλλοντα με απαιτήσεις πραγματικού χρόνου.

Είναι επίσης σημαντικό να ληφθεί υπόψη ο κίνδυνος «κλειδώματος» σε συγκεκριμένο προμηθευτή (**vendor lock-in**). Οι μεγάλοι πάροχοι cloud, όπως οι AWS (με το AWS Lambda), Microsoft (με το Azure Functions), και Google (με το Google Cloud Functions), προσφέρουν εξατομικευμένες υπηρεσίες serverless με μοναδικά χαρακτηριστικά. Ωστόσο, αυτό μπορεί να αυξήσει τη δυσκολία στις διαδικασίες δοκιμών και αποσφαλμάτωσης, καθώς οι προγραμματιστές έχουν περιορισμένο έλεγχο και ορατότητα στις λειτουργίες του back-end συστήματος.

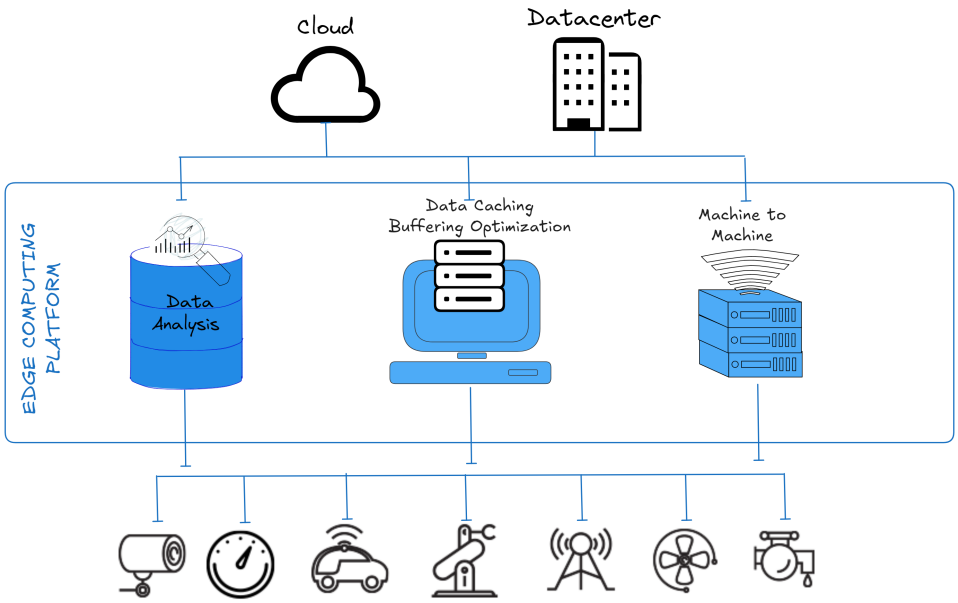
Το serverless computing είναι θεμελιωδώς συνδεδεμένο με το υπολογιστικό νέφος. Οι πάροχοι cloud συνήθως προσφέρουν τις υπηρεσίες serverless μέσω συγκεντρωτικά κέντρα δεδομένων (centralized data centres). Ωστόσο, αυτό δημιουργεί προκλήσεις για εφαρμογές που απαιτούν επεξεργασία σε πραγματικό χρόνο ή αποκρίσεις με χαμηλή καθυστέρηση, καθώς τα δεδομένα πρέπει να μεταφέρονται προς και από αυτά τα κέντρα. Ιδιαίτερα σε περιπτώσεις εφαρμογών IoT και streaming, η μαζική μεταφορά δεδομένων μπορεί να επιβαρύνει το εύρος ζώνης του δικτύου.

Αυτά τα σενάρια, στα οποία η χαμηλή καθυστέρηση (**low latency**)και η επεξεργασία σε πραγματικό χρόνο είναι κρίσιμης σημασίας, οδήγησαν στην ανάπτυξη της υπολογιστικής τεχνολογίας άκρων (**edge computing**), η οποία φέρνει τα πλεονεκτήματα του υπολογιστικού νέφους πιο κοντά στην πηγή των δεδομένων. Είναι συνεπώς προφανές ότι η υπολογιστική τεχνολογία άκρων μπορεί

να λειτουργήσει συμπληρωματικά με το μοντέλο serverless, ιδίως για εφαρμογές που δεν μπορούν να ανεχθούν τις καθυστερήσεις που προκύπτουν από την κεντρική επεξεργασία δεδομένων. Τα γεγονότα που λαμβάνουν χώρα στην άκρη του δικτύου μπορούν να ενεργοποιήσουν serverless λειτουργίες με τρόπο παρόμοιο με εκείνον των γεγονότων στο cloud (π.χ. ένας αισθητήρας IoT μπορεί να ενεργοποιήσει μια λειτουργία που αποστέλλει ειδοποίηση, χωρίς την ανάγκη επικοινωνίας με κεντρικά δεδομένα του νέφους).

1.1.3 Edge Computing: Γεφύρωση των αναγκών serverless και επεξεργασίας σε πραγματικό χρόνο

Ενώ η κεντρικοποιημένη υπολογιστική είναι ιδανική για διεργασίες υψηλής υπολογιστικής έντασης, η υπολογιστική άκρης καλύπτει την απαίτηση για επεξεργασία σε πραγματικό χρόνο και χαμηλή καθυστέρηση, που είναι συνήθως αναγκαία σε αναδυόμενα πεδία όπως το IoT, η μηχανική μάθηση και το AR/VR. Η έννοια της υπολογιστικής άκρης βασίζεται στην τοποθέτηση των υπολογιστικών υπηρεσιών πιο κοντά είτε στις πηγές δεδομένων είτε στους καταναλωτές των υπηρεσιών. Πρόκειται για ένα κατανεμημένο μοντέλο που επιτρέπει την τοπική επεξεργασία δεδομένων σε έξυπνες συσκευές, εξαλείφοντας την ανάγκη αποστολής μεγάλων ποσοτήτων δεδομένων στο κεντρικό νέφος για επεξεργασία, με αποτέλεσμα την αποφυγή της επιστροφής των δεδομένων στην αρχική πηγή (με σημαντικό κόστος σε χρόνο, χρήμα και αποδοτικότητα).



Σχημα 1.4: Εποπτεία μιας πλατφόρμας edge computing

Με το edge computing, η έμφαση μετατοπίζεται στο λογικό άκρο της υποδομής. Εκτός από τη μειωμένη καθυστέρηση, που είναι ένα από τα κύρια οφέλη, η ενεργοποίηση του υπολογισμού στην άκρη επιτρέπει επίσης την καλύτερη ασφάλεια των δεδομένων, καθώς οι ευαίσθητες πληροφορίες στις συσκευές των χρηστών δεν χρειάζεται πλέον να ταξιδεύουν σε κεντρικές τοποθεσίες. Αυτό μειώνει τους κινδύνους επιθέσεων στον κυβερνοχώρο. Επιπλέον, είναι πιο αποδοτικό από πλευράς κόστους, καθώς οι εταιρείες μπορούν να χρησιμοποιούν

μικρότερο εύρος ζώνης και, ως εκ τούτου, να πληρώνουν λιγότερα για υπολογιστικό νέφος. Ταυτόχρονα, αυτό συμβάλλει στην ανακούφιση του δικτύου από περιττή κίνηση, αποτρέποντας τη συμφόρηση του δικτύου.

Ωστόσο, υπάρχουν ορισμένες προκλήσεις που συνοδεύουν το edge computing, οι οποίες θα πρέπει να ληφθούν υπόψη. Αρχικά, στην άκρη, οι διαθέσιμοι υπολογιστικοί πόροι είναι σημαντικά περιορισμένοι σε αντίθεση με τους πόρους στα κέντρα δεδομένων του νέφους. Είναι επίσης πιο δύσκολο να διασφαλιστεί η συνοχή των δεδομένων μεταξύ των κατανεμημένων κόμβων της άκρης, καθώς η διαχείριση κατανεμημένων συσκευών αυξάνει γενικά την επιχειρησιακή πολυπλοκότητα.

Η μετάβαση από την κεντρική στην κατανεμημένη διαχείριση πόρων έρχεται ως φυσική εξέλιξη στις τεχνολογίες του υπολογιστικού νέφους. Με το edge computing, τα οφέλη του serverless μπορούν να ενισχυθούν με τη δυνατότητα τοπικής διαχείρισης πόρων, κάτι που μπορεί να αποδειχθεί κρίσιμο για την απόδοση εφαρμογών με απαιτήσεις πραγματικού χρόνου. Αυτή η συνέργεια μεταξύ του edge computing και του serverless όχι μόνο βελτιώνει την απόκριση, αλλά επηρεάζει επίσης τις στρατηγικές εξισορρόπησης φορτίου και τους μηχανισμούς αυτόματης κλιμάκωσης που εφαρμόζονται. Καθώς αυτές οι αρχιτεκτονικές γίνονται πιο αποκεντρωμένες, η δυναμική διαχείριση των πόρων τόσο στις κεντρικές όσο και στις ακραίες τοποθεσίες καθίσταται απαραίτητη για τη βελτιστοποίηση της επίδοσης και της αποδοτικότητας του κόστους. Η αποτελεσματική εξισορρόπηση του φορτίου και η λήψη αποφάσεων αυτόματης κλιμάκωσης σε πραγματικό χρόνο αποτελούν πλέον κρίσιμα στοιχεία για τη διατήρηση της απρόσκοπτης λειτουργίας των σύγχρονων, κατανεμημένων εφαρμογών.

1.2 Σχετικά με τη σημασία της διαχείρισης των πόρων

Στο serverless computing η διαχείριση πόρων περιλαμβάνει την κατανομή των κατάλληλων πόρων για τις λειτουργίες και τον προγραμματισμό στις κατάλληλες υποδομές. Αυτό είναι κρίσιμο, ώστε να ικανοποιούνται τόσο οι απαιτήσεις ποιότητας υπηρεσιών (**Quality of Service - QoS**) των προγραμματιστών όσο και των παρόχων υπηρεσιών νέφους. Οι στόχοι QoS για τον πελάτη/καταναλωτή επικεντρώνονται κυρίως στο πόσο καλά αποδίδει η πλατφόρμα serverless. Περιλαμβάνουν την απόδοση της εφαρμογής, το κόστος εκτέλεσης και τις ανησυχίες για την ασφάλεια. Από την πλευρά των παρόχων, οι στόχοι QoS σχετίζονται με την βέλτιστη χρήση των υποκείμενων πόρων, την εξισορρόπηση φορτίου και την επίτευξη ενός συστήματος υψηλής απόδοσης. Είναι σημαντικό να σημειωθεί, ωστόσο, ότι η αυξημένη χρήση των πόρων σε ένα σημείο θυσιάζει τις εγγυήσεις QoS για τον καταναλωτή, οπότε οι στόχοι των δύο μερών μπορεί να είναι συχνά αντικρουόμενοι.

Είναι σαφές ότι η διαδικασία διαχείρισης των πόρων είναι ζωτικής σημασίας τόσο για τους προγραμματιστές όσο και για τους παρόχους, και η βελτίωσή της έχει αποτελέσει -και συνεχίζει να αποτελεί- αντικείμενο έντονου ερευνητικού ενδιαφέροντος. Το επίκεντρο των υφιστάμενων λύσεων είναι η βελτιστοποίηση της κατανομής πόρων και του χρονοπρογραμματισμού, καθώς αυτό μπορεί να βελτιώσει την απόδοση, την αποδοτικότητα κόστους, καθώς και την ασφάλεια. Η δυναμική προσαρμογή των πόρων και οι στρατηγικές με επίγνωση των αναγκών των εφαρμογών έχουν ενισχύσει την ευελιξία και την ανταπόκριση των συστημάτων. Ωστόσο, συχνά εστιάζουν σε συγκεκριμένους πόρους (π.χ. CPU), αφήνοντας περιθώριο για ευρύτερες προσεγγίσεις στη διαχείριση πόρων. Όταν

πρόκειται για την αντιμετώπιση "bursty" (εκρηκτικής φύσης) και stateless εφαρμογών, καθώς και άλλων ειδικών απαιτήσεων των serverless φόρτων εργασίας, η πιο υποσχόμενη προσέγγιση είναι ο καινοτόμος σχεδιασμός αρχιτεκτονικής.

Τελικά, αυτές οι λύσεις αναδεικνύουν τη συνεχή εξέλιξη των στρατηγικών διαχείρισης πόρων, με στόχο την ικανοποίηση των αυξανόμενων απαιτήσεων των σύγχρονων, κατανεμημένων εφαρμογών. Αυτό θα διασφαλίσει τη βελτιστοποίηση των επιδόσεων και της αποδοτικότητας κόστους σε όλο και πιο σύνθετες αρχιτεκτονικές νέφους. Καθώς το **serverless computing** ενσωματώνεται περαιτέρω με την **υπολογιστική άκρης (edge computing)**, η ανάγκη για τοπική επεξεργασία γίνεται όλο και πιο επιτακτική. Αυτή η μετάβαση απαιτεί την επαναξιολόγηση των παραδοσιακών μηχανισμών εξισορρόπησης φορτίου και αυτόματης κλιμάκωσης, καθώς υπάρχει αυξανόμενη ανάγκη για αποτελεσματική διαχείριση των πόρων τόσο σε κεντρικά όσο και σε κατανεμημένα περιβάλλοντα.

1.3 Αντικείμενο Διπλωματικής

Καθώς η υπολογιστική χωρίς διακομιστή συνεχίζει να αυξάνει τη δημοτικότητά της, τα οφέλη της δυναμικής κλιμάκωσης και της αποδοτικότητας κόστους συχνά δοκιμάζονται από απρόβλεπτες αυξομειώσεις στους φόρτους εργασίας. Οι υπάρχοντες μηχανισμοί αυτόματης κλιμάκωσης σε αρχιτεκτονικές serverless, όπως η πλατφόρμα Kubernetes του Knative, αντιμετωπίζουν δυσκολίες στη διαχείριση της εκρηκτικής κίνησης και των δυναμικών, ετερογενών φορτίων εργασίας. Αυτές οι προκλήσεις προκύπτουν από τα εξής:

Αναποτελεσματική εξισορρόπηση φορτίου: Οι παραδοσιακές στρατηγικές εξισορρόπησης φορτίου σε περιβάλλοντα serverless είναι συχνά στατικές ή αντιδρούν μόνο σε απλές μετρήσεις, όπως η χρήση CPU ή μνήμης. Αυτές οι μέθοδοι αποτυγχάνουν να λάβουν υπόψη πιο σύνθετα σενάρια, όπου οι διάφορες υπηρεσίες έχουν διαφορετικές δυνατότητες επεξεργασίας και ανταποκρίνονται διαφορετικά στις μεταβαλλόμενες απαιτήσεις.

Προβλήματα κλιμάκωσης με ξαφνικές αιχμές: Παρά το γεγονός ότι οι πλατφόρμες serverless παρέχουν δυνατότητες αυτόματης κλιμάκωσης, συχνά αποτυγχάνουν να ανταποκριθούν επαρκώς σε ξαφνικές εκρήξεις κίνησης, γεγονός που οδηγεί σε καθυστερημένες αποκρίσεις και μη αποδοτική χρήση των πόρων. Αυτό το ζήτημα, γνωστό ως πρόβλημα «ψυχρής εκκίνησης», επηρεάζει την ικανότητα του συστήματος να διαχειρίζεται εφαρμογές ευαίσθητες στον χρόνο ή σε πραγματικό χρόνο.

Υπερ/υποαπασχόληση πόρων: Οι τρέχοντες αλγόριθμοι αυτόματης κλιμάκωσης είτε δεν λαμβάνουν υπόψη τις διαφορές στην επεξεργαστική ισχύ των επιμέρους υπηρεσιών είτε δεν προσαρμόζονται σε πραγματικό χρόνο στις διακυμάνσεις της ζήτησης. Ως αποτέλεσμα, οι πόροι είτε υποαπασχολούνται σε περιόδους χαμηλής ζήτησης είτε υπερφορτώνονται κατά τη διάρκεια αιχμών, οδηγώντας σε αυξημένη καθυστέρηση και μειωμένη απόδοση του συστήματος.

1.3.1 Προτεινόμενη Λύση

Για την αντιμετώπιση αυτών των προκλήσεων, προτείνουμε ένα σύστημα δυναμικής εξισορρόπησης φορτίου και ελέγχου αποδοχής, βασισμένο στο Knative και το Kubernetes. Το σύστημα είναι σχεδιασμένο για τη βελτιστοποίηση της διαχείρισης εκρηκτικών και δυναμικών φόρτων εργασίας, χρησιμοποιώντας τον

αλγόριθμο Additive Increase and Multiplicative Decrease (AIMD) για τη δρομολόγηση της κίνησης και τον έλεγχο εισδοχής.

Τα βασικά στοιχεία του συστήματος είναι τα εξής:

Προσαρμοσμένος εξισορροπιστής φορτίου με δρομολόγηση βάσει AIMD: Ο πυρήνας της λύσης είναι ένας προσαρμοσμένος εξισορροπιστής φορτίου που δρομολογεί τα εισερχόμενα αιτήματα με βάση δυναμικά ενημερωμένα ποσοστά αποδοχής. Τα ποσοστά αυτά ελέγχονται μέσω του αλγορίθμου AIMD, ο οποίος προσαρμόζει τον αριθμό των αιτήσεων που μπορεί να χειριστεί κάθε υπηρεσία, με βάση μετρήσεις επιδόσεων σε πραγματικό χρόνο. Με τη συνεχή εξισορρόπηση του φορτίου, το σύστημα αποτρέπει την υπερφόρτωση των υπηρεσιών και εξασφαλίζει βέλτιστη χρήση των πόρων.

Κεντρική ουρά με ελεγκτές αποδοχής: Μια κεντρική ουρά συλλέγει τα εισερχόμενα αιτήματα, και κάθε υπηρεσία έχει τον δικό της ελεγκτή αποδοχής. Οι ελεγκτές καθορίζουν πόσα αιτήματα μπορεί να επεξεργαστεί κάθε υπηρεσία ανά πάσα στιγμή, με βάση τον τρέχοντα φόρτο εργασίας και την απόδοσή της. Ο έλεγχος εισδοχής μέσω AIMD διασφαλίζει ότι οι υπηρεσίες αυξάνουν σταδιακά τον ρυθμό επεξεργασίας τους όταν υποαπασχολούνται και τον μειώνουν ταχέως όταν πλησιάζουν στη μέγιστη χωρητικότητά τους.

Δυναμική αυτόματη κλιμάκωση μέσω του Knative: Το σύστημα αξιοποιεί τις δυνατότητες αυτόματης κλιμάκωσης του Knative, επιτρέποντας στις υπηρεσίες να κλιμακώνονται από μηδενικά επίπεδα σε περιπτώσεις εκρήξεων κίνησης και να μειώνονται όταν η ζήτηση υποχωρεί. Αυτή η αυτόματη κλιμάκωση είναι στενά ενσωματωμένη με τον προσαρμοσμένο εξισορροπιστή φορτίου, ώστε οι αποφάσεις κλιμάκωσης να βασίζονται τόσο στα τρέχοντα φορτία κίνησης όσο και στα ποσοστά αποδοχής συγκεκριμένων υπηρεσιών.

Ενσωμάτωση Redis και RabbitMQ για μετρήσεις και παρακολούθηση: Οι μετρήσεις επιδόσεων σε πραγματικό χρόνο, όπως τα μήκη ουρών αναμονής και η απόδοση της υπηρεσίας, συλλέγονται και αποθηκεύονται στο Redis. Αυτές οι μετρήσεις τροφοδοτούν τον εξισορροπιστή φορτίου και τους ελεγκτές αποδοχής, επιτρέποντας στο σύστημα να λαμβάνει αποφάσεις σε πραγματικό χρόνο σχετικά με την κατανομή της κίνησης και την κλιμάκωση.

Ανθεκτικότητα σε ξαφνικές αιχμές με λειτουργία πανικού: Σε περιπτώσεις ξαφνικών αιχμών κίνησης, το σύστημα ενεργοποιεί τη λειτουργία πανικού, κατά την οποία οι υπηρεσίες κλιμακώνονται ταχέως χωρίς τους συνήθεις ελέγχους σταθεροποίησης. Αυτό διασφαλίζει ότι το σύστημα μπορεί να διαχειριστεί απρόβλεπτες εκρήξεις κίνησης χωρίς να επηρεάζεται η απόδοση ή να προκαλούνται καθυστερήσεις.

Με την ενσωμάτωση ελέγχου εισδοχής μέσω AIMD, μετρήσεων πραγματικού χρόνου και δυναμικής αυτόματης κλιμάκωσης, το σύστημά μας διαχειρίζεται αποτελεσματικά τόσο σταθερή όσο και εκρηκτική κίνηση, ελαχιστοποιώντας την καθυστέρηση από ψυχρές εκκινήσεις και εξασφαλίζοντας βέλτιστη αξιοποίηση των πόρων. Αυτό το καθιστά ιδιαίτερα κατάλληλο για εφαρμογές πραγματικού χρόνου και υψηλής ευαισθησίας στην καθυστέρηση, που εκτελούνται σε περιβάλλον serverless.

1.3.2 Δομή της διπλωματικής

Η παρούσα εργασία είναι οργανωμένη σε ξεχωριστά κεφάλαια, το καθένα από τα οποία βασίζεται στο προηγούμενο και αποτελεί μια ολοκληρωμένη διερεύνηση της δυναμικής εξισορρόπησης φορτίου σε περιβάλλοντα Knative.

Κεφάλαιο 2: Σύνοψη Συναφών Εργασιών και Υπάρχοντων Πλαισίων Υλοποίησης Το επόμενο κεφάλαιο περιλαμβάνει μια ανασκόπηση των υφιστάμενων προσεγγίσεων στον δυναμικό χρονοπρογραμματισμό και την εξισορρόπηση φορτίου σε πλαίσια serverless. Καλύπτει μια σειρά τεχνικών βελτιστοποίησης, όπως η βελτιστοποίηση σμήνους σωματιδίων (Particle Swarm Optimization - PSO), η βελτιστοποίηση αποικίας μυρμηγκιών (Ant Colony Optimization - ACO) και οι γενετικοί αλγόριθμοι, δίνοντας έμφαση στη συνάφεια τους με τον προγραμματισμό φόρτου εργασίας. Ιδιαίτερη προσοχή δίνεται στον αλγόριθμο AIMD (Additive Increase Multiplicative Decrease), καθώς αποτελεί τον πυρήνα του εξισορροπιστή φορτίου που σχεδιάστηκε στην παρούσα διατριβή.

Κεφάλαιο 3: Υπόβαθρο: Knative και Σχετικά Εργαλεία Σε αυτό το κεφάλαιο, το Knative βρίσκεται στο επίκεντρο του ενδιαφέροντος. Εμβαθύνει στην αρχιτεκτονική του, τη σημασία του σε περιβάλλοντα χωρίς διακομιστές και τα βασικά χαρακτηριστικά του, όπως το Knative Serving και το Eventing. Η συζήτηση επεκτείνεται επίσης σε άλλα κρίσιμα εργαλεία που χρησιμοποιούνται, όπως το RabbitMQ για το χειρισμό της ανταλλαγής μηνυμάτων και το Redis για τη διαχείριση της κατάστασης, παρουσιάζοντας τον τρόπο με τον οποίο ενσωματώνονται στο οικοσύστημα του Knative.

Κεφάλαιο 4: Σχεδιίαση και Υλοποίηση του Συστήματος Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική του συστήματος, ξεκινώντας από μια υψηλού επιπέδου επισκόπηση και εστιάζοντας σταδιακά στις λεπτομέρειες κάθε συστατικού. Συζητείται διεξοδικά ο σχεδιασμός του προσαρμοσμένου εξισορροπιστή φορτίου, των ελεγκτών αποδοχής και των μηχανισμών που χρησιμοποιούνται για τη συλλογή και την παρακολούθηση των μετρήσεων. Κεντρικό ρόλο παίζει ο αλγόριθμος AIMD, ο οποίος χρησιμοποιείται για τη δυναμική προσαρμογή της κατανομής της κυκλοφορίας.

Κεφάλαιο 5: Δοκιμές Απόδοσης και Ανάλυση Συστήματος Υπό Μεταβαλλόμενο Φορτίο Η απόδοση αποτελεί τον πυρήνα κάθε συστήματος εξισορρόπησης φορτίου και το κεφάλαιο αυτό παρέχει μια σε βάθος ανάλυση του τρόπου με τον οποίο το σύστημα αποδίδει υπό διάφορες συνθήκες. Δοκιμάζονται διαφορετικά μοτίβα κίνησης, συμπεριλαμβανομένων σταθερών και εκρηκτικών φορτίων, και τα αποτελέσματα αξιολογούνται χρησιμοποιώντας μετρήσεις όπως η χρήση CPU, η κατανάλωση μνήμης και η καθυστέρηση αιτήσεων. Γίνονται συγκρίσεις μεταξύ του Knative Pod Autoscaler (KPA) και του Horizontal Pod Autoscaler (HPA), αποκαλύπτοντας πώς κλιμακώνεται το σύστημα υπό διαφορετικούς φόρτους εργασίας.

Κεφάλαιο 6: Συμπεράσματα και μελλοντικές κατευθύνσεις Η παρούσα εργασία ολοκληρώνεται με μια σύνοψη των βασικών συμπερασμάτων, ανανακλώντας την απόδοση του προτεινόμενου συστήματος και τη συμβολή του στο πεδίο. Τέλος, συζητούνται πιθανές βελτιώσεις και μελλοντικές ερευνητικές ευκαιρίες, όπως η διερεύνηση τεχνικών μηχανικής μάθησης για πιο έξυπνες αποφάσεις κλιμάκωσης.

Κεφάλαιο 2

Σύνοψη Συναφών Εργασιών και Υπάρχοντων Πλαισίων Υλοποίησης

Σκοπός του παρόντος κεφαλαίου είναι να προσφέρει μια ολοκληρωμένη επισκόπηση της υφιστάμενης βιβλιογραφίας και των πλαισίων υλοποίησης (**frameworks**) που έχουν συμβάλει στη διαμόρφωση της τρέχουσας κατάστασης στο πεδίο του serverless computing. Για να βοηθηθεί η πλαισίωση της προτεινόμενης λύσης, αναφέρονται διάφορες ερευνητικές κατευθύνσεις, ώστε να εντοπιστούν οι βασικές προκλήσεις και οι λύσεις που έχουν ήδη προταθεί. Το κεφάλαιο συνεχίζει με τη διερεύνηση του αλγορίθμου Additive Increase and Multiplicative Decrease (AIMD), μαζί με την προσαρμογή του για την εξισορρόπηση φορτίου και την κατανομή πόρων σε κατανεμημένα συστήματα. Η προτεινόμενη λύση της παρούσας εργασίας βασίζεται στον AIMD, οπότε η ανάλυσή του είναι καθοριστικής σημασίας προκειμένου να αναδειχθεί η χρήση του στη διαχείριση της δυναμικής φύσης των περιβαλλόντων χωρίς διακομιστές, όπου ο φόρτος εργασίας μπορεί να αυξομειώνεται.

2.1 Οι προκλήσεις του χρονοπρογραμματισμού λειτουργιών χωρίς διακομιστή

Οι υφιστάμενοι μηχανισμοί χρονοπρογραμματισμού σε serverless περιβάλλοντα παρουσιάζουν αρκετούς περιορισμούς, καθώς συχνά αδυνατούν να εξυπηρετήσουν τα μοναδικά χαρακτηριστικά και τις απαιτήσεις των εφαρμογών χωρίς διακομιστές. Οι εφαρμογές χωρίς διακομιστή συχνά παρουσιάζουν ξαφνικές αιχμές στη ζήτηση. Αυτό απαιτεί έναν μηχανισμό χρονοπρογραμματισμού που να μπορεί να προσαρμόζεται γρήγορα σε μεγάλο αριθμό κλήσεων συναρτήσεων που συμβαίνουν σε σύντομο χρονικό διάστημα. Το σύστημα πρέπει επίσης να είναι σε θέση να διαχειρίζεται αποτελεσματικά τους πόρους, ώστε να μπορεί να χειρίζεται τόσο τις βραχυπρόθεσμες όσο και τις μακροπρόθεσμες τρέχουσες εργασίες. Ένα από τα κύρια χαρακτηριστικά των εφαρμογών χωρίς διακομιστή είναι η απουσία κατάστασης (**statelessness**). Ωστόσο, τα παραδοσιακά πλαίσια χρονοπρογραμματισμού είναι συχνά σχεδιασμένα για να χειρίζονται μακροχρόνιες, stateful εφαρμογές που συχνά απαιτούν τους ίδιους πόρους σε βάθος χρόνου.

Προκειμένου οι serverless συναρτήσεις να παραμένουν stateless, ελαφριές και εύκολα διαχειρίσιμες, κάθε instance συνάρτησης σχεδιάζεται συνήθως να εκτελείται σε έναν μόνο πυρήνα CPU. Αυτό σημαίνει ότι ένα instance συνάρτησης που λειτουργεί σε έναν μόνο πυρήνα δεν μπορεί να αξιοποιήσει την παράλληλη εκτέλεση εντός του ίδιου instance για την επιτάχυνση της εκτέλεσής του.

Ακόμη και με τις serverless συναρτήσεις να εκτελούνται σε έναν μόνο πυρήνα, ο χρονοπρογραμματιστής (**scheduler**) πρέπει να διασφαλίζει την πλήρη αξιοποίηση όλων των διαθέσιμων πυρήνων του συστήματος. Εάν υπάρχει ανισορροπία στη χρήση των πόρων, ορισμένοι πυρήνες ενδέχεται να υπερφορτωθούν, ενώ άλλοι παραμένουν υποχρησιμοποιημένοι, οδηγώντας σε αυξημένη καθυστέρηση (latency) και μειωμένη απόδοση του συστήματος. Τα σημεία συμφόρησης μπορεί επίσης να προκύψουν σε περιπτώσεις υψηλών επιπέδων ταυτόχρονης εκτέλεσης (**concurrency**), όταν πολλές συναρτήσεις εκτελούνται ταυτόχρονα, εάν οι πυρήνες δεν κατανεμηθούν αποτελεσματικά.

2.2 Δυναμικές Τεχνικές Χρονοπρογραμματισμού σε Περιβάλλοντα Knative

Ο δυναμικός χρονοπρογραμματισμός είναι ζωτικής σημασίας για τη βέλτιστη κατανομή και χρήση πόρων καθώς και την εξισορρόπηση φορτίου σε serverless αρχιτεκτονικές, όπως το **Knative**. Αυτή η ενότητα εξετάζει ορισμένες βασικές τεχνικές χρονοπρογραμματισμού που μπορούν να προσαρμοστούν ώστε να ενισχύσουν την απόδοση της διαχείρισης αιτημάτων σε τέτοια περιβάλλοντα.

2.2.1 Χρονοπρογραμματισμός με Βάση το Particle Swarm Optimization (PSO)

Το Particle Swarm Optimization (PSO) έχει αναδειχθεί ως ένας ισχυρός ευρετικός αλγόριθμος για δυναμικό χρονοπρογραμματισμό εργασιών στο cloud computing. Οι Pandey et al. (2009) εισήγαγαν μια προσέγγιση βασισμένη στο PSO, η οποία μειώνει σημαντικά το κόστος εκτέλεσης και βελτιώνει την κατανομή του φόρτου εργασίας στους διαθέσιμους πόρους.

Ο όρος 'Particle Swarm Optimization' προέρχεται από την έμπνευση του αλγορίθμου, η οποία βασίζεται στη συλλογική συμπεριφορά των σμηνών πουλιών ή των κοπαδιών ψαριών, όπου κάθε 'σωματίδιο' αντιπροσωπεύει μια πιθανή λύση και κινείται στον χώρο λύσεων ακολουθώντας τα σωματίδια με την καλύτερη απόδοση. Μέσω αυτής της προσομοίωσης, ο αλγόριθμος **PSO** επιτρέπει την αποτελεσματική εξερεύνηση του χώρου λύσεων και τη σύγκλιση προς βέλτιστες στρατηγικές κατανομής πόρων.

Οι Feng et al. (2012) επέκτειναν αυτό το πεδίο εφαρμόζοντας μια πολυαντικειμενική προσέγγιση **PSO**, η οποία λαμβάνει υπόψη τόσο τον χρόνο εκτέλεσης των εργασιών όσο και την κράτηση των πόρων. Αυτή η διττή εστίαση είναι καθοριστικής σημασίας για τη διαχείριση του δυναμικού scaling των serverless εφαρμογών, επιτυγχάνοντας ισορροπία μεταξύ απόδοσης και αποδοτικότητας στη χρήση πόρων. Η ενσωμάτωση πολυ-αντικειμενικής βελτιστοποίησης στο PSO ενισχύει την εφαρμοσιμότητά του σε περιβάλλοντα όπου πρέπει να ικανοποιηθούν ταυτόχρονα πολλαπλά κριτήρια, όπως η ελαχιστοποίηση του κόστους και η μεγιστοποίηση της απόδοσης.

Οι Juan et al. (2012) συνέβαλαν περαιτέρω αναπτύσσοντας έναν βελτιωμένο αλγόριθμο PSO ειδικά για συστήματα αποθήκευσης στο cloud. Το έργο τους υπογραμμίζει την προσαρμοστικότητα του PSO σε διάφορα πλαίσια cloud, δείχνοντας την αποτελεσματικότητά του στη βελτιστοποίηση του χρονοπρογραμματισμού εργασιών υπό διαφορετικούς λειτουργικούς περιορισμούς. Οι βελτιώσεις στον αλγόριθμό τους αντιμετωπίζουν ορισμένες από τις αδυναμίες του παραδοσιακού PSO, όπως η ταχύτητα σύγκλισης και η δυνατότητα διαφυγής από

τοπικά άκρα, καθιστώντας το μια ισχυρή επιλογή για δυναμικά περιβάλλοντα όπως το **Knative**.

Επιπροσθέτως, η ευελιξία του αλγορίθμου PSO επιτρέπει την ενσωμάτωσή του με άλλες τεχνικές χρονοπρογραμματισμού, ενισχύοντας περαιτέρω την απόδοσή του σε cloud computing σενάρια. Αυτή η προσαρμοστικότητα είναι ζωτικής σημασίας για περιβάλλοντα με διακυμάνσεις φόρτου εργασίας, όπου απαιτούνται ταχείες προσαρμογές στις στρατηγικές κατανομής πόρων.

2.2.2 Χρονοπρογραμματισμός με Βάση το Ant Colony Optimization (ACO)

Το Ant Colony Optimization (ACO) είναι μια δυναμική τεχνική χρονοπρογραμματισμού που έχει παρουσιάσει σημαντικές δυνατότητες σε περιβάλλοντα cloud computing. Αυτός ο αλγόριθμος εμπνέεται από τη συμπεριφορά αναζήτησης τροφής των μυρμηγκιών και χρησιμοποιεί έναν μηχανισμό θετικής ανατροφοδότησης μέσω φερομονικών διαδρομών για την εύρεση βέλτιστων διαδρομών. Οι Khan και Sharma (2013) ανέδειξαν την αποτελεσματικότητα του **ACO** για την εξισορρόπηση φορτίου στο cloud computing, υπογραμμίζοντας τη συνάφειά του με τα περιβάλλοντα **Knative**, όπου η αποδοτική κατανομή αιτημάτων στις serverless λειτουργίες είναι απαραίτητη. Η έρευνά τους έδειξε πως το ACO μπορεί να προσαρμοστεί δυναμικά σε διακυμάνσεις φόρτου εργασίας, διασφαλίζοντας την αποτελεσματική κατανομή των πόρων σύμφωνα με τις ανάγκες των χρηστών.

Οι Tawfeek et al. (2013) επέκτειναν το ACO εισάγοντας έναν τροποποιημένο κανόνα ενημέρωσης φερομόνης, ο οποίος βελτιώνει την ικανότητα του αλγορίθμου να ανταποκρίνεται σε δυναμικές αλλαγές στον φόρτο εργασίας. Αυτή η βελτίωση είναι ιδιαίτερα επωφελής σε serverless αρχιτεκτονικές, όπου οι φόρτοι εργασίας παρουσιάζουν σημαντικές και απρόβλεπτες διακυμάνσεις. Ο τροποποιημένος κανόνας ενημέρωσης φερομόνης επιτρέπει στον αλγόριθμο ACO να προσαρμόζεται ταχύτερα στις αλλαγές των απαιτήσεων εργασιών, βελτιστοποιώντας την κατανομή πόρων και ενισχύοντας τη συνολική απόδοση του συστήματος.

Επιπλέον, το ACO έχει εφαρμοστεί σε διάφορα προβλήματα χρονοπρογραμματισμού πέραν της εξισορρόπησης φορτίου, συμπεριλαμβανομένου του χρονοπρογραμματισμού εργασιών σε grid και cloud περιβάλλοντα. Για παράδειγμα, οι Srikanth et al. (2012) χρησιμοποίησαν το ACO για τον χρονοπρογραμματισμό εργασιών, αναδεικνύοντας την ικανότητά του να ελαχιστοποιεί τους μέσους χρόνους αναμονής και να βελτιώνει την αξιοποίηση των πόρων. Αυτή η προσαρμοστικότητα και αποτελεσματικότητα καθιστούν το ACO μια αξιόλογη επιλογή για τη διαχείριση των σύνθετων απαιτήσεων του χρονοπρογραμματισμού σε περιβάλλοντα cloud computing.

Η ευελιξία του ACO επιτρέπει επίσης τον συνδυασμό του με άλλες τεχνικές βελτιστοποίησης, δημιουργώντας υβριδικά μοντέλα που αξιοποιούν τα πλεονεκτήματα πολλαπλών αλγορίθμων. Αυτή η υβριδοποίηση μπορεί να οδηγήσει σε βελτιωμένη απόδοση στον χρονοπρογραμματισμό εργασιών, ειδικά σε περιβάλλοντα με υψηλή μεταβλητότητα και αβεβαιότητα.

2.2.3 Χρονοπρογραμματισμός με Βάση τους Γενετικούς Αλγόριθμους

Οι γενετικοί αλγόριθμοι (Genetic Algorithms - GAs) έχουν εφαρμοστεί ευρέως στον χρονοπρογραμματισμό εργασιών, προσφέροντας ισχυρές λύσεις για τη βελτιστοποίηση της κατανομής πόρων σε cloud computing περιβάλλοντα. Αυτοί οι

αλγόριθμοι εμπνέονται από τις αρχές της φυσικής επιλογής και της γενετικής, επιτρέποντάς τους να εξερευνούν αποτελεσματικά ένα μεγάλο χώρο λύσεων. Οι Kaur και Verma (2012) ερεύνησαν μια αποδοτική προσέγγιση GA για τον χρονοπρογραμματισμό εργασιών, η οποία ελαχιστοποιεί τον χρόνο εκτέλεσης και μεγιστοποιεί την αξιοποίηση των πόρων. Η έρευνά τους αναδεικνύει την προσαρμοστικότητα των GAs, γεγονός που τους καθιστά ιδιαίτερα κατάλληλους για περιβάλλοντα **Knative**, όπου η δυνατότητα ταχείας απόκρισης σε μεταβαλλόμενους φόρτους εργασίας είναι κρίσιμη για τη διατήρηση της απόδοσης και της αποδοτικότητας.

Οι Liu et al. (2013) ανέπτυξαν περαιτέρω το πεδίο αναπτύσσοντας έναν πολυ-αντικειμενικό γενετικό αλγόριθμο για τον χρονοπρογραμματισμό εργασιών. Το έργο αυτό υπογραμμίζει το δυναμικό των GAs να εξισορροπούν πολλαπλά κριτήρια, όπως το κόστος, την απόδοση και την κατανομή πόρων, κάτι που είναι ζωτικής σημασίας για την αποτελεσματική εξισορρόπηση φορτίου σε serverless περιβάλλοντα. Με την ενσωμάτωση πολλαπλών αντικειμενικών κριτηρίων, η προσέγγισή τους επιτρέπει μια πιο διαφοροποιημένη στρατηγική χρονοπρογραμματισμού, που μπορεί να ανταποκρίνεται σε ποικίλες απαιτήσεις των χρηστών και λειτουργικούς περιορισμούς.

Επιπλέον, η ευελιξία των GAs τους επιτρέπει να συνδυαστούν με άλλες τεχνικές βελτιστοποίησης, δημιουργώντας υβριδικά μοντέλα που μπορούν να αξιοποιήσουν τα πλεονεκτήματα πολλών αλγορίθμων. Για παράδειγμα, ο συνδυασμός των GAs με ευρετικές μεθόδους μπορεί να βελτιώσει την απόδοσή τους σε πολύπλοκα σενάρια χρονοπρογραμματισμού, επιτρέποντας καλύτερη σύγκλιση και ποιότητα λύσεων. Αυτή η υβριδοποίηση είναι ιδιαίτερα επωφελής σε δυναμικά περιβάλλοντα όπως το **Knative**, όπου οι φόρτοι εργασίας παρουσιάζουν σημαντικές και απρόβλεπτες διακυμάνσεις.

Επιπλέον, οι GAs έχουν αποδειχθεί αποτελεσματικοί στην αντιμετώπιση προκλήσεων, όπως η σύγκρουση πόρων και η προτεραιοποίηση εργασιών, οι οποίες είναι κρίσιμες σε περιβάλλοντα cloud computing. Η ικανότητα εξέλιξης των λύσεων σε διαδοχικές γενεές επιτρέπει στους GAs να ξεφεύγουν από τοπικά άκρα και να εξερευνούν μια ευρύτερη γκάμα πιθανών στρατηγικών χρονοπρογραμματισμού, οδηγώντας τελικά σε πιο αποδοτική χρήση πόρων και βελτιωμένη συνολική απόδοση του συστήματος.

2.2.4 Χρονοπρογραμματισμός με Βάση τη Fuzzy Logic

Η fuzzy logic έχει χρησιμοποιηθεί αποτελεσματικά για τη διαχείριση των αβεβαιοτήτων στην προσβασιμότητα των πόρων και τους φόρτους εργασίας, καθιστώντας την μια κατάλληλη προσέγγιση για δυναμικό χρονοπρογραμματισμό σε cloud περιβάλλοντα. Αυτή η μεθοδολογία είναι ιδιαίτερα επωφελής σε σενάρια όπου η ακριβής πληροφόρηση σχετικά με τους φόρτους εργασίας και τις καταστάσεις των πόρων συχνά απουσιάζει ή μεταβάλλεται. Οι Kong et al. (2010) πρότειναν έναν δυναμικό αλγόριθμο χρονοπρογραμματισμού εργασιών που χρησιμοποιεί fuzzy πρόβλεψη για τη μοντελοποίηση των αβεβαιοτήτων του φόρτου εργασίας. Το έργο τους υπογραμμίζει την ικανότητα της fuzzy logic να διαχειρίζεται ασαφή δεδομένα, επιτρέποντας πιο λεπτομερείς διαδικασίες λήψης αποφάσεων κατά τον χρονοπρογραμματισμό εργασιών.

Στο πλαίσιο των περιβαλλόντων **Knative**, όπου οι φόρτοι εργασίας μπορεί να είναι απρόβλεπτοι και να διαφέρουν σημαντικά, αυτή η προσέγγιση με βάση τη fuzzy logic μπορεί να αποδειχθεί ιδιαίτερα επωφελής. Με την ενσωμάτωση της fuzzy logic, ο αλγόριθμος χρονοπρογραμματισμού μπορεί να προσαρμόζεται σε

πραγματικό χρόνο στις αλλαγές των απαιτήσεων φόρτου εργασίας, οδηγώντας σε πιο άμεσες αποφάσεις χρονοπρογραμματισμού. Αυτή η προσαρμοστικότητα βελτιώνει τη συνολική απόδοση των serverless εφαρμογών, βελτιστοποιώντας την κατανομή των πόρων και ελαχιστοποιώντας την καθυστέρηση.

Επιπλέον, η fuzzy logic διευκολύνει την ενσωμάτωση πολλαπλών κριτηρίων στη διαδικασία χρονοπρογραμματισμού, όπως ο χρόνος απόκρισης, η αξιοποίηση των πόρων και οι προτεραιότητες που ορίζονται από τον χρήστη. Αυτή η πολυδιάστατη προσέγγιση επιτρέπει μια πιο ολοκληρωμένη αξιολόγηση των επιλογών χρονοπρογραμματισμού, οδηγώντας τελικά σε βελτιωμένη αποδοτικότητα στη διαχείριση πόρων. Η δυνατότητα ενσωμάτωσης ποιοτικών αξιολογήσεων παράλληλα με ποσοτικές μετρήσεις αποτελεί σημαντικό πλεονέκτημα της fuzzy logic στο cloud computing.

Επιπροσθέτως, η εφαρμογή της fuzzy logic σε αλγόριθμους χρονοπρογραμματισμού μπορεί να οδηγήσει σε καλύτερη εξισορρόπηση του φορτίου στους πόρους, καθώς μπορεί να προσαρμόζεται δυναμικά στις μεταβαλλόμενες απαιτήσεις των διαφόρων εργασιών. Αυτή η δυνατότητα είναι κρίσιμη σε περιβάλλοντα cloud, όπου η αποδοτική κατανομή των φόρτων εργασίας μπορεί να επηρεάσει σημαντικά τη συνολική απόδοση του συστήματος και την ικανοποίηση των χρηστών.

2.2.5 Άλλες Σημαντικές Εργασίες στον Χρονοπρογραμματισμό Εργασιών

Χρονοπρογραμματισμός με Περιορισμούς Προθεσμίας Οι Abrishami και Naghibzadeh (2012) εισήγαγαν έναν αλγόριθμο χρονοπρογραμματισμού ροής εργασίας με περιορισμούς προθεσμίας, προσαρμοσμένο για το **Software as a Service (SaaS)** σε cloud περιβάλλοντα. Το έργο τους δίνει έμφαση στη σημασία της τήρησης προθεσμιών παράλληλα με την αποδοτική διαχείριση των πόρων, μια κρίσιμη πτυχή για serverless αρχιτεκτονικές όπου η έγκαιρη εκτέλεση είναι υψίστης σημασίας.

Χρονοπρογραμματισμός με Βάση το Κόστος Οι Su et al. (2013) παρουσίασαν έναν αλγόριθμο χρονοπρογραμματισμού εργασιών με κριτήριο την αποτελεσματικότητα ως προς το κόστος, σχεδιασμένο για την εκτέλεση μεγάλων προγραμμάτων στο cloud. Η προσέγγισή τους αντιστοιχίζει δυναμικά τις εργασίες στις πιο αποδοτικές εικονικές μηχανές ως προς το κόστος, κάτι που είναι ιδιαίτερα σημαντικό για τη βελτιστοποίηση της κατανομής πόρων σε serverless περιβάλλοντα, όπου η διαχείριση κόστους είναι καθοριστική.

Υβριδικές Προσεγγίσεις Οι Wang et al. (2012) πρότειναν το **Cloud-DLS**, ένα δυναμικό και αξιόπιστο μοντέλο χρονοπρογραμματισμού για το cloud computing που ενσωματώνει σχέσεις εμπιστοσύνης στους αλγορίθμους χρονοπρογραμματισμού. Αυτό το μοντέλο αντιμετωπίζει την αξιοπιστία των πόρων, που είναι απαραίτητη για τη διατήρηση της απόδοσης σε serverless αρχιτεκτονικές.

2.3 Ο Αλγόριθμος AIMD

Ο αλγόριθμος Προσθετικής Αύξησης και Πολλαπλασιαστικής Μείωσης (**Additive Increase and Multiplicative Decrease Algorithm - AIMD**) είναι ένας αλγόριθμος ελέγχου με βάση την ανατροφοδότηση, που αρχικά αναπτύχθηκε για να προσφέρει έλεγχο συμφόρησης σε πρωτόκολλα δικτύου, όπως το TCP. Όταν το δίκτυο βρίσκεται σε κατάσταση υποχρησιμοποίησης, ο αλγόριθμος επιβάλλει

την προσθετική αύξηση (γραμμική αύξηση του παραθύρου συμφόρησης του αποστολέα) του ρυθμού μετάδοσης. Αντίθετα, όταν ανιχνεύεται συμφόρηση, μειώνει πολλαπλασιαστικά τον ρυθμό (δηλαδή εξασφαλίζει εκθετική μείωση του παραθύρου συμφόρησης του αποστολέα).

Ο AIMD είναι επίσης γνωστός ως αλγόριθμος αποφυγής συμφόρησης (**congestion avoidance algorithm**). Ωστόσο, είναι σημαντικό να διευκρινιστεί ότι ο πρωταρχικός του στόχος είναι να ελαχιστοποιήσει τις πιθανότητες περαιτέρω συμφόρησης, ελέγχοντας την αύξηση του παραθύρου συμφόρησης (c**ongestion window**).

Το Παράθυρο Συμφόρησης

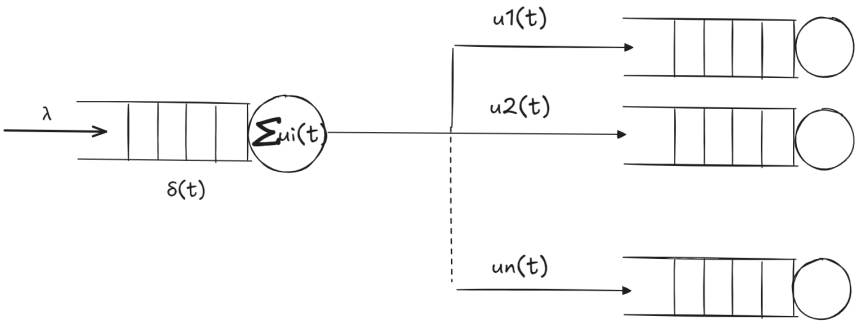
Στα πρωτόκολλα δικτύου, το παράθυρο συμφόρησης (congestion window) χρησιμοποιείται ως μηχανισμός ελέγχου για τον περιορισμό της ποσότητας των δεδομένων που μπορεί να μεταδώσει ένας αποστολέας χωρίς να λάβει επιβεβαίωση (acknowledgment) από τον καταναλωτή. Το μέγεθος του παραθύρου συμφόρησης μπορεί να προσαρμόζεται δυναμικά, τροποποιώντας παράλληλα το ρυθμό μετάδοσης του αποστολέα ανάλογα με τις τρέχουσες συνθήκες του δικτύου.
Όταν ο αλγόριθμος AIMD προσαρμόζεται για εξισορρόπηση φορτίου σε κατανεμημένα συστήματα, η έννοια του παραθύρου συμφόρησης μετατρέπεται στον τρόπο με τον οποίο κατανέμονται οι διαθέσιμοι πόροι (όπως η επεξεργαστική ισχύς) σε διαφορετικά αιτήματα. Σε αυτή την περίπτωση, ο αλγόριθμος δεν ελέγχει τη μετάδοση δεδομένων, αλλά τον αριθμό των εργασιών που ανατίθενται σε έναν συγκεκριμένο διακομιστή.

Στο [21] οι συγγραφείς επεκτείνουν τις αρχές του αλγορίθμου AIMD πέρα από την παραδοσιακή του εφαρμογή για να αντιμετωπίσουν τις προκλήσεις του χρονοπρογραμματισμού και της κατανομής πόρων σε κατανεμημένα υπολογιστικά συστήματα. Η αναφερόμενη εργασία προτείνει έναν νέο τρόπο ελέγχου του τρόπου που δέχεται τα αιτήματα ένα σύστημα χρησιμοποιώντας τη μέθοδο AIMD. Πρόκειται για μια αποκεντρωμένη προσέγγιση, καθώς κάθε υπολογιστικός κόμβος λειτουργεί αυτόνομα και χωρίς άμεση επικοινωνία με τους υπόλοιπους.

Η αρχιτεκτονική του συστήματος βασίζεται σε μια κεντρική ουρά, όπου συγκεντρώνονται αρχικά όλα τα εισερχόμενα αιτήματα. Κάθε υπολογιστικός κόμβος (server) διαθέτει τη δική του ατομική ουρά, ενώ ο κεντρικός buffer αναθέτει εργασίες με βάση τις επικρατούσες συνθήκες. Σε αυτό το σημείο εφαρμόζεται η λογική του αλγορίθμου AIMD, για να ελεγχθεί ο ρυθμός με τον οποίο οι εργασίες μεταφέρονται από την κεντρική ουρά στις μεμονωμένες ουρές των κόμβων.

Για την ενίσχυση της ανταπόκρισης του συστήματος σε διακυμάνσεις του φόρτου εργασίας, εισάγεται ένας ελεγκτής ανατροφοδότησης που ενεργοποιείται από γεγονότα. Όταν όλες οι αιτήσεις στον κεντρικό buffer εκκαθαριστούν, δημιουργείται ένα συμβάν που ωθεί τους κόμβους να μειώσουν τους ρυθμούς αποδοχής, σύμφωνα με τη φάση της Πολλαπλασιαστικής Μείωσης (MD) του αλγορίθμου. Μετά τη μείωση, το σύστημα μεταβαίνει γρήγορα στη φάση της Προσθετικής Αύξησης (AI), κατά την οποία οι ρυθμοί αποδοχής αυξάνονται σταδιακά, επιτρέποντας στο σύστημα να προσαρμόζεται δυναμικά στον φόρτο εργασίας.

Κατά τη διάρκεια της φάσης AI, το σύστημα αυξάνει σταδιακά τον ρυθμό με τον οποίο οι εργασίες εισέρχονται στους κόμβους. Εάν αυτή η αύξηση αφεθεί



Σχημα 2.1: Ένα σύστημα πολλαπλών ουρών αναμονής με πολιτική ελέγχου εισόδου τύπου AIMD

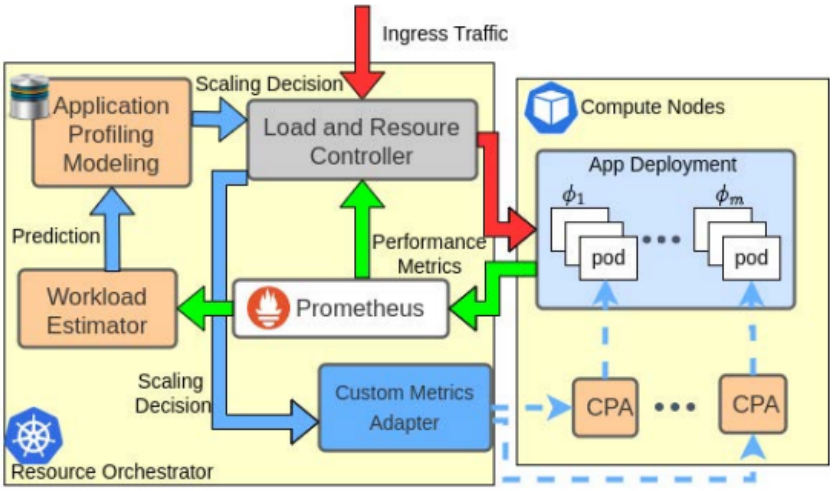
να συνεχιστεί ανεξέλεγκτα, θα μπορούσε τελικά να οδηγήσει σε υπερφόρτωση μόλις καταφθάσουν νέες εργασίες (ιδιαίτερα σε περιπτώσεις ξαφνικών αιχμών του φόρτου εργασίας). Ωστόσο, με την έναρξη της φάσης MD αμέσως μετά την εκκένωση της ουράς, αυτό το σενάριο αποτρέπεται.

Αυτός ο μηχανισμός συμβάλλει στη διασφάλιση ότι όλες οι εργασίες τελικά διεκπεραιώνονται εντός ενός πεπερασμένου χρόνου. Η βαθμιαία αύξηση του ρυθμού ελέγχου στη φάση AI, σε συνδυασμό με τη γρήγορη μείωση των ρυθμών αποδοχής στη φάση MD, δημιουργεί μια ταλάντωση γύρω από ένα σημείο ισορροπίας. Αυτή η ταλάντωση διατηρεί σταθερή την απόδοση του συστήματος, ακόμη και όταν αυτό προσαρμόζεται σε ταχείες διακυμάνσεις του φόρτου εργασίας.

Οι συγγραφείς στο [21] αποδεικνύουν ότι η πολιτική ελέγχου εισόδου που βασίζεται στον AIMD παραμένει σταθερή, ανεξαρτήτως του αριθμού των κόμβων που εμπλέκονται. Παρέχουν επίσης μαθηματικές εγγυήσεις ότι οι ουρές σε κάθε υπολογιστικό κόμβο διατηρούνται περιορισμένες και θα συγκλίνουν σε πεπερασμένο χρόνο. Οι συγγραφείς συνδέουν τις ιδιότητες σταθερότητας και οριοθέτησης της προσέγγισής τους με τη διασφάλιση μετρικών Ποιότητας Υπηρεσιών (QoS). Είναι σε θέση να υπολογίσουν τους χρόνους αναμονής στην ουρά και άλλες μετρικές QoS, προσφέροντας έτσι ντετερμινιστικές εγγυήσεις απόδοσης.

Σε εργασίες που βασίζονται στις αρχές που εισήγαγαν οι συγγραφείς του Spatharakis2022, το εν λόγω πλαίσιο επεκτείνεται ώστε να αντιμετωπιστούν επιπλέον προκλήσεις σε περιβάλλοντα υπολογιστικής ακμής. Οι συγγραφείς του [17] παρουσιάζουν μια επεκτάσιμη αρχιτεκτονική σχεδιασμένη για Kubernetes Edge Clusters (KECs), η οποία συνδυάζει τη μέθοδο χρονοπρογραμματισμού εργασιών AIMD με προηγμένες τεχνικές αυτόματης κλιμάκωσης. Ο προαναφερθείς μηχανισμός ενεργοποίησης συμβάντων ωθεί το σύστημα να αυξήσει ή να μειώσει τον αριθμό των ενεργών αντιγράφων (replicas) των υπολογιστικών κόμβων. Η διαδικασία κλιμάκωσης εκτελείται από έναν Custom Pod Autoscaler (CPA), ο οποίος μπορεί να κλιμακωθεί ακόμη και σε μηδενικά αντίγραφα, αν χρειαστεί, σε αντίθεση με τον τυπικό Kubernetes Horizontal Pod Autoscaler (HPA).

Επιπλέον, οι συγγραφείς ενσωματώνουν μια συνιστώσα Machine Learning Application Profiling Modeling, που χρησιμοποιεί μετρικές απόδοσης, κατανομή πόρων σε πραγματικό χρόνο και προγνωστική ανάλυση φόρτου εργασίας για τον προσδιορισμό του βέλτιστου αριθμού αντιγράφων. Το μοντέλο μηχανικής μάθησης εκπαιδεύεται με ιστορικά δεδομένα και διασφαλίζει ότι οι αποφάσεις κλιμάκωσης δεν είναι απλώς αντιδραστικές στις τρέχουσες συνθήκες, αλλά και προδραστικές, προβλέποντας μελλοντικές απαιτήσεις.



Σχημα 2.2: Η Προτεινόμενη Αρχιτεκτονική Συστήματος για Kubernetes Edge Clusters (KEC)

Η προτεινόμενη αρχιτεκτονική αξιολογείται στη συνέχεια εμπειρικά, με τη χρήση ενός ρεαλιστικού συνόλου δεδομένων σε ένα μικρής κλίμακας περιβάλλον υπολογιστικής ακμής. Τα αποτελέσματα δείχνουν σημαντική μείωση της χρήσης της CPU -έως και 8%- με μόνο μια μικρή αύξηση στις παραβιάσεις των μετρικών ποιότητας υπηρεσιών (QoS).

Κεφάλαιο 3

Υπόβαθρο: Knative και Σχετικά Εργαλεία

3.1 Η σημασία του Knative στο πλαίσιο του serverless computing

Όπως έχει γίνει σαφές από τα προηγούμενα κεφάλαια, τα πλεονεκτήματα των serverless εφαρμογών είναι πολυάριθμα- συγκεκριμένα, η ευελιξία, η γρήγορη ανάπτυξη και η αποδοτικότητα κόστους- οπότε δεν προκαλεί έκπληξη το γεγονός ότι τα τελευταία χρόνια οι επιχειρήσεις στρέφονται όλο και περισσότερο προς το **serverless** μοντέλο . Κατά συνέπεια, η αποτελεσματική υλοποίηση έχει γίνει πλέον επιτακτική ανάγκη.

Σε αυτό το σημείο έρχεται το **Knative**. Το Knative είναι ένα πλαίσιο serverless για το **Kubernetes** και ήδη διαδραματίζει σημαντικό ρόλο στο τοπίο του **serverless computing**. Το Knative έχει αναπτυχθεί πάνω στην πλατφόρμα Kubernetes, αξιοποιώντας τα πλεονεκτήματα της πλατφόρμας και ενσωματώνοντάς τα στην εμπειρία του serverless computing.

Το Kubernetes (επίσης γνωστό ως **k8s**) είναι μια ισχυρή πλατφόρμα ορχήστρωσης **container** ανοικτού κώδικα (**open source**), container για την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης και της διαχείρισης εφαρμογών που βασίζονται σε κοντέινερ. Η ελκυστικότητά του έγκειται σε χαρακτηριστικά όπως η επεκτασιμότητα, με ενσωματωμένους αυτόματους κλιμακωτές για οριζόντια και κάθετη κλιμάκωση, η ανακάλυψη υπηρεσιών και η εξισορρόπηση φορτίου, καθώς και ένα ακμάζον οικοσύστημα εργαλείων και πρόσθετων (plugins) που επεκτείνουν τις δυνατότητές του. Ένα σημαντικό πλεονέκτημα του Kubernetes είναι ότι διασφαλίζει υψηλή διαθεσιμότητα, αντικαθιστώντας και επαναπρογραμματίζοντας αυτόματα τα container ανάλογα με τις ανάγκες.

Το Knative έχει σχεδιαστεί για να επεκτείνει το Kubernetes παρέχοντας μια σειρά από στοιχεία **middleware** για την κάλυψη τόσο της ανάπτυξης και διαχείρισης λειτουργιών όσο και του χειρισμού ροών εργασίας που βασίζονται σε συμβάντα.

Δεν είναι η μόνη λύση serverless που βασίζεται στο Kubernetes, αλλά ξεχωρίζει ανάμεσα στις υπόλοιπες. Άλλες λύσεις serverless στο οικοσύστημα Kubernetes περιλαμβάνουν τις Kubeless, OpenFaaS, OpenWhisk και Fission.

Το **Kubeless** συχνά περιγράφεται ως το πιο Kubernetes-native framework, η εγκατάστασή του είναι εύκολη, διαθέτει μια απλή αρχιτεκτονική με Custom Resource Definitions (CRDs), αλλά -κατά τη στιγμή της συγγραφής- δεν παρέχει καμία λειτουργικότητα scale-to-zero.

OpenFaas παρέχει απλή ανάπτυξη και διαχείριση για εφαρμογές χωρίς διακομιστή και είναι εξαιρετικά ευέλικτο, καθώς μπορεί να τρέξει σε οποιαδήποτε

πλατφόρμα που υποστηρίζει το Docker (συμπεριλαμβανομένου του k8s). Μπορεί επιπλέον να προσφέρει αυτόματη κλιμάκωση με βάση μετρήσεις, εάν ενσωματωθεί με το Prometheus. Ωστόσο, το OpenFaas δεν διαθέτει ένα προηγμένο σύστημα διαχείρισης συμβάντων και μπορεί να εισάγει μεγάλη επιβάρυνση λόγω της εξάρτησής του από τα containers του Docker για κάθε λειτουργία.

OpenWhisk, από την άλλη πλευρά, διαθέτει ισχυρή υποστήριξη για ροές εργασίας που βασίζονται σε συμβάντα. Ωστόσο, δεν είναι στενά ενσωματωμένο με το Kubernetes και μπορεί να εισάγει επιβάρυνση επιδόσεων λόγω των επιπέδων αφαίρεσης που χρησιμοποιεί.

Τέλος, το **Fission** είναι βελτιστοποιημένο για εκτέλεση συναρτήσεων χαμηλής καθυστέρησης και για μικρής κλίμακας, απλούστερα deployments παρέχει μια απλή και γρήγορη εμπειρία ανάπτυξης. Τούτου λεχθέντος, διαθέτει λιγότερο ώριμες δυνατότητες χειρισμού συμβάντων και υποστηρίζει λιγότερες γλώσσες προγραμματισμού σε σύγκριση με άλλες λύσεις.

Εστιάζοντας ξανά στο **Knative**, σε σύγκριση με τις προαναφερθείσες λύσεις, θεωρείται μια πιο ολοκληρωμένη (ολιστική) προσέγγιση, καθώς προσφέρει εργαλεία τόσο για το **serving** όσο και για το **eventing** (περισσότερες λεπτομέρειες σχετικά με αυτές τις δύο έννοιες θα παρουσιαστούν αργότερα). Η ενσωμάτωση του **Knative** με το **Kubernetes** είναι επίσης πιο βαθιά και ολοκληρωμένη. Μια σημαντική διάκριση είναι το προηγμένο σύστημα **eventing** του **Knative** και η υποστήριξη που προσφέρει για το **Istio** (καθώς και για άλλα **service meshes**), το οποίο επιτρέπει την αποτελεσματική διαχείριση πιο σύνθετων και κατανεμημένων εφαρμογών. Πρόκειται για ένα εξαιρετικά ευέλικτο πλαίσιο με ισχυρή κοινότητα ανοιχτού κώδικα και υποστηρίζεται από την **Google**.

Η ευελιξία του **Knative** επιτρέπει στις επιχειρήσεις να αναπτύσσουν ένα ευρύ φάσμα εφαρμογών, που εκτείνεται από μονολιθικές έως αρχιτεκτονικές βασισμένες σε μικροϋπηρεσίες (microservices) και ακόμη και function-level workloads. Επιπλέον, η συμβατότητα του **Knative** με οποιαδήποτε πλατφόρμα **cloud** που υποστηρίζεται από το **Kubernetes** παρέχει στους οργανισμούς την ευελιξία να εκτελούν τα **serverless** φορτία εργασίας τους χωρίς να περιορίζονται από τα συγκεκριμένα χαρακτηριστικά ενός παρόχου **cloud**. Αυτό το επίπεδο ευελιξίας και ανεξαρτησίας επιτρέπει στις επιχειρήσεις να προσαρμόζουν και να κλιμακώνουν δυναμικά τις **serverless** εφαρμογές τους, σε πλήρη ευθυγράμμιση με τις μοναδικές λειτουργικές απαιτήσεις τους, οδηγώντας τελικά σε αυξημένη ευκινησία και αποδοτικότητα.

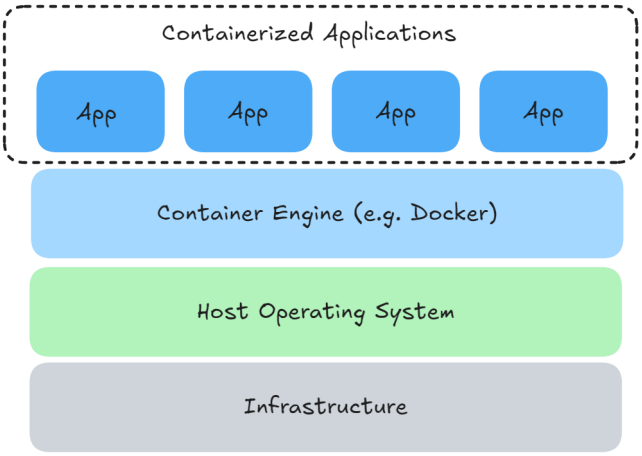
3.2 Μια σύντομη εισαγωγή στο Kubernetes

Στις επόμενες ενότητες αυτού του κεφαλαίου, το Knative θα αναλυθεί και θα συζητηθεί διεξοδικά, καλύπτοντας τις βασικές λειτουργίες του, τα συστατικά που το αποτελούν και τη συνολική αρχιτεκτονική του. Ωστόσο, δεδομένου ότι το Knative βασίζεται στο Kubernetes, είναι απαραίτητο να παρουσιάσουμε πρώτα ορισμένες βασικές έννοιες του Kubernetes, το οποίο θεωρείται πλέον η βάση των σύγχρονων cloud-native αρχιτεκτονικών.

Το Kubernetes, γνωστό και ως K8s, είναι ένα **σύστημα διαχείρισης container** που βοηθά τους προγραμματιστές εφαρμογών στην εύκολη ανάπτυξη, κλιμάκωση και συντήρηση cloud-native εφαρμογών.

Η ανάπτυξη εφαρμογών στο νέφος επιτυγχάνεται ευκολότερα μέσω της χρήσης των containers. Σε σύγκριση με τις εικονικές μηχανές, τα container είναι πιο ελαφριά, με αποτέλεσμα την ταχύτερη ανάπτυξη και τη μείωση του χρόνου

εκκίνησης. Ένα **container** είναι ένα τυποποιημένο πακέτο λογισμικού. Συγκεντρώνει τον κώδικα και όλες τις απαραίτητες εξαρτήσεις, έτσι ώστε οι εφαρμογές να μπορούν να εκτελούνται με συνέπεια σε διαφορετικά υπολογιστικά περιβάλλοντα.



Σχημα 3.1: Διάγραμμα εφαρμογών σε container

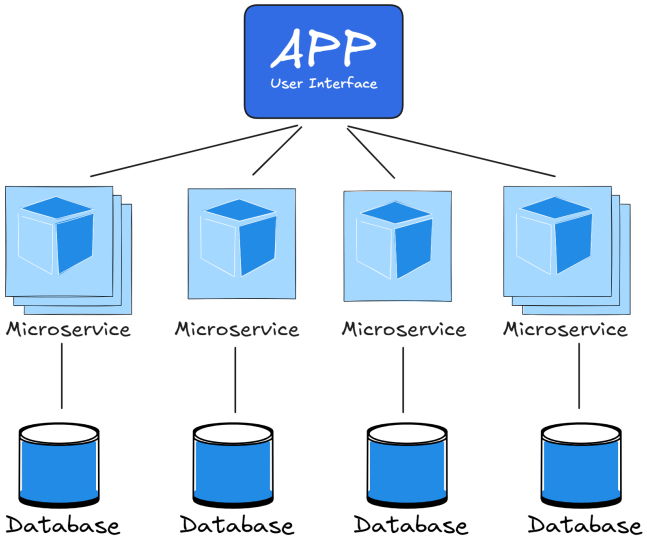
Χρόνος Εκτέλεσης Containers

Ένας χρόνος εκτέλεσης containers (**container runtime**) είναι ένα στοιχείο λογισμικού υπεύθυνο για την εκτέλεση των containers σε ένα λειτουργικό σύστημα υποδοχής (host). Αλληλεπιδρά με το υποκείμενο λειτουργικό σύστημα για να παρέχει τους απαραίτητους πόρους (CPU, μνήμη) για το σύνολο του κύκλου ζωής των containers, δηλαδή τη δημιουργία, την εκτέλεση και τον τερματισμό τους. Το πιο γνωστό πρόγραμμα εκτέλεσης containers είναι το **Docker**, το οποίο υλοποιεί τα πρότυπα της **Open Container Initiative (OCI)**[1]. Εκτός από την εκτέλεση containers, άλλες βασικές λειτουργίες του **Docker** περιλαμβάνουν τη διαχείριση εικόνων (images), τις δυνατότητες δικτύωσης, την απομόνωση πόρων και τη διαχείριση volumes.

Η προσέγγιση της κοντεϊνοποίησης (**containerization**) αναπτύχθηκε παράλληλα με τη στροφή προς την αρχιτεκτονική των μικροϋπηρεσιών (**microservices**), όπου οι εφαρμογές διασπώνται σε μικρότερες, ανεξάρτητες υπηρεσίες που επικοινωνούν μεταξύ τους μέσω του δικτύου. Το κύριο πλεονέκτημα της προσέγγισης των μικροϋπηρεσιών είναι ότι κάθε υπηρεσία μπορεί να αναπτυχθεί, να εκτελεστεί και να κλιμακωθεί ξεχωριστά και ανεξάρτητα από τις άλλες. Αυτό όμως συνεπάγεται ότι υπάρχει μια ποικιλία διαφορετικών containers σε ένα κατανεμημένο περιβάλλον, τα οποία πρέπει να ενορχηστρωθούν ως προς την κλιμάκωση, τη δικτύωση και τη διασφάλιση της συνέπειας κατά την ανάπτυξή τους.

Εδώ έρχεται το Kubernetes, ως εργαλείο ενορχήστρωσης containers. Παρέχει βασικά χαρακτηριστικά, όπως η ανακάλυψη υπηρεσιών και η εξισορρόπηση φορτίου, και απλοποιεί σημαντικά την επιχειρησιακή πολυπλοκότητα της εκτέλεσης μικροϋπηρεσιών σε κλίμακα.

[1]Η Open Container Initiative είναι ένας οργανισμός ανοιχτής διακυβέρνησης με σκοπό τη δημιουργία ανοικτών βιομηχανικών προτύπων σχετικά με τα container formats και τα runtimes. Πηγή: https://opencontainers.org/



Σχημα 3.2: Αρχιτεκτονική Microservices

Οι μικρότερες μονάδες που μπορούν να αναπτυχθούν στο Kubernetes είναι τα pods και αποτελούν τα θεμελιώδη δομικά στοιχεία για την εκτέλεση containerized εφαρμογών. Ένα **pod** αποτελείται από έναν ή περισσότερους containers που μοιράζονται τον ίδιο χώρο ονομάτων δικτύου και αποθήκευσης (namespace).

Τα pods εκτελούνται σε κόμβους (**nodes**). Ένας κόμβος είναι η φυσική ή εικονική μηχανή στην οποία λειτουργεί το Kubernetes. Ένας κόμβος περιέχει τις απαραίτητες υπηρεσίες για την εκτέλεση των pods και διαχειρίζεται από το **Control Plane**.

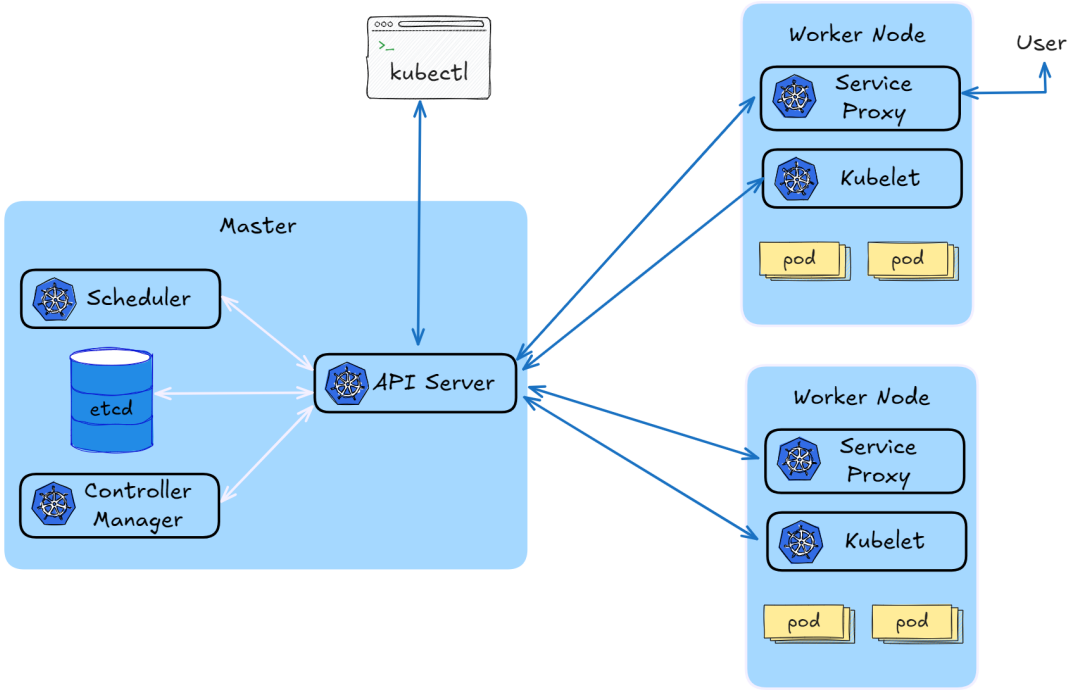
Το **Control Plane** παρέχει βασικά στοιχεία που διαχειρίζονται την κατάσταση του Kubernetes cluster. Ο **API Server** είναι το front-end του control plane και χειρίζεται τα αιτήματα RESTful API. Ο **Scheduler** αναθέτει pods σε κόμβους βάσει της διαθεσιμότητας πόρων. Ο **Controller Manager** διαχειρίζεται τις αναπτύξεις, τα replicas και τα endpoints μέσω εργασιών ρουτίνας. Τέλος, το **etcd** είναι ένας κατανεμημένος αποθηκευτικός χώρος κλειδιού-τιμής, όπου αποθηκεύονται όλα τα δεδομένα και οι διαμορφώσεις του cluster.

Το Kubernetes ακολουθεί μια αρχιτεκτονική **master-slave**, όπου το κύριο στοιχείο αντιπροσωπεύεται από έναν ή (για την αποφυγή μεμονωμένων σημείων αποτυχίας) περισσότερους κύριους κόμβους που είναι υπεύθυνοι για τον έλεγχο της ενορχήστρωσης των containers μέσω του **control plane**. Από την άλλη πλευρά, οι worker κόμβοι είναι οι μηχανές που εκτελούν τα pods.

Το **Kubelet** είναι ένα βασικό στοιχείο (component) που εκτελείται σε κάθε worker κόμβο και επικοινωνεί συνεχώς με το **control plane** για να διασφαλίσει ότι τα containers που εκτελούνται στο pod λειτουργούν σωστά και είναι υγιή. Το άλλο στοιχείο που εκτελείται σε κάθε κόμβο είναι το service proxy, γνωστό και ως **Kube Proxy**. Λειτουργεί ως μεσάζων δικτύου και εξισορροπητής φορτίου, λειτουργώντας τόσο στο επίπεδο δικτύου (δρομολόγηση βάσει διευθύνσεων IP) όσο και στο επίπεδο μεταφοράς (δρομολόγηση βάσει θυρών - **port-based routing**).

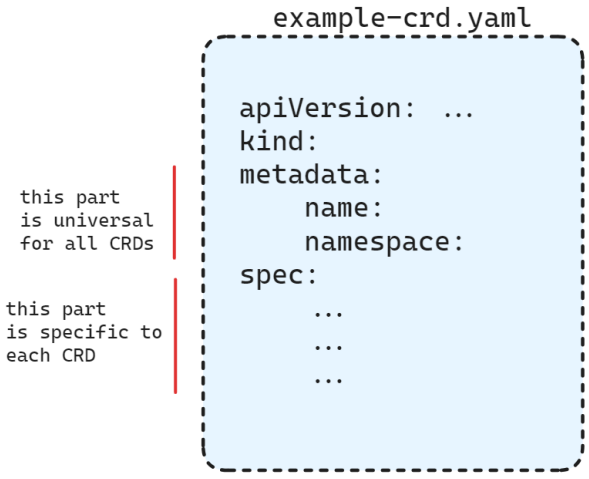
Χτίζοντας πάνω στο K8s

Το Kubernetes διαθέτει ενσωματωμένους πόρους (Pods, Services κ.λπ.), αλλά υπάρχουν περιπτώσεις όπου προκύπτει η ανάγκη επέκτασης του API του K8s. Αυτό μπορεί να επιτευχθεί μέσω του ορισμού προσαρμοσμένων πόρων με Custom



Σχημα 3.3: Kubernetes Architecture

Resource Definitions (CRDs). Τα CRDs αποτελούν ένα ισχυρό χαρακτηριστικό, καθώς, μόλις οριστούν, το K8s μπορεί να διαχειρίζεται τους προσαρμοσμένους πόρους όπως και τους εγγενείς, επιτρέποντας την πλήρη ενσωμάτωσή τους με components του K8s, όπως το 'kubectl', ο API server, κ.λπ.



Σχημα 3.4: Παράδειγμα ενός CRD

Επιστρέφοντας στο Knative, είναι αυτό ακριβώς το χαρακτηριστικό που αξιοποιεί για να χτίσει δυνατότητες χωρίς διακομιστές και με βάση τα συμβάντα πάνω στο Kubernetes.

3.3 Τι είναι το Knative;

3.3.1 Ορισμός του Knative και ο σκοπός του

Το Knative είναι μια **open-source**, **platform-agnostic** λύση σχεδιασμένη να επεκτείνει το **Kubernetes** για την εκτέλεση **serverless** αναπτύξεων. Εργαζόμενοι με το Knative, οι προγραμματιστές μπορούν να κατασκευάσουν, να αναπτύξουν και να διαχειριστούν τις serverless εφαρμογές τους, εστιάζοντας στη συγγραφή κώδικα και όχι στη διαχείριση της υποκείμενης υποδομής. Το Knative προσφέρει βασικά χαρακτηριστικά, όπως η αυτόματη κλιμάκωση, που περιλαμβάνει την κλιμάκωση σε μηδενικά instances, και παρέχει ένα πλαίσιο για τη δημιουργία **event-driven** εφαρμογών. Επιλέγεται από προγραμματιστές που επιθυμούν να αξιοποιήσουν το Kubernetes για serverless computing, διατηρώντας παράλληλα την ευελιξία και τον έλεγχο των εφαρμογών τους. Το Knative παρέχει υψηλότερου επιπέδου αφαίρεση και δυνατότητες στο k8s, ειδικά προσαρμοσμένες για serverless φόρτους εργασίας.

Το Knative δημιουργήθηκε για να απλοποιήσει την ανάπτυξη και τη λειτουργία cloud-native, serverless εφαρμογών, παρέχοντας βασικά components για: Deploying (διαχείριση του κύκλου ζωής ανάπτυξης των εφαρμογών, συμπεριλαμβανομένης της έκδοσης και του διαχωρισμού της κίνησης), **Serving** (αυτόματη κλιμάκωση των εφαρμογών ανάλογα με τη ζήτηση και δρομολόγηση της κίνησης στη σωστή έκδοση), και **Eventing** (διαχείριση αρχιτεκτονικών που βασίζονται σε συμβάντα, με χειρισμό της λήψης, της δρομολόγησης και της παράδοσης συμβάντων).

3.3.2 Προέλευση και εξέλιξη του Knative

Το έργο Knative, που παρουσιάστηκε από την Google τον Ιούλιο του 2018, αναπτύχθηκε σε στενή συνεργασία με ηγέτες του κλάδου, όπως οι VMWare, IBM, Red Hat και SAP. Αν και αρχικά ξεκίνησε από την Google, το έργο πλέον συντηρείται από την ευρύτερη κοινότητα. Το Knative δημιουργήθηκε για να αντιμετωπίσει τις εγγενείς πολυπλοκότητες που σχετίζονται με την κατασκευή, την ανάπτυξη και τη διαχείριση serverless εφαρμογών στην πλατφόρμα Kubernetes. Ως πρωτοβουλία ανοικτού κώδικα, το έργο στοχεύει στην προώθηση ευρείας υποστήριξης από τον κλάδο και στην αποφυγή vendor lock-in, τοποθετώντας το Knative ως μια πραγματικά **platform-agnostic** λύση. Με την πάροδο του χρόνου, η κοινότητα του Knative έχει αναπτυχθεί σημαντικά, με πολυάριθμους συνεισφέροντες και ένα ευρύ φάσμα ενσωματώσεων με διάφορα άλλα έργα.

Οι δυνατότητες **eventing** του Knative έχουν εξελιχθεί με την πάροδο του χρόνου και έχουν ενσωματώσει την εισαγωγή του **CloudEvents για τυποποιημένο χειρισμό συμβάντων**. Το CloudEvents παρέχει έναν τυποποιημένο τρόπο περιγραφής δεδομένων συμβάντων σε διαφορετικές υπηρεσίες και πλατφόρμες. Αυτή η τυποποίηση απλοποιεί την ανάπτυξη εφαρμογών που βασίζονται σε συμβάντα, εξασφαλίζοντας συμβατότητα και διαλειτουργικότητα μεταξύ διαφόρων πηγών και καταναλωτών συμβάντων. Οι προγραμματιστές μπορούν πλέον να δημιουργούν εφαρμογές που μπορούν να χειρίζονται συμβάντα με συνεπή τρόπο, ανεξάρτητα από την προέλευσή τους.

Το CloudEvents επέτρεψε την απρόσκοπτη ενσωμάτωση με διαμεσολαβητές συμβάντων, όπως το Kafka, το RabbitMQ και το Google Cloud Pub/Sub, διευκολύνοντας πιο ισχυρές και κλιμακούμενες αρχιτεκτονικές με γνώμονα τα συμβάντα.

Με την πάροδο του χρόνου, το Knative έχει ενσωματώσει πιο προηγμένες μεθόδους για **autoscaling**. Αυτές περιλαμβάνουν την κλιμάκωση βάσει **concurrency**, η οποία προσαρμόζει την κλιμάκωση με βάση τον αριθμό των ταυτόχρονων αιτημάτων, διαχειριζόμενη ξαφνικές αυξήσεις στην κίνηση χωρίς περιττή κατανομή πόρων. Επιπλέον, προσφέρει **μετρήσεις καθοριζόμενες από τον χρήστη**, επιτρέποντας στους χρήστες να ορίζουν εξατομικευμένες μετρήσεις για την αυτόματη κλιμάκωση, παρέχοντας μεγαλύτερη ακρίβεια και βελτιστοποίηση προσαρμοσμένη στις συγκεκριμένες απαιτήσεις της εφαρμογής. Τέλος, το Knative **ενσωματώνεται με το HPA**, προσφέροντας προσαρμόσιμες και ισχυρές τακτικές κλιμάκωσης.

Αρχικά, το Knative επέτρεπε την απλή κατανομή της κίνησης μεταξύ των διαφόρων εκδόσεων μιας εφαρμογής. Αυτή η λειτουργικότητα έχει εξελιχθεί ώστε να περιλαμβάνει σταδιακές αναπτύξεις, δοκιμές A/B και blue-green αναπτύξεις. Αυτές οι λειτουργίες έχουν προσφέρει στους προγραμματιστές αυξημένο έλεγχο στις αναπτύξεις των εφαρμογών, επιτρέποντας ασφαλέστερες και πιο ελεγχόμενες στρατηγικές έκδοσης με γνώμονα τα δεδομένα.

Επιπλέον, το Knative υποστηρίζει τις εκδόσεις **canary**, επιτρέποντας στους προγραμματιστές να αναπτύσσουν σταδιακά νέες εκδόσεις των εφαρμογών τους και να παρακολουθούν την απόδοσή τους πριν από την πλήρη μετάβαση (Mampage et al., 2022).

3.3.3 Σημαντικά χαρακτηριστικά και τα πλεονεκτήματά τους

Αρχικά, όπως έχει ήδη αναφερθεί, το Knative είναι φορητό σε οποιοδήποτε περιβάλλον Kubernetes, είτε σε εγκαταστάσεις on-premises είτε στους διάφορους διαθέσιμους παρόχους cloud. Το κύριο πλεονέκτημα αυτού είναι η μείωση του συνήθους κινδύνου vendor lock-in, καθώς η διαλειτουργικότητα είναι εγγυημένη.

Το Knative χαρακτηρίζεται επίσης ως μια ιδιαίτερα φιλική προς τους προγραμματιστές πλατφόρμα, καθώς απλοποιεί την πολυπλοκότητα του Kubernetes—ενώ ενσωματώνεται εύκολα με τα υπάρχοντα εργαλεία του k8s—και υποστηρίζει συνεχή ολοκλήρωση και συνεχή ανάπτυξη (γνωστή ως CI/CD).

Παρέχει ένα τυποποιημένο πλαίσιο **CloudEvent** για την ανάπτυξη serverless αρχιτεκτονικής, γεγονός που το διαφοροποιεί από άλλες λύσεις FaaS, προσφέροντας τυποποιημένα συμβάντα και διαλειτουργικότητα με διάφορες πλατφόρμες FaaS. Αυτό απλοποιεί σημαντικά την ανάπτυξη cross-cloud εφαρμογών.

Ένα βασικό χαρακτηριστικό του Knative είναι οι προηγμένες δυνατότητες αυτόματης κλιμάκωσης, οι οποίες περιλαμβάνουν τη δυνατότητα για υπηρεσίες να κλιμακώνονται ακόμη και σε μηδενικά instances. Το Knative υποστηρίζει την αυτόματη κλιμάκωση με βάση το **concurrency**, πράγμα που σημαίνει ότι επιτρέπει στην πλατφόρμα να προσαρμόζει τους πόρους σύμφωνα με τα επίπεδα των ταυτόχρονων αιτήσεων.

Η έννοια της ταυτόχρονης εξυπηρέτησης (concurrency) στα Serverless περιβάλλοντα και το Knative

Στο πλαίσιο των serverless περιβαλλόντων, η ταυτόχρονη εξυπηρέτηση (**concurrency**) αναφέρεται στον αριθμό των αιτημάτων που μπορεί να χειριστεί ταυτόχρονα μια λειτουργία ή υπηρεσία. Η διαχείριση της ταυτόχρονης εξυπηρέτησης είναι ζωτικής σημασίας για τη βελτιστοποίηση της χρήσης των πόρων και τη διασφάλιση της επεκτασιμότητας.

Το Knative διαθέτει τον δικό του αυτόματο κλιμακωτή, το Knative Pod Autoscaler (KPA) -περισσότερα γι' αυτό αργότερα- αλλά ενσωματώνεται επίσης με το HPA του Kubernetes για επιπλέον στρατηγικές κλιμάκωσης. Αυτό, σε συνδυασμό με το γεγονός ότι επιτρέπει τον ορισμό μετρήσεων από τον χρήστη για την αυτόματη κλιμάκωση, επιτρέπει στους προγραμματιστές να εφαρμόζουν προσαρμοσμένη κλιμάκωση που ανταποκρίνεται στις ειδικές απαιτήσεις των εφαρμογών τους.

Το **Knative Serving**, το πρώτο βασικό συστατικό του Knative, επιτρέπει στους προγραμματιστές να διαχειρίζονται εύκολα τον κύκλο ζωής των serverless εφαρμογών, ενώ το **Eventing**, το δεύτερο κύριο συστατικό, προσθέτει τις δυνατότητες αρχιτεκτονικής που βασίζονται σε συμβάντα. Το Knative υποστηρίζει ένα ευρύ φάσμα πηγών και προορισμών συμβάντων, και ενσωματώνεται με διάφορους δημοφιλείς **brokers**, όπως το **Kafka**, το **RabbitMQ** και το **Google Cloud Pub/Sub**. Σε ένα περιβάλλον **eventing** του Knative, οι παραγωγοί συμβάντων είναι ανεξάρτητοι από τους καταναλωτές, επιτρέποντας έτσι ευέλικτες και κλιμακούμενες ροές εργασίας βασισμένες σε συμβάντα.

Όλες οι παραπάνω δυνατότητες καθίστανται δυνατές χάρη στην προσεκτικά σχεδιασμένη αρχιτεκτονική του Knative.

3.3.4 Επισκόπηση της Αρχιτεκτονικής του Knative

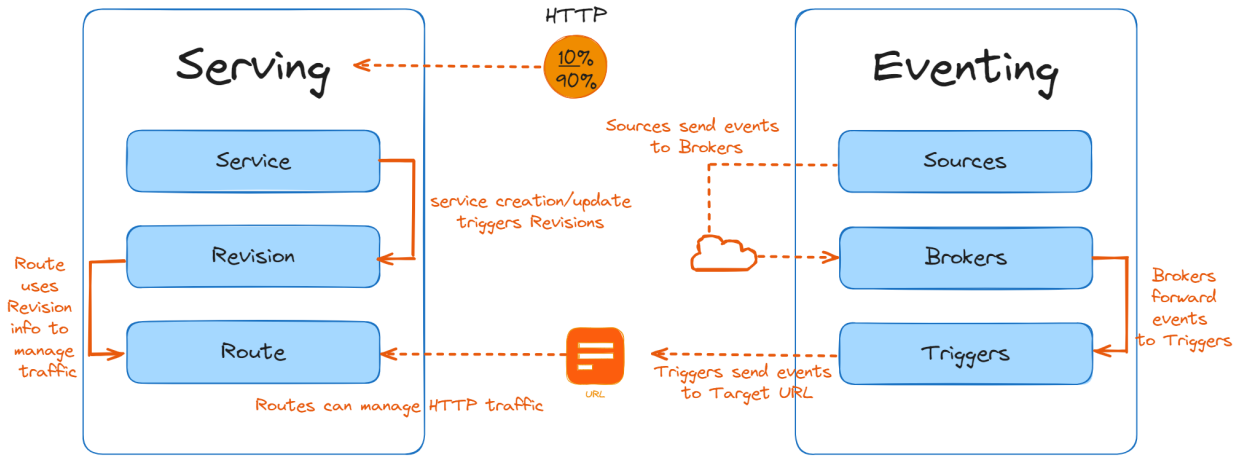
Το Knative αποτελεί ένα σύνολο υπηρεσιών που έχουν σχεδιαστεί ειδικά για την εκτέλεση σε ένα Kubernetes cluster, προκειμένου να διαχειρίζονται τις πρόσθετες εργασίες που απαιτούνται για την υλοποίηση serverless εφαρμογών σε αυτό. Με άλλα λόγια, αποτελεί ένα επίπεδο αφαίρεσης πάνω από το Kubernetes, σχεδιασμένο για να παρέχει έναν τυποποιημένο τρόπο εκκίνησης της ανάπτυξης serverless εφαρμογών. Για την επίτευξη αυτού, το Knative βασίζεται σε δύο κύρια συστατικά: το **Knative Serving** και το **Knative Eventing**. Το κάθε ένα από αυτά περιλαμβάνει διάφορα στοιχεία που έχουν συσκευαστεί ως εμπορευματοκιβώτια (containers) και **CRDs** (Custom Resource Definitions), τα οποία είναι υπεύθυνα για τις διακριτές λειτουργίες που θα αναλυθούν στη συνέχεια.

Συστατικά και οι Αλληλεπιδράσεις τους

Στην ουσία, το **Knative Serving** διαχειρίζεται τον κύκλο ζωής και την κίνηση των serverless εφαρμογών, ενώ το **Knative Eventing** αναλαμβάνει την παραγωγή, το φιλτράρισμα και την κατανάλωση συμβάντων με έναν αποσυνδεδεμένο τρόπο. Τα βέλη στο ακόλουθο διάγραμμα υποδεικνύουν τη ροή των ρυθμίσεων, των αναθεωρήσεων, της κίνησης και των συμβάντων εντός του οικοσυστήματος του Knative:

Οι **συνεχόμενες γραμμές** στο διάγραμμα αντιπροσωπεύουν αλληλεπιδράσεις ή εξαρτήσεις που είναι **άμεσες** και **συνεχείς**. Για παράδειγμα, το Service καθορίζει traffic policies (πολιτικές κίνησης) για το Route και το Route διαχειρίζεται την κίνηση προς τα Revisions. Αυτές οι αλληλεπιδράσεις είναι κρίσιμες για την άμεση λειτουργία των αναφερόμενων components.

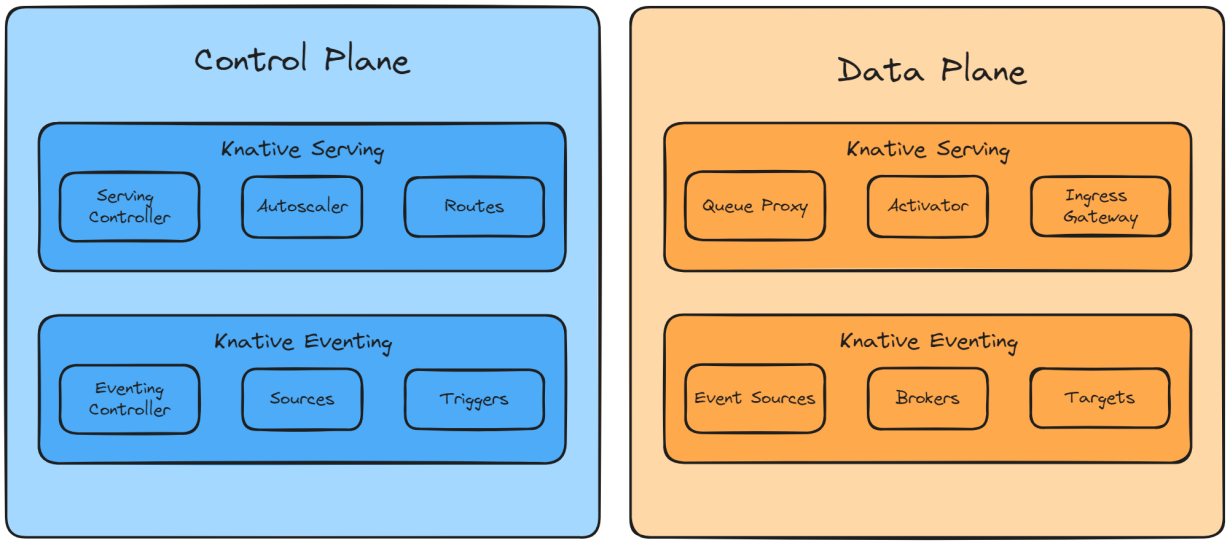
Αντιθέτως, οι **διακεκομμένες γραμμές** στο διάγραμμα αντιπροσωπεύουν **έμμεσες** και **ασύγχρονες** αλληλεπιδράσεις που μπορεί να συμβούν βάσει ορισμένων συνθηκών ή **events**. Παραδείγματα αυτών θα μπορούσαν να είναι η παραγωγή και κατανάλωση events στο πλαίσιο του Knative Eventing.



Σχημα 3.5: The κύρια components του Knative

Control Plane Vs. Data Plane

Στο Knative, το Control Plane και το Data Plane διαδραματίζουν διακριτούς αλλά συμπληρωματικούς ρόλους. Το **Control Plane** περιλαμβάνει components που είναι υπεύθυνα για την ενορχήστρωση (orchestration) της συνολικής λειτουργίας του συστήματος, όπως ο Serving Controller, ο Eventing Controller και ο Autoscaler. Αυτά διαχειρίζονται τη διαμόρφωση (configuration), την ανάπτυξη (deployment), την κλιμάκωση (scaling) και τη δρομολόγηση (routing) των εφαρμογών.



Σχημα 3.6: Control Plane και Data Plane

Παράλληλα, το **Data Plane** επικεντρώνεται στο χειρισμό της πραγματικής επεξεργασίας δεδομένων, όπως η επεξεργασία HTTP requests και events. Περιλαμβάνει βασικά components όπως το Queue Proxy, το Activator και το Ingress Gateway.

Ο διαχωρισμός αυτός είναι χρήσιμος, καθώς επιτρέπει στους προγραμματιστές να καθορίζουν ποια είναι η επιθυμητή κατάσταση του συστήματος, χρησιμοποιώντας το Control Plane, ενώ το Data Plane φροντίζει να διασφαλίζει

ότι αυτές οι ρυθμίσεις εφαρμόζονται σε πραγματικό χρόνο και διαχειρίζεται τον πραγματικό χειρισμό των requests.

3.3.5 Knative Serving

Σκοπός και λειτουργικότητα

Το Knative Serving αποτελεί ένα από τα δύο βασικά components που απαρτίζουν το Knative. Ο κύριος σκοπός του Serving είναι η απλοποίηση της λειτουργικής επιβάρυνσης που συνοδεύει την ανάπτυξη και τη διαχείριση stateless υπηρεσιών.

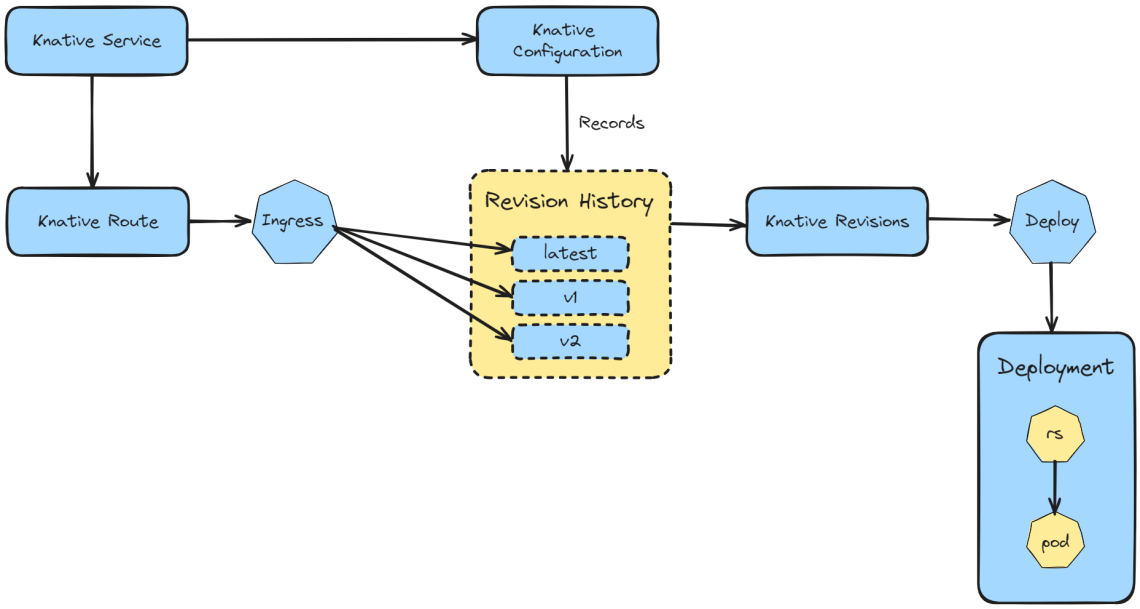
Αυτό το component είναι υπεύθυνο για τις αυτοματοποιημένες δυνατότητες κλιμάκωσης, οι οποίες προσαρμόζουν δυναμικά τον αριθμό των instances με βάση την εισερχόμενη κίνηση. Το Knative μπορεί να εκθέτει μια υπηρεσία μέσω ενός HTTP URL, παρέχοντας λειτουργία αυτόματης κλιμάκωσης έως μηδέν (scale to zero), καθώς και κλιμάκωση βάσει του HTTP φορτίου. Έτσι, διασφαλίζεται η βέλτιστη αξιοποίηση των πόρων και η αποδοτικότητα κόστους.

Επιπλέον, το component αυτό διαχειρίζεται τις service revisions (αναθεωρήσεις υπηρεσιών) και προσφέρει δυνατότητες για seamless rollouts και rollbacks (απρόσκοπτης εκκίνησης και επαναφοράς). Παράλληλα, υποστηρίζει εξελιγμένες πολιτικές διαχείρισης της κίνησης, όπως traffic splitting (διαχωρισμού της κυκλοφορίας) και σταδιακά rollous, οι οποίες συμβάλλουν στην ομαλότητα των deployments και στην εξασφάλιση υψηλής διαθεσιμότητας.

Knative Serving Components

Όταν δημιουργείται μια Knative Serving Service (εν συντομία **ksvc**), ο Knative Serving Controller δημιουργεί επίσης ένα Configuration, ένα Revision και μία Route.

Το παρακάτω διάγραμμα απεικονίζει τα βασικά components της αρχιτεκτονικής του Knative Serving, μαζί με τις αλληλεπιδράσεις μεταξύ τους:



Σχημα 3.7: Η Αρχιτεκτονική του Knative Serving

Με βάση το διάγραμμα, η ακόλουθη περιγραφή παρουσιάζει τη λεπτομερή ροή του **κύκλου ζωής μιας serverless υπηρεσίας** που αναπτύσσεται με τη χρήση του Knative Serving:

Μια υπηρεσία δημιουργείται όταν ένας χρήστης ορίσει μια **Knative Service (ksvc)**, καθορίζοντας την επιθυμητή κατάσταση και διαμόρφωση για την εφαρμογή του. Για να καταγράψει τις καθορισμένες λεπτομέρειες, η Knative Service αλληλεπιδρά με το **Knative Configuration**. Το Configuration διατηρεί την επιθυμητή κατάσταση της ανάπτυξης και προσφέρει διαχωρισμό μεταξύ του κώδικα και της διαμόρφωσης. Κάθε αλλαγή στο Configuration έχει ως αποτέλεσμα τη δημιουργία μιας νέας Revision της υπηρεσίας (οδηγώντας σε μια νέα Kubernetes Deployment).

Το **Knative Revision** λειτουργεί όπως μια ετικέτα ή tag στον έλεγχο εκδόσεων και δεν μπορεί να τροποποιηθεί. Όλες οι Revisions καταγράφονται στο Ιστορικό Αναθεωρήσεων (Revision History), επιτρέποντας την επαναφορά της εφαρμογής σε οποιαδήποτε τελευταία γνωστή καλή διαμόρφωση.

Ένα άλλο βασικό στοιχείο του Serving, το **Knative Route**, ρυθμίζεται για τη διαχείριση της δρομολόγησης της κίνησης. Το Knative Route είναι, ουσιαστικά, η URL μέσω της οποίας μπορεί να γίνει πρόσβαση στην Knative Service. Χρησιμοποιώντας το component **Ingress**, το οποίο λειτουργεί ως σημείο εισόδου (entry point), κατευθύνει την εισερχόμενη HTTP κίνηση στην κατάλληλη Revision. Αυτό επιτρέπει στον προγραμματιστή να δρομολογήσει την κίνηση σύμφωνα με τις ανάγκες του, π.χ., για A/B testing ή Canary Deployments, καθώς μπορεί να ορίσει κανόνες δρομολόγησης που διανέμουν κατάλληλα την κίνηση σε διαφορετικές Revisions της Υπηρεσίας. Κάθε Revision οδηγεί σε μια συγκεκριμένη Deployment, διασφαλίζοντας ότι εκτελούνται οι σωστές εκδόσεις των εφαρμογών.

Το **Deployment** είναι υπεύθυνο για τη διαχείριση των απαραίτητων pods. Διασφαλίζει ότι ο κατάλληλος αριθμός pod replicas είναι σε λειτουργία και, όπως διαχειρίζεται από τον autoscaler, κλιμακώνεται προς τα πάνω ή κάτω με βάση την εισερχόμενη κίνηση και το φορτίο.

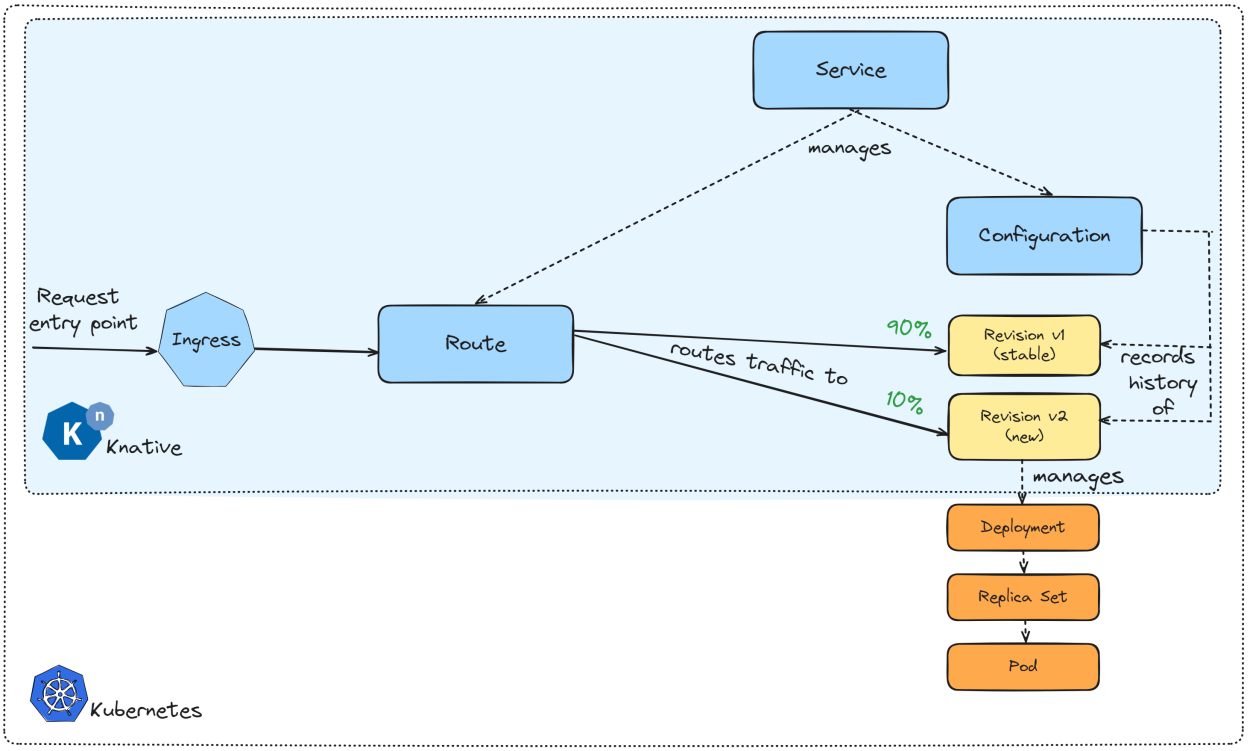
3.3.6 Πώς λειτουργεί το Knative Serving

Διαχείριση κίνησης και δρομολόγηση αιτημάτων

Όπως αναφέρθηκε προηγουμένως, μία από τις σημαντικότερες λειτουργίες του Knative Serving είναι η αποτελεσματική δρομολόγηση αιτημάτων, η οποία διασφαλίζει ότι τα εισερχόμενα αιτήματα HTTP κατευθύνονται στην κατάλληλη έκδοση της αντίστοιχης εφαρμογής. Τα αιτήματα εισέρχονται στο σύστημα μέσω της **Ingress Gateway** και στη συνέχεια δρομολογούνται με βάση τις καθορισμένες **Routes**. Το Serving υποστηρίζει πολλαπλές λύσεις ingress για τη διαχείριση της εισερχόμενης κίνησης.

Μέχρι πρόσφατα, το Knative βασιζόταν αποκλειστικά στο Istio για τη διαχείριση της εξωτερικής κίνησης του cluster και της επικοινωνίας μεταξύ των υπηρεσιών. Το **Istio** είναι ένα ισχυρό service mesh που παρέχει προηγμένη διαχείριση της κίνησης, καθώς και χαρακτηριστικά ασφάλειας και παρατηρησιμότητας. Ωστόσο, προσθέτει, μερικές φορές, ανεπιθύμητη πολυπλοκότητα και κατανάλωση πόρων στο cluster.

Ωστόσο, το στοιχείο δικτύωσης στο Knative είναι πλέον αποσυνδεδεμένο, επιτρέποντας τη χρήση εναλλακτικών υλοποιήσεων δικτύωσης. Μια πιο ελαφριά εναλλακτική λύση για το Istio, για παράδειγμα, είναι το Kourier. Το **Kourier**



Σχημα 3.8: Traffic Splitting στο Knative Serving

είναι ένα Ingress που έχει δημιουργηθεί ειδικά για το Knative Serving. Για την ανάπτυξή του απαιτείται μόνο ένας Envoy proxy και ένα control plane, το οποίο είναι ιδιαίτερα χρήσιμο σε περιπτώσεις όπου αναζητείται μια απλοποιημένη ανάπτυξη.

Σε κάθε περίπτωση, αυτή η ευελιξία του Knative να ενσωματώνεται με διαφορετικές λύσεις ingress επιτρέπει στους προγραμματιστές να προσαρμόζουν το σύστημα στις ειδικές ανάγκες και τους περιορισμούς των εφαρμογών τους.

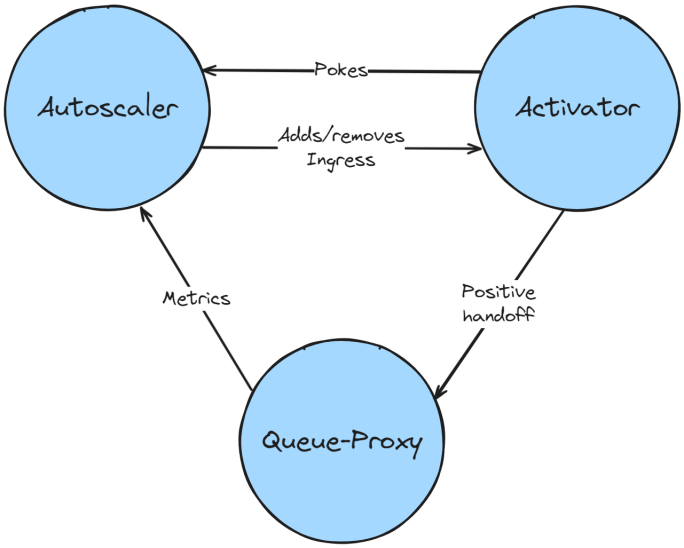
Έτσι, το Knative χρησιμοποιεί το Istio ή άλλους συμβατούς ingress controllers για την υλοποίηση του διαχωρισμού και της δρομολόγησης της κυκλοφορίας. Στην πράξη, αυτό σημαίνει ότι, αφού ο χρήστης καθορίσει τους διαχωρισμούς της κυκλοφορίας στη διαμόρφωση της διαδρομής της υπηρεσίας, το Knative τη μεταφράζει σε ένα αντικείμενο Istio **Virtual Service**, το οποίο ελέγχει τον τρόπο δρομολόγησης της κυκλοφορίας εντός του service mesh.

Οι διαχωρισμοί κυκλοφορίας που ορίζονται από τον χρήστη μπορούν να καθοριστούν σε ποσοστά. Για παράδειγμα, αν μια υπηρεσία έχει δύο αναθεωρήσεις, «v1» και «v2», ο χρήστης μπορεί να ρυθμίσει μια διαδρομή ώστε να κατευθύνει το 90

Αυτή η δυνατότητα επιτρέπει στο Knative να υποστηρίζει διάφορα προηγμένα σενάρια διαχείρισης της κυκλοφορίας, όπως: **Canary Releases** (δηλ. σταδιακή μετατόπιση της κυκλοφορίας από μια παλιά έκδοση σε μια νέα για την ελεγχόμενη δοκιμή αλλαγών), **A/B Testing** (δηλ. καθοδήγηση επιμέρους τμημάτων της κυκλοφορίας σε διαφορετικές αναθεωρήσεις για τη σύγκριση της απόδοσης ή τη συλλογή σχολίων από τους χρήστες) και **Blue-Green Deployments** (δηλ. λειτουργία δύο πανομοιότυπων περιβαλλόντων παραγωγής -μπλε και πράσινο- και εναλλαγή της κυκλοφορίας μεταξύ τους για ενημερώσεις χωρίς διακοπή λειτουργίας).

Knative Serving Autoscaling System

Το **Autoscaling** αποτελεί έναν κρίσιμο μηχανισμό στο Knative για τη δυναμική προσαρμογή του αριθμού των instances των υπηρεσιών σύμφωνα με τη μεταβαλλόμενη ζήτηση. Υπάρχουν **τρία βασικά συστατικά** που εμπλέκονται στον μηχανισμό αυτόματης κλιμάκωσης του Knative: ο **Knative Pod Autoscaler**, ο **Activator** και ο **Queue-Proxy**.



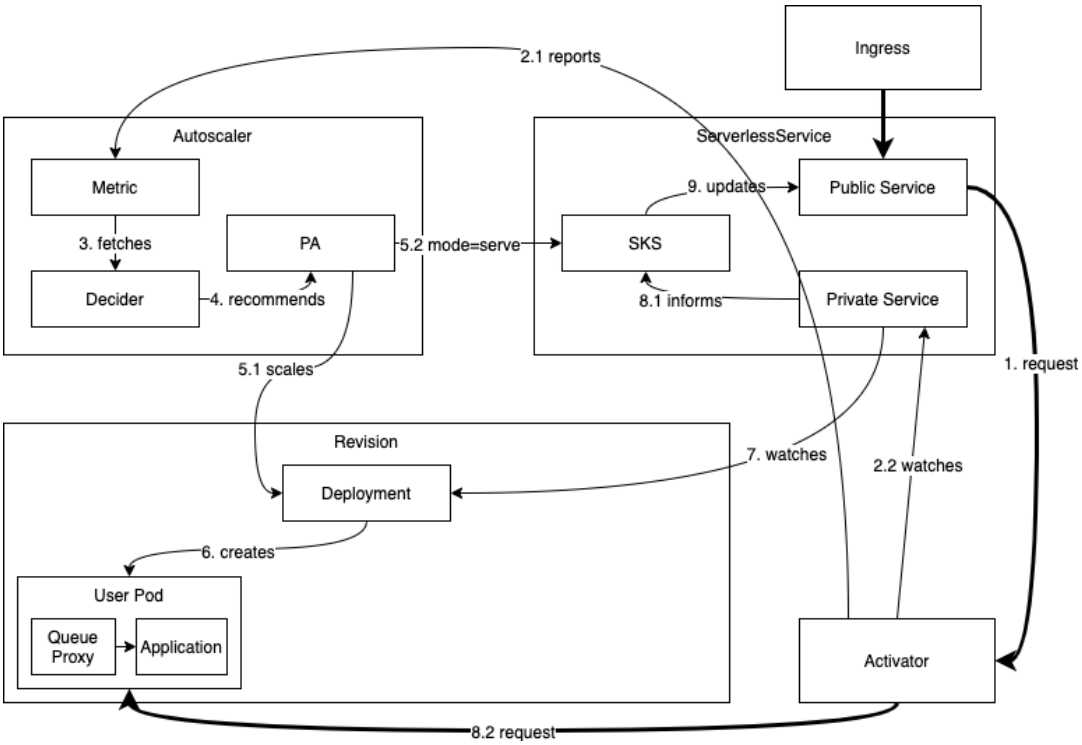
Σχημα 3.9: Τα τρία βασικά συστατικά Autoscaling Components στο Knative

Ο **Knative Pod Autoscaler (KPA)** είναι ένας αντιδραστικός (**reactive**) μηχανισμός αυτόματης κλιμάκωσης. Αυτό σημαίνει ότι λαμβάνει αποφάσεις κλιμάκωσης σε πραγματικό χρόνο με βάση τις τρέχουσες συνθήκες, αντί να προβλέπει τη μελλοντική ζήτηση (όπως θα έκανε ένας μηχανισμός αυτόματης κλιμάκωσης με προγνωστική δυνατότητα -**predictive**). Ο KPA λειτουργεί σε δύο κύριους άξονες: τον αριθμό των εισερχόμενων αιτημάτων και τον αριθμό των διαθέσιμων instances για τη διαχείριση αυτών των αιτημάτων.

Ο KPA αποτελεί την προεπιλεγμένη υλοποίηση του API του PodAutoscaler (PA). Το συστατικό **PodAutoscaler** διαχειρίζεται τον στόχο κλιμάκωσης, τη μετρική στην οποία βασίζεται η κλιμάκωση και οποιεσδήποτε άλλες παραμέτρους που είναι απαραίτητες για τη διαδικασία λήψης αποφάσεων ή τη συλλογή δεδομένων σχετικά με την αυτόματη κλιμάκωση. Υπάρχει επίσης ένας προσαρμογέας που μπορεί να υλοποιήσει το **Kubernetes HPA**.

Ένα ακόμη API που παρέχεται από το Knative Serving Autoscaling System και αξίζει να αναφερθεί είναι το **ServerlessServices (SKS)** API. Το SKS αποτελεί μια αφαίρεση πάνω από τα Services του Kubernetes και έχει σχεδιαστεί για να ελέγχει τη ροή δεδομένων. Σε περιπτώσεις όπου δεν υπάρχουν διαθέσιμα instances, εισάγεται ένας buffer στη ροή δεδομένων, ο οποίος ονομάζεται Activator και αφαιρείται όταν τα instances κλιμακώνονται σε αριθμό μεγαλύτερο του μηδενός.

Το SKS παρέχει δύο Kubernetes Services για κάθε αναθεώρηση: μία δημόσια και μία ιδιωτική υπηρεσία. Το API του SKS εναλλάσσει δυναμικά μεταξύ της ιδιωτικής και της δημόσιας υπηρεσίας, με βάση την κατάσταση κλιμάκωσης της υπηρεσίας.



Σχημα 3.10: Η ροή δεδομένων scale-from-zero του Knative
Source: Knative Official Documentation

Το **public service** χρησιμοποιείται για τη δρομολόγηση της εξωτερικής κίνησης που εισέρχεται στο Kubernetes cluster προς την Knative service, συνήθως μέσω του Ingress Gateway. Ο Activator αποτελεί μέρος της διαδρομής αυτής. Σε περιπτώσεις όπου η υπηρεσία έχει κλιμακωθεί στο μηδέν, το SKS διασφαλίζει ότι η κίνηση δρομολογείται μέσω του public service με την ενσωμάτωση του Activator. Αυτή η υπηρεσία διαδραματίζει κρίσιμο ρόλο, ειδικά όταν δεν υπάρχουν ενεργά pods, καθώς εξασφαλίζει ότι η εισερχόμενη κίνηση δεν χάνεται. Συνοπτικά, οι κύριες λειτουργίες του public service περιλαμβάνουν τη διαχείριση των ψυχρών εκκινήσεων και τη δρομολόγηση της εξωτερικής πρόσβασης στο cluster.

Όταν η υπηρεσία κλιμακώνεται, η κίνηση δρομολογείται μέσω του **private service** για λόγους αποδοτικότητας. Αυτό συνεπάγεται άμεση δρομολόγηση στα pods, παρακάμπτοντας τον Activator ώστε να μειωθεί η καθυστέρηση (latency). Η υπηρεσία αυτή είναι εσωτερική και χρησιμοποιείται αποκλειστικά για τη διαχείριση της επικοινωνίας εντός του cluster (intra-cluster), χωρίς να εκτίθεται απευθείας σε εξωτερική κίνηση.

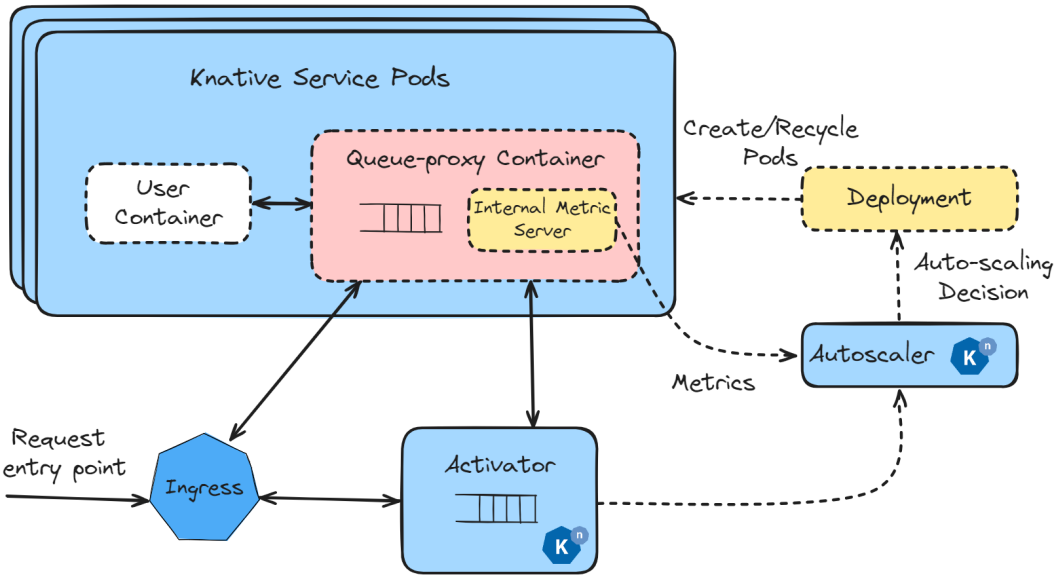
3.3.7 Κλιμάκωση από το μηδέν (cold start)

Κάθε υπηρεσία Knative είναι προσβάσιμη μέσω μιας μοναδικής διεύθυνσης URL. Το επίπεδο δρομολόγησης διασφαλίζει ότι τα αιτήματα προς αυτή τη διεύθυνση URL κατευθύνονται στην κατάλληλη αναθεώρηση (ή αναθεωρήσεις -**revision(s)**-) με βάση τη διαμόρφωση κατανομής της κυκλοφορίας. Μια σημαντική πτυχή της λειτουργίας μιας serverless υπηρεσίας είναι η δυνατότητα κλιμάκωσης στο μηδέν όταν δεν υπάρχει φορτίο. Αυτό συνεπάγεται ότι η υποκείμενη υποδομή πρέπει να διαχειρίζεται και τα σενάρια cold start. Σε αυτό το σημείο έρχεται ο Activator. Εάν μια αναθεώρηση έχει κλιμακωθεί στο μηδέν

λόγω έλλειψης κίνησης, ο Activator αναχαιτίζει τα εισερχόμενα αιτήματα και τα αποθηκεύει προσωρινά, ενώ η αναθεώρηση κλιμακώνεται ξανά, διασφαλίζοντας ότι τα αιτήματα δεν απορρίπτονται. Όταν η απαιτούμενη χωρητικότητα υπερβαίνει τη διαθέσιμη, ο Activator σηματοδοτεί στον autoscaler να δημιουργήσει επιπλέον pods. Μόλις τα νέα pods είναι έτοιμα, ο Activator προωθεί τα αποθηκευμένα αιτήματα, ξεκινώντας την υπηρεσία. Αυτή η διαδικασία είναι γνωστή ως cold start.

Διατήρηση των επιπέδων των instances

Ο Activator θα παραμείνει στη διαδρομή δεδομένων για να διαχειρίζεται την burst capacity και να διασφαλίζει ομαλές μεταβάσεις κλιμάκωσης. Ο autoscaler εξασφαλίζει ότι διατηρείται ο κατάλληλος αριθμός instances για να διαχειρίζεται το φορτίο καθώς αυξάνεται και σταθεροποιείται. Μόλις η κίνηση φτάσει σε μια σταθερή κατάσταση, ο Activator θα απομακρυνθεί από τη διαδρομή δεδομένων και η ευθύνη για την αποθήκευση των αιτημάτων θα μεταφερθεί στον Queue-Proxy.



Σχημα 3.11: Πώς λειτουργεί ο Queue-Proxy

3.3.8 Ο Queue-Proxy

Ένα σημαντικό στοιχείο του data plane του Knative Serving είναι ο queue-proxy. Κάθε φορά που ένας χρήστης αναπτύσσει μια εφαρμογή, εισάγεται ένα queue-proxy sidecar container. Η κύρια λειτουργία του είναι να παρέχει μια επιφανειακή προσωρινή αποθήκευση (buffer) για την ομαλή ροή των αιτημάτων προς την εφαρμογή, συμβάλλοντας έτσι στη μείωση των διακυμάνσεων στον χρόνο απόκρισης. Κάθε αίτημα που φτάνει στο user-container της εφαρμογής θα διαβιβαστεί πρώτα στον queue-proxy και μετά θα δρομολογηθεί στο user-container. Δευτερεύουσα λειτουργία του queue-proxy είναι η συλλογή μετρήσεων σχετικά με τον αριθμό των αιτημάτων και τους χρόνους επεξεργασίας, τα οποία χρησιμοποιούνται από τον autoscaler για τη λήψη τεκμηριωμένων αποφάσεων σχετικά με την κλιμάκωση. Επιπλέον, ο queue-proxy διασφαλίζει ότι δεν θα φτάσουν περισσότερα αιτήματα από το προκαθορισμένο όριο "concurrency"

της εφαρμογής ταυτόχρονα, τοποθετώντας τα υπόλοιπα αιτήματα σε τοπική ουρά αναμονής.

Η κατάσταση πανικού (Panic Mode)

Ο autoscaler μπορεί να εισέλθει σε μια κατάσταση που ονομάζεται «κατάσταση πανικού» όταν υπάρχει μια ξαφνική αύξηση της κίνησης, η οποία απαιτεί άμεση και σημαντική αύξηση του αριθμού των instances. Σε αυτή την κατάσταση, ο autoscaler δίνει προτεραιότητα στην ταχεία κλιμάκωση για να ανταποκριθεί στην αυξημένη ζήτηση, παρακάμπτοντας τους συνήθεις ελέγχους σταθεροποίησης που θα επιβράδυναν τη διαδικασία κλιμάκωσης υπό κανονικές συνθήκες.

Ο αλγόριθμος αυτόματης κλιμάκωσης

Η συλλογή μετρήσεων αποτελεί κρίσιμο μέρος της διαδικασίας αυτόματης κλιμάκωσης και περιλαμβάνει την περιοδική συλλογή δεδομένων από τον Queue-Proxy και άλλες περιπτώσεις σε τακτά χρονικά διαστήματα (συνήθως κάθε δύο δευτερόλεπτα). Οι συλλεγμένες μετρήσεις συγκεντρώνονται σε «κάδους» (buckets), καθώς είναι απαραίτητος ο υπολογισμός του μέσου αριθμού των ταυτόχρονων αιτημάτων ανά instance.

Ο στόχος του αλγορίθμου autoscaler είναι η διατήρηση ενός επιθυμητού ποσοστού χρήσης (utilization ratio). Αυτός ο λόγος υπολογίζεται ως ο λόγος των ταυτόχρονων αιτημάτων προς τον μέγιστο αριθμό ταυτόχρονων αιτημάτων ανά instance. Για παράδειγμα, εάν ο μέγιστος αριθμός ταυτόχρονων αιτημάτων είναι 100 και η στοχευόμενη χρήση είναι 70

Ο αλγόριθμος υπολογίζει τον μέσο αριθμό ταυτόχρονων αιτημάτων σε ένα συγκεκριμένο χρονικό παράθυρο (window) και στη συνέχεια διαιρεί αυτήν την τιμή με τον στοχευόμενο αριθμό ταυτόχρονων αιτημάτων ανά instance. Με αυτόν τον τρόπο καθορίζεται ο αριθμός των επιθυμητών instances.

Σε κατάσταση σταθερότητας (stable mode), ο αριθμός των instances μπορεί να αυξάνεται ή να μειώνεται ανάλογα με τις διακυμάνσεις της κίνησης. Ωστόσο, σε κατάσταση πανικού (panic mode), προκειμένου να αντιμετωπιστεί αποτελεσματικά μια αιφνίδια αύξηση της ζήτησης, ο αριθμός των instances μπορεί μόνο να αυξηθεί.

Η συμπεριφορά του autoscaler στο Knative μπορεί να προσαρμοστεί μέσω ρυθμίσεων παραμετροποίησης (configuration settings), οι οποίες μπορούν να εφαρμοστούν είτε παγκοσμίως από τους διαχειριστές (operators) είτε σε επίπεδο υπηρεσίας από τους προγραμματιστές. Αυτές οι παραμετροποιήσεις μπορούν να οριστούν μέσω εγγραφών Kubernetes ConfigMap ή μέσω annotations στα αρχεία παραμετροποίησης της υπηρεσίας.

KPA Vs. HPA

Horizontal Pod Autoscaler

Ο Horizontal Pod Autoscaler (HPA) είναι ένας εγγενής αυτοδιαβαθμιστής του Kubernetes, ο οποίος προσαρμόζει τον αριθμό των αντιγράφων (replicas) των pods με βάση παρατηρούμενες μετρήσεις, όπως η χρήση της CPU, η χρήση μνήμης, ή προσαρμοσμένες μετρήσεις. Συλλέγει περιοδικά μετρήσεις και καθορίζει πότε θα πραγματοποιηθεί αύξηση ή μείωση της κλίμακας, με βάση προκαθορισμένα όρια.

Σε αντίθεση με το **KPA**, το **HPA** μπορεί να θεωρηθεί πιο προδραστικό, καθώς προβλέπει τις ανάγκες σε πόρους βάσει των παρατηρούμενων τάσεων των μετρήσεων. Το **KPA**, από την άλλη πλευρά, έχει έναν **αντιδραστικό χαρακτήρα**. Το HPA λειτουργεί με ένα διάστημα δειγματοληψίας (το προεπιλεγμένο είναι 15 δευτερόλεπτα) και ένα παράθυρο σταθεροποίησης, που μπορεί να εισάγει μικρές καθυστερήσεις όταν προκύπτουν γρήγορες αλλαγές στο φορτίο. Λαμβάνοντας υπόψη αυτό το γεγονός, ίσως να μην είναι τόσο κατάλληλο για serverless εφαρμογές όπου απαιτείται σχεδόν άμεση κλιμάκωση για να αντιμετωπιστούν ξαφνικές αυξήσεις στην κίνηση. Το HPA επίσης δεν θεωρείται ιδανικό για φορτία εργασίας με γνώμονα τα συμβάντα (event-driven workloads), καθώς η δύναμή του έγκειται στην κλιμάκωση με βάση τους πόρους και όχι με βάση εξωτερικά συμβάντα. Το HPA αποδίδει καλύτερα σε μακροχρόνιες, stateful εφαρμογές, όπου η συνεπής διαχείριση πόρων είναι πιο σημαντική από την κλιμάκωση που βασίζεται σε συμβάντα. Η σταδιακή προσέγγιση κλιμάκωσης του HPA, σε συνδυασμό με το παράθυρο σταθεροποίησης, αποτρέπει μεγάλες αλλαγές στην κατανομή των πόρων. Αυτό είναι ιδανικό σε περιπτώσεις όπου η ομαλή κλιμάκωση και η σταθερότητα αποτελούν προτεραιότητα για το σύστημα.

Ο Πίνακας 3.1 συνοψίζει τις βασικές διαφορές μεταξύ των δύο Autoscaler.

Παράμετρος	KPA (Knative Pod Autoscaler)	HPA (Horizontal Pod Autoscaler)
Παράγοντες Ενεργοποίησης Κλιμάκωσης	• Ενεργοποιείται από την κυκλοφορία αιτημάτων HTTP και τη ζήτηση σε πραγματικό χρόνο • Επικεντρώνεται στην ταυτόχρονη εξυπηρέτηση και το φορτίο αιτημάτων	• Ενεργοποιείται από την χρήση CPU ή προσαρμοσμένες μετρήσεις • Κλιμακώνεται βάσει μετρήσεων κατανάλωσης πόρων
Ενσωμάτωση/Περίπτωση Χρήσης	• Σχεδιασμένος για serverless εφαρμογές εντός του Knative • Εξειδικευμένος για αιτήματα HTTP και event-driven workloads	• Γενικής χρήσης autoscaler για φορτία εργασίας Kubernetes • Κατάλληλος για οποιαδήποτε εφαρμογή με σχετικές μετρήσεις
Κλιμάκωση σε Μηδενικά Instances	• Υποστηρίζει κλιμάκωση σε μηδενικές περιπτώσεις • Ιδανικό για περιστασιακή κίνηση σε serverless σενάρια	• Διατηρεί συνήθως τουλάχιστον ένα instance • Δεν σχεδιάστηκε για κλιμάκωση σε μηδενικά instances εξ ορισμού
Απόκριση Χαμηλής Καθυστέρησης	• Βελτιστοποιημένο για χαμηλή καθυστέρηση, γρήγορη κλιμάκωση • Άμεση προσαρμογή σε αιφνίδια αύξηση κίνησης	• Πιο αργή απόκριση λόγω του διαστήματος δειγματοληψίας και του παραθύρου σταθεροποίησης • Ενδέχεται να υπάρχουν καθυστερήσεις στην κλιμάκωση προς τα πάνω
Λεπτομερής Κλιμάκωση	• Λεπτομερής κλιμάκωση βάσει επιπέδων ταυτόχρονων αιτημάτων • Ακριβής έλεγχος κατανομής πόρων	• Ευρύτερη κλιμάκωση βάσει συνολικών μετρήσεων πόρων • Λιγότερο ακριβής για workloads με απότομη αύξηση ζήτησης
Υποστήριξη Μετρήσεων	• Επικεντρώνεται κυρίως στην ταυτόχρονη εξυπηρέτηση αιτημάτων HTTP • Ιδανικό για web και API workloads	• Υποστηρίζει ευρύ φάσμα μετρήσεων (CPU, μνήμη, προσαρμοσμένες μετρήσεις) • Ευέλικτη επιλογή μετρήσεων για διάφορες περιπτώσεις χρήσης
Καταλληλότητα για Μακροχρόνιες Εφαρμογές	• Κατάλληλο για βραχύβιες, stateless εφαρμογές • Αποτελεσματικό για δυναμικά, bursty workloads	• Ιδανικό για μακροχρόνιες, stateful εφαρμογές • Παρέχει σταθερή διαχείριση πόρων
Πολυπλοκότητα Ρύθμισης	• Απαιτεί συγκεκριμένη ρύθμιση για ταυτόχρονες αιτήσεις και μοτίβα αιτημάτων • Πιο εξειδικευμένη και λεπτομερής ρύθμιση	• Ευκολότερη ρύθμιση για κλιμάκωση βάσει τυπικών πόρων • Απλή ρύθμιση για κοινές περιπτώσεις χρήσης

3.3.9 Knative Eventing

Μια Εισαγωγή στο Knative Eventing

Το Serving παρέχει στους προγραμματιστές δυναμική αυτόματη κλιμάκωση με βάση το φορτίο HTTP, επιτρέποντας ακόμα και την κλιμάκωση των pods στο μηδέν. Ωστόσο, υπάρχουν περιπτώσεις όπου οι υπηρεσίες πρέπει να κλιμακώνονται με βάση άλλες πηγές εκτός του HTTP. Σε αυτό το σημείο εισέρχεται το δεύτερο βασικό συστατικό της Knative, το Eventing.

Το Knative Eventing επεκτείνει τις προαναφερθείσες δυνατότητες του Knative, επιτρέποντας στις εφαρμογές να ενεργοποιούνται και να ανταποκρίνονται σε συμβάντα από διάφορους τύπους πηγών. Το Eventing είναι το συστατικό του Knative που καθιστά δυνατή τη δημιουργία αρχιτεκτονικών με γνώμονα τα συμβάντα. Σε ένα τέτοιο περιβάλλον, οι υπηρεσίες μπορούν να ενεργοποιηθούν από συμβάντα, όπως μηνύματα από συστήματα ανταλλαγής μηνυμάτων, αλλαγές σε βάσεις δεδομένων ή προσαρμοσμένα συμβάντα από άλλες εφαρμογές. Για παράδειγμα, μηνύματα που φτάνουν σε μια ουρά RabbitMQ ή ένα θέμα Apache Kafka θα μπορούσαν να προκαλέσουν την κλιμάκωση ή τη μείωση μιας υπηρεσίας.

Η προσέγγιση που βασίζεται στα συμβάντα παρέχει επίσης τα μέσα για την ενορχήστρωση σύνθετων ροών εργασίας, όπου κάθε βήμα ενεργοποιείται από το συμβάν του προηγούμενου. Επιπλέον, το Knative Eventing διευκολύνει την ασύγχρονη επεξεργασία, δίνοντας τη δυνατότητα στις υπηρεσίες να χειρίζονται συμβάντα με αποσυνδεδεμένο τρόπο.

Τα τρία κύρια μοτίβα του Knative Eventing

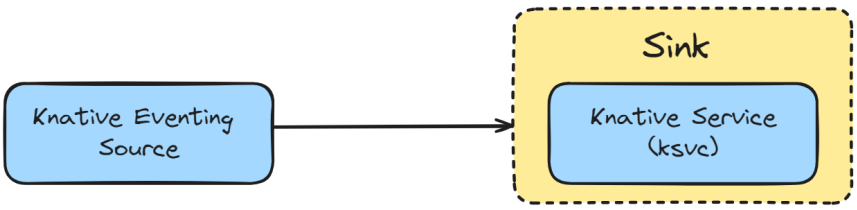
Το Knative Eventing μπορεί να δομηθεί γύρω από τρία βασικά μοτίβα:

- Source to Sink
- Channels και Subscriptions
- Brokers και Triggers

Source to Sink

Οι **Πηγές-Sources** αποτελούν τους κύριους παραγωγούς συμβάντων. Ο ρόλος μιας Πηγής στο πλαίσιο του Eventing είναι να μετατρέπει συμβάντα από εξωτερικά συστήματα σε μια μορφή που το Knative Eventing μπορεί να κατανοήσει και να επεξεργαστεί. Ουσιαστικά, αποτελεί μια αφαίρεση που ενσωματώνει τη λογική για την παραγωγή και προώθηση συμβάντων στο σύστημα Knative Eventing.

Από την άλλη πλευρά, οι **Υποδοχές-Sinks** είναι πόροι που μπορούν να λάβουν διευθύνσιμα εισερχόμενα συμβάντα από άλλους πόρους.

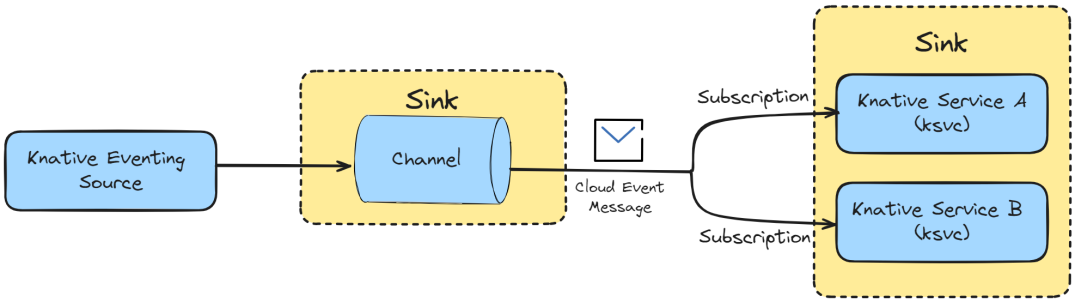


Σχημα 3.12: Το μοτίβο Source to Sink

Στο πρότυπο συμβάντων Source to Sink, το Sink είναι αυστηρά μια Knative Service (ksvc). Τα Channels και Brokers μπορούν επίσης να χρησιμοποιηθούν ως Sinks, αλλά αυτό μεταφράζεται σε διαφορετικά eventing patterns. Το pattern Source to Sink είναι το απλούστερο στο Knative Eventing. Η ευθύνη για την παράδοση των μηνυμάτων ανήκει στο Source, το οποίο δεν περιμένει κάποια απόκριση από την consuming service. Σε αυτό το pattern, δεν υπάρχει επίσης ούτε queueing ούτε filtering κατά τη διαδικασία της παράδοσης.

Channels and Subscriptions

Ένα **Channel** είναι ένα ειδικό είδος Sink. Ένα Channel λειτουργεί ως διεπαφή μεταξύ ενός Source συμβάντων και των Consumers συμβάντων. Ο ρόλος αυτού του component είναι να αποθηκεύει τα εισερχόμενα δεδομένα συμβάντων και να τα διανέμει στους Subscribers.



Σχημα 3.13: The Channel and Subscription Eventing Pattern

Κάθε Channel μπορεί να έχει έναν ή περισσότερους Subscribers με τη μορφή Sink Services. Κάποιος μπορεί να το θεωρήσει ως μια επέκταση του προηγούμενου eventing pattern, που επιτρέπει την υποστήριξη πολλαπλών Sinks και την επίμονη αποθήκευση των μηνυμάτων. Ένα Channel μπορεί να συνδεθεί με διαφορετικά backends για την προμήθεια του συμβάντος, και υπάρχουν τρεις τύποι καναλιών στο Knative:

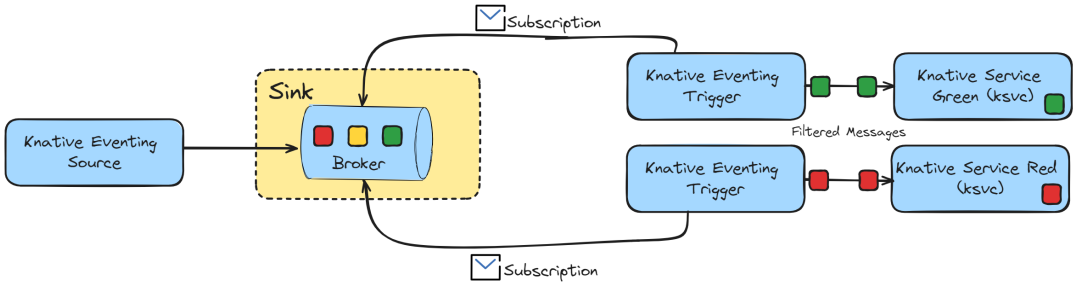
- In-Memory Channel
- Apache Kafka Channel
- Google Cloud Platform Pub-Sub Channel

Το άλλο σημαντικό στοιχείο αυτού του eventing pattern είναι οι Subscriptions. Οι **Subscriptions** είναι υπεύθυνες για τη σύνδεση ενός Channel με τις services που καταναλώνουν τα συμβάντα και επιθυμούν να λαμβάνουν τα μηνύματα που αποθηκεύει το Channel. Εάν μια υπηρεσία πρέπει να καταναλώνει μηνύματα (events) από ένα συγκεκριμένο Channel, τότε πρέπει να εγγραφεί σε αυτό. Όλες οι services που είναι subscribed στο ίδιο Channel θα λαμβάνουν τα ίδια μηνύματα από το event Source. Σημαντικό είναι να σημειωθεί ότι σε αυτό το eventing pattern, δεν υπάρχει filtering των μηνυμάτων που περνούν μέσα από το Channel.

Σε αντίθεση με το pattern Source and Sink, τα Channels διασφαλίζουν αξιόπιστη παράδοση συμβάντων και αποσυνδέουν τα event Sources από τους event Consumers.

Αυτή η ρύθμιση αντικατοπτρίζει στενά το παραδοσιακό μοντέλο Pub/Sub, όπου οι Publishers στέλνουν μηνύματα σε έναν broker (Channel) και οι Subscribers λαμβάνουν αυτά τα μηνύματα μέσω εγγραφής σε σχετικά topics ή channels.

Brokers and Triggers



Σχημα 3.14: The Broker and Trigger Eventing Pattern

Το τρίτο και τελευταίο eventing pattern που υποστηρίζεται από το Knative εμπλουτίζει περαιτέρω τις δυνατότητες του eventing framework της πλατφόρμας, καθώς επιτρέπει και το filtering των events.

Ένας **Broker** είναι ένας custom resource του Knative που καθορίζει ένα event mesh και συλλέγει μια ομάδα events. Πρόκειται ουσιαστικά για έναν μηχανισμό δρομολόγησης των events που μπορεί να δέχεται συμβάντα από πολλές διαφορετικές Sources και να τα κατευθύνει στους κατάλληλους Consumers. Για την επίτευξη φιλτραρίσματος των συμβάντων που περνούν μέσω του Broker, χρειάζεται να οριστούν Triggers. Ένα **Trigger** χρησιμοποιείται για να φιλτράρει και να προωθεί τα events από τον Broker στις Knative Services που είναι προορισμένες να τα καταναλώσουν. Η λειτουργία του είναι ως εξής: το Knative Eventing δημιουργεί αυτόματα ένα Knative Eventing Channel για κάθε Broker. Το Trigger εγγράφεται στον Broker και εφαρμόζει ένα φίλτρο στα μηνύματα που λαμβάνονται από αυτόν.

Τι είναι τα CloudEvents; Στις αρχιτεκτονικές με βάση τα συμβάντα, κάθε υπηρεσία μπορεί να παράγει και/ή να καταναλώνει συμβάντα σε διαφορετική μορφή, μια ασυνέπεια που μπορεί να αποτελέσει σημαντική πρόκληση για τους προγραμματιστές, καθώς θα πρέπει να αναπτύξουν νέα λογική χειρισμού για κάθε διαφορετική πηγή συμβάντων. Συνεπώς, υπάρχει ανάγκη για τυποποίηση της μορφής των συμβάντων, ώστε η διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων να είναι εύκολα επιτεύξιμη, ανεξάρτητα από το υποκείμενο τεχνολογικό υπόβαθρο.

Ορισμός των CloudEvents

Τα CloudEvents είναι μια προδιαγραφή για την περιγραφή των δεδομένων συμβάντων με έναν κοινό τρόπο, που δημιουργήθηκε για να καλύψει την ανάγκη τυποποίησης. Παρέχει SDKs για διάφορες γλώσσες προγραμματισμού, τα οποία μπορούν να χρησιμοποιηθούν για τη δημιουργία δρομολογητών συμβάντων, συστημάτων ιχνηλάτησης και άλλων εργαλείων, βοηθώντας στην αντιμετώπιση αυτού του ζητήματος.

Η Knative υιοθετεί την προδιαγραφή CloudEvents.

Η προδιαγραφή των CloudEvents ορίζει ένα σύνολο από υποχρεωτικά και προαιρετικά χαρακτηριστικά που εξασφαλίζουν την ολοκληρωμένη περιγραφή των συμβάντων:

Τι δεν περιλαμβάνεται στα CloudEvents

Τα CloudEvents δεν περιλαμβάνουν χαρακτηριστικά ασφαλείας, όπως κρυπτογράφηση από άκρο σε άκρο ή ψηφιακές υπογραφές. Αυτό οφείλεται στο

Field	Description
id	Μοναδικός αναγνωριστικός κωδικός για το συμβάν.
source	Το πλαίσιο στο οποίο συνέβη το συμβάν.
specversion	Η έκδοση της προδιαγραφής CloudEvents.
type	Ο τύπος του συμβάντος που συνδέεται με την προέλευσή του.
datacontenttype	Ο τύπος περιεχομένου των δεδομένων του συμβάντος.
data	Το ωφέλιμο φορτίο του συμβάντος, το οποίο είναι προαιρετικό και μπορεί να ποικίλλει σε μορφή.

Πινακας 3.2: Κύρια Πεδία και οι Περιγραφές τους στο CloudEvents

γεγονός ότι το πρότυπο εστιάζει στις βασικές λειτουργίες και την έγκαιρη παράδοση. Η προδιαγραφή αποφεύγει επίσης τον ορισμό χαρακτηριστικών σε επίπεδο μεταφοράς, όπως "to" και "reply-to". Η απουσία του χαρακτηριστικού "to" συμφωνεί με τη φύση των event-driven αρχιτεκτονικών, όπου τα συμβάντα δεν απευθύνονται σε συγκεκριμένους καταναλωτές αλλά δημοσιεύονται ώστε οι συνδρομητές να επιλέξουν. Αντίστοιχα, η απουσία του χαρακτηριστικού "reply-to" είναι σκόπιμη, καθώς τα συμβάντα σε αυτό το μοντέλο είναι γενικά μονόδρομα και ενδέχεται να παραδίδονται σε πολλούς αποδέκτες.

3.3.10 RabbitMQ ως διαμεσολαβητής μηνυμάτων για Knative Eventing

Το RabbitMQ είναι ένα **λογισμικό ουράς μηνυμάτων**, γνωστό και ως μεσίτης μηνυμάτων ή διαχειριστής ουράς. Με απλά λόγια, πρόκειται για ένα λογισμικό που χρησιμοποιείται για τον ορισμό ουρών αναμονής, στις οποίες συνδέονται εφαρμογές για τη μεταφορά μηνυμάτων.

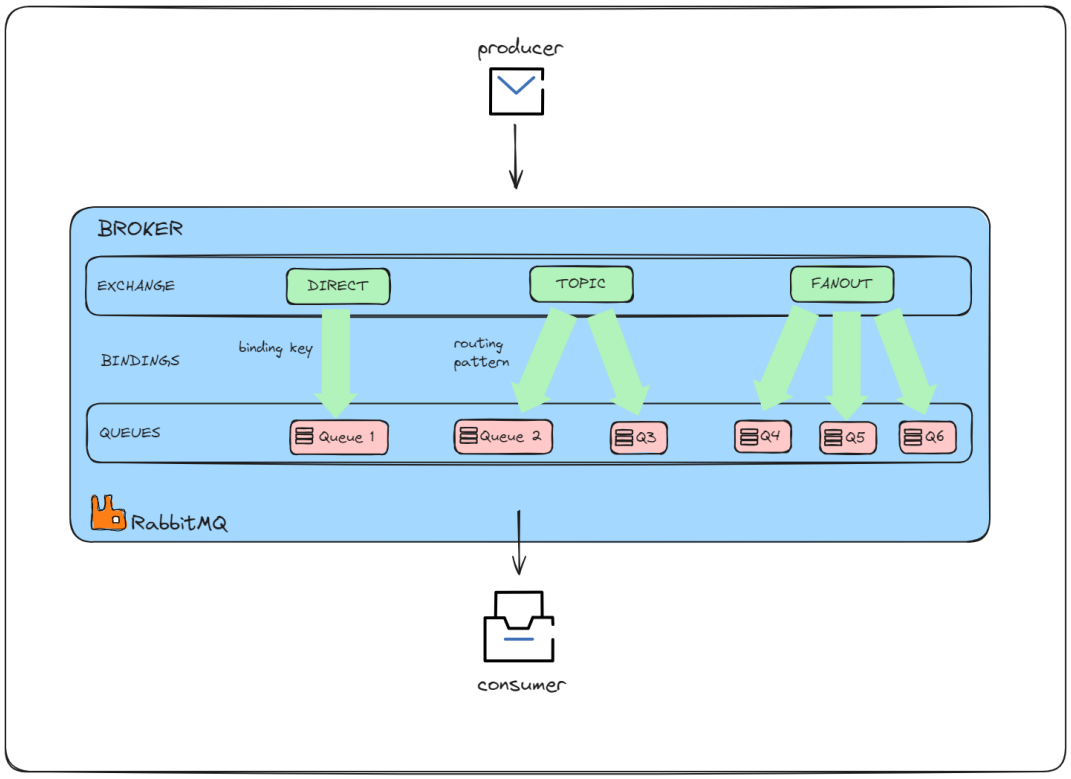
Το πρωτόκολλο που χρησιμοποιείται είναι το **Advanced Message Queueing Protocol (AMQP)**, το οποίο προτιμάται συχνά για αξιόπιστη παράδοση μηνυμάτων και είναι συμβατό με εφαρμογές cloud-native, όπως το Knative.

Στο RabbitMQ, τα μηνύματα δεν δημοσιεύονται απευθείας σε μια ουρά. Αντίθετα, οι πηγές πρέπει πρώτα να τα στείλουν σε ένα **exchange**. Τα **exchanges** είναι υπεύθυνα για τη δρομολόγηση των μηνυμάτων σε διάφορες ουρές, και αυτό επιτυγχάνεται μέσω των bindings και των routing keys.

Ένα **binding** είναι ένας σύνδεσμος που συνδέει μια ουρά με ένα exchange. Ένα **routing key** είναι ένα χαρακτηριστικό του μηνύματος που επιτρέπει στα exchanges να αποφασίσουν πώς θα δρομολογήσουν το μήνυμα στις ουρές.

Κατά τη δημιουργία ενός exchange, μπορεί να οριστεί ως **durable**, **temporary**, ή **auto-delete**. Εάν ένα exchange δηλωθεί ως **durable**, θα παραμείνει διαθέσιμο ακόμη και αν ο διακομιστής αποτύχει και επανεκκινήσει, εκτός εάν διαγραφεί ρητά. Τα **temporary** exchanges επιβιώνουν μέχρι να τερματιστεί το RabbitMQ, ενώ τα **auto-deleted** exchanges αφαιρούνται όταν αποδεσμευτεί το τελευταίο αντικείμενο που συνδέεται με αυτά.

Τα μηνύματα στο RabbitMQ μπορούν να δρομολογηθούν σε ουρές ανάλογα με τον τύπο του exchange. Για παράδειγμα, ένα **Direct** exchange δρομολογεί μηνύματα σε ουρές με βάση την ακριβή αντιστοιχία μεταξύ του routing key του μηνύματος και του binding key. Ένα **Fanout** exchange μεταδίδει μηνύματα σε όλες τις ουρές που είναι συνδεδεμένες με αυτό (ανεξαρτήτως routing key). Ένα **Topic** exchange δρομολογεί μηνύματα με βάση την αντιστοίχιση προτύπων μεταξύ του routing key και του binding key. Τέλος, το **Headers** exchange, αντί για



Σχημα 3.15: Οι διαφορετικοί τύποι RabbitMQ exchanges

routing keys, δρομολογεί τα μηνύματα με βάση τα χαρακτηριστικά της επικεφαλίδας.

Το πρωτόκολλο AMQP

Το πρωτόκολλο AMQP έχει σχεδιαστεί για να διευκολύνει την αξιόπιστη επικοινωνία μεταξύ διαφορετικών στοιχείων σε κατανεμημένα δίκτυα. Προσδιορίζει μια τυποποιημένη μορφή μηνύματος που περιλαμβάνει μια επικεφαλίδα με πληροφορίες δρομολόγησης και παράδοσης και ένα σώμα που περιέχει τα πραγματικά δεδομένα.

Το πρωτόκολλο εγγυάται ότι τα μηνύματα δεν χάνονται ακόμη και όταν ο παραλήπτης δεν είναι διαθέσιμος για την παραλαβή τους. Σε αυτές τις περιπτώσεις, το AMQP επιτρέπει την προσωρινή αποθήκευση των μηνυμάτων στην ουρά. Επιπρόσθετα, το πρωτόκολλο διασφαλίζει την αξιοπιστία της παράδοσης μηνυμάτων μέσω επιβεβαιώσεων παραλαβής και επιμονής των μηνυμάτων στην ουρά, ακόμα και σε περιπτώσεις επανεκκίνησης του broker ή του συστήματος.

Εκτός από τη δρομολόγηση μέσω των exchanges, το AMQP ορίζει επίσης μεθόδους όπως η αυθεντικοποίηση και η κρυπτογράφηση για να διασφαλίσει την ασφάλεια των μηνυμάτων κατά τη μεταφορά. Επιπλέον, περιλαμβάνει μηχανισμούς για τη ρύθμιση του ρυθμού αποστολής και επεξεργασίας των μηνυμάτων, αποτρέποντας την υπερφόρτωση τόσο του broker όσο και των καταναλωτών.

Ενσωμάτωση με τα στοιχεία Knative Eventing

Όπως αναφέρθηκε προηγουμένως, στο πλαίσιο του Knative Eventing, η πλατφόρμα βασίζεται σε έναν broker για την κάλυψη των πιο σύνθετων αναγκών

στη διαχείριση και δρομολόγηση συμβάντων μεταξύ των παραγωγών και καταναλωτών. Το RabbitMQ ενσωματώνεται ομαλά με στοιχεία του Knative Eventing, όπως οι Brokers, Triggers, Subscriptions, και Channels, επιτρέποντας στους προγραμματιστές να διαμορφώνουν ροές συμβάντων που εξυπηρετούν τις συγκεκριμένες ανάγκες των εφαρμογών τους. Με την επιλογή του RabbitMQ ως broker, το Knative αξιοποιεί τους διάφορους τύπους exchanges και στρατηγικών δρομολόγησης που προσφέρει η πλατφόρμα, διασφαλίζοντας παράλληλα αξιόπιστη παράδοση και διατήρηση των μηνυμάτων. Αυτό επιτρέπει την ανάπτυξη πολύπλοκων event-driven workflows σε serverless περιβάλλοντα. Η χρήση του πρωτοκόλλου AMQP στο RabbitMQ συμβάλλει επίσης στη συνεπή και αξιόπιστη παράδοση μηνυμάτων σε διαφορετικά περιβάλλοντα cloud, υποστηρίζοντας τον στόχο του Knative για ανεξαρτησία πλατφόρμας και εκτέλεση σε διαφορετικούς cloud providers.

3.3.11 Το Redis ως In-Memory Data Store για Serverless Εφαρμογές

Το Redis είναι ένα in-memory data store που παρουσιάζει εντυπωσιακή ταχύτητα και προσαρμοστικότητα σε εφαρμογές caching. Ο όρος "in-memory" σημαίνει ότι τα δεδομένα αποθηκεύονται στη μνήμη RAM ενός υπολογιστή και όχι σε κάποιον δίσκο, καθιστώντας το Redis ιδανική επιλογή για χρήση σε serverless περιβάλλοντα όπου η γρήγορη ανάκτηση και ενημέρωση των δεδομένων είναι κρίσιμη. Το Redis προτιμάται συχνά σε αρχιτεκτονικές microservices λόγω των υψηλών επιδόσεων που παρέχει στη μετάδοση δεδομένων. Καθώς οι serverless εφαρμογές βασίζονται σε αρχιτεκτονική microservices, το Redis αποτελεί επίσης κατάλληλη επιλογή για stateless υπηρεσίες. Επιπλέον, το Redis ενσωματώνεται με διάφορα serverless frameworks (συμπεριλαμβανομένου του Knative), προσφέροντας ένα αποδοτικό backend για αποθήκευση κατάστασης, διαχείριση συνεδριών ή caching δεδομένων.

Κεφάλαιο 4

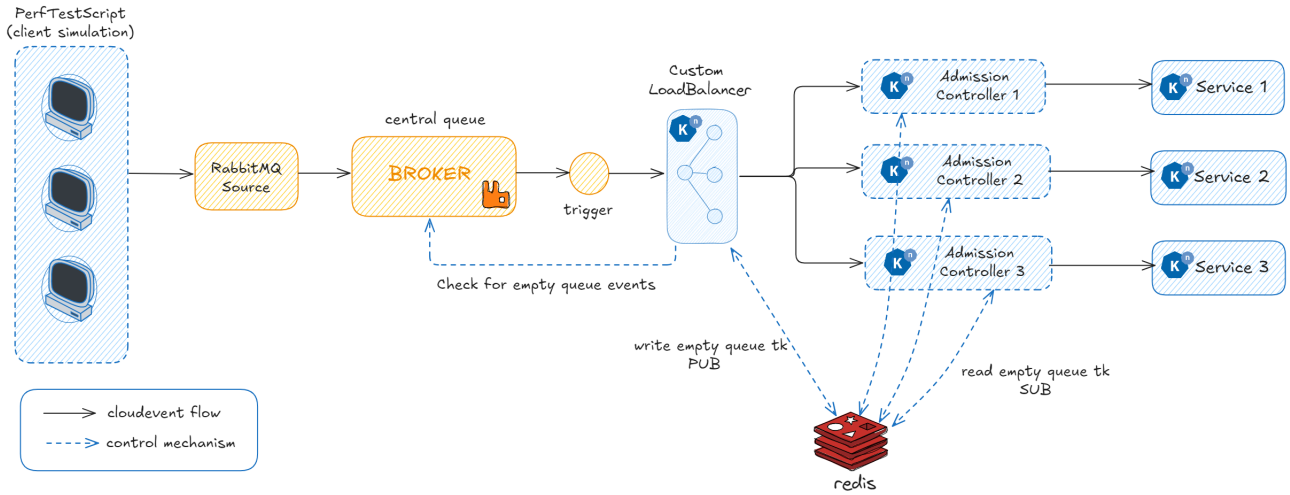
Σχεδίαση Συστήματος

4.1 Ανασκόπηση της αρχιτεκτονικής του συστήματος

Αυτό το κεφάλαιο προσφέρει μια λεπτομερή επισκόπηση του σχεδιασμού και της αρχιτεκτονικής του προσαρμοσμένου συστήματος εξισορρόπησης φορτίου που αναπτύχθηκε για serverless περιβάλλοντα με χρήση του Knative. Η προτεινόμενη μέθοδος εξισορρόπησης φορτίου προσαρμόζεται στη δυναμική φύση των serverless workloads, καθώς και στις δυνατότητες αυτόματης κλιμάκωσης της πλατφόρμας Knative. Η λύση σχεδιάστηκε για να εξασφαλίζει ότι η κίνηση δρομολογείται αποτελεσματικά στις διαθέσιμες υπηρεσίες, ενώ προσαρμόζεται σε πραγματικό χρόνο στις αλλαγές του φορτίου συστήματος, των καταστάσεων ουρών και των γεγονότων κλιμάκωσης. Οι ακόλουθες ενότητες αναλύουν τα εμπλεκόμενα στοιχεία, τις μεταξύ τους αλληλεπιδράσεις και τους υποκείμενους μηχανισμούς που χρησιμοποιούνται για την αντιμετώπιση των προκλήσεων της εξισορρόπησης φόρτου σε ένα δυναμικό, κατανεμημένο σύστημα.

4.1.1 Ανασκόπηση συστήματος υψηλού επιπέδου

Το ακόλουθο διάγραμμα παρουσιάζει μια οπτική αναπαράσταση της αρχιτεκτονικής του συστήματος, καταδεικνύοντας τα κύρια εμπλεκόμενα στοιχεία καθώς και τις μεταξύ τους αλληλεπιδράσεις.



Σχημα 4.1: Διάγραμμα αρχιτεκτονικής συστήματος

4.1.2 Ανάλυση των Εξαρτημάτων

Προσομοίωση πελάτη (PerfTestScript)

Για την προσομοίωση της κίνησης πελατών και τον έλεγχο της εγκατάστασης υπό φορτίο, δημιουργούμε αιτήματα πελατών με το εργαλείο RabbitMQ PerfTest, ένα αναγνωρισμένο εργαλείο δοκιμών επιδόσεων που επιτρέπει τη συγκριτική αξιολόγηση συστημάτων βασισμένων στο RabbitMQ υπό διάφορες συνθήκες φορτίου.

RabbitMQ Source

Το στοιχείο RabbitMQ Source επιτρέπει στο Knative να επεξεργάζεται εισερχόμενα συμβάντα (αιτήματα). Μετατρέπει τα εισερχόμενα μηνύματα σε μια μορφή με την οποία το Knative μπορεί να συνεργαστεί, τα CloudEvents που αναφέρθηκαν προηγουμένως. Αυτό το στοιχείο διασφαλίζει την ανεξαρτησία των Knative υπηρεσιών (των καταναλωτών) από τους πελάτες παραγωγής μηνυμάτων.

Η Κεντρική Ουρά (RabbitMQ)

Η κεντρική ουρά του συστήματος υλοποιείται ως ένας Knative RabbitMQ Broker που αποτελείται από ένα ανταλλακτήριο και μια ουρά. Το ανταλλακτήριο λαμβάνει CloudEvents (δημιουργημένα από αιτήματα πελατών) και τα δρομολογεί στην κατάλληλη ουρά, όπου αποθηκεύονται έως ότου επεξεργαστούν από τις υπηρεσίες. Η ουρά λειτουργεί ως buffer, γεγονός που είναι ζωτικής σημασίας σε περιπτώσεις υψηλού φορτίου, καθώς αποσυνδέει τη δημιουργία αιτημάτων πελατών από την επεξεργασία τους από τις υπηρεσίες. Τα αιτήματα περιμένουν στην ουρά μέχρι να βρεθεί μια διαθέσιμη υπηρεσία στην οποία θα δρομολογηθούν για επεξεργασία.

Ο Προσαρμοσμένος Εξισορροπιστής Φορτίου

Αυτό είναι ένα βασικό στοιχείο του συστήματος. Γραμμένος σε Go και αναπτυγμένος ως Knative Service, ο εξισορροπιστής φορτίου λειτουργεί ως διαμεσολαβητής μεταξύ της κεντρικής ουράς και των υπηρεσιών κατανάλωσης. Υλοποιεί έναν σταθμισμένο τυχαίο αλγόριθμο δρομολόγησης, όπου τα βάρη καθορίζονται από τα τρέχοντα ποσοστά αποδοχής κάθε υπηρεσίας. Παράλληλα, λειτουργεί ως ελεγκτής, πραγματοποιώντας συνεχή παρακολούθηση της κεντρικής ουράς, αναμένοντας ένα συμβάν κενής ουράς και δημοσιεύοντας το σχετικό timestamp στο Redis μέσω του μηχανισμού Pub/Sub.

Ελεγκτές Εισδοχής (Admission Controllers)

Κάθε υπηρεσία κατανάλωσης συνοδεύεται από έναν ελεγκτή εισδοχής. Αυτοί οι ελεγκτές αναπτύσσονται επίσης ως Knative Services και λειτουργούν σε συνεργασία με τον Custom Load Balancer, όντας συνδρομητές των συμβάντων κενής ουράς που δημοσιεύονται στο Redis. Λειτουργούν ως ρυθμιστές ταχύτητας και ενημερώνουν τον ρυθμό εισδοχής στις αντίστοιχες υπηρεσίες τους με χρήση του αλγορίθμου AIMD.

Redis

Το Redis χρησιμοποιείται για τη διαχείριση της κατάστασης του συστήματος σε πραγματικό χρόνο, ειδικά για τα συμβάντα κενής ουράς. Όταν ο εξισορροπιστής φορτίου εντοπίζει ότι η ουρά του RabbitMQ είναι κενή, το Redis δημοσιεύει το timestamp του σχετικού συμβάντος. Οι συνδρομητές του συμβάντος αυτού είναι οι ελεγκτές εισδοχής, οι οποίοι, μόλις ειδοποιηθούν, προσαρμόζουν ανάλογα τον ρυθμό εισδοχής των αντίστοιχων υπηρεσιών τους.

4.2 Σχεδιασμός Προσαρμοσμένου Εξισορροπιστή Φορτίου

Ο προσαρμοσμένος εξισορροπιστής φορτίου, ο οποίος θα αναλυθεί σε αυτή την ενότητα, αποτελεί κρίσιμο στοιχείο του κατανεμημένου συστήματος χωρίς διακομιστή. Είναι υπεύθυνος για την αποτελεσματική δρομολόγηση των προσομοιωμένων αιτημάτων πελατών προς τις καταναλώτριες υπηρεσίες Knative, με βάση δυναμικούς παράγοντες. Οι παραδοσιακοί μηχανισμοί εξισορρόπησης φορτίου που παρέχονται από το RabbitMQ και το Istio, αν και λειτουργικοί, δεν επαρκούν για τις δυναμικές και προσαρμοστικές απαιτήσεις που είναι απαραίτητες για τη διαχείριση υψηλών και κυμαινόμενων όγκων κυκλοφορίας. Αυτή η ενότητα παρέχει μια λεπτομερή επισκόπηση του προσαρμοσμένου αλγορίθμου εξισορρόπησης φορτίου και της ενσωμάτωσής του με τα Redis και RabbitMQ.

Ο αλγόριθμος ακολουθεί μια δυναμική προσέγγιση δρομολόγησης βασισμένη στη θεωρία προσαρμοστικού ελέγχου, και ειδικότερα, χρησιμοποιεί μια τροποποιημένη εκδοχή του αλγορίθμου Additive Increase Multiplicative Decrease (AIMD). Ο βασικός στόχος είναι να επιτρέπει στο σύστημα να προσαρμόζεται σε κυμαινόμενα φορτία μέσω της ρύθμισης των ποσοστών αποδοχής που αποδίδονται σε κάθε υπηρεσία. Ο εξισορροπιστής φορτίου παρακολουθεί συνεχώς την ουρά του RabbitMQ και προσαρμόζει ανάλογα τα βάρη των υπηρεσιών. Η ενσωμάτωση με το Redis επιτρέπει την επικοινωνία κρίσιμων συμβάντων σε πραγματικό χρόνο, διασφαλίζοντας ότι οι ελεγκτές εισδοχής μπορούν να προσαρμόζουν δυναμικά τη ροή της κίνησης των υπηρεσιών ανάλογα με το τρέχον φορτίο.

Ο εξισορροπιστής φορτίου ενσωματώνει επίσης έναν μηχανισμό συλλογής μετρήσεων, πραγματοποιώντας αιτήσεις στο Prometheus και στο Kubernetes API για την ανάκτηση δεικτών επιδόσεων σε πραγματικό χρόνο, όπως τρέχοντα ποσοστά αποδοχής, μήκη ουρών και αριθμό ενεργών υπηρεσιών για κάθε Knative service. Αυτό επιτρέπει την προσαρμοστική δρομολόγηση με βάση την τρέχουσα κατάσταση κλιμάκωσης του συστήματος.

Τέλος, ο εξισορροπιστής συνεργάζεται με τους ελεγκτές εισδοχής που βρίσκονται μπροστά από κάθε υπηρεσία για τον έλεγχο της ροής κίνησης στις αντίστοιχες υπηρεσίες τους. Οι ελεγκτές αυτοί λειτουργούν ως περιοριστές ρυθμού και ενημερώνονται συνεχώς.

4.2.1 Σταθμισμένη Τυχαία Δρομολόγηση με Βάση τους Ρυθμούς Εισδοχής

Ο αλγόριθμος δρομολόγησης που χρησιμοποιείται από τον εξισορροπιστή φορτίου είναι ένας σταθμισμένος τυχαίος αλγόριθμος που προσαρμόζει δυναμικά την κατανομή της κυκλοφορίας με βάση τα τρέχοντα ποσοστά αποδοχής

που ανακτώνται από το Redis. Αυτή η προσέγγιση διασφαλίζει ότι η κυκλοφορία κατανέμεται δίκαια, ανάλογα με τη χωρητικότητα πραγματικού χρόνου κάθε υπηρεσίας, η οποία καθορίζεται από το ποσοστό αποδοχής της.

Κάθε υπηρεσία λαμβάνει ένα βάρος που αντιστοιχεί στο ποσοστό αποδοχής της, το οποίο αντικατοπτρίζει το σχετικό μερίδιο αιτημάτων που θα πρέπει να λάβει. Τα βάρη όλων των υπηρεσιών συνενώνονται σε έναν πίνακα αθροίσματος προθέματος, που είναι εγγενώς ταξινομημένος, επιτρέποντας μια επιλογή με βάση δυαδική αναζήτηση. Όταν καταφτάνει ένα νέο αίτημα, αντιπροσωπεύεται από έναν τυχαία παραγόμενο αριθμό. Στη συνέχεια, η δυαδική αναζήτηση καθορίζει το εύρος βαρών στο οποίο εμπίπτει ο αριθμός, με αποτέλεσμα να καθορίζεται ο προορισμός δρομολόγησης σε χρονική πολυπλοκότητα $O(\log n)$, όπου n είναι ο αριθμός των διαθέσιμων υπηρεσιών κατανάλωσης.

Περιγραφή Αλγορίθμου

Έστω $W = \{w_0, w_1, w_2, \ldots, w_{n-1}\}$ το σύνολο των βαρών που αποδίδονται στις περιπτώσεις υπηρεσιών, όπου κάθε βάρος w_i αντιστοιχεί στο ποσοστό αποδοχής της υπηρεσίας i. Ο στόχος είναι η τυχαία επιλογή μιας υπηρεσίας με την πιθανότητα επιλογής της υπηρεσίας i να είναι ανάλογη του βάρους της w_i. Ο αλγόριθμος λειτουργεί ως εξής:

1. **Υπολογισμός Αθροίσματος Προθέματος**: Δημιουργείται ένας πίνακας $S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$, όπου
$$s_i = \sum_{j=0}^{i} w_j.$$
Κάθε s_i αναπαριστά το συνολικό άθροισμα των βαρών από την πρώτη έως την i-οστή υπηρεσία.

2. **Παραγωγή Τυχαίας Τιμής**: Δημιουργείται ένας τυχαίος ακέραιος r μεταξύ 0 και του συνολικού αθροίσματος των βαρών, δηλαδή
$$r \in [0, s_{n-1}).$$

3. **Δυαδική Αναζήτηση για την Υπηρεσία-Στόχο**: Εκτελείται δυαδική αναζήτηση στον πίνακα αθροίσματος προθέματος S για τον εντοπισμό του μικρότερου δείκτη i ώστε
$$s_i \geq r.$$
Αυτό το βήμα εξασφαλίζει ότι η πιθανότητα επιλογής μιας υπηρεσίας είναι ανάλογη του βάρους της.

   Δεδομένου ότι τα αθροίσματα προθέματος αντιπροσωπεύουν αθροιστικά βάρη, το βήμα αυτό διασφαλίζει ότι υπηρεσίες με υψηλότερα βάρη διαθέτουν μεγαλύτερο εύρος τιμών r, για τις οποίες επιλέγονται. Συνεπώς, η πιθανότητα επιλογής της υπηρεσίας i είναι
$$P(\text{επιλογή υπηρεσίας } i) = \frac{w_i}{\sum_{j=0}^{n-1} w_j}.$$

4. **Επιλογή Υπηρεσίας**: Η επιλεγμένη υπηρεσία αντιστοιχεί στον δείκτη i που προέκυψε από τη δυαδική αναζήτηση. Αυτή η υπηρεσία χρησιμοποιείται ως προορισμός για τη δρομολόγηση του εισερχόμενου αιτήματος.

4.2.2 Αναπροσαρμογή Ρυθμού Πρόσβασης με Βάση τον Αλγόριθμο AIMD

Για να προσαρμόζει δυναμικά τα βάρη που χρησιμοποιούνται στις αποφάσεις δρομολόγησης, ο εξισορροπιστής φορτίου βασίζεται στον **αλγόριθμο Additive Increase Multiplicative Decrease (AIMD)**, που καθορίζει τις ενημερώσεις του ρυθμού αποδοχής για κάθε υπηρεσία. Αυτή η προσέγγιση επιτρέπει στις υπηρεσίες να ανταποκρίνονται σε μεταβαλλόμενες συνθήκες φόρτου, αυξάνοντας ή μειώνοντας τα βάρη τους με βάση την παρατηρούμενη συμπεριφορά της ουράς, ανάλογα με τη χωρητικότητα και την απόδοσή τους.

Ο αλγόριθμος **AIMD** έχει σχεδιαστεί για να ρυθμίζει τους ρυθμούς αποδοχής των υπηρεσιών με βάση τις συνθήκες πραγματικού χρόνου και έχει δύο βασικές φάσεις, οι οποίες αντικατοπτρίζονται στον ακόλουθο τύπο, ο οποίος ορίζει τον **ρυθμό αποδοχής προς τον i κόμβο επεξεργασίας**:

$$u_i(t) = \beta_i u_i(t_k) + \alpha_i R_i(t - t_k), \quad i = 1, \ldots, n,$$

όπου:

1. $u_i(t)$ είναι το τρέχον βάρος (ποσοστό αποδοχής) της υπηρεσίας i τη χρονική στιγμή t.
2. β_i είναι ο πολλαπλασιαστικός συντελεστής μείωσης για την υπηρεσία i.
3. $u_i(t_k)$ είναι το βάρος της υπηρεσίας i κατά την τελευταία χρονική στιγμή ενημέρωσης t_k.
4. α_i είναι ο προσθετικός συντελεστής αύξησης για την υπηρεσία i.
5. R_i είναι ο αριθμός των ενεργών αντιγράφων για την περίπτωση υπηρεσίας i^{th}.
6. $(t - t_k)$ είναι ο χρόνος που παρήλθε από την τελευταία ενημέρωση.

Κατά τη διάρκεια της φάσης **Additive Increase**, το βάρος κάθε υπηρεσίας αυξάνεται γραμμικά και ο ρυθμός αύξησης ελέγχεται από την παράμετρο **alpha**. Αυτή η φάση αύξησης του βάρους έχει σκοπό να αντανακλά την εμπιστοσύνη του συστήματος ότι η υπηρεσία μπορεί να χειριστεί περισσότερες εισερχόμενες αιτήσεις. Η φάση συνεχίζεται μέχρι να ανιχνευθεί ένα γεγονός κενής ουράς από τον εξισορροπιστή φορτίου. Ο ρυθμός αύξησης επηρεάζεται επίσης άμεσα από τον **αριθμό των ενεργών αντιγράφων** για κάθε υπηρεσία. Με αυτόν τον τρόπο διασφαλίζεται ότι οι υπηρεσίες με περισσότερα αντίγραφα, και άρα μεγαλύτερη χωρητικότητα, μπορούν να κλιμακώσουν πιο επιθετικά τους ρυθμούς αποδοχής. Με άλλα λόγια, οι υπηρεσίες με υψηλότερο α και περισσότερα αντίγραφα θα δουν ταχύτερο ρυθμό αύξησης του βάρους που θα τους επιτρέψει να αναλάβουν περισσότερο φορτίο, έτσι ώστε το φορτίο να εξισορροπείται βέλτιστα με τρόπο που να αντικατοπτρίζει την ικανότητα επεξεργασίας σε πραγματικό χρόνο των υπηρεσιών που καταναλώνουν.

Η φάση **Multiplicative Decrease** ενεργοποιείται από ένα συμβάν κενής ουράς. Με αυτόν τον τρόπο, οι ρυθμοί αποδοχής των υπηρεσιών μπορούν να ευθυγραμμιστούν με την τρέχουσα ζήτηση. Ο ρυθμός μείωσης ελέγχεται από την παράμετρο **beta**, η οποία είναι συνήθως μικρότερη από 1, οδηγώντας σε ταχεία μείωση των ρυθμών εισδοχής. Αυτό σημαίνει ότι τα νέα βάρη είναι ένα κλάσμα των παλαιών. Η λογική πίσω από αυτή την προσέγγιση είναι ότι αν η κεντρική

ουρά αδειάσει, υπονοείται ότι καμία από τις υπηρεσίες δεν είναι υπερφορτωμένη με νέες εργασίες, οπότε όλες οι υπηρεσίες έχουν την πολυτέλεια να μειώσουν τα ποσοστά αποδοχής τους ώστε να αντικατοπτρίζουν το τρέχον χαμηλότερο επίπεδο ζήτησης. Αυτή η μείωση γίνεται με ομοφωνία σε όλες τις υπηρεσίες, πράγμα που σημαίνει ότι το βάρος της καθεμιάς μειώνεται κατά ένα κλάσμα, επιτρέποντας στο σύστημα να σταθεροποιηθεί και να είναι έτοιμο για την επόμενη έκρηξη κίνησης χωρίς να δεσμεύεται υπερβολικά σε αδρανείς πόρους.

4.2.3 Συλλογή και παρακολούθηση μετρήσεων

Είναι πλέον σαφές ότι, προκειμένου ο εξισορροπητής φορτίου να λειτουργεί σωστά, πρέπει να συλλέγει διάφορες μετρικές (όπως τα ποσοστά αποδοχής, το μήκος ουράς και τον αριθμό των ενεργών αντιγράφων). Χωρίς την ακριβή και έγκαιρη συλλογή αυτών των τιμών το σύστημα δεν θα ήταν σε θέση να προσαρμοστεί δυναμικά στους κυμαινόμενους φόρτους εργασίας. Προκειμένου να αντιμετωπίσουμε τις συγκεκριμένες απαιτήσεις του συστήματος, χρησιμοποιήσαμε διαφορετικούς μηχανισμούς για τη συλλογή και τον διαμοιρασμό των απαραίτητων μετρήσεων, με βάση την πηγή τους και τη φύση των δεδομένων.

Μετρικές βασισμένες στο Redis: Ρυθμοί εισόδου και συμβάντα κενής ουράς

Το Redis χρησιμοποιείται ως ο κύριος αποθηκευτικός χώρος για τις καταστάσεις του συστήματος σε πραγματικό χρόνο, όπως τα ποσοστά εισδοχής και τα συμβάντα κενής ουράς. Σε κάθε υπηρεσία εκχωρείται ένα βάρος ή ποσοστό αποδοχής, το οποίο καθορίζει το ποσοστό της κίνησης που πρέπει να δρομολογηθεί σε αυτήν. Αυτό το ποσοστό αποδοχής ενημερώνεται δυναμικά χρησιμοποιώντας τον αλγόριθμο AIMD (Additive Increase, Multiplicative Decrease).

1. **Admission Rates**: Τα ποσοστά εισόδου για κάθε υπηρεσία αποθηκεύονται στο Redis κάτω από συγκεκριμένα κλειδιά, με κάθε κλειδί να σχετίζεται με μια συγκεκριμένη υπηρεσία. Ο εξισορροπιστής φορτίου ανακτά αυτές τις τιμές για να λάβει αποφάσεις δρομολόγησης. Το ποσοστό αποδοχής για μια υπηρεσία ενημερώνεται σύμφωνα με τον αλγόριθμο AIMD με βάση τα δεδομένα μήκους ουράς σε πραγματικό χρόνο και τον αριθμό των ενεργών αντιγράφων. Στη συνέχεια δημοσιεύεται στο Redis, ώστε οι ελεγκτές αποδοχής να μπορούν να ενημερωθούν για την αλλαγή και να εφαρμόσουν τα νέα όρια ρυθμού στις αντίστοιχες υπηρεσίες που καταναλώνουν.

2. **Empty Queue Events**: Τα συμβάντα κενής ουράς σηματοδοτούν όταν η κεντρική ουρά RabbitMQ δεν έχει εκκρεμή μηνύματα. Όταν συμβαίνει ένα συμβάν κενής ουράς, το Redis ενημερώνεται με τη χρονοσφραγίδα του συμβάντος. Αυτή η χρονοσφραγίδα είναι κρίσιμη, καθώς ενεργοποιεί τη φάση Multiplicative Decrease του αλγορίθμου AIMD, η οποία μειώνει τους ρυθμούς αποδοχής όλων των υπηρεσιών. Το Redis δημοσιεύει αυτή τη χρονοσφραγίδα χρησιμοποιώντας τον μηχανισμό Pub/Sub και οι ελεγκτές αποδοχής των υπηρεσιών ενεργούν βάσει αυτού του συμβάντος για να προσαρμόσουν τους αντίστοιχους ρυθμούς αποδοχής.

Polling RabbitMQ: Παρακολούθηση Μήκους Ουράς

Η κεντρική ουρά στην αρχιτεκτονική εξισορρόπησης φορτίου υλοποιείται με τη χρήση του RabbitMQ. Ο εξισορροπητής φορτίου πραγματοποιεί συνεχώς

polls στο RabbitMQ για να ελέγχει το μήκος της ουράς, το οποίο αποτελεί κρίσιμη μετρική για την κατανόηση του τρέχοντος φορτίου του συστήματος. Ένα μεγάλο μήκος ουράς υποδηλώνει ένα ανεκτέλεστο υπόλοιπο μη επεξεργασμένων αιτήσεων, ενώ μια άδεια ουρά υποδηλώνει ότι το σύστημα είναι αδρανές ή ότι όλες οι αιτήσεις έχουν επεξεργαστεί.

Ο εξισορροπιστής φορτίου παρακολουθεί την ουρά χρησιμοποιώντας τη συνάρτηση `QueueInspect` από το API του RabbitMQ. Αυτή η συνάρτηση ανακτά τον αριθμό των εκκρεμών μηνυμάτων στην ουρά και οι πληροφορίες αυτές χρησιμοποιούνται για τη δυναμική προσαρμογή των ρυθμών αποδοχής των υπηρεσιών. Εάν η ουρά είναι άδεια, ενεργοποιεί το προαναφερθέν συμβάν empty queue, σηματοδοτώντας ότι το σύστημα είναι έτοιμο να μειώσει τους ρυθμούς αποδοχής μέσω της φάσης Multiplicative Decrease.

Kubernetes API: Replica Count Monitoring

Ο εξισορροπητής φορτίου χρησιμοποιεί το Kubernetes API για την παρακολούθηση του αριθμού ενεργών αντιγράφων κάθε υπηρεσίας, πληροφορία που είναι κρίσιμη για τον υπολογισμό των admission rates και των βάρων στο routing. Η συνάρτηση FetchReplicaNum ανακτά τον αριθμό των ενεργών pods για κάθε υπηρεσία, και αυτή η πληροφορία ενσωματώνεται στον αλγόριθμο AIMD. Ο αριθμός των αντιγράφων επηρεάζει όχι μόνο το admission rate αλλά και την κατανομή των αιτημάτων στις υπηρεσίες, εξισορροπώντας το φορτίο ανάλογα με τους διαθέσιμους πόρους.

4.3 Σχεδιασμός του Ελεγκτή Αποδοχής (Admission Controller)

Ο Ελεγκτής Αποδοχής (Admission Controller) αποτελεί βασικό στοιχείο του συστήματος, υπεύθυνο για την παρακολούθηση και τη ρύθμιση του ρυθμού αποδοχής αιτημάτων, ανάλογα με τις δυναμικές συνθήκες που επικρατούν. Ο σχεδιασμός του βασίζεται στην αρχή της ευελιξίας και της επεκτασιμότητας, αξιοποιώντας το Redis για την αποθήκευση και διανομή των δεδομένων αποδοχής σε πραγματικό χρόνο, καθώς και το Prometheus για τη συλλογή μετρικών.

4.3.1 Αρχιτεκτονική του Ελεγκτή Αποδοχής

Η αρχιτεκτονική του Ελεγκτή Αποδοχής αποτελείται από τα εξής υποσυστήματα:

- **Φόρτωση Παραμέτρων από το Περιβάλλον (config package)**: Ο Ελεγκτής Αποδοχής διαμορφώνεται δυναμικά μέσω μεταβλητών περιβάλλοντος. Συγκεκριμένα:
  - Το όνομα της υπηρεσίας (`ServiceName`).
  - Οι παράμετροι `Alpha` και `Beta`, οι οποίες καθορίζουν τον τρόπο υπολογισμού του ρυθμού αποδοχής.
  - Οι ρυθμίσεις για το Redis (`RedisURL` και `RedisPass`).

  Αυτές οι παράμετροι φορτώνονται κατά την εκκίνηση της υπηρεσίας και, σε περίπτωση που δεν οριστούν, χρησιμοποιούνται προκαθορισμένες τιμές.
- **Διαχείριση Ρυθμού Αποδοχής (controller package)**: Ο ρυθμός αποδοχής ελέγχεται από το `RateController`, ο οποίος παρέχει τις εξής δυνατότητες:

  - Χρήση του `sync.Mutex` για την ασφαλή ταυτόχρονη πρόσβαση από πολλαπλά νήματα.
  - Δυνατότητα ενημέρωσης του ρυθμού αποδοχής (`admissionRate`) και δημιουργία ενός rate limiter που περιορίζει τα αιτήματα σύμφωνα με τον ενημερωμένο ρυθμό.

  Το αρχικό `admissionRate` τίθεται -τυπικά- στο 1, και κάθε νέα ενημέρωση από το Redis οδηγεί σε αναπροσαρμογή της τιμής αυτής.

- **Συνδρομή για Ενημερώσεις Ρυθμού μέσω Redis (events package)**: Ο Ελεγκτής Αποδοχής παρακολουθεί σε πραγματικό χρόνο τα μηνύματα από το Redis μέσω του καναλιού `admission_rate:<ServiceName>`. Κάθε φορά που λαμβάνεται νέα τιμή για τον ρυθμό αποδοχής:
  - Γίνεται ανάλυση της νέας τιμής και εφαρμόζεται στον `RateController`.
  - Ενημερώνονται οι αντίστοιχες μετρικές στο Prometheus.
- **Παρακολούθηση και Ενημέρωση Μετρικών (metrics package)**: Το σύστημα χρησιμοποιεί το Prometheus για την παρακολούθηση του τρέχοντος ρυθμού αποδοχής μέσω του gauge μετρικού `admission_rate`. Η μετρική αυτή ενημερώνεται κάθε φορά που μεταβάλλεται ο ρυθμός αποδοχής, και είναι διαθέσιμη για παρακολούθηση μέσω HTTP στον πόρο `/metrics`.

4.3.2 Ροή Εργασίας του Ελεγκτή Αποδοχής

Η βασική ροή εργασίας του Ελεγκτή Αποδοχής περιγράφεται ως εξής:

1. Φορτώνει τις παραμέτρους από το περιβάλλον.
2. Αρχικοποιεί τον `RateController` με τις παραμέτρους `Alpha` και `Beta`.
3. Δημιουργεί συνδρομή στο Redis για την παρακολούθηση του καναλιού ενημέρωσης ρυθμού αποδοχής.
4. Καθώς λαμβάνει νέες ενημερώσεις, προσαρμόζει τον ρυθμό αποδοχής και ενημερώνει τις μετρικές στο Prometheus.

Ο σχεδιασμός αυτός εξασφαλίζει ότι ο Ελεγκτής Αποδοχής μπορεί να αντιδρά άμεσα σε μεταβαλλόμενες συνθήκες, προσαρμόζοντας τον ρυθμό αποδοχής αιτημάτων και παρέχοντας ακριβείς μετρικές για την παρακολούθηση της απόδοσής του.

4.4 Σχεδιασμός του Καταναλωτή Συμβάντων (Event Consumer)

Ο Καταναλωτής Συμβάντων (Event Consumer) έχει ως κύρια αρμοδιότητα την επεξεργασία δεδομένων εικόνας μέσω της τεχνολογίας *CloudEvents* και την εκτέλεση ανίχνευσης αντικειμένων με το μοντέλο *YOLOv3*. Η σχεδίαση του επικεντρώνεται στην ευελιξία και την ικανότητα κλιμάκωσης, επιτρέποντας την επεξεργασία αιτημάτων από πολλούς εργαζόμενους (*workers*) ταυτόχρονα, καθώς και την παρακολούθηση μετρικών με χρήση του *Prometheus*.

4.4.1 Αρχιτεκτονική του Καταναλωτή Συμβάντων

Η αρχιτεκτονική του Καταναλωτή Συμβάντων περιλαμβάνει τα ακόλουθα βασικά μέρη:

- **Φόρτωση Ρυθμίσεων από το Περιβάλλον (config package)**: Οι ρυθμίσεις του Καταναλωτή Συμβάντων ορίζονται μέσω μεταβλητών περιβάλλοντος, οι οποίες περιλαμβάνουν:
  - Το όνομα της υπηρεσίας (`ServiceName`),
  - Τον αριθμό των εργαζομένων (`NumWorkers`) που αναλαμβάνουν την επεξεργασία των αιτημάτων,
  - Το μέγεθος της ουράς (`RequestQueue`) που διατηρεί τα αιτήματα *CloudEvents*.

  Ο αριθμός των εργαζομένων και το μέγεθος της ουράς μπορούν να προσαρμοστούν, επιτρέποντας τη δημιουργία εργαζομένων με διαφορετικές δυνατότητες επεξεργασίας. Αυτό είναι ιδιαίτερα χρήσιμο για την εκτέλεση δοκιμών, καθώς μας επιτρέπει να δοκιμάσουμε και να αξιολογήσουμε τον αλγόριθμο δρομολόγησης (routing algorithm) υπό διαφορετικές συνθήκες φόρτου και απόδοσης. Η ευελιξία αυτή συμβάλλει στην καλύτερη κατανόηση της συμπεριφοράς του συστήματος κατά τη δρομολόγηση αιτημάτων σε υπηρεσίες με διαφορετικές δυνατότητες επεξεργασίας.

- **Μετρικές και Παρακολούθηση (metrics package)**: Ο Καταναλωτής Συμβάντων χρησιμοποιεί το *Prometheus* για την παρακολούθηση του αριθμού των αιτημάτων που βρίσκονται στην ουρά. Οι μετρικές αυτές αρχικοποιούνται και ενημερώνονται δυναμικά, προσφέροντας σημαντικές πληροφορίες σχετικά με την κατάσταση του συστήματος.

- **Επεξεργασία Εικόνων μέσω του YOLOv3 (processor package)**: Η ανίχνευση αντικειμένων στις εικόνες πραγματοποιείται από κάθε εργαζόμενο, ο οποίος λαμβάνει αιτήματα από την ουρά και εκτελεί την ανίχνευση χρησιμοποιώντας το *YOLOv3*. Κάθε εικόνα αποκωδικοποιείται από τη μορφή *base64*, ενώ ελέγχεται η εγκυρότητά της πριν την επεξεργασία. Τα αποτελέσματα της ανίχνευσης καταγράφονται στο αρχείο καταγραφής (log), με αναφορά στην κλάση του αντικειμένου, την ακρίβεια (confidence) και τα όρια του αντικειμένου (bounding box).

- **Υποδομή για CloudEvents (main package)**: Η εφαρμογή αξιοποιεί το *CloudEvents SDK* για τη λήψη αιτημάτων και την εκκίνηση του δέκτη (receiver). Οι εργαζόμενοι δημιουργούνται δυναμικά μέσω των *goroutines*, και κάθε εργαζόμενος διαχειρίζεται την επεξεργασία αιτημάτων από την ουρά. Παράλληλα, το σύστημα περιλαμβάνει μηχανισμούς καταγραφής (logging) για την παρακολούθηση και διαχείριση των αιτημάτων.

4.4.2 Ροή Εργασίας του Καταναλωτή Συμβάντων

Η βασική ροή εργασίας του Καταναλωτή Συμβάντων περιλαμβάνει τα εξής βήματα:

1. Φόρτωση των ρυθμίσεων από το περιβάλλον.
2. Εκκίνηση του εξυπηρετητή για τη διαχείριση των μετρικών και την παρακολούθηση αιτημάτων μέσω *Prometheus*.

3. Δημιουργία και εκκίνηση πολλαπλών εργαζομένων που επεξεργάζονται αιτήματα *CloudEvents*.

4. Καταχώρηση και αποκωδικοποίηση των αιτημάτων εικόνας και εκτέλεση ανίχνευσης αντικειμένων με το *YOLOv3*.

5. Ενημέρωση των μετρικών κατά την εξυπηρέτηση των αιτημάτων και καταγραφή των αποτελεσμάτων της ανίχνευσης.

Ο σχεδιασμός αυτός επιτρέπει την κλιμάκωση του συστήματος με τη συμμετοχή περισσότερων εργαζομένων, ενώ η παρακολούθηση των μετρικών σε πραγματικό χρόνο συμβάλλει στην άμεση αξιολόγηση της απόδοσης του συστήματος.

Κεφάλαιο 5

Δοκιμές Απόδοσης και Ανάλυση Συστήματος Υπό Μεταβαλλόμενο Φορτίο

5.1 Εισαγωγή

Η μελέτη της απόδοσης ενός συστήματος υπολογιστικής χωρίς διακομιστή (serverless), όπως το σύστημα που παρουσιάζεται σε αυτή τη διπλωματική εργασία, απαιτεί μια βαθιά ανάλυση των μηχανισμών που ενεργοποιούν τη δυναμική κλιμάκωση και τη διαχείριση των πόρων. Καθώς οι φόρτοι εργασίας αυξάνονται και μειώνονται απότομα, είναι κρίσιμο να κατανοηθεί πώς ένα σύστημα ανταποκρίνεται χωρίς την υποστήριξη αυτόματης κλιμάκωσης, αλλά και πώς βελτιστοποιείται όταν αυτοί οι μηχανισμοί τίθενται σε λειτουργία.

Ο σκοπός αυτού του κεφαλαίου είναι η λεπτομερής ανάλυση της συμπεριφοράς του συστήματος υπό διάφορες συνθήκες φόρτου. Αρχικά, διεξάγονται βασικές δοκιμές χωρίς ενεργοποίηση κλιμάκωσης, ώστε να εντοπιστούν οι δυνατότητες και οι περιορισμοί του συστήματος σε σταθερό και μεταβλητό φορτίο. Στη συνέχεια, πραγματοποιούνται πιο σύνθετες δοκιμές με τη χρήση μηχανισμών αυτόματης κλιμάκωσης, τόσο με βάση τον ρυθμό αιτήσεων (Knative KPA) όσο και την αξιοποίηση των πόρων (Knative HPA).

Αυτές οι δοκιμές δεν αποτελούν μόνο μια προσπάθεια να διαπιστωθεί το μέγιστο σταθερό φορτίο που μπορεί να διαχειριστεί το σύστημα χωρίς να αποτύχει, αλλά και να κατανοηθούν οι προκλήσεις που σχετίζονται με εκρηκτικά φορτία και αιχμές στην κίνηση. Ειδικότερα, η παρούσα ανάλυση εστιάζει σε μετρήσεις όπως η χρήση της CPU, η κατανάλωση μνήμης, οι χρόνοι απόκρισης, η καθυστέρηση αποστολής συμβάντων και η απόδοση (αιτήσεις ανά δευτερόλεπτο).

Με τη δομή αυτή, το κεφάλαιο αυτό θέτει τα θεμέλια για την εκτίμηση της αποδοτικότητας των αλγορίθμων και των τεχνικών διαχείρισης φορτίων που παρουσιάστηκαν νωρίτερα στο θεωρητικό πλαίσιο της εργασίας. Τα πειράματα αυτά προσφέρουν χρήσιμες πληροφορίες για τη συμπεριφορά του συστήματος υπό πραγματικές συνθήκες λειτουργίας, και επιπλέον, παρέχουν μια ολοκληρωμένη εικόνα της ανάγκης για αυτόματη κλιμάκωση και βελτιστοποίηση της απόδοσης σε εφαρμογές χωρίς διακομιστή.

5.2 Πειραματική Υποδομή και Προετοιμασία Συστήματος

Σε αυτή την ενότητα παρουσιάζουμε την υποδομή που υλοποιήθηκε για την εκτέλεση των δοκιμών απόδοσης και την ανάλυση του συστήματος υπό μεταβαλλόμενο φορτίο. Η υποδομή αυτή βασίστηκε στην πλατφόρμα Knative για την υποστήριξη serverless υπηρεσιών και περιλαμβάνει τα απαραίτητα εργαλεία για τη διαχείριση των πόρων, τη συλλογή μετρήσεων και τη δρομολόγηση μηνυμάτων σε περιβάλλοντα με υψηλές απαιτήσεις.

Για την ορθή αξιολόγηση της απόδοσης του συστήματος, εγκαταστάθηκαν και ρυθμίστηκαν υποσυστήματα όπως το Minikube για τη δημιουργία του Kubernetes cluster, το RabbitMQ για τη διαχείριση των ουρών μηνυμάτων και το Redis για την προσωρινή αποθήκευση. Ο σχεδιασμός της υποδομής επικεντρώθηκε στην ευκολία αναπαραγωγής των δοκιμών και στην αποτελεσματική χρήση πόρων, ώστε να διευκολύνεται η ανάπτυξη και οι δοκιμές σε τοπικά περιβάλλοντα ή μηχανήματα με περιορισμένες υπολογιστικές δυνατότητες.

5.3 Διαθέσιμος Κώδικας και Υλοποίηση

Η υλοποίηση του έργου είναι διαθέσιμη στο GitHub repository:

https://github.com/christidia/DiplomaThesis

Το repository περιλαμβάνει τον κώδικα για τις υπηρεσίες καταναλωτών (consumer services), τον custom load balancer, τους admission controllers, καθώς και τα αρχεία διαμόρφωσης για τα deployments και το script εγκατάστασης. Όλα τα απαραίτητα εργαλεία και scripts για την πλήρη εγκατάσταση και λειτουργία του συστήματος βρίσκονται εκεί, επιτρέποντας τη δοκιμή και την αναπαραγωγή του συστήματος σε διαφορετικά περιβάλλοντα.

5.4 Οδηγίες Εγκατάστασης και Ανάπτυξης

5.4.1 Ρύθμιση του Minikube

Το **Minikube** είναι συχνά προτιμητέο, καθώς απλοποιεί τη διαδικασία ρύθμισης του Kubernetes, επιτρέποντας επαναληπτική ανάπτυξη και γρήγορη δοκιμή διαφορετικών ρυθμίσεων. Επιπλέον, αποτελεί μια ελαφριά λύση που μπορεί να διαμορφωθεί ώστε να χρησιμοποιεί μόνο τους απαραίτητους πόρους, ιδανικό για εκτέλεση σε VM με περιορισμένη υπολογιστική ισχύ.

Προαπαιτούμενο για την εκτέλεση του Minikube είναι ένας **container manager**, όπως Docker. Το Minikube λειτουργεί κυρίως μέσω γραμμής εντολών, προσφέροντας τις απαραίτητες εντολές για τη δημιουργία και τη διαχείριση ενός τοπικού Kubernetes cluster. Ενσωματώνεται με το **kubectl**, το επίσημο εργαλείο γραμμής εντολών του Kubernetes, επιτρέποντας τη διαχείριση των φορτίων εργασίας και των πόρων του cluster.

5.4.2 Εγκατάσταση των Συνιστωσών Knative

Τα συστατικά του Knative Serving και Eventing εγκαθίστανται και αναπτύσσονται ξεχωριστά. Όπως αναφέρθηκε, τα συστατικά του Knative επεκτείνουν

το Kubernetes με τη χρήση των κατάλληλων **custom resources (CRDs)**. Για να αξιοποιηθούν όλες οι δυνατότητες που παρέχονται από το Knative Serving, απαιτείται ένα δίκτυο για τη διαχείριση της **εισερχόμενης κίνησης (ingress)**. Για το συγκεκριμένο project, επιλέχθηκε το **Istio** ως το δίκτυο.

5.4.3 Εγκατάσταση των Συνιστωσών RabbitMQ

Το RabbitMQ είναι ο επιλεγμένος διαμεσολαβητής μηνυμάτων. Παρέχει μια πλατφόρμα για την αποστολή και τη λήψη μηνυμάτων μεταξύ των serverless υπηρεσιών και έναν ασφαλή χώρο για αυτά τα μηνύματα μέχρι να ληφθούν. Το RabbitMQ θα χρησιμοποιηθεί για την υλοποίηση της κεντρικής ουράς του συστήματος, όπου αποθηκεύονται τα συμβάντα πριν δρομολογηθούν στις υπηρεσίες κατανάλωσης.

5.4.4 Παρακολούθηση και Μετρήσεις

Το **Prometheus** είναι ένα ισχυρό εργαλείο παρακολούθησης που χρησιμοποιείται για τη συλλογή και αναζήτηση μετρήσεων από το Kubernetes cluster. Αποφασίσαμε να εγκαταστήσουμε το Prometheus χρησιμοποιώντας το **Kube-Prometheus Stack**, το οποίο παρέχει όλα τα απαραίτητα στοιχεία για την παρακολούθηση του cluster, όπως το Prometheus, το Alertmanager, το Node Exporter, και το Grafana.

Το **Grafana** είναι μια πλατφόρμα ανοικτού κώδικα για την οπτικοποίηση των μετρήσεων. Ενσωματώνεται με το Kubernetes για να οπτικοποιεί τις μετρήσεις που συλλέγονται από το cluster, καθιστώντας ευκολότερη την παρακολούθηση της απόδοσης του συστήματος.

5.4.5 Εγκατάσταση του Redis

Το Redis θα χρησιμοποιηθεί για την προσωρινή αποθήκευση δεδομένων που ζητούνται συχνά. Αποτελείται από έναν master κόμβο και replicas, με υψηλή διαθεσιμότητα και επεκτασιμότητα.

5.4.6 Βασικές δοκιμές - Εισαγωγή και ρύθμιση

Πριν από την ενεργοποίηση της αυτόματης κλιμάκωσης, πραγματοποιήθηκαν βασικές δοκιμές για να προσδιοριστεί το μέγιστο σταθερό φορτίο που μπορούσε να διαχειριστεί το σύστημα υπό σταθερές συνθήκες. Αυτές οι δοκιμές ήταν ζωτικής σημασίας για τον καθορισμό των εγγενών περιορισμών και της συμπεριφοράς του συστήματος όταν υποβάλλεται σε αυξανόμενα φορτία χωρίς τη δυναμική διαχείριση πόρων που παρέχει η αυτόματη κλιμάκωση του Knative.

Οι βασικές δοκιμές επικεντρώθηκαν στον προσδιορισμό:

- Της ικανότητας του συστήματος να επεξεργάζεται αιτήσεις υπό σταθερή κατάσταση.
- Του σημείου στο οποίο η κεντρική ουρά αρχίζει να συσσωρεύει μηνύματα, υποδεικνύοντας ότι το σύστημα είναι υπερφορτωμένο.
- Της ικανότητας του συστήματος να διαχειρίζεται εκρηκτική κίνηση και να ανακάμπτει από περιόδους αδράνειας.

- Των μοτίβων χρήσης της CPU και της μνήμης σε διάφορους ρυθμούς μηνυμάτων.

Με τη διεξαγωγή αυτών των βασικών δοκιμών, καθορίστηκε ένα επίπεδο απόδοσης αναφοράς, το οποίο χρησίμευσε ως σημείο σύγκρισης για τα επόμενα πειράματα που αφορούσαν τις δυνατότητες αυτόματης κλιμάκωσης της Knative (KPA και HPA).

5.4.7 Ρύθμιση Δοκιμών

Οι δοκιμές πραγματοποιήθηκαν με τρεις υπηρεσίες κατανάλωσης, καθεμία με διαφορετικές κατανομές CPU και μνήμης:

- Καταναλωτική υπηρεσία 1: 1 CPU, 2 GiB μνήμη, 1 εργαζόμενος και μέγεθος ουράς 2.
- Καταναλωτική υπηρεσία 2: 1 CPU, 3 GiB μνήμη, 1 εργαζόμενος και μέγεθος ουράς 4.
- Καταναλωτική υπηρεσία 3: 2 CPUs, 5 GiB μνήμη, 2 εργαζόμενοι και μέγεθος ουράς 5.

Ένα προσαρμοσμένο σενάριο, που χρησιμοποιούσε το εργαλείο RabbitMQ PerfTest, χρησιμοποιήθηκε για την προσομοίωση της κίνησης μηνυμάτων σε διάφορους ρυθμούς (μετρούμενο σε αιτήσεις ανά δευτερόλεπτο ή RPS). Το σενάριο επέτρεπε τόσο σταθερά όσο και μεταβλητά μοτίβα κίνησης, συμπεριλαμβανομένων περιόδων αδράνειας, για την παρατήρηση της συμπεριφοράς αποκατάστασης του συστήματος.

Οι βασικές δοκιμές περιλάμβαναν σταθερούς ρυθμούς μηνυμάτων και μοτίβα κίνησης μεταβλητού ρυθμού, για να αξιολογηθεί η ικανότητα του συστήματος να διαχειρίζεται εκρηκτική κίνηση. Τα χαρακτηριστικά κλιμάκωσης της Knative ήταν απενεργοποιημένα, έτσι ώστε το σύστημα να λειτουργεί με τους διαθέσιμους πόρους κάθε υπηρεσίας.

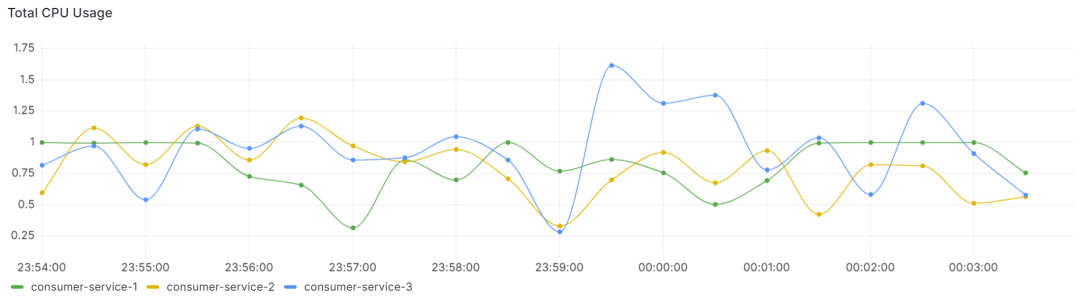
Μετρήθηκαν οι εξής βασικές μετρήσεις:

- Συνολική χρήση CPU σε όλες τις υπηρεσίες κατανάλωσης.
- Χρήση μνήμης για κάθε υπηρεσία.
- Αριθμός συμβάντων ανά κωδικό απάντησης (επιτυχημένες αιτήσεις).
- Καθυστέρηση αποστολής συμβάντων.
- Απόδοση (αιτήσεις ανά δευτερόλεπτο).
- Χρόνος απόκρισης κάθε υπηρεσίας κατανάλωσης.

5.4.8 Βασική δοκιμή 1: Σταθερό χαμηλό φορτίο

Ρύθμιση δοκιμής: 1 αίτηση ανά δευτερόλεπτο (1 RPS) **Περιγραφή:** Κατά τη διάρκεια αυτής της βασικής δοκιμής, το σύστημα υποβλήθηκε σε σταθερό φορτίο 1 RPS χωρίς περιόδους αδράνειας. Ο σκοπός ήταν να παρατηρηθεί η συμπεριφορά του συστήματος κάτω από σταθερές συνθήκες, χωρίς την παρέμβαση κλιμάκωσης. **Σκοπός:** Η δοκιμή αυτή αποσκοπούσε στη διερεύνηση της ικανότητας του συστήματος να επεξεργάζεται αιτήματα με σταθερό ρυθμό, καθώς και στη μελέτη της χρήσης πόρων όπως η CPU και η μνήμη, χωρίς δυναμική κλιμάκωση.

Χρήση CPU (Συνολική χρήση CPU για υπηρεσίες καταναλωτών)



Σχημα 5.1: Βασικές δοκιμές: Συνολική χρήση CPU για σταθερό φορτίο 1 RPS

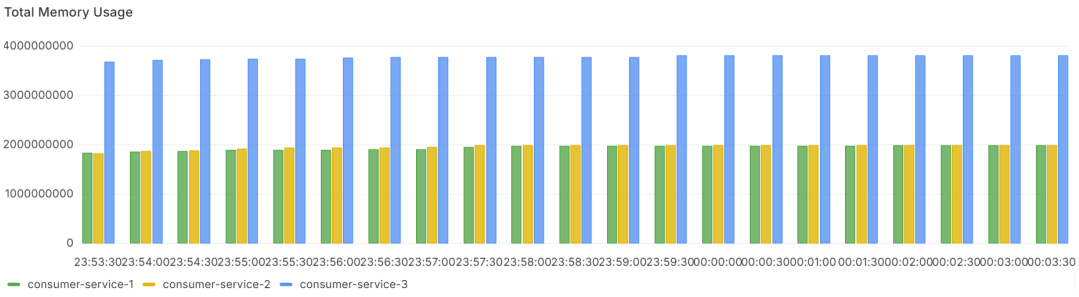
Από το διάγραμμα που απεικονίζει τη συνολική χρήση της CPU, παρατηρούμε ότι οι τρεις καταναλωτικές υπηρεσίες παρουσιάζουν μεταβολές στη χρήση των πόρων τους, ακόμα και χωρίς αυτόματη κλιμάκωση. Η υπηρεσία καταναλωτή-3 φαίνεται να χρησιμοποιεί περισσότερους πόρους CPU από τις υπόλοιπες, αλλά οι διακυμάνσεις είναι περιορισμένες.

Βασικές παρατηρήσεις:

- Οι τρεις υπηρεσίες παρουσιάζουν διακυμάνσεις στη χρήση CPU, με την υπηρεσία καταναλωτή-3 να καταγράφει ελαφρώς υψηλότερη κατανάλωση.
- Οι διακυμάνσεις αντικατοπτρίζουν την επεξεργασία αιτημάτων, με τις αυξήσεις να σηματοδοτούν έντονη δραστηριότητα και τις μειώσεις να υποδηλώνουν περιόδους χαμηλότερης ζήτησης.

Αναμενόμενη συμπεριφορά: Υπό συνθήκες σταθερού φορτίου, η χρήση της CPU θα πρέπει να παραμένει σταθερή με μικρές διακυμάνσεις λόγω της απουσίας δυναμικής κλιμάκωσης.

Χρήση Μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)



Σχημα 5.2: Βασικές δοκιμές: Συνολική χρήση μνήμης για σταθερό φορτίο 1 RPS

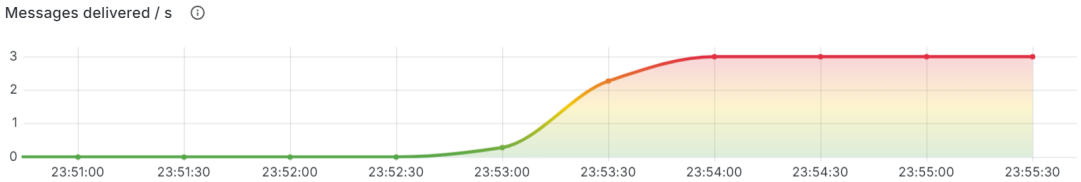
Όσον αφορά τη χρήση μνήμης, το διάγραμμα δείχνει μια γενική σταθερότητα, με την καταναλωτική υπηρεσία-3 να εμφανίζει αυξημένη κατανάλωση σε σύγκριση με τις άλλες δύο υπηρεσίες. Ωστόσο, η συνολική συμπεριφορά είναι σχετικά σταθερή.

Βασικές παρατηρήσεις:

- Η καταναλωτική υπηρεσία-3 συνεχίζει να καταναλώνει περισσότερη μνήμη από τις άλλες δύο.
- Η μνήμη διατηρείται σε σταθερά επίπεδα, χωρίς μεγάλες αυξομειώσεις, γεγονός που υποδηλώνει ότι οι υπηρεσίες διαχειρίζονται το φορτίο χωρίς υπερκατανάλωση πόρων.

Αναμενόμενη συμπεριφορά: Καθώς δεν υπάρχει κλιμάκωση, η χρήση μνήμης θα πρέπει να είναι σταθερή, και δεν αναμένονται μεγάλες αυξήσεις ή μειώσεις.

Παραδοθέντα μηνύματα (ρυθμοί εξερχόμενων μηνυμάτων)



Σχημα 5.3: Βασικές δοκιμές: Ρυθμός παραδοθέντων μηνυμάτων για σταθερό φορτίο 1 RPS

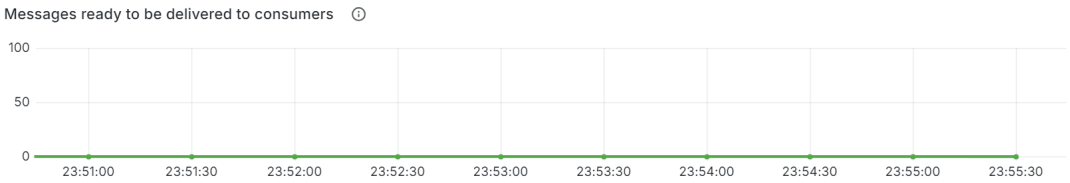
Ο ρυθμός παράδοσης μηνυμάτων δείχνει ότι το σύστημα μπορεί να διαχειριστεί το σταθερό φορτίο χωρίς καθυστερήσεις ή συσσώρευση. Όπως φαίνεται, ο ρυθμός παραμένει σταθερός στις 3 αιτήσεις ανά δευτερόλεπτο, γεγονός που υποδεικνύει την αποτελεσματική διαχείριση του συστήματος.

Βασικές παρατηρήσεις:

- Ο σταθερός ρυθμός παράδοσης δείχνει ότι το σύστημα μπορεί να διατηρήσει σταθερή απόδοση.
- Δεν υπάρχουν διακυμάνσεις ή ανησυχητικά σημάδια που να υποδηλώνουν πρόβλημα στην επεξεργασία των αιτημάτων.

Αναμενόμενη συμπεριφορά: Το σύστημα παραδίδει τα μηνύματα χωρίς καθυστερήσεις ή προβλήματα επεξεργασίας, κάτι που είναι σύμφωνο με τη σταθερή ζήτηση που υφίσταται.

Μηνύματα στην ουρά



Σχημα 5.4: Βασικές δοκιμές: Μηνύματα στην ουρά για σταθερό φορτίο 1 RPS

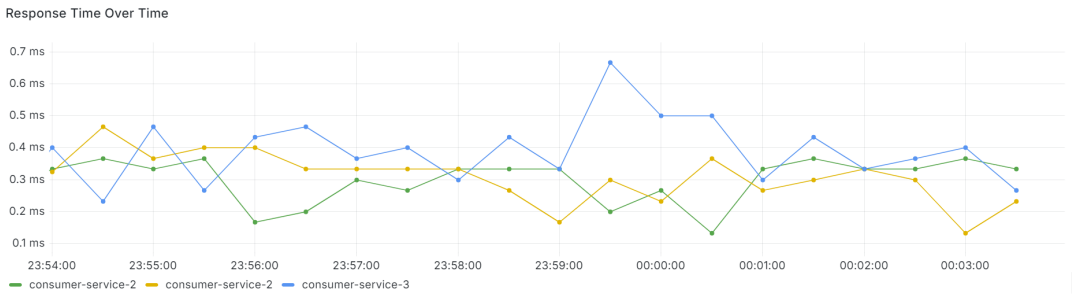
Το διάγραμμα της ουράς αποκαλύπτει ότι δεν υπάρχει συσσώρευση μηνυμάτων, καθώς η ουρά παραμένει άδεια καθ' όλη τη διάρκεια της δοκιμής. Αυτό υποδηλώνει ότι το σύστημα μπορεί να επεξεργάζεται άμεσα όλα τα αιτήματα.

Βασικές παρατηρήσεις:

- Η απουσία μηνυμάτων στην ουρά δείχνει ότι δεν υπάρχει καθυστέρηση στην επεξεργασία των αιτημάτων.

Αναμενόμενη συμπεριφορά: Σε ένα καλά ρυθμισμένο σύστημα, με σταθερό φορτίο και χωρίς κλιμάκωση, η ουρά θα πρέπει να παραμένει άδεια, όπως παρατηρείται εδώ.

Χρόνος Απόκρισης



Σχημα 5.5: Βασικές δοκιμές: Χρόνος απόκρισης για σταθερό φορτίο 1 RPS

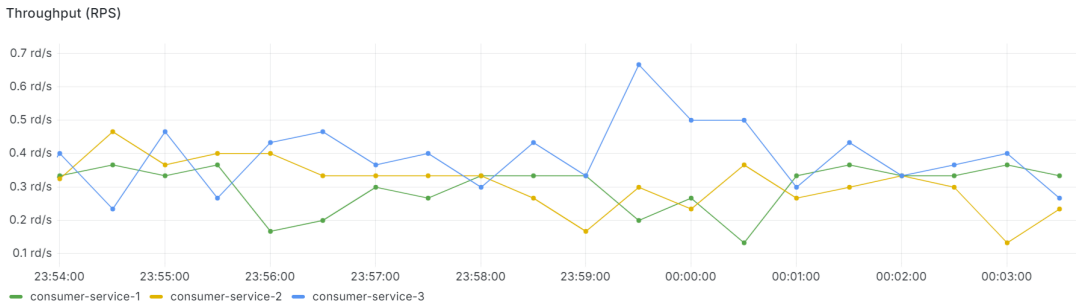
Οι χρόνοι απόκρισης για τις καταναλωτικές υπηρεσίες παρουσιάζουν μικρές διακυμάνσεις, παραμένοντας όμως σε εξαιρετικά χαμηλά επίπεδα. Οι χρόνοι κυμαίνονται από 0.2ms έως 0.5ms, γεγονός που υποδηλώνει την αποτελεσματική διαχείριση του σταθερού φορτίου.

Βασικές παρατηρήσεις:

- Ο χρόνος απόκρισης είναι γρήγορος και οι μικρές διακυμάνσεις είναι φυσιολογικές για το σταθερό φορτίο που υποβάλλεται το σύστημα.
- Οι τρεις καταναλωτικές υπηρεσίες εμφανίζουν παρόμοια απόδοση, κάτι που δείχνει ότι δεν υπάρχει ανισότητα στην κατανομή του φορτίου.

Αναμενόμενη συμπεριφορά: Οι χαμηλοί χρόνοι απόκρισης επιβεβαιώνουν την ομαλή λειτουργία του συστήματος υπό τις παρούσες συνθήκες.

Απόδοση (RPS)



Σχημα 5.6: Βασικές δοκιμές: Απόδοση αιτημάτων (RPS) για σταθερό φορτίο 1 RPS

Όπως φαίνεται από το γράφημα της απόδοσης, η επεξεργασία αιτημάτων παραμένει σταθερή με μικρές διακυμάνσεις, κυρίως γύρω από τα 0.3-0.5 αιτήματα ανά δευτερόλεπτο. Οι αυξομειώσεις σχετίζονται με την επεξεργασία των αιτημάτων από τις καταναλωτικές υπηρεσίες.

Βασικές παρατηρήσεις:

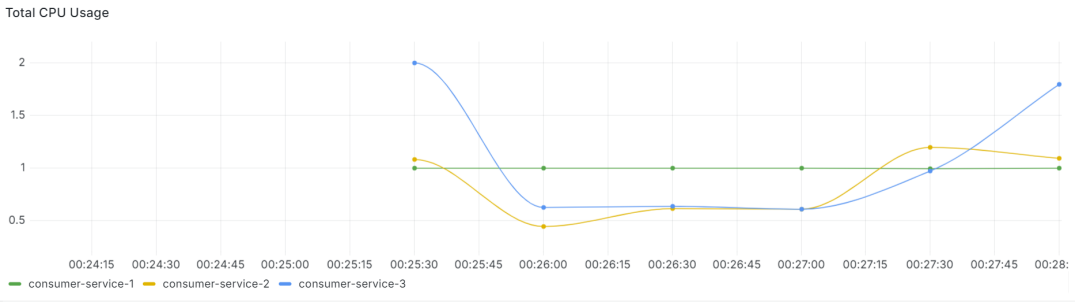
- Η απόδοση των αιτημάτων παραμένει σταθερή, γεγονός που δείχνει ότι το σύστημα διαχειρίζεται σωστά το φορτίο.

Αναμενόμενη συμπεριφορά: Σε ένα σύστημα με σταθερό φορτίο, η απόδοση θα πρέπει να παραμένει σταθερή, όπως παρατηρείται στο γράφημα.

5.4.9 Βασική δοκιμή 2: Ελαφρώς έντονη κυκλοφορία

Ρύθμιση δοκιμής: 3 αιτήσεις ανά δευτερόλεπτο (3 RPS) **Περιγραφή**: Αυτή η δοκιμή εφαρμόζει ένα σταθερό φορτίο με 3 αιτήσεις ανά δευτερόλεπτο, χωρίς περιόδους αδράνειας, με στόχο την παρακολούθηση της συμπεριφοράς του συστήματος υπό συνθήκες αυξημένου φορτίου. **Σκοπός**: Ο στόχος αυτής της δοκιμής είναι η διερεύνηση της ικανότητας του συστήματος να διαχειρίζεται υψηλότερο και σταθερό φορτίο, εξετάζοντας πώς ανταποκρίνονται οι υπηρεσίες καταναλωτών και πώς χρησιμοποιούνται οι πόροι.

Χρήση CPU (Συνολική χρήση CPU για υπηρεσίες καταναλωτών)



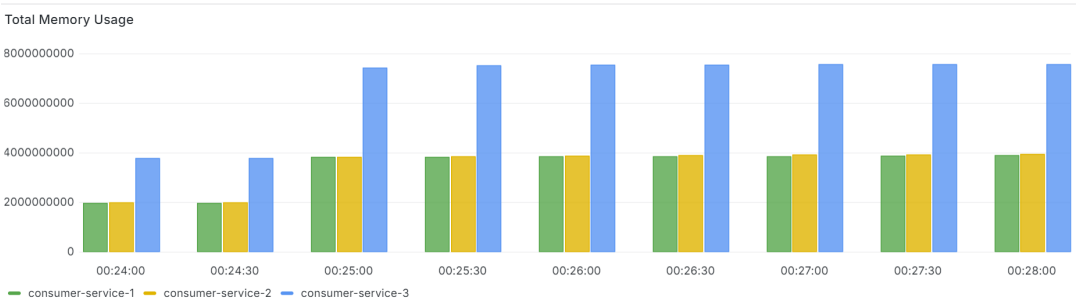
Σχημα 5.7: Βασικές δοκιμές: Συνολική χρήση CPU για σταθερό φορτίο 3 RPS

Η συνολική χρήση CPU εμφανίζει διακυμάνσεις ανάμεσα στις τρεις καταναλωτικές υπηρεσίες. Ενώ η υπηρεσία καταναλωτή-1 διατηρεί σταθερή κατανάλωση CPU, οι άλλες δύο υπηρεσίες παρουσιάζουν αυξομειώσεις, με την υπηρεσία καταναλωτή-3 να καταναλώνει τους περισσότερους πόρους.

Βασικές παρατηρήσεις:

- Η καταναλωτική υπηρεσία-3 χρησιμοποιεί τη μεγαλύτερη ποσότητα CPU, ενώ η καταναλωτική υπηρεσία-1 διατηρεί σταθερή χρήση.
- Οι διακυμάνσεις που παρατηρούνται δείχνουν αυξομειώσεις στη διαχείριση των αιτημάτων από τις υπηρεσίες.

Αναμενόμενη συμπεριφορά: Καθώς το φορτίο είναι υψηλότερο (3 RPS), αναμένονται μεγαλύτερες αυξομειώσεις στη χρήση CPU, ιδιαίτερα για τις πιο "βαριές" υπηρεσίες, όπως η καταναλωτική υπηρεσία-3.



Σχημα 5.8: Βασικές δοκιμές: Συνολική χρήση μνήμης για σταθερό φορτίο 3 RPS

Χρήση Μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

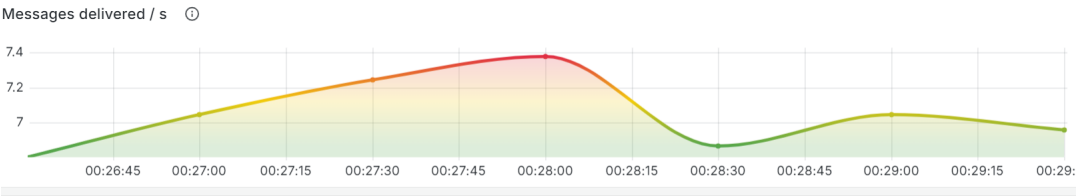
Η χρήση μνήμης παραμένει σχετικά σταθερή, με την καταναλωτική υπηρεσία-3 να καταναλώνει σταθερά τη μεγαλύτερη ποσότητα μνήμης, ενώ οι υπόλοιπες δύο υπηρεσίες καταναλώνουν παρόμοια επίπεδα.

Βασικές παρατηρήσεις:

- Η καταναλωτική υπηρεσία-3 διατηρεί τη μεγαλύτερη κατανάλωση μνήμης, όπως αναμένεται δεδομένου του αυξημένου αριθμού αντιγράφων της.
- Οι άλλες δύο υπηρεσίες παρουσιάζουν σταθερή κατανάλωση μνήμης χωρίς μεγάλες αυξομειώσεις.

Αναμενόμενη συμπεριφορά: Σε συνθήκες σταθερού φορτίου 3 RPS, η χρήση μνήμης θα πρέπει να παραμένει σχετικά σταθερή, ενώ η καταναλωτική υπηρεσία-3 αναμενόμενα καταναλώνει περισσότερους πόρους λόγω του μεγαλύτερου όγκου εργασίας που διαχειρίζεται.

Παραδοθέντα μηνύματα (ρυθμοί εξερχόμενων μηνυμάτων)



Σχημα 5.9: Βασικές δοκιμές: Ρυθμός παραδοθέντων μηνυμάτων για σταθερό φορτίο 3 RPS

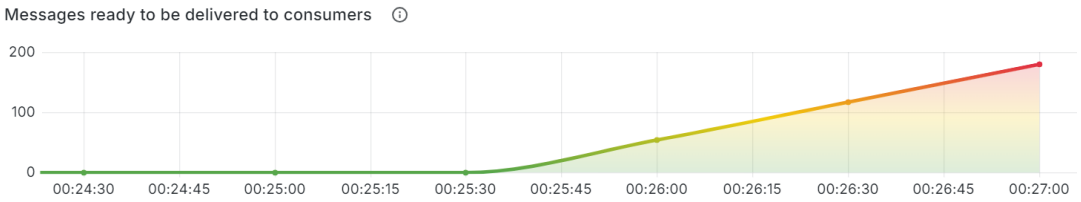
Η παράδοση μηνυμάτων παραμένει σταθερή με αυξανόμενο ρυθμό, καθώς το σύστημα επεξεργάζεται τα εισερχόμενα αιτήματα. Το σύστημα φτάνει σε μέγιστο ρυθμό παράδοσης 7.4 αιτημάτων ανά δευτερόλεπτο, πριν μειωθεί σταδιακά.

Βασικές παρατηρήσεις:

- Το σύστημα επεξεργάζεται τα αιτήματα χωρίς σημαντική καθυστέρηση, με έναν μέγιστο ρυθμό παράδοσης περίπου 7.4 αιτημάτων ανά δευτερόλεπτο.
- Η σταδιακή μείωση του ρυθμού παράδοσης πιθανώς υποδηλώνει σταδιακή μείωση της δραστηριότητας μετά το πέρας των αιτημάτων.

Αναμενόμενη συμπεριφορά: Σε συνθήκες αυξημένου φορτίου, η παράδοση μηνυμάτων αναμένεται να παραμένει σταθερή όσο το σύστημα μπορεί να διαχειρίζεται τα αιτήματα.

Μηνύματα στην ουρά



Σχημα 5.10: Βασικές δοκιμές: Μηνύματα στην ουρά για σταθερό φορτίο 3 RPS

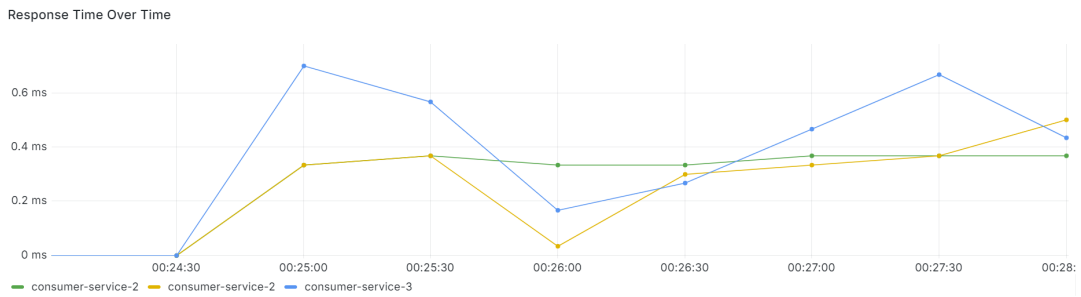
Το μέγεθος της ουράς αυξάνεται σταδιακά καθώς εισάγονται νέα αιτήματα στο σύστημα. Ο αριθμός των μηνυμάτων στην ουρά φτάνει περίπου τα 200 καθώς το σύστημα συνεχίζει να επεξεργάζεται τα εισερχόμενα αιτήματα.

Βασικές παρατηρήσεις:

- Η ουρά των αιτημάτων αυξάνεται όσο ο ρυθμός των εισερχόμενων αιτημάτων παραμένει υψηλός, με τη συσσώρευση να φτάνει τα 200 μηνύματα.

Αναμενόμενη συμπεριφορά: Με σταθερό φορτίο 3 RPS, η ουρά των μηνυμάτων αναμένεται να αυξάνεται καθώς το σύστημα διαχειρίζεται τα αιτήματα, αλλά δεν τα επεξεργάζεται άμεσα όλα.

Χρόνος Απόκρισης



Σχημα 5.11: Βασικές δοκιμές: Χρόνος απόκρισης για σταθερό φορτίο 3 RPS

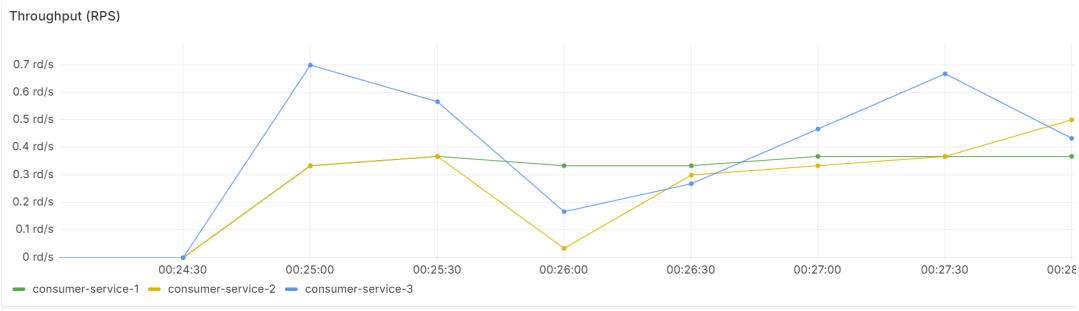
Ο χρόνος απόκρισης παρουσιάζει κάποιες διακυμάνσεις, με τις τιμές να κυμαίνονται μεταξύ 0.2ms και 0.6ms. Η υπηρεσία καταναλωτή-3 εμφανίζει τις μεγαλύτερες αυξήσεις στον χρόνο απόκρισης, ενώ οι άλλες δύο υπηρεσίες παρουσιάζουν μικρότερες διακυμάνσεις.

Βασικές παρατηρήσεις:

- Η υπηρεσία καταναλωτή-3 εμφανίζει τις μεγαλύτερες αυξήσεις στον χρόνο απόκρισης, κάτι που αναμένεται λόγω του μεγαλύτερου όγκου αιτημάτων που διαχειρίζεται.
- Οι άλλες δύο υπηρεσίες παραμένουν πιο σταθερές, με μικρότερες αυξομειώσεις.

Αναμενόμενη συμπεριφορά: Ο χρόνος απόκρισης για την καταναλωτική υπηρεσία-3 αναμένεται να είναι μεγαλύτερος λόγω της αυξημένης επεξεργασίας αιτημάτων. Οι διακυμάνσεις που παρατηρούνται είναι λογικές για τις συνθήκες του αυξημένου φορτίου.

Απόδοση (RPS)



Σχημα 5.12: Βασικές δοκιμές: Απόδοση αιτημάτων (RPS) για σταθερό φορτίο 3 RPS

Η απόδοση αιτημάτων κυμαίνεται γύρω από 0.4 έως 0.7 αιτήματα ανά δευτερόλεπτο, με την υπηρεσία καταναλωτή-3 να εμφανίζει την υψηλότερη απόδοση σε σύγκριση με τις άλλες δύο υπηρεσίες.

Βασικές παρατηρήσεις:

- Η υπηρεσία καταναλωτή-3 επιτυγχάνει την υψηλότερη απόδοση, γεγονός που συμβαδίζει με την αυξημένη της χρήση CPU.
- Οι υπόλοιπες υπηρεσίες διατηρούν πιο σταθερή απόδοση, με μικρότερες αυξομειώσεις.

Αναμενόμενη συμπεριφορά: Με σταθερό φορτίο 3 RPS, η απόδοση των αιτημάτων αναμένεται να παραμένει σε υψηλά επίπεδα, ειδικά για την υπηρεσία καταναλωτή-3, που επεξεργάζεται το μεγαλύτερο όγκο αιτημάτων.

5.4.10 Βασική δοκιμή 3: Μεταβαλλόμενο φορτίο

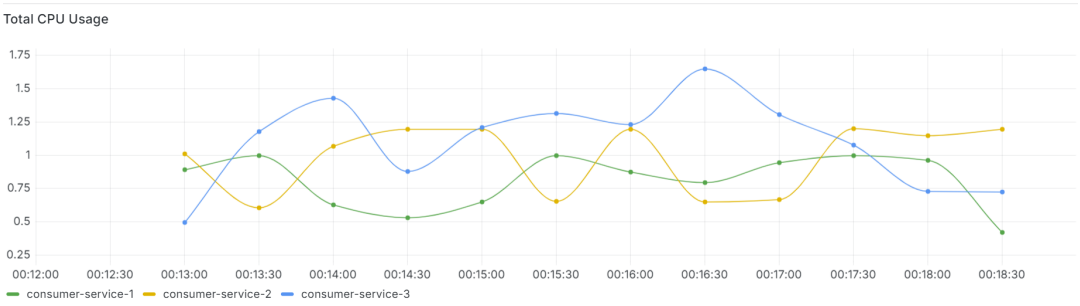
Ρύθμιση δοκιμής: 1 RPS για 30 δευτερόλεπτα, 0 RPS για 10 δευτερόλεπτα, 3 RPS για 10 δευτερόλεπτα. **Περιγραφή:** Σε αυτή τη δοκιμή εφαρμόζεται ένα μεταβαλλόμενο φορτίο με φάσεις σταθερής και μηδενικής κίνησης. Σκοπός είναι να παρατηρηθεί πώς το σύστημα προσαρμόζεται στις αλλαγές του φορτίου και πώς ανταποκρίνεται σε περιόδους αδράνειας και αυξημένης κίνησης. **Σκοπός:** Αυτή η δοκιμή έχει στόχο να αναδείξει την ικανότητα του συστήματος να ανταποκρίνεται δυναμικά σε εναλλαγές κίνησης, ειδικά στην αλλαγή από αδράνεια σε αιχμή κίνησης.

Χρήση CPU (Συνολική χρήση CPU για υπηρεσίες καταναλωτών)

Η συνολική χρήση CPU παρουσιάζει σημαντικές διακυμάνσεις ανάμεσα στις φάσεις της δοκιμής. Κατά τη φάση των 1 RPS, η χρήση CPU είναι σταθερή και σχετικά χαμηλή. Κατά τη φάση των 0 RPS, η χρήση μειώνεται περαιτέρω, ενώ στη φάση των 3 RPS η χρήση αυξάνεται σημαντικά, ειδικά για την καταναλωτική υπηρεσία-3.

Βασικές παρατηρήσεις:

- Η χρήση CPU μειώνεται κατά τη διάρκεια των περιόδων αδράνειας (0 RPS) και αυξάνεται σημαντικά όταν το φορτίο ανεβαίνει στα 3 RPS.

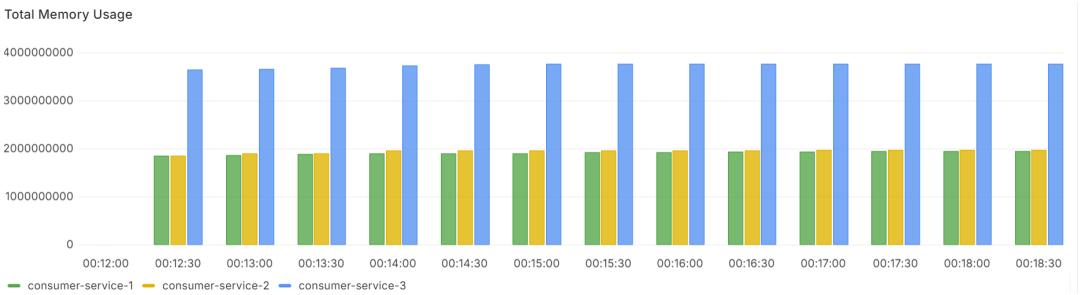


Σχημα 5.13: Μεταβαλλόμενο φορτίο: Συνολική χρήση CPU για 1 RPS, 0 RPS, και 3 RPS

- Η καταναλωτική υπηρεσία-3 παρουσιάζει την πιο έντονη αύξηση στη χρήση CPU, επιβεβαιώνοντας τη δυνατότητά της να διαχειρίζεται αυξημένο φόρτο.

Αναμενόμενη συμπεριφορά: Σε περιόδους αδράνειας, η χρήση CPU θα πρέπει να μειώνεται, ενώ όταν αυξάνεται το φορτίο, η CPU αναμένεται να ανταποκρίνεται ανάλογα, με αυξημένες ανάγκες επεξεργασίας.

Χρήση Μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)



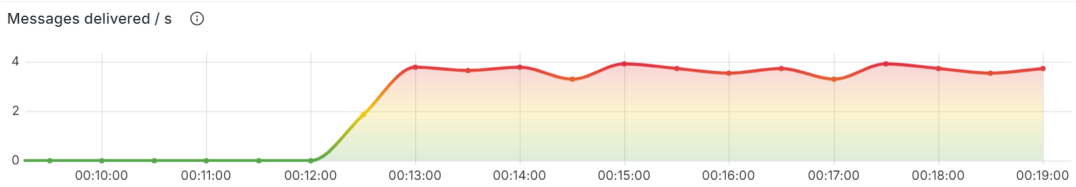
Σχημα 5.14: Μεταβαλλόμενο φορτίο: Χρήση μνήμης για 1 RPS, 0 RPS, και 3 RPS

Η χρήση μνήμης παραμένει σχετικά σταθερή καθ' όλη τη διάρκεια της δοκιμής, ανεξάρτητα από τις μεταβολές στο φορτίο. Η καταναλωτική υπηρεσία-3 συνεχίζει να καταναλώνει τη μεγαλύτερη ποσότητα μνήμης, ενώ οι άλλες δύο υπηρεσίες εμφανίζουν παρόμοιες καταναλώσεις.

Βασικές παρατηρήσεις:

- Η χρήση μνήμης δεν παρουσιάζει σημαντικές διακυμάνσεις, παρά την εναλλαγή του φορτίου.
- Η καταναλωτική υπηρεσία-3 παραμένει η υπηρεσία με τη μεγαλύτερη χρήση μνήμης.

Αναμενόμενη συμπεριφορά: Σε περιπτώσεις μεταβαλλόμενου φορτίου, η χρήση μνήμης αναμένεται να είναι πιο σταθερή, καθώς η ανάγκη για πόρους μνήμης δεν αλλάζει τόσο δραστικά όσο η χρήση CPU.



Σχημα 5.15: Μεταβαλλόμενο φορτίο: Ρυθμός παραδοθέντων μηνυμάτων για 1 RPS, 0 RPS, και 3 RPS

Παραδοθέντα μηνύματα (ρυθμοί εξερχόμενων μηνυμάτων)

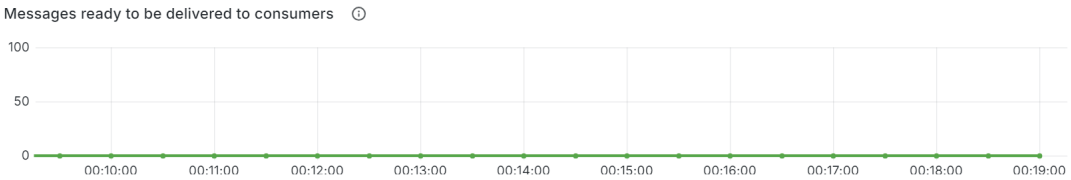
Το γράφημα δείχνει έναν καθαρό ρυθμό παραδοθέντων μηνυμάτων, ο οποίος αυξάνεται σταδιακά κατά τη διάρκεια της φάσης των 1 RPS και κορυφώνεται στην τελευταία φάση των 3 RPS. Κατά τη διάρκεια της αδράνειας (0 RPS), η παράδοση μηνυμάτων σταματά.

Βασικές παρατηρήσεις:

- Η παράδοση μηνυμάτων σταματά εντελώς κατά τη φάση των 0 RPS, όπως αναμένεται.
- Ο ρυθμός παραδόσεων αυξάνεται γρήγορα στην τελευταία φάση των 3 RPS.

Αναμενόμενη συμπεριφορά: Κατά τη φάση των 0 RPS, η παράδοση μηνυμάτων αναμένεται να σταματά, ενώ η παράδοση αυξάνεται ραγδαία μόλις το σύστημα δεχτεί περισσότερα αιτήματα.

Μηνύματα στην ουρά



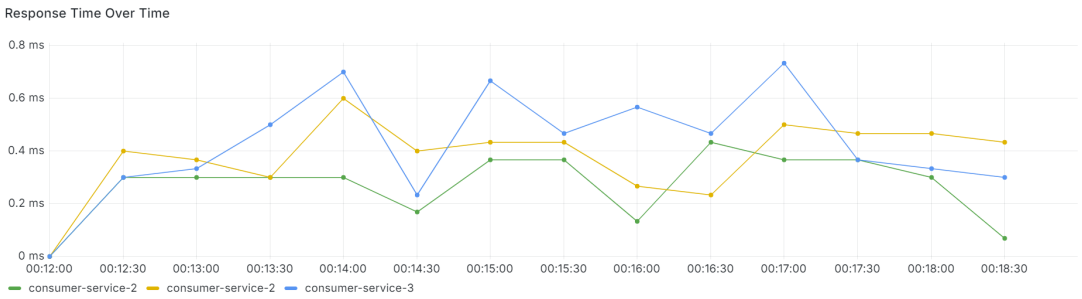
Σχημα 5.16: Μεταβαλλόμενο φορτίο: Μηνύματα στην ουρά για 1 RPS, 0 RPS, και 3 RPS

Το μέγεθος της ουράς παραμένει σχεδόν μηδενικό κατά τη διάρκεια της φάσης των 0 RPS, ενώ αυξάνεται ελαφρώς όταν το φορτίο αυξάνεται. Η ουρά διαχειρίζεται αποτελεσματικά τα εισερχόμενα αιτήματα, με το σύστημα να παραμένει σε μεγάλο βαθμό άδειο.

Βασικές παρατηρήσεις:

- Η ουρά παραμένει μηδενική ή πολύ χαμηλή κατά τη διάρκεια όλης της δοκιμής, υποδεικνύοντας την αποτελεσματική επεξεργασία των αιτημάτων.

Αναμενόμενη συμπεριφορά: Η ουρά δεν αναμένεται να γεμίζει όταν η κίνηση είναι μικρή, και κατά τη διάρκεια της φάσης των 0 RPS, η ουρά λογικά παραμένει κενή.



Σχημα 5.17: Μεταβαλλόμενο φορτίο: Χρόνος απόκρισης για 1 RPS, 0 RPS, και 3 RPS

Χρόνος Απόκρισης

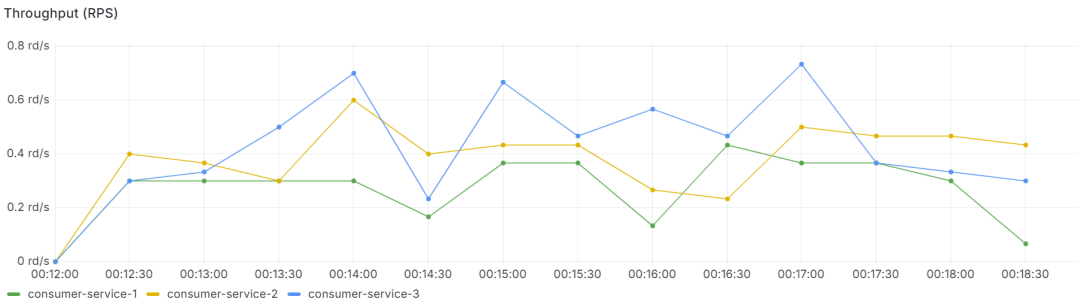
Ο χρόνος απόκρισης εμφανίζει διακυμάνσεις, κυρίως κατά τη φάση των 3 RPS, με τις αυξήσεις να οφείλονται στον αυξημένο φόρτο. Στις περιόδους 1 RPS και 0 RPS, οι χρόνοι απόκρισης παραμένουν χαμηλοί και σταθεροί.

Βασικές παρατηρήσεις:

- Κατά τη φάση των 3 RPS, παρατηρούνται αυξημένοι χρόνοι απόκρισης, όπως αναμένεται λόγω του αυξημένου φόρτου.
- Στις υπόλοιπες φάσεις (1 RPS, 0 RPS), οι χρόνοι απόκρισης παραμένουν χαμηλοί.

Αναμενόμενη συμπεριφορά: Ο χρόνος απόκρισης αναμένεται να αυξηθεί καθώς αυξάνεται το φορτίο, και να είναι μικρός όταν το σύστημα δεν δέχεται αιτήματα (0 RPS).

Απόδοση Αιτήσεων (RPS)



Σχημα 5.18: Μεταβαλλόμενο φορτίο: Απόδοση αιτήσεων για 1 RPS, 0 RPS, και 3 RPS

Η απόδοση των αιτήσεων (RPS) ακολουθεί τα μοτίβα του εισερχόμενου φορτίου. Κατά τη φάση των 0 RPS, η απόδοση μηδενίζεται, ενώ στις φάσεις των 1 και 3 RPS, η απόδοση αυξάνεται και προσαρμόζεται στο εισερχόμενο φορτίο.

Βασικές παρατηρήσεις:

- Κατά τη φάση των 0 RPS, η απόδοση μηδενίζεται όπως αναμένεται.
- Στις φάσεις των 1 και 3 RPS, η απόδοση αυξάνεται ανάλογα με το εισερχόμενο φορτίο.

Αναμενόμενη συμπεριφορά: Η απόδοση αιτήσεων θα πρέπει να ακολουθεί τη δυναμική του εισερχόμενου φορτίου, αυξάνοντας σε περιόδους υψηλού φόρτου και μηδενιζόμενη κατά την αδράνεια.

5.4.11 Συμπεράσματα βασικών δοκιμών

Στις βασικές δοκιμές μας, πραγματοποιήσαμε μετρήσεις για σταθερά φορτία 1 RPS και 3 RPS, καθώς και για μεταβαλλόμενο φορτίο (μεταξύ 1 RPS, 0 RPS, και 3 RPS). Για το σταθερό φορτίο 1 RPS, η χρήση CPU και μνήμης παρέμεινε σταθερή, με μικρές διακυμάνσεις μεταξύ των καταναλωτικών υπηρεσιών. Η καταναλωτική υπηρεσία-3 παρουσίασε υψηλότερη κατανάλωση πόρων, ενώ η απόδοση των αιτημάτων και οι χρόνοι απόκρισης ήταν σταθεροί και ικανοποιητικοί. Για το σταθερό φορτίο 3 RPS, παρατηρήθηκαν μεγαλύτερες αυξομειώσεις στη χρήση πόρων, ειδικά για την υπηρεσία-3, ενώ η ουρά μηνυμάτων αυξήθηκε, υποδεικνύοντας τη μεγαλύτερη πίεση στο σύστημα.

Στη δοκιμή με μεταβαλλόμενο φορτίο, η χρήση πόρων αντανακλούσε τις εναλλαγές του φορτίου. Η CPU αυξήθηκε κατά τη φάση των 3 RPS, ενώ η μνήμη παρέμεινε σχετικά σταθερή. Η ουρά των αιτημάτων παρέμεινε κενή κατά τη διάρκεια της αδράνειας, ενώ αυξήθηκε κατά τις φάσεις υψηλότερου φορτίου. Η απόδοση αιτημάτων και οι χρόνοι απόκρισης προσαρμόστηκαν αντίστοιχα στις απαιτήσεις του φορτίου.

Στο επόμενο στάδιο των δοκιμών, θα ενεργοποιήσουμε τον μηχανισμό κλιμάκωσης για να αξιολογήσουμε πώς το σύστημα προσαρμόζεται δυναμικά στις αλλαγές του φορτίου. Θα εξετάσουμε την ικανότητα του συστήματος να προσαρμόζει τη χρήση πόρων ανάλογα με τις ανάγκες, βελτιστοποιώντας την απόδοση και την αποδοτικότητα.

5.5 Συμπεριφορά Κλιμάκωσης με Knative Pod Autoscaler (KPA) υπό Μεταβαλλόμενες Συνθήκες Φορτίου

Για να αξιολογηθεί συνολικά η συμπεριφορά του συστήματος αυτόματης κλιμάκωσης υπό δυναμικά μεταβαλλόμενες συνθήκες κυκλοφορίας, διεξήχθη μια σειρά πειραμάτων. Οι δοκιμές αυτές επικεντρώθηκαν στην ικανότητα του συστήματος να χειρίζεται μεταβαλλόμενα πρότυπα κίνησης, ιδίως σε σενάρια όπου θα μπορούσε να προκύψει συσσώρευση μηνυμάτων εάν το σύστημα αποτύχει να κλιμακωθεί σωστά. Η ταυτόχρονη εξυπηρέτηση των target containers ρυθμίστηκε έτσι ώστε να προσαρμόζεται αυτόματα ο αριθμός των εργαζομένων με βάση το εισερχόμενο φορτίο, διασφαλίζοντας ότι το σύστημα θα μπορούσε να κατανέμει δυναμικά τους πόρους ανάλογα με τις ανάγκες. Στόχος ήταν να παρατηρηθεί πόσο καλά το σύστημα μπορούσε να αποφύγει τη συσσώρευση μηνυμάτων, ενώ παράλληλα κλιμακωνόταν αποτελεσματικά τόσο σε περιόδους υψηλού όσο και σε περιόδους χαμηλού φόρτου.

Τα πειράματα σχεδιάστηκαν για να προσομοιώσουν σενάρια του πραγματικού κόσμου, όπως αυτά που συναντώνται σε υπηρεσίες επεξεργασίας εικόνας, όπου η κίνηση μπορεί να είναι διακοπτόμενη, bursty ή σταθερή. Εξετάζοντας την απόδοση του συστήματος σε αυτά τα σενάρια, μπορούμε να κατανοήσουμε καλύτερα την αποτελεσματικότητα της αυτόματης κλιμάκωσης, καθώς και την ικανότητά του να διατηρεί την απόδοση ελαχιστοποιώντας τη σπατάλη πόρων.

Σε κάθε πείραμα εφαρμόστηκαν διαφορετικά φορτία κίνησης μεταβλητού ρυθμού, προσομοιώνοντας απαιτήσεις πραγματικού χρόνου. Το σύστημα αξιολογήθηκε ως προς την ικανότητά του να αυξάνει την κλίμακα κατά τη διάρκεια υψηλής κίνησης και να μειώνει την κλίμακα κατά τη διάρκεια περιόδων χαμηλότερης ζήτησης ή αδράνειας, διασφαλίζοντας την αποτελεσματική χρήση των πόρων. Παρακάτω περιγράφουμε τα πειράματα που πραγματοποιήθηκαν, το σκεπτικό πίσω από κάθε δοκιμή και τις αναμενόμενες συμπεριφορές.

Για τα πειράματα, η ταυτόχρονη εξυπηρέτηση των καταναλωτικών υπηρεσιών διαμορφώθηκε ως εξής:

Η υπηρεσία Consumer-Service-1 είχε ρυθμιστεί με containerConcurrency 2, δηλαδή κάθε instance μπορούσε να επεξεργάζεται έως και 2 αιτήματα ταυτόχρονα. Αυτή η υπηρεσία είχε μέγιστη κλίμακα 3, με όρια CPU ορισμένα στα 1000m και όρια μνήμης στα 2048Mi. Επίσης, είχε μέγεθος ουράς 2 και λειτουργούσε με 1 εργαζόμενο.

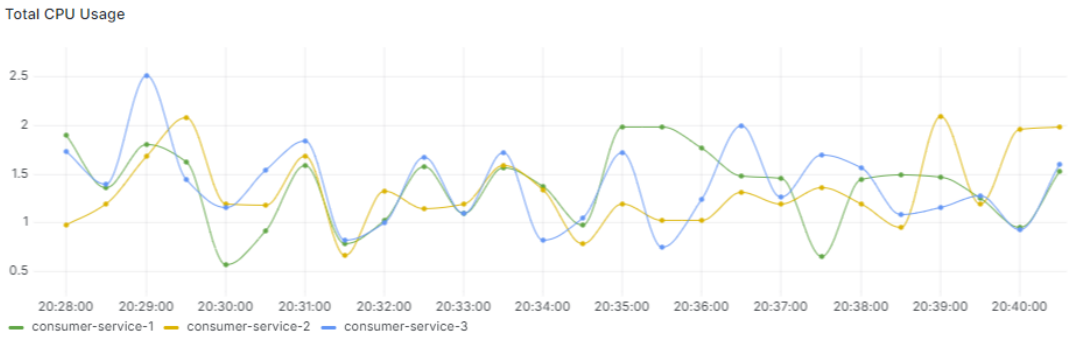
Η υπηρεσία Consumer-Service-2 είχε υψηλότερο containerConcurrency 4, επιτρέποντάς της να διαχειρίζεται έως και 4 αιτήματα ταυτόχρονα ανά instance. Η μέγιστη κλίμακα για αυτή την υπηρεσία ήταν 2, με όρια CPU στα 1200m και όρια μνήμης στα 3Gi. Είχε ρυθμιστεί με μέγεθος ουράς 4 και χρησιμοποιούσε 1 εργαζόμενο.

Η υπηρεσία Consumer-Service-3 είχε το υψηλότερο containerConcurrency, ρυθμισμένο στα 5, επιτρέποντας σε κάθε instance να επεξεργάζεται έως και 5 αιτήματα ταυτόχρονα. Είχε μέγιστη κλίμακα 2, με όρια CPU στα 2000m και όρια μνήμης στα 5Gi. Αυτή η υπηρεσία είχε μέγεθος ουράς 5 και λειτουργούσε με 2 εργαζόμενους.

5.5.1 Υψηλό φορτίο με ανάκαμψη σε αδράνεια

Ρύθμιση δοκιμής: 4:60, 0:60
Περιγραφή: Σε αυτή τη δοκιμή εφαρμόστηκε ένα συνεχές υψηλό φορτίο με 4 αιτήματα ανά δευτερόλεπτο για 60 δευτερόλεπτα, και ακολούθησαν 60 δευτερόλεπτα αδράνειας (0 αιτήματα ανά δευτερόλεπτο).
Σκοπός: Ο στόχος ήταν να παρατηρηθεί η ικανότητα του συστήματος να αυξάνει την κλίμακα για να διαχειρίζεται υψηλή κίνηση και στη συνέχεια να ανακάμπτει κατά τη διάρκεια των περιόδων αδράνειας μειώνοντας την κλίμακα. Αυτή η δοκιμή εξετάζει την ισορροπία μεταξύ της αποτελεσματικότητας της κλιμάκωσης και της χρήσης πόρων, ειδικά στο πόσο καλά το σύστημα αποφεύγει τη συσσώρευση μηνυμάτων κατά την υψηλή κίνηση και αν απελευθερώνει σωστά πόρους κατά τις περιόδους αδράνειας. Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να διαχειρίζεται το υψηλό φορτίο χωρίς να συσσωρεύονται μηνύματα και να μειώνει την κλίμακα κατά τη διάρκεια της αδράνειας για να απελευθερώνει πόρους που δεν είναι πλέον απαραίτητοι.
Περίπτωση χρήσης: Αυτό το μοτίβο παρατηρείται συχνά σε ροές εργασιών επεξεργασίας εικόνας, όπου τα batch jobs υποβάλλονται σε επεξεργασία περιοδικά, ακολουθούμενα από περιόδους αδράνειας. Για παράδειγμα, ένα σύστημα επιτήρησης που επεξεργάζεται εικόνες μαζικά κάθε λίγα λεπτά.

5.5.2 Χρήση CPU (Συνολική χρήση CPU για υπηρεσίες καταναλωτή)



Σχημα 5.19: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Συνολική χρήση CPU

Από το γράφημα, η χρήση της CPU για τις τρεις καταναλωτικές υπηρεσίες αυξομειώνεται σημαντικά κατά τη διάρκεια του πειράματος. Είναι ορατές οι κορυφές και τα χαμηλά επίπεδα στη χρήση της CPU, γεγονός που υποδηλώνει διακυμάνσεις στο φορτίο του συστήματος. Η χρήση της CPU αυξάνεται καθώς το σύστημα χειρίζεται εισερχόμενα αιτήματα και μειώνεται όταν υπάρχουν περίοδοι αδράνειας. Αυτή η συμπεριφορά αντικατοπτρίζει την ικανότητα του συστήματος να προσαρμόζει δυναμικά τη χρήση των πόρων του καθώς επεξεργάζεται διαφορετικά φορτία, γεγονός που ευθυγραμμίζεται με τα χαρακτηριστικά του αλγορίθμου AIMD (Additive Increase Multiplicative Decrease) που υλοποιήσατε.

Βασικές παρατηρήσεις:

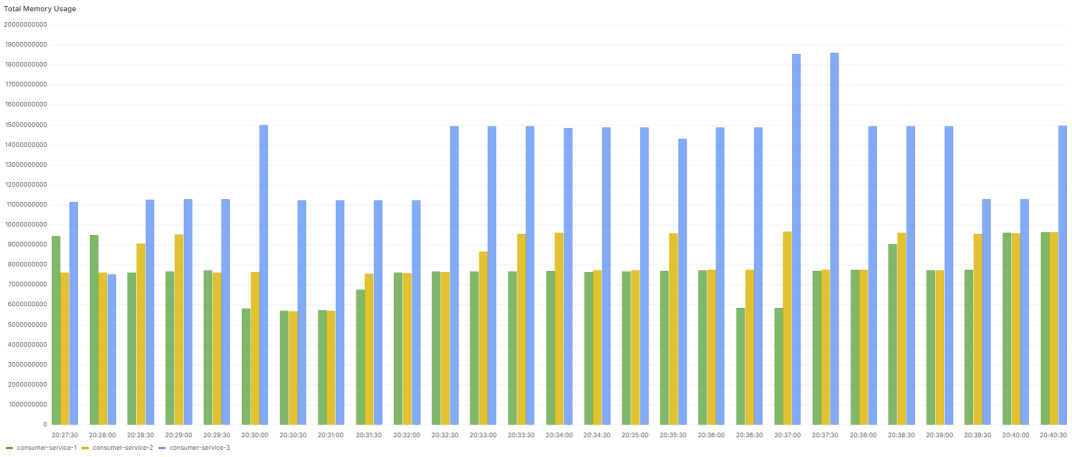
- Και οι τρεις υπηρεσίες παρουσιάζουν παρόμοιες διακυμάνσεις, αλλά η υπηρεσία καταναλωτή-3 τείνει να χρησιμοποιεί ελαφρώς περισσότερη CPU από τις άλλες υπηρεσίες.
- Οι αιχμές CPU εμφανίζονται παράλληλα σε όλες τις υπηρεσίες, με την υπηρεσία καταναλωτή-2 και την υπηρεσία καταναλωτή-3 να παρουσιάζουν τα υψηλότερα επίπεδα χρήσης σε ορισμένα σημεία.
- Αυτό το γράφημα δείχνει ότι οι υπηρεσίες διαχειρίζονται καλά το εισερχόμενο φορτίο και κλιμακώνουν δυναμικά τη χρήση της CPU τους.

Αναμενόμενη συμπεριφορά: Η χρήση CPU συσχετίζεται άμεσα με το ρυθμό παράδοσης μηνυμάτων και τον αριθμό των αντιγράφων. Καθώς η χρήση της CPU κορυφώνεται, υποδεικνύει ότι οι υπηρεσίες επεξεργάζονται μηνύματα, ενώ οι πτώσεις στη χρήση συμπίπτουν με περιόδους αδράνειας όπου επεξεργάζονται λιγότερα μηνύματα.

5.5.3 Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Η κατανάλωση μνήμης, όπως φαίνεται στο δεύτερο γράφημα, παρουσιάζει μια πιο σταθερή αλλά σταδιακά αυξανόμενη τάση. Υπάρχουν σταθερές αιχμές και βυθίσεις στη χρήση μνήμης και στις τρεις υπηρεσίες, με την υπηρεσία καταναλωτή-3 να εμφανίζει και πάλι την υψηλότερη χρήση μνήμης καθ' όλη τη διάρκεια της δοκιμής.

Βασικές παρατηρήσεις:



Σχημα 5.20: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Χρήση μνήμης

- Τα μοτίβα χρήσης μνήμης αντικατοπτρίζουν σταθερή αύξηση της κατανάλωσης πόρων, η οποία πιθανώς οφείλεται στη συσσώρευση αιτήσεων στην ουρά ή στην αύξηση του αριθμού των αντιγράφων.
- Ενώ η χρήση CPU κορυφώνεται απότομα, οι αυξήσεις της μνήμης φαίνονται πιο ομαλές, γεγονός που υποδηλώνει ότι το αποτύπωμα μνήμης αυξάνεται καθώς το σύστημα κλιμακώνει τις υπηρεσίες ή καθώς συσσωρεύονται ουρές μηνυμάτων.

Αναμενόμενη συμπεριφορά: Η χρήση της μνήμης θα πρέπει να αυξάνεται καθώς αυξάνεται ο αριθμός των μηνυμάτων στο σύστημα, αλλά δεν θα πρέπει να αυξομειώνεται τόσο δραματικά όσο η CPU. Εάν η κατανάλωση μνήμης αυξάνεται ανεξέλεγκτα, αυτό μπορεί να υποδεικνύει πρόβλημα με την επεξεργασία μηνυμάτων ή ανεπαρκή κλιμάκωση.

5.5.4 Παραδοθέντα μηνύματα (ρυθμοί εξερχόμενων μηνυμάτων)

Σχημα 5.21: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Παραδοθέντα μηνύματα

Το γράφημα που απεικονίζει τον αριθμό των μηνυμάτων που παραδίδονται ανά δευτερόλεπτο δείχνει ένα σαφές κυκλικό μοτίβο επεξεργασίας μηνυμάτων. Κάθε κύκλος αρχίζει με μια αιχμή που ακολουθείται από μια σταδιακή πτώση του ρυθμού παράδοσης μηνυμάτων.

Βασικές παρατηρήσεις:

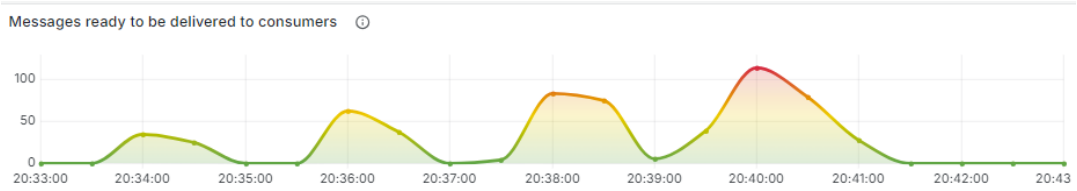
- Ο ρυθμός παράδοσης ποικίλλει σημαντικά, με κορύφωση γύρω στα 10 μηνύματα ανά δευτερόλεπτο και στη συνέχεια πτώση κοντά στο μηδέν κατά τη διάρκεια περιόδων αδράνειας.

- Αυτό συνάδει με την ικανότητα του συστήματος να διαχειρίζεται εκρήξεις φορτίου που ακολουθούνται από σύντομες περιόδους αδράνειας.
- Το μοτίβο αντανακλά πιθανώς την επίδραση της AIMD, όπου ο ρυθμός αυξάνεται καθώς τα μηνύματα επεξεργάζονται επιτυχώς και μειώνεται κατά τη διάρκεια περιόδων αδράνειας ή χαμηλότερου φορτίου.

Αναμενόμενη συμπεριφορά: Η ικανότητα του συστήματος να καθαρίζει γρήγορα τα μηνύματα κατά τη διάρκεια περιόδων υψηλού φόρτου και στη συνέχεια να σταθεροποιείται κατά τη διάρκεια περιόδων χαμηλού φόρτου είναι ενδεικτική της αποτελεσματικής κλιμάκωσης. Σε μια εφαρμογή πραγματικού χρόνου όπως η επεξεργασία εικόνων, θα περιμέναμε τέτοια μοτίβα φόρτου, όπου εκρήξεις δεδομένων (π.χ. πολλαπλές εικόνες που φτάνουν ταυτόχρονα) θα προκαλούσαν αιχμές στη δραστηριότητα επεξεργασίας.

5.5.5 Μηνύματα στην ουρά

Το γράφημα μεγέθους ουράς δείχνει μια κυκλική συσσώρευση και εξάντληση των μηνυμάτων που περιμένουν να επεξεργαστούν, που αντιστοιχεί στις περιόδους αυξημένου φορτίου.



Σχημα 5.22: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Μηνύματα στην ουρά

Βασικές παρατηρήσεις:

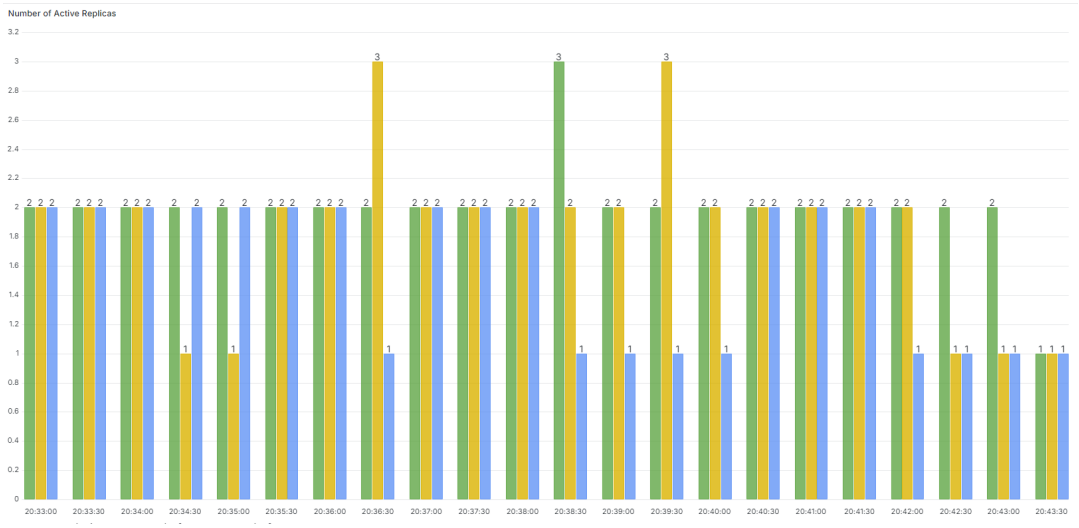
- Η συσσώρευση μηνυμάτων είναι περιοδική, με σαφείς αυξήσεις στο μέγεθος της ουράς που ακολουθούνται από περιόδους πλήρους εκκαθάρισης.
- Αυτή η κυκλική συμπεριφορά της ουράς υποδηλώνει ότι το σύστημα χειρίζεται αποτελεσματικά τις εκρήξεις εισερχόμενης κίνησης, αλλά δεν επιτρέπει τη συσσώρευση συσσωρευμένων μηνυμάτων για παρατεταμένες περιόδους.

Αναμενόμενη συμπεριφορά: Σε ένα σύστημα πραγματικού χρόνου, αναμένεται περιστασιακή συσσώρευση μηνυμάτων στην ουρά κατά τη διάρκεια εκρήξεων, αλλά το σύστημα θα πρέπει να καθαρίζει την ουρά τακτικά. Το γεγονός ότι η ουρά επανέρχεται σταθερά στο μηδέν υποδηλώνει ότι το σύστημα κλιμάκωσης λειτουργεί όπως προβλέπεται, διασφαλίζοντας ότι κανένα μήνυμα δεν παραμένει ανεπεξέργαστο για πολύ μεγάλο χρονικό διάστημα.

Αριθμός Ενεργών Αντιγράφων

Ο αριθμός των ενεργών αντιγράφων στις τρεις υπηρεσίες μεταβάλλεται δυναμικά κατά τη διάρκεια της δοκιμής, όπως φαίνεται στο γράφημα συμπεριφοράς κλιμάκωσης.

Βασικές παρατηρήσεις:



Σχημα 5.23: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Αριθμός Ενεργών Αντιγράφων

- Ο αριθμός των αντιγράφων κυμαίνεται μεταξύ 1, 2 και περιστασιακά 3 αντιγράφων, δείχνοντας ότι το σύστημα ανταποκρίνεται στις αλλαγές του φορτίου με κλιμάκωση προς τα πάνω και προς τα κάτω, ανάλογα με τις ανάγκες.

- Η υπηρεσία καταναλωτή-3 παρουσιάζει τη μεγαλύτερη δραστηριότητα κλιμάκωσης, γεγονός που υποδηλώνει ότι ενδέχεται να λαμβάνει μεγαλύτερο μερίδιο του φόρτου εργασίας.

Αναμενόμενη Συμπεριφορά: Αυτή η δυναμική συμπεριφορά κλιμάκωσης είναι κρίσιμη για τη διατήρηση της αποδοτικότητας του συστήματος και την ανταπόκριση στις μεταβολές του φορτίου. Σε εφαρμογές όπως η επεξεργασία εικόνας, όπου ο αριθμός των εισερχόμενων εργασιών μπορεί να ποικίλλει σημαντικά, είναι απαραίτητο για το σύστημα να προσαρμόζει τον αριθμό των ενεργών αντιγράφων για να διατηρεί την απόδοση.

Χρόνος Απόκρισης

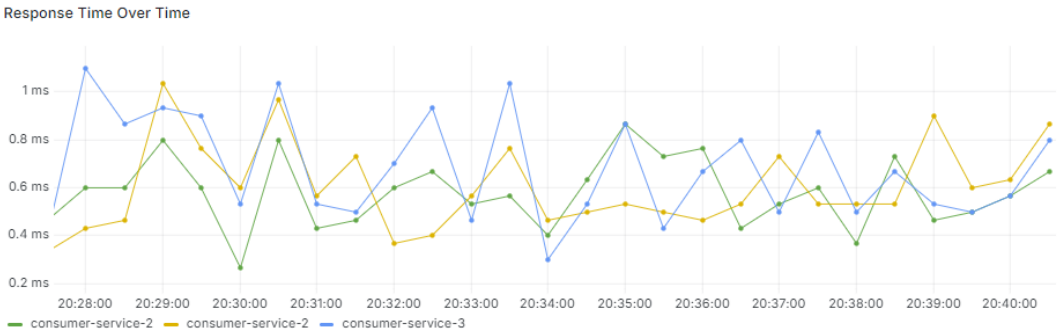
Το γράφημα του χρόνου απόκρισης δείχνει διακυμάνσεις σε όλες τις υπηρεσίες, με τους περισσότερους χρόνους απόκρισης να κυμαίνονται μεταξύ 0,3 και 1 χιλιοστού του δευτερολέπτου. Αυτές οι διακυμάνσεις αντιστοιχούν σε αλλαγές στη χρήση της CPU και τον αριθμό των αντιγράφων.

Βασικές Παρατηρήσεις:

- Οι αιχμές στον χρόνο απόκρισης συσχετίζονται με περιόδους υψηλότερου φόρτου, όπως δείχνει η χρήση της CPU και οι ρυθμοί παράδοσης μηνυμάτων.

- Και οι τρεις υπηρεσίες εμφανίζουν σχετικά παρόμοιους χρόνους απόκρισης, γεγονός που υποδηλώνει ομοιόμορφη κατανομή του φορτίου.

Αναμενόμενη Συμπεριφορά: Οι κυμαινόμενοι χρόνοι απόκρισης είναι τυπικοί για συστήματα υπό μεταβλητό φορτίο. Η σταθερή επιστροφή σε χαμηλότερους χρόνους απόκρισης κατά τη διάρκεια περιόδων αδράνειας δείχνει ότι το σύστημα μπορεί να διατηρεί χαμηλή καθυστέρηση όταν η ζήτηση μειώνεται. Σε

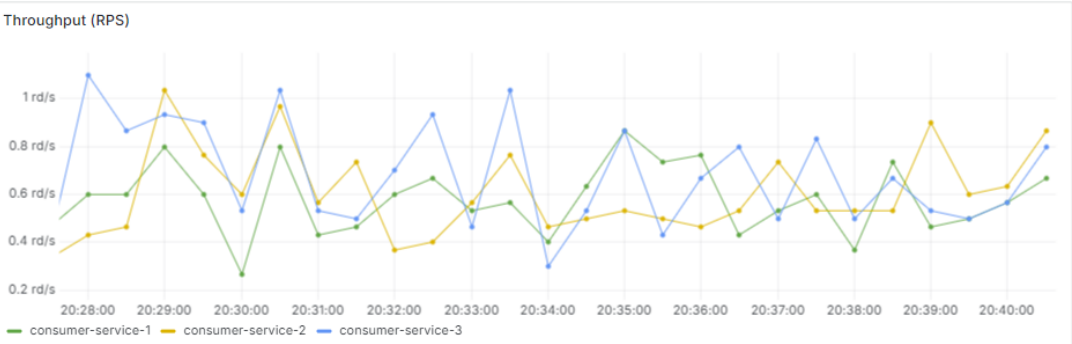


Σχημα 5.24: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Χρόνος Απόκρισης

εφαρμογές πραγματικού χρόνου, όπως η επεξεργασία εικόνας, οι χαμηλοί χρόνοι απόκρισης είναι απαραίτητοι για την καλή εμπειρία του χρήστη, ειδικά σε εργασίες όπως η ανίχνευση αντικειμένων ή το φιλτράρισμα.

Απόδοση Αιτήσεων (RPS)

Το γράφημα της απόδοσης (RPS) δείχνει το ποσοστό των αιτήσεων που υποβάλλονται σε επεξεργασία ανά δευτερόλεπτο, και αντικατοπτρίζει τη συμπεριφορά της χρήσης της CPU και των ρυθμών παράδοσης μηνυμάτων.



Σχημα 5.25: Πειράματα Κλιμάκωσης με KPA: Υψηλό φορτίο με ανάκαμψη σε αδράνεια : Απόδοση Αιτήσεων (RPS)

Βασικές Παρατηρήσεις:

- Η απόδοση κυμαίνεται μεταξύ 0,2 και 1 αίτησης ανά δευτερόλεπτο, με αιχμές που αντιστοιχούν σε περιόδους αυξημένης χρήσης της CPU και περισσότερων ενεργών αντιγράφων.

Αναμενόμενη Συμπεριφορά: Αυτό το γράφημα αντικατοπτρίζει την ικανότητα του συστήματος να διατηρεί την απόδοση υπό διαφορετικές συνθήκες φόρτου. Σε σενάρια πραγματικού χρόνου, η υψηλή απόδοση κατά τις εκρήξεις και η σταθερότητα κατά τις περιόδους αδράνειας είναι απαραίτητες για την εξισορρόπηση της χρήσης πόρων και της απόδοσης.

Συμπέρασμα

Τα αποτελέσματα αυτής της δοκιμής παρέχουν ισχυρές ενδείξεις ότι το σύστημα κλιμάκωσης λειτουργεί όπως προβλέπεται. Οι μηχανισμοί δρομολόγησης

και ελέγχου εισδοχής που βασίζονται στην AIMD επιτρέπουν στο σύστημα να διαχειρίζεται αποτελεσματικά την εκρηκτική κυκλοφορία, αποφεύγοντας παράλληλα την υπερβολική κατανάλωση πόρων κατά τη διάρκεια περιόδων αδράνειας. Αυτή η συμπεριφορά είναι κρίσιμη σε εφαρμογές επεξεργασίας εικόνας, όπου η κυκλοφορία μπορεί να φτάνει κατά κύματα, και το σύστημα πρέπει να κλιμακώνεται για να επεξεργάζεται τις εργασίες χωρίς να υπερφορτώνει τους πόρους ή να προκαλεί υπερβολικές καθυστερήσεις.

Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας

Εγκατάσταση δοκιμής: 1:60, 2:60, 3:60, 4:60, 0:30, 3:60, 2:60, 1:60, 0:30
Περιγραφή: Αυτή η δοκιμή εφάρμοσε εναλλασσόμενο φορτίο, που κυμαινόταν από 1 έως 4 αιτήσεις ανά δευτερόλεπτο, με σύντομες φάσεις αδράνειας 30 δευτερολέπτων που παρεμβάλλονταν μεταξύ περιόδων υψηλότερου φορτίου.
Σκοπός: Ο πρωταρχικός στόχος ήταν να αξιολογηθεί η ανταπόκριση του συστήματος σε σταδιακά αυξανόμενα φορτία κίνησης και η ικανότητά του να μειώνει γρήγορα την κλίμακα και να διαγράφει εργασίες κατά τη διάρκεια σύντομων περιόδων αδράνειας. Η δοκιμή εξέτασε επίσης την απόδοση του συστήματος σε ταχέως μεταβαλλόμενες συνθήκες κυκλοφορίας.
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να προσαρμόζει δυναμικά την κλιμάκωσή του ώστε να διαχειρίζεται τον αυξανόμενο και μειούμενο φόρτο, διατηρώντας παράλληλα την ελάχιστη συσσώρευση μηνυμάτων κατά τη διάρκεια των περιόδων αιχμής και την άμεση ανάκαμψη κατά τις φάσεις αδράνειας.
Περίπτωση χρήσης: Αυτό το μοτίβο μπορεί να παρατηρηθεί σε πλατφόρμες ανάλυσης εικόνας που βασίζονται σε νέφος, όπου οι χρήστες υποβάλλουν εργασίες σε ριπές, οδηγώντας σε εναλλασσόμενες περιόδους υψηλής και χαμηλής ζήτησης.

Χρήση CPU (Συνολική χρήση CPU για τις Υπηρεσίες Καταναλωτών): Από το γράφημα, η χρήση της CPU στις τρεις υπηρεσίες καταναλωτών δείχνει σαφή εναλλασσόμενα μοτίβα φόρτου, αυξανόμενη και μειούμενη ταυτόχρονα. Αυτό αντικατοπτρίζει το εναλλασσόμενο φορτίο που δέχεται το σύστημα, καθώς προσαρμόζεται δυναμικά στα εισερχόμενα αιτήματα και την ικανότητα κλιμάκωσης του συστήματος.

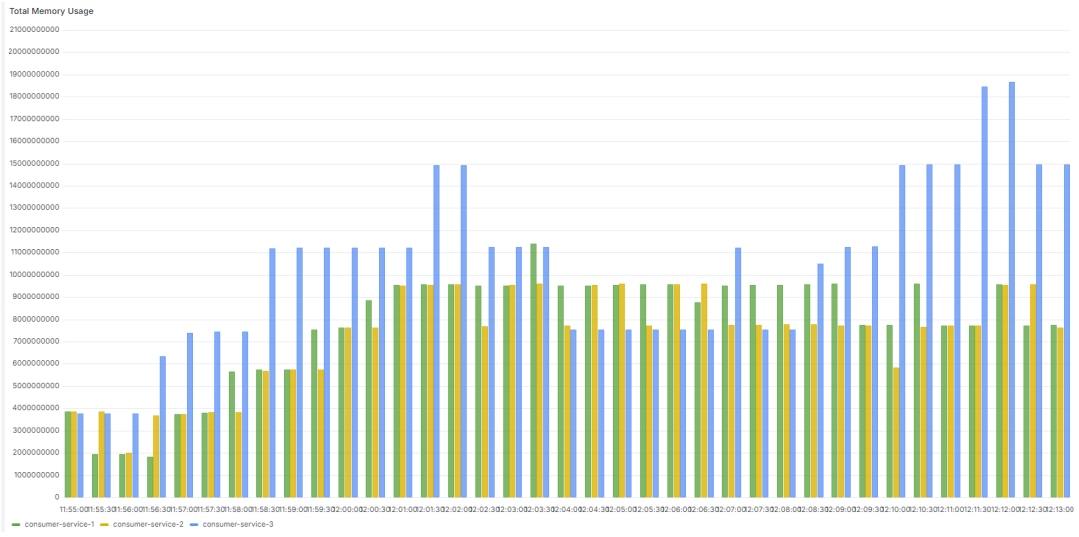
Σχημα 5.26: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Χρήση CPU

Βασικές Παρατηρήσεις:

- **Παρόμοιες τάσεις μεταξύ των υπηρεσιών**: Όλες οι υπηρεσίες ακολουθούν ένα συγχρονισμένο μοτίβο στη χρήση CPU, αντανακλώντας κοινή κατανομή του φόρτου. Η υπηρεσία καταναλωτή-3 εμφανίζει ελαφρώς υψηλότερες κορυφές, γεγονός που υποδηλώνει ότι ίσως επεξεργάζεται περισσότερα δεδομένα.
- **Σύντομες Περίοδοι Αδράνειας**: Υπάρχουν ορατές κοιλάδες που δείχνουν σύντομες περιόδους αδράνειας, όπου οι υπηρεσίες επεξεργάζονται λιγότερα μηνύματα. Αυτό είναι σύμφωνο με τη φύση της δοκιμής που περιλαμβάνει εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας.

Αναμενόμενη Συμπεριφορά: Η χρήση της CPU θα πρέπει να κλιμακώνεται ανάλογα με το εισερχόμενο φορτίο. Κατά τις φάσεις υψηλού φορτίου, αναμένονται αιχμές στη χρήση της CPU, ενώ κατά τις περιόδους αδράνειας αναμένεται πτώση. Αυτή η συμπεριφορά εξασφαλίζει αποτελεσματική διαχείριση πόρων χωρίς περιττή κατανάλωση κατά τις περιόδους χαμηλής κυκλοφορίας.

Χρήση Μνήμης (Συνολική Χρήση Μνήμης για τις Υπηρεσίες Καταναλωτών): Το γράφημα χρήσης μνήμης δείχνει ένα πιο σταθερό και συνεπές μοτίβο με εμφανείς αυξήσεις κατά τις περιόδους υψηλού φορτίου. Η κατανάλωση μνήμης φαίνεται πιο σταθερή σε σύγκριση με τη χρήση της CPU, με λιγότερες αιχμές και βυθίσεις.



Σχημα 5.27: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Χρήση Μνήμης

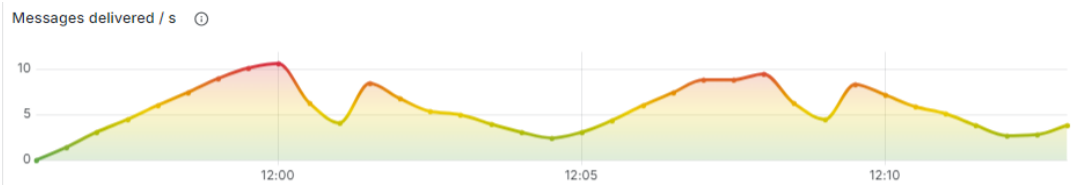
Βασικές Παρατηρήσεις:

- **Σταθερή Αύξηση**: Η υπηρεσία καταναλωτή-3 καταναλώνει σταθερά περισσότερη μνήμη από τις άλλες υπηρεσίες, αντανακλώντας το υψηλότερο φορτίο της.
- **Ομαλές Διακυμάνσεις**: Ενώ η CPU δείχνει απότομες μεταβάσεις, η χρήση της μνήμης αυξάνεται και μειώνεται πιο ομαλά, υποδεικνύοντας αποδοτική διαχείριση μνήμης.

Αναμενόμενη Συμπεριφορά: Η χρήση μνήμης αναμένεται να αυξάνεται καθώς οι υπηρεσίες συσσωρεύουν περισσότερες εργασίες ή κλιμακώνονται. Οι σχετικά ομαλές διακυμάνσεις σε σχέση με την CPU δείχνουν ότι οι υπηρεσίες δεν

παρουσιάζουν απότομες αυξήσεις στην κατανομή μνήμης, κάτι που ευθυγραμμίζεται με την αποδοτική κλιμάκωση του συστήματος.

Παραδοθέντα Μηνύματα (Ρυθμοί Εξερχόμενων Μηνυμάτων): Το γράφημα ρυθμού παράδοσης αντικατοπτρίζει κυκλικές κορυφές στα εξερχόμενα μηνύματα, σύμφωνες με τις εκρήξεις φορτίου που ακολουθούνται από σύντομες περιόδους αδράνειας.



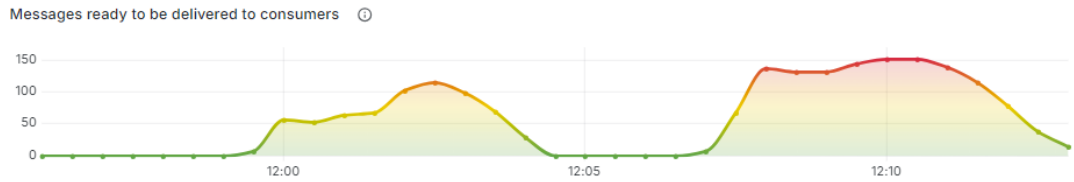
Σχημα 5.28: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Παραδοθέντα Μηνύματα

Βασικές Παρατηρήσεις:

- **Κανονικές Κορυφές και Κοιλάδες:** Ο ρυθμός παράδοσης κορυφώνεται κοντά στα 10 μηνύματα ανά δευτερόλεπτο κατά τις περιόδους υψηλού φορτίου και πέφτει κοντά στο μηδέν κατά τις περιόδους αδράνειας.
- **Συγχρονισμένη Παράδοση:** Ο ρυθμός παράδοσης συνδέεται άμεσα με την επεξεργαστική ικανότητα και τη χρήση της CPU.

Αναμενόμενη Συμπεριφορά: Σε σενάριο εκρηκτικού φορτίου, αναμένονται διακυμάνσεις στην απόδοση, με ταχεία επεξεργασία κατά τις φάσεις υψηλού φορτίου και χαμηλότερη επεξεργασία στις περιόδους αδράνειας.

Μηνύματα στην Ουρά: Το γράφημα της ουράς αποκαλύπτει ένα μοτίβο περιοδικής συσσώρευσης και εκκαθάρισης μηνυμάτων, ευθυγραμμισμένο με τις φάσεις εναλλασσόμενου φορτίου της δοκιμής.



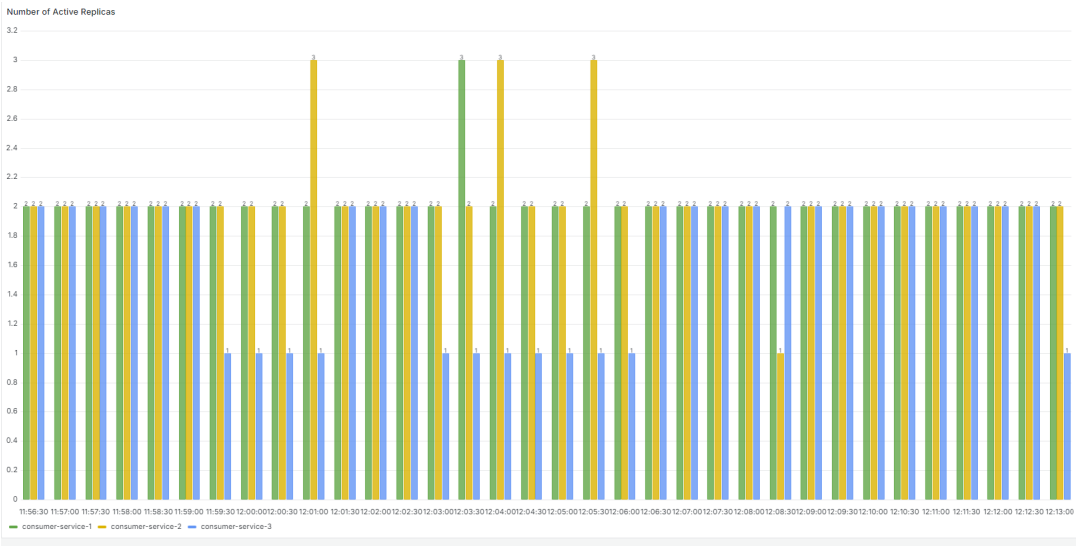
Σχημα 5.29: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Μηνύματα στην Ουρά

Βασικές Παρατηρήσεις:

- **Κυκλική Συσσώρευση Ουράς:** Το μέγεθος της ουράς αυξάνεται κατά τις περιόδους υψηλού φορτίου και μειώνεται σταδιακά καθώς το σύστημα επεξεργάζεται τα εισερχόμενα μηνύματα.
- **Εκκαθάριση Ουράς:** Το σύστημα καταφέρνει να εκκαθαρίζει την ουρά κατά τις περιόδους αδράνειας, αποδεικνύοντας ότι μπορεί να διαχειρίζεται τις εκρήξεις χωρίς να αφήνει ακατέργαστα μηνύματα.

Αναμενόμενη Συμπεριφορά: Η ουρά θα πρέπει να δείχνει περιοδική συσσώρευση κατά τις περιόδους υψηλού φορτίου, αλλά να επανέρχεται στο μηδέν κατά τις περιόδους αδράνειας. Αυτό δείχνει ότι το σύστημα διαχειρίζεται την κυκλοφορία αποτελεσματικά.

Αριθμός Ενεργών Αντιγράφων: Το γράφημα του αριθμού ενεργών αντιγράφων δείχνει τη δυναμική συμπεριφορά κλιμάκωσης, με τον αριθμό των αντιγράφων να προσαρμόζεται ανάλογα με το φορτίο του συστήματος.



Σχημα 5.30: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Αριθμός Ενεργών Αντιγράφων

Βασικές Παρατηρήσεις:

- **Διακυμάνσεις Αντιγράφων**: Ο αριθμός των αντιγράφων κυμαίνεται μεταξύ 1 και 3, ανάλογα με το φορτίο.
- **Κλιμάκωση για συγκεκριμένες υπηρεσίες**: Η υπηρεσία καταναλωτή-3 παρουσιάζει τη μεγαλύτερη δραστηριότητα κλιμάκωσης, υποδηλώνοντας ότι χειρίζεται μεγαλύτερο μερίδιο του φόρτου.

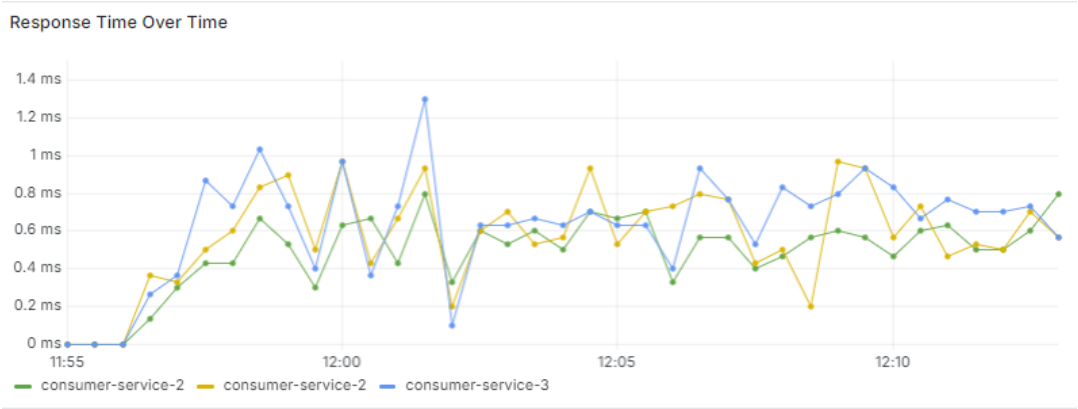
Αναμενόμενη Συμπεριφορά: Η δυναμική κλιμάκωση αντιγράφων είναι κρίσιμη για την αποδοτική διαχείριση των μεταβαλλόμενων φορτίων.

Χρόνος Απόκρισης: Το γράφημα χρόνου απόκρισης δείχνει διακυμάνσεις με αυξήσεις στις περιόδους υψηλού φορτίου.

Βασικές Παρατηρήσεις:

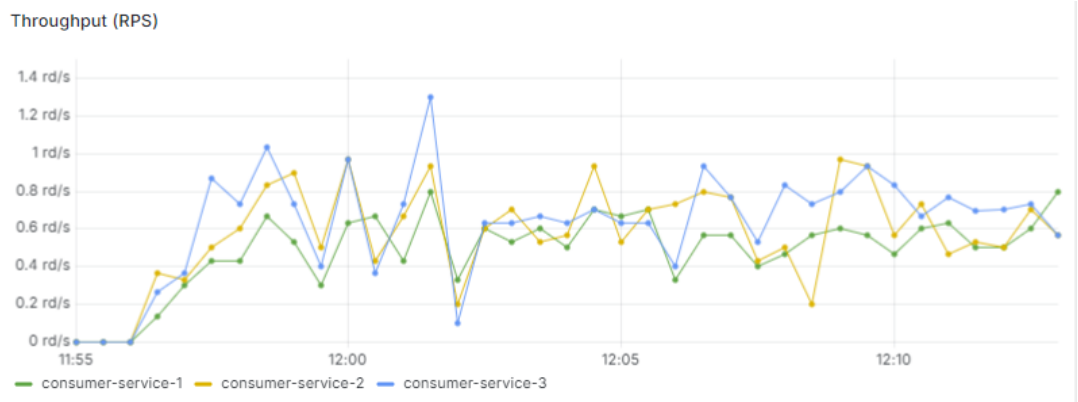
- **Συσχέτιση Χρόνου Απόκρισης με Φορτίο**: Οι αιχμές στο χρόνο απόκρισης ευθυγραμμίζονται με τις αυξήσεις στη χρήση της CPU.
- **Σχετικά χαμηλή καθυστέρηση**: Παρά τις διακυμάνσεις, οι χρόνοι απόκρισης παραμένουν χαμηλοί, κάτω από 1 χιλιοστό του δευτερολέπτου.

Αναμενόμενη Συμπεριφορά: Οι διακυμάνσεις είναι αναμενόμενες κατά τις περιόδους υψηλού φορτίου, αλλά η επιστροφή σε χαμηλότερους χρόνους υποδηλώνει αποδοτική απόδοση του συστήματος.



Σχημα 5.31: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Χρόνος Απόκρισης

Απόδοση Αιτήσεων (RPS): Το γράφημα του ρυθμού μετάδοσης αντικατοπτρίζει τη συμπεριφορά της χρήσης της CPU και των ρυθμών παράδοσης μηνυμάτων.



Σχημα 5.32: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο Φορτίο με Σύντομες Περιόδους Αδράνειας : Απόδοση Αιτήσεων (RPS)

Βασικές Παρατηρήσεις:

- **Διακυμάνσεις στη Ρυθμαπόδοση:** Η απόδοση κυμαίνεται μεταξύ 0,2 και 1,4 αιτήσεων ανά δευτερόλεπτο.

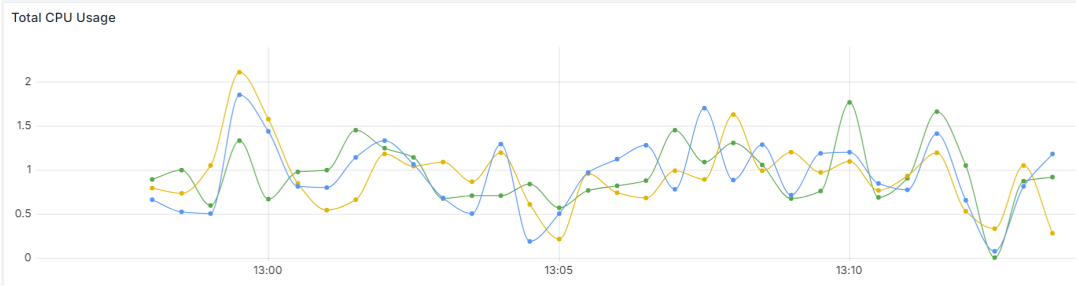
Αναμενόμενη Συμπεριφορά: Το σύστημα καταφέρνει να διατηρεί υψηλή απόδοση κατά τις φάσεις υψηλού φορτίου και να μειώνει την απόδοση κατά τις περιόδους αδράνειας.

Συμπέρασμα: Τα αποτελέσματα αυτής της δοκιμής δείχνουν ότι το σύστημα διαχειρίζεται αποτελεσματικά τις εναλλασσόμενες εκρήξεις φορτίου με σύντομες περιόδους αδράνειας. Η χρήση CPU και μνήμης προσαρμόζεται δυναμικά, οι ρυθμοί παράδοσης και τα μεγέθη της ουράς παρουσιάζουν κυκλικά μοτίβα και ο αριθμός ενεργών αντιγράφων κλιμακώνεται για να ανταποκριθεί στις απαιτήσεις του φόρτου εργασίας.

5.5.6 Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας

Εγκατάσταση δοκιμής: 1:30, 3:15, 5:15, 1:30, 4:15, 2:15, 1:15
Περιγραφή: Αυτή η δοκιμή εισήγαγε σύντομες εκρήξεις κυκλοφορίας που κυμαίνονταν από 1 έως 5 αιτήσεις ανά δευτερόλεπτο, με εκρήξεις διάρκειας 15 έως 30 δευτερολέπτων η κάθε μία.
Σκοπός: Ο στόχος ήταν να αξιολογηθεί πόσο καλά το σύστημα χειρίζεται τις αιχμές κίνησης υψηλής έντασης και αν μπορεί να αποτρέψει τη συσσώρευση μηνυμάτων κατά τη διάρκεια αυτών των σύντομων αιφνιδιασμών. Η δοκιμή αυτή επικεντρώθηκε στην ικανότητα του συστήματος να αυξάνει και να μειώνει γρήγορα την κλίμακα κατά τη διάρκεια σύντομων εκρήξεων.
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να είναι σε θέση να κλιμακώνεται δυναμικά ώστε να ανταποκρίνεται στις απαιτήσεις των εκρήξεων χωρίς σημαντικές καθυστερήσεις ή συσσώρευση μηνυμάτων, εκκαθαρίζοντας αποτελεσματικά τις εργασίες κατά την άφιξή τους.
Περίπτωση χρήσης: Αυτό το μοτίβο μπορεί να παρατηρηθεί σε πλατφόρμες επεξεργασίας εικόνας σε πραγματικό χρόνο στα μέσα κοινωνικής δικτύωσης, όπου εμφανίζονται αιχμές κυκλοφορίας κατά τη διάρκεια σημαντικών γεγονότων ή κυκλοφοριών λειτουργιών, ακολουθούμενες από χαμηλότερους αλλά σταθερούς ρυθμούς κυκλοφορίας.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)



Σχημα 5.33: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Συνολική χρήση CPU

Από το γράφημα, είναι προφανές ότι η χρήση της CPU αυξομειώνεται έντονα κατά τη διάρκεια των εκρήξεων υψηλής έντασης. Υπάρχουν σαφείς αιχμές σε περιόδους όπου οι ρυθμοί ριπής αυξάνονται, οι οποίες στη συνέχεια πέφτουν όταν η ριπή τελειώνει. Και οι τρεις υπηρεσίες παρουσιάζουν παρόμοια συμπεριφορά στην κατανάλωση CPU, αλλά η υπηρεσία καταναλωτής-2 εμφανίζει ελαφρώς υψηλότερες αιχμές κατά τη διάρκεια των πιο εντατικών ριπών (γύρω στις 13:00), ακολουθούμενη στενά από την υπηρεσία καταναλωτής-1 και την υπηρεσία καταναλωτής-3. Αυτό δείχνει ότι το σύστημα κλιμακώνεται δυναμικά και ανταποκρίνεται στις σύντομες εκρήξεις αυξάνοντας την κατανάλωση CPU για να αντιμετωπίσει το αυξημένο φορτίο.

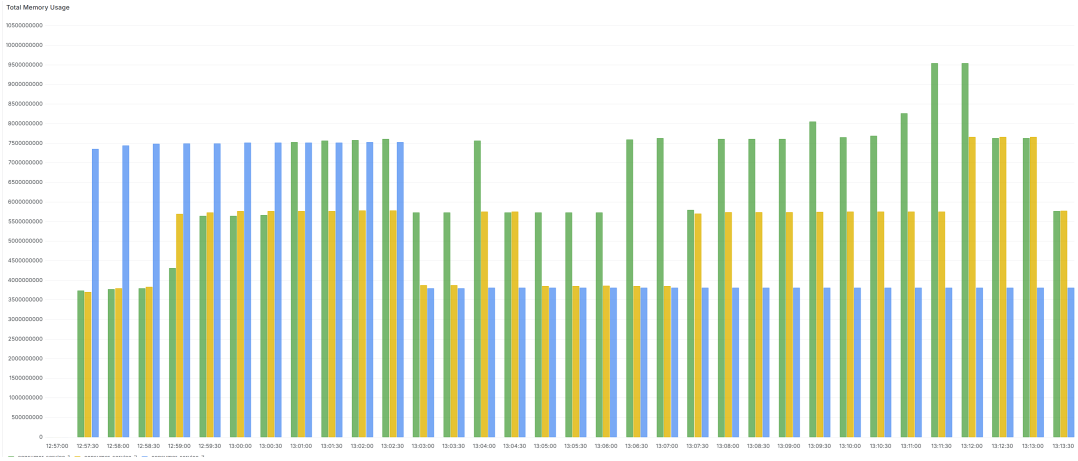
Βασικές παρατηρήσεις:

- Και οι τρεις υπηρεσίες επεξεργάζονται παρόμοια φορτία, με την υπηρεσία καταναλωτής-2 να χρησιμοποιεί ελαφρώς περισσότερη CPU κατά τις περιόδους αιχμής.
- Οι αιχμές στη χρήση CPU ευθυγραμμίζονται με τις εκρήξεις κυκλοφορίας, γεγονός που αποδεικνύει τη δυναμική κατανομή των πόρων.

- Το σύστημα φαίνεται να χειρίζεται αποτελεσματικά αυτές τις εκρήξεις, καθώς η χρήση της CPU μειώνεται μετά από κάθε έκρηξη, αποφεύγοντας τη συνεχή υψηλή χρήση ή την υπερθέρμανση.

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται ως απόκριση στην αυξημένη κυκλοφορία κατά τη διάρκεια των εκρήξεων και να επιστρέφει στα βασικά επίπεδα κατά τη διάρκεια των περιόδων αδράνειας. Ο αλγόριθμος AIMD (Additive Increase Multiplicative Decrease) πιθανώς βοηθά το σύστημα να ρυθμίσει την κατανάλωση CPU με τρόπο που αποτρέπει την υπερκατανομή σε περιόδους χαμηλότερης ζήτησης.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)



Σχημα 5.34: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Χρήση μνήμης

Το γράφημα χρήσης μνήμης δείχνει μια πιο συνεπή αλλά σταδιακή αύξηση της κατανάλωσης μνήμης, ιδίως για την υπηρεσία καταναλωτής-3. Οι αιχμές και οι πτώσεις συμβαίνουν παράλληλα με τις εκρήξεις κίνησης, αν και οι αλλαγές δεν είναι τόσο δραματικές όσο αυτές που παρατηρούνται στη χρήση της CPU. Η υπηρεσία καταναλωτής-3 χρησιμοποιεί σταθερά περισσότερη μνήμη σε σύγκριση με τις άλλες υπηρεσίες, γεγονός που μπορεί να αντικατοπτρίζει διαφορές στο φόρτο εργασίας ή στο πόσο αποτελεσματικά διαχειρίζεται τη μνήμη αυτή η υπηρεσία.

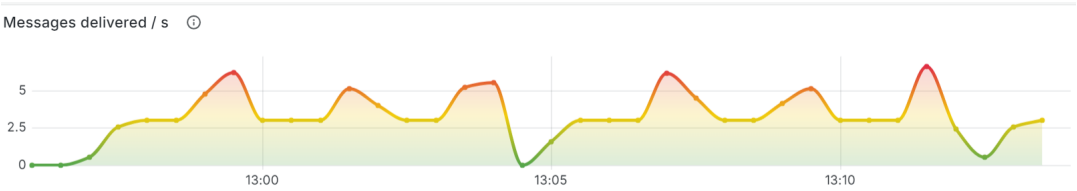
Βασικές παρατηρήσεις:

- Η χρήση μνήμης αυξάνεται σταδιακά, με την υπηρεσία καταναλωτής-3 να έχει υψηλότερη κατανάλωση καθ' όλη τη διάρκεια των εκρήξεων.
- Το μοτίβο της μνήμης είναι πιο ομαλό σε σύγκριση με τη χρήση της CPU, υποδεικνύοντας ότι η κατανομή της μνήμης είναι λιγότερο ευμετάβλητη από την CPU κατά τη διάρκεια των εκρήξεων.

Αναμενόμενη συμπεριφορά: Η χρήση της μνήμης θα πρέπει να αυξάνεται όταν χειρίζεται περισσότερα μηνύματα, αλλά δεν θα πρέπει να αυξάνεται τόσο γρήγορα όσο η CPU. Μια συνεπής, σταδιακή αύξηση κατά τη διάρκεια των ριπών αντανακλά έναν ισορροπημένο μηχανισμό κατανομής μνήμης.

Παραδοθέντα μηνύματα (εξερχόμενοι ρυθμοί)

Το γράφημα ρυθμού παράδοσης μηνυμάτων δείχνει σαφείς κορυφές κατά τη διάρκεια κάθε έκρηξης κίνησης, με ρυθμούς παράδοσης που φτάνουν έως και 5 μηνύματα ανά δευτερόλεπτο. Αυτό το κυκλικό μοτίβο αντικατοπτρίζει την ικανότητα του συστήματος να επεξεργάζεται αποτελεσματικά τα μηνύματα κατά τη διάρκεια των εκρήξεων και στη συνέχεια να επιστρέφει σε σταθερή κατάσταση μόλις η έκρηξη υποχωρήσει.



Σχημα 5.35: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Παραδοθέντα μηνύματα

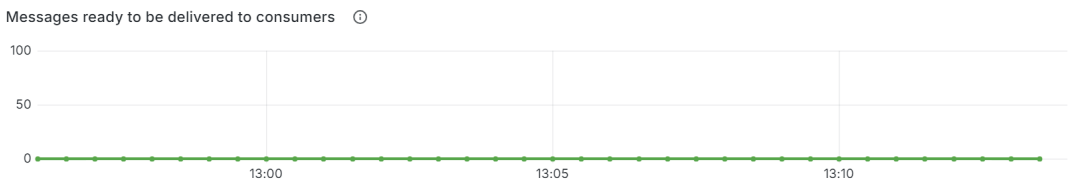
Βασικές παρατηρήσεις:

- Το σύστημα χειρίζεται καλά τις εκρήξεις κυκλοφορίας, χωρίς σημαντικές καθυστερήσεις μεταξύ των εκρήξεων.
- Σε κάθε έκρηξη παρατηρείται αύξηση του ρυθμού παράδοσης μηνυμάτων, ακολουθούμενη από σταδιακή επιστροφή στο μηδέν κατά τη διάρκεια περιόδων χαμηλής ζήτησης.

Αναμενόμενη συμπεριφορά: Ο ρυθμός παράδοσης θα πρέπει να αυξάνεται ανάλογα με τα εισερχόμενα μηνύματα κατά τη διάρκεια των εκρήξεων και να μειώνεται καθώς η κίνηση υποχωρεί.

Μηνύματα στην ουρά

Είναι ενδιαφέρον ότι το γράφημα ουράς δεν δείχνει σημαντική συσσώρευση μηνυμάτων κατά τη διάρκεια των εκρήξεων. Αυτό υποδεικνύει ότι το σύστημα είναι σε θέση να επεξεργάζεται την εισερχόμενη κίνηση γρήγορα και αποτελεσματικά, αποτρέποντας τη δημιουργία συσσωρευμένων μηνυμάτων.



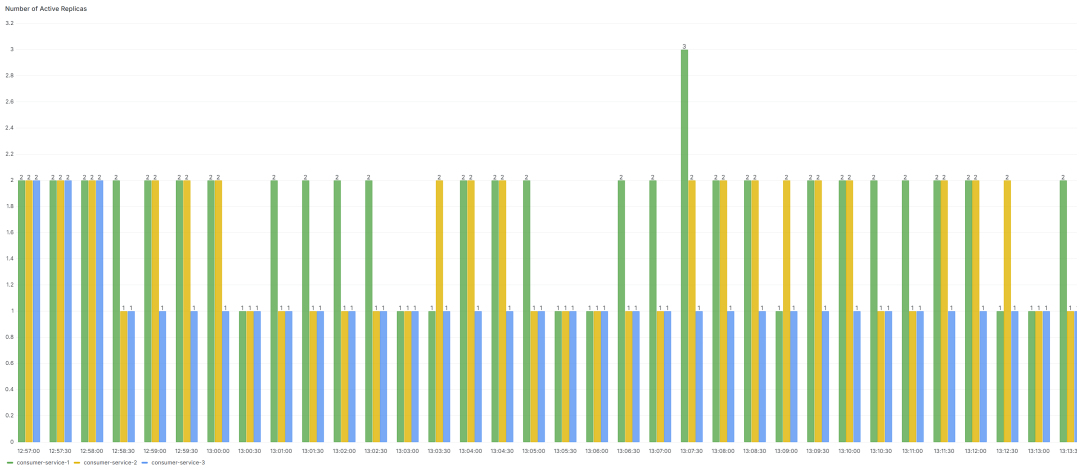
Σχημα 5.36: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Μηνύματα στην ουρά

Βασικές παρατηρήσεις:

- Δεν υπάρχουν ορατές συσσωρεύσεις στην ουρά, γεγονός που υποδηλώνει ότι τα μηνύματα επεξεργάζονται σχεδόν αμέσως μόλις φτάσουν.
- Το σύστημα συμβαδίζει με τις εισερχόμενες εκρήξεις, αποτρέποντας τη συσσώρευση μη επεξεργασμένων μηνυμάτων.

Αναμενόμενη συμπεριφορά: Σε ένα καλά ισορροπημένο σύστημα, τα μηνύματα δεν πρέπει να συσσωρεύονται στην ουρά για μεγάλα χρονικά διαστήματα.

Αριθμός ενεργών αντιγράφων



Σχημα 5.37: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Αριθμός ενεργών αντιγράφων

Ο αριθμός των ενεργών αντιγράφων αυξομειώνεται δυναμικά, με την υπηρεσία καταναλωτής-1 και την υπηρεσία καταναλωτής-3 να παρουσιάζουν περισσότερες διακυμάνσεις από την υπηρεσία καταναλωτής-2. Κάποια στιγμή, η υπηρεσία καταναλωτής-1 κλιμακώνεται σε 3 αντίγραφα, ενώ οι υπόλοιπες διατηρούν 2 ή λιγότερα.

Βασικές παρατηρήσεις:

- Τα αντίγραφα κλιμακώνονται δυναμικά, αντανακλώντας την προσπάθεια του συστήματος να προσαρμόσει την κατανομή των πόρων στις απαιτήσεις των εκρήξεων.
- Η υπηρεσία καταναλωτής-1 παρουσιάζει πιο επιθετική κλιμάκωση, πιθανώς λόγω του ότι δέχεται υψηλότερο φορτίο κατά τη διάρκεια ορισμένων εκρήξεων.

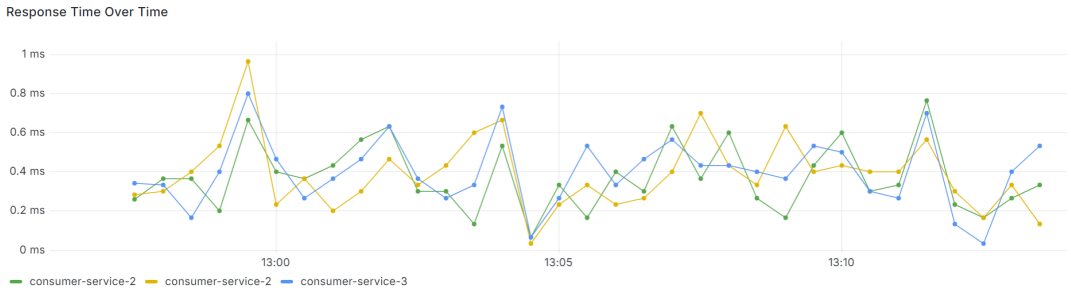
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να προσαρμόζει τον αριθμό των ενεργών αντιγράφων ως απόκριση στην ένταση της έκρηξης.

Χρόνος απόκρισης

Το γράφημα του χρόνου απόκρισης δείχνει αιχμές κατά τη διάρκεια των εκρήξεων, με τους χρόνους απόκρισης να κορυφώνονται λίγο κάτω από 1 χιλιοστό του δευτερολέπτου. Αυτό δείχνει ότι, ενώ το σύστημα είναι σε θέση να επεξεργάζεται αποτελεσματικά τα μηνύματα, υπάρχει κάποια καθυστέρηση κατά τη διάρκεια περιόδων υψηλής ζήτησης.

Βασικές παρατηρήσεις:

- Οι χρόνοι απόκρισης αυξάνονται κατά τη διάρκεια των εκρήξεων, αλλά παραμένουν σχετικά χαμηλοί συνολικά, με μέγιστη τιμή λίγο κάτω από 1 χιλιοστό του δευτερολέπτου.
- Η συνέπεια των χρόνων απόκρισης μεταξύ των υπηρεσιών υποδηλώνει ότι η εξισορρόπηση φορτίου λειτουργεί αποτελεσματικά.

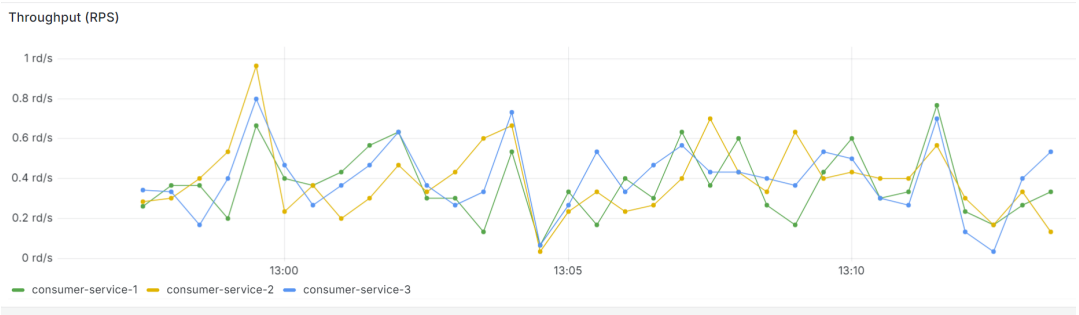


Σχημα 5.38: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Χρόνος απόκρισης

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμείνουν χαμηλοί, αλλά αναμένεται κάποια αύξηση κατά τη διάρκεια περιόδων υψηλής ζήτησης.

Απόδοση αιτήσεων (RPS)

Το γράφημα ρυθμού μετάδοσης αντικατοπτρίζει την ικανότητα του συστήματος να διαχειρίζεται αιτήσεις κατά τη διάρκεια εκρήξεων, με ρυθμούς που αυξάνονται σε περίπου 0,8 αιτήσεις ανά δευτερόλεπτο κατά τη διάρκεια των μεγαλύτερων φορτίων.



Σχημα 5.39: Πειράματα Κλιμάκωσης με KPA: Εκρήξεις Φορτίου Υψηλής Έντασης Βραχείας Διάρκειας : Απόδοση αιτήσεων (RPS)

Βασικές παρατηρήσεις:

- Η απόδοση αντικατοπτρίζει το μοτίβο των εισερχόμενων ριπών, με κορύφωση σε περιόδους υψηλού φόρτου και πτώση σε περιόδους χαμηλού φόρτου.
- Το σύστημα φαίνεται να διατηρεί σταθερή απόδοση, χειριζόμενο αποτελεσματικά τις εκρήξεις.

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να αυξάνεται κατά τη διάρκεια των εκρήξεων και να μειώνεται καθώς η κίνηση υποχωρεί.

Συμπέρασμα

Τα αποτελέσματα αυτής της δοκιμής αποδεικνύουν ότι το σύστημα μπορεί να διαχειριστεί αποτελεσματικά σύντομες εκρήξεις κυκλοφορίας υψηλής έντασης. Η χρήση της CPU αυξάνεται κατά τη διάρκεια των ριπών, αλλά η κατανάλωση

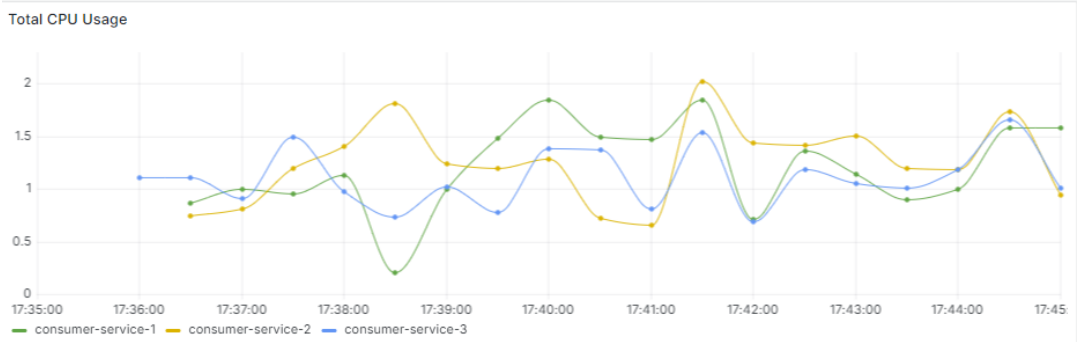
μνήμης παραμένει σταθερή. Το σύστημα επεξεργάζεται γρήγορα τα μηνύματα, αποτρέποντας τη συσσώρευση στην ουρά, ενώ κλιμακώνει δυναμικά τα αντίγραφα για να διαχειριστεί το αυξημένο φορτίο.

5.5.7 Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας

Εγκατάσταση δοκιμής: 1:60, 3:60, 5:60, 0:120
Περιγραφή: Αυτή η δοκιμή αύξησε σταδιακά το φορτίο κίνησης από 1 έως 5 αιτήσεις ανά δευτερόλεπτο σε μια περίοδο 60 δευτερολέπτων, ακολουθούμενη από μια εκτεταμένη φάση αδράνειας 120 δευτερολέπτων χωρίς φορτίο.
Σκοπός: Ο στόχος ήταν να αξιολογηθεί πόσο ομαλά το σύστημα χειρίζεται τις σταδιακές αυξήσεις του φορτίου και αν μπορεί να ανακάμψει πλήρως και να μειώσει την κλίμακα κατά τη διάρκεια εκτεταμένων περιόδων αδράνειας. Η δοκιμή αυτή αναδεικνύει την ικανότητα του συστήματος να διαχειρίζεται αποτελεσματικά τόσο τη σταδιακή κλιμάκωση όσο και τις φάσεις ανάκαμψης.
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να επιδεικνύει ομαλή κλιμάκωση με ελάχιστη συσσώρευση μηνυμάτων, ακολουθούμενη από αποτελεσματική απελευθέρωση πόρων κατά τη φάση αδράνειας.
Περίπτωση χρήσης: Μια τυπική περίπτωση χρήσης μπορεί να περιλαμβάνει συστήματα επεξεργασίας εικόνων σε δέσμη, όπου η κίνηση αυξάνεται αργά με την πάροδο του χρόνου καθώς συλλέγονται περισσότερες εικόνες, ακολουθούμενη από μια περίοδο αδράνειας όπου δεν επεξεργάζονται νέες εικόνες, όπως σε μια καθημερινή ροή εργασίας δημιουργίας αναφορών.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Από το γράφημα, μπορούμε να παρατηρήσουμε ότι η χρήση της CPU αυξομειώνεται καθώς το σύστημα ανταποκρίνεται στις σταδιακές αυξήσεις του φορτίου. Η χρήση της CPU κορυφώνεται κατά τη φάση φόρτου 5 RPS, αντικατοπτρίζοντας το υψηλότερο φορτίο κίνησης πριν μειωθεί κατά τη φάση αδράνειας. Η υπηρεσία καταναλωτή-3 παρουσιάζει ελαφρώς υψηλότερες κορυφές σε σύγκριση με τις υπηρεσίες καταναλωτή-1 και καταναλωτή-2, αλλά όλες οι υπηρεσίες ακολουθούν παρόμοια μοτίβα όσον αφορά την κλιμάκωση της κατανάλωσης CPU.



Σχημα 5.40: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Χρήση CPU

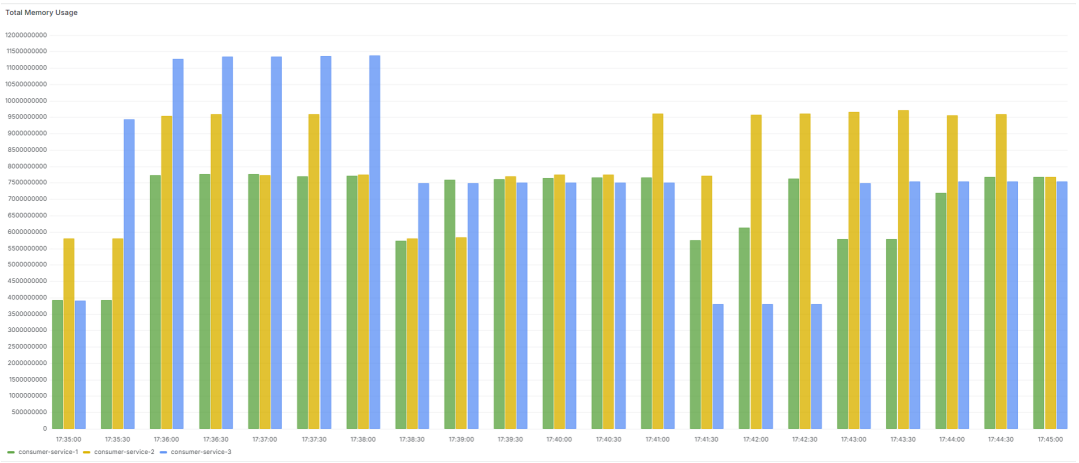
Βασικές παρατηρήσεις:

- Και οι τρεις υπηρεσίες καταναλωτών κλιμακώνουν τη χρήση της CPU τους σύμφωνα με τη σταδιακή αύξηση της κυκλοφορίας.
- Η υπηρεσία καταναλωτή-3 παρουσιάζει σταθερά υψηλότερες αιχμές στη χρήση CPU, υποδεικνύοντας μεγαλύτερο φόρτο εργασίας ή απαίτηση πόρων.
- Μετά την αύξηση του φορτίου, η χρήση CPU πέφτει σταδιακά στα βασικά επίπεδα κατά τη φάση αδράνειας, γεγονός που δείχνει αποτελεσματική απελευθέρωση πόρων.

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται καθώς το σύστημα ανεβαίνει για να διαχειριστεί το υψηλότερο φορτίο και στη συνέχεια να μειώνεται αποτελεσματικά κατά τη διάρκεια των περιόδων αδράνειας. Αυτή η συμπεριφορά ευθυγραμμίζεται με τις προσδοκίες για δοκιμές σταδιακού φορτίου, όπου το σύστημα πρέπει να κλιμακώνει δυναμικά τους πόρους χωρίς υπερβάσεις κατά τη διάρκεια των περιόδων αδράνειας.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Το γράφημα χρήσης μνήμης αναδεικνύει τη διαφορά στην κατανάλωση μνήμης μεταξύ των τριών υπηρεσιών, με την υπηρεσία καταναλωτή-3 να καταναλώνει τη μεγαλύτερη μνήμη καθ' όλη τη διάρκεια της δοκιμής. Η χρήση μνήμης αυξάνεται σταθερά καθώς αυξάνεται το φορτίο, και μπορούμε να παρατηρήσουμε ότι η υπηρεσία-καταναλωτής-1 και η υπηρεσία-καταναλωτής-2 χρησιμοποιούν συνολικά λιγότερη μνήμη, αλλά ακολουθούν την ίδια ανοδική τάση. Κατά τη φάση αδράνειας, η χρήση μνήμης μειώνεται για όλες τις υπηρεσίες, γεγονός που αντικατοπτρίζει την αποτελεσματική μείωση της κλίμακας.



Σχημα 5.41: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Χρήση μνήμης

Βασικές παρατηρήσεις:

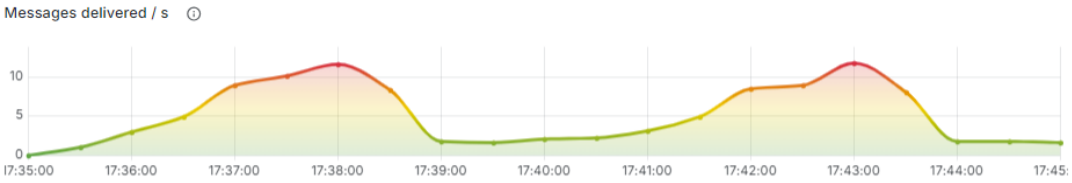
- Η υπηρεσία-καταναλωτής-3 χρησιμοποιεί σταθερά περισσότερη μνήμη από τις άλλες δύο υπηρεσίες, πιθανότατα λόγω υψηλότερων απαιτήσεων επεξεργασίας.

- Η κατανάλωση μνήμης αυξάνεται όπως αναμένεται κατά την αύξηση του φορτίου και απελευθερώνεται κατά τη φάση αδράνειας.
- Το σύστημα δεν παρουσιάζει ενδείξεις διαρροής μνήμης ή υπερκατανάλωσης κατά τη διάρκεια αυτής της δοκιμής.

Αναμενόμενη συμπεριφορά: Η κατανάλωση μνήμης θα πρέπει να αυξάνεται με την αύξηση του φορτίου και να πέφτει σε χαμηλότερα επίπεδα κατά τη διάρκεια των περιόδων αδράνειας. Το σύστημα αποδίδει καλά, προσαρμόζοντας τη χρήση της μνήμης ανάλογα με το φορτίο και αποδεσμεύοντας κατάλληλα τους πόρους.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (εξερχόμενη κίνηση)

Το γράφημα «Παραδιδόμενα μηνύματα ανά δευτερόλεπτο» αντικατοπτρίζει τις σταδιακές αυξήσεις του φορτίου στην εγκατάσταση δοκιμής. Βλέπουμε ευδιάκριτες κορυφές σε κάθε φάση αύξησης του φορτίου, ιδιαίτερα στα 3 RPS και 5 RPS, ακολουθούμενες από μια απότομη μείωση κατά τη φάση αδράνειας. Αυτό υποδηλώνει ότι το σύστημα επεξεργάζεται αποτελεσματικά τα μηνύματα καθώς αυξάνεται το φορτίο και στη συνέχεια σταματά μόλις σταματήσει η κίνηση.



Σχημα 5.42: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Παραδιδόμενα μηνύματα

Βασικές παρατηρήσεις:

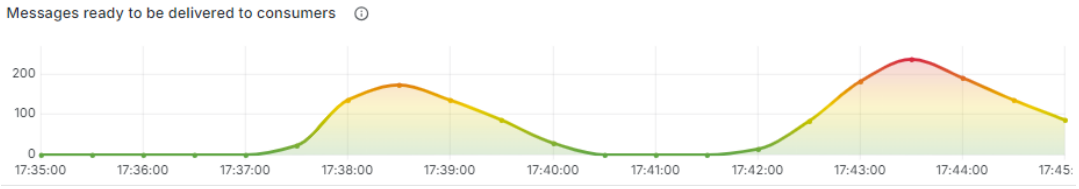
- Η παράδοση μηνυμάτων αυξάνεται ανάλογα με το φορτίο κίνησης κατά τη διάρκεια κάθε φάσης.
- Το σύστημα διαχειρίζεται τους μέγιστους ρυθμούς παράδοσης χωρίς σημαντικές καθυστερήσεις ή συμφόρηση.
- Η φάση αδράνειας αντικατοπτρίζεται σαφώς, με το ρυθμό μηνυμάτων να μηδενίζεται όταν δεν υπάρχει φορτίο.

Αναμενόμενη συμπεριφορά: Το σύστημα αναμένεται να παραδίδει μηνύματα με ρυθμούς που αντιστοιχούν στην εισερχόμενη κίνηση και να διακόπτει την επεξεργασία μηνυμάτων κατά τη διάρκεια των περιόδων αδράνειας. Το σύστημα ανταποκρίνεται σε αυτές τις προσδοκίες, κλιμακώνεται για την αιχμή της ζήτησης και διακόπτει σωστά κατά τη διάρκεια των φάσεων αδράνειας.

Μέγεθος ουράς (μηνύματα έτοιμα να παραδοθούν στους καταναλωτές)

Το γράφημα μεγέθους ουράς δείχνει ελάχιστη συσσώρευση μηνυμάτων, γεγονός που αποτελεί εξαιρετικό σημάδι αποδοτικής επεξεργασίας. Υπάρχει μια

μικρή συσσώρευση κατά τη διάρκεια των φάσεων με υψηλότερο RPS, αλλά η ουρά αδειάζει γρήγορα. Κανένα μήνυμα δεν παραμένει στην ουρά κατά την περίοδο αδράνειας, επιβεβαιώνοντας ότι το σύστημα χειρίζεται και επεξεργάζεται επιτυχώς το εισερχόμενο φορτίο.



Σχημα 5.43: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Μέγεθος ουράς

Βασικές παρατηρήσεις:

- Υπάρχει μια μικρή συσσώρευση μηνυμάτων στην ουρά κατά τη διάρκεια των φάσεων αύξησης του φορτίου, αλλά ομαλοποιείται γρήγορα.
- Το σύστημα επεξεργάζεται αποτελεσματικά τα μηνύματα που βρίσκονται στην ουρά, χωρίς συσσωρευμένο όγκο κατά τη διάρκεια της φάσης αδράνειας.
- Το μέγεθος της ουράς αντιστοιχεί καλά στην αύξηση του φορτίου και στον επακόλουθο χρόνο αδράνειας, χωρίς ενδείξεις συμφόρησης.

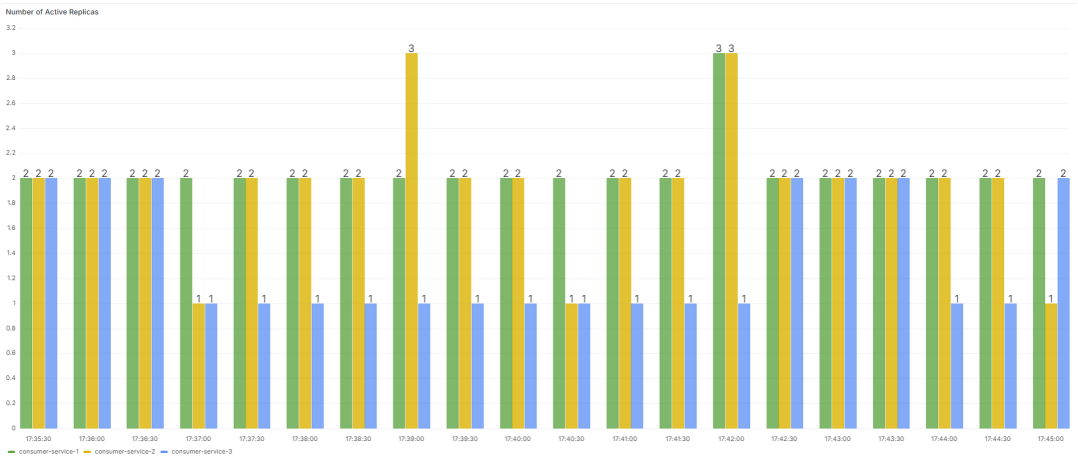
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να επεξεργάζεται έγκαιρα τα εισερχόμενα αιτήματα για να αποτρέπει τη συσσώρευση ουράς, ιδίως κατά τη διάρκεια περιόδων υψηλού φόρτου. Τα αποτελέσματα ευθυγραμμίζονται με τις προσδοκίες, δείχνοντας ελάχιστη συσσώρευση μηνυμάτων και αποτελεσματική διαχείριση της ουράς.

Αριθμός ενεργών αντιγράφων (συμπεριφορά αυτόματης κλιμάκωσης)

Το γράφημα «Αριθμός ενεργών αντιγράφων» δείχνει ότι όλες οι υπηρεσίες διατηρούν ένα σταθερό επίπεδο αντιγράφων, με μια σύντομη αύξηση της κλίμακας για τις υπηρεσίες καταναλωτή-2 και καταναλωτή-3. Κατά τη διάρκεια των φάσεων αιχμής του φορτίου, οι υπηρεσίες αυτές αυξάνουν τον αριθμό των ενεργών αντιγράφων και στη συνέχεια μειώνουν την κλίμακα καθώς ο φόρτος κίνησης μειώνεται σε περιόδους αδράνειας.

Βασικές παρατηρήσεις:

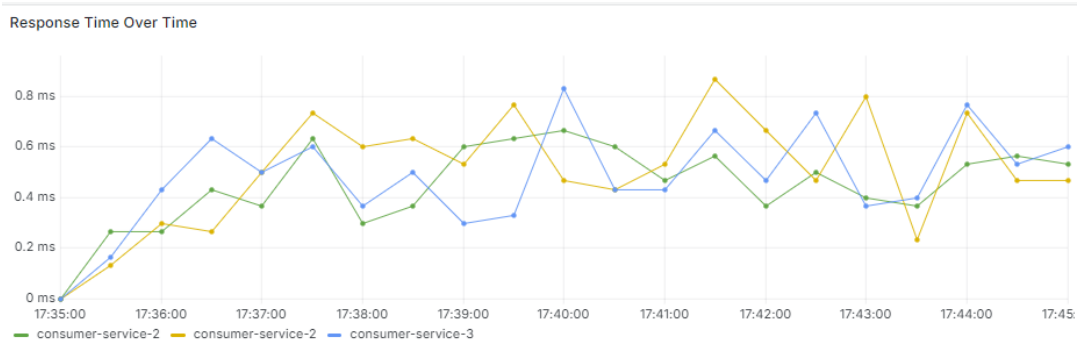
- Η υπηρεσία καταναλωτή-2 και η υπηρεσία καταναλωτή-3 κλιμακώνονται για λίγο σε 3 αντίγραφα κατά τη φάση του υψηλότερου φορτίου, ενώ η υπηρεσία καταναλωτή-1 παραμένει σε 2 αντίγραφα.
- Το σύστημα κλιμακώνεται αποτελεσματικά, προσθέτοντας και αφαιρώντας αντίγραφα ανάλογα με τις ανάγκες κατά τις φάσεις αύξησης του φορτίου και αδράνειας.
- Η συμπεριφορά αυτόματης κλιμάκωσης εμφανίζεται ισορροπημένη, χωρίς υπερβολική κατανομή πόρων.



Σχημα 5.44: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Αριθμός ενεργών αντιγράφων

Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να κλιμακώνει δυναμικά τα αντίγραφα με βάση το φορτίο κίνησης και να τα μειώνει κατά τις περιόδους αδράνειας. Η παρατηρούμενη συμπεριφορά κλιμάκωσης είναι σύμφωνη με τις προσδοκίες για τον χειρισμό σταδιακών αυξήσεων του φορτίου και την αποδοτικότητα των πόρων κατά τις περιόδους αδράνειας.

Χρόνος απόκρισης με την πάροδο του χρόνου Το γράφημα του χρόνου απόκρισης δείχνει ότι, παρά τις σταδιακές αυξήσεις του φορτίου, οι χρόνοι απόκρισης παραμένουν χαμηλοί, με σύντομη κορύφωση κατά τη διάρκεια των περιόδων υψηλότερου φορτίου. Και οι τρεις υπηρεσίες παρουσιάζουν παρόμοιους χρόνους απόκρισης, υποδεικνύοντας ότι το σύστημα διαχειρίζεται την αυξημένη κίνηση χωρίς σημαντικές καθυστερήσεις.



Σχημα 5.45: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Χρόνος απόκρισης

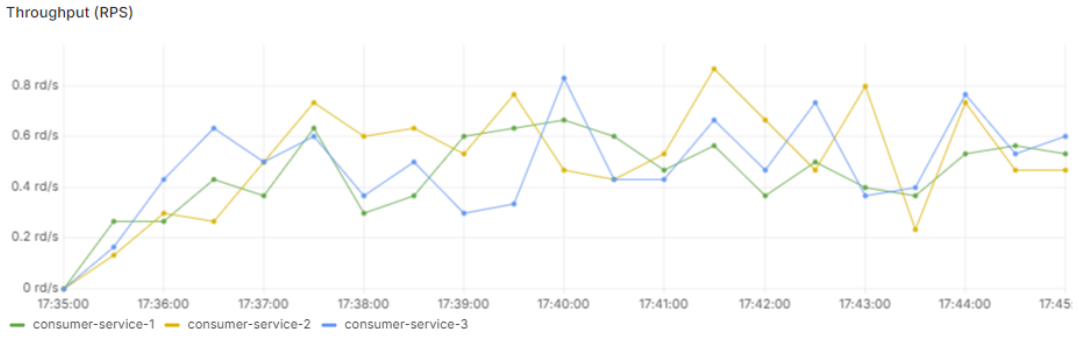
Βασικές παρατηρήσεις:

- Οι χρόνοι απόκρισης κορυφώνονται κατά τη διάρκεια της φάσης 5 RPS, αλλά παραμένουν εντός αποδεκτών ορίων.
- Η φάση αδράνειας παρουσιάζει σταθερά χαμηλούς χρόνους απόκρισης, καθώς το σύστημα εκκαθαρίζει τις εκκρεμότητες και επιστρέφει σε σταθερή κατάσταση.

- Όλες οι υπηρεσίες παρουσιάζουν παρόμοια μοτίβα απόκρισης, με ελάχιστες διακυμάνσεις κατά τη διάρκεια της δοκιμής.

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμένουν χαμηλοί και σταθεροί ακόμη και όταν αυξάνεται το φορτίο. Το σύστημα έχει την αναμενόμενη συμπεριφορά, αντιμετωπίζοντας την αυξημένη ζήτηση με μικρές αιχμές χρόνου απόκρισης και επιστρέφοντας στη βασική γραμμή κατά τις φάσεις αδράνειας.

Απόδοση (αιτήσεις ανά δευτερόλεπτο - RPS) Το γράφημα της απόδοσης αντικατοπτρίζει τη σταδιακή αύξηση του φορτίου, με την απόδοση να κορυφώνεται κατά τη διάρκεια κάθε φάσης κίνησης και να μηδενίζεται κατά τη διάρκεια της περιόδου αδράνειας. Τα αποτελέσματα καταδεικνύουν ότι το σύστημα διαχειρίζεται ομαλά την εισερχόμενη κίνηση και κλιμακώνει την ικανότητα επεξεργασίας του ώστε να ανταποκρίνεται στο φορτίο.



Σχημα 5.46: Πειράματα Κλιμάκωσης με KPA: Σταδιακή Αύξηση Φορτίου με Φάση Αδράνειας : Απόδοση (RPS)

Βασικές παρατηρήσεις:

- Η ρυθμαπόδοση αυξάνεται ανάλογα με το φορτίο κίνησης, παρουσιάζοντας σαφείς κορυφές σε 1 RPS, 3 RPS και 5 RPS.
- Το σύστημα επεξεργάζεται αποτελεσματικά τις αιτήσεις, χωρίς πτώσεις ή ασυνέπειες κατά τη διάρκεια των φάσεων φόρτου.
- Η απόδοση πέφτει στο μηδέν κατά τη διάρκεια της φάσης αδράνειας, επιβεβαιώνοντας ότι το σύστημα σταματά την επεξεργασία των αιτήσεων όπως αναμένεται.

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να αυξάνεται ανάλογα με το φορτίο και στη συνέχεια να πέφτει κατά τη διάρκεια των περιόδων αδράνειας. Το σύστημα ανταποκρίνεται στις προσδοκίες, με ομαλή κλιμάκωση και χωρίς καθυστερήσεις επεξεργασίας κατά τις φάσεις αιχμής.

Συμπέρασμα

Τα αποτελέσματα αυτής της δοκιμής επιβεβαιώνουν ότι το σύστημα είναι ικανό να διαχειριστεί αποτελεσματικά τη σταδιακή αύξηση του φορτίου και

τις περιόδους αδράνειας. Η χρήση CPU και μνήμης αυξάνεται και μειώνεται ανάλογα με το εισερχόμενο φορτίο, ενώ το σύστημα παραμένει αποδοτικό και προσαρμόζεται δυναμικά χωρίς υπερβολική κατανάλωση πόρων. Η παράδοση μηνυμάτων παραμένει σύμφωνη με την εισερχόμενη κίνηση, χωρίς σημαντική συσσώρευση στην ουρά, ενώ οι χρόνοι απόκρισης παραμένουν χαμηλοί και σταθεροί. Η συμπεριφορά της αυτόματης κλιμάκωσης αποδεικνύει ότι το σύστημα μπορεί να προσαρμοστεί δυναμικά στις ανάγκες του φορτίου, με τον αριθμό των ενεργών αντιγράφων να αυξάνεται και να μειώνεται κατάλληλα. Συνολικά, το σύστημα αποδεικνύει την ικανότητά του να διαχειρίζεται μεταβαλλόμενα φορτία με αποδοτικό τρόπο, διασφαλίζοντας την ισορροπία μεταξύ χρήσης πόρων και απόδοσης.

5.5.8 Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας

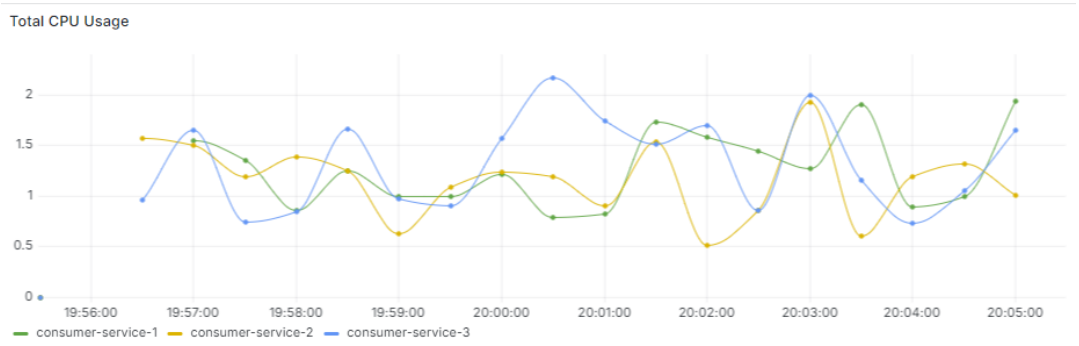
Εγκατάσταση δοκιμής: 2:15, 5:5, 3:15, 0:10, 1:5, 4:5, 0:15
Περιγραφή: Αυτή η δοκιμή εφάρμοσε συχνές σύντομες εκρήξεις υψηλής κίνησης, διανθισμένες με σύντομες περιόδους αδράνειας, προσομοιώνοντας ταχέως μεταβαλλόμενες συνθήκες.
Σκοπός: Αυτή η δοκιμή είχε ως στόχο να καταπονήσει την ικανότητα του συστήματος να προσαρμόζεται γρήγορα σε συχνές και ξαφνικές αλλαγές φορτίου, παρατηρώντας πόσο καλά κλιμακώνεται προς τα πάνω και προς τα κάτω χωρίς να συσσωρεύει μηνύματα κατά τη διάρκεια των ταχέων μεταβάσεων.
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να αποτρέπει τη συσσώρευση μηνυμάτων, ενώ θα πρέπει να κλιμακώνεται αποτελεσματικά κατά τη διάρκεια των εκρήξεων και να μειώνεται κατά τη διάρκεια των σύντομων περιόδων αδράνειας.
Περίπτωση χρήσης: Αυτό θα μπορούσε να μιμηθεί ένα σενάριο επεξεργασίας πλαισίων βίντεο σε πραγματικό χρόνο για τη συγκράτηση περιεχομένου, όπου εμφανίζονται αιχμές στην κίνηση κατά τη διάρκεια ζωντανών γεγονότων ή ροών, ακολουθούμενες από σύντομες διακοπές.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Από το γράφημα, παρατηρούμε συχνές διακυμάνσεις στη χρήση της CPU, με αξιοσημείωτες κορυφές που ευθυγραμμίζονται με τις περιόδους αιχμής κατά τη διάρκεια της δοκιμής. Κάθε έκρηξη προκαλεί άμεση αύξηση της κατανάλωσης CPU σε όλες τις υπηρεσίες, ακολουθούμενη από απότομη πτώση κατά τη διάρκεια των σύντομων κενών αδράνειας. Η υπηρεσία καταναλωτή-3 φαίνεται να παρουσιάζει ελαφρώς υψηλότερη χρήση CPU κατά τη διάρκεια αυτών των εκρήξεων, ιδίως στις 19:57 και 20:02, ενώ οι υπηρεσίες καταναλωτή-1 και καταναλωτή-2 ακολουθούν παρόμοιο, αλλά ελαφρώς χαμηλότερο, μοτίβο.

Βασικές παρατηρήσεις:

- Και οι τρεις υπηρεσίες παρουσιάζουν συγχρονισμένες κορυφές χρήσης CPU που αντιστοιχούν σε εκρήξεις κίνησης.
- Η υπηρεσία-καταναλωτής-3 παρουσιάζει υψηλότερες αιχμές σε σύγκριση με τις άλλες υπηρεσίες κατά τη διάρκεια έντονων εκρήξεων.
- Το σύστημα χειρίζεται αποτελεσματικά τις μεταβάσεις μεταξύ των εκρήξεων και των περιόδων αδράνειας, καθώς η χρήση της CPU μειώνεται σταθερά κατά τη διάρκεια των περιόδων αδράνειας.

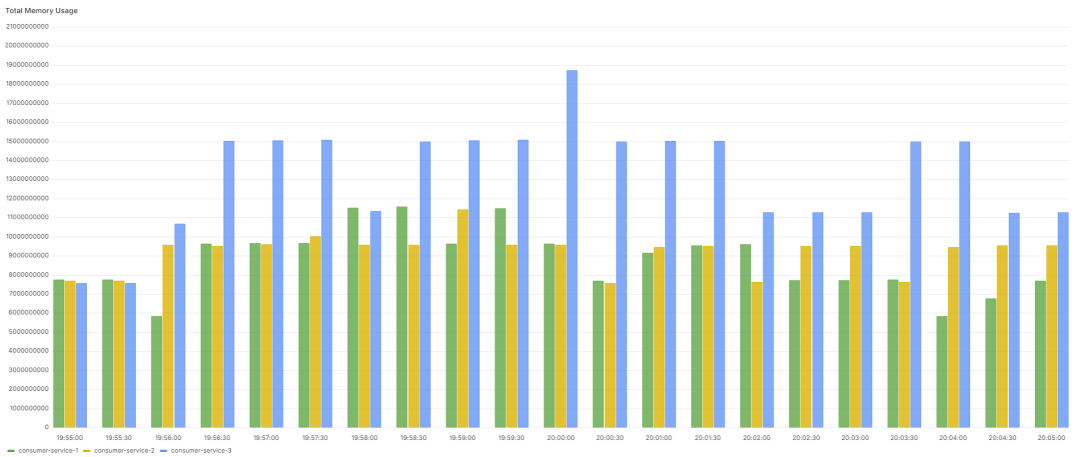


Σχημα 5.47: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Συνολική χρήση CPU

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται γρήγορα για τη διαχείριση των εκρήξεων και να μειώνεται κατά τη διάρκεια των περιόδων αδράνειας. Η γρήγορη προσαρμογή του συστήματος, ιδίως η ικανότητα να αυξάνει την κλίμακα κατά τη διάρκεια εκρήξεων και να απελευθερώνει πόρους κατά τη διάρκεια κενών αδράνειας, ευθυγραμμίζεται με τις προσδοκίες για αυτό το σενάριο δοκιμής.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις καταναλωτικές υπηρεσίες)

Το γράφημα χρήσης μνήμης δείχνει ένα αρκετά σταθερό μοτίβο με μικρές διακυμάνσεις, ακόμη και κατά τη διάρκεια των εκρήξεων υψηλής έντασης. Η υπηρεσία καταναλωτή-3 παρουσιάζει τη μεγαλύτερη συνολική κατανάλωση μνήμης, ακολουθούμενη από την υπηρεσία καταναλωτή-1 και την υπηρεσία καταναλωτή-2. Οι περίοδοι αδράνειας δεν επηρεάζουν σημαντικά τη χρήση μνήμης, η οποία παραμένει σταθερή καθ' όλη τη διάρκεια της δοκιμής.



Σχημα 5.48: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Χρήση μνήμης

Βασικές παρατηρήσεις:

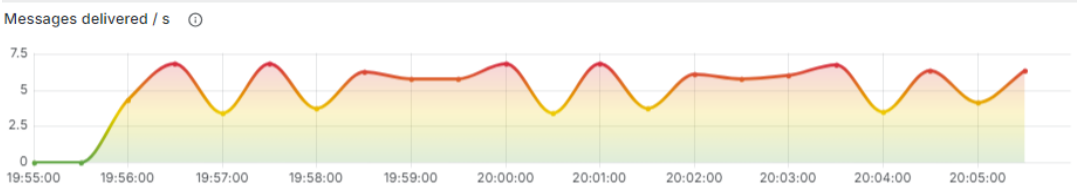
- Η χρήση μνήμης παραμένει σχετικά σταθερή, με ελάχιστη διακύμανση κατά τη διάρκεια των ριπών και των περιόδων αδράνειας.

- Η υπηρεσία καταναλωτή-3 χρησιμοποιεί σταθερά τη μεγαλύτερη μνήμη, ενώ οι άλλες δύο υπηρεσίες παρουσιάζουν μικρότερες διακυμάνσεις.
- Το σύστημα διατηρεί αποτελεσματική διαχείριση της μνήμης παρά τις συχνές αλλαγές του φορτίου.

Αναμενόμενη συμπεριφορά: Το σύστημα αναμένεται να διατηρεί σταθερή χρήση μνήμης, με πιθανές μικρές αυξήσεις κατά τη διάρκεια των εκρήξεων. Αυτή η συμπεριφορά παρατηρείται, υποδεικνύοντας αποτελεσματική κατανομή μνήμης ακόμη και υπό συνθήκες ταχείας διακύμανσης.

Παραδοθέντα μηνύματα (Παραδοθέντα μηνύματα ανά δευτερόλεπτο)

Το γράφημα για τα μηνύματα που παραδίδονται ανά δευτερόλεπτο παρουσιάζει συχνές κορυφές που ευθυγραμμίζονται με τις σύντομες εκρήξεις κυκλοφορίας. Το σύστημα διαχειρίζεται καλά αυτές τις εκρήξεις, διατηρώντας σταθερή απόδοση που φτάνει έως και τα 7,5 μηνύματα ανά δευτερόλεπτο κατά τις πιο πολυσύχναστες περιόδους. Υπάρχουν σαφείς βυθίσεις κατά τη διάρκεια των κενών αδράνειας, αλλά αυτές είναι αναμενόμενες δεδομένης της διάταξης δοκιμής.



Σχημα 5.49: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Παραδοθέντα μηνύματα

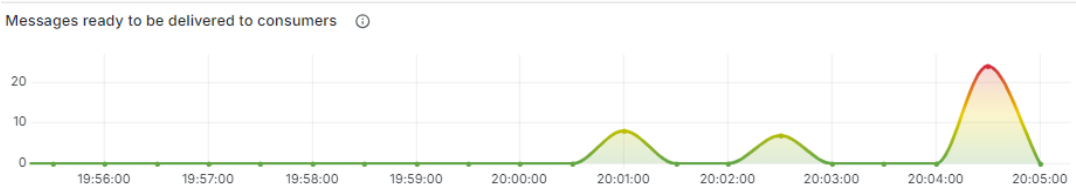
Βασικές παρατηρήσεις:

- Το σύστημα κλιμακώνεται αποτελεσματικά για να ανταποκριθεί στις εκρήξεις κίνησης, παρέχοντας έως και 7,5 μηνύματα ανά δευτερόλεπτο.
- Οι πτώσεις στην παράδοση ευθυγραμμίζονται με τα σύντομα κενά αδράνειας, αποδεικνύοντας ότι το σύστημα μειώνεται κατάλληλα όταν μειώνεται η ζήτηση.

Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να κλιμακώνει ταχέως τους ρυθμούς παράδοσης κατά τη διάρκεια εκρήξεων κυκλοφορίας και να προσαρμόζεται γρήγορα κατά τη διάρκεια περιόδων αδράνειας. Αυτή η συμπεριφορά επιβεβαιώνεται από τις συνεχείς αιχμές στους ρυθμούς παράδοσης, γεγονός που δείχνει την ικανότητα του συστήματος να χειρίζεται συχνά σενάρια εκρήξεων.

Μήκος ουράς (μηνύματα έτοιμα προς παράδοση)

Το γράφημα μήκους ουράς υποδεικνύει ελάχιστη καθυστέρηση κατά τη διάρκεια των εκρήξεων, με μικρές μόνο κορυφές συσσωρευμένων μηνυμάτων, ιδίως γύρω στις 20:04. Κατά το μεγαλύτερο μέρος της δοκιμής, το σύστημα είναι σε θέση να παραδίδει μηνύματα σε πραγματικό χρόνο, διατηρώντας την ουρά αναμονής σχεδόν κενή.



Σχημα 5.50: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Μήκος ουράς

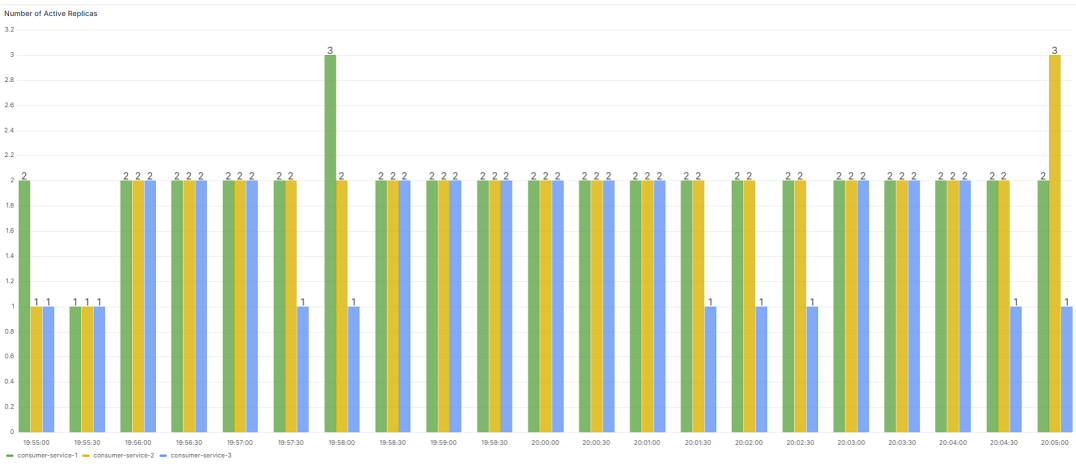
Βασικές παρατηρήσεις:

- Ελάχιστη συσσώρευση μηνυμάτων, με μικρή μόνο αύξηση του μεγέθους της ουράς κατά τη διάρκεια ορισμένων εκρήξεων, ιδίως προς το τέλος της δοκιμής.
- Το σύστημα αποτρέπει τη συσσώρευση μηνυμάτων, διατηρώντας μια αποτελεσματική ροή καθ' όλη τη διάρκεια των συχνών εκρήξεων.

Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να αποφεύγει τη σημαντική συσσώρευση μηνυμάτων κατά τη διάρκεια των ριπών, επεξεργαζόμενο αποτελεσματικά τα αιτήματα. Η δοκιμή το επιβεβαιώνει αυτό, καθώς η ουρά παραμένει σχεδόν κενή καθ' όλη τη διάρκεια του μεγαλύτερου μέρους της δοκιμής.

Αριθμός ενεργών αντιγράφων

Ο αριθμός των ενεργών αντιγράφων παραμένει σταθερός στα δύο για τις περισσότερες υπηρεσίες, αλλά η υπηρεσία καταναλωτή-1 και η υπηρεσία καταναλωτή-2 παρουσιάζουν κάποιες σύντομες προσαρμογές, μειώνοντας την κλίμακα σε ένα αντίγραφο κατά τη διάρκεια περιόδων μειωμένου φόρτου.



Σχημα 5.51: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Αριθμός ενεργών αντιγράφων

Βασικές παρατηρήσεις:

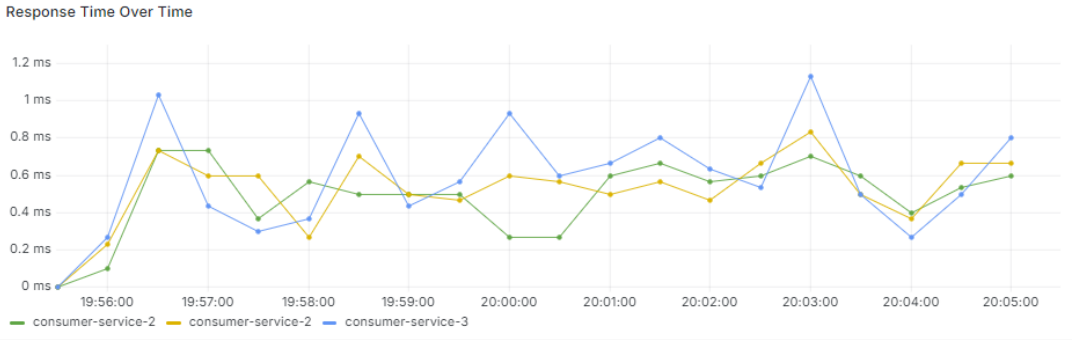
- Το σύστημα προσαρμόζει δυναμικά τα αντίγραφα, ιδίως για την υπηρεσία καταναλωτή-1 και την υπηρεσία καταναλωτή-2, τα οποία μειώνονται για λίγο σε περιόδους μειωμένης ζήτησης.

- Η υπηρεσία καταναλωτή-3 παραμένει σταθερή με δύο αντίγραφα καθ' όλη τη διάρκεια της δοκιμής, πιθανότατα λόγω της υψηλότερης χρήσης μνήμης και του φόρτου εργασίας της.

Αναμενόμενη συμπεριφορά: Το σύστημα αναμένεται να προσαρμόζει τον αριθμό των αντιγράφων ανάλογα με τη ζήτηση, ιδίως κατά τη διάρκεια κενών αδράνειας. Αυτό αντικατοπτρίζεται στο γράφημα, με την κλιμάκωση των αντιγράφων κατά τη διάρκεια των περιόδων αδράνειας και τη διατήρηση της σταθερότητας κατά τη διάρκεια των εκρήξεων.

Χρόνος απόκρισης με την πάροδο του χρόνου

Το γράφημα του χρόνου απόκρισης παρουσιάζει συχνές αιχμές, με τους χρόνους απόκρισης να φτάνουν έως και τα 0,8 ms κατά τη διάρκεια των εκρήξεων. Η υπηρεσία καταναλωτή-3 παρουσιάζει ελαφρώς υψηλότερους χρόνους απόκρισης σε σύγκριση με τις άλλες, ιδίως γύρω στις 19:57 και 20:03, αλλά οι διαφορές είναι ελάχιστες.



Σχημα 5.52: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Χρόνος απόκρισης

Βασικές παρατηρήσεις:

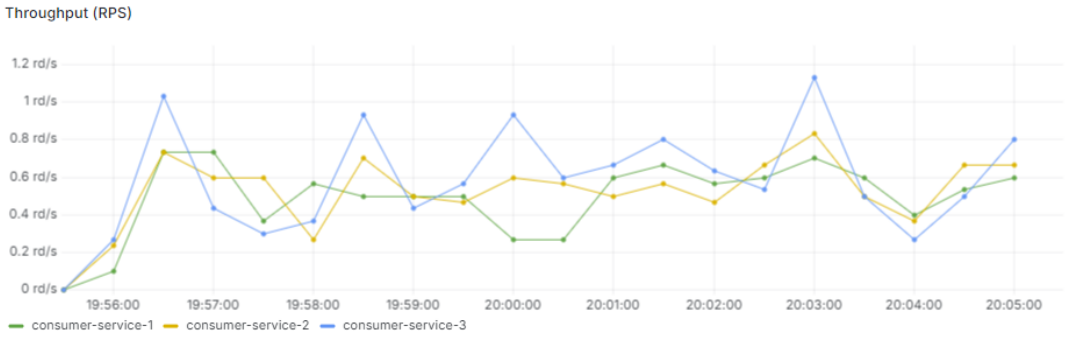
- Οι χρόνοι απόκρισης αυξάνονται κατά τη διάρκεια των ριπών, αλλά παραμένουν εντός των αποδεκτών ορίων, με μέγιστη τιμή περίπου 0,8 ms.
- Η υπηρεσία καταναλωτή-3 έχει σταθερά υψηλότερους χρόνους απόκρισης σε σύγκριση με τις άλλες δύο υπηρεσίες.

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να αυξάνονται ελαφρώς κατά τη διάρκεια εκρήξεων κίνησης, αλλά να παραμένουν εντός λογικών ορίων. Η συμπεριφορά του συστήματος το επιβεβαιώνει αυτό, με τους χρόνους απόκρισης να παραμένουν χαμηλοί ακόμη και σε περιόδους υψηλής κίνησης.

Απόδοση (RPS)

Το γράφημα ρυθμού μετάδοσης ευθυγραμμίζεται με τις συχνές περιόδους ριπής της δοκιμής, με αιχμές κατά τη διάρκεια των ριπών και βυθίσεις κατά τη διάρκεια των κενών αδράνειας. Η υπηρεσία καταναλωτή-3 εμφανίζει και πάλι

ελαφρώς υψηλότερη απόδοση κατά τη διάρκεια των αιφνιδιασμών, αλλά όλες οι υπηρεσίες διατηρούν παρόμοιο μοτίβο συνολικά.



Σχημα 5.53: Πειράματα Κλιμάκωσης με KPA: Συχνές Εκρήξεις Φορτίου με Σύντομα Κενά Αδράνειας : Απόδοση (RPS)

Βασικές παρατηρήσεις:

- Η απόδοση αυξάνεται σε περίπου 1 αίτηση ανά δευτερόλεπτο κατά τη διάρκεια των διαλείψεων, αποδεικνύοντας την ικανότητα του συστήματος να διαχειρίζεται αποτελεσματικά την υψηλή κυκλοφορία.
- Όλες οι υπηρεσίες παρουσιάζουν παρόμοια μοτίβα απόδοσης, με μικρές διαφορές, ιδίως κατά τις περιόδους αιχμής.

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να αυξάνεται κατά τη διάρκεια των εκρήξεων και να μειώνεται κατά τις περιόδους αδράνειας. Το σύστημα ανταποκρίνεται σε αυτές τις προσδοκίες, με σταθερές αιχμές ρυθμού μετάδοσης που ευθυγραμμίζονται με τις εκρήξεις κυκλοφορίας.

Συμπέρασμα

Το σύστημα επιδεικνύει ισχυρή προσαρμοστικότητα και αποδοτικότητα στη διαχείριση συχνών εκρήξεων κυκλοφορίας με σύντομα κενά αδράνειας. Αποτρέπει τη συσσώρευση μηνυμάτων, διατηρεί σταθερή χρήση πόρων και προσαρμόζεται δυναμικά στο μεταβαλλόμενο φορτίο. Αυτό είναι ιδιαίτερα σημαντικό για περιπτώσεις χρήσης όπως η επεξεργασία βίντεο σε πραγματικό χρόνο, όπου οι ξαφνικές αιχμές στην κυκλοφορία πρέπει να αντιμετωπίζονται χωρίς τη συσσώρευση καθυστερήσεων ή την εξάντληση πόρων. Η απόδοση του συστήματος σε αυτή τη δοκιμή επιβεβαιώνει την ικανότητά του να ανταποκρίνεται σε αυτές τις απαιτήσεις, διασφαλίζοντας ομαλή κλιμάκωση και φάσεις ανάκαμψης.

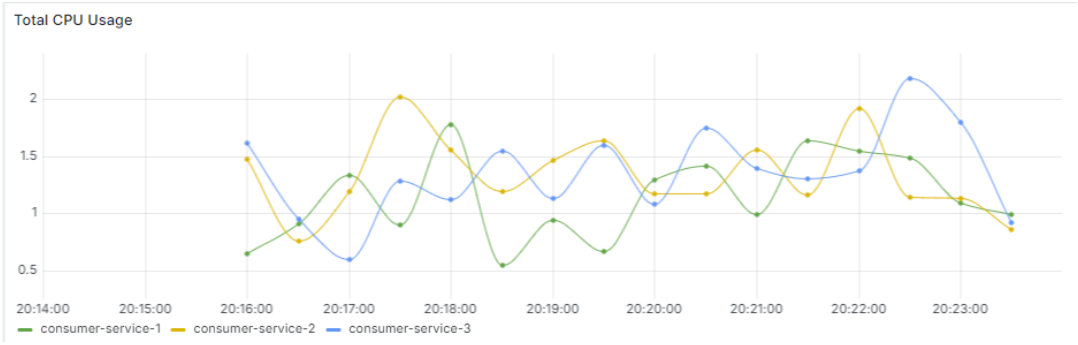
5.5.9 Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας

Εγκατάσταση δοκιμής: 1:5, 4:5, 0:4, 5:3, 2:5, 3:5, 0:5
Περιγραφή: Αυτή η δοκιμή παρουσίασε εναλλασσόμενες φάσεις φόρτου μεταξύ 1 και 5 αιτήσεων ανά δευτερόλεπτο, με ελάχιστο χρόνο αδράνειας μεταξύ των φάσεων.
Σκοπός: Η δοκιμή αυτή αξιολόγησε τον τρόπο με τον οποίο το σύστημα λειτουργεί υπό σχεδόν συνεχή φόρτο, χωρίς σχεδόν καθόλου χρόνο αποκατάστασης

μεταξύ των ριπών. Η έμφαση δόθηκε στην ικανότητα του συστήματος να κλιμακώνεται αποτελεσματικά υπό συνεχή πίεση.
Αναμενόμενη συμπεριφορά: Το σύστημα θα πρέπει να είναι σε θέση να επεξεργάζεται αιτήσεις συνεχώς χωρίς σημαντική συσσώρευση μηνυμάτων, ακόμη και καθώς οι χρόνοι αδράνειας ελαχιστοποιούνται.
Περίπτωση χρήσης: Αυτό το μοτίβο θα μπορούσε να παρατηρηθεί σε συστήματα επιτήρησης όπου διαφορετικές κάμερες στέλνουν ποικίλους φόρτους εργασίας κατά τη διάρκεια της ημέρας, αλλά οι περίοδοι αδράνειας είναι σύντομες ή ανύπαρκτες.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Από το γράφημα παρατηρούμε ότι η χρήση της CPU παρουσιάζει μέτριες διακυμάνσεις και στις τρεις υπηρεσίες. Η υπηρεσία-καταναλωτής-2 παρουσιάζει τις υψηλότερες αιχμές, ακολουθούμενη από την υπηρεσία-καταναλωτής-3, με την υπηρεσία-καταναλωτής-1 να διατηρεί χαμηλότερη αλλά σταθερή χρήση. Οι φάσεις εναλλασσόμενου φορτίου μεταξύ 1 και 5 αιτήσεων ανά δευτερόλεπτο παρουσιάζουν σαφείς κορυφές που αντιστοιχούν στο υψηλότερο φορτίο, και οι υπηρεσίες γενικά κλιμακώνουν αποτελεσματικά τη χρήση της CPU τους. Ωστόσο, η υπηρεσία-καταναλωτής-2 φαίνεται να κλιμακώνεται ταχύτερα σε σύγκριση με τις άλλες υπηρεσίες, ιδίως κατά τις φάσεις με υψηλότερη κίνηση.



Σχημα 5.54: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Συνολική χρήση CPU

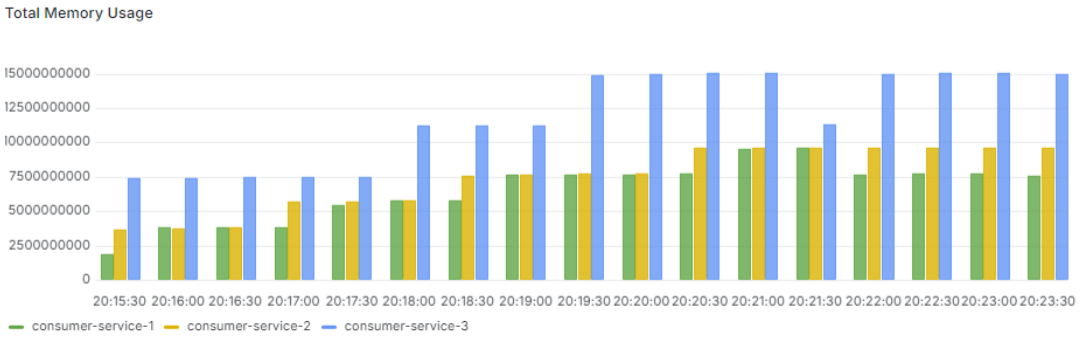
Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-2 παρουσιάζει τις σημαντικότερες αιχμές χρήσης CPU, ιδίως κατά τις περιόδους υψηλότερου φόρτου.
- Η υπηρεσία-καταναλωτής-3 ακολουθεί από κοντά, ενώ η υπηρεσία-καταναλωτής-1 διατηρεί συγκριτικά χαμηλότερη και πιο σταθερή χρήση CPU.
- Το σύστημα κλιμακώνεται δυναμικά, καθώς η χρήση της CPU προσαρμόζεται σε πραγματικό χρόνο για να διαχειρίζεται την κυκλοφορία χωρίς συνεχείς αιχμές.

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται κατά τη διάρκεια φάσεων υψηλότερου φόρτου (4-5 RPS) και να επιστρέφει σε χαμηλότερα επίπεδα κατά τη διάρκεια φάσεων ελαφρύτερου φόρτου (1 RPS). Ο μηχανισμός AIMD πιθανότατα βοηθά στη ρύθμιση της χρήσης των πόρων με τον αποτελεσματικό χειρισμό τόσο των αιχμών όσο και των βυθίσεων της κίνησης.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Στο γράφημα χρήσης μνήμης, βλέπουμε μια κάπως σταθερή κατανομή μνήμης στις τρεις υπηρεσίες, με την υπηρεσία καταναλωτή-3 να καταναλώνει συνήθως περισσότερη μνήμη, ακολουθούμενη από την υπηρεσία καταναλωτή-2 και την υπηρεσία καταναλωτή-1. Η χρήση της μνήμης κλιμακώνεται ως απόκριση στις φάσεις υψηλότερου φορτίου, αλλά δεν παρουσιάζει απότομες αιχμές, γεγονός που υποδηλώνει αποτελεσματική διαχείριση της μνήμης από το σύστημα υπό συνεχή φόρτο.



Σχημα 5.55: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Χρήση μνήμης

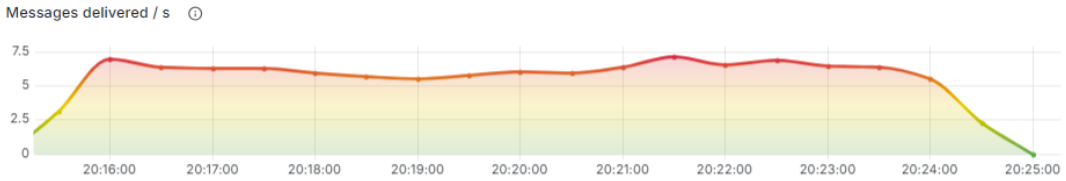
Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-3 καταναλώνει τη μεγαλύτερη μνήμη, ενώ η υπηρεσία-καταναλωτής-1 τη μικρότερη.
- Η χρήση της μνήμης παραμένει σχετικά σταθερή τόσο κατά τις περιόδους υψηλού όσο και κατά τις περιόδους χαμηλού φορτίου, με σταδιακή αύξηση και όχι απότομα άλματα.
- Το σύστημα διαχειρίζεται αποτελεσματικά τη μνήμη, αποτρέποντας την υπερκατανάλωση ακόμη και υπό συνεχή φόρτο.

Αναμενόμενη συμπεριφορά: Η χρήση μνήμης θα πρέπει να παραμένει σταθερή με μικρές μόνο αυξήσεις κατά τη διάρκεια φάσεων υψηλότερου φορτίου. Η ικανότητα του συστήματος να διατηρεί σταθερή χρήση μνήμης κατά τη διάρκεια εναλλασσόμενων φορτίων υποδηλώνει καλό χειρισμό της κατανομής των πόρων, αποφεύγοντας την εξάντληση της μνήμης.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (ρυθμός εξερχόμενων μηνυμάτων)

Το γράφημα του ρυθμού παράδοσης μηνυμάτων δείχνει σταθερές κορυφές παράδοσης που ευθυγραμμίζονται με τις φάσεις εναλλασσόμενου φορτίου. Κατά τη διάρκεια φάσεων υψηλότερου φορτίου, ο ρυθμός παράδοσης κορυφώνεται σε περίπου 7,5 μηνύματα ανά δευτερόλεπτο, ο οποίος στη συνέχεια μειώνεται ελαφρώς κατά τη διάρκεια φάσεων ελαφρύτερου φορτίου. Αυτό υποδηλώνει ότι το σύστημα διατηρεί μια ομαλή ροή μηνυμάτων, κλιμακούμενο αποτελεσματικά για την αντιμετώπιση του εισερχόμενου φορτίου με ελάχιστο χρόνο αδράνειας μεταξύ των εκρήξεων.



Σχημα 5.56: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Παραδιδόμενα μηνύματα

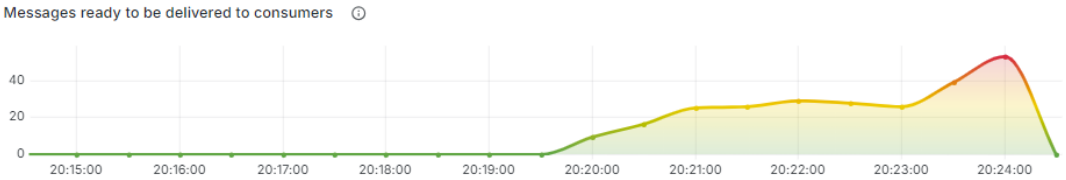
Βασικές παρατηρήσεις:

- Ο ρυθμός παράδοσης μηνυμάτων ταιριάζει με το αναμενόμενο μοτίβο κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια φάσεων υψηλού φόρτου και ελαφριά πτώση κατά τη διάρκεια ελαφρύτερων φάσεων.
- Το σύστημα διατηρεί μια συνεπή ροή μηνυμάτων, αποδεικνύοντας την αποτελεσματική διαχείριση της εξερχόμενης κίνησης.

Αναμενόμενη συμπεριφορά: Ο ρυθμός εξερχόμενων μηνυμάτων θα πρέπει να ακολουθεί στενά το μοτίβο εισερχόμενης κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια μεγαλύτερων φορτίων. Η ικανότητα του συστήματος να διαχειρίζεται συνεχή παράδοση μηνυμάτων με ελάχιστο χρόνο αδράνειας υποδηλώνει αποτελεσματική κλιμάκωση.

Μήκος ουράς (μηνύματα έτοιμα προς παράδοση)

Το γράφημα βάθους ουράς υποδεικνύει μια ελαφρά συσσώρευση μηνυμάτων κατά τη διάρκεια φάσεων υψηλού φόρτου, ιδίως προς το τέλος κάθε ριπής. Ωστόσο, η συσσώρευση δεν είναι υπερβολική και το σύστημα είναι σε θέση να καθαρίζει την ουρά κατά τη διάρκεια περιόδων με μικρότερο φορτίο. Η ουρά δεν αυξάνεται ποτέ πέρα από ένα διαχειρίσιμο επίπεδο, αποδεικνύοντας την ικανότητα του συστήματος να επεξεργάζεται αιτήσεις χωρίς σημαντικές καθυστερήσεις.



Σχημα 5.57: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Μήκος ουράς

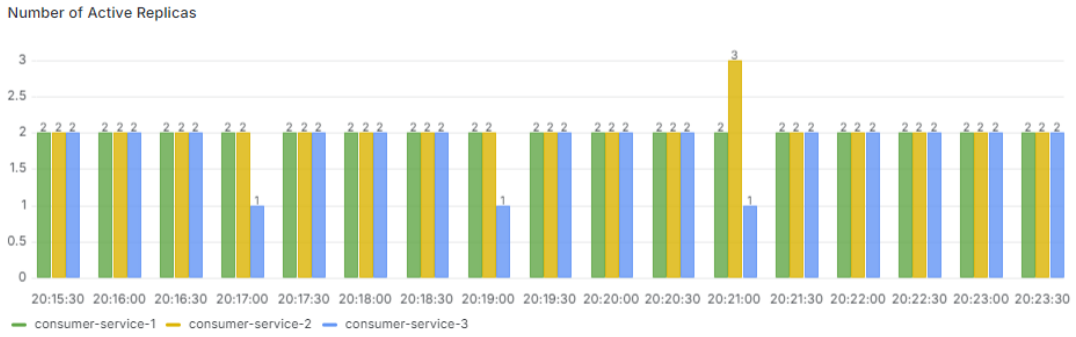
Βασικές παρατηρήσεις:

- Υπάρχει μια μικρή συσσώρευση μηνυμάτων κατά τη διάρκεια φάσεων υψηλότερου φόρτου, η οποία όμως εκκαθαρίζεται κατά τη διάρκεια ελαφρύτερων φάσεων.
- Το σύστημα αποτρέπει την υπερβολική συσσώρευση μηνυμάτων, διασφαλίζοντας την ομαλή ροή μηνυμάτων προς τους καταναλωτές.

Αναμενόμενη συμπεριφορά: Αναμένεται κάποια συσσώρευση ουράς κατά τη διάρκεια περιόδων υψηλού φόρτου, αλλά το σύστημα θα πρέπει να επεξεργάζεται αποτελεσματικά αυτά τα μηνύματα κατά τη διάρκεια ελαφρύτερων φάσεων. Το γεγονός ότι η ουρά παραμένει μικρή υποδηλώνει τον καλό χειρισμό από το σύστημα των εναλλασσόμενων φορτίων με ελάχιστο χρόνο αδράνειας.

Αριθμός ενεργών αντιγράφων

Ο αριθμός των ενεργών αντιγράφων αυξομειώνεται, με τις περισσότερες υπηρεσίες να διατηρούν 2 αντίγραφα καθ' όλη τη διάρκεια της δοκιμής. Η υπηρεσία καταναλωτή-2 κλιμακώνεται σε 3 αντίγραφα για λίγο κατά τη διάρκεια φάσεων με μεγαλύτερο φορτίο, ενώ η υπηρεσία καταναλωτή-1 και η υπηρεσία καταναλωτή-3 παραμένουν ως επί το πλείστον στα 2 αντίγραφα. Αυτή η δυναμική κλιμάκωση δείχνει ότι το σύστημα προσθέτει αποτελεσματικά πόρους όταν χρειάζεται κατά τη διάρκεια φάσεων υψηλού φόρτου και μειώνει την κλίμακα κατά τη διάρκεια ασθενέστερων περιόδων.



Σχημα 5.58: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Αριθμός ενεργών αντιγράφων

Βασικές παρατηρήσεις:

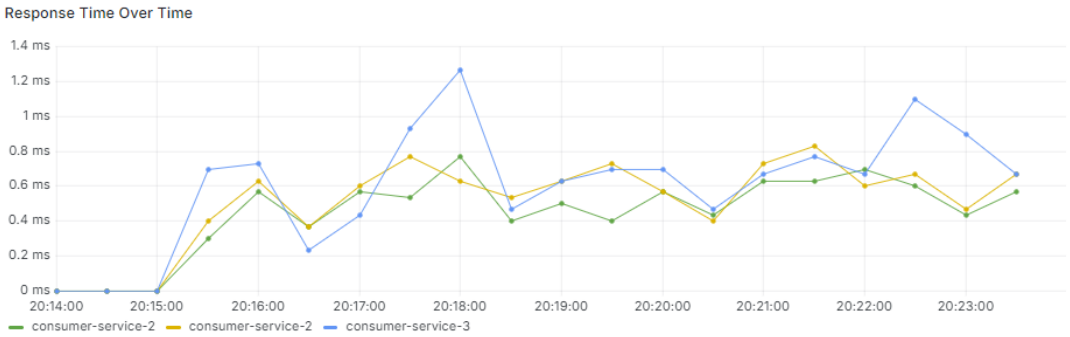
- Η υπηρεσία-καταναλωτής-2 κλιμακώνεται για λίγο σε 3 αντίγραφα, ενώ οι άλλες υπηρεσίες παραμένουν γενικά σε 2 αντίγραφα.
- Το σύστημα προσαρμόζει δυναμικά τον αριθμό των αντιγράφων με βάση το εισερχόμενο φορτίο, υποδεικνύοντας αποτελεσματική κλιμάκωση.

Αναμενόμενη συμπεριφορά: Η κλιμάκωση των αντιγράφων θα πρέπει να γίνεται σε ανταπόκριση στην εισερχόμενη κίνηση, με περισσότερα αντίγραφα κατά τη διάρκεια φάσεων υψηλού φόρτου. Η ικανότητα του συστήματος να κλιμακώνει δυναμικά τα αντίγραφα συμβάλλει στη διατήρηση της απόδοσης χωρίς υπερφόρτωση οποιασδήποτε μεμονωμένης υπηρεσίας.

Χρόνος απόκρισης με την πάροδο του χρόνου

Το γράφημα του χρόνου απόκρισης παρουσιάζει κάποια διακύμανση, με αιχμές κατά τη διάρκεια φάσεων υψηλότερου φορτίου. Ωστόσο, ο συνολικός χρόνος απόκρισης παραμένει κάτω από 1 ms για το μεγαλύτερο μέρος της δοκιμής, γεγονός που υποδηλώνει ότι το σύστημα είναι σε θέση να διαχειριστεί το εναλλασσόμενο φορτίο χωρίς σημαντικές καθυστερήσεις. Η υπηρεσία καταναλωτή-3

παρουσιάζει υψηλότερες αιχμές στο χρόνο απόκρισης κατά τη διάρκεια φάσεων υψηλού φορτίου, ενώ η υπηρεσία καταναλωτή-1 διατηρεί τους πιο σταθερούς χρόνους απόκρισης.



Σχημα 5.59: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Χρόνος απόκρισης

Βασικές παρατηρήσεις:

- Οι χρόνοι απόκρισης παραμένουν ως επί το πλείστον κάτω από 1 ms, με μικρές αιχμές κατά τις φάσεις υψηλού φόρτου.
- Η υπηρεσία καταναλωτή-3 παρουσιάζει τις μεγαλύτερες διακυμάνσεις στους χρόνους απόκρισης, ενώ η υπηρεσία καταναλωτή-1 παραμένει πιο σταθερή.

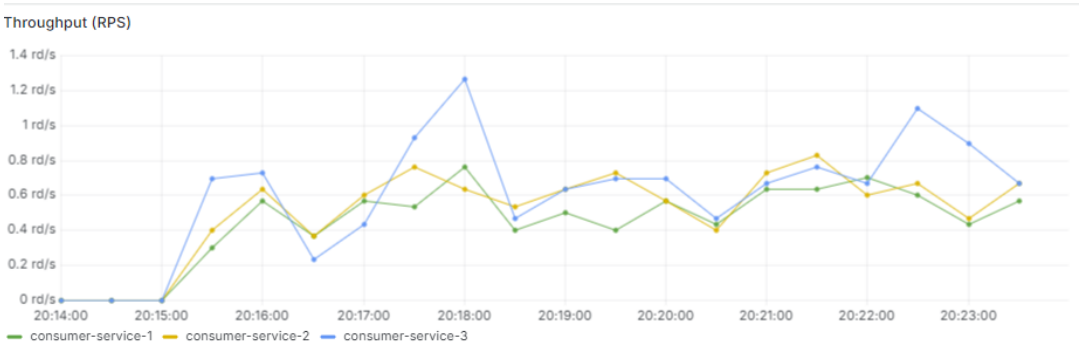
Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμένουν χαμηλοί, ακόμη και κατά τη διάρκεια φάσεων υψηλότερου φόρτου. Αναμένονται μικρές αυξήσεις κατά τη διάρκεια υψηλότερης κίνησης, αλλά το σύστημα θα πρέπει να αποφεύγει τις σημαντικές καθυστερήσεις, όπως φαίνεται στο γράφημα.

Απόδοση (RPS)

Το γράφημα ρυθμού μετάδοσης δείχνει ότι και οι τρεις υπηρεσίες διεκπεραιώνουν τα αιτήματα με σχετικά σταθερό ρυθμό, με την υπηρεσία καταναλωτή-2 να παρουσιάζει υψηλότερη ρυθμό μετάδοσης κατά τη διάρκεια φάσεων με μεγαλύτερο φόρτο. Το σύστημα διατηρεί την απόδοση σε αντιστοιχία με την εισερχόμενη κίνηση, διασφαλίζοντας ότι τα αιτήματα επεξεργάζονται αποτελεσματικά, ακόμη και όταν ελαχιστοποιούνται οι χρόνοι αδράνειας.

Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-2 παρουσιάζει υψηλότερη απόδοση κατά τη διάρκεια των φάσεων υψηλού φόρτου, με την υπηρεσία-καταναλωτής-3 και την υπηρεσία-καταναλωτής-1 να ακολουθούν από κοντά.
- Το σύστημα διατηρεί σταθερή απόδοση, ταιριάζοντας αποτελεσματικά με το μοτίβο εναλλασσόμενου φορτίου.



Σχημα 5.60: Πειράματα Κλιμάκωσης με KPA: Εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας: Απόδοση (RPS)

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να ταιριάζει με το μοτίβο εισερχόμενης κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια φάσεων υψηλού φόρτου. Η ικανότητα του συστήματος να διατηρεί σταθερή ρυθμαπόδοση υπό συνεχή φόρτο καταδεικνύει αποτελεσματική επεξεργασία των αιτήσεων.

Συμπέρασμα

Το σύστημα λειτούργησε καλά υπό εναλλασσόμενο φορτίο με ελάχιστο χρόνο αδράνειας. Η χρήση της CPU κλιμακώθηκε αποτελεσματικά, με την υπηρεσία καταναλωτή-2 να παρουσιάζει τις υψηλότερες αιχμές, ενώ η χρήση της μνήμης παρέμεινε σταθερή σε όλες τις υπηρεσίες. Ο ρυθμός παράδοσης μηνυμάτων ακολουθούσε πιστά το μοτίβο φόρτου και το βάθος της ουράς γινόταν καλά διαχειρίσιμο, αποτρέποντας σημαντικές καθυστερήσεις. Τα ενεργά αντίγραφα κλιμακώθηκαν δυναμικά ανάλογα με το φορτίο και οι χρόνοι απόκρισης παρέμειναν χαμηλοί, εξασφαλίζοντας την ομαλή επεξεργασία των αιτημάτων. Συνολικά, το σύστημα επέδειξε στιβαρό χειρισμό του συνεχούς φορτίου με αποτελεσματική κλιμάκωση των πόρων και ελάχιστες καθυστερήσεις.

5.5.10 Συμπέρασμα: Συμπεριφορά του συστήματος υπό κλιμάκωση KPA σε διάφορα σενάρια φορτίου

Κατά τη διάρκεια της σειράς πειραμάτων, η συμπεριφορά του συστήματος υπό την κλιμάκωση KPA (Knative Pod Autoscaler) αξιολογήθηκε σε πολλαπλά σενάρια, από σταδιακή αύξηση φορτίου με φάσεις αδράνειας έως συχνές εκρήξεις και συνεχή φορτίο με ελάχιστο χρόνο αδράνειας. Τα πειράματα ανέδειξαν την ικανότητα του συστήματος να κλιμακώνει δυναμικά τους πόρους του — τόσο τη CPU όσο και τη μνήμη — ανταποκρινόμενο στις διακυμάνσεις της κίνησης, διασφαλίζοντας αποδοτική χρήση των πόρων και παράλληλα διατηρώντας σταθερή απόδοση και χαμηλούς χρόνους απόκρισης.

Συνολική απόδοση του συστήματος υπό κλιμάκωση KPA

Σε όλες τις περιπτώσεις δοκιμών, ο μηχανισμός κλιμάκωσης KPA παρουσίασε τις ακόλουθες συνεπείς συμπεριφορές:

Αποτελεσματική κλιμάκωση πόρων: Το σύστημα επέδειξε αποτελεσματική κλιμάκωση των πόρων — τόσο της CPU όσο και της μνήμης — ως ανταπόκριση σε

διακυμάνσεις του φορτίου. Ο μηχανισμός KPA προσαρμόζε δυναμικά τον αριθμό των ενεργών αντιγράφων ανάλογα με την ένταση της κίνησης. Οι αιχμές στη χρήση των πόρων συσχετίστηκαν με τις εκρήξεις κυκλοφορίας, ενώ το σύστημα κλιμάκωσε προς τα κάτω κατά τις φάσεις αδράνειας για να αποτρέψει την υπερκατανάλωση πόρων.

Διαχείριση ουρών αναμονής: Σε κάθε σενάριο δοκιμής, το σύστημα επέδειξε αξιόλογη ικανότητα διαχείρισης των ουρών μηνυμάτων. Ακόμα και σε περιόδους έντονης κυκλοφορίας, το βάθος της ουράς παρέμεινε διαχειρίσιμο, με ελάχιστες συσσωρεύσεις. Κατά τις φάσεις αδράνειας ή μικρότερου φορτίου, το σύστημα εκκαθάριζε τις ουρές αποτελεσματικά, αποτρέποντας τη συσσώρευση μηνυμάτων μακροχρόνια.

Χρόνοι απόκρισης: Οι χρόνοι απόκρισης των υπηρεσιών παρέμειναν σταθερά χαμηλοί, ακόμη και σε περιόδους υψηλής κίνησης. Παρότι σημειώθηκαν μικρές αυξήσεις κατά τις αιχμές του φορτίου, το σύστημα διατήρησε καθυστερήσεις κάτω του χιλιοστού του δευτερολέπτου στις περισσότερες περιπτώσεις, εξασφαλίζοντας ομαλή και έγκαιρη διεκπεραίωση αιτήσεων.

Σταθερή απόδοση: Το σύστημα διατηρούσε σταθερή απόδοση που προσαρμοζόταν στο μοτίβο της εισερχόμενης κίνησης. Κατά τις περιόδους αιχμής, διαχειριζόταν υψηλότερες αιτήσεις ανά δευτερόλεπτο (RPS), επεξεργαζόμενος τις αιτήσεις χωρίς σημαντικές πτώσεις στην απόδοση. Αυτό υποδηλώνει ότι το σύστημα μπορεί να ανταποκρίνεται σε απαιτήσεις πραγματικού χρόνου χωρίς να αντιμετωπίζει καθυστερήσεις ή συμφόρηση.

Σύνοψη των επιδόσεων του συστήματος

Τα πειράματα παρείχαν μια ολοκληρωμένη εικόνα της συμπεριφοράς του συστήματος υπό διάφορα σενάρια φορτίου, από σταδιακή αύξηση μέχρι ταχείες διακυμάνσεις της κίνησης. Ο μηχανισμός κλιμάκωσης KPA αποδείχθηκε ιδιαίτερα αποδοτικός, προσαρμόζοντας δυναμικά τη χρήση της CPU και της μνήμης για την αντιμετώπιση των μεταβαλλόμενων απαιτήσεων, διατηρώντας διαχειρίσιμες τις ουρές μηνυμάτων και σταθερή απόδοση.

Τα κύρια συμπεράσματα από τα πειράματα περιλαμβάνουν:

- Το σύστημα μπορεί να διαχειριστεί ομαλά τόσο σταδιακές όσο και συχνές αυξήσεις φορτίου, προσαρμόζοντας την κατανομή πόρων χωρίς υπερβολική κλιμάκωση.
- Η συσσώρευση μηνυμάτων μειώθηκε στο ελάχιστο, με το σύστημα να εκκαθαρίζει αποτελεσματικά τις ουρές κατά τις φάσεις αδράνειας ή ελαφρύτερου φορτίου.
- Οι χρόνοι απόκρισης παρέμειναν χαμηλοί, εξασφαλίζοντας επεξεργασία αιτήσεων σε πραγματικό χρόνο, ακόμη και υπό συνεχή φόρτο.
- Ο δυναμικός μηχανισμός κλιμάκωσης του KPA επέτρεψε στο σύστημα να διατηρεί σταθερή απόδοση, να διαχειρίζεται αποτελεσματικά τις διακυμάνσεις φορτίου και να αποφεύγει την υπερκατανάλωση πόρων.

Συνολικά, τα πειράματα καταδεικνύουν ότι το σύστημα, με τη δυναμική κλιμάκωση KPA, μπορεί να διαχειριστεί αποτελεσματικά μια ποικιλία μοτίβων φορτίου, από σταδιακές αυξήσεις έως ταχείες εκρήξεις, χωρίς να επηρεαστεί η απόδοσή του ή να σημειωθούν καθυστερήσεις στην επεξεργασία μηνυμάτων. Το σύστημα είναι κατάλληλο για εφαρμογές πραγματικού χρόνου που απαιτούν δυναμική κατανομή πόρων και αποδοτική διαχείριση απρόβλεπτων μοτίβων κυκλοφορίας, όπως συστήματα επεξεργασίας βίντεο, συγκράτησης περιεχομένου ή ανάλυσης εικόνας.

5.5.11 Περιοχές Βελτίωσης και Δυνητικές Ανησυχίες κατά την Κλιμάκωση του Συστήματος με KPA

Ενώ οι συνολικές επιδόσεις του συστήματος στο πλαίσιο της κλιμάκωσης του KPA είναι ιδιαίτερα αποδοτικές, υπάρχουν μερικές μικρές ανησυχίες ή δυνητικοί τομείς για βελτίωση που θα μπορούσαν να επισημανθούν:

Μικρή συσσώρευση ουράς κατά τη διάρκεια υψηλής κίνησης: Παρόλο που το σύστημα διαχειρίστηκε αποτελεσματικά τις ουρές μηνυμάτων και απέτρεψε σημαντικές καθυστερήσεις, υπήρχαν στιγμές –ιδιαίτερα σε περιόδους συνεχούς ή μεγάλης έκρηξης– κατά τις οποίες το βάθος της ουράς εμφάνιζε αξιοσημείωτη αύξηση. Αυτό θα μπορούσε να υποδεικνύει ότι το σύστημα βρίσκεται κοντά στα όρια κλιμάκωσής του κατά τη διάρκεια της μέγιστης κίνησης, και σε πιο απαιτητικά σενάρια ή με υψηλότερο φορτίο, η συσσώρευση ουράς θα μπορούσε να γίνει πιο προβληματική.

Μεταβλητότητα στην κατανάλωση CPU μεταξύ των υπηρεσιών: Ενώ οι υπηρεσίες κλιμακώθηκαν κατάλληλα ως απάντηση στο φορτίο, η υπηρεσία καταναλωτή-2 εμφάνισε συχνά υψηλότερες αιχμές CPU σε σύγκριση με άλλες υπηρεσίες, ιδίως κατά τη διάρκεια συχνών εκρήξεων. Αυτή η μεταβλητότητα μπορεί να υποδηλώνει ότι ορισμένες υπηρεσίες είναι πιο απαιτητικές σε πόρους, οδηγώντας ενδεχομένως σε ανομοιόμορφη χρήση των πόρων. Θα άξιζε να διερευνηθεί η υποκείμενη αιτία για να εξασφαλιστεί πιο ισορροπημένη κλιμάκωση σε όλες τις υπηρεσίες.

Κλιμάκωση αντιγράφων με ελαφρά καθυστέρηση: Σε ορισμένες περιπτώσεις, υπήρχε μια σύντομη καθυστέρηση πριν το σύστημα αυξήσει τον αριθμό των ενεργών αντιγράφων κατά τη διάρκεια εκρήξεων κυκλοφορίας, ιδίως στα σενάρια «Συχνές εκρήξεις με μικρά κενά αδράνειας» και «Εναλλασσόμενος φόρτος». Παρόλο που αυτό δεν επηρέασε σημαντικά τη συνολική απόδοση, υποδηλώνει ότι μπορεί να υπάρχει περιθώριο βελτιστοποίησης στην κλιμάκωση αντιγράφων για ταχύτερη αντίδραση σε ξαφνικές αλλαγές της κίνησης, ιδίως σε συστήματα υψηλής ζήτησης και πραγματικού χρόνου.

Αιχμές μνήμης για την υπηρεσία καταναλωτή-3: Η κατανάλωση μνήμης στην υπηρεσία καταναλωτή-3 περιστασιακά εμφάνιζε υψηλότερες αιχμές από ό,τι στις άλλες υπηρεσίες, ιδίως κατά τη διάρκεια σεναρίων συνεχούς φόρτου. Παρόλο που αυτό δεν οδήγησε σε σημαντικά προβλήματα επιδόσεων, μπορεί να υποδηλώνει την ανάγκη βελτιστοποίησης ή καλύτερης διαχείρισης των πόρων εντός της εν λόγω υπηρεσίας για την αποφυγή της υπερδέσμευσης μνήμης μακροπρόθεσμα.

Μεταβλητότητα του χρόνου απόκρισης: Παρόλο που οι χρόνοι απόκρισης παρέμειναν εντός αποδεκτών ορίων καθ' όλη τη διάρκεια των δοκιμών, υπήρξαν στιγμές, ιδίως κατά τη διάρκεια ξαφνικών εκρήξεων κυκλοφορίας, όπου οι χρόνοι απόκρισης παρουσίαζαν διακυμάνσεις μεγαλύτερες από τις αναμενόμενες. Πρόκειται για μια μικρή ανησυχία, αλλά υποδηλώνει ότι υπό υψηλότερα φορτία, το σύστημα θα μπορούσε να εμφανίσει πιο αισθητές καθυστερήσεις, οι οποίες θα μπορούσαν να επηρεάσουν εφαρμογές ευαίσθητες στην καθυστέρηση.

Πιθανές ανησυχίες:

Το σύστημα ενδέχεται να αντιμετωπίσει προκλήσεις με υψηλότερα ή παρατεταμένα φορτία κυκλοφορίας, ιδίως σε περιβάλλοντα πραγματικού χρόνου ή υψηλής ζήτησης, όπου η συσσώρευση ουρών αναμονής θα μπορούσε να κλιμακωθεί πέρα από τα διαχειρίσιμα όρια.

Η χρήση της CPU και της μνήμης, αν και γενικά καλά ελεγχόμενη, θα μπορούσε να βελτιστοποιηθεί περαιτέρω για την αποφυγή περιστασιακών αιχμών, ιδίως στις υπηρεσίες καταναλωτή-2 και καταναλωτή-3.

Η καθυστέρηση στην κλιμάκωση αντιγράφων κατά τη διάρκεια ταχέων εκρήξεων κυκλοφορίας θα μπορούσε να γίνει πιο προβληματική σε σενάρια που απαιτούν εξαιρετικά γρήγορη κλιμάκωση και επεξεργασία σε πραγματικό χρόνο.

5.6 Συμπεριφορά Κλιμάκωσης με Horizontal Pod Autoscaler (HPA) υπό Μεταβαλλόμενες Συνθήκες Φορτίου

Στην παρούσα ενότητα, εξετάζεται η απόδοση του Horizontal Pod Autoscaler (HPA) μέσω μιας σειράς ελεγχόμενων πειραμάτων, τα οποία αντικατοπτρίζουν τις συνθήκες των δοκιμών που χρησιμοποιήθηκαν για τον Knative Pod Autoscaler (KPA). Ο HPA βασίζεται στη μέτρηση της χρήσης της CPU (Κεντρικής Μονάδας Επεξεργασίας) για να κλιμακώνει δυναμικά τον αριθμό των pods, σε αντίθεση με τον KPA, ο οποίος βασίζεται στην ταυτόχρονη χρήση. Σκοπός αυτής της ενότητας είναι να αξιολογηθεί η αποδοτικότητα, η επεκτασιμότητα και η απόκριση του HPA κάτω από ποικίλα φορτία και μοτίβα κίνησης, και να παρασχεθεί μια συγκριτική ανάλυση των αποτελεσμάτων που προέκυψαν με τη χρήση του KPA.

Τα πειράματα επικεντρώνονται σε σενάρια πραγματικού κόσμου, μιμούμενα τυπικές περιπτώσεις χρήσης σε περιβάλλοντα βασισμένα στο νέφος και χωρίς διακομιστή, όπως η επεξεργασία παρτίδων, η ανάλυση εικόνας και η συγκράτηση περιεχομένου. Προσομοιώνονται υψηλά φορτία κίνησης, εναλλασσόμενες εκρήξεις και φάσεις αδράνειας, με σκοπό να παρατηρηθεί η ικανότητα του HPA να διαχειρίζεται την κατανομή πόρων, την κατανομή φορτίου και την επεξεργασία μηνυμάτων, αποφεύγοντας καθυστερήσεις ή αναποτελεσματική χρήση των πόρων.

Η ανάλυση περιλαμβάνει μια λεπτομερή μελέτη της συμπεριφοράς του HPA υπό διαφορετικές διαμορφώσεις στόχων CPU, οι οποίες προσαρμόζονται για κάθε δοκιμή ώστε να βελτιστοποιηθεί η απόδοση βάσει των απαιτήσεων του συστήματος. Στόχος είναι να αξιολογηθεί κατά πόσο ο HPA μπορεί να διατηρήσει σταθερή απόδοση, να ελαχιστοποιήσει τη συσσώρευση μηνυμάτων και να επιτύχει αποτελεσματική κλιμάκωση σε ποικίλα μοτίβα κίνησης. Επιπλέον, τα πειράματα θα παράσχουν πληροφορίες για τις διαφορές στη συμπεριφορά μεταξύ της κλιμάκωσης βάσει της CPU (HPA) και της κλιμάκωσης βάσει της

ταυτόχρονης χρήσης (KPA), προσφέροντας έτσι μια ολοκληρωμένη αξιολόγηση της κλιμάκωσης και της προσαρμοστικότητας του συστήματος.

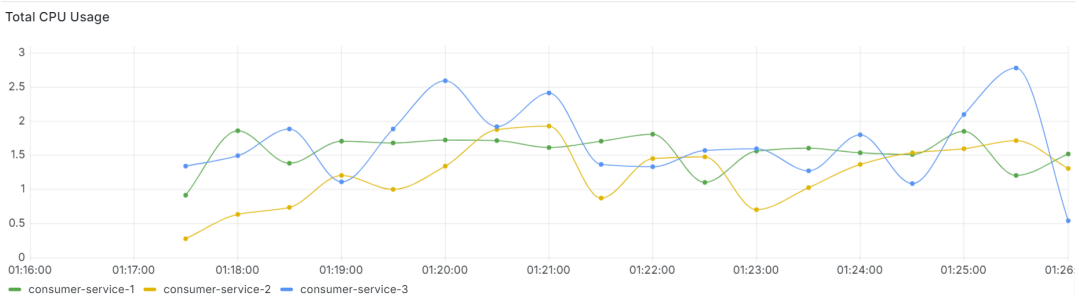
5.6.1 Υψηλό φορτίο με ανάκτηση σε αδράνεια

Στόχος χρήσης CPU: 75%

Σε αυτή την υποενότητα, το σύστημα υποβάλλεται σε συνθήκες συνεχούς υψηλού φορτίου, ακολουθούμενες από μια περίοδο πλήρους αδράνειας. Ο στόχος χρήσης CPU έχει οριστεί στο 75%, εξασφαλίζοντας ότι το σύστημα κατανέμει αποτελεσματικά τους πόρους κατά τη φάση υψηλού φορτίου. Αυτό επιτρέπει τη διαχείριση των εισερχόμενων αιτημάτων χωρίς καθυστερήσεις ή συσσώρευση μηνυμάτων. Η περίοδος αδράνειας ελέγχει την ικανότητα του συστήματος να μειώνει δυναμικά την κλίμακα και να απελευθερώνει πόρους. Αυτό το πείραμα θα παρέχει σημαντικές πληροφορίες για την αποτελεσματικότητα του συστήματος υπό αντίθετες συνθήκες κυκλοφορίας.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Από το γράφημα παρατηρούμε ότι η χρήση της CPU παρουσιάζει μέτριες διακυμάνσεις και στις τρεις υπηρεσίες. Η υπηρεσία-καταναλωτής-2 παρουσιάζει τις υψηλότερες αιχμές, ακολουθούμενη από την υπηρεσία-καταναλωτής-3, με την υπηρεσία-καταναλωτής-1 να διατηρεί χαμηλότερη αλλά σταθερή χρήση. Οι φάσεις εναλλασσόμενου φορτίου μεταξύ 1 και 5 αιτήσεων ανά δευτερόλεπτο παρουσιάζουν σαφείς κορυφές που αντιστοιχούν στο υψηλότερο φορτίο, και οι υπηρεσίες γενικά κλιμακώνουν αποτελεσματικά τη χρήση της CPU τους. Ωστόσο, η υπηρεσία-καταναλωτής-2 φαίνεται να κλιμακώνεται ταχύτερα σε σύγκριση με τις άλλες υπηρεσίες, ιδίως κατά τις φάσεις με υψηλότερη κίνηση.



Σχημα 5.61: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Συνολική χρήση CPU

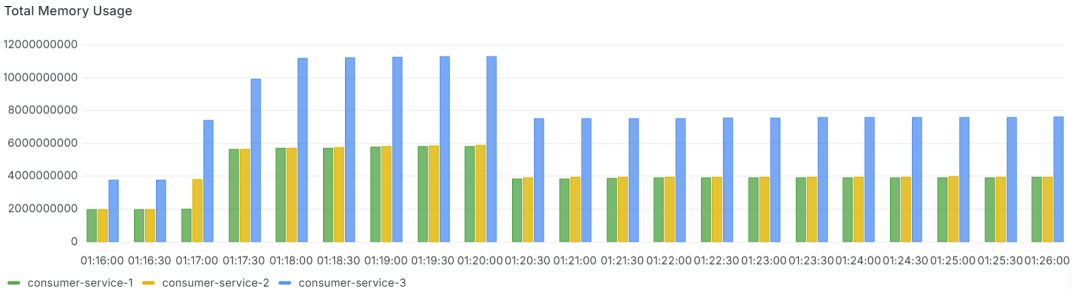
Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-2 παρουσιάζει τις σημαντικότερες αιχμές χρήσης CPU, ιδίως κατά τις περιόδους υψηλότερου φόρτου.
- Η υπηρεσία-καταναλωτής-3 ακολουθεί από κοντά, ενώ η υπηρεσία-καταναλωτής-1 διατηρεί συγκριτικά χαμηλότερη και πιο σταθερή χρήση CPU.
- Το σύστημα κλιμακώνεται δυναμικά, καθώς η χρήση της CPU προσαρμόζεται σε πραγματικό χρόνο για να διαχειρίζεται την κυκλοφορία χωρίς συνεχείς αιχμές.

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται κατά τη διάρκεια φάσεων υψηλότερου φόρτου (4-5 RPS) και να επιστρέφει σε χαμηλότερα επίπεδα κατά τη διάρκεια φάσεων ελαφρύτερου φόρτου (1 RPS). Ο μηχανισμός AIMD πιθανότατα βοηθά στη ρύθμιση της χρήσης των πόρων με τον αποτελεσματικό χειρισμό τόσο των αιχμών όσο και των βυθίσεων της κίνησης.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Στο γράφημα χρήσης μνήμης, βλέπουμε μια κάπως σταθερή κατανομή μνήμης στις τρεις υπηρεσίες, με την υπηρεσία καταναλωτή-3 να καταναλώνει συνήθως περισσότερη μνήμη, ακολουθούμενη από την υπηρεσία καταναλωτή-2 και την υπηρεσία καταναλωτή-1. Η χρήση της μνήμης κλιμακώνεται ως απόκριση στις φάσεις υψηλότερου φορτίου, αλλά δεν παρουσιάζει απότομες αιχμές, γεγονός που υποδηλώνει αποτελεσματική διαχείριση της μνήμης από το σύστημα υπό συνεχή φόρτο.



Σχημα 5.62: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Χρήση μνήμης

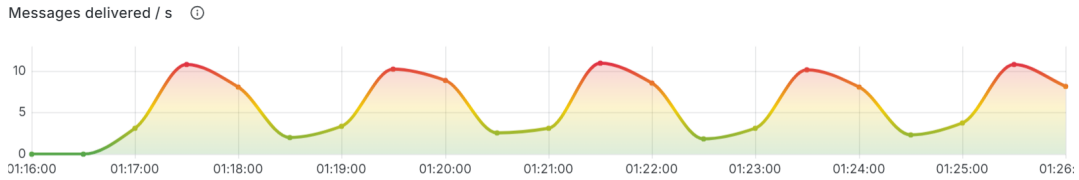
Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-3 καταναλώνει τη μεγαλύτερη μνήμη, ενώ η υπηρεσία-καταναλωτής-1 τη μικρότερη.
- Η χρήση της μνήμης παραμένει σχετικά σταθερή τόσο κατά τις περιόδους υψηλού όσο και κατά τις περιόδους χαμηλού φορτίου, με σταδιακή αύξηση και όχι απότομα άλματα.
- Το σύστημα διαχειρίζεται αποτελεσματικά τη μνήμη, αποτρέποντας την υπερκατανάλωση ακόμη και υπό συνεχή φόρτο.

Αναμενόμενη συμπεριφορά: Η χρήση μνήμης θα πρέπει να παραμένει σταθερή με μικρές μόνο αυξήσεις κατά τη διάρκεια φάσεων υψηλότερου φορτίου. Η ικανότητα του συστήματος να διατηρεί σταθερή χρήση μνήμης κατά τη διάρκεια εναλλασσόμενων φορτίων υποδηλώνει καλό χειρισμό της κατανομής των πόρων, αποφεύγοντας την εξάντληση της μνήμης.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (ρυθμός εξερχόμενων μηνυμάτων)

Το γράφημα του ρυθμού παράδοσης μηνυμάτων δείχνει σταθερές κορυφές παράδοσης που ευθυγραμμίζονται με τις φάσεις εναλλασσόμενου φορτίου. Κατά



Σχημα 5.63: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Παραδιδόμενα μηνύματα

τη διάρκεια φάσεων υψηλότερου φορτίου, ο ρυθμός παράδοσης κορυφώνεται σε περίπου 7,5 μηνύματα ανά δευτερόλεπτο, ο οποίος στη συνέχεια μειώνεται ελαφρώς κατά τη διάρκεια φάσεων ελαφρύτερου φόρτου. Αυτό υποδηλώνει ότι το σύστημα διατηρεί μια ομαλή ροή μηνυμάτων, κλιμακούμενο αποτελεσματικά για την αντιμετώπιση του εισερχόμενου φορτίου με ελάχιστο χρόνο αδράνειας μεταξύ των εκρήξεων.

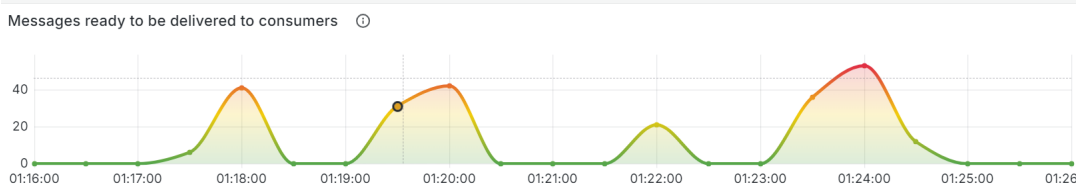
Βασικές παρατηρήσεις:

- Ο ρυθμός παράδοσης μηνυμάτων ταιριάζει με το αναμενόμενο μοτίβο κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια φάσεων υψηλού φόρτου και ελαφριά πτώση κατά τη διάρκεια ελαφρύτερων φάσεων.
- Το σύστημα διατηρεί μια συνεπή ροή μηνυμάτων, αποδεικνύοντας την αποτελεσματική διαχείριση της εξερχόμενης κίνησης.

Αναμενόμενη συμπεριφορά: Ο ρυθμός εξερχόμενων μηνυμάτων θα πρέπει να ακολουθεί στενά το μοτίβο εισερχόμενης κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια μεγαλύτερων φορτίων. Η ικανότητα του συστήματος να διαχειρίζεται συνεχή παράδοση μηνυμάτων με ελάχιστο χρόνο αδράνειας υποδηλώνει αποτελεσματική κλιμάκωση.

Μήκος ουράς (μηνύματα έτοιμα προς παράδοση)

Το γράφημα βάθους ουράς υποδεικνύει μια ελαφρά συσσώρευση μηνυμάτων κατά τη διάρκεια φάσεων υψηλού φόρτου, ιδίως προς το τέλος κάθε ριπής. Ωστόσο, η συσσώρευση δεν είναι υπερβολική και το σύστημα είναι σε θέση να καθαρίζει την ουρά κατά τη διάρκεια περιόδων με μικρότερο φορτίο. Η ουρά δεν αυξάνεται ποτέ πέρα από ένα διαχειρίσιμο επίπεδο, αποδεικνύοντας την ικανότητα του συστήματος να επεξεργάζεται αιτήσεις χωρίς σημαντικές καθυστερήσεις.



Σχημα 5.64: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Μήκος ουράς

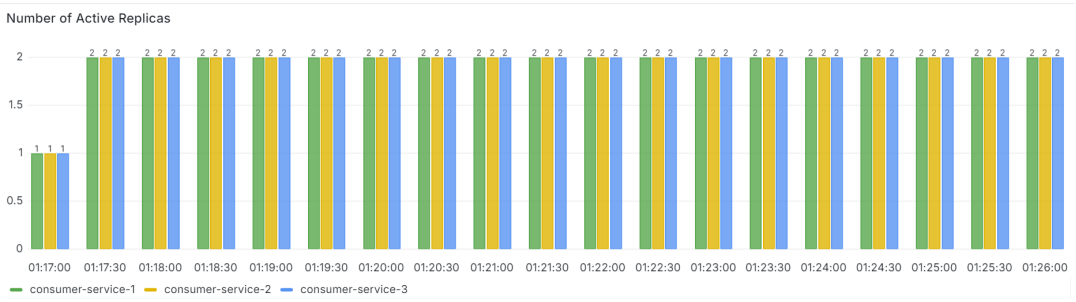
Βασικές παρατηρήσεις:

- Υπάρχει μια μικρή συσσώρευση μηνυμάτων κατά τη διάρκεια φάσεων υψηλότερου φόρτου, η οποία όμως εκκαθαρίζεται κατά τη διάρκεια ελαφρύτερων φάσεων.
- Το σύστημα αποτρέπει την υπερβολική συσσώρευση μηνυμάτων, διασφαλίζοντας την ομαλή ροή μηνυμάτων προς τους καταναλωτές.

Αναμενόμενη συμπεριφορά: Αναμένεται κάποια συσσώρευση ουράς κατά τη διάρκεια περιόδων υψηλού φόρτου, αλλά το σύστημα θα πρέπει να επεξεργάζεται αποτελεσματικά αυτά τα μηνύματα κατά τη διάρκεια ελαφρύτερων φάσεων. Το γεγονός ότι η ουρά παραμένει μικρή υποδηλώνει τον καλό χειρισμό από το σύστημα των εναλλασσόμενων φορτίων με ελάχιστο χρόνο αδράνειας.

Αριθμός ενεργών αντιγράφων

Ο αριθμός των ενεργών αντιγράφων αυξομειώνεται, με τις περισσότερες υπηρεσίες να διατηρούν 2 αντίγραφα καθ' όλη τη διάρκεια της δοκιμής. Η υπηρεσία καταναλωτή-2 κλιμακώνεται σε 3 αντίγραφα για λίγο κατά τη διάρκεια φάσεων με μεγαλύτερο φορτίο, ενώ η υπηρεσία καταναλωτή-1 και η υπηρεσία καταναλωτή-3 παραμένουν ως επί το πλείστον στα 2 αντίγραφα. Αυτή η δυναμική κλιμάκωση δείχνει ότι το σύστημα προσθέτει αποτελεσματικά πόρους όταν χρειάζεται κατά τη διάρκεια φάσεων υψηλού φόρτου και μειώνει την κλίμακα κατά τη διάρκεια ασθενέστερων περιόδων.



Σχημα 5.65: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Αριθμός ενεργών αντιγράφων

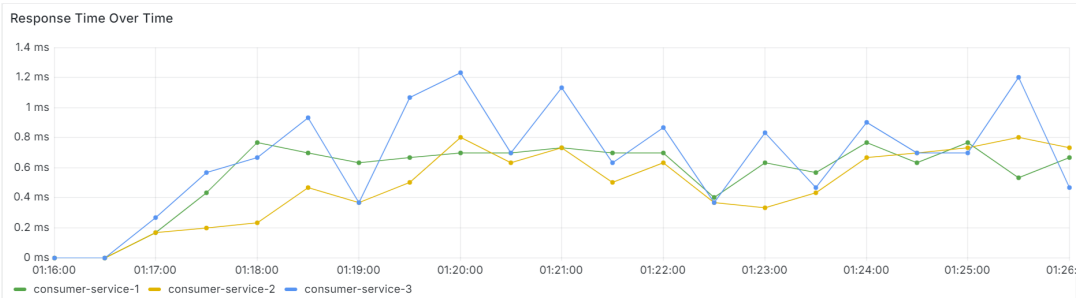
Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-2 κλιμακώνεται για λίγο σε 3 αντίγραφα, ενώ οι άλλες υπηρεσίες παραμένουν γενικά σε 2 αντίγραφα.
- Το σύστημα προσαρμόζει δυναμικά τον αριθμό των αντιγράφων με βάση το εισερχόμενο φορτίο, υποδεικνύοντας αποτελεσματική κλιμάκωση.

Αναμενόμενη συμπεριφορά: Η κλιμάκωση των αντιγράφων θα πρέπει να γίνεται σε ανταπόκριση στην εισερχόμενη κίνηση, με περισσότερα αντίγραφα κατά τη διάρκεια φάσεων υψηλού φόρτου. Η ικανότητα του συστήματος να κλιμακώνει δυναμικά τα αντίγραφα συμβάλλει στη διατήρηση της απόδοσης χωρίς υπερφόρτωση οποιασδήποτε μεμονωμένης υπηρεσίας.

Χρόνος απόκρισης με την πάροδο του χρόνου

Το γράφημα του χρόνου απόκρισης παρουσιάζει κάποια διακύμανση, με αιχμές κατά τη διάρκεια φάσεων υψηλότερου φόρτου. Ωστόσο, ο συνολικός χρόνος απόκρισης παραμένει κάτω από 1 ms για το μεγαλύτερο μέρος της δοκιμής, γεγονός που υποδηλώνει ότι το σύστημα είναι σε θέση να διαχειριστεί το εναλλασσόμενο φορτίο χωρίς σημαντικές καθυστερήσεις. Η υπηρεσία καταναλωτή-3 παρουσιάζει υψηλότερες αιχμές στο χρόνο απόκρισης κατά τη διάρκεια φάσεων υψηλού φόρτου, ενώ η υπηρεσία καταναλωτή-1 διατηρεί τους πιο σταθερούς χρόνους απόκρισης.



Σχημα 5.66: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Χρόνος απόκρισης

Βασικές παρατηρήσεις:

- Οι χρόνοι απόκρισης παραμένουν ως επί το πλείστον κάτω από 1 ms, με μικρές αιχμές κατά τις φάσεις υψηλού φόρτου.
- Η υπηρεσία καταναλωτή-3 παρουσιάζει τις μεγαλύτερες διακυμάνσεις στους χρόνους απόκρισης, ενώ η υπηρεσία καταναλωτή-1 παραμένει πιο σταθερή.

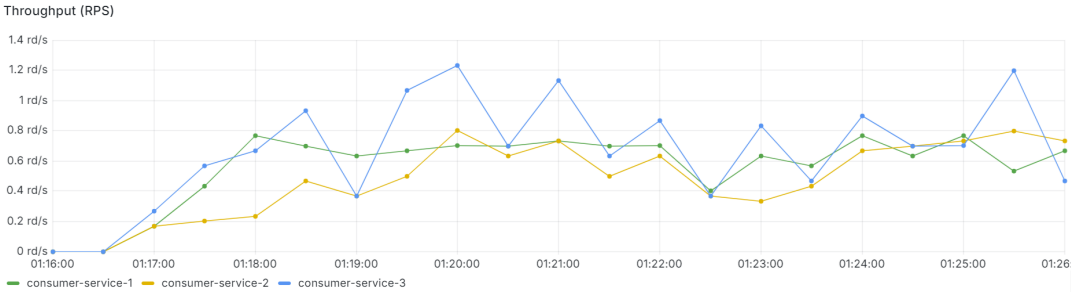
Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμένουν χαμηλοί, ακόμη και κατά τη διάρκεια φάσεων υψηλότερου φόρτου. Αναμένονται μικρές αυξήσεις κατά τη διάρκεια υψηλότερης κίνησης, αλλά το σύστημα θα πρέπει να αποφεύγει τις σημαντικές καθυστερήσεις, όπως φαίνεται στο γράφημα.

Απόδοση (RPS)

Το γράφημα ρυθμού μετάδοσης δείχνει ότι και οι τρεις υπηρεσίες διεκπεραιώνουν τα αιτήματα με σχετικά σταθερό ρυθμό, με την υπηρεσία καταναλωτή-2 να παρουσιάζει υψηλότερη ρυθμό μετάδοσης κατά τη διάρκεια φάσεων με μεγαλύτερο φόρτο. Το σύστημα διατηρεί την απόδοση σε αντιστοιχία με την εισερχόμενη κίνηση, διασφαλίζοντας ότι τα αιτήματα επεξεργάζονται αποτελεσματικά, ακόμη και όταν ελαχιστοποιούνται οι χρόνοι αδράνειας.

Βασικές παρατηρήσεις:

- Η υπηρεσία-καταναλωτής-2 παρουσιάζει υψηλότερη απόδοση κατά τη διάρκεια των φάσεων υψηλού φόρτου, με την υπηρεσία-καταναλωτής-3 και την υπηρεσία-καταναλωτής-1 να ακολουθούν από κοντά.



Σχημα 5.67: Πειράματα Κλιμάκωσης με HPA: Υψηλό φορτίο με ανάκτηση σε αδράνεια: Απόδοση (RPS)

- Το σύστημα διατηρεί σταθερή απόδοση, ταιριάζοντας αποτελεσματικά με το μοτίβο εναλλασσόμενου φορτίου.

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να ταιριάζει με το μοτίβο εισερχόμενης κίνησης, με υψηλότερους ρυθμούς κατά τη διάρκεια φάσεων υψηλού φόρτου. Η ικανότητα του συστήματος να διατηρεί σταθερή ρυθμαπόδοση υπό συνεχή φόρτο καταδεικνύει αποτελεσματική επεξεργασία των αιτήσεων.

Σύγκριση με τα αντίστοιχα αποτελέσματα KPA

Στα αποτελέσματα του KPA, είδαμε ότι το σύστημα κλιμακώνεται ταχύτερα και αποτελεσματικότερα για να ανταποκριθεί στις απαιτήσεις της φάσης υψηλού φόρτου. Η χρήση της CPU κορυφώθηκε παρόμοια γύρω στους 2 πυρήνες, αλλά το σύστημα μειώθηκε ταχύτερα κατά τη φάση αδράνειας. Η χρήση μνήμης παρουσίασε επίσης παρόμοιο μοτίβο, αλλά με ταχύτερη απελευθέρωση κατά τη διάρκεια των περιόδων αδράνειας. Η παράδοση μηνυμάτων και η δημιουργία ουρών αναμονής ήταν ελάχιστες με τον KPA λόγω της ικανότητάς του να αντιδρά ταχύτερα στις αλλαγές φορτίου, και ο χρόνος απόκρισης ήταν πιο σταθερός.

Βασικές συγκριτικές παρατηρήσεις:

- Ο KPA παρουσίασε ταχύτερους χρόνους αντίδρασης τόσο στην αύξηση όσο και στη μείωση της κλίμακας, καθιστώντας τον πιο αποτελεσματικό στην αποφυγή της συσσώρευσης μηνυμάτων.
- Ο HPA είχε πιο αργή συμπεριφορά κλιμάκωσης, οδηγώντας σε μεγαλύτερη συσσώρευση μηνυμάτων κατά τη φάση υψηλού φόρτου και πιο αργή ανάκαμψη κατά τη διάρκεια των περιόδων αδράνειας.
- Τόσο ο HPA όσο και ο KPA παρουσίασαν παρόμοιες κορυφές στη χρήση CPU και στους ρυθμούς παράδοσης μηνυμάτων, αλλά ο KPA διατήρησε πιο σταθερούς χρόνους απόκρισης και ρυθμούς απόδοσης.

Συμπέρασμα: Ενώ τόσο ο HPA όσο και ο KPA κατάφεραν να διαχειριστούν το υψηλό φορτίο χωρίς να προκαλέσουν κρίσιμες αποτυχίες, ο KPA αποδείχθηκε πιο αποτελεσματικός στην απόκριση στις αιχμές της κίνησης και στην άμεση εκκαθάριση της ουράς μηνυμάτων. Ο HPA, από την άλλη πλευρά, έδειξε βραδύτερη ανταπόκριση στην κλιμάκωση κατά τη διάρκεια περιόδων αδράνειας, οδηγώντας σε διατήρηση πόρων και μικρές καθυστερήσεις στην επεξεργασία

μηνυμάτων. Αυτό θα μπορούσε να υποδηλώνει ότι για φόρτους εργασίας με εξαιρετικά μεταβλητά πρότυπα κίνησης, ο KPA είναι πιο κατάλληλος για δυναμική κλιμάκωση, ενώ ο HPA μπορεί να είναι καλύτερος για συνεχή ή σταδιακά αυξανόμενα φορτία.

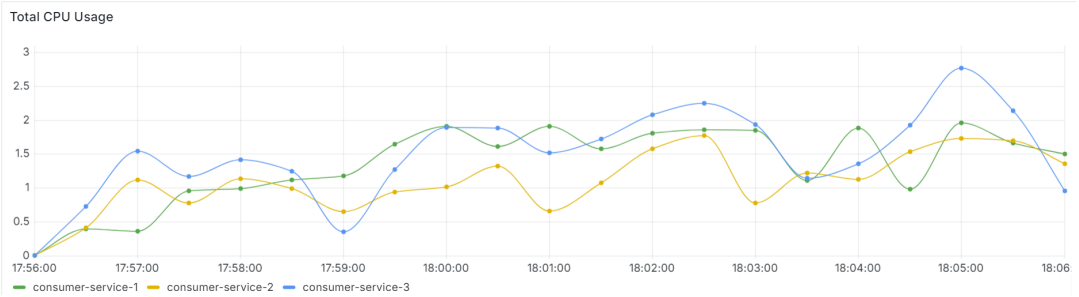
5.6.2 Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας

Στόχος χρήσης CPU: 60%

Αυτό το πείραμα εξετάζει εναλλασσόμενα φορτία, με κυμαινόμενα αιτήματα ανά δευτερόλεπτο και σύντομες φάσεις αδράνειας. Ο στόχος χρήσης CPU ορίζεται στο 60%, διατηρώντας ένα μετριοπαθές όριο για την κλιμάκωση του συστήματος, επιτρέποντας δυναμικές προσαρμογές στις διακυμάνσεις της κίνησης. Οι σύντομες φάσεις αδράνειας δοκιμάζουν την ικανότητα του συστήματος να μειώνει γρήγορα την κλίμακα και να προετοιμάζεται για τον επόμενο κύκλο φορτίου. Αυτή η υποενότητα επικεντρώνεται στη διαχείριση των συχνών αλλαγών στην κυκλοφορία και στην ικανότητα του συστήματος να διατηρεί ομαλή λειτουργία.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Από το γράφημα χρήσης CPU, μπορούμε να δούμε ότι η κατανάλωση CPU και των τριών καταναλωτικών υπηρεσιών κυμαίνεται σύμφωνα με τις εναλλασσόμενες φάσεις φορτίου. Η υπηρεσία 3 τείνει να εμφανίζει ελαφρώς υψηλότερες αιχμές σε σύγκριση με τις άλλες δύο υπηρεσίες, ιδίως σε φάσεις υψηλότερης κίνησης, αλλά συνολικά, όλες οι υπηρεσίες διατηρούν ένα ισορροπημένο μοτίβο κατανάλωσης CPU. Οι μεταβάσεις μεταξύ υψηλότερων και χαμηλότερων φορτίων κυκλοφορίας αντικατοπτρίζονται στις σταδιακές κορυφές και κοιλάδες της χρήσης της CPU. Οι φάσεις αδράνειας είναι εμφανείς καθώς η χρήση της CPU βυθίζεται πιο κοντά στη βασική γραμμή. Δεν υπάρχει ένδειξη σημαντικής υπερκατανάλωσης πόρων κατά τις περιόδους αδράνειας, γεγονός που δείχνει αποτελεσματική κλιμάκωση.



Σχημα 5.68: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Συνολική χρήση CPU

Βασικές παρατηρήσεις:

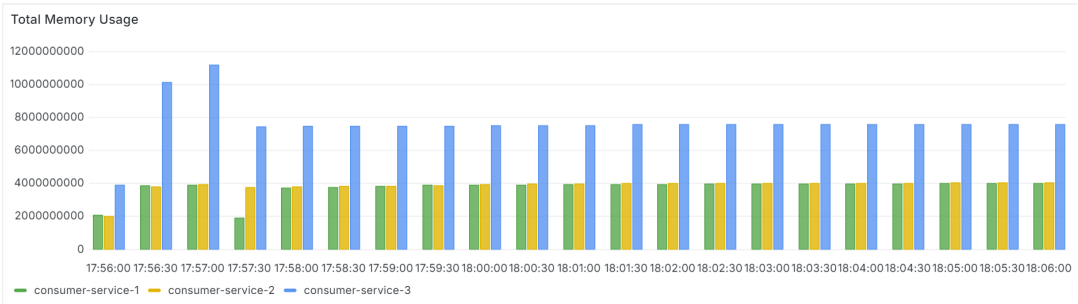
- Η υπηρεσία 3 παρουσιάζει τις πιο εμφανείς αιχμές κατά τη διάρκεια υψηλής κυκλοφορίας.
- Οι υπηρεσίες 1 και 2 παραμένουν κοντά στην κατανάλωση CPU, με ελαφρώς μικρότερες αιχμές σε σύγκριση με την υπηρεσία 3.

- Η χρήση CPU ακολουθεί αποτελεσματικά το μοτίβο του εναλλασσόμενου φόρτου κυκλοφορίας, προσαρμοζόμενη δυναμικά χωρίς δραστικές διακυμάνσεις.

Αναμενόμενη συμπεριφορά: Η κατανάλωση CPU θα πρέπει να συσχετίζεται με τις φάσεις εναλλασσόμενου φορτίου, αυξάνοντας σε περιόδους 2-4 RPS και μειώνοντας σε περιόδους αδράνειας ή ελαφρύτερου φορτίου. Αυτή η συμπεριφορά είναι σύμφωνη με την αναμενόμενη χρήση πόρων στο πλαίσιο της HPA, η οποία διαχειρίζεται αποτελεσματικά το φορτίο χωρίς περιττές αιχμές CPU κατά τις φάσεις αδράνειας.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Το γράφημα χρήσης μνήμης δείχνει μια λίγο πολύ σταθερή κατανομή της μνήμης στις υπηρεσίες, με την υπηρεσία 3 να καταναλώνει και πάλι τη μεγαλύτερη ποσότητα, ακολουθούμενη από την υπηρεσία 2 και την υπηρεσία 1 να χρησιμοποιεί τη λιγότερη μνήμη. Παρόμοια με το γράφημα της CPU, η χρήση μνήμης δεν παρουσιάζει ακραίες αιχμές και παραμένει σε ένα διαχειρίσιμο εύρος καθ' όλη τη διάρκεια των φάσεων εναλλασσόμενου φόρτου. Υπάρχει μια μικρή αύξηση στη χρήση μνήμης κατά τη διάρκεια περιόδων υψηλής κίνησης, αλλά το σύστημα μειώνεται γρήγορα κατά τις φάσεις αδράνειας.



Σχημα 5.69: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Χρήση μνήμης

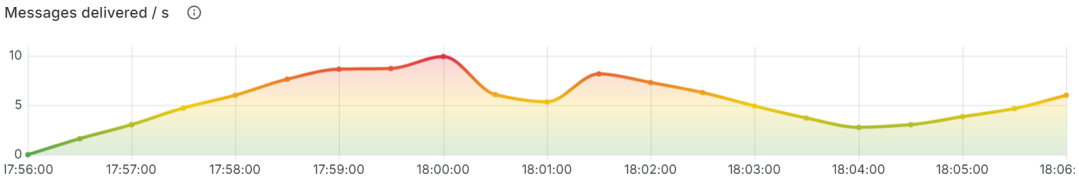
Βασικές παρατηρήσεις:

- Η υπηρεσία 3 καταναλώνει σταθερά περισσότερη μνήμη, ενώ η υπηρεσία 1 χρησιμοποιεί τη λιγότερη.
- Η χρήση μνήμης κλιμακώνεται μέτρια ως απόκριση σε υψηλότερα φορτία κυκλοφορίας χωρίς απότομες αυξήσεις.
- Το σύστημα αποδεσμεύει αποτελεσματικά μνήμη κατά τη διάρκεια περιόδων αδράνειας, γεγονός που υποδηλώνει αποτελεσματική διαχείριση των πόρων.

Αναμενόμενη συμπεριφορά: Η κατανάλωση μνήμης πρέπει να κλιμακώνεται ομαλά χωρίς απότομες αιχμές, γεγονός που αντικατοπτρίζει την αποτελεσματική κλιμάκωση. Η ικανότητα του συστήματος να διατηρεί σταθερή τη χρήση μνήμης κατά τη διάρκεια διαφορετικών φορτίων κυκλοφορίας αποδεικνύει ότι το HPA χειρίζεται αποτελεσματικά την κατανομή μνήμης.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (ρυθμός εξερχόμενων μηνυμάτων)

Το γράφημα των μηνυμάτων που παραδίδονται ανά δευτερόλεπτο δείχνει σαφείς κορυφές σε περιόδους υψηλότερης κίνησης, με ρυθμούς παράδοσης που φτάνουν περίπου τα 10 μηνύματα ανά δευτερόλεπτο στα υψηλότερα σημεία. Ο ρυθμός παράδοσης πέφτει κατά τις φάσεις αδράνειας, όπως ήταν αναμενόμενο, αλλά υπάρχει σταθερή ροή μηνυμάτων κατά τις περιόδους εναλλασσόμενου φόρτου. Το σύστημα προσαρμόζει αποτελεσματικά τον ρυθμό παράδοσης μηνυμάτων στην εισερχόμενη κίνηση, αποφεύγοντας σημαντικές καθυστερήσεις.



Σχημα 5.70: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Παραδιδόμενα μηνύματα

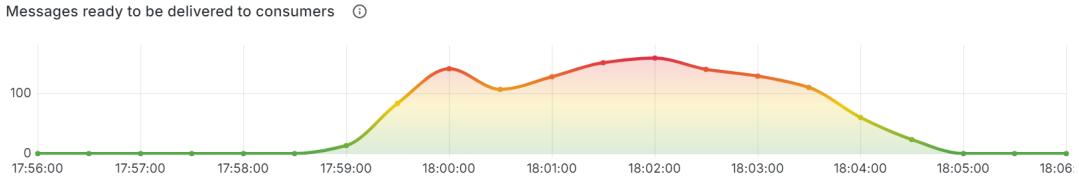
Βασικές παρατηρήσεις:

- Ο ρυθμός παράδοσης ευθυγραμμίζεται με τις φάσεις εναλλασσόμενου φόρτου, κορυφώνοντας κατά τη διάρκεια υψηλότερης κίνησης και μειώνοντας κατά τις φάσεις αδράνειας.
- Δεν υπάρχουν σημαντικές καθυστερήσεις στην παράδοση μηνυμάτων και το σύστημα χειρίζεται αποτελεσματικά τις μεταβάσεις μεταξύ των φορτίων κυκλοφορίας.

Αναμενόμενη συμπεριφορά: Ο ρυθμός παράδοσης θα πρέπει να ακολουθεί στενά το μοτίβο της κίνησης, με υψηλότερους ρυθμούς σε περιόδους 3-4 RPS και μείωση σε περιόδους αδράνειας. Το γεγονός ότι το σύστημα διατηρεί σταθερό ρυθμό παράδοσης μηνυμάτων υποδηλώνει αποτελεσματική κλιμάκωση.

Μήκος ουράς (μηνύματα έτοιμα προς παράδοση)

Το γράφημα βάθους ουράς μηνυμάτων υποδεικνύει μια μικρή συσσώρευση μηνυμάτων κατά τις φάσεις υψηλότερης κίνησης, ιδίως κατά την περίοδο φόρτου 4 RPS, όπου η ουρά φτάνει έως και τα 150 μηνύματα. Ωστόσο, το σύστημα εκκαθαρίζει επιτυχώς την ουρά κατά τις περιόδους με μικρότερη κίνηση, αποφεύγοντας την υπερβολική συσσώρευση μηνυμάτων. Το μήκος της ουράς παραμένει διαχειρίσιμο και δεν αυξάνεται πέραν του ελέγχου.



Σχημα 5.71: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Μήκος ουράς

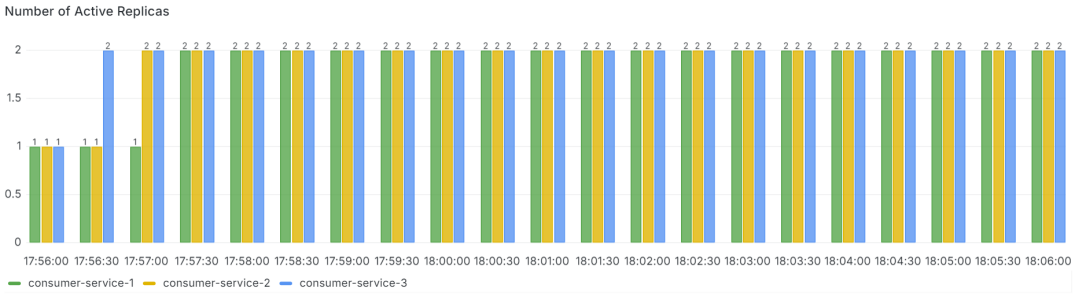
Βασικές παρατηρήσεις:

- Υπάρχει μέτρια συσσώρευση μηνυμάτων κατά τη διάρκεια υψηλότερης κίνησης, αλλά το σύστημα καθαρίζει αποτελεσματικά την ουρά κατά τη διάρκεια περιόδων με μικρότερο φόρτο.
- Το μήκος της ουράς παραμένει εντός διαχειρίσιμων ορίων, γεγονός που υποδηλώνει καλή απόδοση επεξεργασίας μηνυμάτων.

Αναμενόμενη συμπεριφορά: Αναμένουμε κάποια συσσώρευση μηνυμάτων κατά τη διάρκεια περιόδων μεγαλύτερης κίνησης, αλλά το σύστημα θα πρέπει να είναι σε θέση να επεξεργάζεται αποτελεσματικά τα μηνύματα αυτά κατά τη διάρκεια φάσεων ελαφρύτερου φόρτου, πράγμα που παρατηρούμε σε αυτή την περίπτωση.

Αριθμός ενεργών αντιγράφων

Το γράφημα αριθμού αντιγράφων δείχνει ότι οι περισσότερες υπηρεσίες διατηρούν 2 αντίγραφα καθ' όλη τη διάρκεια της δοκιμής, με σύντομες στιγμές όπου η υπηρεσία 2 κλιμακώνεται σε 3 αντίγραφα ως απάντηση στην υψηλότερη κίνηση. Η δυναμική κλιμάκωση των αντιγράφων αντικατοπτρίζει την ικανότητα του συστήματος να κατανέμει αποτελεσματικά τους πόρους με βάση τη ζήτηση κίνησης, ιδίως κατά τις σύντομες περιόδους υψηλότερου φόρτου.



Σχημα 5.72: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Αριθμός ενεργών αντιγράφων

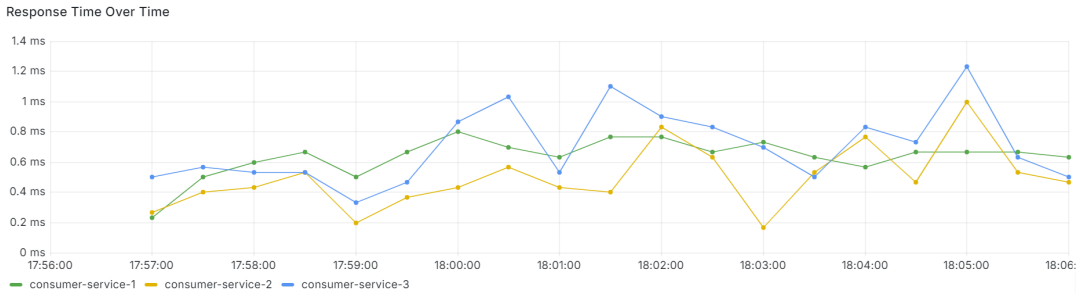
Βασικές παρατηρήσεις:

- Η υπηρεσία 2 κλιμακώνεται περιστασιακά σε 3 αντίγραφα κατά τη διάρκεια περιόδων υψηλής κίνησης, ενώ οι υπηρεσίες 1 και 3 παραμένουν ως επί το πλείστον σε 2 αντίγραφα.
- Το σύστημα προσαρμόζει δυναμικά τον αριθμό των αντιγράφων για να αντιμετωπίσει την αυξανόμενη κίνηση και στη συνέχεια μειώνει την κλίμακα κατά τη διάρκεια των φάσεων αδράνειας.

Αναμενόμενη συμπεριφορά: Ο αριθμός των αντιγράφων θα πρέπει να αυξάνεται κατά τη διάρκεια φάσεων υψηλού φόρτου και να μειώνεται κατά τη διάρκεια περιόδων αδράνειας. Η ικανότητα του συστήματος να προσαρμόζει δυναμικά τον αριθμό των αντιγράφων εξασφαλίζει αποτελεσματική χρήση των πόρων.

Χρόνος απόκρισης με την πάροδο του χρόνου

Το γράφημα του χρόνου απόκρισης παρουσιάζει μέτρια διακύμανση, με αιχμές κατά τη διάρκεια υψηλότερων φορτίων κυκλοφορίας. Ωστόσο, ο χρόνος απόκρισης παραμένει ως επί το πλείστον κάτω από 1 ms, το οποίο είναι αποδεκτό για τον φόρτο εργασίας. Η υπηρεσία 3 παρουσιάζει τους υψηλότερους χρόνους απόκρισης, ιδίως κατά τη διάρκεια της αιχμής της κυκλοφορίας, ενώ η υπηρεσία 1 διατηρεί πιο σταθερούς χρόνους απόκρισης καθ' όλη τη διάρκεια της δοκιμής.



Σχημα 5.73: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Χρόνος απόκρισης

Βασικές παρατηρήσεις:

- Οι χρόνοι απόκρισης είναι ως επί το πλείστον σταθεροί, με την υπηρεσία 3 να παρουσιάζει τις υψηλότερες αιχμές.
- Η υπηρεσία 1 παρουσιάζει τους πιο σταθερούς χρόνους απόκρισης.

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμείνουν χαμηλοί, με μικρές αυξήσεις σε περιόδους υψηλότερης κίνησης. Η ικανότητα του συστήματος να διατηρεί τους χρόνους απόκρισης κάτω από 1 ms κατά τη διάρκεια του μεγαλύτερου μέρους της δοκιμής αποδεικνύει καλή απόδοση.

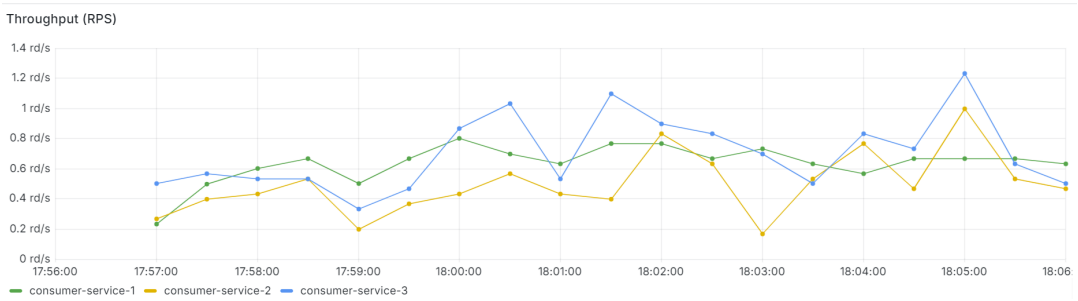
Απόδοση (RPS)

Το γράφημα της απόδοσης αντικατοπτρίζει το εναλλασσόμενο φορτίο κίνησης, με την απόδοση να κορυφώνεται σε περιόδους υψηλής κίνησης και να μειώνεται σε φάσεις αδράνειας. Η υπηρεσία 3 παρουσιάζει σταθερά υψηλότερη απόδοση, ενώ οι υπηρεσίες 1 και 2 ακολουθούν από κοντά. Η ρυθμαπόδοση παραμένει σταθερή και ταιριάζει στενά με την εισερχόμενη κυκλοφορία.

Βασικές παρατηρήσεις:

- Η υπηρεσία 3 παρουσιάζει σταθερά υψηλότερη απόδοση κατά τη διάρκεια της αιχμής της κυκλοφορίας.
- Η απόδοση ταιριάζει με το μοτίβο εναλλασσόμενου φόρτου, αποδεικνύοντας την αποτελεσματική διαχείριση των αιτήσεων.

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να ταιριάζει με τον εισερχόμενο φόρτο κίνησης, με υψηλότερους ρυθμούς σε περιόδους 3-4 RPS. Η ικανότητα του συστήματος να χειρίζεται αποτελεσματικά την εναλλασσόμενη κίνηση αντικατοπτρίζεται στη σταθερή απόδοση.



Σχημα 5.74: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με σύντομες φάσεις αδράνειας: Απόδοση (RPS)

Σύγκριση με τα αντίστοιχα αποτελέσματα KPA

Όταν συγκρίνουμε τα αποτελέσματα από τον KPA, παρατηρούμε παρόμοια μοτίβα αλλά με μια βασική διαφορά στην ανταπόκριση:

Ο KPA αντιδρά ταχύτερα τόσο στην αύξηση όσο και στη μείωση της κλίμακας κατά τη διάρκεια του εναλλασσόμενου φορτίου, με λιγότερα συσσωρευμένα μηνύματα κατά τη διάρκεια των φορτίων αιχμής. Ο HPA, αν και αποτελεσματικός, επιδεικνύει πιο αργή κλιμάκωση, οδηγώντας σε ελαφρώς υψηλότερη συσσώρευση ουράς κατά τη διάρκεια του μέγιστου φορτίου.

Βασικές συγκριτικές παρατηρήσεις:

- **Χρήση CPU**: Τόσο ο HPA όσο και ο KPA παρουσιάζουν παρόμοιες κορυφές CPU, αλλά ο KPA κλιμακώνεται ταχύτερα κατά τη διάρκεια των φάσεων αδράνειας.
- **Χρήση μνήμης**: Τα μοτίβα χρήσης μνήμης είναι παρόμοια και στις δύο περιπτώσεις, με τον KPA να χειρίζεται πιο αποτελεσματικά τις περιόδους αδράνειας.
- **Παράδοση μηνυμάτων**: Ο KPA παρουσιάζει πιο αποτελεσματική παράδοση, ιδίως κατά τη διάρκεια υψηλού φόρτου.
- **Βάθος ουράς**: Ο HPA παρουσιάζει μεγαλύτερη συσσώρευση από τον KPA κατά τη διάρκεια υψηλού φόρτου.
- **Χρόνος απόκρισης**: Ο KPA διατηρεί πιο σταθερούς χρόνους απόκρισης κατά τη διάρκεια διακυμάνσεων φορτίου.

Συμπέρασμα: Ενώ τόσο ο HPA όσο και ο KPA χειρίζονται με επιτυχία τα εναλλασσόμενα φορτία, ο KPA παρουσιάζει ταχύτερη και αποδοτικότερη απόκριση κλιμάκωσης. Αυτό οδηγεί σε καλύτερη διαχείριση ουρών αναμονής και πιο σταθερή απόδοση, ιδίως σε σενάρια με ιδιαίτερα δυναμικά μοτίβα κίνησης. Ο HPA, αν και πιο αργός, εξακολουθεί να διατηρεί τη σταθερότητα και να χειρίζεται αποτελεσματικά τις εκρήξεις.

5.6.3 Σύντομες εκρήξεις φορτίου υψηλής έντασης

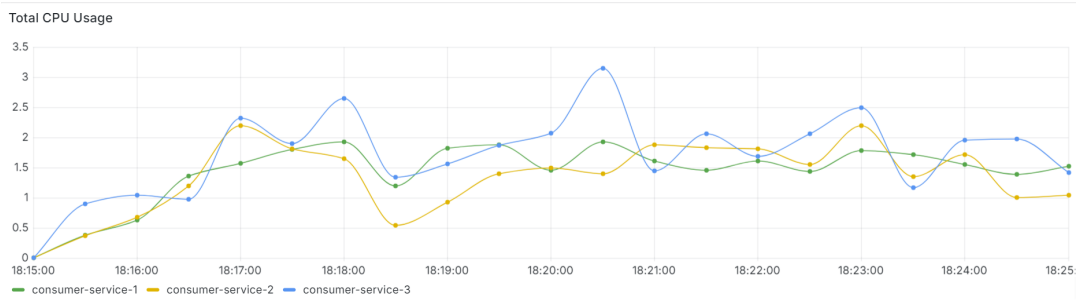
Στόχος χρήσης CPU: 85%

Σε αυτή τη δοκιμή, το σύστημα εκτίθεται σε σύντομες, υψηλής έντασης εκρήξεις κυκλοφορίας, που απαιτούν ταχεία κλιμάκωση για την αποφυγή συσσώρευσης

μηνυμάτων. Ο στόχος χρήσης CPU στο 85% αντανακλά την ανάγκη για αυξημένη κατανομή πόρων για την κάλυψη αυτών των αιχμών. Το πείραμα θα αναλύσει την ικανότητα του συστήματος να κλιμακώνεται γρήγορα, διατηρώντας τη σταθερότητα και διασφαλίζοντας ότι κανένα μήνυμα δεν παραμένει ανεπεξέργαστο κατά τη διάρκεια αυτών των περιόδων υψηλής ζήτησης.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Παρατηρήσεις: Η χρήση της CPU σε όλες τις υπηρεσίες εμφανίζει σαφείς αιχμές σύμφωνα με τα πρότυπα της εκρηκτικής κυκλοφορίας, με την υπηρεσία-3 να εμφανίζει πιο έντονες αιχμές που φτάνουν έως και 3 πυρήνες στην υψηλότερη τιμή τους. Οι υπηρεσίες 1 και 2 παραμένουν σε ένα εύρος 1-2 πυρήνων κατά τη διάρκεια των εκρήξεων. Κάθε έκρηξη οδηγεί σε μια σημαντική αύξηση στη χρήση της CPU, ακολουθούμενη από γρήγορη μείωση μόλις η έκρηξη υποχωρήσει.



Σχημα 5.75: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Συνολική χρήση CPU

Αναμενόμενη συμπεριφορά: Το σύστημα αναμένεται να κατανέμει δυναμικά τους πόρους της CPU ώστε να ταιριάζει με τις εισερχόμενες εκρήξεις. Η ταχεία αύξηση της χρήσης της CPU κατά τη διάρκεια κάθε έκρηξης, ακολουθούμενη από μείωση, αποτελεί ένδειξη αποτελεσματικής κλιμάκωσης και άμεσης ανταπόκρισης στην κίνηση, διασφαλίζοντας την επεξεργασία των αιτημάτων χωρίς καθυστερήσεις.

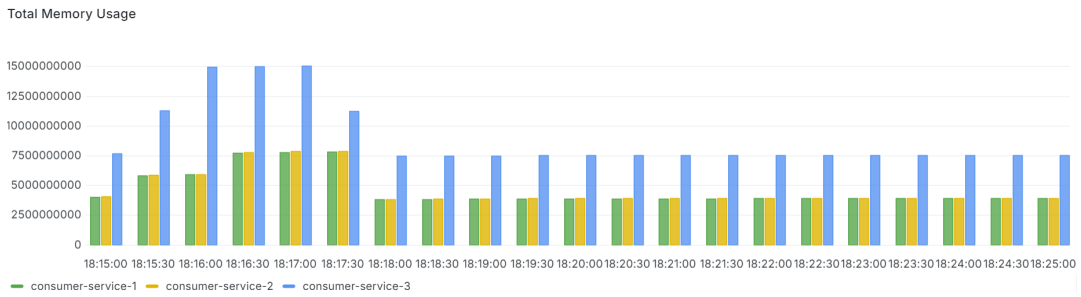
Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Παρατήρηση: Η χρήση της μνήμης παραμένει σχετικά σταθερή σε όλες τις υπηρεσίες, με την υπηρεσία-3 να καταναλώνει σταθερά περισσότερη μνήμη, αντανακλώντας τις υψηλότερες ανάγκες της σε πόρους. Δεν παρατηρείται απότομη αύξηση της μνήμης ακόμη και σε περιόδους αιχμής της κυκλοφορίας.

Αναμενόμενη συμπεριφορά: Η κατανάλωση μνήμης αναμένεται να παραμένει σταθερή, χωρίς απότομες αιχμές ή υπερβολική συσσώρευση. Η ομαλή κατανάλωση μνήμης κατά τη διάρκεια των περιόδων υψηλής κυκλοφορίας δείχνει ότι το σύστημα δεν υπερκατανέμει πόρους άσκοπα.

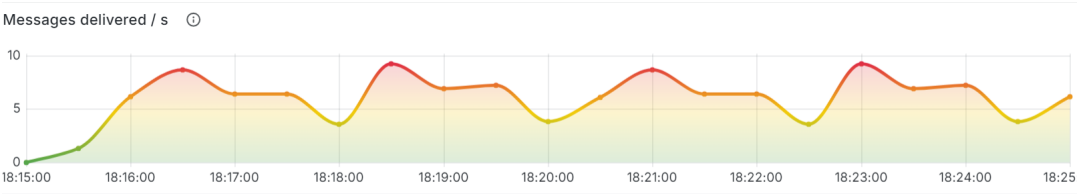
Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (Throughput)

Παρατήρηση: Ο ρυθμός παράδοσης των μηνυμάτων αυξάνεται ομαλά με την εκρηκτική κυκλοφορία, κορυφώνοντας γύρω στα 10 μηνύματα ανά δευτερόλεπτο



Σχημα 5.76: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Χρήση μνήμης

στο αποκορύφωμα κάθε έκρηξης. Ο ρυθμός παραμένει συνεπής και ακολουθεί το μοτίβο των εκρήξεων χωρίς καθυστερήσεις ή πτώσεις.

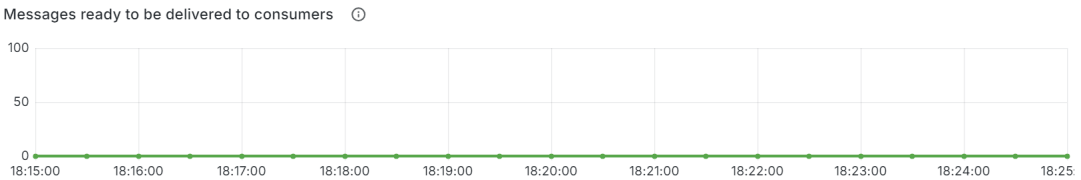


Σχημα 5.77: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Παραδιδόμενα μηνύματα

Αναμενόμενη συμπεριφορά: Το σύστημα πρέπει να διαχειρίζεται την παράδοση των μηνυμάτων αποτελεσματικά, προσαρμοζόμενο στην εκρηκτική κυκλοφορία, διασφαλίζοντας ότι δεν παραμένουν καθυστερημένα μηνύματα προς επεξεργασία. Η ομαλή απόδοση ανταποκρίνεται στην έγκαιρη διαχείριση των μηνυμάτων.

Μήκος ουράς (Μηνύματα έτοιμα προς παράδοση)

Παρατήρηση: Δεν παρατηρείται συσσώρευση μηνυμάτων στην ουρά καθ' όλη τη διάρκεια της δοκιμής. Το σύστημα φαίνεται να επεξεργάζεται τα εισερχόμενα αιτήματα άμεσα, χωρίς να αφήνει τα μηνύματα στην ουρά για εκτεταμένη χρονική περίοδο.

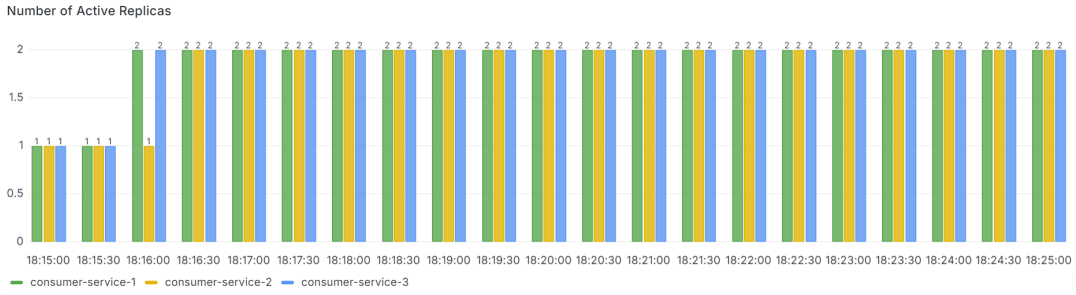


Σχημα 5.78: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Μήκος ουράς

Αναμενόμενη συμπεριφορά: Η ουρά πρέπει να παραμένει κενή ή σχεδόν κενή κατά τη διάρκεια της δοκιμής, καθώς το σύστημα πρέπει να επεξεργάζεται αιτήματα σε πραγματικό χρόνο. Η απουσία συσσώρευσης στην ουρά υποδηλώνει ότι η κλιμάκωση λειτουργεί όπως αναμένεται.

Αριθμός ενεργών αντιγράφων

Παρατήρηση: Ο αριθμός των ενεργών αντιγράφων για κάθε υπηρεσία προσαρμόζεται δυναμικά κατά τη διάρκεια της δοκιμής. Η υπηρεσία-3 εμφανίζει πιο επιθετική κλιμάκωση, φτάνοντας έως και τα 3 αντίγραφα κατά τη διάρκεια των αιχμών, ενώ οι άλλες υπηρεσίες κλιμακώνονται μεταξύ 1 και 2 αντιγράφων.

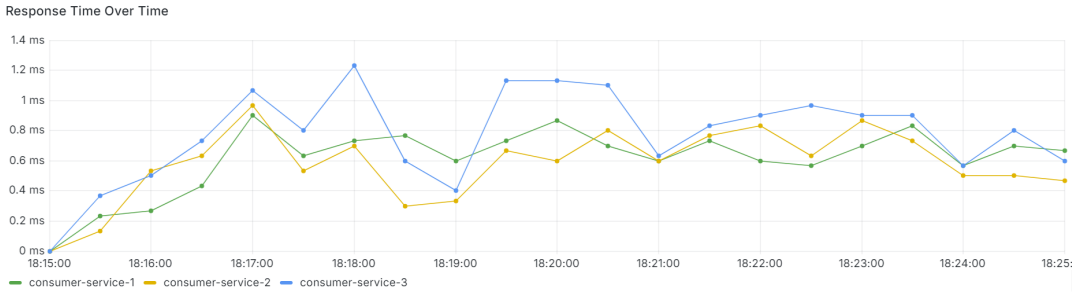


Σχημα 5.79: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Αριθμός ενεργών αντιγράφων

Αναμενόμενη συμπεριφορά: Το σύστημα πρέπει να αυξάνει τον αριθμό των ενεργών αντιγράφων κατά τη διάρκεια των εκρήξεων και να μειώνει την κλίμακα όταν η κυκλοφορία υποχωρεί. Η παρατηρούμενη κλιμάκωση ανταποκρίνεται στις προσδοκίες, με κάθε υπηρεσία να προσαρμόζεται ανάλογα με το φορτίο κίνησης.

Χρόνος απόκρισης

Παρατήρηση: Οι χρόνοι απόκρισης παραμένουν σταθερά χαμηλοί κατά τη διάρκεια της δοκιμής, γενικά κάτω από 1ms, με λίγες μόνο ελαφρές αυξήσεις κατά τη διάρκεια των αιχμών κίνησης. Αυτό δείχνει ότι το σύστημα διαχειρίζεται τις εκρήξεις κυκλοφορίας χωρίς να προκαλεί καθυστερήσεις.

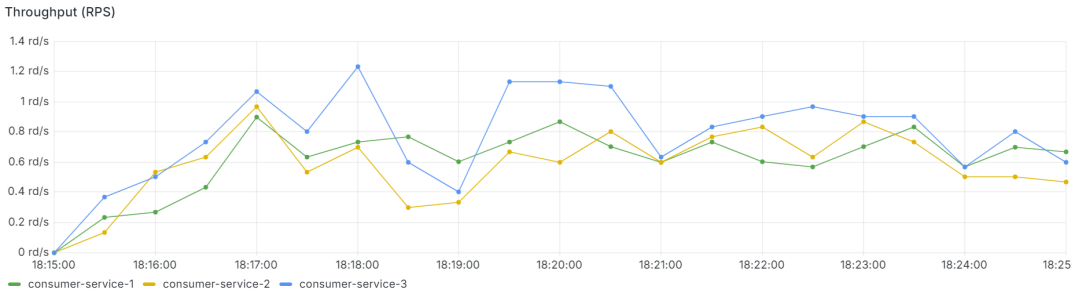


Σχημα 5.80: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Χρόνος απόκρισης

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης πρέπει να παραμένουν χαμηλοί, ακόμα και κατά τη διάρκεια περιόδων υψηλής κυκλοφορίας. Σημαντικές καθυστερήσεις στην απόκριση θα υποδείκνυαν ότι το σύστημα δυσκολεύεται να ανταποκριθεί στο φορτίο. Οι χαμηλοί χρόνοι απόκρισης δείχνουν ότι το σύστημα λειτουργεί αποτελεσματικά.

Απόδοση (αιτήσεις ανά δευτερόλεπτο - RPS)

Παρατήρηση: Η απόδοση ακολουθεί το μοτίβο των εκρήξεων, κορυφώνοντας κατά τη διάρκεια των πιο έντονων εκρήξεων (περίπου 1,2 RPS για την υπηρεσία-3) και μειώνεται μόλις υποχωρήσει η κυκλοφορία.



Σχημα 5.81: Πειράματα Κλιμάκωσης με HPA: Σύντομες εκρήξεις φορτίου υψηλής έντασης: Απόδοση (RPS)

Αναμενόμενη συμπεριφορά: Η απόδοση πρέπει να προσαρμόζεται ανάλογα με την εισερχόμενη κίνηση για να αποφεύγονται οι καθυστερήσεις. Η ομαλή και άμεση προσαρμογή της απόδοσης επιβεβαιώνει ότι το σύστημα διατηρεί αποδοτικούς ρυθμούς επεξεργασίας, χωρίς συμφόρηση κατά τις εκρήξεις.

Σύγκριση με τα αντίστοιχα αποτελέσματα της KPA

Βασικές συγκριτικές παρατηρήσεις:

- **Χρήση CPU**: Στο πλαίσιο του KPA, η χρήση της CPU αυξήθηκε πιο ακανόνιστα, ιδίως για την υπηρεσία 2, όπου παρατηρήθηκαν μεγαλύτερες και λιγότερο ελεγχόμενες διακυμάνσεις. Συγκριτικά, ο HPA είχε ως αποτέλεσμα ομαλότερη χρήση CPU και πιο ελεγχόμενη κλιμάκωση, ειδικά στην υπηρεσία-3, η οποία παρουσίασε καλύτερη διαχείριση των πόρων CPU κατά τη διάρκεια αιφνιδιασμών στην κυκλοφορία.

- **Χρήση μνήμης**: Ο KPA παρουσίασε μεγαλύτερη διακύμανση στην κατανάλωση μνήμης, ιδίως στις υπηρεσίες 1 και 2, όπου παρατηρήθηκαν ξαφνικές αυξήσεις στη χρήση μνήμης. Αντίθετα, ο HPA εμφάνισε πιο σταθερή κατανάλωση μνήμης σε όλες τις υπηρεσίες, γεγονός που υποδεικνύει καλύτερο έλεγχο και κατανομή των πόρων μνήμης κατά τη διάρκεια εκρηκτικών συνθηκών.

- **Messaging (Παράδοση μηνυμάτων)**: Τόσο ο HPA όσο και ο KPA κλιμακώθηκαν αποτελεσματικά όσον αφορά την παράδοση μηνυμάτων, αλλά ο HPA παρείχε πιο ομαλούς ρυθμούς ανταλλαγής μηνυμάτων, με λιγότερες απότομες μεταβολές μεταξύ περιόδων αιχμής και μη αιχμής. Ο KPA έδειξε πιο έντονες διακυμάνσεις, με απότομες αυξήσεις και μειώσεις στους ρυθμούς παράδοσης μηνυμάτων.

- **Βάθος ουράς**: Ο KPA παρουσίασε μικρή συσσώρευση ουρών κατά τη διάρκεια των πιο έντονων εκρήξεων, όπου τα μηνύματα παρέμεναν στην ουρά για λίγο κατά τη διάρκεια του υψηλότερου φόρτου. Αντίθετα, ο HPA κατάφερε να εκκαθαρίσει τις ουρές πιο γρήγορα, διατηρώντας σχεδόν μηδενικό

βάθος ουράς ακόμη και κατά τη διάρκεια των μεγαλύτερων εκρήξεων κυκλοφορίας.

- **Χρόνος απόκρισης**: Ο HPA διατήρησε χαμηλότερους και πιο σταθερούς χρόνους απόκρισης σε σύγκριση με τον KPA. Ο KPA παρουσίασε υψηλότερες αιχμές στους χρόνους απόκρισης κατά τη διάρκεια εκρηκτικών φάσεων κυκλοφορίας, ειδικά για την υπηρεσία 3, όπου παρατηρήθηκαν καθυστερήσεις πάνω από 1ms. Ο HPA επέδειξε καλύτερο έλεγχο στους χρόνους απόκρισης, διατηρώντας χαμηλότερη καθυστέρηση σε όλες τις υπηρεσίες.

Συμπέρασμα: Ο HPA εμφανίζει μεγαλύτερη σταθερότητα στη χρήση των πόρων (CPU και μνήμη) και αποδίδει καλύτερα στη διαχείριση εκρηκτικής κυκλοφορίας χωρίς να επιτρέπει σημαντική συσσώρευση ουρών ή αιχμές στους χρόνους απόκρισης. Αντίθετα, ο KPA, αν και αποτελεσματικός, έδειξε πιο επιθετική συμπεριφορά κλιμάκωσης, οδηγώντας σε μεγαλύτερες διακυμάνσεις των πόρων και ελαφρώς υψηλότερους χρόνους απόκρισης κατά τις εκρήξεις. Συνολικά, για υψηλής έντασης, σύντομες εκρήξεις, ο HPA προσφέρει πιο σταθερή απόδοση και πιο αποδοτική κλιμάκωση, ενώ ο KPA παρουσιάζει πιο αντιδραστική αλλά λιγότερο ελεγχόμενη κατανομή πόρων.

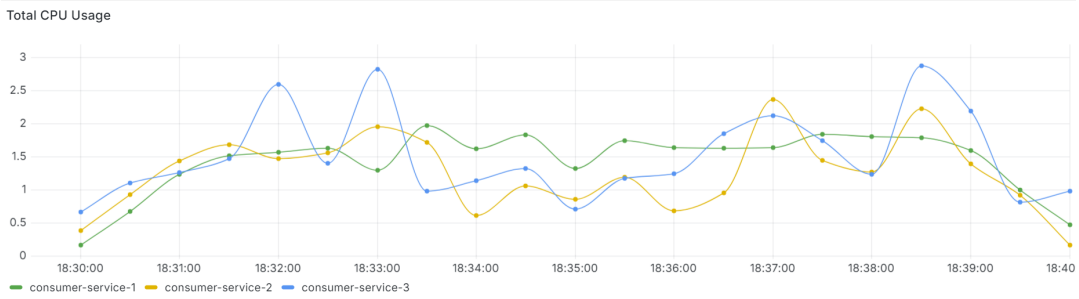
5.6.4 Σταδιακή αύξηση του φορτίου με φάση αδράνειας

Στόχος χρήσης CPU: 70%

Αυτή η υποενότητα εξετάζει τη σταδιακή αύξηση του φορτίου, ακολουθούμενη από μια εκτεταμένη φάση αδράνειας. Ο στόχος χρήσης CPU είναι 70%, επιτρέποντας τη σταθερή κλιμάκωση καθώς αυξάνεται το φορτίο. Το πείραμα δοκιμάζει την ικανότητα του συστήματος να κατανεμήσει ομαλά τους πόρους χωρίς πρόωρη υπερκλιμάκωση και αξιολογεί τη μείωση της κλίμακας κατά τη φάση αδράνειας, παρέχοντας πληροφορίες για τη διαχείριση της σταδιακής κίνησης.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Παρατήρηση: Η συνολική χρήση της CPU αυξάνεται σταθερά καθώς το φορτίο αυξάνεται από 1 σε 5 αιτήσεις ανά δευτερόλεπτο, με κορύφωση κατά την περίοδο υψηλότερου φορτίου. Κάθε υπηρεσία παρουσιάζει παρόμοιες τάσεις, με την υπηρεσία καταναλωτή-3 να καταναλώνει ελαφρώς περισσότερους πόρους CPU σε σύγκριση με τις άλλες δύο υπηρεσίες. Κατά τη φάση αδράνειας, η χρήση της CPU μειώνεται σημαντικά καθώς το φορτίο μειώνεται στο μηδέν.

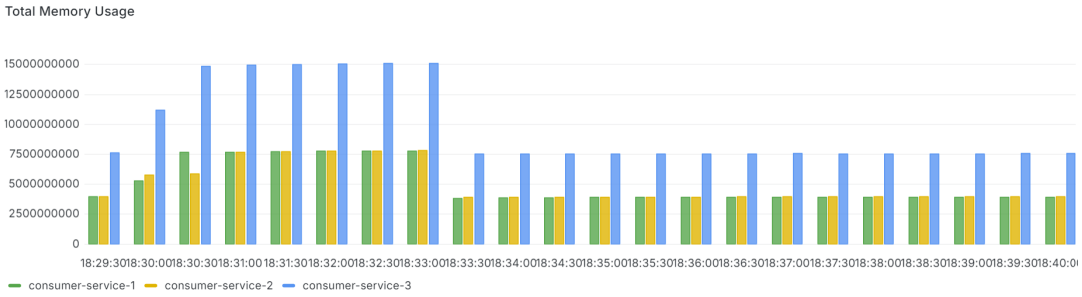


Σχημα 5.82: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Συνολική χρήση CPU

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να κλιμακώνεται αναλογικά με την αύξηση του φορτίου και να μειώνεται κατά τη διάρκεια της φάσης αδράνειας. Το σύστημα λειτούργησε όπως αναμενόταν, με σαφή αύξηση της κατανάλωσης πόρων που αντιστοιχεί στην αύξηση του φορτίου και κατάλληλη πτώση κατά τη διάρκεια της περιόδου αδράνειας.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις καταναλωτικές υπηρεσίες)

Παρατήρηση: Η χρήση μνήμης παρουσιάζει αξιοσημείωτη αύξηση κατά τη φάση φόρτου, με αιχμές που αντιστοιχούν στις περιόδους με την υψηλότερη κυκλοφορία. Η υπηρεσία καταναλωτή-3 ξεχωρίζει με υψηλότερη κατανάλωση μνήμης, ενώ οι υπηρεσίες καταναλωτή-1 και καταναλωτή-2 ακολουθούν ένα πιο μέτριο μοτίβο. Η χρήση μνήμης μειώνεται απότομα κατά τη φάση αδράνειας.

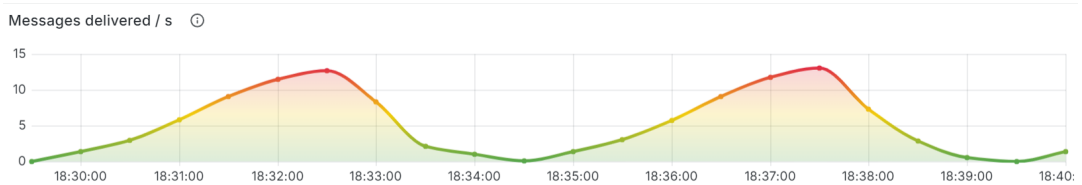


Σχημα 5.83: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Χρήση μνήμης

Αναμενόμενη συμπεριφορά: Η κατανάλωση μνήμης αναμένεται να αυξάνεται κατά τις περιόδους φορτίου και να μειώνεται κατά τη φάση αδράνειας. Η παρατηρούμενη συμπεριφορά ευθυγραμμίζεται με τις προσδοκίες.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (Throughput)

Παρατήρηση: Ο ρυθμός παράδοσης μηνυμάτων αυξάνεται σταθερά, φτάνοντας τα 10-12 μηνύματα ανά δευτερόλεπτο κατά τις περιόδους αιχμής. Κατά τη φάση αδράνειας, ο ρυθμός παράδοσης πέφτει κοντά στο μηδέν.

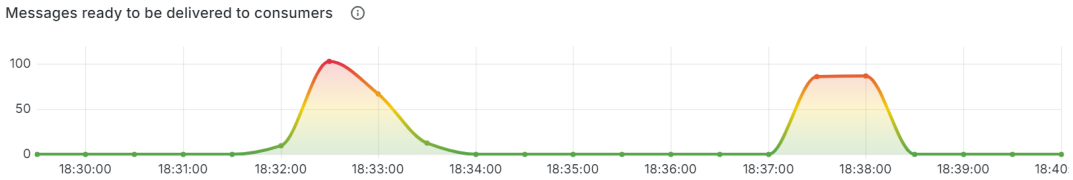


Σχημα 5.84: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Παραδιδόμενα μηνύματα

Αναμενόμενη συμπεριφορά: Η παράδοση των μηνυμάτων πρέπει να κλιμακώνεται με την εισερχόμενη κίνηση και να μειώνεται κατά τη φάση αδράνειας, όπως παρατηρήθηκε.

Μήκος ουράς (Μηνύματα έτοιμα προς παράδοση)

Παρατήρηση: Το βάθος της ουράς αυξάνεται κατά τη διάρκεια των περιόδων φορτίου, με κορύφωση περίπου 100 μηνύματα. Μόλις ο φόρτος μειωθεί, η ουρά καθαρίζεται αποτελεσματικά.

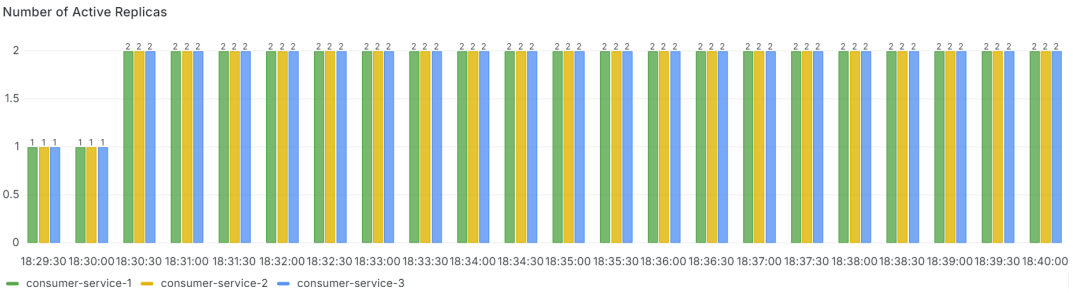


Σχημα 5.85: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Μήκος ουράς

Αναμενόμενη συμπεριφορά: Το βάθος της ουράς θα πρέπει να αυξάνεται κατά την αιχμή και να μειώνεται γρήγορα καθώς τα μηνύματα επεξεργάζονται, όπως παρατηρήθηκε.

Αριθμός ενεργών αντιγράφων

Παρατήρηση: Ο αριθμός των ενεργών αντιγράφων αυξάνεται σε 2 για όλες τις υπηρεσίες κατά τη διάρκεια του φορτίου, με την υπηρεσία 2 να φτάνει προσωρινά τα 3 αντίγραφα. Κατά τη φάση αδράνειας, όλες οι υπηρεσίες επιστρέφουν σε 1 αντίγραφο.



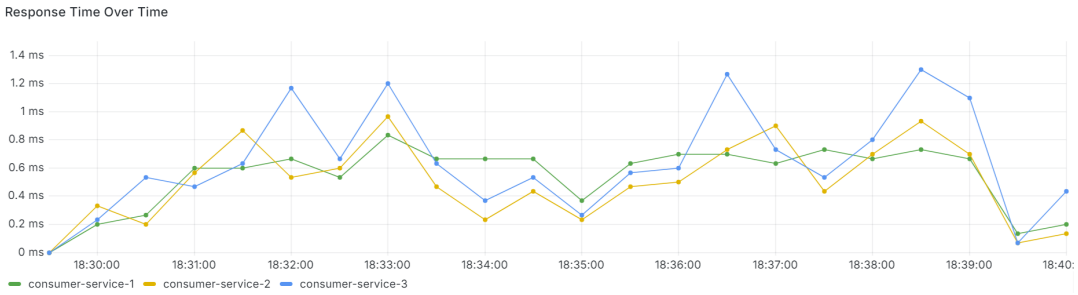
Σχημα 5.86: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Αριθμός ενεργών αντιγράφων

Αναμενόμενη συμπεριφορά: Ο αριθμός των αντιγράφων θα πρέπει να αυξάνεται κατά τις αιχμές και να μειώνεται κατά τη φάση αδράνειας, όπως παρατηρήθηκε.

Χρόνος απόκρισης

Παρατήρηση: Οι χρόνοι απόκρισης παραμένουν χαμηλοί, με μια ελαφρά αύξηση στις περιόδους αιχμής (μέγιστη περίπου 1,2 ms). Κατά τη φάση αδράνειας, οι χρόνοι απόκρισης μειώνονται σε λιγότερο από 0,5 ms.

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να παραμένουν χαμηλοί κατά τη διάρκεια του φορτίου και να μειώνονται περαιτέρω κατά τη φάση αδράνειας, όπως παρατηρήθηκε.



Σχημα 5.87: Πειράματα Κλιμάκωσης με HPA: Σταδιακή αύξηση του φορτίου με φάση αδράνειας: Χρόνος απόκρισης

Σύγκριση με τα αντίστοιχα αποτελέσματα του KPA

Βασικές συγκριτικές παρατηρήσεις:

- **Χρήση CPU**: Σε σύγκριση με τα αποτελέσματα του KPA, το μοτίβο χρήσης της CPU είναι παρόμοιο, αλλά ο KPA εμφανίζει ελαφρώς πιο σταθερές αιχμές σε όλες τις υπηρεσίες. Η κλιμάκωση των πόρων στον KPA φαίνεται να ανταποκρίνεται πιο γρήγορα στις αλλαγές του φορτίου, οδηγώντας σε ταχύτερη προσαρμογή και πιο ομαλές αυξήσεις.

- **Χρήση μνήμης**: Ο KPA εμφανίζει πιο συνεπή κατανομή της μνήμης σε όλες τις υπηρεσίες, με λιγότερες απότομες διακυμάνσεις σε σχέση με τον HPA. Ο KPA κατανέμει τους πόρους μνήμης με πιο ομοιόμορφο τρόπο, επιδεικνύοντας καλύτερη σταθερότητα κατά τη διάρκεια της αυξημένης κίνησης.

- **Αποστολή μηνυμάτων**: Και τα δύο συστήματα, KPA και HPA, χειρίζονται την ανταλλαγή μηνυμάτων αποτελεσματικά. Ωστόσο, ο KPA μειώνει ταχύτερα τον ρυθμό των μηνυμάτων μετά τις αιχμές φορτίου, λόγω της πιο λεπτομερούς προσέγγισης κλιμάκωσης, η οποία αντιδρά γρηγορότερα στη μείωση της κίνησης.

- **Βάθος ουράς**: Ο KPA παρουσιάζει μικρότερης διάρκειας αιχμές στο βάθος ουράς, υποδεικνύοντας ότι επεξεργάζεται τα μηνύματα πιο γρήγορα από τον HPA κατά τη διάρκεια αιχμών φορτίου. Παρόλο που ο HPA διαχειρίζεται καλά την ουρά, χρειάζεται περισσότερο χρόνο για να μειώσει το βάθος ουράς μετά από περιόδους έντονου φόρτου.

- **Χρόνος απόκρισης**: Οι χρόνοι απόκρισης μεταξύ KPA και HPA είναι συγκρίσιμοι, αλλά ο KPA διατηρεί χαμηλότερο μέσο χρόνο απόκρισης, ιδιαίτερα κατά τις φάσεις αύξησης του φορτίου. Η ταχύτερη κλιμάκωση των πόρων στον KPA συμβάλλει στη βελτίωση των χρόνων απόκρισης.

Συμπέρασμα: Ο KPA εμφανίζεται πιο ομαλός και αποτελεσματικός στη διαχείριση τόσο των φάσεων αύξησης όσο και των φάσεων αδράνειας, με ταχύτερη κλιμάκωση πόρων που οδηγεί σε καλύτερη διαχείριση των ουρών και των χρόνων απόκρισης. Οι ταχύτερες ενέργειες κλιμάκωσης του KPA οδηγούν σε καλύτερη βελτιστοποίηση πόρων, ιδίως σε σενάρια με δυναμικές αλλαγές φορτίου.

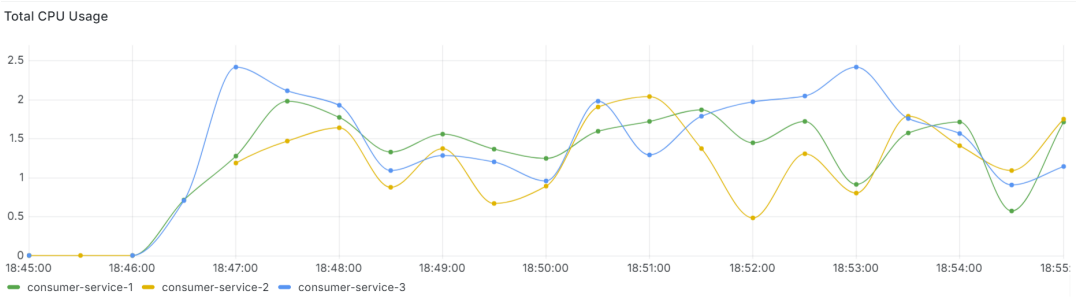
5.6.5 Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας

Στόχος χρήσης CPU: 80%

Σε αυτό το πείραμα, εισάγονται συχνές εκρήξεις κυκλοφορίας, ακολουθούμενες από μικρά διαστήματα αδράνειας. Με στόχο χρήσης CPU στο 80%, το σύστημα πρέπει να ανταποκριθεί γρήγορα στις αυξήσεις του φορτίου, ενώ διατηρεί τη σταθερότητα στις φάσεις αδράνειας. Το πείραμα αξιολογεί την ικανότητα του συστήματος να προσαρμόζεται στις διακυμάνσεις, αποφεύγοντας τη συσσώρευση μηνυμάτων και διασφαλίζοντας την αποδοτική κλιμάκωση.

Χρήση CPU (Συνολική χρήση CPU για τις καταναλωτικές υπηρεσίες)

Παρατήρηση: Η συνολική χρήση της CPU παρουσιάζει σημαντικές διακυμάνσεις κατά τη διάρκεια των περιόδων αιχμής. Η υπηρεσία καταναλωτή-3 παρουσιάζει σταθερά υψηλότερες κορυφές, που φτάνουν έως και 2 CPUs, ενώ οι υπηρεσίες καταναλωτή-1 και 2 έχουν παρόμοιες αλλά χαμηλότερες κορυφές. Το σύστημα κλιμακώνεται αποτελεσματικά για να χειριστεί κάθε έκρηξη και μειώνει τη χρήση της CPU κατά τη διάρκεια των κενών αδράνειας, παρουσιάζοντας απότομες μειώσεις στη χρήση.



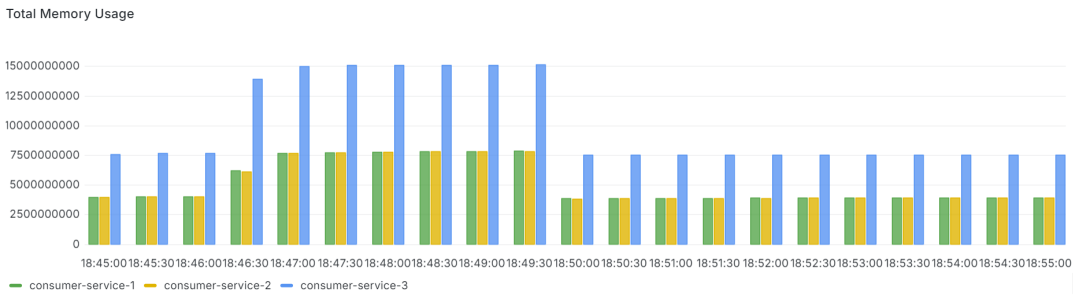
Σχημα 5.88: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Συνολική χρήση CPU

Αναμενόμενη συμπεριφορά: Η χρήση της CPU θα πρέπει να αυξάνεται δυναμικά κατά τη διάρκεια εκρήξεων και να μειώνεται σημαντικά κατά τη διάρκεια κενών αδράνειας. Το σύστημα συμπεριφέρεται όπως αναμενόταν, κλιμακούμενο κατάλληλα με φορτία ριπής και μειώνοντας την κατανάλωση CPU κατά τη διάρκεια περιόδων αδράνειας.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Παρατήρηση: Η χρήση μνήμης αυξάνεται κατά τη διάρκεια των εκρήξεων, ιδίως για την υπηρεσία καταναλωτή-3, η οποία καταναλώνει περισσότερη μνήμη (με μέγιστη τιμή τα 15 GB), ενώ οι υπηρεσίες καταναλωτή-1 και 2 χρησιμοποιούν συγκριτικά λιγότερη μνήμη, αλλά εξακολουθούν να ακολουθούν την ίδια τάση. Η χρήση μνήμης σταθεροποιείται κατά τις περιόδους αδράνειας, αλλά δεν μειώνεται σημαντικά.

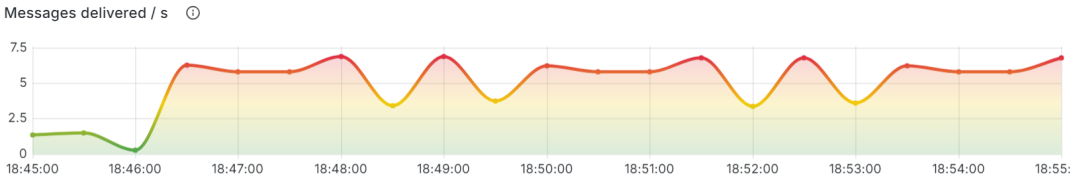
Αναμενόμενη συμπεριφορά: Η χρήση μνήμης θα πρέπει να αυξάνεται κατά τη διάρκεια των εκρήξεων και να παραμένει σταθερή κατά τις φάσεις αδράνειας, γεγονός που συνάδει με την παρατηρούμενη συμπεριφορά.



Σχημα 5.89: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Χρήση μνήμης

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (Throughput)

Παρατήρηση: Η απόδοση μηνυμάτων αυξάνεται γρήγορα κατά τη διάρκεια των εκρήξεων, φτάνοντας περίπου τα 7-8 μηνύματα ανά δευτερόλεπτο, και μειώνεται κατά τις φάσεις αδράνειας. Αυτό το σταθερό μοτίβο καταδεικνύει την ικανότητα του συστήματος να διαχειρίζεται αποτελεσματικά υψηλά φορτία.

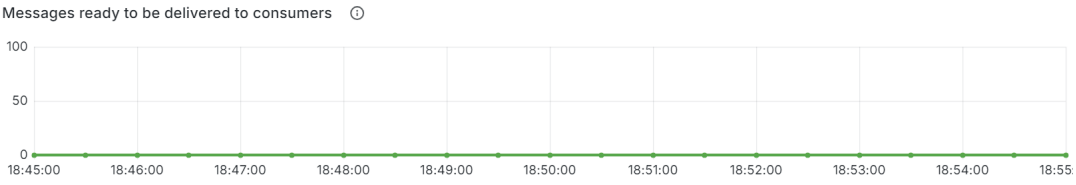


Σχημα 5.90: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Παραδιδόμενα μηνύματα

Αναμενόμενη συμπεριφορά: Η παράδοση μηνυμάτων θα πρέπει να αυξάνεται κατά τη διάρκεια των ριπών και να μειώνεται κατά τις περιόδους αδράνειας. Η απόκριση του συστήματος ανταποκρίνεται στο αναμενόμενο μοτίβο, επεξεργαζόμενο αποτελεσματικά τις εκρήξεις κίνησης.

Μήκος ουράς (Μηνύματα έτοιμα να παραδοθούν στους καταναλωτές)

Παρατήρηση: Το βάθος ουράς παραμένει ελάχιστο καθ' όλη τη διάρκεια της δοκιμής, με μικρές μόνο αυξήσεις προς το τέλος των ριπών. Το σύστημα επεξεργάζεται αποτελεσματικά την εκρηκτική κίνηση χωρίς να αφήνει την ουρά να συσσωρευτεί.

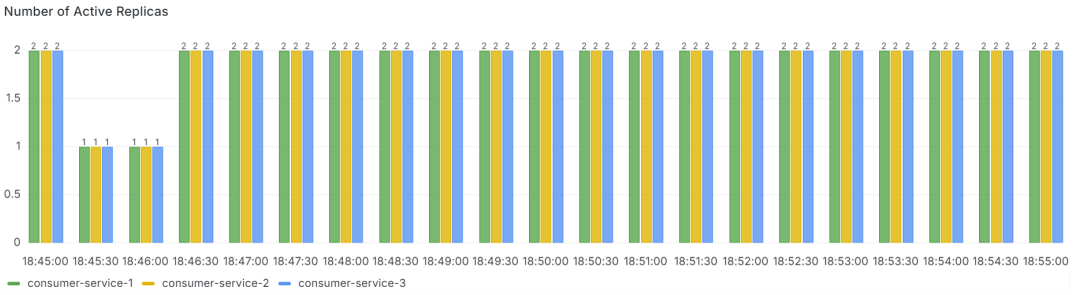


Σχημα 5.91: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Μήκος ουράς

Αναμενόμενη συμπεριφορά: Το βάθος ουράς θα πρέπει να παραμένει χαμηλό εάν το σύστημα κλιμακώνεται σωστά. Η παρατηρούμενη συμπεριφορά είναι σύμφωνη με τις προσδοκίες, υποδεικνύοντας ότι το σύστημα διαχειρίζεται καλά τα φορτία εκρήξεων.

Αριθμός ενεργών αντιγράφων

Παρατήρηση: Ο αριθμός των ενεργών αντιγράφων αυξάνεται ραγδαία κατά τη διάρκεια εκρήξεων. Η υπηρεσία καταναλωτή-1 και η υπηρεσία καταναλωτή-2 κλιμακώνονται σε 2 αντίγραφα, με την υπηρεσία καταναλωτή-2 να φτάνει για λίγο τα 3 αντίγραφα πριν μειωθεί η κλίμακα. Κατά τη διάρκεια των φάσεων αδράνειας, τα αντίγραφα σταθεροποιούνται στο 1.

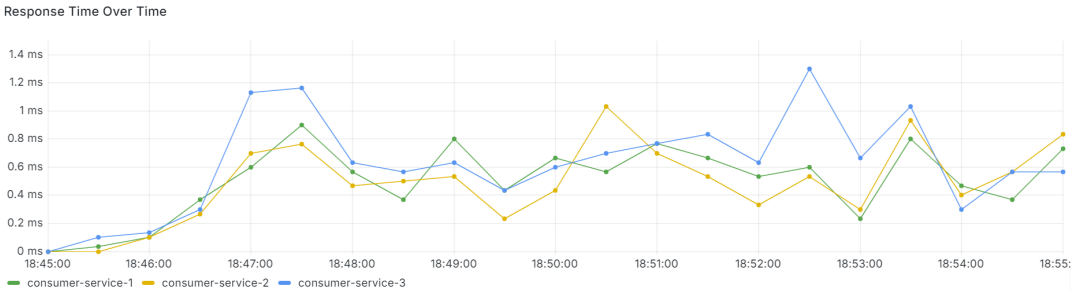


Σχημα 5.92: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Αριθμός ενεργών αντιγράφων

Αναμενόμενη συμπεριφορά: Ο αριθμός των αντιγράφων θα πρέπει να αυξάνεται κατά τη διάρκεια εκρήξεων κίνησης και να μειώνεται κατά τις περιόδους αδράνειας, γεγονός που ταιριάζει με την παρατηρούμενη συμπεριφορά κλιμάκωσης.

Χρόνος απόκρισης

Παρατήρηση: Οι χρόνοι απόκρισης παραμένουν σχετικά χαμηλοί, με μέγιστη τιμή περίπου 1,2 ms κατά τη διάρκεια των εκρήξεων. Οι χρόνοι απόκρισης μειώνονται κάτω από 0,5 ms κατά τη διάρκεια των φάσεων αδράνειας. Όλες οι υπηρεσίες παρουσιάζουν παρόμοιες τάσεις, με την υπηρεσία καταναλωτή-3 να εμφανίζει ελαφρώς υψηλότερες αιχμές.



Σχημα 5.93: Πείραμα κλιμάκωσης με HPA: Συχνές εκρήξεις φορτίου με μικρά κενά αδράνειας: Χρόνος απόκρισης

Αναμενόμενη συμπεριφορά: Οι χρόνοι απόκρισης θα πρέπει να κορυφώνονται κατά τη διάρκεια των ριπών αλλά να παραμένουν συνολικά χαμηλοί, μειούμενοι κατά τις φάσεις αδράνειας. Οι παρατηρούμενοι χρόνοι απόκρισης ανταποκρίνονται στις προσδοκίες αυτές.

Σύγκριση με τα αντίστοιχα αποτελέσματα του KPA

Βασικές συγκριτικές παρατηρήσεις:

- **Χρήση CPU**: Ο HPA εμφανίζει πιο δυναμική κλιμάκωση κατά τη διάρκεια εκρήξεων, ιδιαίτερα για την υπηρεσία καταναλωτή-3, η οποία φτάνει πάνω από 2 CPU. Σε σύγκριση, ο KPA παρουσιάζει πιο σταθερή αλλά ελαφρώς χαμηλότερη χρήση CPU, με πιο ομαλή συμπεριφορά κλιμάκωσης
- **Χρήση μνήμης**: Η χρήση μνήμης στον HPA είναι πιο σταθερή αλλά υψηλότερη για την υπηρεσία καταναλωτή-3, ενώ παραμένει αυξημένη κατά τις φάσεις αδράνειας. Ο KPA παρουσιάζει μικρότερες διακυμάνσεις και πιο ομοιόμορφη κατανομή της μνήμης σε όλες τις υπηρεσίες, με ταχύτερη μείωση κατά τις φάσεις αδράνειας.
- **Αποστολή μηνυμάτων (Throughput)**: Και στα δύο συστήματα, η παράδοση μηνυμάτων φτάνει παρόμοια επίπεδα, με τον HPA να φτάνει στα 7-8 μηνύματα ανά δευτερόλεπτο κατά τις εκρήξεις. Ο KPA, ωστόσο, δείχνει ταχύτερη σταθεροποίηση της απόδοσης κατά τις φάσεις αδράνειας.
- **Βάθος ουράς**: Ο HPA διατηρεί μια σχεδόν άδεια ουρά καθ' όλη τη διάρκεια της δοκιμής, με μικρές αυξήσεις στο τέλος των εκρήξεων. Ο KPA, αν και επεξεργάζεται αποτελεσματικά τα μηνύματα, παρουσιάζει μικρή καθυστέρηση στην εκκαθάριση της ουράς, ειδικά κατά τη διάρκεια αιχμών φορτίου.
- **Χρόνος απόκρισης**: Στον HPA, οι χρόνοι απόκρισης κορυφώνονται γύρω στα 1,2 ms, ενώ ο KPA διατηρεί πιο σταθερούς χρόνους απόκρισης, με αιχμές κοντά στο 1 ms, κάτι που υποδεικνύει ελαφρώς καλύτερη απόδοση.

Συμπεράσματα: Ο HPA χειρίζεται καλά τις συχνές εκρήξεις με μικρά κενά αδράνειας, επιδεικνύοντας αποδοτική κλιμάκωση της CPU και της απόδοσης μηνυμάτων, διατηρώντας παράλληλα χαμηλό βάθος ουράς και χρόνους απόκρισης. Σε σύγκριση με τον KPA, ο HPA είναι ελαφρώς πιο δυναμικός στη χρήση της CPU, αλλά παρουσιάζει μεγαλύτερη μεταβλητότητα στη χρήση της μνήμης και ελαφρώς υψηλότερους χρόνους απόκρισης. Ο KPA, από την άλλη πλευρά, προσφέρει ομαλότερη κλιμάκωση και ταχύτερη μείωση της μνήμης και του βάθους ουράς κατά τις φάσεις αδράνειας, γεγονός που μπορεί να οδηγήσει σε αποδοτικότερη χρήση των πόρων κατά τη διάρκεια κυμαινόμενων συνθηκών κυκλοφορίας. Και τα δύο συστήματα αποδίδουν αποτελεσματικά, αλλά η ομαλότερη κλιμάκωση του KPA ενδέχεται να προσφέρει καλύτερη διαχείριση πόρων σε μεταβαλλόμενες συνθήκες.

5.6.6 Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις

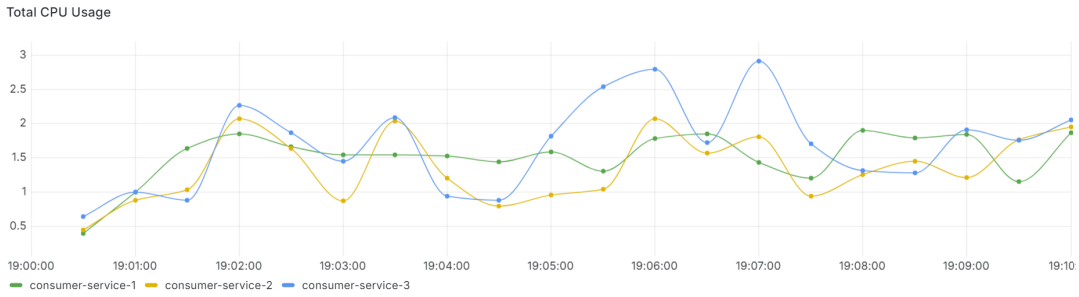
Στόχος χρήσης CPU: 65%

Αυτή η υποενότητα αναλύει την απόδοση του συστήματος υπό εναλλασσόμενο φορτίο, με ελάχιστο χρόνο αδράνειας. Με στόχο χρήσης CPU στο 65%, το σύστημα επιδιώκει την ισορροπία μεταξύ της συνεχούς επεξεργασίας αιτήσεων και της αποδοτικής διαχείρισης πόρων. Το πείραμα δοκιμάζει την ικανότητα του συστήματος να κλιμακώνεται γρήγορα, αποφεύγοντας τη δημιουργία καθυστερήσεων σε συνθήκες σχεδόν συνεχούς λειτουργίας.

Χρήση CPU (Συνολική χρήση CPU για υπηρεσίες καταναλωτών)

Παρατήρηση: Η χρήση της CPU παρουσιάζει σαφείς διακυμάνσεις κατά τη διάρκεια εκρήξεων φορτίου, με κάθε υπηρεσία να παρουσιάζει παρόμοιο μοτίβο.

Η υπηρεσία καταναλωτή-3 τείνει να καταναλώνει ελαφρώς περισσότερη CPU, με αιχμή υψηλότερη από τις άλλες, ενώ η υπηρεσία καταναλωτή-1 έχει πιο σταδιακή αύξηση. Υπάρχει ένα σαφές μοτίβο πτώσης της κατανάλωσης μετά από κάθε έκρηξη, αλλά η ανάκαμψη φαίνεται πιο αργή σε σύγκριση με την KPA.

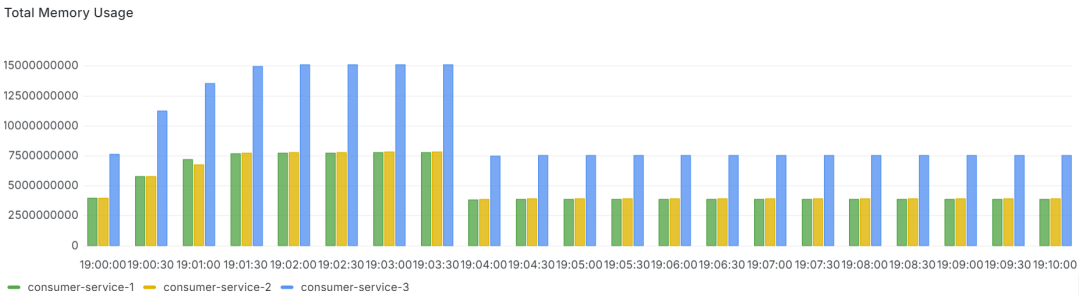


Σχημα 5.94: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Συνολική χρήση CPU

Αναμενόμενη συμπεριφορά: Η κατανάλωση CPU θα πρέπει να αυξάνεται κατά τη διάρκεια των εκρήξεων και να μειώνεται κατά τη φάση αδράνειας. Το σύστημα ακολουθεί γενικά αυτή τη συμπεριφορά, αλλά φαίνεται να χρειάζεται περισσότερο χρόνο για να προσαρμοστεί μετά από κάθε έκρηξη.

Χρήση μνήμης (Συνολική χρήση μνήμης για τις υπηρεσίες καταναλωτών)

Παρατήρηση: Η χρήση μνήμης αυξάνεται κατά τη διάρκεια κάθε ριπής και μειώνεται κάπως στις φάσεις αδράνειας. Η υπηρεσία καταναλωτή-3 καταναλώνει σταθερά περισσότερη μνήμη, ενώ οι άλλες δύο υπηρεσίες είναι πιο ισορροπημένες. Τα μοτίβα μνήμης παρουσιάζουν επίσης απότομες αυξήσεις κατά τις φάσεις έκρηξης, αλλά πιο αργή μείωση σε σύγκριση με την CPU.

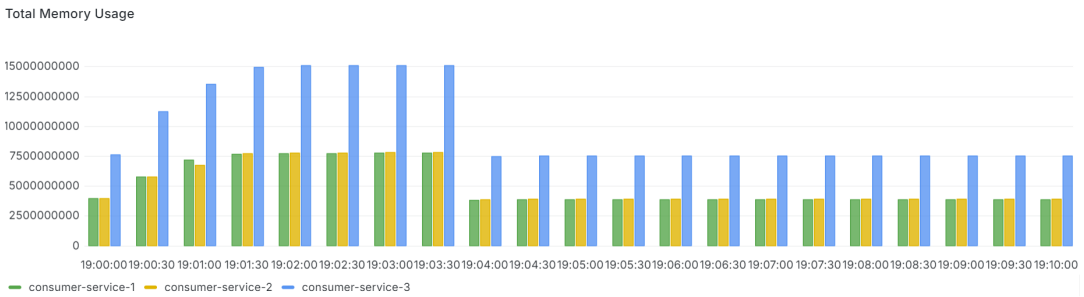


Σχημα 5.95: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Χρήση μνήμης

Αναμενόμενη συμπεριφορά: Η χρήση μνήμης θα πρέπει να αυξάνεται με την αύξηση του φορτίου και να μειώνεται καθώς το φορτίο υποχωρεί. Η διαχείριση της μνήμης εδώ παρουσιάζει πιο αργές αντιδράσεις στις μεταβολές του φορτίου, με την κατανάλωση να παραμένει υψηλότερη για μεγαλύτερο χρονικό διάστημα σε σύγκριση με τη χρήση της CPU.

Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (Throughput)

Παρατήρηση: Η απόδοση φτάνει περίπου τα 7,5 μηνύματα ανά δευτερόλεπτο κατά τη διάρκεια των περιόδων αιχμής, με μικρές πτώσεις κατά τις φάσεις αδράνειας. Το σύστημα φαίνεται να διαχειρίζεται σχετικά καλά την απόδοση μηνυμάτων, με μικρές διακυμάνσεις μεταξύ των ριπών.

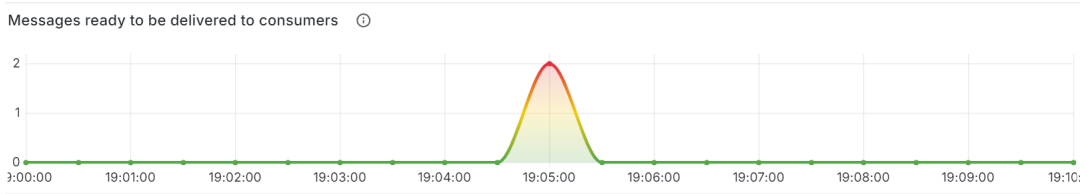


Σχημα 5.96: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Παραδιδόμενα μηνύματα

Αναμενόμενη συμπεριφορά: Η απόδοση θα πρέπει να αυξάνεται αναλογικά με το εισερχόμενο φορτίο και να μειώνεται κατά τη διάρκεια των περιόδων αδράνειας, κάτι που γενικά παρατηρείται.

Μήκος ουράς (μηνύματα έτοιμα προς παράδοση)

Παρατήρηση: Το βάθος της ουράς αυξάνεται κατά τη διάρκεια των εκρήξεων, κορυφώνοντας σε περίπου 40 μηνύματα πριν μειωθεί στο μηδέν. Το σύστημα εκκαθαρίζει την ουρά σχετικά γρήγορα μετά από κάθε έκρηξη, αλλά παρουσιάζει μια ελαφρώς μεγαλύτερη καθυστέρηση κατά την εκκαθάριση.

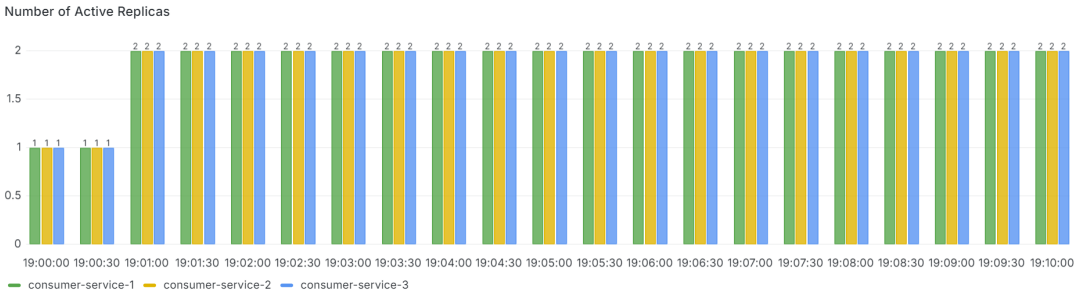


Σχημα 5.97: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Μήκος ουράς

Αναμενόμενη συμπεριφορά: Το βάθος της ουράς θα πρέπει να αυξάνεται κατά τη διάρκεια φάσεων υψηλού φόρτου και να καθαρίζει γρήγορα καθώς το σύστημα κλιμακώνεται για να διαχειριστεί το φορτίο. Το σύστημα καθαρίζει τις ουρές αποτελεσματικά, αλλά η καθυστέρηση είναι μεγαλύτερη από την αναμενόμενη.

Αριθμός ενεργών αντιγράφων

Παρατήρηση: Ο αριθμός των ενεργών αντιγράφων παραμένει ως επί το πλείστον σταθερός στα 2 για όλες τις υπηρεσίες, με περιστασιακές αιχμές για την υπηρεσία καταναλωτή-2, όπου κλιμακώνεται για λίγο σε 3 αντίγραφα πριν μειωθεί ξανά στα 2.

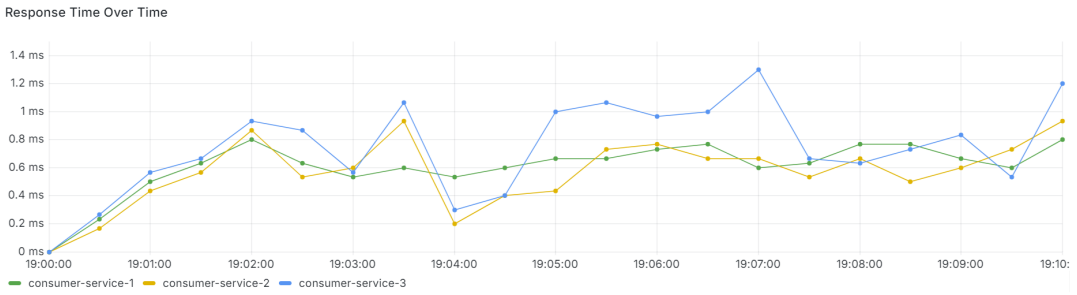


Σχημα 5.98: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Αριθμός ενεργών αντιγράφων

Αναμενόμενη συμπεριφορά: Ο αριθμός των αντιγράφων θα πρέπει να αυξάνεται κατά τις φάσεις αιχμής και να μειώνεται κατά τις περιόδους αδράνειας. Ενώ η συμπεριφορά κλιμάκωσης είναι παρούσα, θα μπορούσε να είναι ταχύτερη και να ανταποκρίνεται καλύτερα στις αλλαγές του φορτίου.

Χρόνος απόκρισης

Παρατήρηση: Οι χρόνοι απόκρισης παρουσιάζουν διακυμάνσεις, κορυφώνονται κατά τη διάρκεια εκρήξεων φορτίου και μειώνονται κατά τις φάσεις αδράνειας. Ο μέγιστος χρόνος απόκρισης είναι περίπου 1,2 ms και η φάση αδράνειας τον φέρνει κάτω από 0,5 ms.



Σχημα 5.99: Πειράματα Κλιμάκωσης με HPA: Εναλλασσόμενο φορτίο με ταχείες μεταβάσεις: Χρόνος απόκρισης

Αναμενόμενη συμπεριφορά: Ο χρόνος απόκρισης θα πρέπει να αυξάνεται κατά τη διάρκεια υψηλής κίνησης και να σταθεροποιείται κατά τη φάση αδράνειας, κάτι που παρατηρείται γενικά.

Σύγκριση με τα αποτελέσματα του KPA

Βασικές συγκριτικές παρατηρήσεις:

- **Χρήση CPU**: Ο KPA παρουσιάζει πιο σταθερές κορυφές σε όλες τις υπηρεσίες, ενώ ο HPA έχει ελαφρώς πιο ακανόνιστες κορυφές. Ο KPA φαίνεται να αντιδρά ταχύτερα στις αλλαγές του φορτίου, παρέχοντας ταχύτερη κλιμάκωση της CPU.
- **Χρήση μνήμης**: Ο KPA επιδεικνύει πιο ισορροπημένη και συνεπή χρήση μνήμης μεταξύ των υπηρεσιών, ενώ ο HPA παρουσιάζει μεγαλύτερες αιχμές

μνήμης, ιδίως για την υπηρεσία καταναλωτή-3. Η διαχείριση της μνήμης του KPA φαίνεται να είναι πιο αποτελεσματική και σταθερή.

- **Παραδιδόμενα μηνύματα ανά δευτερόλεπτο (Throughput)**: Και τα δύο συστήματα χειρίζονται αποτελεσματικά την απόδοση μετάδοσης μηνυμάτων, αν και ο KPA φαίνεται να καθαρίζει τα μηνύματα πιο σταθερά με λιγότερες διακυμάνσεις μεταξύ των εκρήξεων. Ο HPA χειρίζεται καλά την ανταλλαγή μηνυμάτων, αλλά με ελαφρώς μεγαλύτερη διακύμανση στην απόδοση.

- **Βάθος ουράς**: Ο KPA εμφανίζει μικρότερες και συντομότερες αιχμές ουράς, υποδεικνύοντας ταχύτερη επεξεργασία μηνυμάτων. Τα βάθη ουράς του HPA είναι μεγαλύτερα και χρειάζονται περισσότερο χρόνο για να εκκαθαριστούν, γεγονός που υποδηλώνει βραδύτερους χρόνους αντίδρασης σε φορτία εκρήξεων.

- **Αριθμός ενεργών αντιγράφων**: Ο KPA παρουσιάζει ταχύτερη κλιμάκωση των αντιγράφων, προσαρμοζόμενος πιο δυναμικά στις αλλαγές φορτίου. Ο HPA κλιμακώνει τα αντίγραφα, αλλά με ελαφρώς πιο αργές αντιδράσεις, όπως φαίνεται στην περιστασιακή καθυστέρηση στην προσαρμογή του αριθμού των αντιγράφων κατά τη διάρκεια των φάσεων έκρηξης.

- **Χρόνος απόκρισης**: Ο KPA διατηρεί σταθερά χαμηλότερους χρόνους απόκρισης, ιδίως κατά τη διάρκεια περιόδων υψηλού φόρτου, ενώ οι χρόνοι απόκρισης του HPA τείνουν να είναι υψηλότεροι και πιο ποικίλοι, ιδίως κατά τη διάρκεια αιχμής της κυκλοφορίας.

Συμπέρασμα: Συνολικά, ο KPA παρέχει ομαλότερη και ταχύτερη κλιμάκωση, ιδίως όσον αφορά τον χειρισμό δυναμικών αλλαγών φορτίου και τη διατήρηση σταθερών επιδόσεων σε επίπεδο CPU, μνήμης και χρόνων απόκρισης. Ο HPA διαχειρίζεται επαρκώς τις διακυμάνσεις του φορτίου, αλλά επιδεικνύει πιο αργές αντιδράσεις κλιμάκωσης, ιδίως όσον αφορά τη διαχείριση αντιγράφων και την εκκαθάριση βάθους ουράς.

Κεφάλαιο 6

Συμπεράσματα και μελλοντικές κατευθύνσεις

6.1 Σύνοψη του προβλήματος και της λύσης

Στην παρούσα διπλωματική εργασία, η κύρια πρόκληση που αντιμετωπίστηκε ήταν ο δυναμικός χρονοπρογραμματισμός διεργασιών σε περιβάλλοντα Knative, συγκεκριμένα στο πλαίσιο του serverless computing. Οι τεχνολογίες νέφους συνεχώς εξελίσσονται, ιδίως στους τομείς της βελτιστοποίησης πόρων και της υποδομής χωρίς διακομιστή. Αυτό καθιστά όλο και πιο σημαντική την αποτελεσματική διαχείριση των φόρτων εργασίας με παράλληλη διατήρηση της επεκτασιμότητας και της απόδοσης. Το ζήτημα αυτό γίνεται πιο εμφανές σε περιβάλλοντα υπολογισμού άκρων, όπου οι απαιτήσεις επεξεργασίας σε πραγματικό χρόνο πρέπει να ικανοποιούνται με ελάχιστη καθυστέρηση.

Η προτεινόμενη λύση περιελάμβανε τον σχεδιασμό και την υλοποίηση ενός προσαρμοσμένου εξισορροπιστή φορτίου που ενσωματώνεται στο οικοσύστημα Knative. Ο εξισορροπιστής φορτίου χρησιμοποιεί τον αλγόριθμο Additive Increase Multiplicative Decrease (AIMD) για τη δυναμική διαχείριση της κατανομής των εισερχόμενων αιτημάτων σε πολλαπλές υπηρεσίες. Οι αρχές του AIMD -αυξάνοντας σταδιακά την κατανομή των πόρων όταν οι συνθήκες φόρτου είναι ευνοϊκές και μειώνοντας τους πόρους κατά τη διάρκεια της συμφόρησης- αποδείχθηκε ότι είναι μια κατάλληλη προσέγγιση για την εξισορρόπηση των ρυθμών εισδοχής σε ένα δυναμικό περιβάλλον χωρίς διακομιστές.

Ο σχεδιασμός του συστήματος ενσωμάτωσε επίσης μηχανισμούς αυτόματης κλιμάκωσης για την προσαρμογή της χρήσης των πόρων σε απόκριση στους μεταβαλλόμενους φόρτους εργασίας. Διερευνήθηκαν τόσο ο Knative Pod Autoscaler (KPA) όσο και ο Horizontal Pod Autoscaler (HPA), με τον προσαρμοσμένο εξισορροπιστή φορτίου να προσαρμόζει τη συμπεριφορά του με βάση μετρήσεις επιδόσεων όπως το μήκος ουράς και τα ενεργά αντίγραφα των υπηρεσιών-καταναλωτών. Η συγκριτική ανάλυση έδειξε ότι ο Knative Pod Autoscaler (KPA) υπερίσχυσε, καθώς παρείχε πιο άμεση απόκριση στις διακυμάνσεις του φορτίου και φαίνεται να βελτίωσε τη συνολική απόδοση του συστήματος, ειδικά σε περιπτώσεις σύντομων αυξήσεων του φορτίου, όπου ο HPA καθυστερούσε να αντιδράσει επαρκώς.

Η υλοποίηση ενσωματώνει το RabbitMQ για την ανταλλαγή μηνυμάτων και το Redis για τη διαχείριση της κατάστασης. Μέσω αυτών, το σύστημα απέδειξε την ικανότητά του να κλιμακώνεται δυναμικά και να διαχειρίζεται διαφορετικά μοτίβα φορτίου αιτημάτων. Αυτό εξασφάλισε την αποτελεσματική δρομολόγηση των αιτημάτων με την ταυτόχρονη ανταπόκριση των υπηρεσιών-καταναλωτών.

Η προτεινόμενη λύση, η οποία επικυρώθηκε μέσω εκτεταμένων δοκιμών επιδόσεων, ανέδειξε τα πλεονεκτήματα της προσέγγισης εξισορρόπησης φορτίου

με βάση τον αλγόριθμο AIMD. Με την προσαρμογή των ποσοστών αποδοχής σε πραγματικό χρόνο και την αξιοποίηση των χαρακτηριστικών αυτόματης κλιμάκωσης του Knative, το σύστημα μπόρεσε να ανταποκριθεί στις απαιτήσεις των δυναμικών φόρτων εργασίας, συμβάλλοντας έτσι στην αποτελεσματικότερη διαχείριση των πόρων σε περιβάλλοντα υπολογιστών χωρίς διακομιστή και υπολογιστών ακμής.

6.2 Επιτεύγματα και συνεισφορές

Η συνεισφορά της παρούσας διπλωματικής επικεντρώνεται στον τομέα του serverless computing, ειδικά στο πλαίσιο της δυναμικής διαχείρισης πόρων σε περιβάλλοντα Knative. Ένα από τα κύρια επιτεύγματα είναι η επιτυχής υλοποίηση ενός προσαρμοσμένου αλγορίθμου εξισορρόπησης φορτίου που βασίζεται στη μέθοδο Additive Increase Multiplicative Decrease (AIMD). Ο αλγόριθμος επιτρέπει την προσαρμογή σε πραγματικό χρόνο των ποσοστών αποδοχής των εισερχόμενων αιτήσεων, κατανέμοντας αποτελεσματικά το φορτίο σε πολλαπλές υπηρεσίες. Με τη δυναμική διαχείριση της κίνησης, το σύστημα διασφαλίζει τη βέλτιστη αξιοποίηση των πόρων, ιδίως υπό κυμαινόμενο φόρτο εργασίας.

Επιπλέον, η ενσωμάτωση του RabbitMQ ως διαμεσολαβητή μηνυμάτων και του Redis για τη διαχείριση της κατάστασης σε πραγματικό χρόνο αποδείχθηκε βασικό στοιχείο για τη διασφάλιση της επεκτασιμότητας και της απόκρισης του συστήματος. Το RabbitMQ διευκόλυνε τον αποτελεσματικό χειρισμό μηνυμάτων μεταξύ των ελεγκτών αποδοχής και των υπηρεσιών κατανάλωσης, ενώ το Redis επέτρεψε την παρακολούθηση σε πραγματικό χρόνο και τη συλλογή μετρήσεων που τροφοδοτούσαν τις αποφάσεις κλιμάκωσης του εξισορροπιστή φορτίου.

Μια άλλη σημαντική συνεισφορά είναι η λεπτομερής αξιολόγηση των επιδόσεων, η οποία συνέκρινε τη συμπεριφορά του συστήματος υπό διαφορετικές συνθήκες φορτίου με τη χρήση τόσο του Knative Pod Autoscaler (KPA) όσο και του Horizontal Pod Autoscaler (HPA). Τα πειράματα, τα οποία εξέτασαν σταθερούς και εκρηκτικά φορτία εργασίας, έδειξαν ότι το σύστημα μπορούσε να χειριστεί αποτελεσματικά ποικίλα φορτία, διατηρώντας παράλληλα χαμηλή καθυστέρηση και υψηλή απόδοση. Η χρήση του AIMD για τον έλεγχο εισδοχής συνέβαλε επίσης στην εξομάλυνση των αιχμών της κίνησης και στην αποτροπή της συμφόρησης της ουράς, ένας κρίσιμος παράγοντας για τη διασφάλιση σταθερής απόδοσης.

Συνοψίζοντας, η παρούσα διπλωματική εργασία παρέχει μια ολοκληρωμένη λύση για τη δυναμική εξισορρόπηση φορτίου και την κλιμάκωση πόρων σε περιβάλλοντα χωρίς διακομιστή. Μέσω της ανάπτυξης ενός προσαρμοσμένου εξισορροπιστή φορτίου και της ενσωμάτωσης των Knative, RabbitMQ και Redis, εισάγει μια ευέλικτη και κλιμακούμενη αρχιτεκτονική που συμβάλλει στη βελτιστοποίηση της διαχείρισης πόρων χωρίς διακομιστή. Τα επιτεύγματα αυτά προωθούν την κατανόηση του τρόπου με τον οποίο μπορεί να υλοποιηθεί αποτελεσματικά η δυναμική αυτόματη κλιμάκωση σε σύγχρονες cloud-native εφαρμογές.

6.3 Επισκόπηση Αποτελεσμάτων

Το σύστημα αξιολογήθηκε με τη χρήση τόσο του Knative Pod Autoscaler (KPA) όσο και του Horizontal Pod Autoscaler (HPA), προσφέροντας μια λεπτομερή σύγκριση της αποτελεσματικότητάς τους στη διαχείριση δυναμικών φορτίων εργασίας. Με την πειραματική αυτή ανάλυση προέκυψαν χρήσιμες πληροφορίες σχετικά με την απόδοση της προτεινόμενης λύσης υπό διάφορες συνθήκες φορτίου.

Σε σενάρια όπου το σύστημα υποβλήθηκε σε σταθερά χαμηλό φορτίο, ο KPA, σε συνδυασμό με τον μηχανισμό ελέγχου εισδοχής, απέδειξε την ικανότητά του να διατηρεί αποτελεσματικά τη χρήση των πόρων, κλιμακώνοντας τον αριθμό των ενεργών αντιγράφων ώστε να ανταποκρίνεται στη ζήτηση. Αυτό οδήγησε σε ελάχιστες καθυστερήσεις αιτήσεων και σταθερά χαμηλά μήκη ουρών, εξασφαλίζοντας ομαλή και συνεχή εξυπηρέτηση.

Σε συνθήκες αυξημένου και κυμαινόμενου φορτίου, ο KPA συνέχισε να λειτουργεί αξιόπιστα, αν και, να σημειωθεί ότι, παρατηρήθηκαν περιστασιακές καθυστερήσεις στις αποκρίσεις κλιμάκωσης. Η αυτόματη κλιμάκωση του συστήματος αντέδρασε στον υψηλότερο όγκο κίνησης αυξάνοντας τα αντίγραφα, γεγονός που βοήθησε στη διαχείριση των αυξανόμενων ουρών αιτήσεων. Υπήρξαν, βέβαια, περιπτώσεις σύντομης υποβάθμισης της απόδοσης όταν το φορτίο αυξήθηκε απότομα, γεγονός που υποδεικνύει ορισμένους περιορισμούς στον χρόνο απόκρισης του KPA. Παρόλα αυτά, ο εξισορροπιστής φορτίου που βασίζεται στην AIMD μετρίασε τις επιπτώσεις ελέγχοντας τους ρυθμούς εισδοχής, μειώνοντας τον κίνδυνο υπερφόρτωσης του συστήματος.

Σε ό,τι αφορά στον HPA, που βασιζόταν στη χρήση της CPU ως κύρια μετρική κλιμάκωσης, ανταποκρίθηκε καλά σε σενάρια υψηλής χρήσης CPU, αλλά δυσκολεύτηκε με φόρτους εργασίας που παρουσίαζαν πιο ποικίλες απαιτήσεις πόρων. Το σύστημα κλιμακωνόταν περιστασιακά πιο αργά από ό,τι με το KPA, ιδίως σε περιπτώσεις σύντομων εκρήξεων κίνησης. Ωστόσο, το HPA έδειξε καλύτερη ικανότητα να χειρίζεται την αποδοτικότητα των πόρων για παρατεταμένες περιόδους, προσαρμόζοντας τα αντίγραφα σε απόκριση σε συνεχή υψηλά φορτία κυκλοφορίας.

Συνολικά, τα αποτελέσματα επιβεβαίωσαν την αποτελεσματικότητα της στρατηγικής εξισορρόπησης φορτίου. Ενώ και οι δύο αυτόματοι κλιμακωτές επέδειξαν πλεονεκτήματα σε διαφορετικές πτυχές της κλιμάκωσης, οι δυναμικές προσαρμογές του ρυθμού αποδοχής του αλγορίθμου AIMD έπαιξαν καθοριστικό ρόλο στη διασφάλιση ότι το σύστημα παρέμεινε ευέλικτο και αποδοτικό υπό διαφορετικές συνθήκες φόρτου εργασίας. Αυτός ο συνδυασμός εξισορρόπησης φορτίου και αυτόματης κλιμάκωσης επέτρεψε στο σύστημα να προσαρμόζεται ρευστά στις αλλαγές της ζήτησης, βελτιστοποιώντας παράλληλα τη χρήση των πόρων.

Τέλος, αξίζει να σημειωθεί ότι οι δοκιμές πραγματοποιήθηκαν υπό περιορισμένες υπολογιστικές δυνατότητες, λόγω των περιορισμένων πόρων του περιβάλλοντος εικονικής μηχανής. Συνεπώς, υπάρχει εκκρεμότητα για τη διεξαγωγή περαιτέρω δοκιμών σε μεγαλύτερη κλίμακα με περισσότερους πόρους, ώστε να αξιολογηθούν οι πλήρεις δυνατότητες του συστήματος και να επιβεβαιωθούν τα ευρήματα σε συνθήκες μεγαλύτερης φόρτισης.

6.4 Επόμενα Βήματα

6.4.1 Συγκριτική αξιολόγηση αλγορίθμων δρομολόγησης

Ένας από τους τομείς για μελλοντική έρευνα είναι η συγκριτική αξιολόγηση της απόδοσης του αλγορίθμου AIMD σε σύγκριση με άλλους βασικούς αλγορίθμους δρομολόγησης, όπως το Round Robin. Αν και ο προτεινόμενος custom load balancer έχει σχεδιαστεί με βάση τον AIMD αλγόριθμο, ο κώδικας του υποστηρίζει ήδη την ενσωμάτωση άλλων αλγορίθμων δρομολόγησης. Αυτή η δυνατότητα επιτρέπει τη διεξαγωγή πειραμάτων για να συγκριθεί η αποδοτικότητα του AIMD με άλλες παραδοσιακές μεθόδους, υπό διαφορετικές συνθήκες φόρτου εργασίας. Μέσω αυτών των πειραμάτων θα προκύψουν συμπεράσματα σχετικά με τα πλεονεκτήματα και τις αδυναμίες κάθε μεθόδου, εστιάζοντας στην απόκριση του συστήματος, τη χρήση πόρων και τη σταθερότητα κατά τη διαχείριση αυξημένων φορτίων. Εφόσον η υποδομή για την ενσωμάτωση αυτών των αλγορίθμων είναι ήδη έτοιμη, το μόνο που απομένει είναι η συλλογή μετρήσεων για την πλήρη αξιολόγηση των εναλλακτικών μεθόδων δρομολόγησης.

6.4.2 Βελτιστοποίηση των αλγορίθμων κλιμάκωσης

Ένας από τους πιο υποσχόμενους δρόμους για μελλοντική έρευνα είναι η βελτιστοποίηση των αλγορίθμων εξισορρόπησης φορτίου και αυτόματης κλιμάκωσης που υλοποιούνται σε αυτό το σύστημα. Ενώ ο αλγόριθμος ελέγχου αποδοχής με βάση την AIMD αποδείχθηκε αποτελεσματικός στη διαχείριση δυναμικών προτύπων κυκλοφορίας, η περαιτέρω βελτίωση θα μπορούσε να οδηγήσει σε ακόμη πιο αποτελεσματική χρήση των πόρων, ιδίως σε περιβάλλοντα όπου η συμπεριφορά του φόρτου εργασίας είναι εξαιρετικά απρόβλεπτη.

Ενσωμάτωση της αυτόματης κλιμάκωσης με βάση τη μηχανική μάθηση

Εκτός από την τελειοποίηση των υφιστάμενων αλγορίθμων αυτόματης κλιμάκωσης και εξισορρόπησης φορτίου, μια σημαντική κατεύθυνση για μελλοντική εργασία θα ήταν η ενσωμάτωση των μηχανισμών αυτόματης κλιμάκωσης που βασίζονται στη μηχανική μάθηση και προτείνονται από τους Spatharakis et al. στην εργασία τους με τίτλο «Distributed Resource Autoscaling in Kubernetes Edge Clusters»[17]. Αυτή η προσέγγιση θα μπορούσε να ενισχύσει σημαντικά την προσαρμοστικότητα του συστήματος συνδυάζοντας την προληπτική διαχείριση πόρων με την πρόβλεψη του φόρτου εργασίας και τη λήψη αποφάσεων με βάση τη μηχανική μάθηση.

Το σύστημα που προτείνεται στην παρούσα εργασία αξιοποιεί ήδη τεχνικές που βασίζονται στον αλγόριθμο AIMD για τη διαχείριση του ελέγχου εισδοχής και της δυναμικής δρομολόγησης. Η εργασία των Spatharakis et al. επεκτείνει την προσέγγιση AIMD ενσωματώνοντας έναν εκτιμητή φόρτου εργασίας και ένα μοντέλο προφίλ εφαρμογών βασισμένο στη μηχανική μάθηση (ML). Αυτά τα στοιχεία προβλέπουν τη ζήτηση εισερχόμενου φόρτου εργασίας και προσαρμόζουν την κλιμάκωση των πόρων με βάση ιστορικά δεδομένα, βελτιώνοντας την αποδοτικότητα των πόρων και διατηρώντας παράλληλα την ποιότητα υπηρεσιών (QoS).

Το πρώτο βήμα αυτής της ενσωμάτωσης θα περιλαμβάνει την ενσωμάτωση ενός εκτιμητή φόρτου εργασίας παρόμοιου με αυτόν που περιγράφεται από τους Spatharakis et al. Αυτός ο εκτιμητής θα χρησιμοποιεί μοντέλα πρόβλεψης, όπως το ARIMA, για την πρόβλεψη των εισερχόμενων αιτήσεων με βάση προηγούμενα

δεδομένα κίνησης. Στο σημερινό σύστημα, η κλιμάκωση είναι σε μεγάλο βαθμό αντιδραστική, ανταποκρινόμενη στις υπάρχουσες συνθήκες φόρτου.Με την ενσωμάτωση ενός μοντέλου πρόβλεψης φόρτου εργασίας, ο αυτόματος κλιμακωτής θα μπορούσε να κλιμακώσει τους πόρους εν αναμονή της εισερχόμενης κίνησης, οδηγώντας σε αποτελεσματικότερο χειρισμό των εκρηκτικών φόρτων εργασίας, αποφεύγοντας παράλληλα την υποαπασχόληση των πόρων κατά τη διάρκεια περιόδων αδράνειας. Ένα άλλο βασικό στοιχείο της προτεινόμενης μεθόδου είναι το μοντέλο προφίλ εφαρμογών που βασίζεται σε ML, το οποίο ορίζει δυναμικά προφίλ πόρων για διαφορετικούς φόρτους εργασίας και προσαρμόζει ανάλογα τις αποφάσεις κλιμάκωσης.Αυτό το μοντέλο θα μπορούσε να εκπαιδευτεί χρησιμοποιώντας δεδομένα από τις υπάρχουσες δοκιμές και μετρήσεις επιδόσεων, προσφέροντας πιο λεπτομερή έλεγχο της κλιμάκωσης των πόρων.Με την ταξινόμηση διαφορετικών προτύπων φόρτου εργασίας και την αντιστοίχισή τους με προκαθορισμένα προφίλ πόρων, το σύστημα θα μπορούσε να διασφαλίσει ότι κατανέμεται η σωστή ποσότητα πόρων για κάθε κατάσταση φόρτου, αποφεύγοντας τόσο την υπερπροσφορά όσο και τις παραβιάσεις QoS.

Επιπλέον, η ενσωμάτωση ενός Custom Pod Autoscaler (CPA), όπως περιγράφεται στην εργασία των Spatharakis et al., θα μπορούσε να βελτιώσει τη διαδικασία κλιμάκωσης υποστηρίζοντας την κλιμάκωση των πόρων σε μηδενικά αντίγραφα, κάτι που είναι ιδιαίτερα επωφελές σε περιβάλλοντα χωρίς διακομιστές όπως το Knative. Ο CPA θα προσέφερε πιο λεπτομερή έλεγχο της διαδικασίας κλιμάκωσης σε σύγκριση με τον συμβατικό Horizontal Pod Autoscaler (HPA), ελαχιστοποιώντας τη σπατάλη πόρων κατά τις περιόδους αδράνειας και ενισχύοντας την αποδοτικότητα κόστους.

Μετά την ενσωμάτωση αυτών των προηγμένων τεχνικών αυτόματης κλιμάκωσης, θα ήταν απαραίτητο να διεξαχθεί μια συγκριτική αξιολόγηση της απόδοσης του συστήματος. Μετρικές όπως η χρήση CPU και μνήμης, η απόδοση, η καθυστέρηση και οι παραβιάσεις QoS θα πρέπει να χρησιμοποιηθούν για την ανάλυση της αποδοτικότητας των νέων στρατηγικών αυτόματης κλιμάκωσης.Ο στόχος θα ήταν να επικυρωθεί κατά πόσον τα μοντέλα προγνωστικής κλιμάκωσης μειώνουν την κατανάλωση πόρων διατηρώντας ή βελτιώνοντας την ποιότητα των υπηρεσιών που παρέχουν οι εφαρμογές.

Συνοψίζοντας, η μελλοντική έρευνα θα πρέπει να επικεντρωθεί στην ενίσχυση της προσαρμοστικότητας του συστήματος με τη βελτιστοποίηση τόσο της εξισορρόπησης φορτίου όσο και των μηχανισμών αυτόματης κλιμάκωσης. Μέσω της ενσωμάτωσης προηγμένων αλγορίθμων, συμπεριλαμβανομένης της πρόβλεψης φόρτου εργασίας με βάση τη μηχανική μάθηση και της σκιαγράφησης πόρων, το σύστημα θα μπορούσε να επιτύχει ανώτερες επιδόσεις και αποδοτικότητα πόρων. Αυτή η προσέγγιση όχι μόνο θα βελτίωνε την απόκριση του συστήματος, αλλά και θα ευθυγραμμιζόταν με τις σύγχρονες τάσεις στο cloud-native και το edge computing, διασφαλίζοντας ότι το σύστημα παραμένει εξαιρετικά κλιμακούμενο και αποδοτικό υπό διαφορετικές συνθήκες φόρτου εργασίας.

Κεφάλαιο 7

Ευχαριστίες

Φτάνοντας στο τέλος αυτής της διπλωματικής εργασίας, αισθάνομαι την ανάγκη να εκφράσω την ευγνωμοσύνη μου σε όσους στάθηκαν δίπλα μου και με υποστήριξαν σε αυτήν την απαιτητική πορεία.

Πρώτα απ᾽ όλα, θέλω να ευχαριστήσω τον επιβλέποντά μου, κύριο Συμεών Παπαβασιλείου. Τον ευχαριστώ για την εμπιστοσύνη που μου έδειξε από την αρχή αυτής της διαδρομής, τις συμβουλές και τη συνεχή υποστήριξή του. Το κλίμα στο εργαστήριο που διευθύνει αντικατοπτρίζει απόλυτα τον ίδιο, προσφέροντας ένα περιβάλλον συνεργασίας, δημιουργικότητας και επιστημονικής αφοσίωσης.

Επίσης, ένα μεγάλο ευχαριστώ στον Δημήτρη Σπαθαράκη, με τον οποίο είχα τη χαρά να συνεργαστώ καθ᾽ όλη τη διάρκεια της εργασίας. Ήταν πάντα εκεί όταν χρειαζόμουν υποστήριξη, δίνοντας χρήσιμες συμβουλές και λύσεις στα προβλήματα που προέκυπταν, είτε τεχνικά είτε στρατηγικά.

Αισθάνομαι ιδιαίτερη ευγνωμοσύνη για την οικογένειά μου – τον πατέρα μου Χρήστο, τη μητέρα μου Ρίτσα, τις αδερφές μου Ιωάννα και Αναστασία, αλλά και τον σκύλο μας Μίκυ. Η αγάπη και η υπομονή τους ήταν η βάση που με κράτησε δυνατή κατά τη διάρκεια αυτής της διαδικασίας.

Δεν θα μπορούσα να μην ευχαριστήσω και τους φίλους μου – τον Νικόλα, τη Δέσποινα, την Εύα, τις σχεδόν συνονόματες Μαριλένα και Μαριαλένα, και την Ελίνα – που μου έδωσαν τις πολύτιμες στιγμές χαλάρωσης, αλλά και την ενθάρρυνση που χρειαζόμουν. Ήταν οι άνθρωποι που μου θύμιζαν ότι, ακόμα και στις πιο αγχωτικές στιγμές, είναι σημαντικό να βρίσκουμε χρόνο για τους φίλους και τον εαυτό μας.

Η εργασία αυτή αποτέλεσε μια ευκαιρία να καταπιαστώ με έναν δύσκολο αλλά συναρπαστικό τομέα, αυτόν των serverless υποδομών και της εξισορρόπησης φορτίου, και ελπίζω ότι τα αποτελέσματα αντανακλούν την προσπάθεια και την αφοσίωση που της αφιέρωσα.

Βιβλιογραφία

[1] S. Abrishami και M. Naghibzadeh. «Deadline-constrained workflow scheduling in Software as a Service Cloud». Στο: *Scientia Iranica* 19.3 (2012).

[2] AR. Arunarani, D. Manjula και Vijayan Sugumaran. «Task scheduling techniques in cloud computing: A literature survey». Στο: *Future Generation Computer Systems* 91 (2019), σσ. 407–415. DOI: 10.1016/j.future.2018.09.014.

[3] Jacques Chester. *Knative in Action*. Foreword by Ville Aikas. Shelter Island, NY: Manning Publications, 2020.

[4] Dah-Ming Chiu και Raj Jain. «Analysis of the increase and decrease algorithms for congestion avoidance in computer networks». Στο: *Computer Networks and ISDN Systems* 17.1 (1989), σσ. 1–14. ISSN: 0169-7552. DOI: 10.1016/0169-7552(89)90019-6. URL: https://doi.org/10.1016/0169-7552(89)90019-6.

[5] M. Feng κ.ά. «Multi-Objective Particle Swarm Optimization for Resource Allocation in Cloud Computing». Στο: *IEEE* (2012).

[6] W. Juan, L. Fei και C. Aidong. «An improved PSO based task scheduling algorithm for cloud storage system». Στο: *Advances in Information Science and Service Science* 4.18 (2012).

[7] G. Kaur και A. Verma. «An efficient approach to genetic algorithm for task scheduling in cloud computing environment». Στο: *International Journal of Information Technology and Computer Science* 10 (2012).

[8] S. Khan και N. Sharma. «Ant colony optimization for effective load balancing in cloud computing». Στο: *International Journal of Emerging Trends in Technology and Computer Science* 2.6 (2013).

[9] *Knative Documentation*. https://knative.dev/docs/. Knative Project. 2024. URL: https://knative.dev/docs/.

[10] X. Kong κ.ά. «Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction». Στο: *Journal of Network and Computer Applications* 34.4 (2010).

[11] *Kubernetes Documentation*. https://kubernetes.io/docs/. Kubernetes Project. 2024. URL: https://kubernetes.io/docs/.

[12] J. Liu κ.ά. «Job scheduling model for cloud computing based on multi-objective genetic algorithm». Στο: *IJCSI International Journal of Computer Science Issues* 10.1:3 (2013).

[13] Anupama Mampage, Shanika Karunasekera και Rajkumar Buyya. «A Holistic View on Resource Management in Serverless Computing Environments: Taxonomy and Future Directions». Στο: *Journal of Cloud Computing* 20.2 (2024), σσ. 101–115.

[14] L. Pandey κ.ά. «A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments». Στο: *Technical reports, University of Melbourne* (2009).

[15] Ha-Duong Phung και Younghan Kim. «A Prediction based Autoscaling in Serverless Computing». Στο: *Journal of Cloud Computing Research* 15.3 (Αύγ. 2024), σσ. 123–130. URL: https://example.com/article12345.

[16] *RabbitMQ Documentation*. https://www.rabbitmq.com/documentation.html. Pivotal Software, Inc. 2024. URL: https://www.rabbitmq.com/documentation.html.

[17] Dimitrios Spatharakis κ.ά. «Distributed Resource Autoscaling in Kubernetes Edge Clusters». Στο: *2022 18th International Conference on Network and Service Management (CNSM) - Mini Conference* (2022), σσ. 163–169.

[18] U. Srikanth κ.ά. «Tasks scheduling using Ant Colony Optimization». Στο: *Journal of Computer Science* 8.8 (2012), σσ. 1314–1320.

[19] S. Su κ.ά. «Cost-efficient task scheduling for executing large programs in the cloud». Στο: *Parallel Computing* 39 (2013).

[20] M. A. Tawfeek κ.ά. «An ant algorithm for cloud task scheduling». Στο: (2013).

[21] Eleftherios Vlahakis, Nikolaos Athanasopoulos και Seán McLoone. «AIMD scheduling and resource allocation in distributed computing systems». Στο: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, σσ. 4642–4647. DOI: 10.1109/CDC45484.2021.9683321.

[22] W. Wang κ.ά. «Cloud-DLS: dynamic trusted scheduling for cloud computing». Στο: *Expert Systems with Applications* 39 (2012).

[23] Jinfeng Wen κ.ά. «Rise of the Planet of Serverless Computing: A Systematic Review». Στο: *Journal of Cloud Computing* 18.4 (2024), σσ. 203–220.