Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Βελτιστοποίηση εικονικού διαχειριστή πόρων για MPI εφαρμογές σε υπολογιστικά συστήματα

Διπλωματικη Εργασια

του

ΚΟΥΡΤΗ ΑΛΕΞΑΝΔΡΟΥ

Επιβλέπων: Γεώργιος Γκούμας
Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Βελστιστοποίηση εικονικού διαχειριστή πόρων για MPI εφαρμογές σε υπολογιστικά συστήματα

Διπλωματικη Εργασια

του

ΚΟΥΡΤΗ ΑΛΕΞΑΝΔΡΟΥ

Επιβλέπων: Γεώργιος Γκούμας
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Τετάρτη, 26 Ιουλίου 2023.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Γκούμας	Διονύσιος Πνευματικάτος	Νεκτάριος Κοζύρης
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Αλέξανδρος Κούρτης

Μάρτιος 2023

Περίληψη

Οι σύγχρονοι υπερυπολογιστές (high performance computers) εμφανίζονται με τη μορφή συστάδων (clusters) μηχανημάτων. Συνδυάζουν τα επιμέρους resources τους, ώστε να σχηματίσουν μια οντότητα αυξημένων δυνατοτήτων ικανή να επιλύσει σύνθετα προβλήματα. Η αποδοτική τους χρήση στηρίζεται στην προσεκτική δρομολόγηση των εργασιών που υποβάλλονται, ώστε να διασφαλιστεί η βελτίωση της επίδοσης τους, χωρίς να καταναλωθούν αλόγιστα ποσά ενέργειας.

Το σύστημα διαχείρισης πόρων είναι υπεύθυνο για τη χρονοδρομολόγηση των εφαρμογών και την κατανομή των αναγκαίων πόρων σε αυτές. Και οι δύο διαδικασίες γίνονται με κριτήρια καθορισμένα, που αφορούν τον αλγόριθμο χρονοδρομολόγησης και την πολιτική διαμοιρασμού πόρων αντίστοιχα. Στις μέρες μας, υπάρχει ποικιλία εργαλείων διαχείρισης πόρων που μπορούν να ενσωματωθούν σε υπάρχοντα clusters. Η επιλογή τους είναι στην ευχέρεια των ιδιοκτητών των συστημάτων αυτών.

Αν και πολύτιμοι, οι διαχειριστές πόρων είναι δύσκολα τροποποιήσιμοι. Οι παράμετροι λειτουργίας τους μπορούν να αλλάξουν μόνο με παρέμβαση των administrators του συστήματος. Το γεγονός αυτό δυσχεραίνει τους χρήστες που θέλουν να εκτελέσουν τις εφαρμογές τους με τρόπο διαφορετικό από τον προκαθορισμένο.

Για τον λόγο αυτό, έχει φτιαχτεί μια Python εφαρμογή η οποία εκτελεί χρέη ενός εικονικού διαχειριστή πόρων που, ρυθμιζόμενο από το χρήστη, αναθέτει Message Passing Interface (MPI) εργασίες στα μηχανήματα που δεσμεύει. Συγκεκριμένα, η εφαρμογή χρησιμοποιεί μια ουρά εργασιών, που απαρτίζονται από εφαρμογές Nas Parallel Benchmarks (NPB).

Στόχος της παρούσας εργασίας είναι η επέκταση των δυνατοτήτων της ήδη υπαρχουσας Python εφαρμογής, δίνοντάς της περισσότερες επιλογές στην δρομολόγηση εφαρμογών εξειδικευμένες για την κάθε μια από αυτές στην ουρά, όπως επίσης και η επέκταση της συλογής και προβολής των αποτελεσμάτων για μια καλύτερη εποπτεία της απόδοσης της κάθε εφαρμογής.

Λέξεις Κλειδιά

HPC, Συστάδες Υπολογιστών, Διαχειριστής Πόρων, Χρονοδρομολόγηση, Κατανομή Πόρων, Python, MPI, NPB

Abstract

Modern supercomputers (high performance computers) appear in the form of clusters (clusters) of machines. They combine their individual resources to form an enhanced entity capable of solving complex problems. Their efficient use is based on the careful routing of the tasks submitted, in order to ensure the improvement of their performance, without consuming unreasonable amounts of energy.

The resource management system is responsible for scheduling applications and allocating the necessary resources to them. Both processes are done with criteria defined, which concern the scheduling algorithm and the resource allocation policy respectively. Nowadays, there are a variety of resource management tools that can be integrated into existing clusters. Their choice is at the discretion of the owners of these systems.

Although valuable, resource managers are difficult to modify. Their operating parameters can only be changed with the intervention of the system's administrators. This fact makes it difficult for users who want to run their applications in a way other than the default.

For this reason, a Python application has been built which acts as a virtual resource manager that, configurable by the user, delegates Message Passing Interface (MPI) tasks to the machines it binds. Specifically, the application uses a task queue, which consists of Nas Parallel Benchmarks (NPB) applications.

The aim of this work is to extend the capabilities of the already existing Python application, giving it more options in routing applications specialized for each of them in the queue, as well as expanding the collection and display of results for a better performance monitoring of each application.

Keywords

HPC, Computer Clusters, Resource Manager, Scheduling, Resource Allocation, Python, MPI, NPB

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Γεώργιο Γκούμα για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στον τομέα Τεχνολογίας Πληροφορικής και Υπολογιστών (CSLAB). Επίσης ευχαριστώ ιδιαίτερα τον Δρ. Νικόλαο Τριανταφύλλη και τον Αλέξη Παπαβασιλείου για την καθοδήγησή τους και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου και την γάτα μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Μάρτιος 2023

Αλέξανδρος Κούρτης

Περιεχόμενα

Κατάλογος Σχημάτων

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ)	Central Processing Unit (CPU)
Υπερυπολογιστής	HPC
Συστάδα υπολογιστών	Cluster
Διαχειριστής πόρων	Resource Manager

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Οι δυσκολίες που συναντώνται στη μελέτη διαφόρων επιστημών (όπως η βιοϊατρική και η μετεωρολογία), μαζί με την εκρηκτική ανάπτυξη της αρχιτεκτονικής υπολογιστών και του λογισμικού, λειτούργησαν ως καταλύτης για την ανάπτυξη μαζικά παράλληλων συστημάτων πολλαπλής επεξεργασίας, των οποίων η συνολική υπολογιστική ισχύς συνεχίζει να αυξάνεται. Λόγω του μεγέθους τους, αυτά τα συστήματα έχουν σημαντικές απαιτήσεις συντήρησης καθώς και απαιτήσεις κατανάλωσης ενέργειας. Για αυτόν τον λόγο δίνεται μεγάλη προσοχή στον τρόπο χρήσης των πόρων όσο το δυνατόν αποτελεσματικότερα. Οι διαχειριστές πόρων είναι εξειδικευμένες εφαρμογές λογισμικού που χρησιμοποιούνται για την επίβλεψη αυτών.

Ένας διαχειριστής πόρων είναι υπεύθυνος για την κατανομή πόρων και τη δρομολόγηση στις εφαρμογές της ουράς εργασιών. Η πρώτη, η οποία χρησιμοποιεί μια συγκεκριμένη μέθοδο, επιλέγει τη σειρά με την οποία θα διεκπεραιωθούν οι εκκρεμείς εργασίες. Το δεύτερο, το οποίο βασίζεται σε μια ήδη υπάρχουσα πολιτική, συνεπάγεται την κατανομή των μόνων πόρων που απαιτούνται για τα καθήκοντα με τρόπο που να αποτρέπει τον ανταγωνισμό μεταξύ τους. Η διαχείριση πόρων προσφέρει επίσης μια διεπαφή χρήστη που επιτρέπει την υποβολή εργασιών και την παρακολούθηση προόδου. Οι προηγούμενες λειτουργίες έχουν αυστηρές προδιαγραφές που μπορούν να αλλάξουν μόνο από τους διαχειριστές συστήματος.

Ως αποτέλεσμα, είναι δύσκολο για τους ερευνητές που θέλουν να πειραματιστούν με τις καθορισμένες παραμέτρους. Με την ανάπτυξη ενός εικονικού διαχειριστή πόρων (virtual resource manager), η εργασία μας στοχεύει να αντιμετωπίσει αυτό το πρόβλημα. Πρόκειται για ένα πρόγραμμα, που υποβάλλεται ως εργασία στο πραγματικό σύστημα και δημιουργεί ένα σύμπλεγμα μηχανημάτων το οποίο και δεσμεύει. Για όσο διάστημα το σύμπλεγμα αυτό είναι δεσμευμένο, ο χρήστης έχει πλήρη έλεγχο πάνω του. Μετά από μια ανασκόπηση των βασικών στοιχείων αυτού του εργαλείου, πραγματοποιούμε πειραματικές αξιολογήσεις για το πόσο καλά λειτουργεί με τις εφαρμογές MPI.

1.2 Δομή εργασίας

Η εργασία αποτελείται απο συνολικά επτά ενότητες.

- Η **πρώτη ενότητα** περιέχει την εισαγωγή και την δομή της εργασίας.
- Η **δεύτερη ενότητα** ασχολείται σχετικά με τους υπερυπολογιστές και τις βασικές μορφές των μηχανημάτων που τους απαρτίζουν. Επίσης γίνεται αναφορά στο σύστημα MPI.
- Η **τρίτη ενότητα** αναλύει τις λειτουργίες ενός διαχειρηστή πόρων, όπως επίσης και τις διάφορες πολιτικές δρομόγησης εργασιών.
- Η **τέταρτη ενότητα** περιέχει την περιγραφή της ήδη υπάρχουσας εφαρμογής vJRMS.
- Η **πέμπτη ενότητα** αναλύει τις επιπλέον λειτουργίες που έχουν προσθεθεί στα πλαίσια αυτής της διπλωματικής εργασίας στην εφαρμογή vJRMS.
- Η **έκτη ενότητα** ασχολείται σχετικά με τα διάφορα πειράματα που έγιναν μάζι με τα αποτελέσματα αυτών.
- Η **έβδομη ενότητα** υπάρχουν τα τελικά σχόλια όπως επίσης και πιθονές μελλόντικές επεκτάσεις τις εφαρμογής.

Κεφάλαιο 2

HPC

2.1 Εισαγωγή

Οι "Υπερυπολογιστές" είναι υπολογιστικά συστήματα που αξιοποιούνται σε επιστημονικές εφαρμογές οι οποίες απαιτούν την εκτέλεση πολλών εκατομμυρίων μαθηματικών πράξεων ή την επεξεργασία μεγάλου όγκου δεδομένων. Λόγω αυτών των απαιτήσεων τέτοιου είδους προβλήματα είτε θα χρειάζονταν απαγορευτικά μεγάλο χρόνο για να ολοκληρωθούν σε έναν απλό υπολογιστή γραφείου είτε λόγω περιορισμένων πόρων (π.χ. κεντρική μνήμη, αποθηκευτικός χώρος) δεν θα ήταν εφικτό να πραγματοποιηθούν καθόλου. Οι υπερυπολογιστές ξεπερνούν τους περιορισμούς αυτούς χρησιμοποιώντας εξειδικευμένο υλικό τελευταίας τεχνολογίας κάθε χρονική στιγμή, εκμεταλλευόμενοι παράλληλα την υπολογιστική ισχύ από πολλαπλές υπολογιστικές μονάδες. Ένας υπερυπολογιστής σήμερα είναι στην πραγματικότητα ένα σύστημα από εκατοντάδες ή και χιλιάδες υπολογιστές (στους οποίους αναφερόμαστε συνήθως ως "κόμβους) που επικοινωνούν μεταξύ τους χρησιμοποιώντας ένα πολύ γρήγορο δίκτυο και οι οποίοι συνεργατικά μπορούν να επιλύουν προβλήματα με μεγάλη ταχύτητα.[5]

2.2 Πολυεπεξεργαστικά συστήματα

Μηχανήματα με αυτές τις δυνατότητες είναι τα πλέον κατάλληλα εργαλεία για την επίλυση προβλημάτων που απαιτούν πολύ μεγάλη υπολογιστική ισχύ, λόγω της αυξημένης ικανότητας τους να εκτελούν παράλληλη επεξεργασία δεδομένων.Με την πάροδο των χρόνων και την ανάπτυξη της τεχνολογίας, τέτοια συστήματα μπορούν να διαχωριστούν στις εξείς κατηγορίες.[6]

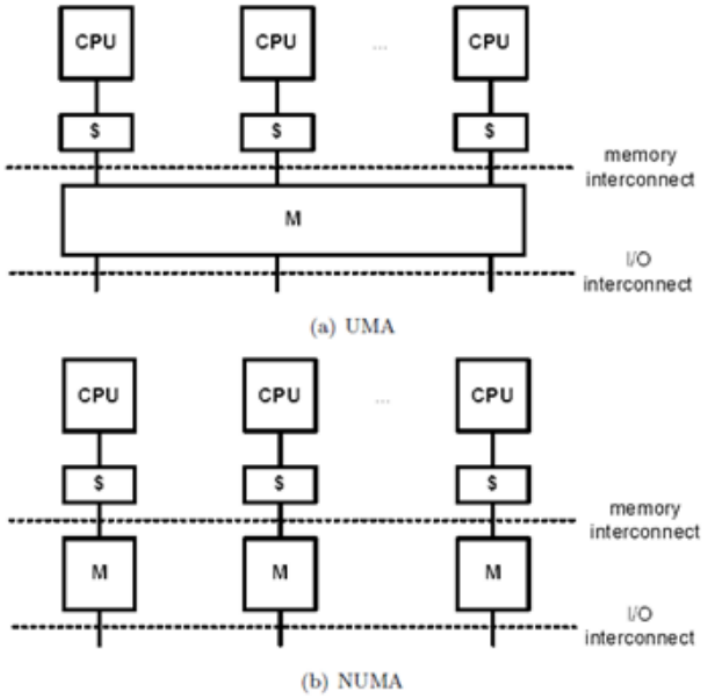
- Συστήματα κοινής μνήμης
- Συστήματα κατανεμημένης μνήμης
- Συστάδες (clusters) υπολογιστών

2.2.1 Συστήματα κοινής μνήμης

Σε ένα σύστημα υπολογιστών υψηλής απόδοσης κοινής μνήμης, πολλοί επεξεργαστές ή πυρήνες μοιράζονται ένα ενιαίο, μεγάλο μπλοκ μνήμης. Αυτό επιτρέπει σε διαφορετικούς

επεξεργαστές να έχουν πρόσβαση και να χειρίζονται δεδομένα χωρίς να απαιτείται ρητή επικοινωνία ή μεταφορά δεδομένων μεταξύ τους. Αυτή η αρχιτεκτονική κοινής μνήμης μπορεί να προσφέρει σημαντικά πλεονεκτήματα απόδοσης για ορισμένους τύπους φόρτου εργασίας HPC που απαιτούν συχνή πρόσβαση σε μεγάλα σύνολα δεδομένων ή κοινόχρηστους πόρους. Ωστόσο, η επεκτασιμότητα των συστημάτων κοινής μνήμης περιορίζεται από τους φυσικούς περιορισμούς της χωρητικότητας μνήμης και του εύρους ζώνης του συστήματος και μπορεί να μην είναι κατάλληλο για εργασίες παράλληλων υπολογιστών πολύ μεγάλης κλίμακας. Για να ξεπεραστούν αυτοί οι περιορισμοί, πολλά συστήματα HPC χρησιμοποιούν έναν συνδυασμό αρχιτεκτονικών κοινής και κατανεμημένης μνήμης.

Ένας πολύ σημαντικός παράγοντας για την απόδοση στα συστήματα αυτά είναι ο χρόνος που απαιτείται για την προσπέλαση μνήμης από κάθε επεξεργαστή. Η απόσταση του κάθε επεξεργαστή από την μνήμη επηρεάζει αυτόν τον χρόνο. Εάν αυτή η απόσταση είναι ίση για κάθε επεξεργαστή, τότε η αρχιτεκτονική καλείται Uniform Memory Architecture (UMA), εάν όχι τότε έχουμε Non Uniform Memory Architecture (NUMA)[1]



Σχήμα 2.1: *Συστήματα κοινής μνήμης[1]*

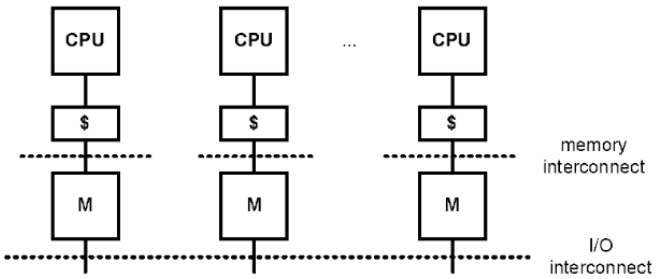
Ένα σύστημα αρχιτεκτονικής UMA καθιστά εύκολη για τον προγραμματιστή την ανάπτυξη για τον λόγο ότι ο χρόνος προσπέλασης είναι ίδιος για κάθε CPU και δεν χρειάζεται να ληφθεί υπόψιν η τοπικότητα των δεδομένων.Βέβαια, το γεγονός οτι μόνο ένας επεξαργαστής μπορεί να εξυπηρετηθεί απο την μνήμη, καθιστά την κλιμάκωση του συστήματος περιορισμένη.

Αντίθετα, ένα NUMA σύστημα είναι ευκολότερα κλιμακώσιμο διότι, λόγω ετεροχρονισμού, υποστηρίζει περισσότερους πυρήνες ταυτόχρονα. Στην περίπτωση αυτή η τοπικότητα των δεδομένων παίζει μεγάλο ρόλο, και χρειάζεται πρόσοχη σε αυτήν για να γίνει καλύτερη εκμετάλευση των μικρότερων χρόνων προσπέλασης και άρα να επιτευχθεί μεγαλύτερη απόδοση.

2.2.2 Συστήματα κατανεμημένης μνήμης

Τα συστήματα υπολογιστών υψηλής απόδοσης κατανεμημένης μνήμης είναι υπερυπολογιστές που αποτελούνται από πολλούς κόμβους ή επεξεργαστές, όπου κάθε κόμβος έχει τη δική του μνήμη και λειτουργεί ανεξάρτητα. Αυτά τα συστήματα έχουν σχεδιαστεί για να επιλύουν μεγάλης κλίμακας επιστημονικά και μηχανολογικά προβλήματα, διασπώντας τα σε μικρότερα κομμάτια που μπορούν να υποβληθούν σε επεξεργασία ταυτόχρονα σε πολλούς κόμβους.

Σε ένα σύστημα κατανεμημένης μνήμης, κάθε κόμβος έχει τη δική του μνήμη και επικοινωνεί με άλλους κόμβους μέσω ενός δικτύου υψηλής ταχύτητας μέσω μηνυμάτων (message passing). Όταν ένα πρόβλημα υποβάλλεται στο σύστημα, χωρίζεται σε μικρότερα υποπροβλήματα που μπορούν να υποβληθούν σε επεξεργασία από διαφορετικούς κόμβους ταυτόχρονα. Κάθε κόμβος υπολογίζει το δικό του τμήμα του προβλήματος και στη συνέχεια τα αποτελέσματα συνδυάζονται για να παράγουν την τελική απάντηση. Αυτή η διαδικασία είναι γνωστή ως παράλληλη επεξεργασία.



Σχήμα 2.2: *Συστήματα κατανεμημένης μνήμης[1]*

Τα συστήματα αυτά χρησιμοποιούνται ευρέως στην επιστημονική έρευνα, την πρόβλεψη καιρού, τις προσομοιώσεις και άλλες υπολογιστικά εντατικές εφαρμογές. Προσφέρουν υψηλή επεκτασιμότητα και απόδοση, καθώς μπορούν να προστεθούν περισσότεροι κόμβοι στο σύμπλεγμα για να αυξηθεί η ισχύς επεξεργασίας του. Ενω η ανάπτυξη εφαρμογών για συστήματα κατανεμημένης μνήμης μπορεί να είναι πιο περίπλοκη από ό,τι για συστήματα κοινής μνήμης, καθώς το πρόγραμμα πρέπει να είναι ρητά σχεδιασμένο για την επικοινωνία δεδομένων μεταξύ κόμβων μέσω του δικτύου, παρακάμπτει τα προβλήματα και τις καθυστερήσεις που σχετίζονται με την επικοινωνία μέσω διαύλου. Λογώ αυτής της υψήλης απόδοσης που προσφέρουν, αυτή η δομή έχει επικτρατήσει στους σύγχρονους πολυεπεξεργαστές.

2.2.3 Συστάδες Υπολογιστών

Μία συστάδα υπολογιστών(cluster) είναι μια ομάδα διασυνδεδεμένων υπολογιστών ή διακομιστών που συνεργάζονται ως ένα ενιαίο σύστημα για να εκτελέσουν μια συγκεκριμένη εργασία ή εφαρμογή. Τα clusters χρησιμοποιούνται συνήθως για εφαρμογές που απαιτούν μεγάλες ποσότητες πόρων υπολογισμού, αποθήκευσης ή μνήμης και που δεν είναι δυνατός ο χειρισμός από έναν μόνο υπολογιστή ή διακομιστή. Χρησιμοποιώντας πολλούς υπολογιστές σε ένα cluster, οι χρήστες μπορούν να αξιοποιήσουν τη συνδυασμένη επεξεργαστική ισχύ, τη μνήμη και την ικανότητα αποθήκευσης για να επιταχύνουν την εκτέλεση των εφαρμογών τους.

Ένας τύπος συμπλέγματος είναι αυτός των υπολογιστών υψηλής απόδοσης (HPC), ο οποίος έχει σχεδιαστεί για να εκτελεί εργασίες εντάσεως υπολογισμού, όπως προσομοιώσεις, μοντελοποίηση και ανάλυση δεδομένων. Τα HPC clusters αποτελούνται συνήθως από πολλούς κόμβους ή διακομιστές, που συνδέονται μέσω ενός δικτύου υψηλής ταχύτητας και διαχειρίζονται από ένα λειτουργικό σύστημα συμπλέγματος. Τα συμπλέγματα αυτά περιλαμβάνουν επίσης εξειδικευμένο λογισμικό, εργαλεία και βιβλιοθήκες που είναι βελτιστοποιημένα για παράλληλους υπολογιστές, όπως MPI (Διασύνδεση μετάδοσης μηνυμάτων), OpenMP (Open Multi-Processing), CUDA (Compute Unified Device Architecture) και BLAS (Basic Linear Algebra Subprograms).

2.3 Σημασία των συστάδων υπολογιστών σήμερα και εφαρμογές

Τα συμπλέγματα υπολογιστών υψηλής απόδοσης έχουν γίνει όλο και πιο δημοφιλή με τα χρόνια για διάφορους λόγους. Μερικοί από τους λόγους που συνέβαλαν σε αυτή την στροφή είναι οι εξής:

- **Αυξημένη ζήτηση για υπολογιστική ισχύ:** Με την ανάπτυξη μεγάλων δεδομένων και πολύπλοκων προσομοιώσεων σε πολλούς τομείς, η ανάγκη για μεγάλης κλίμακας υπολογιστική ισχύ έχει αυξηθεί εκθετικά.
- **Μειωμένο κόστος υλικού:** Με την πάροδο του χρόνου, το κόστος του υλικού έχει μειωθεί σημαντικά, καθιστώντας πιο εφικτή τη δημιουργία και τη συντήρηση συμπλεγμάτων HPC.
- **Βελτιωμένες τεχνολογίες δικτύου και αποθήκευσης:** Οι βελτιώσεις στις τεχνολογίες δικτύου και αποθήκευσης επέτρεψαν την αποτελεσματικότερη επικοινωνία μεταξύ των κόμβων σε ένα σύμπλεγμα και την ταχύτερη πρόσβαση στα δεδομένα.
- **Πρόοδοι στο λογισμικό και στους αλγόριθμους:** Η πρόοδος στο λογισμικό και στους αλγόριθμους κατέστησε δυνατή την παραλληλοποίηση των υπολογισμών και τη διανομή τους σε πολλούς κόμβους σε ένα σύμπλεγμα, με αποτέλεσμα ταχύτερους χρόνους υπολογισμού.
- **Διαθεσιμότητα υπηρεσιών HPC που βασίζονται σε cloud-based εφαρμογές**: Η διαθεσιμότητα υπηρεσιών HPC που βασίζονται σε cloud έχει καταστήσει ευκολότερη και πιο οικονομικά αποδοτική για τους οργανισμούς την πρόσβαση σε υπολογιστικούς πόρους υψηλής απόδοσης χωρίς να χρειάζεται να δημιουργήσουν και να διατηρήσουν τα δικά τους cluster.

Συνολικά, η αυξανόμενη ζήτηση για υπολογιστική ισχύ, σε συνδυασμό με βελτιώσεις στο υλικό, το λογισμικό και τις τεχνολογίες δικτύωσης, έχουν κάνει τα HPC clusters μια όλο και πιο ελκυστική επιλογή για οργανισμούς σε ένα ευρύ φάσμα βιομηχανιών, για παράδειγμα σε :

- **Επιστημονική έρευνα:** Τα HPC clusters χρησιμοποιούνται εκτενώς στην επιστημονική έρευνα, ιδιαίτερα σε τομείς όπως η φυσική, η χημεία, η βιολογία και η αστρονομία.

Αυτά τα συμπλέγματα χρησιμοποιούνται για την προσομοίωση πολύπλοκων φαινομένων, την ανάλυση μεγάλων συνόλων δεδομένων και την εκτέλεση άλλων υπολογιστικών εργασιών.

- **Μηχανική:** Εφαρμογές μηχανικής όπως σχεδιασμός με τη βοήθεια υπολογιστή (CAD), ανάλυση πεπερασμένων στοιχείων (FEA), δυναμική ρευστών (CFD) και άλλες προσομοιώσεις που απαιτούν υπολογιστική ισχύ μεγάλης κλίμακας.
- **Χρηματοοικονομική μοντελοποίηση:** Για την εκτέλεση σύνθετων υπολογισμών και προσομοιώσεων για διαχείριση κινδύνου, βελτιστοποίηση χαρτοφυλακίου και άλλες οικονομικές εφαρμογές.
- **Πρόβλεψη καιρού:** Εκτέλεση προσομοιώσεων και την ανάλυση δεδομένων από δορυφόρους καιρού και άλλες πηγές.
- **Εξερεύνηση πετρελαίου και αερίου:** Προσομοιώσεις δεξαμενών, ανάλυση σεισμικών δεδομένων και άλλες υπολογιστικές εργασίες που σχετίζονται με την εξερεύνηση και την παραγωγή.
- **Μηχανική μάθηση και τεχνητή νοημοσύνη:** Εκπαίδευση μεγάλων νευρωνικών δικτύων, την ανάλυση μεγάλων συνόλων δεδομένων και την εκτέλεση άλλων εργασιών με υπολογιστική ένταση.

2.4 MPI

Το MPI (Message Passing Interface) είναι ένα τυποποιημένο και φορητό σύστημα μετάδοσης μηνυμάτων που χρησιμοποιείται για παράλληλους υπολογιστές. Είναι ένα πρωτόκολλο επικοινωνίας που επιτρέπει σε πολλούς επεξεργαστές να ανταλλάσσουν πληροφορίες και να συντονίζουν τις ενέργειές τους κατά την εκτέλεση ενός προγράμματος σε ένα κατανεμημένο υπολογιστικό περιβάλλον και χρησιμοποιείται συνήθως σε εφαρμογές υπολογιστών υψηλής απόδοσης, όπως επιστημονικές προσομοιώσεις, αριθμητική ανάλυση και επεξεργασία δεδομένων.

Το MPI επιτρέπει σε πολλαπλές διεργασίες που εκτελούνται σε διαφορετικούς κόμβους ενός συμπλέγματος να επικοινωνούν μεταξύ τους, να ανταλλάσσουν δεδομένα και να συγχρονίζουν τις δραστηριότητές τους. Είναι ένα ευρέως διαδεδομένο πρότυπο για παράλληλο προγραμματισμό επειδή είναι φορητό, ευέλικτο και επεκτάσιμο.Παρέχει επίσης ένα πλούσιο σύνολο λειτουργιών επικοινωνίας, συμπεριλαμβανομένης της επικοινωνίας Point-to-point και collective communication. Το πρώτο επιτρέπει στις διεργασίες να στέλνουν και να λαμβάνουν μηνύματα απευθείας με άλλες διεργασίες, ενώ το collective communication επιτρέπει σε ομάδες διεργασιών να συνεργάζονται για την εκτέλεση μιας λειτουργίας υπολογισμού ή δεδομένων. Το MPI υποστηρίζει non-blocking communication, επιτρέποντας στις διαδικασίες να συνεχίσουν να δουλεύουν ενώ περιμένουν την ολοκλήρωση της επικοινωνίας με άλλες διαδικασίες.

Μια διαδικασία MPI είναι μια ακολουθία βημάτων[2] που εκτελούνται από τη βιβλιοθήκη του MPI με σκοπο την δημιουργία και ενεργοποίηση της μεταφοράς ή/και τον συγχρονισμού

δεδομένων. Αποτελείται από τέσσερα στάδια: **αρχικοποίηση**, **εκκίνηση**, **ολοκλήρωση** και **απελευθέρωση**.

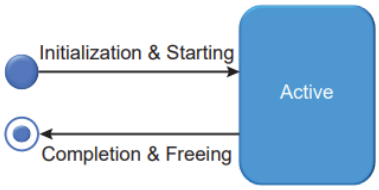
- **Η αρχικοποίηση**, παραδίδει προς εκτέλεση τη λίστα ορισμάτων.
- **Η εκκίνηση**, παραδίδει τον έλεγχο των buffer δεδομένων, εάν υπάρχουν, στη σχετική λειτουργία.
- **Η ολοκλήρωση**, επιστρέφει τον έλεγχο των buffer δεδομένων και το σηματοδοτεί ότι τα buffer εξόδου και τα ορίσματα, εάν υπάρχουν, έχουν ενημερωθεί.
- **Η απελευθέρωση**, επιστρέφει τον έλεγχο της υπόλοιπης λίστας ορισμάτων

Οι διαδικασίες MPI μπορούν να χωριστούν στις εξής κατηγορίες:

- **Blocking operation** Για ένα Blocking operation, και τα τέσσερα στάδια συνδυάζονται σε ένα κλήση.

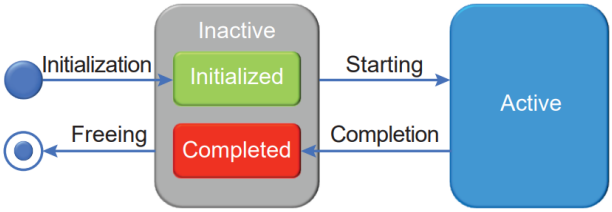
Σχήμα 2.3: *Blocking operation[2]*

- **Nonblocking operation** Για ένα Nonblocking operation, τα στάδια αρχικοποίησης και έναρξης συνδυάζονται σε μια κλήση διαδικασίας χωρίς αποκλεισμό και τα στάδια ολοκλήρωσης και απελευθέρωσης συνδυάζονται σε μια ξεχωριστή, ενιαία κλήση διαδικασίας, η οποία μπορεί είτε να είναι αποκλειστική είτε όχι.



Σχήμα 2.4: *Nonblocking operation[2]*

- **Persistent operation** Για ένα Persistent operation, υπάρχει ξεχωριστή διαδικασία για καθένα από τα τέσσερα στάδια. Κάθε μία από αυτές τις διαδικασίες μπορεί να είναι αποκλειστική ή μη.



Σχήμα 2.5: *Persistent operation[2]*

Κεφάλαιο 3

Resourse Manager

3.1 Εισαγωγή

Ο διαχειριστής πόρων είναι ένα στοιχείο λογισμικού που είναι υπεύθυνο για την κατανομή και τη διαχείριση των πόρων ενός συστήματος υπολογιστή ή ενός δικτύου. Αυτοί οι πόροι μπορεί να περιλαμβάνουν πόρους υλικού όπως CPU, μνήμη και αποθήκευση, καθώς και πόρους λογισμικού όπως άδειες χρήσης, λογαριασμούς χρηστών και δικαιώματα.

Ο πρωταρχικός στόχος ενός διαχειριστή πόρων είναι να διασφαλίσει ότι οι διαθέσιμοι πόροι χρησιμοποιούνται αποτελεσματικά και αποτελεσματικά, ελαχιστοποιώντας τις συγκρούσεις και μεγιστοποιώντας τη συνολική απόδοση του συστήματος. Οι διαχειριστές πόρων χρησιμοποιούνται συνήθως σε υπολογιστικά περιβάλλοντα μεγάλης κλίμακας, όπως κέντρα δεδομένων, υποδομές υπολογιστικού νέφους και δίκτυα επιχειρήσεων.

Παραδείγματα διαχειριστών πόρων περιλαμβάνουν το Resource Manager στο Microsoft Azure, το Apache Mesos cluster Manager και το σύστημα ενορχήστρωσης Kubernetes[7]. Αυτοί οι διαχειριστές πόρων επιτρέπουν στους χρήστες να κατανέμουν και να διαχειρίζονται υπολογιστικούς πόρους με επεκτάσιμο και ευέλικτο τρόπο και να βελτιστοποιούν τη χρήση των πόρων με βάση τις απαιτήσεις φόρτου εργασίας.

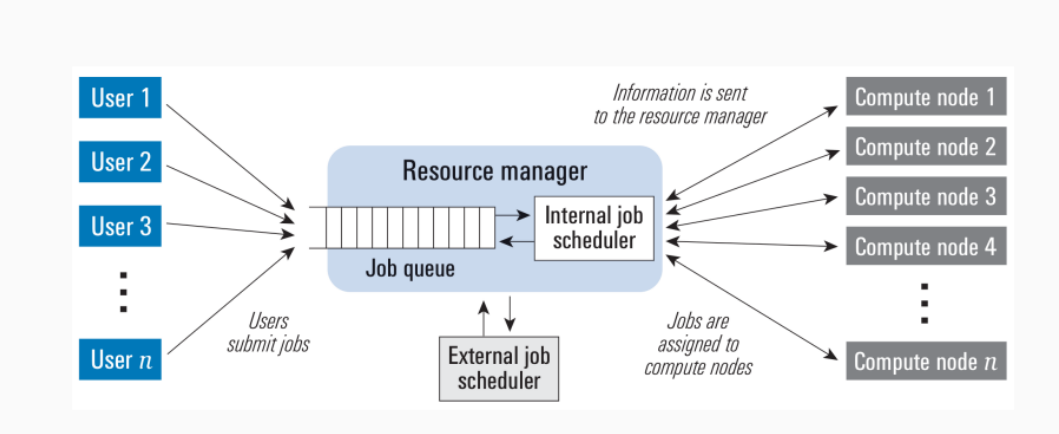
3.2 Εργασίες

Σε έναν διαχειριστή πόρων, μια εργασία αναφέρεται συνήθως σε μια εργασία ή ένα σύνολο εργασιών που υποβάλλονται στο σύστημα για εκτέλεση. Μια εργασία μπορεί να αποτελείται από μία ή περισσότερες διεργασίες ή νήματα που απαιτούν πρόσβαση στους πόρους του συστήματος, όπως η CPU, η μνήμη και η αποθήκευση.

Οι εργασίες μπορεί να ποικίλλουν ως προς την πολυπλοκότητα και τις απαιτήσεις τους και μπορούν να ταξινομηθούν σε διαφορετικούς τύπους με βάση τα χαρακτηριστικά τους. Για παράδειγμα, ορισμένες εργασίες μπορεί να είναι υπολογιστικής έντασης, που σημαίνει ότι απαιτούν σημαντική ποσότητα επεξεργαστικής ισχύος, ενώ άλλες μπορεί να είναι έντασης μνήμης ή έντασης εισόδου/εξόδου.

3.3 Διαχειρηστής πόρων

Όταν ένας χρήστης υποβάλλει μια εργασία σε έναν διαχειριστή πόρων, το σύστημα εκχωρεί τους απαραίτητους πόρους για την εκτέλεση της εργασίας και, στη συνέχεια, προγραμματίζει την εκτέλεση της εργασίας σε έναν ή περισσότερους διαθέσιμους κόμβους υπολογιστών. Ο διαχειριστής πόρων παρακολουθεί επίσης την πρόοδο της εργασίας και διασφαλίζει ότι εκτελείται σύμφωνα με τις καθορισμένες απαιτήσεις του χρήστη, όπως η προθεσμία, η προτεραιότητα και η ποιότητα της υπηρεσίας.



Σχήμα 3.1: *Αναπαράσταση resource manager[3]*

Οι διαχειριστές πόρων συνήθως παρέχουν μια ποικιλία λειτουργιών διαχείρισης και προγραμματισμού εργασιών, όπως υποβολή εργασιών, παρακολούθηση εργασίας, ιεράρχηση προτεραιοτήτων εργασίας, διαχείριση εξάρτησης από την εργασία και λογιστική εργασία. Αυτές οι δυνατότητες επιτρέπουν στους χρήστες να διαχειρίζονται αποτελεσματικά τους υπολογιστικούς τους πόρους και να βελτιστοποιούν την εκτέλεση των εργασιών τους. Μερικές από τις βασικές εργασίες που εκτελεί ένας διαχειριστής πόρων περιλαμβάνουν:

- **Κατανομή πόρων:** Ένας διαχειριστής πόρων εκχωρεί πόρους, όπως CPU, μνήμη και αποθήκευση, σε διαδικασίες και εργασίες με βάση τις απαιτήσεις και τη διαθεσιμότητά τους.

- **Προγραμματισμός:** Προγραμματίζει εργασίες και εργασίες για εκτέλεση σε διαθέσιμους κόμβους υπολογιστών, λαμβάνοντας υπόψη παράγοντες όπως η προτεραιότητα εργασίας, η προθεσμία και η διαθεσιμότητα πόρων.

- **Παρακολούθηση:** Παρακολουθεί την απόδοση και την κατάσταση των υπολογιστικών κόμβων, εργασιών και εργασιών για να διασφαλίσει ότι εκτελούνται σωστά και αποτελεσματικά.

- **Διαχείριση εργασιών:** Παρέχει μια σειρά από λειτουργίες διαχείρισης εργασιών, όπως υποβολή εργασιών, ιεράρχηση εργασιών και διαχείριση εξάρτησης από την εργασία, για να βοηθήσει τους χρήστες να διαχειρίζονται αποτελεσματικά τους υπολογιστικούς τους πόρους.

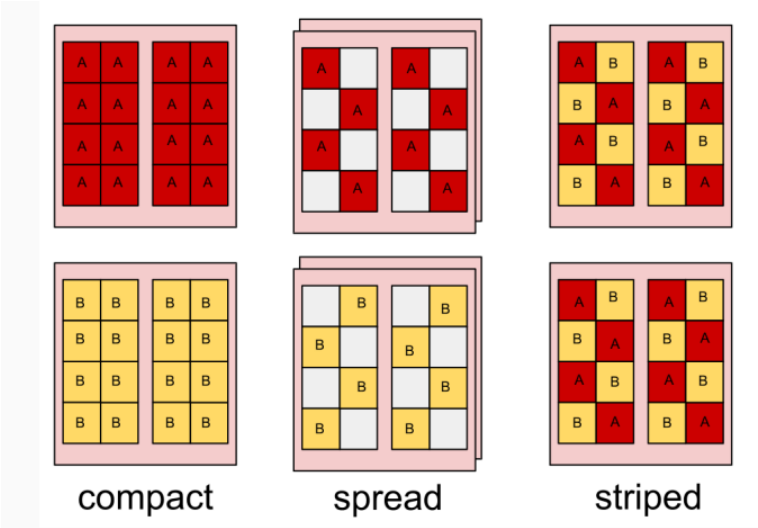
- **Βελτιστοποίηση πόρων:** Βελτιστοποιεί τη χρήση πόρων εξισορροπώντας τον φόρτο εργασίας μεταξύ των κόμβων υπολογιστών και ελαχιστοποιώντας τις συγκρούσεις πόρων.

- **Ασφάλεια:** Μόνο εξουσιοδοτημένοι χρήστες και εφαρμογές έχουν πρόσβαση σε υπολογιστικούς πόρους και ότι η χρήση πόρων ελέγχεται και ελέγχεται.

3.4 Πολιτικές δρομολόγησης εργασιών

Ένα από τα μέσα που διαχειρίζεται ένας resource manager είναι οι επεξεργαστές του κάθε συστήματος.Στα περισσότερα σύγχρονα μηχανήματα, οι επιμέρους πυρήνες ομαδοποιούνται σε σοχκετς.

Οι διαχειριστές πόρων παρέχουν τη δυνατότητα αποκλειστικής ή από κοινού κατανομής πηρύνων σε εφαρμογές, μέσα από τις πολιτικές compact, spare/spread και strip[3]



Σχήμα 3.2: *Κατανομή πυρηνών σε compact, spare/spread και strip [3]*

3.4.1 Compact

Ένας επεξεργαστής με n πυρήνες κατανέμεται σε n εργασίες της ίδιας εφαρμογής. Έτσι το εύρος ζώνης της μνήμης και η LLC μοιράζονται σε n τμήματα.

3.4.2 Spare/Spread

Ένας επεξεργαστής με n πυρήνες κατανέμεται σε n/2 εργασίες της ίδιας εφαρμογής. Σε αντίθεση με την compact πολιτική, σε κάθε δουλειά τώρα αντιστοιχεί διπλάσιο bandwidth και cache space. Σε αυτήν την πολιτική οι μισοί συνολικά πυρήνες μένουν ανενεργοί, έχοντας ως αποτέλεσμα το σύστημα να χρησιμοποιείται στο μισό των δυνατοτήτων του. Λόγω του οτι η βελτίωση της επίδοσης των προγραμμάτων τις περισσότερες φορές δεν αντισταθμίζει την μείωση αυτή, η συγκεκριμένη πολιτική δεν προτιμάται.

3.4.3 Strip

Η πολιτική αυτή συνδιάζει το πλήρη utilization της compact πολιτικής κρατόντας τον μικρότερο ανταγωνισμό μεταξύ των εργασιών της spare. Τώρα, ένας επεξεργαστής με n πυρήνες κατανέμεται ίσα σε εργασίες δύο εφαρμογών (n/2 πυρήνες η κάθε μία από αυτές).Με σώστο ταίριασμα εφαρμογών (π.χ. μια memory intensive εφαρμογή με μία compute intensive εφαρμογή) η πολιτική αυτή εμφανίζει αύξηση της απόδοσης.Αυτός ο τρόπος κατανομής των εφαρμογών ονομάζεται και co-scheduling.

Κεφάλαιο 4

Εικονικός διαχειριστής πόρων vRJMS

4.1 Εισαγωγή

Ο εικονικός διαχειριστής πόρων vRJMS[8] είναι μια εφαρμογή που αναπτύχθηκε από τον Αλέξη Παπαβασιλειου στα πλαίσια της διπλωματικής του εργασίας. Η εφαρμογή αυτή εκτελεί χρέη ενός διαχειριστή πόρων (όπως αυτός περιγράφεται στο κεφάλαιο 3.3) χρησιμοποιώντας μια ουρά εργασιών NPB και υποβάλλοντας τις εργασίες μέσω του Torque Resource Manager[9].

Με την εκκίνησή της, δεσμεύει για κάποιο χρονικό διάστημα συγκεκριμένα μηχανήματα του cluster, τα οποία αφού δεσμευτούν δουλεύουν ανεξάρτητα από τα υπόλοιπα.

Η εφαρμογή αυτή είναι διαθέσιμη στο παρακάτω GitHub repository : https://github.com/alexispapabil/thesis

4.2 Υποσυστήματα

Η εφαρμογή αποτελείται από αρκετές μεθόδους, κάθε μία από αυτές είναι υπεύθυνη για ένα διαφορετικό κομμάτι της εκτέλεσης. Το κυριότερο μέρος της εφαρμογής είναι η κλάση manager και οι μέθοδοι που περιέχονται σε αυτήν. Η κλάση manager είναι υπεύθυνη να κρατάει, εκτός από την ουρά εργασιών, τα χαρακτηριστικά του συστήματος και τους ελεύθερους πόρους του κάθε μηχανήματος. Επίσης διαχειρίζεται τον resource manager, στην περίπτωσή μας το MPI, δίνοντας του εντολές με οδηγίες για την σωστή κατανομή των πόρων. Τέλος, μέσα στην κλάση υπάρχει και η λογική για να μπορέσει να αποφανθεί η εφαρμογή εάν υπάρχουν διαθέσιμοι πόροι για την επόμενη εργασία στην ουρά. Μέσα στην κλάση manager υπάρχουν οι υποκλάσεις node και socket, που περιέχουν πληροφορίες για το κάθε μηχάνημα και τον επεξεργαστή του.

Εκτός από την κλάση manager, η εφαρμογή είναι υπεύθυνη να παράξει ένα καινούριο αρχείο ουράς εάν ένα δεν υπάρχει ήδη, να παράξει τα αποτελέσματα της εκτέλεσης παρέχοντας τον συνολικό χρόνο εκτέλεσης, και πλέον είναι υπευθύνη για την επεξεργασία αυτών τον αποτελεσμάτων για την εμφάνισή τους σε γραφήματα, παρέχοντας τον ακριβή χρόνο εκτέλεσης της κάθε εργασίας.

4.3 Χρήση

Όπως περιγράφεται και στο documentation της εφαρμογής, η σωστή χρήση της μπορεί να χωριστεί σε τρεις κατηγορίες

- **Πριν την εκτέλεση:** Αρχικά πρέπει να υποβληθεί το πρόγραμμα σαν bash script στην κατάλληλη ουρά. Στην περίπτωσή μας, ο διαχειριστής πόρων του συστήματος είναι το Torque, οπότε η υποβολή γίνεται με qsub. Επίσης πρέπει να έχουν ελεγχθεί και αλλάξει ανάλογα τα yaml αρχεία που χρησιμοποιούνται σαν το configuration file της εφαρμογής.

```
#!/bin/bash

#Give the job a descriptive name
#PBS -N rmanager

## Output and error files
#PBS -o demo.out
#PBS -e demo.err

## Limit memory, runtime etc.
#PBS -l walltime=00:20:00

## How many nodes:processors_per_node should we get?
#PBS -l nodes=2:ppn=8

export PATH=/various/common_tools/gcc-4.5.2/bin:$PATH
export LD_LIBRARY_PATH=/various/common_tools/gcc-4.5.2/lib64/

module load openmpi/1.8.3

## Replace with your path
cd /home/users/akourtis/rmanager

python3 main.py -c config/compact.yaml
```

Σχήμα 4.1: *Bash script*

- **Κατά την εκτέλεση:** Παρατηρείται η δημιουργία 2 καινούριων φακέλων, ο φάκελος pids που έχει τις ενεργές εφαρμογές στο cluster σε κάθε στιγμή και ο φάκελος rankfiles, όπου υπάρχουν τα rankfiles για κάθε εφαρμογή που έχει εκτελεστεί.

  Επίσης έχουν δημιουργηθεί και τα αρχεία queue και state, στο πρώτο υπάρχει η ουρά με τις εφαρμογές που είναι για εκτέλεση και στο δεύτερο η κατάσταση του cluster.

- **Μετά την εκτέλεση:** Διαγράφεται ο φάκελος pids και δημιουργούνται 2 νέα αρχεία, τα .err και .out. Το .err περιέχει τυχόν λάθη κατά την εκτέλεση (πχ υπέρβαση χρονικού ορίου) ενώ το .out έχει το output της εφαρμογής.

  Τέλος, κάθε επιμέρους εφαρμογή που έτρεξε έχει το δικό της .err και .out όπου στο .err καταγράφονται τα bindings μεταξύ πυρήνων και MPI processes και στο .out τα αποτελέσματα της εκτέλεσης.

Κεφάλαιο 5

Επέκταση vRJMS

5.1 Προσθήκη επιπλέον λειτουργιών

Στα πλαίσια της συγκεκριμένης διπλωματικής εργασίας, οι λειτουργίες της εφαρμογής vRJMS επεκτάθηκαν. Αυτό έχει ως σκοπό την βελτίωση της απόδοσης της εφαρμογής, προσθέτοντας την επιλογή για πιο πολύπλοκα πειράματα προς εκτέλεση όπως επίσης και μια καλύτερη εποπτεία των αποτελεσμάτων.

5.2 Τεχνικές επεκτάσεις

Σε αυτό το κεφάλαιο θα γίνει αναφορά και ανάλυση των χαρακτηριστικών που έχουν προστεθεί και έχουν να κάνουν σχετικά με την εκτέλεση της εφαρμογής και των επιπλέον χαρακτηριστικών που τώρα υπάρχουν.

5.2.1 Πολλαπλά policies σε μια ουρά

Στην ουρά πλέον εκτός από τα ήδη υπάρχοντα στοιχεία (όνομα, διεργασίες, εκτιμώμενος χρόνος, exclusivity), υπάρχουν αλλά 2 στοιχεία που περιγράφουν την πολιτική που θα τρέξει κάθε εργασία, με το 1ο στοιχείο να είναι η πρώτη επιλογή πολιτικής και το 2ο να είναι η δεύτερη. Τα στοιχειά αυτά προστίθενται με τον ίδιο τρόπο όπως τα υπόλοιπα, μέσω της συνάρτησης generator, όπου διαλέγει με τυχαίο τρόπο εάν θα προσθέσει 0 ή 1, με το 1 να είναι για compact και το 0 να είναι για strip. Η ήδη υπάρχουσα συνάρτηση generator.py που είναι υπεύθυνη για την κατασκευή της ουράς έχει επίσης αλλάξει για να προσθέτει τα επιπλέον πεδία.

```
n0 0 4/4 Full lu.B.x.0 lu.B.x.0 lu.B.x.0 lu.B.x.0
n0 1 4/4 Full lu.B.x.0 lu.B.x.0 lu.B.x.0 lu.B.x.0

n1 0 4/4 Full lu.B.x.0 lu.B.x.0 lu.B.x.0 lu.B.x.0
n1 1 4/4 Full lu.B.x.0 lu.B.x.0 lu.B.x.0 lu.B.x.0

n2 0 4/4 Full bt.B.x.2 bt.B.x.2 bt.B.x.2 bt.B.x.2
n2 1 4/4 Full bt.B.x.2 bt.B.x.2 bt.B.x.2 bt.B.x.2

n3 0 4/4 Full bt.B.x.2 bt.B.x.2 bt.B.x.2 bt.B.x.2
n3 1 4/4 Full bt.B.x.2 bt.B.x.2 bt.B.x.2 bt.B.x.2

n4 0 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3
n4 1 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3

n5 0 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3
n5 1 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3

n6 0 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3
n6 1 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3

n7 0 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3
n7 1 4/4 Full ep.B.x.1 ep.B.x.1 cg.B.x.3 cg.B.x.3
```

Σχήμα 5.1: *Snapshot συστήματος με πολλαπλές πολιτικές*

Για να μπορέσει να γίνει αυτό, χρειάστηκε στην κλάση job να προστεθεί ένα επιπλέον στοιχείο το οποίο ονομάστηκε prefpol, όπου μπαίνει η πολιτική που είναι δοσμένη για την συγκεκριμένη εργασία. Για να εξασφαλιστεί η ορθή εκτέλεση της εφαρμογής και με τον αρχικό τρόπο, η εφαρμογή χρησιμοποιεί την global δοσμένη πολιτική εάν η μεταβλητή prefpol είναι κενη στην συγκεκριμένη εργασία. Αυτό σημαίνει ότι η εφαρμογή τρέχει κανονικά και με την αρχική υλοποίηση άλλα και έχοντας κάποιες εργασίες να έχουν κάποιο δικό τους policy και κάποιες άλλες όχι. Όταν λοιπόν η εφαρμογη αρχίζει την διαδικασία για να αποφανθεί εάν υπάρχει χωρος για μια εργασία, η πολιτική που χρησιμοποιεί είναι εκείνη της συγκεκριμένης εργασίας ,εάν υπαρχει, αλλίως την global πολιτική που εχει δωθει.

Με τον ίδιο τρόπο άλλαξε και το backfilling. Για να μπορέσει να κάνει backfill σε μια εργασία, χρειάζεται να ληφθεί υπόψη και η πολιτική κάθε εργασίας, για αυτό χρειάζεται να γίνει ένας αρχικός έλεγχος για το εάν υπάρχει χώρος για strip και για compact πολιτική, και στην συνέχεια εργασίες που υπάρχει χώρος για την δική τους πολιτική να μπουν προς

εκτέλεση, πχ είναι πιθανό να υπάρχει χώρος για μια compact εργασία αλλά για όχι strip, οπότε η εφαρμογή θα απορρίψει όλες τις εργασίες με δοσμένη πολιτική strip και θα ελένξει αυτές που εχουν compact.

Ένα από τα θεματα που βγήκαν σε αυτήν την υλοποίηση ήταν ότι η εφαρμογή όταν έλεγχε εάν υπάρχει χώρος για μια εργασία, έλεγχε βάση της συγκεκριμένης πολιτικής εκείνης της εργασίας που ήταν να μπει σαν όλες οι εργασίες που τρέχουν ήδη στα clones να έχουν αυτήν την πολιτικη. Αυτό δεν δημιουργούσε θέμα με την αρχική υλοποίηση γιατί όντως όλες οι εργασίες θα χρησιμοποιούσαν μια μονο πολιτική, αλλά τώρα αυτό δεν συνέβαινε. Για παράδειγμα, μια προηγούμενη εργασία με πολιτική compact τρέχει στα μηχανήματα και τα έχει δεσμεύσει όλα. Η επόμενη εργασία που έρχεται από την ουρά είναι πολιτική strip και θέλει κάποιους πόρους. Με την αρχική υλοποίηση αυτό δεν θα γινόταν,να μπει μια strip και μια compact εργασία ταυτόχρονα, αλλά τώρα η εφαρμογή καλείται να το κάνει αυτό, οπότε κοίταγε ότι έχει να βάλει μια strip εργασία, και σε όλα τα clones υπάρχει μόνο μια εργασία. Υποθέτοντας ότι και η δουλεια που ήδη τρέχει είναι ίδιας πολίτικης, δήλαδη strip, και επειδή σε κάθε μηχάνημα τρέχει μια μόνο εργασία, λόγω strip πολιτικής θα χώραγε και μια δεύτερη. Αυτό δημιουργούσε πρόβλημα στην εφαρμογή γιατί ζήταγε πόρους οι οποίοι είχαν ήδη δεσμευτεί. Το θέμα αυτό αντιμετωπίστηκε κοιτώντας, κάθε φορά που είναι να μπεί μια strip πολίτική, σε κάθε επεξεργαστή πόσοι πυρήνες του είναι ελεύθεροι.

5.2.2 Συνάρτηση επιλογής

Πλέον, με τα πολλαπλά policies σε μία ουρά και το γεγονός ότι κάθε εργασία μπορεί να έχει διαφορετική πολιτική από μία άλλη, φτιάχτηκε μια συνάρτηση επιλογής πολιτικής.

Αυτή η συνάρτηση είναι μια lambda function η οποία έχει ως ορίσματα τα χαρακτηριστικά της συγκεκριμένης εργασίας και τα χαρακτηριστικά του συστήματος. Με αυτόν τον τρόπο μπορεί να διαλέξει εάν η εργασία θα χρησιμοποιήσει την πολιτική που έχει σαν 1η επιλογή ή θα χρησιμοποιήσει την πολιτική που είναι σαν δεύτερη επιλογή.

Οι συναρτήσεις αυτές υπάρχουν σε ένα καινούριο αρχείο με όνομα SelectionFunc.py και εκεί μπορεί να υπάρχουν περισσότερες από μια συναρτήσεις επιλογής. Για παράδειγμα, οι συναρτήσεις οι οποίες φτιάχτηκαν και χρησιμοποιήθηκαν κατά κόρον στην ανάπτυξη αυτής της εργασίας ήταν τρείς.

```
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  a = lambda Vars ,Job : Job[4] if Vars[0]*Vars[1]*len(Vars[2])/2 >= int(Job[1]) else 1
5  AllComp = lambda Vars,Jobs : 1
6  AllStrip = lambda Vars,Jobs : 0
```

Σχήμα 5.2: *Συναρτήσεις Επιλογής*

Η πρώτη ελέγχει εάν μια εργασία έχει μπει με πολιτική strip και θέλει να δεσμεύσει παραπάνω πόρους από αυτούς που έχει διαθέσιμο το σύστημα. Εκ παραδρομής υπάρχει πιθανότητα να έχουν δεσμευτεί 8 μηχανήματα με 64 πυρήνες συνολικά, αλλά μια εργασία να έχει πολιτική strip και να θέλει 64 πυρήνες. Σε αυτήν την περίπτωση, λόγω της πολιτικής, θα χρειαζόντουσαν 16 μηχανήματα οδηγώντας την εκτέλεση σε λάθος. Η συγκέκριμένη συνάρτηση επιλογής το αποτρέπει αυτό.

Οι επόμενες δύο, είναι συναρτήσεις που αλλάζουν την πολιτική κάθε εργασίας σε compact ή strip αντίστοιχα. Αύτο έχει ως σκοπό να μπορέσει να τρέξει ένα συγκεκριμένο data set εργασίων με έναν αλλό τρόπο, χωρίς να πρέπει να αλλάξει καθόλου. Εάν δεν υπήρχαν θα έπρεπε να αλλάξουν τα επιπλέον πεδία σχετικά με την πολιτική κάθε εργασίας στο αρχείο της ουράς.

5.2.3 Export module

Όπως περιγράφτηκε στο κεφάλαιο 4.3, κάθε εργασία η οποία έτρεξε στο cluster έχει το δικό της αρχείο εξόδου που έχει διάφορες πληροφορίες σχετικά με την εκτέλεσή της.

Ένα καινούριο αρχείο δημιουργήθηκε με όνομα NPB_module.py που διαβάζει κάθε τέτοιο αρχείο για κάθε εργασία που έτρεξε και παίρνει τους χρόνους εκτέλεσης. Συγκεκριμένα παίρνει τους totcomp και totcomm χρόνους που είναι οι χρόνοι επεξεργασίας (computation) και επικοινωνίας(communication). Το άθροισμά τους είναι ο συνολικός χρόνος που έκανε η εργασία από την αρχή μέχρι το πέρας της.

Το Export module αυτό έχει φτιαχτεί για να δουλεύει με τα nas parallel benchmarks, βέβαια έχει ληφθεί υπόψη η μελλοντική χρήση και άλλων benchmarks, πράγμα που θα αναλυθεί σε επόμενο κεφάλαιο.

5.2.4 CoScheduling

Ένα από τα θέματα που δυσκόλευαν την λήψη ορθών αποτελεσμάτων από την εφαρμογή ήταν η αδυναμία να οριστούν ζευγάρια εργασίων που θα έτρεχαν μαζί, για να μπορέσει ο χρήστης να έχει αποτελέσματα για την συμπεριφορά κάθε εργασίας όταν έτρεχε με κάποια άλλη.

Δημιουργήθηκε λοιπόν μία επιπλέον λειτουργία στο αρχείο ουράς, που εάν σε μία γραμμή υπάρχουν δύο εργασίες, αυτές είναι σίγουρο ότι θα τρέξουν μαζί. Αξίζει να σημειωθεί ότι εάν οι εργασίες δεν είναι σε πολιτική strip ή οι πυρήνες που δεσμεύουν δεν είναι ίδιοι σε αριθμό, τότε οι εργασίες θα τρέξουν χώρις κάποια αλλαγή, σαν να ήταν η μία κάτω από την άλλη.

Για να γίνει αυτό και να εγγυηθεί η εφαρμογή ότι δύο εργασίες θα τρέξουν μαζί, οτάν καλείται να βρει χώρο για την πρώτη, τότε ψάχνει διαθέσιμο χώρο σαν εκείνη να ήταν σε πολιτική spare(δηλ. να τρέχει με τα χαρακτηριστικά της strip αλλά οι υπόλοιποι πυρήνες των συγκεκριμένων μηχανιμάτων να είναι ελεύθεροι). Αφού βρει των χώρο, δεσμεύει τους μισούς πόρους στην πρώτη εργασία και τους υπόλοιπους μισούς στην δεύτερη.

Η εικόνα 5.1 έχει για παράδειγμα τις εργασίες ep και cg να τρέχουν με CoScheduling.

```
1  lu.B.x 16 60 0 1 1
2  bt.B.x 16 90 0 1 0
3  ep.B.x 16 20 0 0 0 cg.B.x 16 150 0 0 0
```

Σχήμα 5.3: *Παράδειγμα ουράς με CoScheduling*

5.2.5 Η δομή JobStats

Οι επιπλέον πληροφορίες που κρατάει η εφαρμόγη οδήγησαν στην ανάγκη να δημιουργηθεί μια ακόμα δομή για την αποθήκευση των δεδομένων.

```
4   class JobStats:
5       def __init__(self,id,name,procs,confpol,actualpol):
6           self.id = id
7           self.name = name
8           self.Class = name[3]
9           self.procs= procs
10          self.ConfPolicy = confpol
11          self.ActualPolicy = actualpol
12          self.RunOn = ''
13          self.RunWith = ''
14          self.time = ''
```

Σχήμα 5.4: *Η δομή JobStats*

Η δομή JobStats εκτός από τα βασικά χαρακτηριστικά μίας εργασίας(όνομα, id, κλάση, πυρήνες) περίεχει και τις επιπλέον πληροφορίες από τις λειτουργίες που έχουν προστεθεί.

- **Configured policy:** Αυτή είναι η πολιτική που έχει γραφτεί στην ουρά για να τρέξει η εργασία.
- **Actual policy:** Η πολιτική που όντως έτρεξε η εργασία. Αυτή μπορεί να είναι διαφορετική από την πρώτη εάν η συνάρτηση επιλογής έχει διαλέξει άλλη πολιτική από αυτήν που ήταν configured να τρέξει.
- **Μηχανήματα που χρησιμοποίησε:** Αποθηκεύονται τα μηχανήματα που δεσμεύτηκαν για την εργασία.
- **Εργασία/Εργασίες που έτρεξαν μαζί:** Αποθηκεύονται οι εργασίες που έτρεξαν μαζί με την δεδομένη εργασία. Σε μία compact ή spare πολιτική αυτό το πεδίο θα έχει την τιμή 'Run Alone' ένω σε μία strip πολιτική θα έχει τις εργασίες (όνομα, κλάση, ID) που έτρεξαν μαζί.
- **Χρόνος:** Αυτός είναι ο χρόνος που γίνεται export από το Export module του υποκεφαλαίου 5.2.3. Σε αυτό το σημείο, ο χρόνος αυτός είναι το άθροισμα των δύο χρόνων(επεξεργασίας και επικοινωνίας) για να μπορεί να φανεί καλύτερα ο χρόνος που η συγκεκριμένη εργασία δέσμευσε πόρους.

5.2.6 Αλλαγές σε αρχεία εισόδου και εξόδου

Στα αρχεία εισόδου, η ουρά, το script και το yaml configuration αρχείο έχουν αλλάξει. Η ουρά έχει δύο επιπλέον πεδία σε κάθε εργασία για την πολιτική της και υπάρχει και η δυνατότητα να υπάρχουν δύο εργασίες στην ίδια σειρά για την περίπτωση του CoScheduling.

Στο yaml configuration αρχείο υπάρχουν νέα πεδία σχετικά με την συνάρτηση επιλογής, το Export module και το SpeedUp των εργασίων. Στο πρώτο δίνεται το αρχείο, το όνομα της συνάρτησης επιλογής και τα χαρακτηριστικά του συστήματος που θα λάβει υπόψη. Στο δεύτερο χρειάζεται μόνο το όνομα του φακέλου και το όνομα του module. Εάν η εφαρμογή έχει τρέξει, και τα rankfiles και η ουρά δεν έχουν πειραχτεί, αλλάζοντας την μεταβλητη

AlreadyRun σε True, η εφαρμογή τρέχει χωρίς να εκτελέσει καμία εργασία, παράγοντας τα αρχεία εξόδου από τα ήδη υπάρχοντα rankfiles. Τέλος, το καινούριο πεδίο σχετικά με το SpeedUp περιέχει την τοποθεσία και το αρχείο με τους χρόνους των διεργασιών που καταγράφτηκαν στο συγκεκριμένο σύστημα που έγιναν τα πειράματα. Το SpeedUp θα αναλυθεί σε περισσότερο βάθος σε επόμενο κεφάλαιο.

```yaml
Version: 1.0

Applications:
  Path: /home/users/akourtis/test/NPB3.4/NPB3.4-MPI/bin
  Queue: /home/users/akourtis/git_rmanager/queue
  Heatmap: /home/users/akourtis/git_rmanager/heatmap

Scheduling:
  Algorithm: FCFS
  Backfilling: 0

Nodes: 8

Sockets: 2
Cores: 4

Allocation:
  Policy: compact

Selection:
  Variables: cores, sockets, nodes
  LambdaLocation: config.SelectionFunc
  LambdaName : a

Export:
  ExportModuleFolder : export_modules
  Module : NPB

SpeedUp:
  File: config.SCIrouter
  Name: SpeedUp
AlreadyRun : False

Log: /home/users/akourtis/git_rmanager/log/compact/fcfs

State: /home/users/akourtis/git_rmanager/state
```

Σχήμα 5.5: *Καινούριο αρχείο yaml*

Τέλος, στο bash script αρχείο της εικόνας 4.1 έχει μπει μία επιπλέον γραμμή για να μπορεί το NPB να τυπώνει τους χρόνους, όπως επίσης η αρχικοποίηση του conda περιβάλλοντος(5.2.7).

```
#!/bin/bash

#Give the job a descriptive name
#PBS -N rmanager

## Output and error files
#PBS -o demo.out
#PBS -e demo.err

## Limit memory, runtime etc.
#PBS -l walltime=10:30:00

## How many nodes:processors_per_node should we get?
#PBS -l nodes=8:ppn=8

export PATH=/various/common_tools/gcc-4.5.2/bin:$PATH
export LD_LIBRARY_PATH=/various/common_tools/gcc-4.5.2/lib64/

module load openmpi/1.8.3

## Replace with your path
cd /home/users/akourtis/git_rmanager
export NPB_TIMER_FLAG=on
/home/users/akourtis/miniconda3/condabin/conda init bash
source /home/users/akourtis/miniconda3/bin/activate
/home/users/akourtis/miniconda3/condabin/conda activate thesis
python3 main.py -c config/compact.yaml
```

Σχήμα 5.6: *Καινούριο αρχείο bash script*

5.2.7 Γενικές αλλαγές

Στης γενικές αλλάγες υπάρχουν διάφορα Quality of life όπως για παράδειγμα να μπορεί η εφαρμόγη να τρέχει με ένα ήδη υπάρχον και ορισμένο αρχείο ουράς, να διαγράφει παλιούς φακέλους και αρχεία από παλαιότερες εκτελέσεις που τερματίστηκαν πρόωρα όπως και μία μεγαλύτερη ανοχή σε λάθη που αλλιώς θα τερμάτιζαν την εφαρμογή. Επίσης η προσθήκη της μεταβλητής AlreadyRun του προηγούμενου κεφαλαίου είναι μία καλή λύση εξοικονόμησης χρόνου για την κατασκεύη εκ νέου των αρχείων εξόδου(πχ Γραφήματα 5.3) χώρις να χρειαστεί να τρέξουν ξανά οι εργασίες.

Σημαντική προσθήκη τέλος είναι εκείνη του conda περιβάλλοντος, δίνοντας μία καλύτερη λύση στην ανάπτυξη της εφαρμογής για την προσθήκη καινουριών βιβλιοθηκών χώρις να υπάρχει φόβος να χαλάσει κάτι άλλο στο σύστημα.

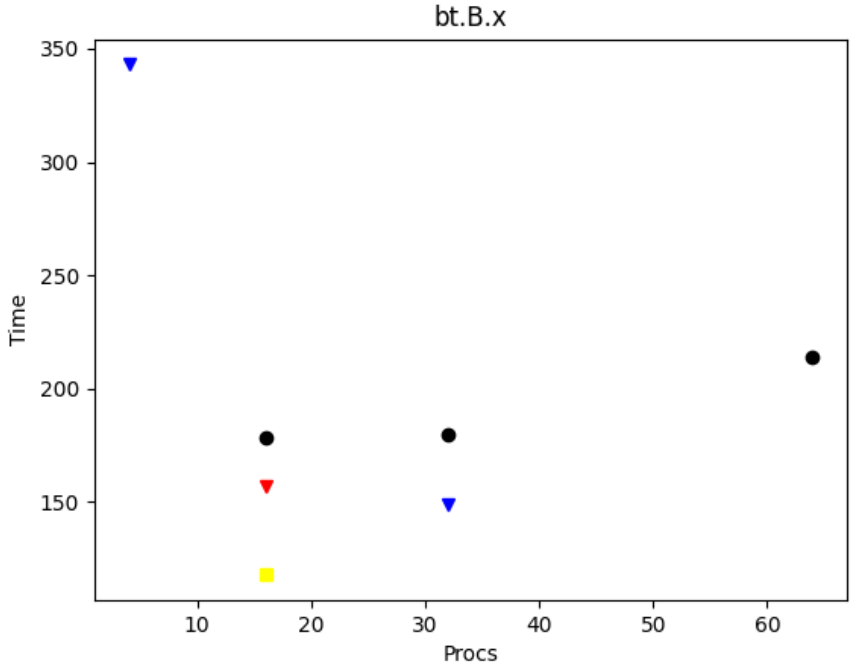
5.3 Γραφήματα

Ένα ξεχωριστό κεφάλαιο των αλλαγών αποτελεί εκείνο των γραφημάτων. Σκοπός αυτών είναι να παρέχουν στον χρήστη μια καλύτερη εποπτεία των αποτελεσμάτων για να μπορέσει να γίνει καλύτερη η αξιολόγηση του κάθε πειράματος.

Στην συνέχεια γίνεται αναφορά στα διαφορετικά γραφήματα που έχουν προστεθεί για την καλύτερη κατανόηση τους.

5.3.1 Job Plot

Αυτό το γράφημα παράγεται για κάθε εργασία που έτρεξε στο πρόγραμμά μας. Σε αυτό εμφανίζεται κάθε διαφορετική εκτέλεση της εργασίας στο πείραμα, για παράδειγμα εάν η εργασία Bt έτρεξε τρεις φορές στο συγκεκριμένο Data set, στο διάγραμμα της Bt θα υπάρχουν τρία σημεια, ένα για κάθε ένα από αυτά. Εάν μια εργασία έχει τρέξει σε πολλαπλές κλάσεις, θα φτιαχτεί ένα διάγραμμα για κάθε μία από αυτές.



Σχήμα 5.7: *Παράδειγμα Job Plot της εργασίας Bt*

Το διάγραμμα έχει τα processes συναρτήσει του χρόνου, και κάθε σημάδι στο διάγραμμα δίνει διαφορετική πληροφορία για την συγκεκριμένη εκτέλεση ώς εξής:

- **Κύκλος:** Αυτό σημαίνει ότι η εκτέλεση εκείνο της εργασίας ήταν με πολιτική compact
- **Τετράγωνο:**Αυτό σημαίνει ότι η εκτέλεση εκείνο της εργασίας ήταν με πολιτική spare, δηλαδή είτε είχε job exclusive είτε ήταν σε πολιτική strip και δεν έτρεξε με καμία άλλη.
- **Ανάδελτα:** Το ανάδελτα δείχνει οτι μία εργασίας στο συγκεκριμένο instance έτρεξε με πολιτική strip. Αξίζει να σημειωθεί οτι το χρώμα του ανάδελτα δίνει πληροφορία σχετικά με την εργασία που έτρεξε μαζί, με αντιστοιχεία όπως περιγράφεται στην εικόνα 5.8

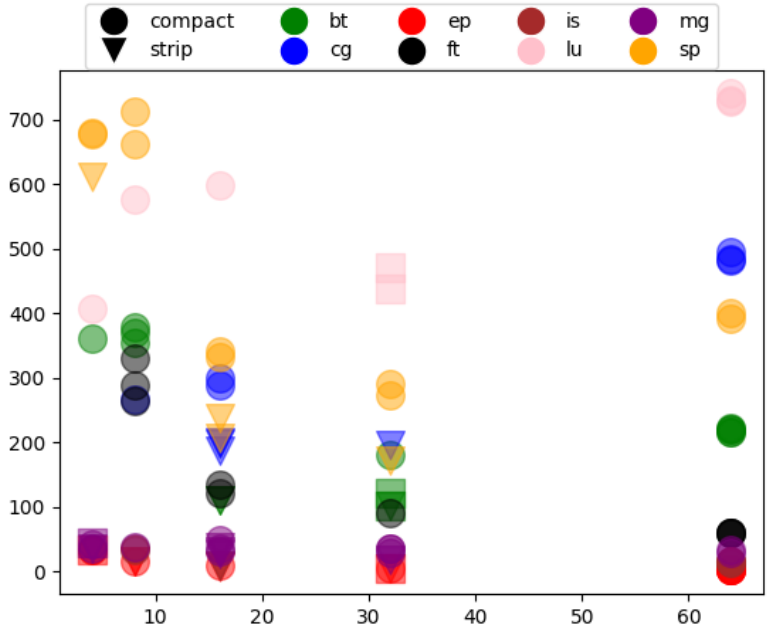
```
'bt' : 'green',
'cg' : 'blue',
'ep' : 'red',
'ft' : 'black',
'is' : 'brown',
'lu' : 'pink',
'mg' : 'purple',
'sp' : 'orange'}
```

Σχήμα 5.8: *Αντιστοιχεία χρωμάτων και εργασιών*

5.3.2 Scatter Plot

Το Scatter Plot είναι ένα γράφημα που περιέχει όλες τις εργασίες του συγκεκριμένου πειράματος.Το διάγραμμα έχει τα processes συναρτήσει του χρόνου και ακολουθεί την ίδια αντιστοιχεία χρωμάτων της εικόνας 5.8 όπως επίσης και τα σχήματα του προηγούμενου υποκεφαλαίου.

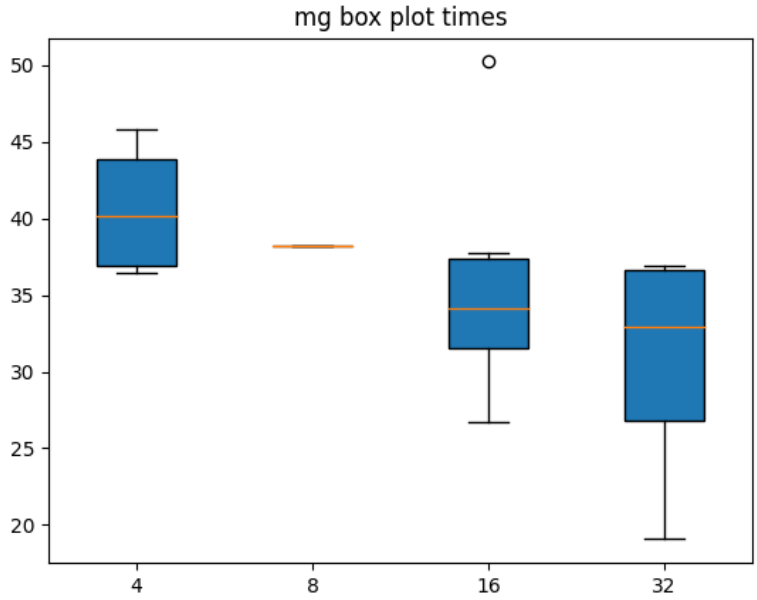
Το διάγραμμα αυτό δίνει στον χρήστη μία γενική ιδέα των εργασίων του προγράμματος, βλέποντας εύκολα και γρήγορα εκείνες που χρειάστηκαν τον περισσότερο χρόνο και των πολιτικών τους.



Σχήμα 5.9: *Παράδειγμα Scatter Plot σε πείραμα με 100 εργασίες*

5.3.3 Box Plot

Αυτό το διάγραμμα ομαδοποιεί τις εκτελέσεις κάθε εργασίας στο πείραμα ανά τα processes που χρησιμοποίησαν, και φτιάχνει το διάγραμμα συναρτήσει του χρόνου σύμφωνα με τους κανόνες ενός Box Plot[10]. Αξίζει να σημειωθεί ότι εκτελέσεις ίδιας εργασίας με ίδια processes αλλά διαφορετικών κλάσεων επίσης θα ομαδοποιηθούν.

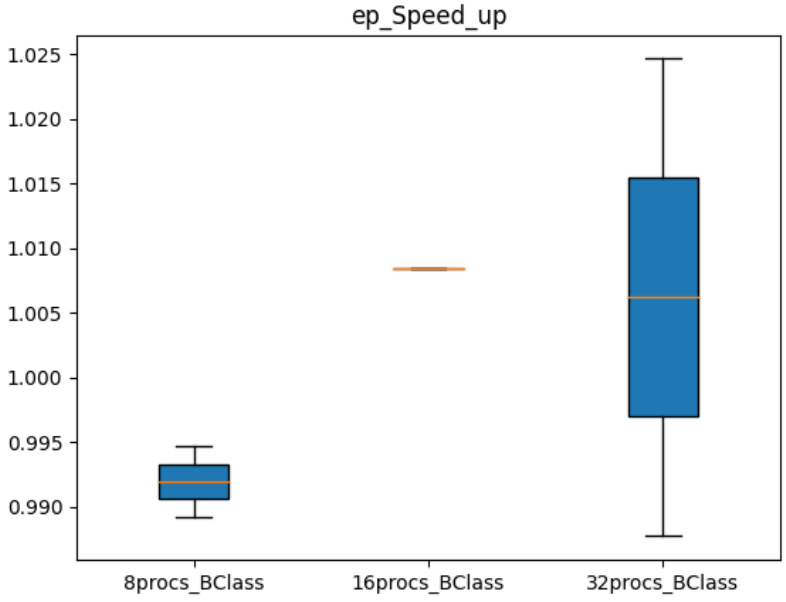


Σχήμα 5.10: *Box Plot της εργασίας Mg*

Συμφωνα με τον ορισμό του Box Plot, η κίτρινη γραμμή μεσά στα κουτιά της εικόνας 5.10 δηλώνει την μέση τιμή. Οι τιμές μέσα στο μπλε κουτί είναι το 50% των τιμών κοντινότερες στο μέσο(25% από πάνω και 25% από κάτω). Οι γραμμές πάνω και κάτω του κουτιού δείχνουν τις εναπομείναντες τιμές, 25% πάνω και 25% κάτω. Τέλος, τιμές που εμφανίζονται με κύκλο δηλώνουν τιμές που απέχουν πολύ από την μέση τιμή και θεωρούνται ανωμαλίες.

5.3.4 Speed-up Plot

Παρόμοια με το Box Plot, το Speed-up Plot ομαδοποιεί με παρόμοιο τρόπο τις εργασίες, αυτήν την φορά όμως δεν γίνεται σε συνάρτηση με τον χρόνο που έκανε η εργασία, αλλά με το πυλίκο του χρόνου που θα χρειάζονταν εάν είχε εκτελεστεί σε πολιτική compact με τον χρόνο εκτέλεσής της. Μια μεγάλη διαφόρα στην ομαδοποίηση των εργασιών απο το Box Plot είναι ότι μια εργασία με διαφορετικες κλάσεις θα έχει διαφορετικά entries στον οριζόντιο άξονα για κάθε κλάση, και τέλος το Speed-up Plot χρησιμοποιεί τις εκτελέσεις που έχουν πολιτική strip ή και spare.

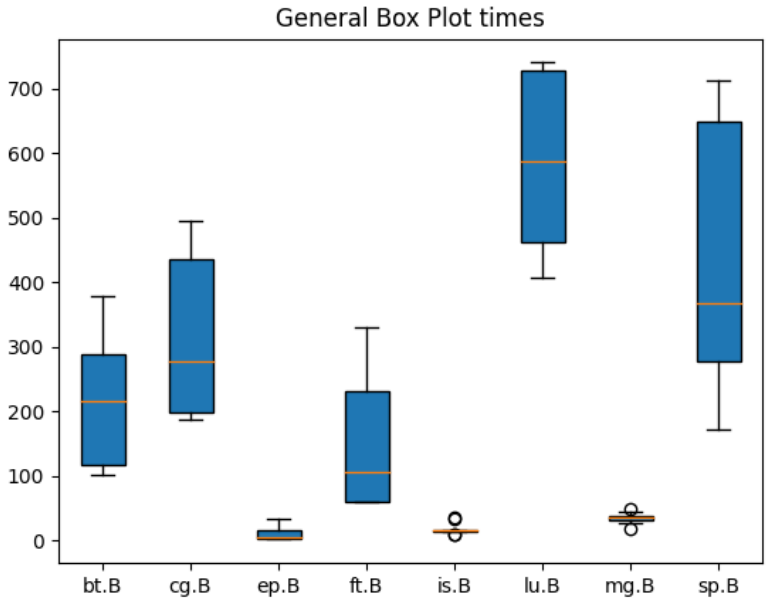


Σχήμα 5.11: *Παράδειγμα Speed-up Plot*

Μια τιμή μεγαλύτερη του ενός θα σημαίνει οτι η εφαρμογή εκτελείται πιο γρήγορα σε πολιτική strip παρά σε compact, ενώ μία τιμή μικρότερη του ένα θα σημαίνει το αντίθετο. Για παράδειγμα, στην εικόνα 5.11, στο συγκεκριμένο πείραμα φαίνεται πως η εργασία ep με 8 processes και B κλάση είναι λίγο πιο αργή σε πολιτική strip, ενώ με 16 processes λίγο πιο γρήγορη.

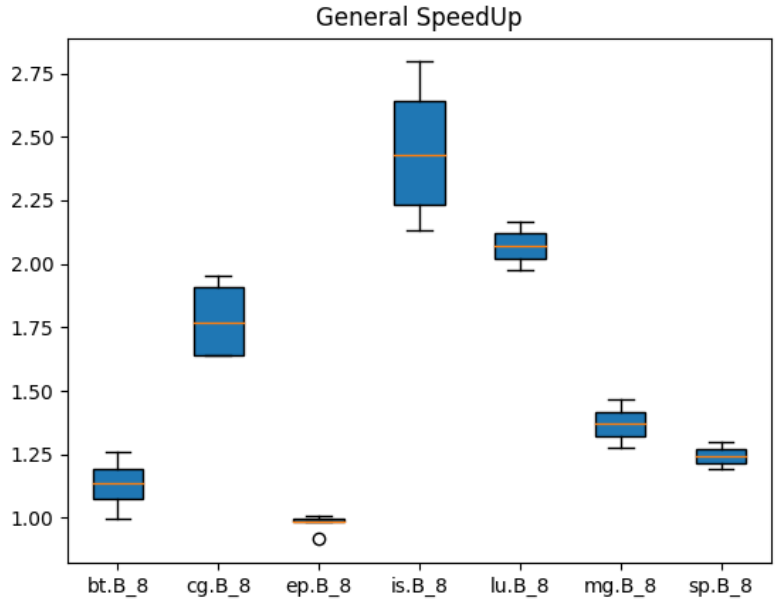
5.3.5 Γενικό Box Plot και Speed-up Plot

Συνοψίζοντας με τα γραφήματα, το γενικό Box Plot είναι το τελευταίο διάγραμμα που έχει προστεθεί. Σε αυτό το διάγραμμα, ομαδοποιούνται εργασίες ανά κλάση ανεξαρτήτως των processes της κάθε μίας, και εκφράζονται σε συνάρτηση με τον χρόνο. Παρόμοια με το Scatter plot, το γενικό Box Plot βοήθαει τον χρήστη να μπορέσει να δει μια γενική εικόνα του πειράματος που έτρεξε με έναν εύκολο και γρήγορο τρόπο.



Σχήμα 5.12: *Παράδειγμα Γενικό Box Plot*

Ομοίως με το γενικό Box Plot, το γενικό Speed-up Plot περιέχει τα speed-up των εργασιών του πειράματος με σκόπο να βοηθήσει πάλι τον χρήστη να δει μία γενική εικόνα του πειράματος εύκολα και γρήγορα.



Σχήμα 5.13: *Παράδειγμα Γενικού Speed-up Plot*

5.4 Χρήση

Η χρήση της εφαρμογής δεν έχει μεγάλες διαφορές από εκείνη που περιγράφτηκε στο κεφάλαιο 4.3. Πριν την εκτέλεση θα πρέπει να γίνει σίγουρο ότι μέσα έχουν οριστεί σωστά τα καινούρια πεδία στο yaml configuration αρχείο. Αυτό σημαίνει ότι πρέπει να μπουν και να οριστούν σωστά η συνάρτηση επιλογής(5.2.2) και το Export Module(5.2.3). Επιπλέον, εάν χρείαζεται η εφαρμογή να εκτελέση ένα συγκεκριμένο data set, θα πρέπει αυτό να οριστεί εκ τον προτέρων στο αρχείο της ουράς, αλλιώς να διαγραφτεί τελείως για την αυτόματη δημιουργία ενώς νέου. Στην περίπτωση που χρειάζεται να τεθεί η μεταβλητή AlreadyRun για την εκτέλεση της εφαρμογής μόνο για επεξεργασία ήδη υπάρχοντων αποτελεσμάτων, πρέπει τα rankfiles και το αρχείο της ουράς να έχουν τοποθετηθεί με σωστό τρόπο για να εξασφαλιθεί η ορθή εκτέλεση της εφαρμογής.

Το στάδιο κατά την εκτέλεση παραμένει το ίδιο χωρίς αλλαγές.

Τέλος, μετά την πέρας της εφαρμογής, όλα τα γραφήματα που περιγράφτηκαν στο κεφάλαιο 5.3 μπορούν να βρεθούν στον ίδιο φάκελο με τα rankfiles, δηλαδή μέσα στον φάκελο logs.

5.5 GitHub Branch

Πηγαίος κώδικας μαζί με τις λειτουργίες που έχουν προστεθεί μπορούν να βρεθούν στο παρακάτω GitHub repository :https://github.com/IrrelevantX/vRJMS2.0

Κεφάλαιο 6

Εκτίμηση Αποτελεσμάτων

6.1 Σύστημα που χρησιμοποιήθηκε

Το συστημα που χρησιμοποιήθηκε ήταν το cluster των clones του CsLab, που αποτελείται από οκτώ όμοιους υπολογιστές. Κάθε υπολογιστής χρησιμοποιεί δύο επεξεργαστές τύπου Intel® Xeon® Processor E5335[11]. Ο κάθε ένας από αυτούς έχει τέσσερις πυρήνες 8Mb L2 cache και ταχύτητα διάβλου(Bus speed) 1333MHz. Κάθε σύστημα επίσης έχει μνήμη RAM μεγάθους 8 Gb και λειτουργικό σύστημα Debian GNU.Αναλυτικότερα, τα Specifications του κάθε μηχανήματος φαίνονται στην εικόνα 6.1.

```
(base) akourtis@clone2:~$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      1
Core(s) per socket:      4
Socket(s):               2
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   15
Model name:              Intel(R) Xeon(R) CPU           E5335  @ 2.00GHz
Stepping:                7
CPU MHz:                 2000.182
BogoMIPS:                4000.47
Virtualization:          VT-x
L1d cache:               32K
L1i cache:               32K
L2 cache:                4096K
NUMA node0 CPU(s):       0-7
```

Σχήμα 6.1: *Specifications ενός συστήματος*

Αξίζει να σημειωθεί πως, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, τα μηχανήματα έχουν ένα conda περιβάλλον, με σκοπό να είναι πιο εύκολη και ασφαλής η προσθήκη καινούριων packages.

```
# packages in environment at /home/users/akourtis/miniconda3/envs/thesis:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                         main
_openmp_mutex             5.1                         1_gnu
ca-certificates           2022.10.11                  h06a4308_0
certifi                   2022.9.24                py39h06a4308_0
ld_impl_linux-64          2.38                        h1181459_1
libffi                    3.4.2                       h6a678d5_6
libgcc-ng                 11.2.0                      h1234567_1
libgomp                   11.2.0                      h1234567_1
libstdcxx-ng              11.2.0                      h1234567_1
ncurses                   6.3                         h5eee18b_3
openssl                   1.1.1s                      h7f8727e_0
pip                       22.3.1                   py39h06a4308_0
python                    3.9.15                      h7a1cb2a_2
readline                  8.2                         h5eee18b_0
setuptools                65.5.0                   py39h06a4308_0
sqlite                    3.40.0                      h5082296_0
tk                        8.6.12                      h1ccaba5_0
tzdata                    2022g                       h04d1e81_0
wheel                     0.37.1                     pyhd3eb1b0_0
xz                        5.2.8                       h5eee18b_0
zlib                      1.2.13                      h5eee18b_0
```

Σχήμα 6.2: *Conda specifications*

6.2 Dataset που χρησιμοποιήθηκε

Τα πειράματα που έγιναν χρησιμοποίησαν τα Dataset που είναι διαθέσιμα στο παράρτημα A′.

Συγκεκριμένα, τα πειράματα που έγιναν ήταν τα εξής:

- **Τριάντα τυχαίες εργασίες, οκτώ διεργασιών η καθεμία (A′.1)**
- **Σαράντα τυχαίες εργασίες, τριάντα δύο διεργασιών η καθεμία (A′.2)**
- **Πενήντα τυχαίες εργασίες, δεκαέξι διεργασιών η καθεμία (A′.3)**
- **Εκατό τυχαίες εργασίες με τυχαίο αριθμό διεργασιών η καθεμία (A′.4)**
- **Διακόσιες πενήντα τυχαίες εργασίες με τυχαίο αριθμό διεργασιών η καθεμία (A′.5)**

Αξίζει να συμειωθεί, οτι για τα πειράματα των τριάντα, σαράντα και πενήντα εργασιών, η εκτέλεση τους έγινε δύο φορές, μία όπου όλες έτρεξαν με πολιτική compact και μία που έτρεξαν με πολιτική strip, χρησιμοποιώντας την καινούρια λειτουργία του Co-Scheduling.

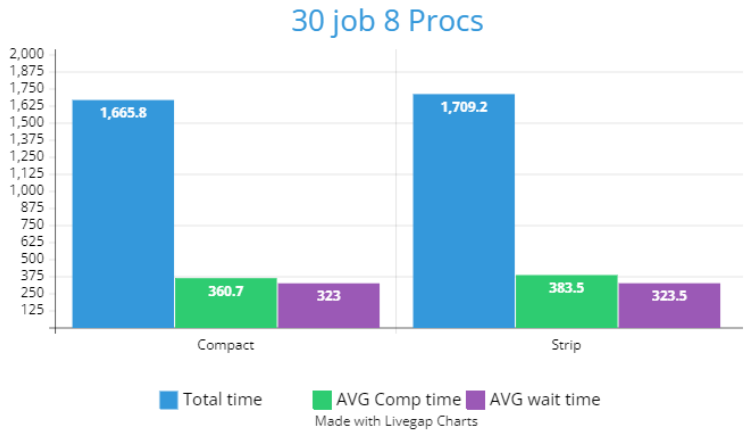
Η επιλογή των συγκεκριμένων πειραμάτων έχει ως σκοπό να δώσει μια εικόνα της εκτέλεσης της εφαρμογής με τις καινούριες λειτουργίες όπως επίσης και να υπάρξουν συμπεράσματα σχετικά με την συμπεριφορά των NAS Parallel Benchmarks σε διαφορερικά σενάρια εκτέλεσης.

Τέλος, λόγω διαφόρων προβλημάτων στην εκτέλεση που καθστούσαν αδύνατη την διεκπεραίωση κάποιων πειραμάτων, η εργασία ft δεν έχει συμπεριληφθεί στα πειράματα, πλην μόνο εκείνου των εκατό εργασιών.

6.3 Πειράματα/Αποτελέσματα

Σε αυτό το κεφάλαιο θα προσθεθούν τα αποτελέσματα τον πειραμάτων που περιγράφτηκαν στο 6.2. Αρχικά σε κάθε πείραμα υπάρχουν οι χρόνοι που έκανε συνολικά το πείραμα να ολοκληρωθεί, τον μέσο χρόνο που παρέμεινε μια διεργασία στην ουρά, όπως επίσης και τον μέσω χρόνο ολοκλήρωσης μίας εργασίας. Στην συνέχεια περιλαμβάνονται όλα τα γραφήματα που δημιουργήθηκαν από την εκτέλεση του κάθε πειράματος.

6.3.1 30job 8processes

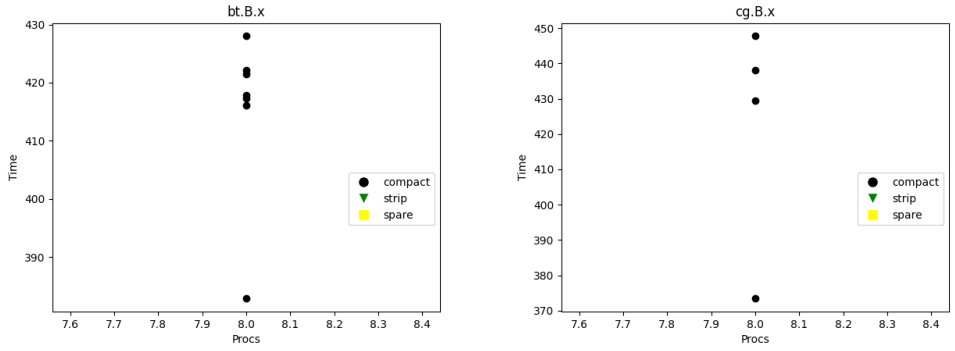


Σχήμα 6.3: *Διάγραμμα χρόνων πειράματος 30 εργασιών*

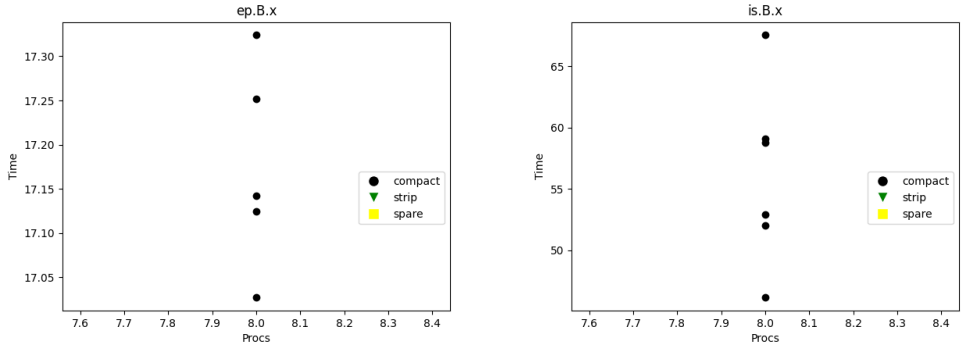
6.3.1.1 All compact

Συνολικός χρόνος εκτέλεσης : 1665.8 δευτερόλεπτα (28 λεπτά)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 360.7 δευτερόλεπτα (6 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 323 δευτερόλεπτα (5.3 λεπτά)

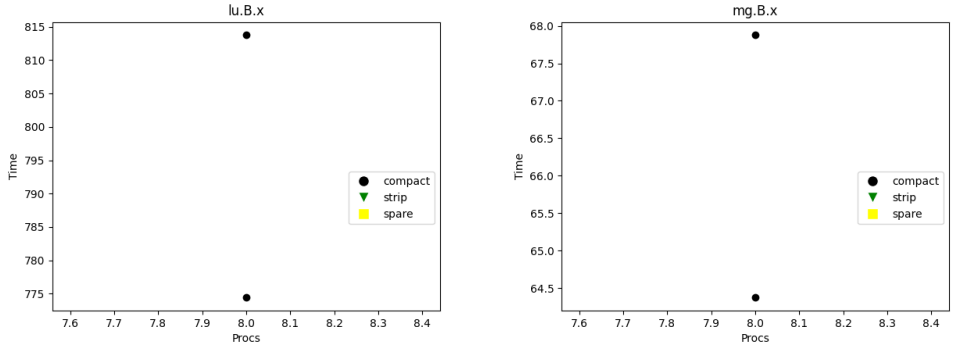
Γραφήματα εργασιών:



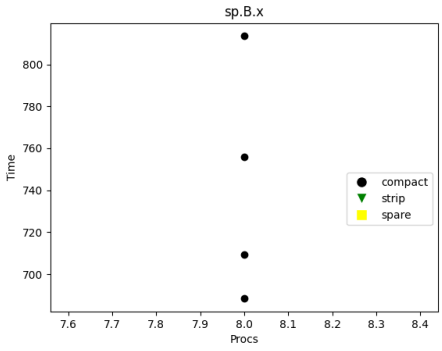
Σχήμα 6.4: *Διαγράμματα bt και cg*



Σχήμα 6.5: *Διαγράμματα ep και is*

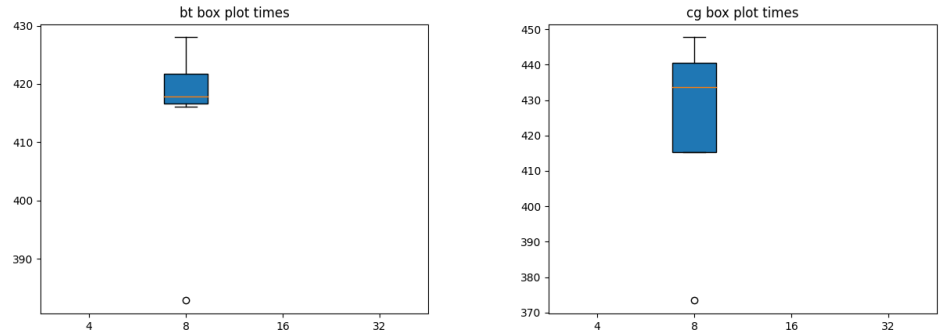


Σχήμα 6.6: *Διαγράμματα lu και mg*

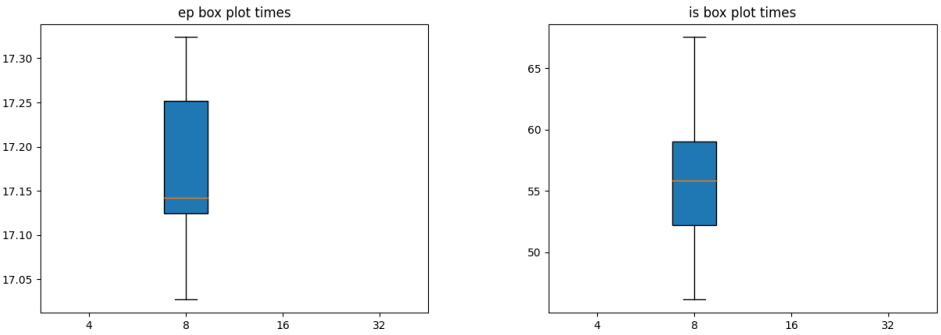


Σχήμα 6.7: *Διάγραμμα sp*

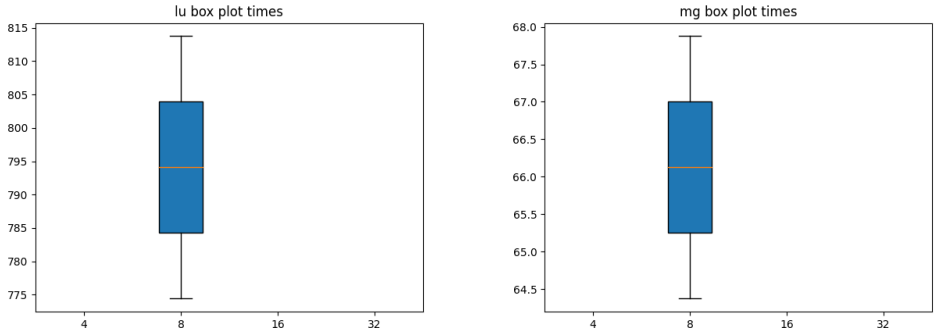
Γραφήματα τύπου Box plot των εργασιών:



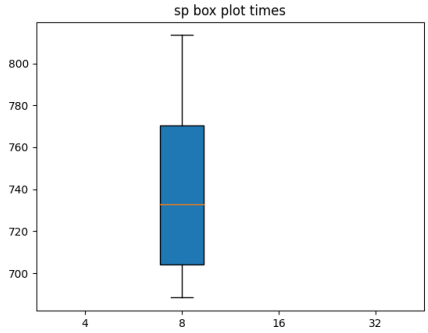
Σχήμα 6.8: *Box plot διαγράμματα bt και cg*



Σχήμα 6.9: *Box plot διαγράμματα ep και is*

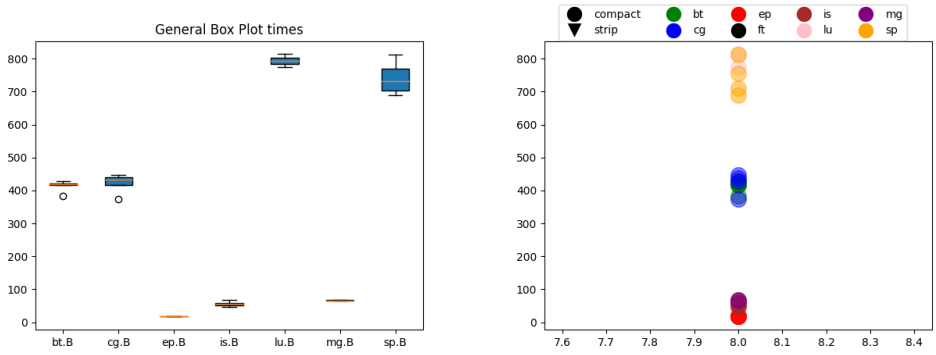


Σχήμα 6.10: *Box plot διαγράμματα lu και mg*



Σχήμα 6.11: *Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα

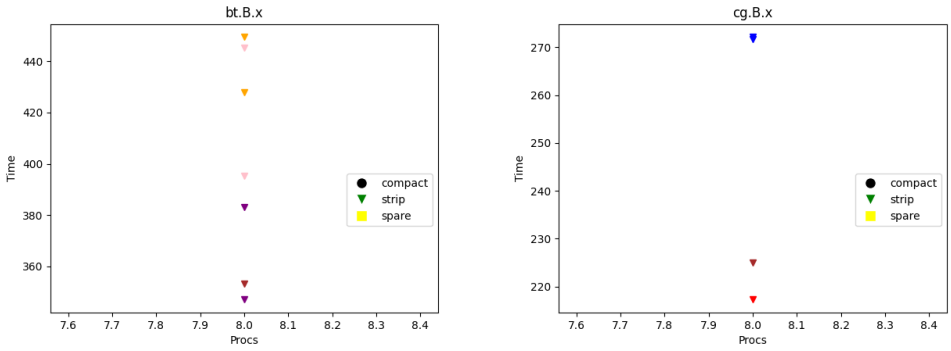


Σχήμα 6.12: *General Box plot και Scatter plot διαγράμματα του πειράματος*

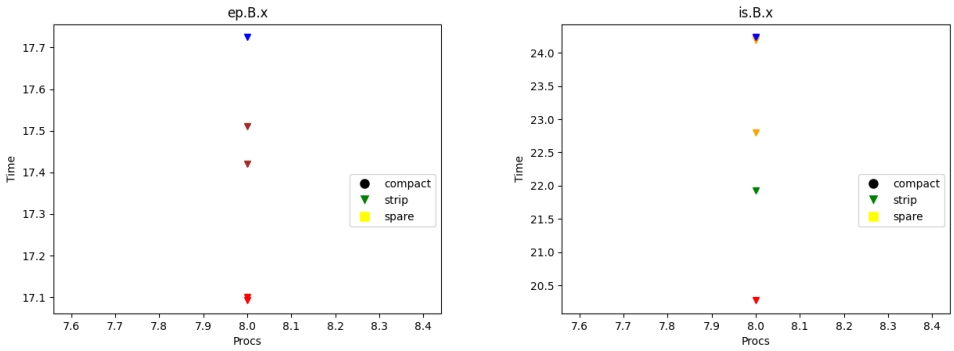
6.3.1.2 All strip

Συνολικός χρόνος εκτέλεσης : 1709.2 δευτερόλεπτα (28 λεπτά)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 383.5 δευτερόλεπτα (6.3 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 323.5 δευτερόλεπτα (5.3 λεπτά)

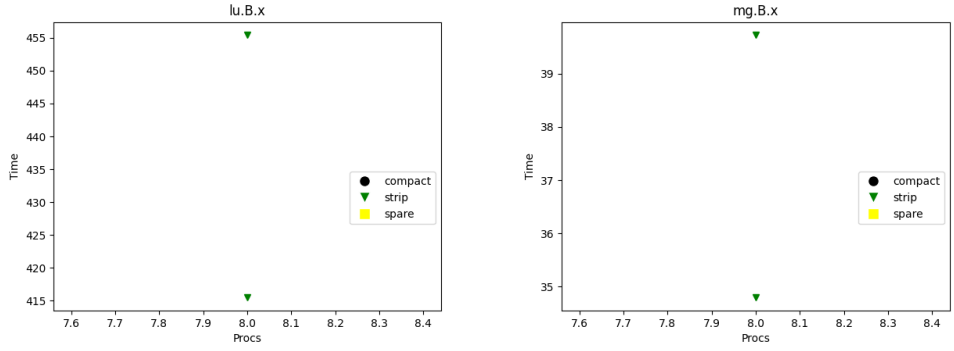
Γραφήματα εργασιών:



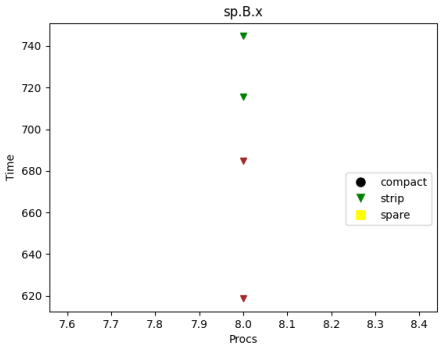
Σχήμα 6.13: *Διαγράμματα bt και cg*



Σχήμα 6.14: *Διαγράμματα ep και is*

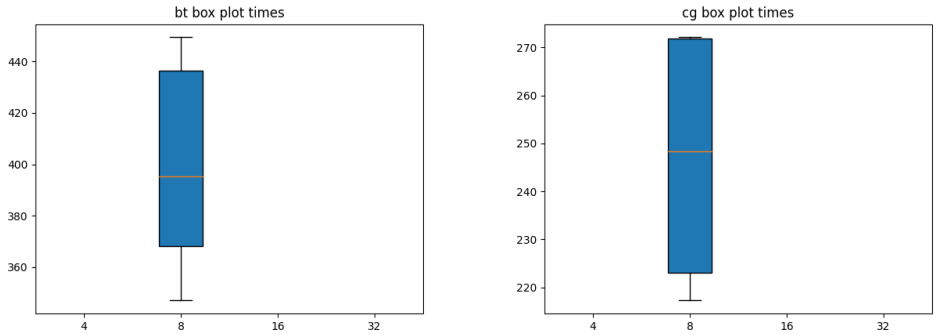


Σχήμα 6.15: *Διαγράμματα lu και mg*

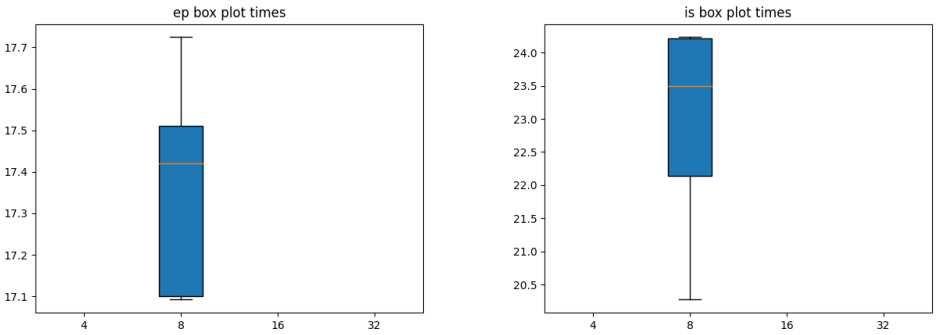


Σχήμα 6.16: *Διάγραμμα sp*

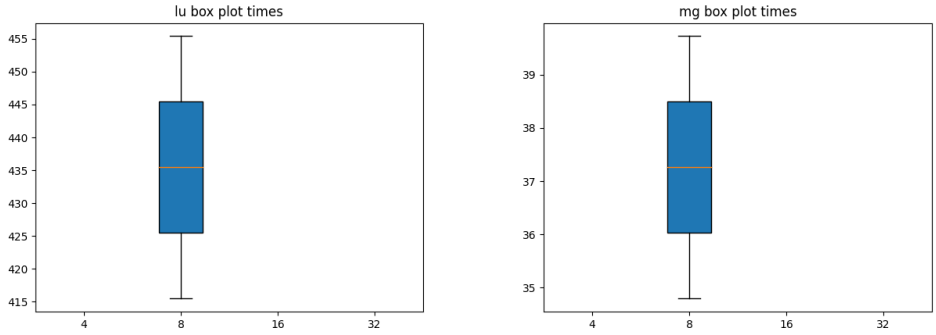
Γραφήματα τύπου Box plot των εργασιών:



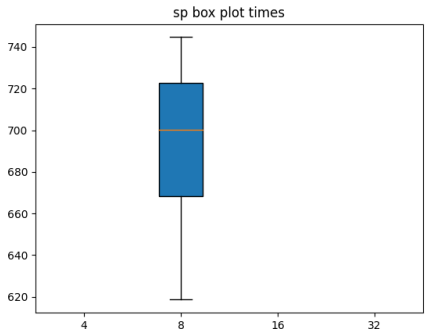
Σχήμα 6.17: *Box plot διαγράμματα bt και cg*



Σχήμα 6.18: *Box plot διαγράμματα ep και is*

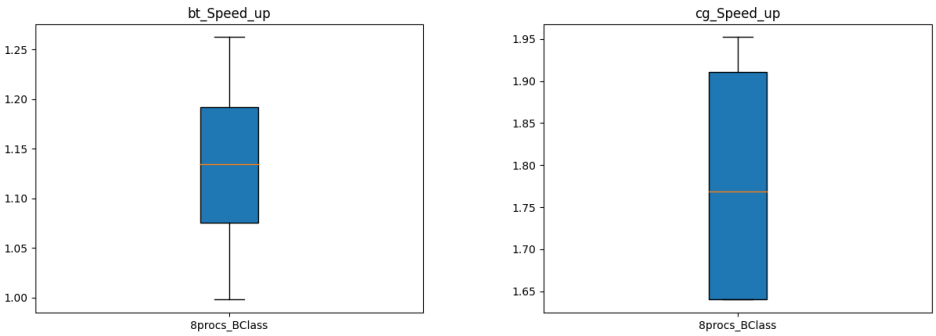


Σχήμα 6.19: *Box plot διαγράμματα lu και mg*

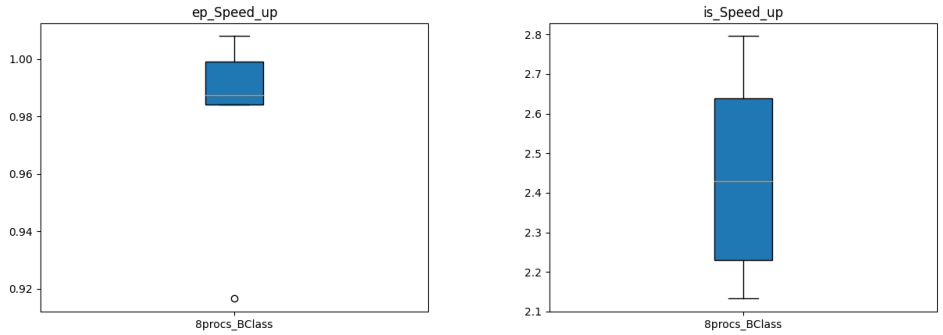


Σχήμα 6.20: *Box plot διάγραμμα sp*

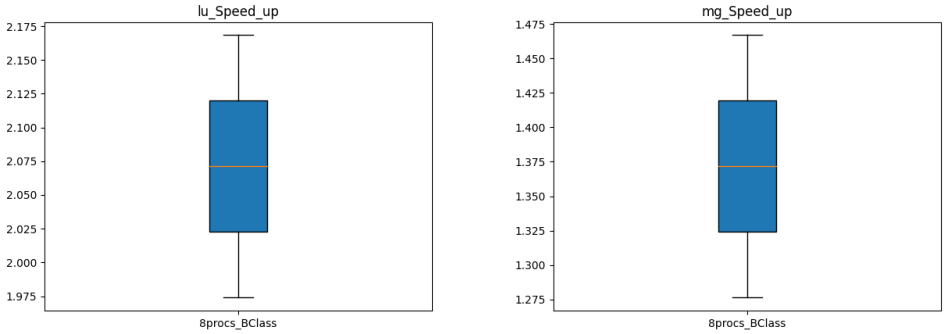
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



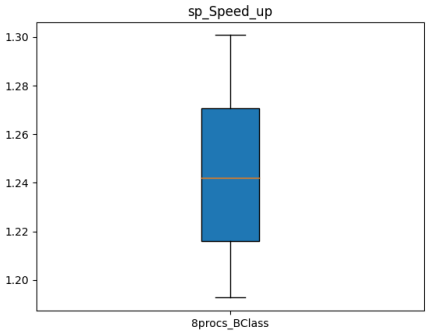
Σχήμα 6.21: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.22: *Speed-up Box plot διαγράμματα ep και is*

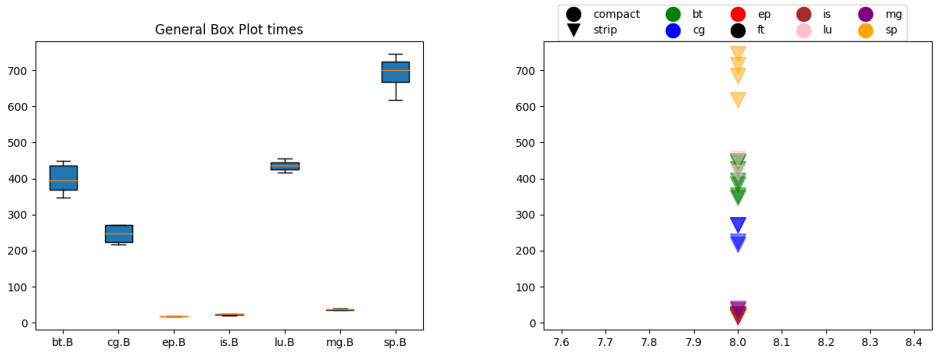


Σχήμα 6.23: *Speed-up Box plot διαγράμματα lu και mg*

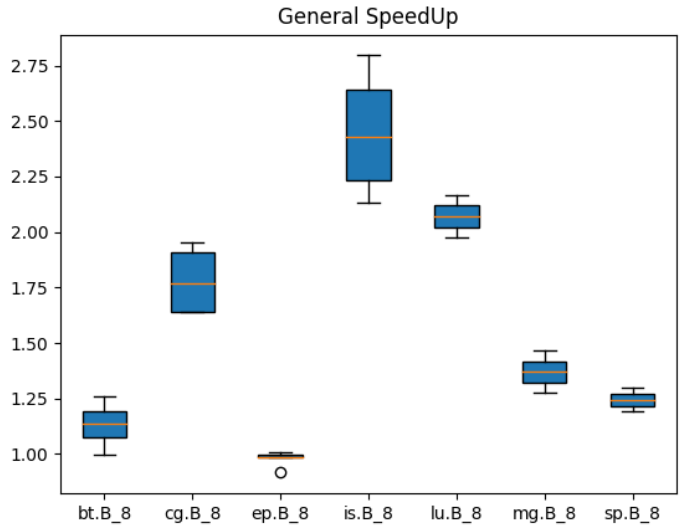


Σχήμα 6.24: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα

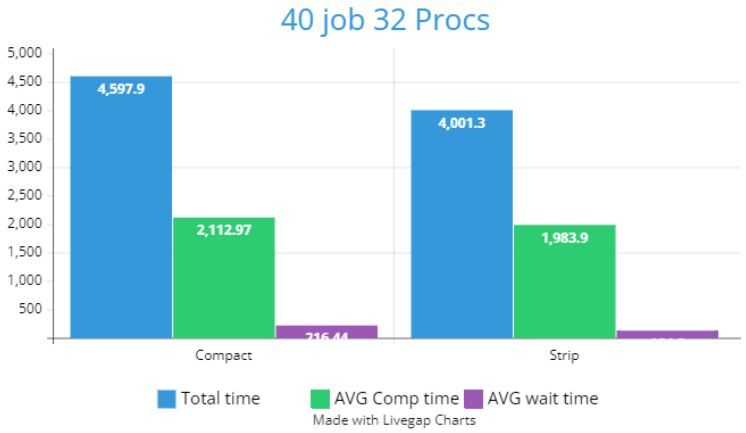


Σχήμα 6.25: *General Box plot και Scatter plot διαγράμματα του πειράματος*



Σχήμα 6.26: *Γενικό Speed-up plot διάγραμμα*

6.3.2 40job 32processes

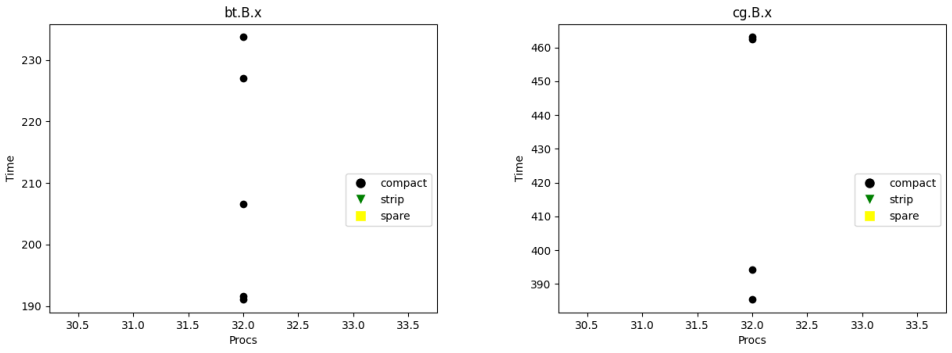


Σχήμα 6.27: *Διάγραμμα χρόνων πειράματος 40 εργασιών*

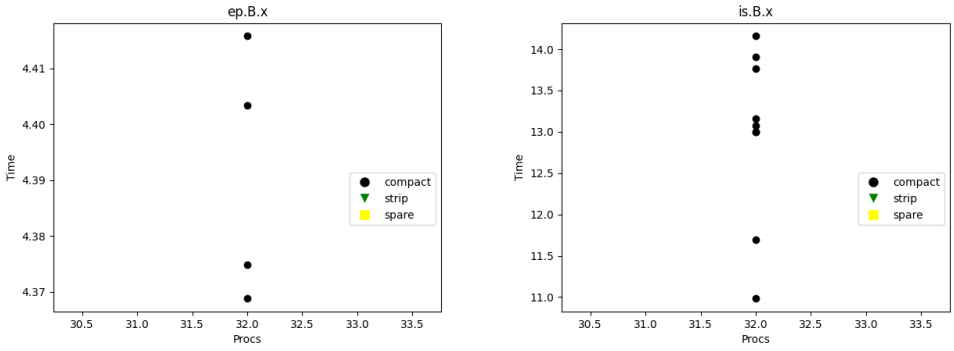
6.3.2.1 All compact

Συνολικός χρόνος εκτέλεσης : 4597.9 δευτερόλεπτα (1.2 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 2112.97 δευτερόλεπτα (35 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 216.44 δευτερόλεπτα (3.6 λεπτά)

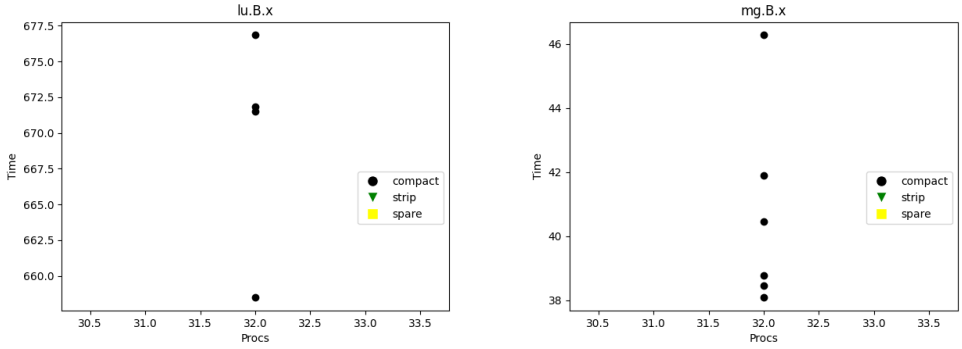
Γραφήματα εργασιών:



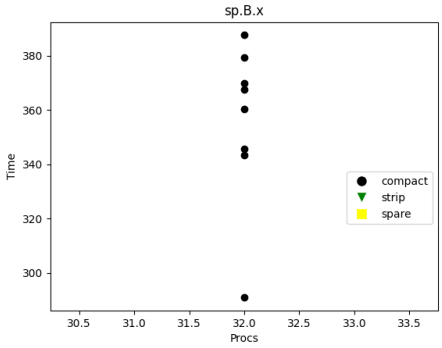
Σχήμα 6.28: *Διαγράμματα bt και cg*



Σχήμα 6.29: *Διαγράμματα ep και is*

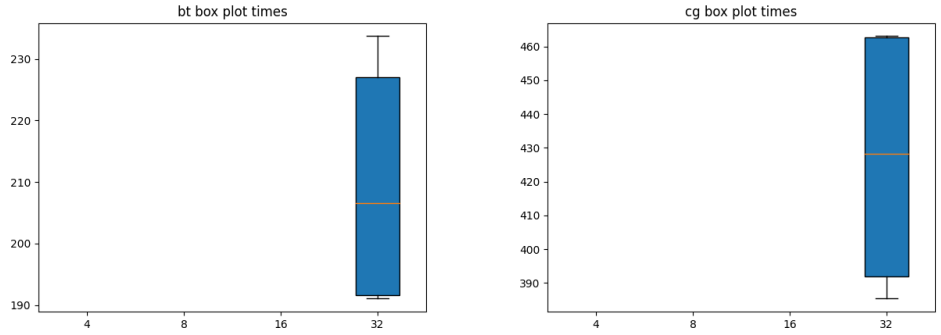


Σχήμα 6.30: *Διαγράμματα lu και mg*

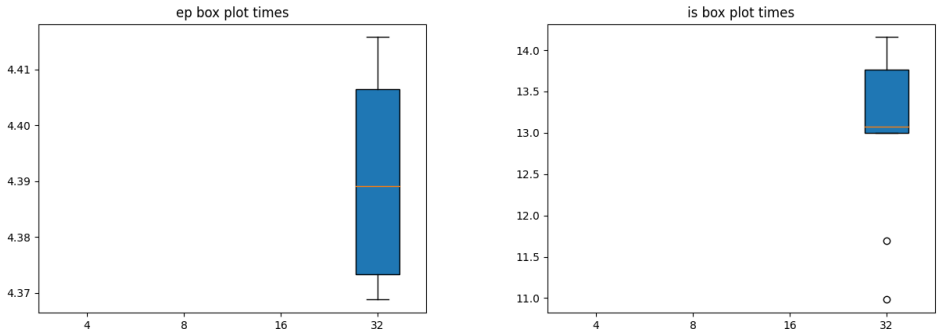


Σχήμα 6.31: *Διάγραμμα sp*

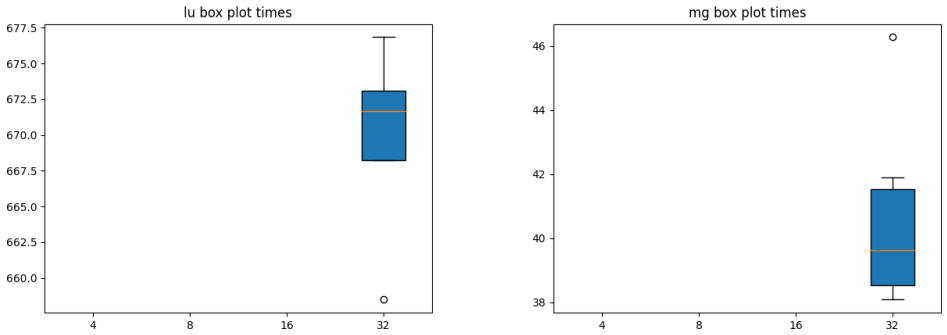
Γραφήματα τύπου Box plot των εργασιών:



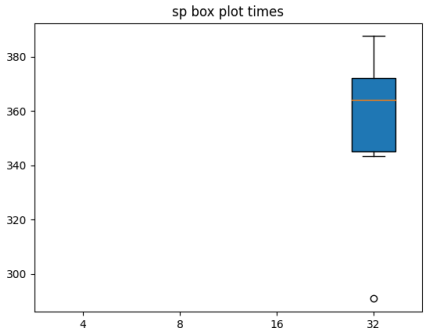
Σχήμα 6.32: *Box plot διαγράμματα bt και cg*



Σχήμα 6.33: *Box plot διαγράμματα ep και is*

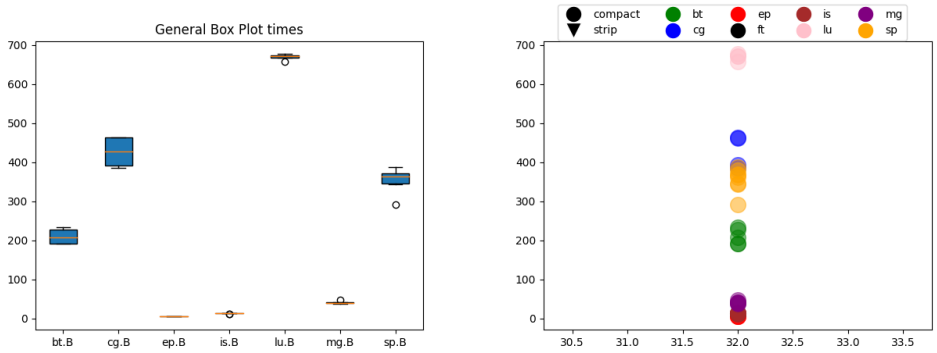


Σχήμα 6.34: *Box plot διαγράμματα lu και mg*



Σχήμα 6.35: *Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα

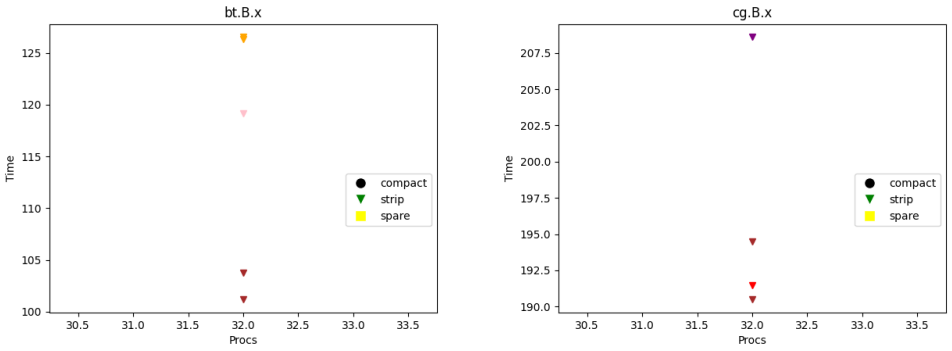


Σχήμα 6.36: *General Box plot και Scatter plot διαγράμματα του πειράματος*

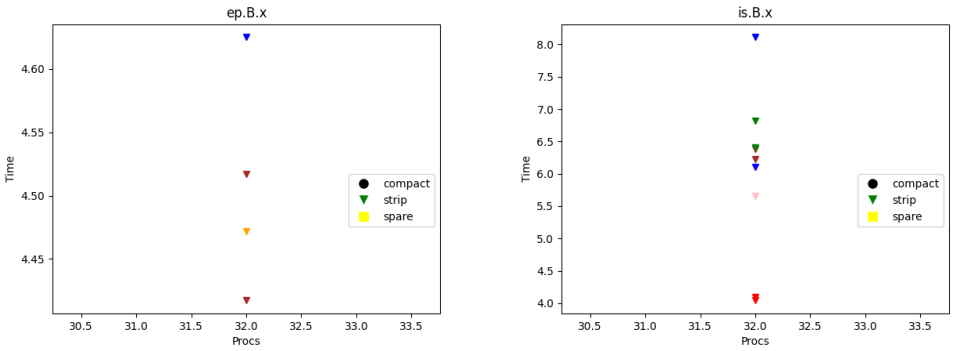
6.3.2.2 All strip

Συνολικός χρόνος εκτέλεσης : 4001.3 δευτερόλεπτα (1.1 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 1983.9 δευτερόλεπτα (33 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 124.8 δευτερόλεπτα (2 λεπτά)

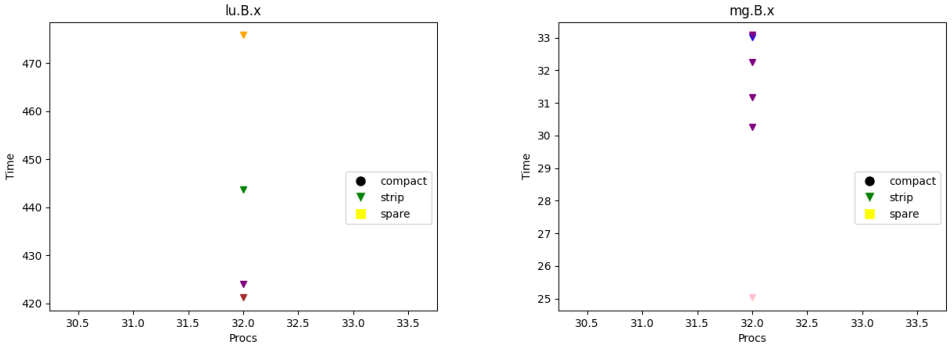
Γραφήματα εργασιών:



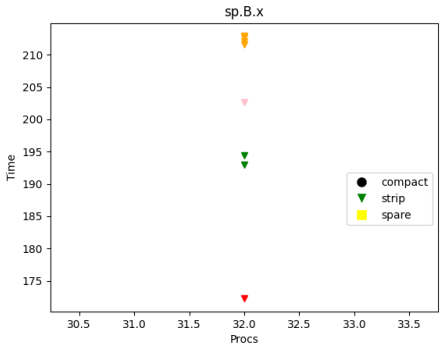
Σχήμα 6.37: *Διαγράμματα bt και cg*



Σχήμα 6.38: *Διαγράμματα ep και is*

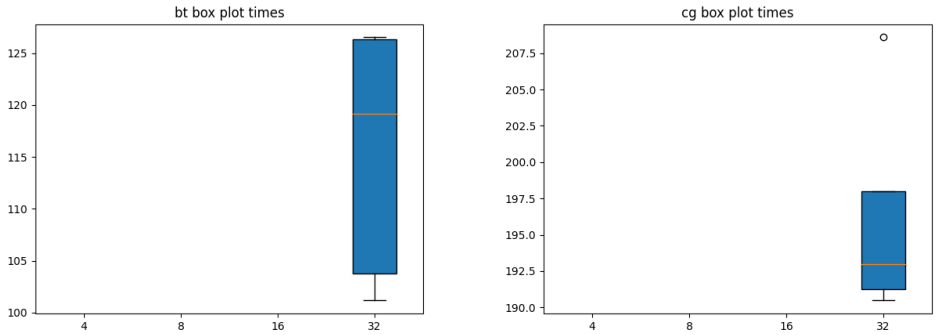


Σχήμα 6.39: *Διαγράμματα lu και mg*

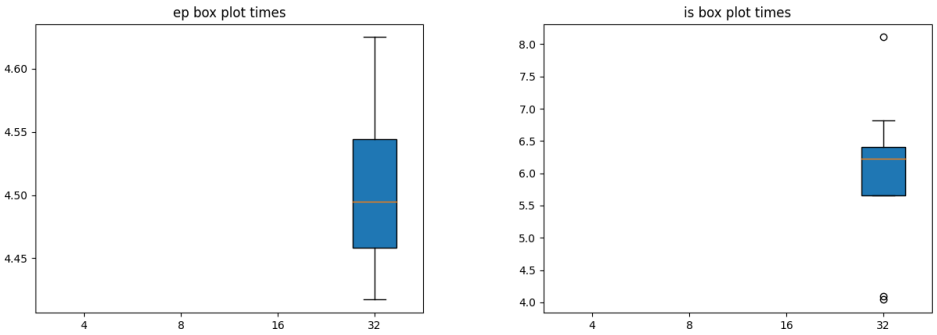


Σχήμα 6.40: *Διάγραμμα sp*

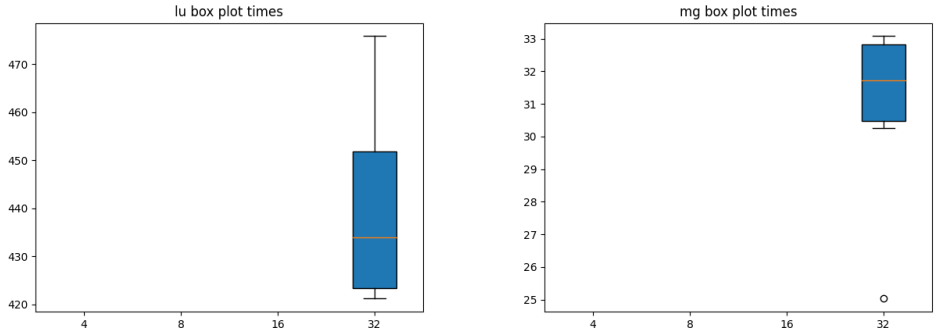
Γραφήματα τύπου Box plot των εργασιών:



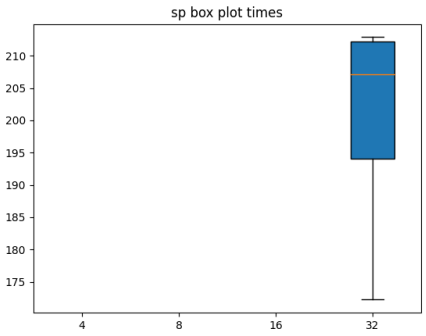
Σχήμα 6.41: *Box plot διαγράμματα bt και cg*



Σχήμα 6.42: *Box plot διαγράμματα ep και is*

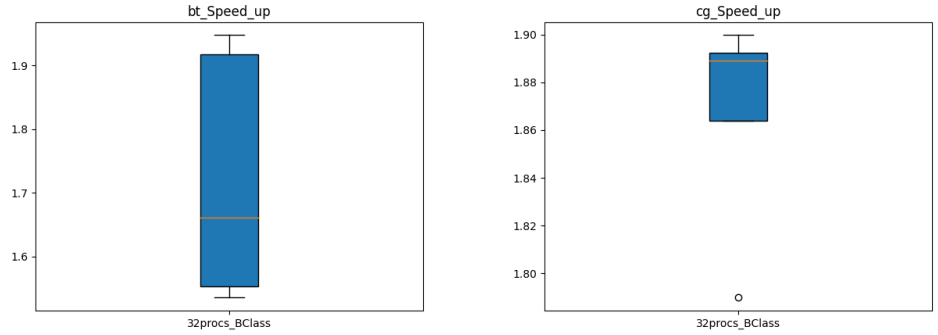


Σχήμα 6.43: *Box plot διαγράμματα lu και mg*

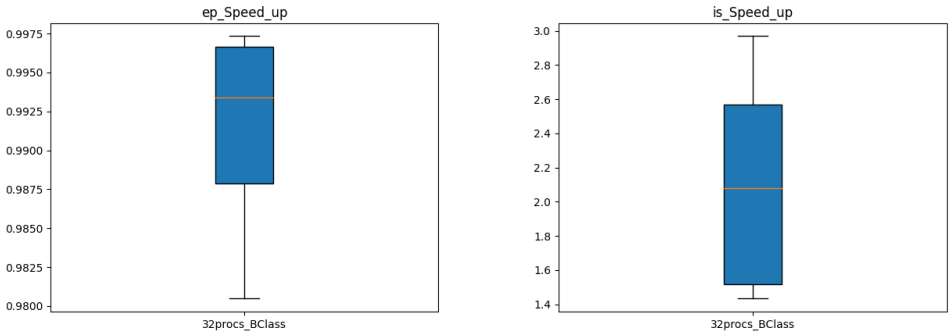


Σχήμα 6.44: *Box plot διάγραμμα sp*

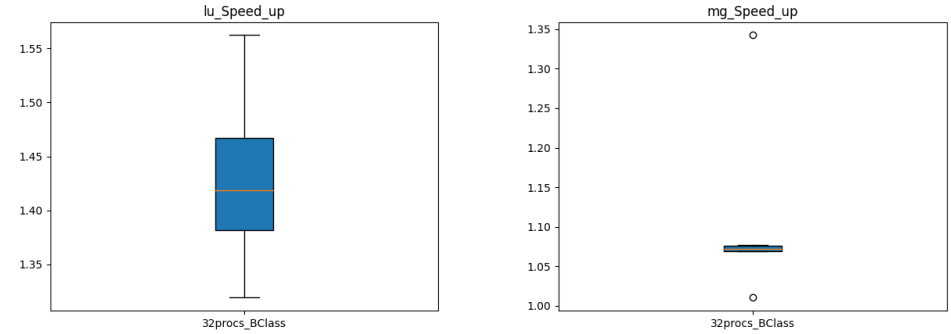
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



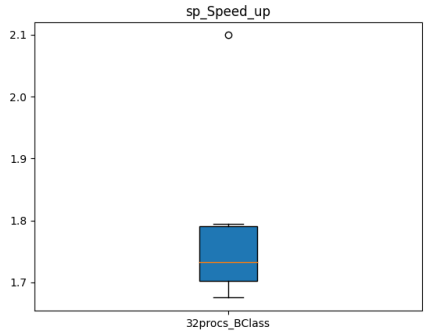
Σχήμα 6.45: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.46: *Speed-up Box plot διαγράμματα ep και is*

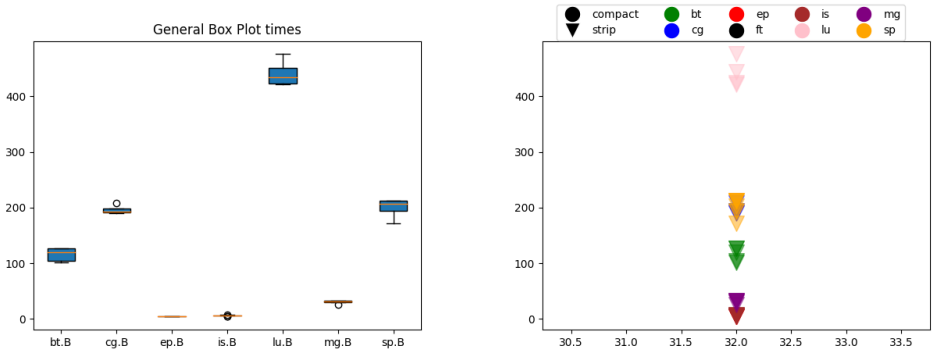


Σχήμα 6.47: *Speed-up Box plot διαγράμματα lu και mg*

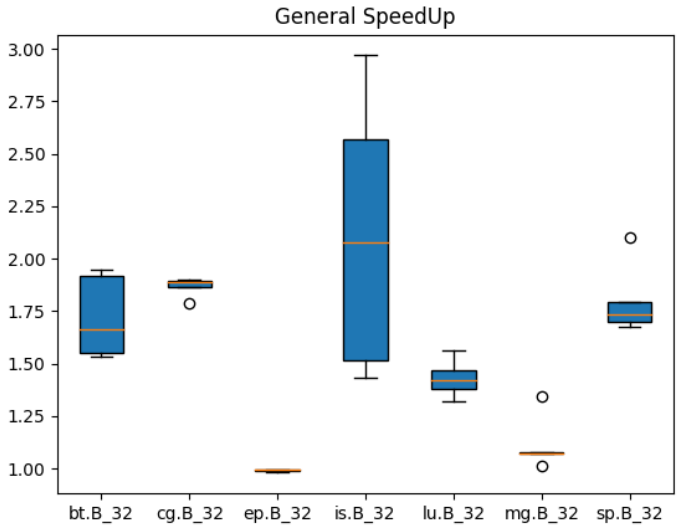


Σχήμα 6.48: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα

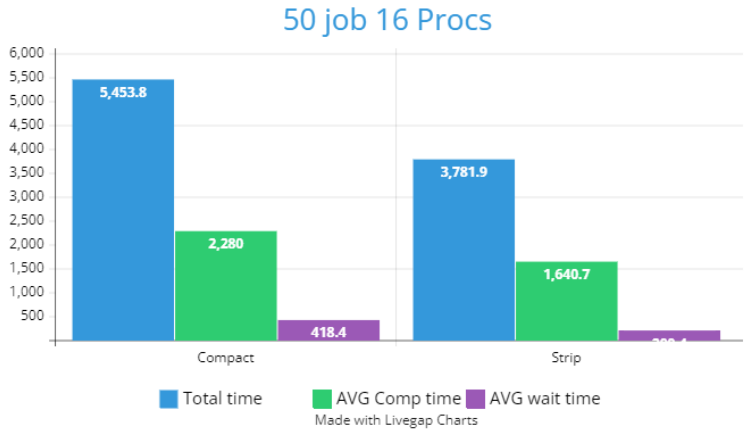


Σχήμα 6.49: *General Box plot και Scatter plot διαγράμματα του πειράματος*



Σχήμα 6.50: *Γενικό Speed-up plot διάγραμμα*

6.3.3 50job 16processes

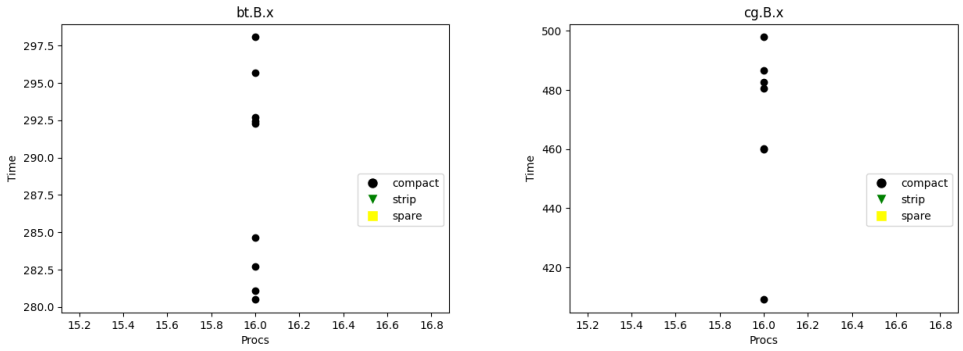


Σχήμα 6.51: *Διάγραμμα χρόνων πειράματος 50 εργασιών*

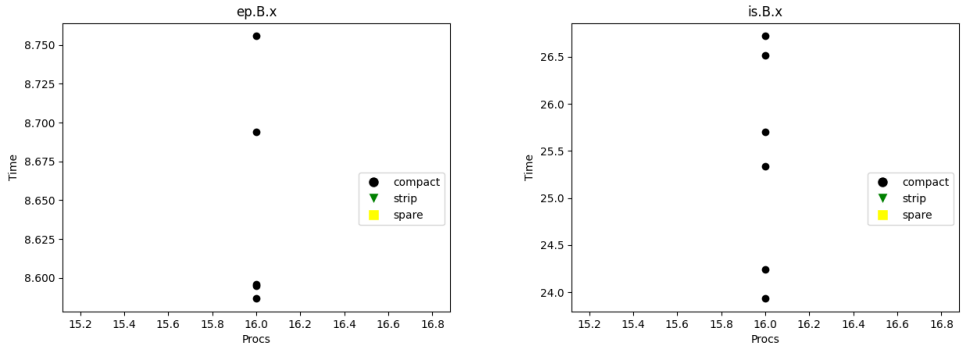
6.3.3.1 All compact

Συνολικός χρόνος εκτέλεσης : 5453.8 δευτερόλεπτα (1.5 ώρα)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 2280 δευτερόλεπτα (38 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 418.4 δευτερόλεπτα (6.9 λεπτά)

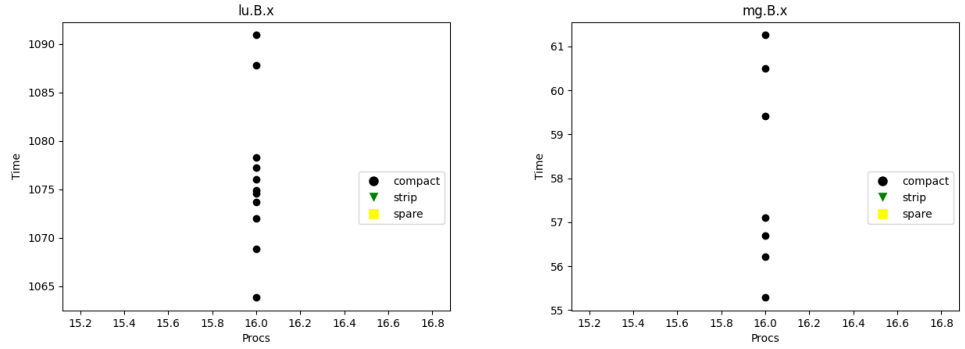
Γραφήματα εργασιών:



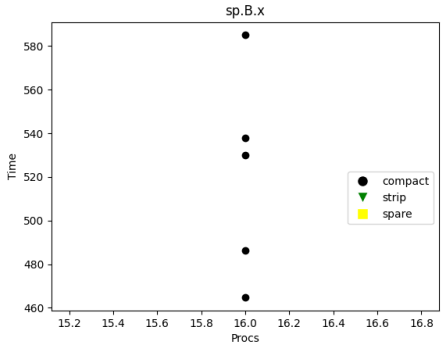
Σχήμα 6.52: *Διαγράμματα bt και cg*



Σχήμα 6.53: *Διαγράμματα ep και is*

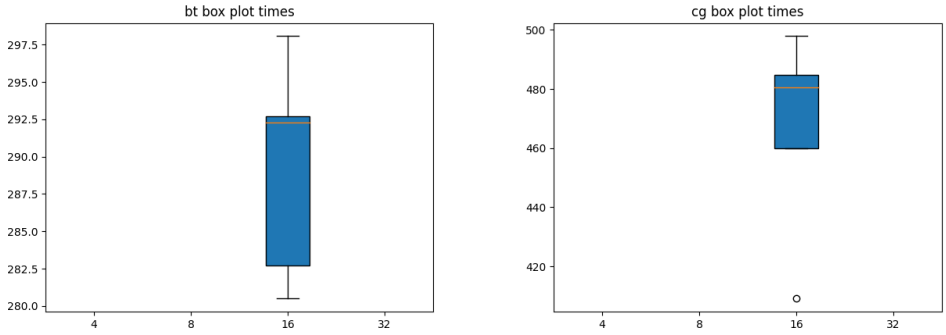


Σχήμα 6.54: *Διαγράμματα lu και mg*

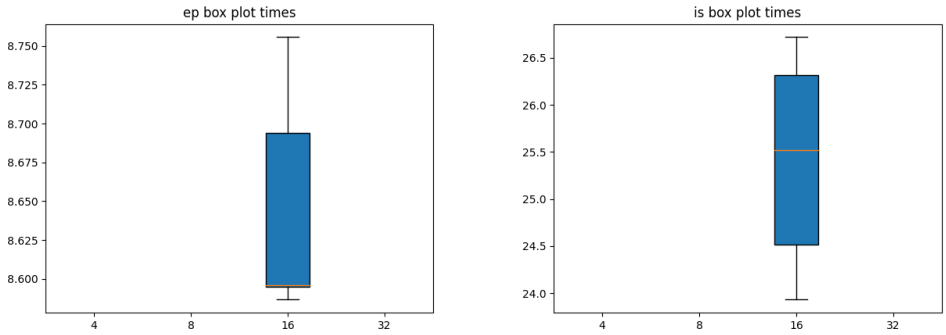


Σχήμα 6.55: *Διάγραμμα sp*

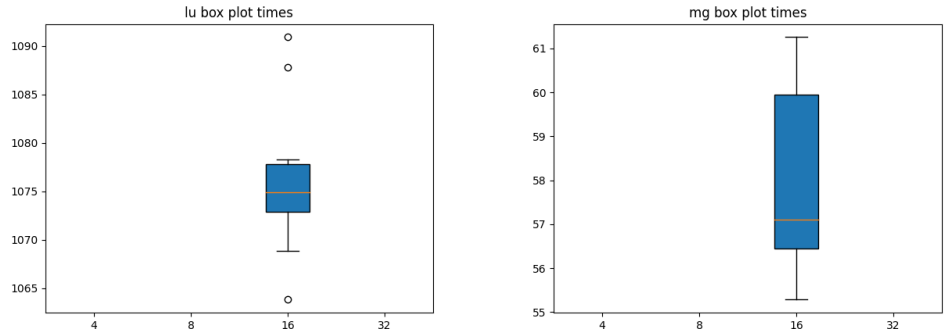
Γραφήματα τύπου Box plot των εργασιών:



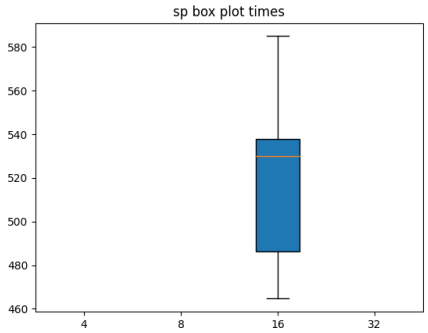
Σχήμα 6.56: *Box plot διαγράμματα bt και cg*



Σχήμα 6.57: *Box plot διαγράμματα ep και is*

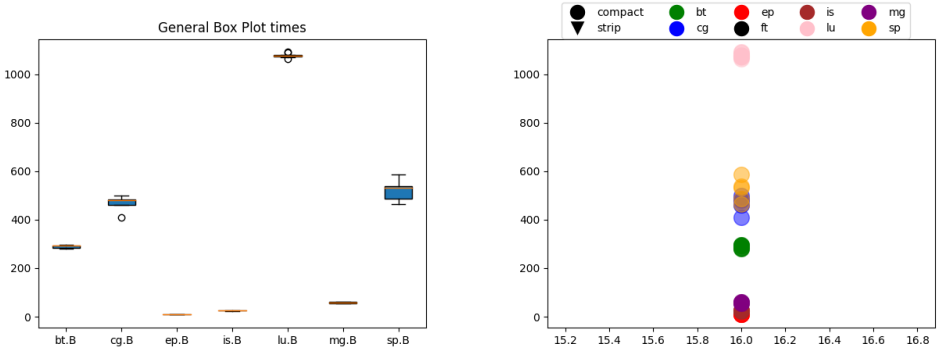


Σχήμα 6.58: *Box plot διαγράμματα lu και mg*



Σχήμα 6.59: *Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα

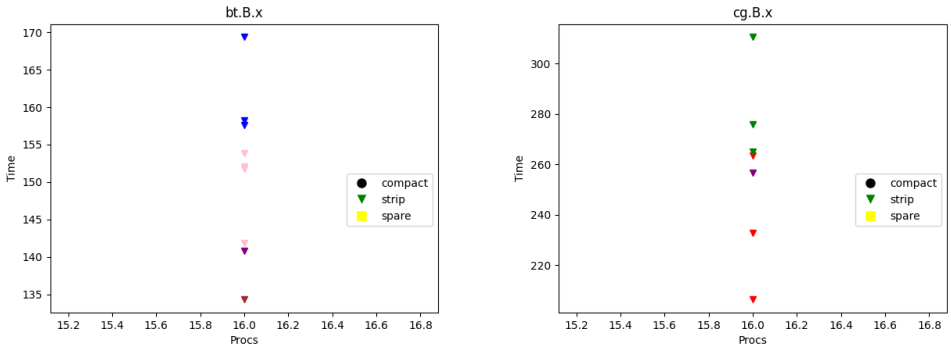


Σχήμα 6.60: *General Box plot και Scatter plot διαγράμματα του πειράματος*

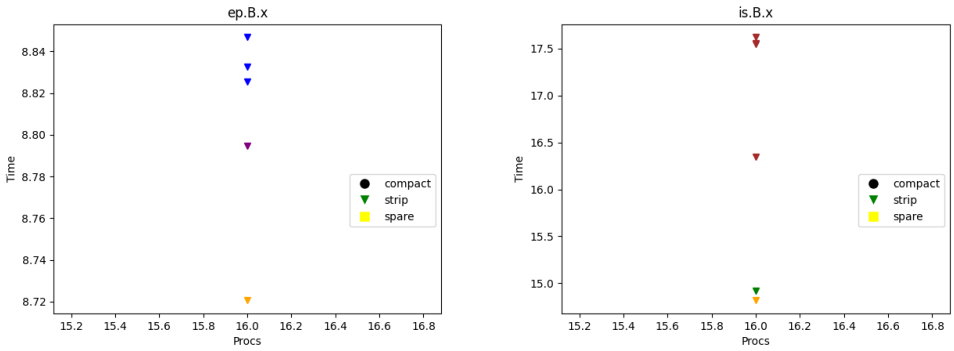
6.3.3.2 All strip

Συνολικός χρόνος εκτέλεσης : 3781.9 δευτερόλεπτα (1 ώρα)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 1640.7 δευτερόλεπτα (27 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 200.4 δευτερόλεπτα (3.3 λεπτά)

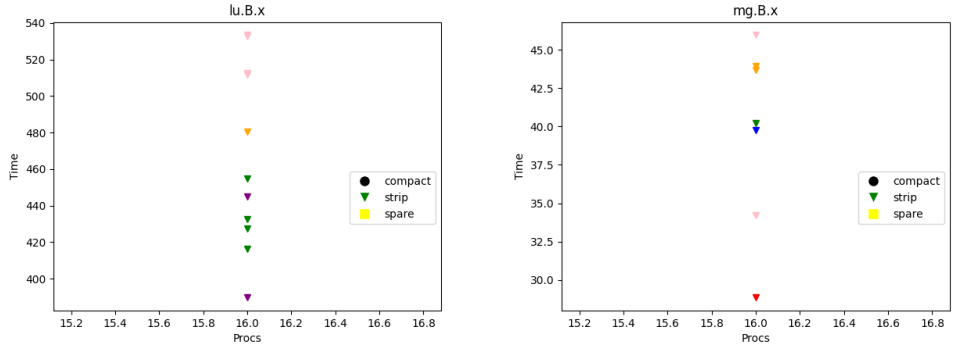
Γραφήματα εργασιών:



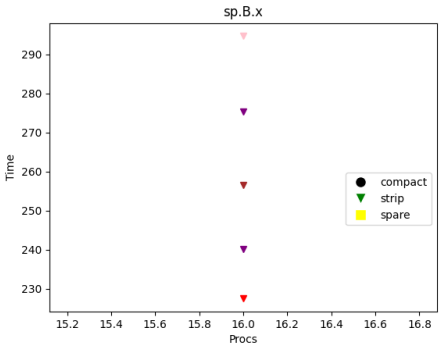
Σχήμα 6.61: *Διαγράμματα bt και cg*



Σχήμα 6.62: *Διαγράμματα ep και is*

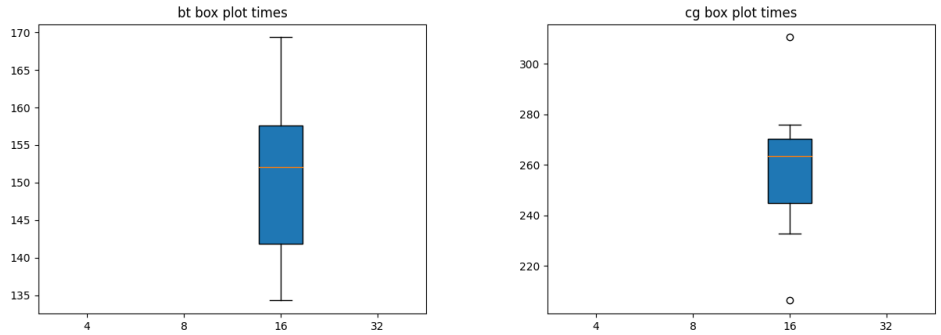


Σχήμα 6.63: *Διαγράμματα lu και mg*

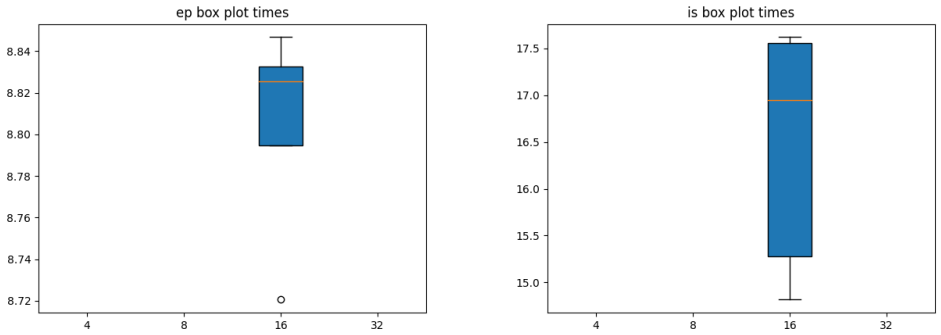


Σχήμα 6.64: *Διάγραμμα sp*

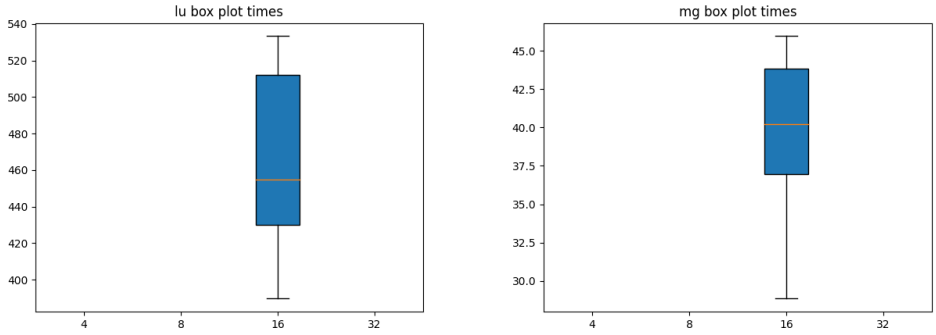
Γραφήματα τύπου Box plot των εργασιών:



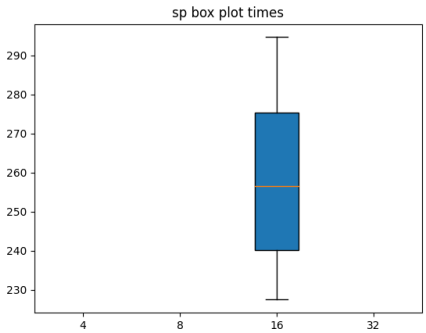
Σχήμα 6.65: *Box plot διαγράμματα bt και cg*



Σχήμα 6.66: *Box plot διαγράμματα ep και is*

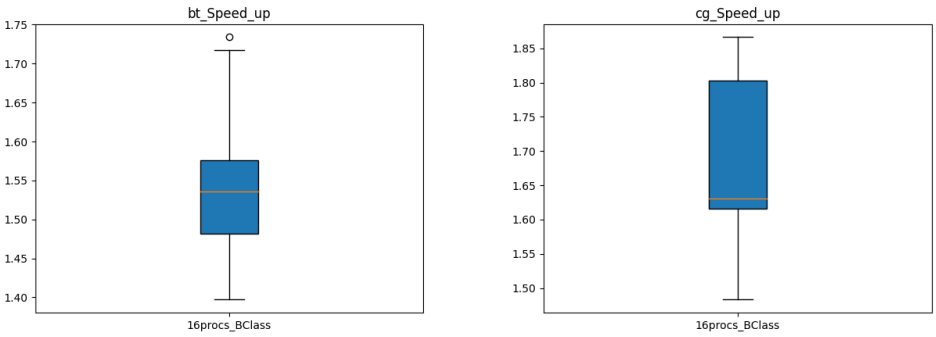


Σχήμα 6.67: *Box plot διαγράμματα lu και mg*

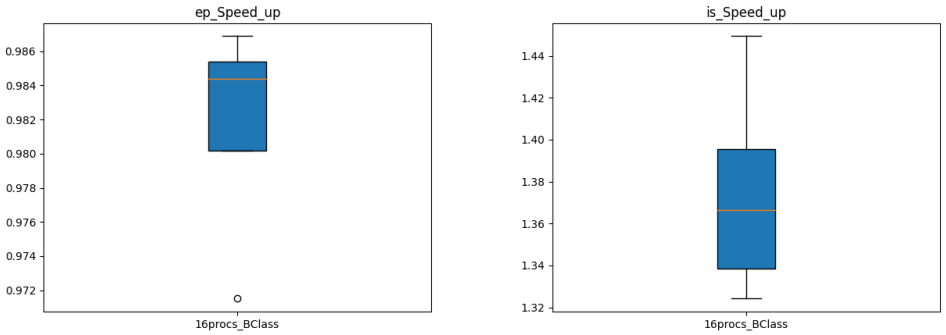


Σχήμα 6.68: *Box plot διάγραμμα sp*

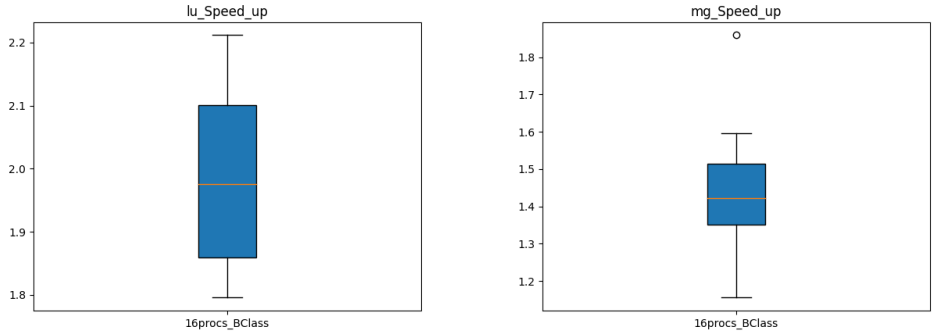
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



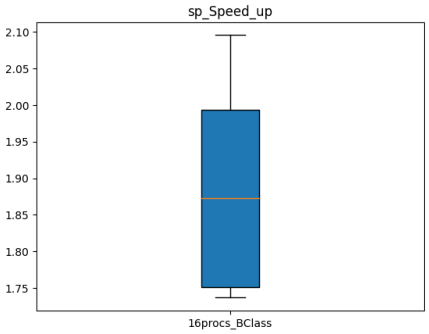
Σχήμα 6.69: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.70: *Speed-up Box plot διαγράμματα ep και is*

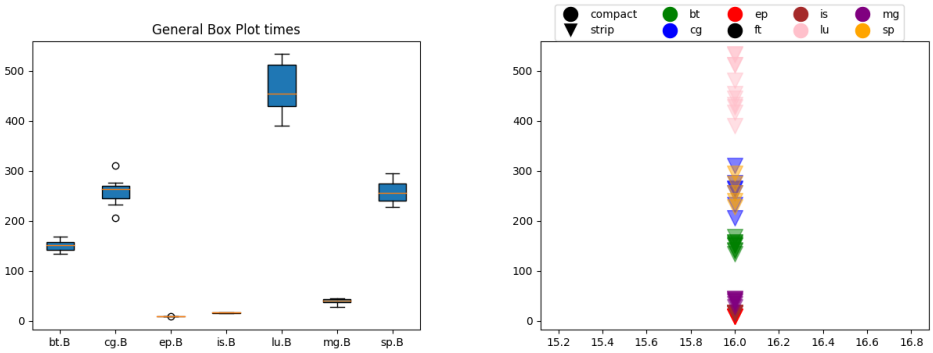


Σχήμα 6.71: *Speed-up Box plot διαγράμματα lu και mg*

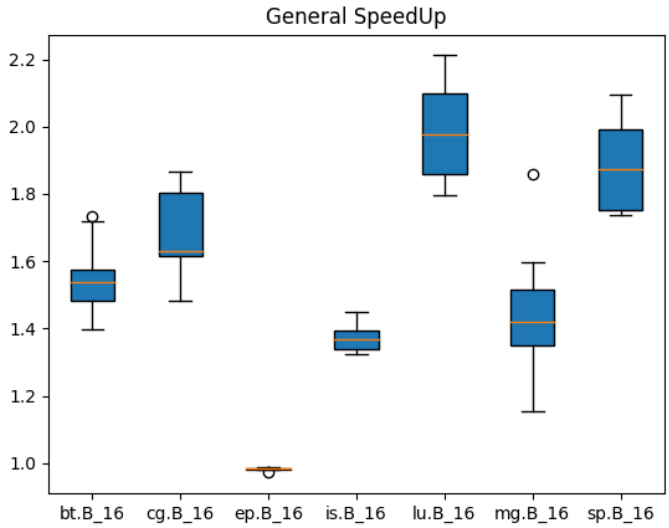


Σχήμα 6.72: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.73: *General Box plot και Scatter plot διαγράμματα του πειράματος*

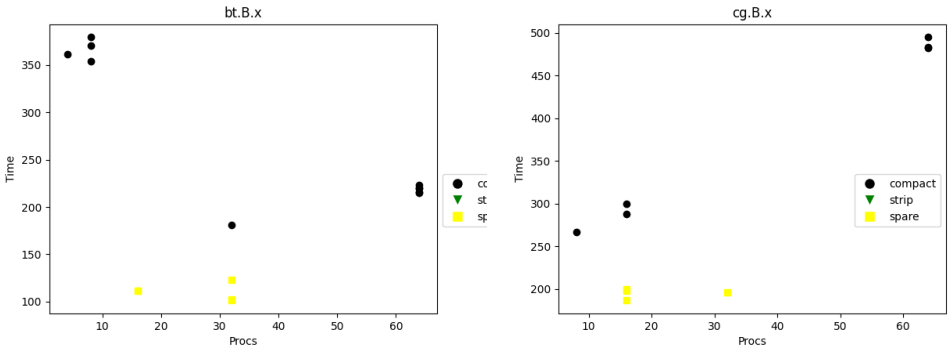


Σχήμα 6.74: *Γενικό Speed-up plot διάγραμμα*

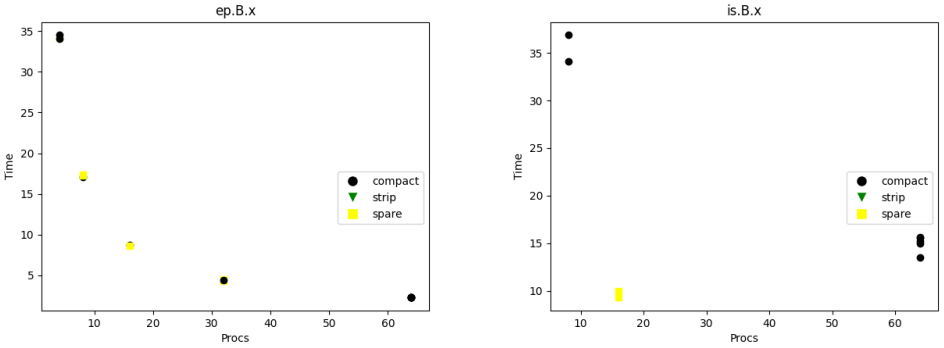
6.3.4 100job random

Συνολικός χρόνος εκτέλεσης : 13990.4 δευτερόλεπτα (3.8 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 6649.3 δευτερόλεπτα (1.8 ώρες)
Μέσος χρόνος εκτέλεσης μιας εργασία: 199.68 δευτερόλεπτα (3.3 λεπτά)

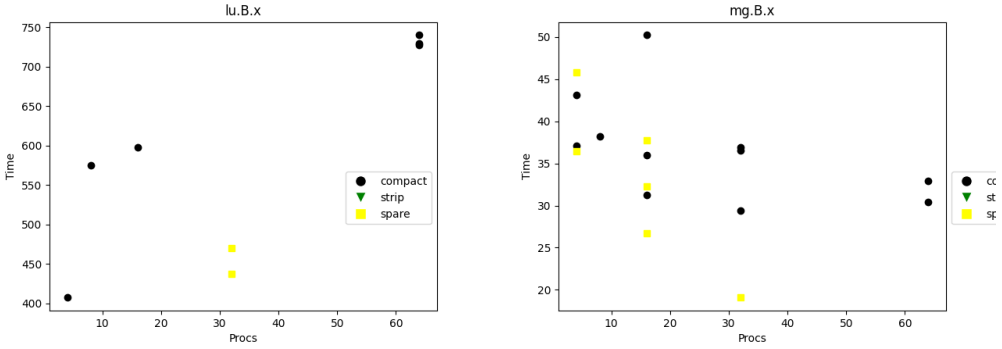
Γραφήματα εργασιών:



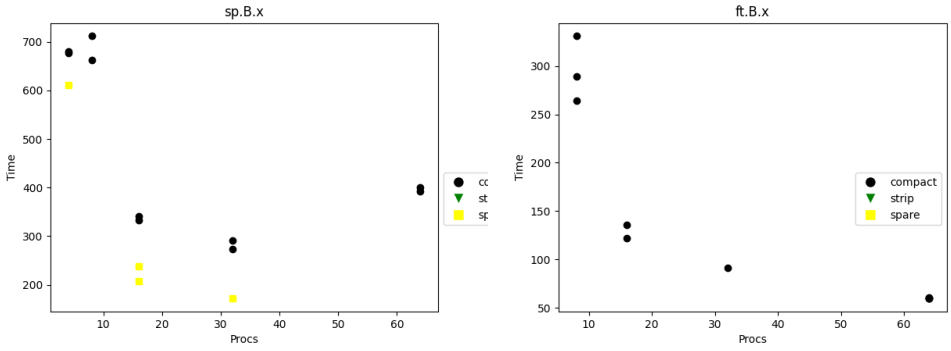
Σχήμα 6.75: *Διαγράμματα bt και cg*



Σχήμα 6.76: *Διαγράμματα ep και is*

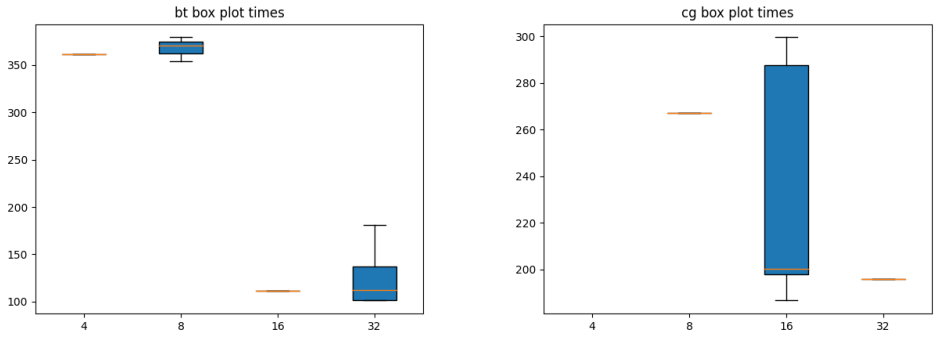


Σχήμα 6.77: *Διαγράμματα lu και mg*

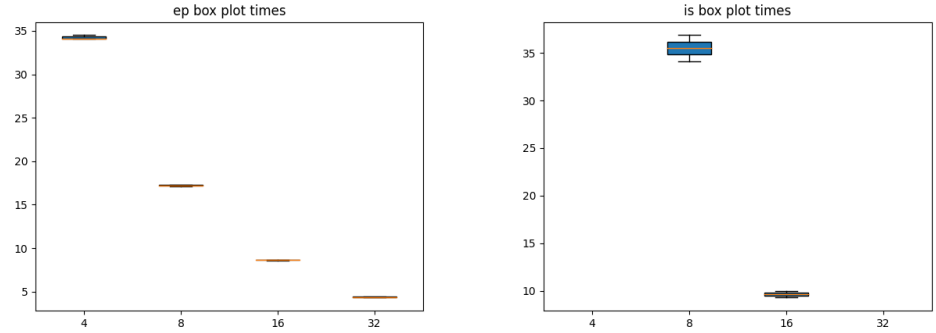


Σχήμα 6.78: *Διαγράμματα sp και ft*

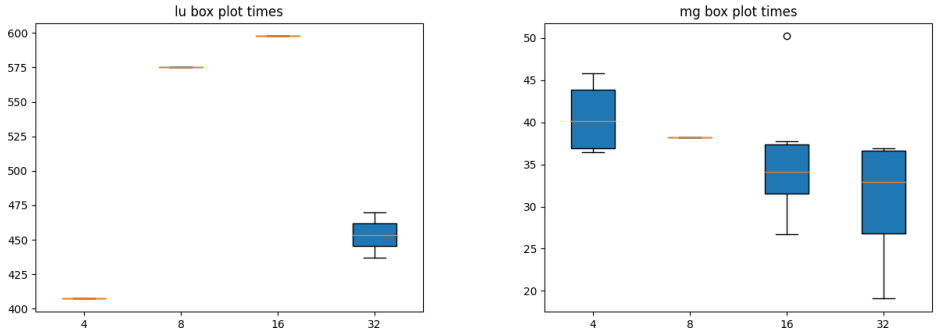
Γραφήματα τύπου Box plot των εργασιών:



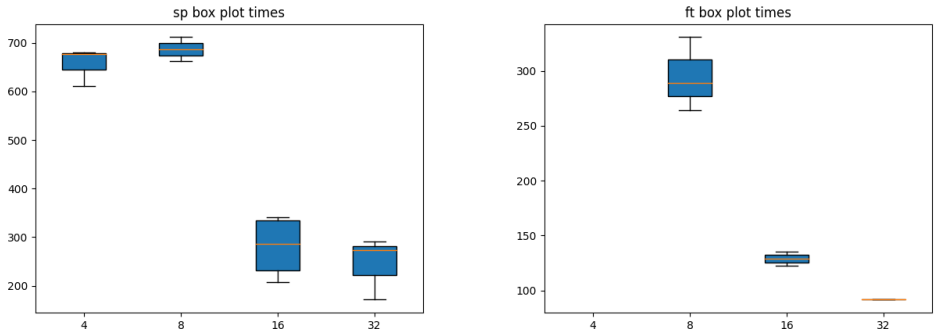
Σχήμα 6.79: *Box plot διαγράμματα bt και cg*



Σχήμα 6.80: *Box plot διαγράμματα ep και is*

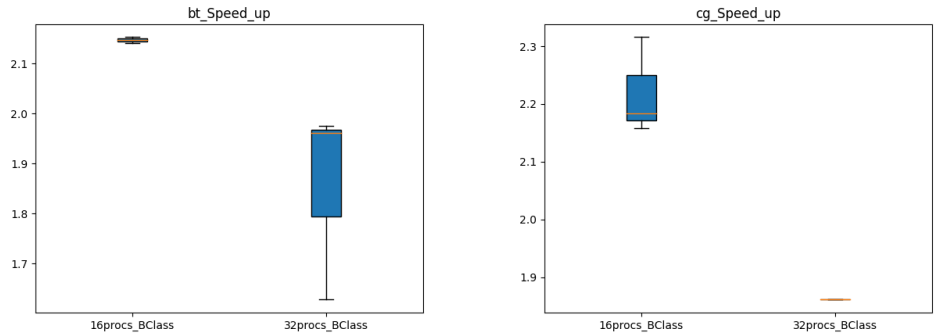


Σχήμα 6.81: *Box plot διαγράμματα lu και mg*

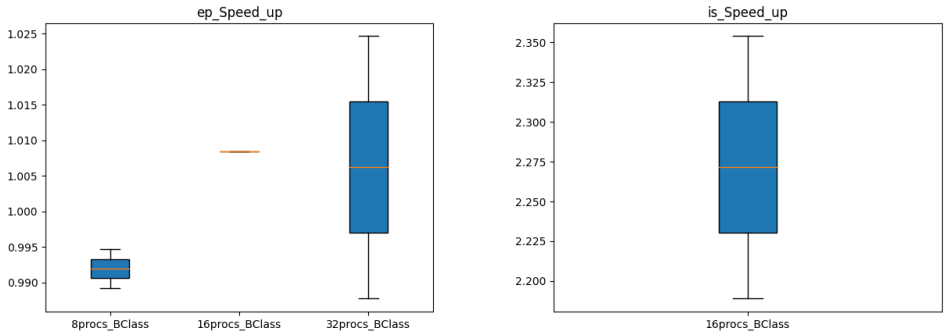


Σχήμα 6.82: *Box plot διαγράμματα sp και ft*

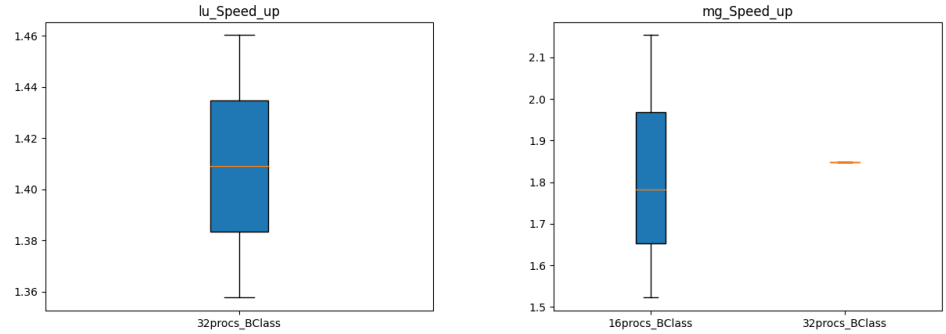
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



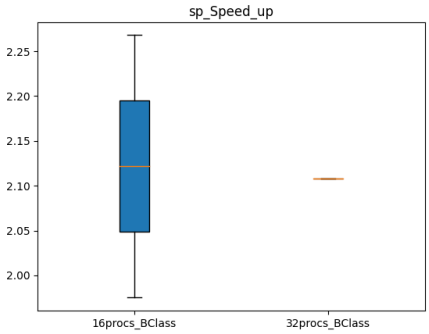
Σχήμα 6.83: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.84: *Speed-up Box plot διαγράμματα ep και is*

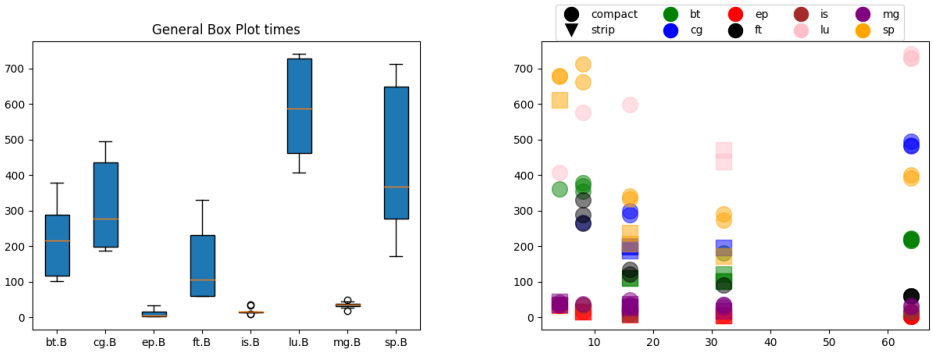


Σχήμα 6.85: *Speed-up Box plot διαγράμματα lu και mg*

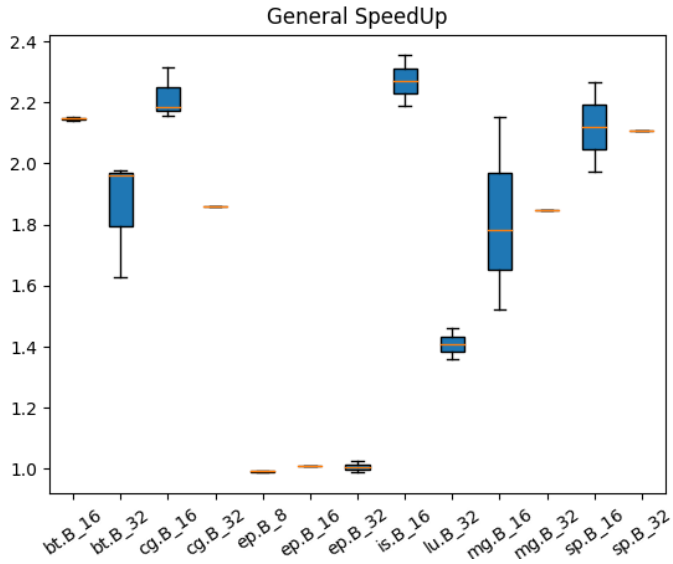


Σχήμα 6.86: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.87: *General Box plot και Scatter plot διαγράμματα του πειράματος*

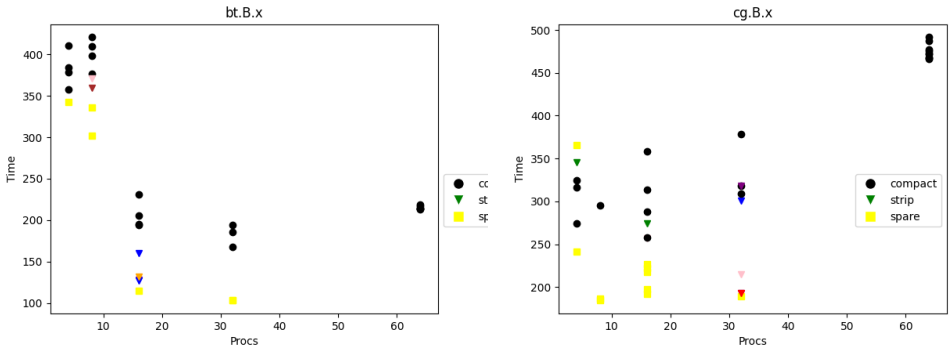


Σχήμα 6.88: *Γενικό Speed-up plot διάγραμμα*

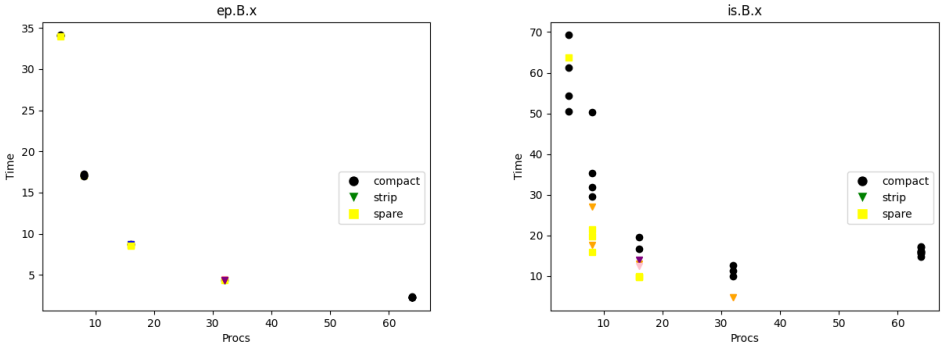
6.3.5 250job random

Συνολικός χρόνος εκτέλεσης : 38524.5 δευτερόλεπτα (10.7 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 19720.23 δευτερόλεπτα (5.4 ώρες)
Μέσος χρόνος εκτέλεσης μιας εργασία: 239.47 δευτερόλεπτα (4 λεπτά)

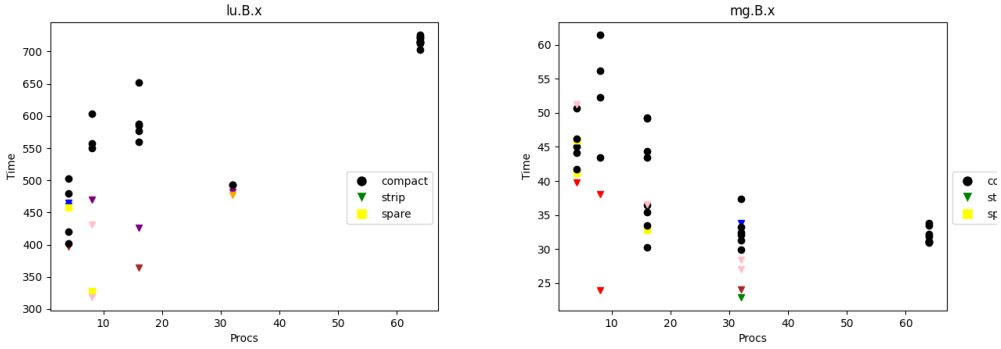
Γραφήματα εργασιών:



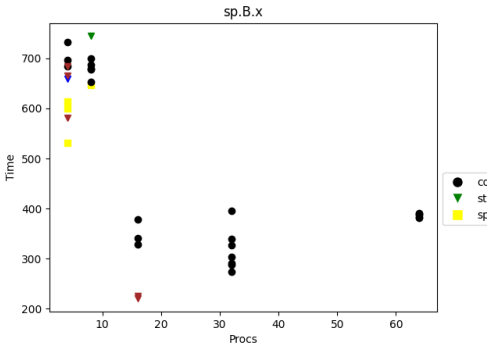
Σχήμα 6.89: *Διαγράμματα bt και cg*



Σχήμα 6.90: *Διαγράμματα ep και is*

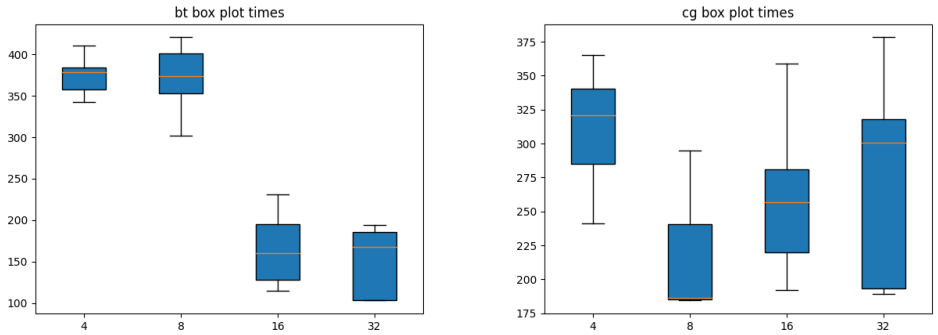


Σχήμα 6.91: *Διαγράμματα lu και mg*

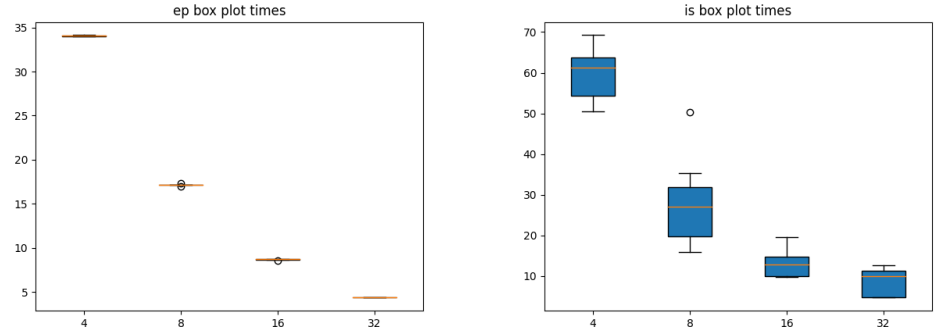


Σχήμα 6.92: *Διάγραμμα sp*

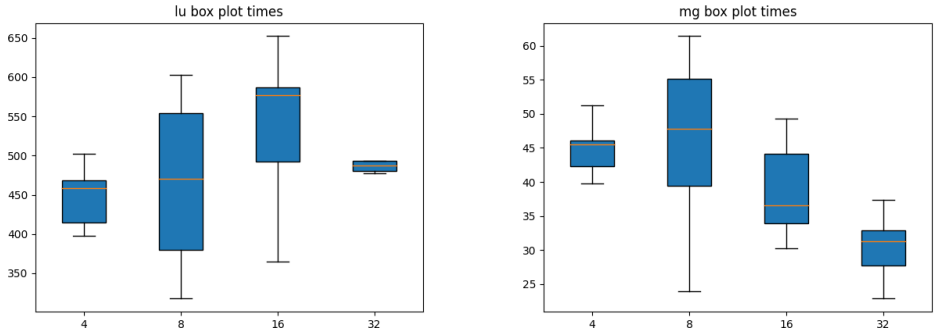
Γραφήματα τύπου Box plot των εργασιών:



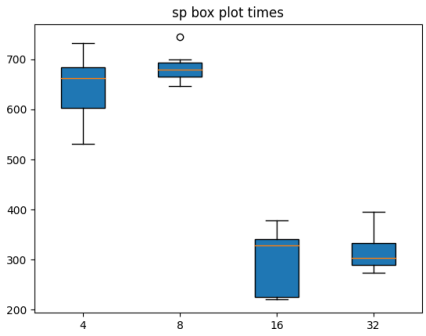
Σχήμα 6.93: *Box plot διαγράμματα bt και cg*



Σχήμα 6.94: *Box plot διαγράμματα ep και is*

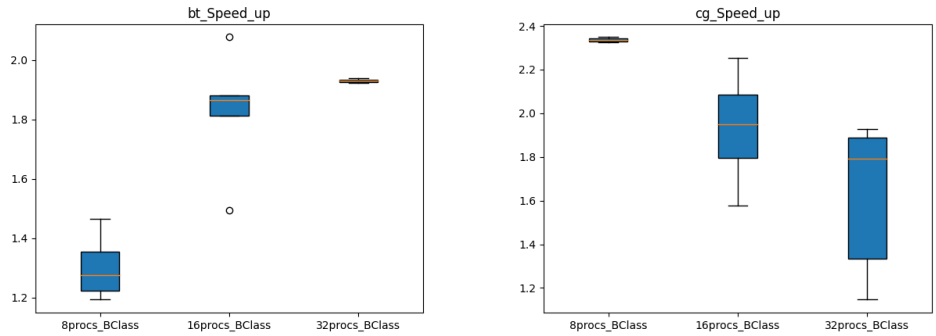


Σχήμα 6.95: *Box plot διαγράμματα lu και mg*

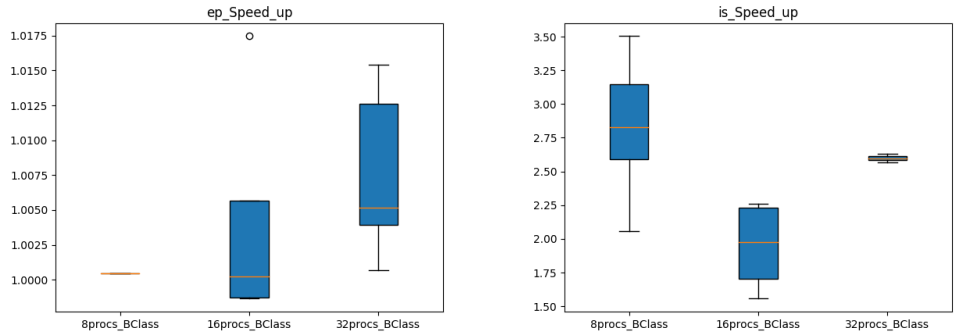


Σχήμα 6.96: *Box plot διάγραμμα sp*

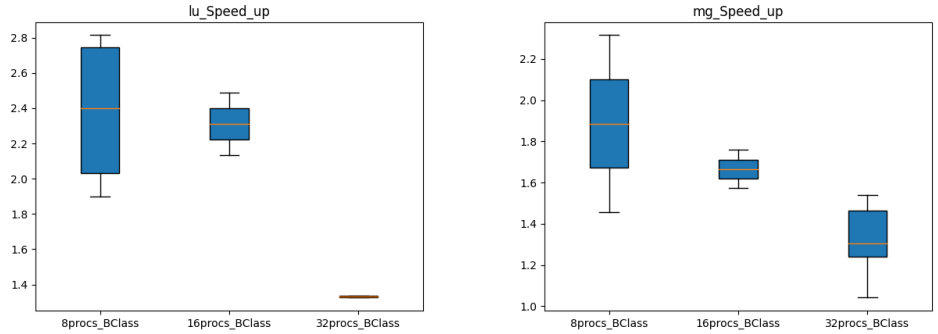
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



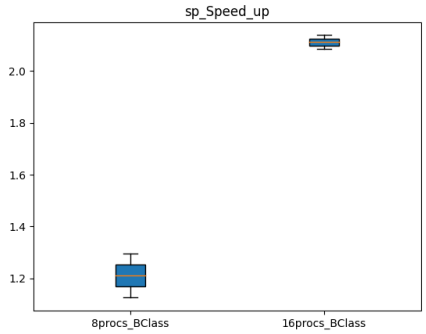
Σχήμα 6.97: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.98: *Speed-up Box plot διαγράμματα ep και is*

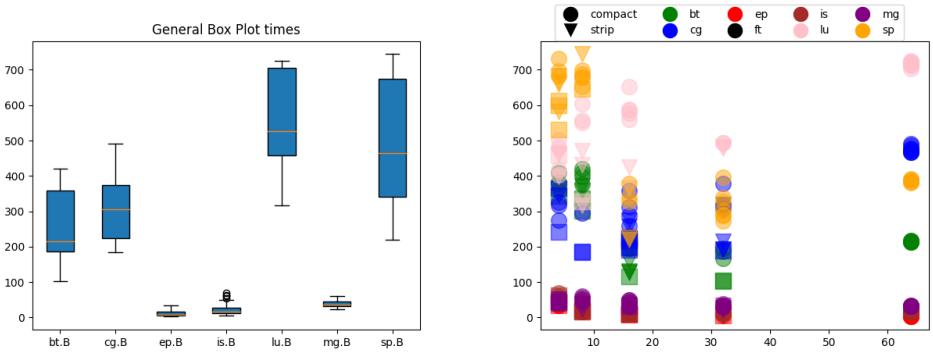


Σχήμα 6.99: *Speed-up Box plot διαγράμματα lu και mg*

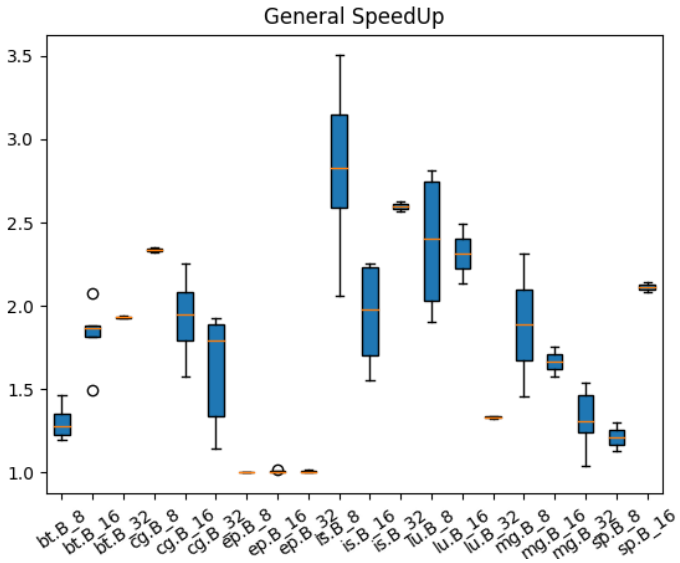


Σχήμα 6.100: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.101: *General Box plot και Scatter plot διαγράμματα του πειράματος*



Σχήμα 6.102: *Γενικό Speed-up plot διάγραμμα*

6.4 Σχόλια

Από τα πειράματα και τα γραφήματα του προηγούμενου κεφαλαίου μπορούμε να βγάλουμε αρκετά συμπεράσματα για την συμπεριφορά των εργασιών που εκτελέσαμε. Αρχικά, μία από τις πρώτες παρατηρήσεις που μπορεί να κάνει κανείς είναι ο χρόνος των πειραμάτων όταν εκείνα εκτελούνταν με όλες τις εργασίες τους να ήταν compact και όταν ήταν όλες strip. Δεδομένου ότι τα data set ήταν ίδια στα πειράματα των τριάντα, σαράντα και πενήντα εργασιών, βλέπουμε πως τα πειράματα με πολιτική strip είχαν καλύτερο χρόνο απο εκείνον της πολιτικής compact (30 εργασίες ίδιος χρόνος, 40 εργασίες 10 λεπτα γρηγορότερα, 50 εργασίες 27 λεπτά γρηγορότερα). Το αποτέλεσμα αυτό μπορεί επίσης να επιβεβαιωθεί από τα speed-up γραφήμματα των εργασιών από τα πειράματα των εκατό και διακοσίων πενήντα εργασίων, όπου φαίνεται ότι οι εργασίες κυμαίνονται σε τιμές μικρότερες του ενός, πράγμα που σημαίνει ότι κάθε μία από αυτές έχει καλύτερο χρόνο κατά μέσω όρο σε πολιτική strip από ότι έχει με τις ίδιες παραμέτρους σε πολιτική compact.

Συνεχίζοντας για τις εργασίες, ξεκινόντας απο την lu(Lower-Upper Gauss-Seidel solver), παρατηρούμε ότι είναι από τις πιο σταθερά αργές εργασίες (∼500-600 δευτερόλεπτα) στα πειράματα μας, αυξάνοντας αισθητά τον χρόνο όταν χρησιμοποιεί εξήντα τέσσερα processes, εμφανίζοντας μεγάλο speed-up σε λιγότερα processes. Όπως φαίνεται λοιπόν, η lu είναι μία εργασία που πραγματοποιεί την καλύτερη της απόδοση με πολιτική strip και τα processes στο εύρος των οκτώ με δεκαέξη, χωρίς να έχει ιδιαίτερη σημασία η εργασία που έτρεξε μαζί.

Η sp(Scalar Penta-diagonal solver) παρουσιάζει παρόμοιους χρόνους με την lu (∼500 δευτερόλεπτα) με την διαφορά ότι αυξάνεται κατά πολύ η απόδοσή της με την αύξηση τον διαθέσιμων πόρων που της δίνεται. Το Speed-up επίσης συμπεριφέρεται με τον ίδιο τρόπο. Για αυτόν τον λόγο, η sp παρουσιάζει μείωση έως και στο μισό τον χρόνο εκτέσης της όταν έχει διαθέσιμα δεκαέξη ή τριανταδύο πυρήνες σε αντίθεση με όταν έχει τέσσερις ή οκτώ.

Οι εργασίες cg(Conjugate Gradient, irregular memory access and communication) και bt(Block Tri-diagonal solver) είναι εργασίες που χρειάζονται παρόμοιο χρόνο (∼ 250 δευτερόλεπτα) και εμφανίζουν βελτίωση στον χρόνο τους με τους περισσότερους πυρήνες σε πολιτική strip (Η cg εμφανίζει την καλύτερη απόδοση σε 32 πυρήνες). Αξίζει να σημειωθεί ότι ενώ η bt δεν εμφανίζει μεγάλη διαφορά στην απόδοση και στο Speed-up ανάλογα με ποιά εργασία τρέχει μαζί, η sp φαίνεται να επιρρεάζεται αρκετά παραπάνω από αυτό το γεγονός όπως φαίνεται στο πείραμα τον διακοσίων πενήντα εργασιών, ο χρόνος της αυξήθηκε μέχρι και 50% ανάλογα με την εργασία που έτρεχε μαζί. Η επιρροή που έχει μια δεύτερη εργασία στην sp περιγράφεται στο Heatmap του παραρτήματος B′.1. Τέλος, μια ακόμα διαφορά των δύο αυτών εργασίων είναι ότι σε πολιτική cmpact, η bt μειώνει τον χρόνο της με την αύξηση των πυρηνών, ενώ η cg αυξάνει τον χρόνο της με την ίδια αύξηση.

Οι εργασίες ep(Embarrassingly Parallel), is(Integer Sort, random memory access) και mg(Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive) είναι και οι τρεις πολύ γρήγορες εργασίες (<100 δευτερόλεπτα) με την ep να είναι λίγο πιο γρήγορη από τις υπόλοιπες δύο και να εμφανίζει τον καλύτερο χρόνο το βέλτιστο σενάριο. Και οι τρεις εμφανίζουν οριακά γραμμική μείωση του χρόνου με την αύξηση των processes με τις mg και ep να έχουν πολύ καλύτερο χρόνο σε πολιτική strip. Η ep φαίνεται να μην έχει μεγάλη επιρροή από την εργασία που τρέχει μαζί της σε πολιτική strip, ενώ στην

mg παρατηρούμε μια μικρή αύξηση του χρόνο σε ένα ¨κακό' σενάριο co-scheduling(+30-40%). Η ep είναι η μόνη εργασία που φαίνεται να έχει σχεδόν την ίδια απόδοση ανεξαρτήτως της πολιτικής που ακολουθεί(Speed-up $\sim= 1$).

Τέλος, από το πείραμα των εκατό εργασίων, η ft(discrete 3D fast Fourier Transform, all-to-all communication) φαίνεται να έχει λίγο καλύτερο μέσω χρόνο εκτέλεσης από τις bt και cg ($\sim$100 δευτερόλεπτα) και να εμφανίζει και αυτή σχετικά γραμμική μείωση του χρόνου εκτέλεσης της ανάλογα με τα processes ($\sim$5 λεπτά με 8 πυρήνες έναντι $<$1 λεπτό με 64 πυρήνες).

Σαν ένα γενικό συμπέρασμα λοιπόν βλέπουμε πως κατά μέσω όρο, ένα data set εργασιών θα έχει καλύτερη απόδοση με τον συνδιασμό strip πολιτικής και αριθμό processes δεκαέξη ή τριανταδύο.

6.5 ARIS

Εκτός από το clustes του CSlab, δόθηκε η δυνατότητα τα πειράματα αυτά να τρέξουν και στο σύστημα ARIS(Advanced Research Information System)[4].Ο ARIS είναι ένας υπερυπολογιστής που περιέρχει 532 υπολογιστικούς κόμβους και κατασκευάστηκε απο το Εθνικό Δίκτυο Υποδομών Τεχνολογίας και Έρευνας (GRNET). Οι κόμβοι είναι αρχιτεκτονικής x86-64, οι οποίοι συνδέονται μεταξύ τους μέσα από ένα δίκτυο Infiniband FDR14 τοπολογίας fat tree. Κάθε μηχάνημα έχει επεξεργαστή 5-2680v2 της Intel, ο οποίος έχει δύο socket και δέκα πυρήνες στο κάθε ένα από αυτά.

Architecture	x86-64
Operating System	Redhat/Centos 6.7
Interconnect	
Technology	Infiniband FDR
Topology	Fat tree
Bandwidth [Gb/s]	56
Storage	
Type	IBM GPFS
Size [PByte]	1
Bandwidth [GB/s]	6
System Software	
Operating system	RedHat/Centos Linux 6.7
Batch system	SLURM
System Management	xCat IBM
Monitoring	Nagios, Ganglia

THIN nodes technical information	
Architecture	x86-64
System	IBM NeXtScale nx360 M4
Total number of nodes	426
Total number of cores	8520
Total amount of RAM [TByte]	27
Total Linpack Performance [TFlop/s]	180
Components	
Processor Type	Ivy Bridge - Intel Xeon E5-2680v2
Nominal Frequency [GHz]	2.8
Processors per Node	2
Cores per Processor	10
Cores per Node	20
Hyperthreading	OFF
Memory	
Memory per Node [GByte]	64

Σχήμα 6.103: *Specifications του συστήματος ARIS[4]*

6.5.1 Πειράματα

Τα πειράματα που τρέξαμε στο σύστημα του ARIS είναι τα εξής:

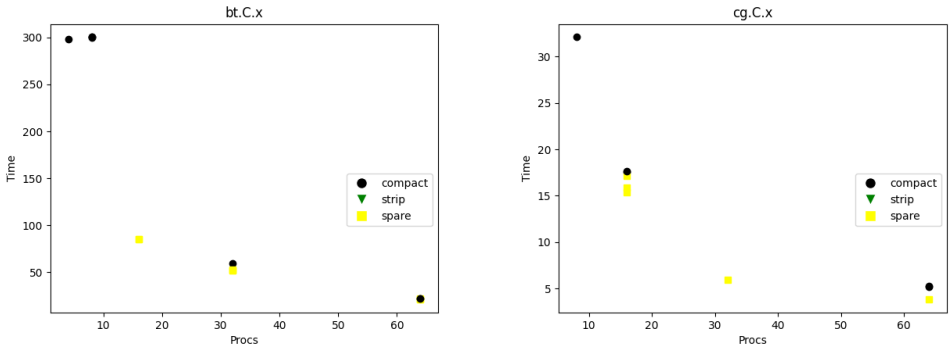
- **Κλάση C**
  - **Ίδιο data set εκατό και διακοσίων πενήντα εργασιών με αλλαγή της κλάσης από B σε C.**
  - **8 Μηχανήματα με 2 socket και 10 πυρήνες το κάθε ένα.**
- **Κλάση D**
  - **Ίδιο data set εκατό και διακοσίων πενήντα εργασιών με αλλαγή της κλάσης από B σε D.**
  - **Μετατροπή των διεργασιών κάθε εργασίας από 8,16,32,64 σε 128,256, 512.**
  - **52 Μηχανήματα με 2 socket και 10 πυρήνες το κάθε ένα.**

6.5.2 Κλάση C

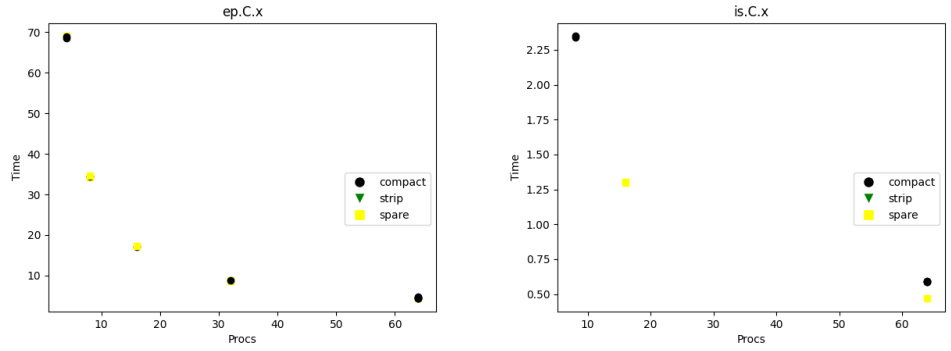
6.5.2.1 100 Εργασίες

Συνολικός χρόνος εκτέλεσης : 1553.7 δευτερόλεπτα (25.9 λεπτά)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 641.61 δευτερόλεπτα (10.6 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 50.08 δευτερόλεπτα (0.8 λεπτά)

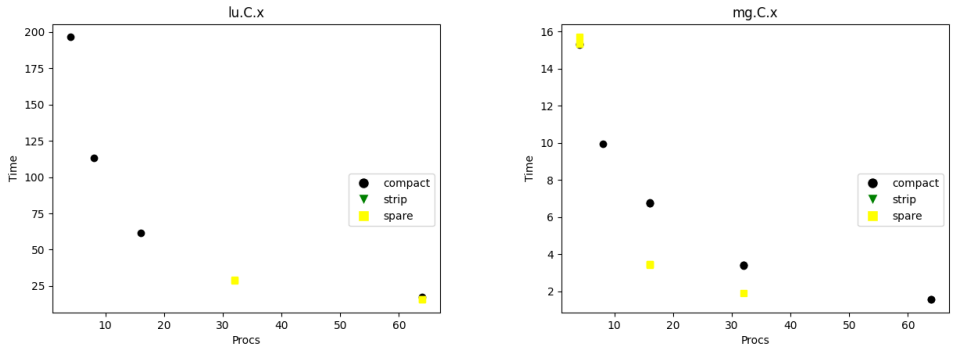
Γραφήματα εργασιών:



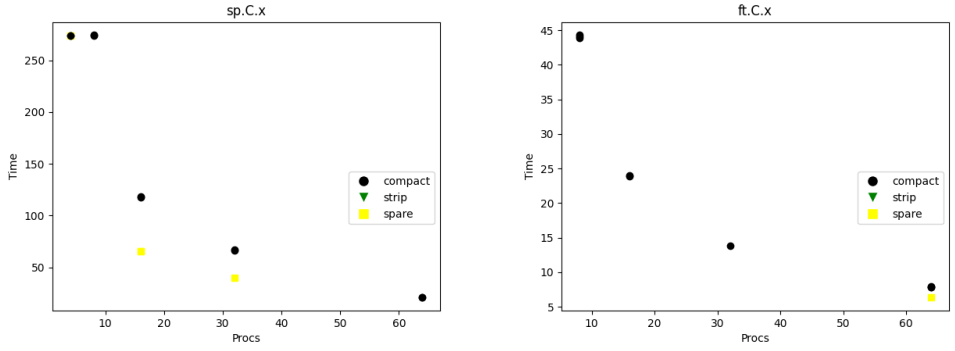
Σχήμα 6.104: *Διαγράμματα bt και cg*



Σχήμα 6.105: *Διαγράμματα ep και is*

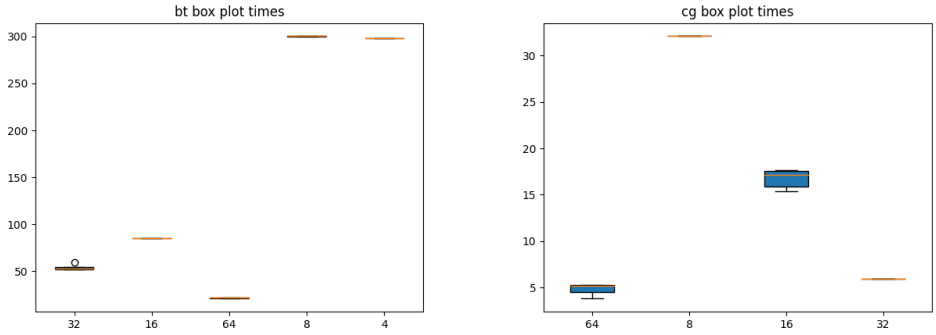


Σχήμα 6.106: *Διαγράμματα lu και mg*

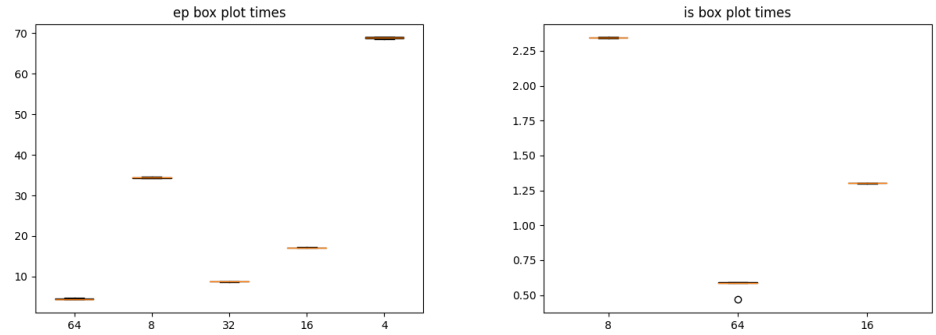


Σχήμα 6.107: *Διαγράμματα sp και ft*

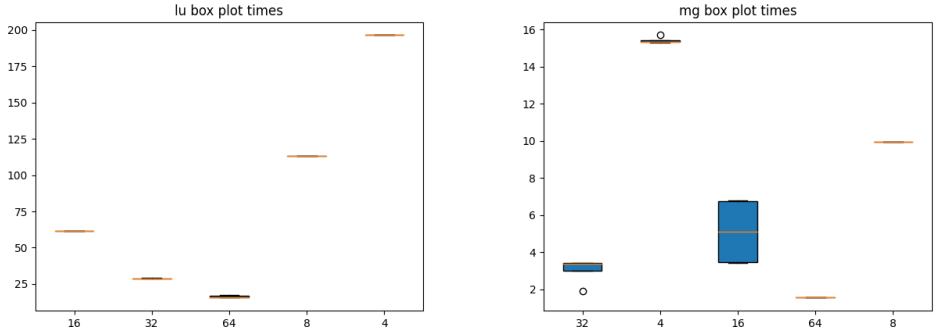
Γραφήματα τύπου Box plot των εργασιών:



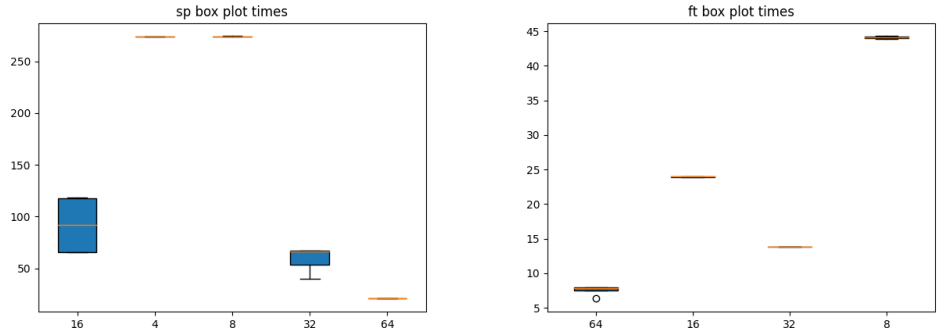
Σχήμα 6.108: *Box plot διαγράμματα bt και cg*



Σχήμα 6.109: *Box plot διαγράμματα ep και is*

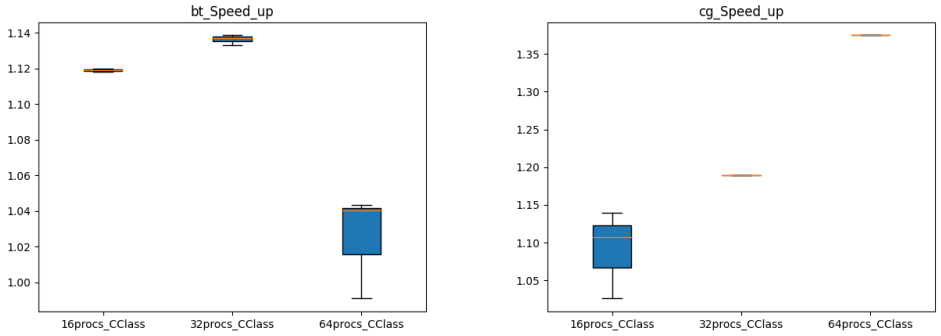


Σχήμα 6.110: *Box plot διαγράμματα lu και mg*

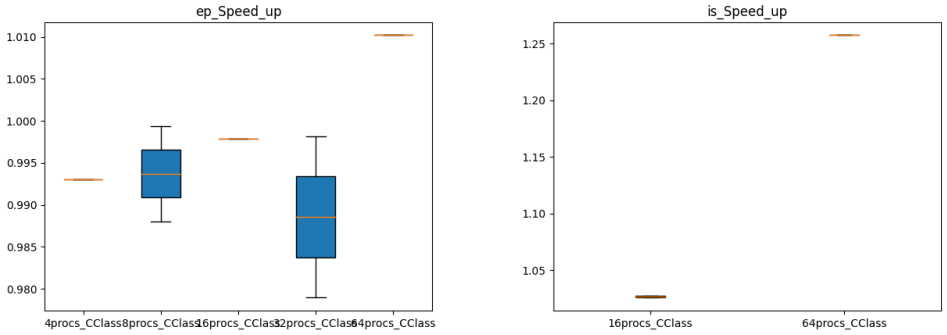


Σχήμα 6.111: *Box plot διαγράμματα sp και ft*

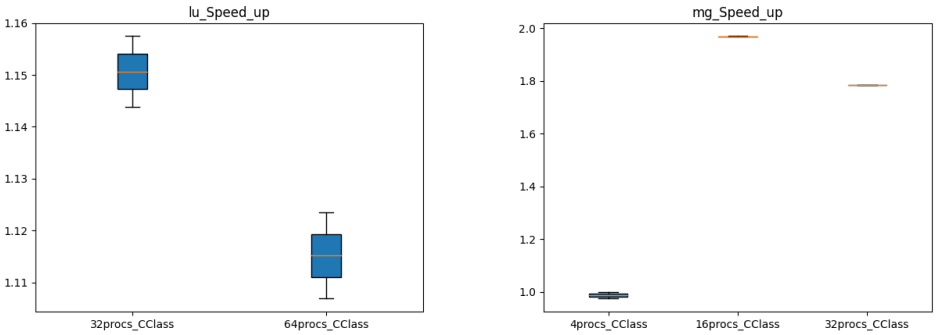
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



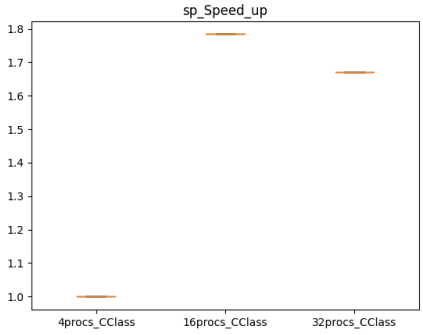
Σχήμα 6.112: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.113: *Speed-up Box plot διαγράμματα ep και is*

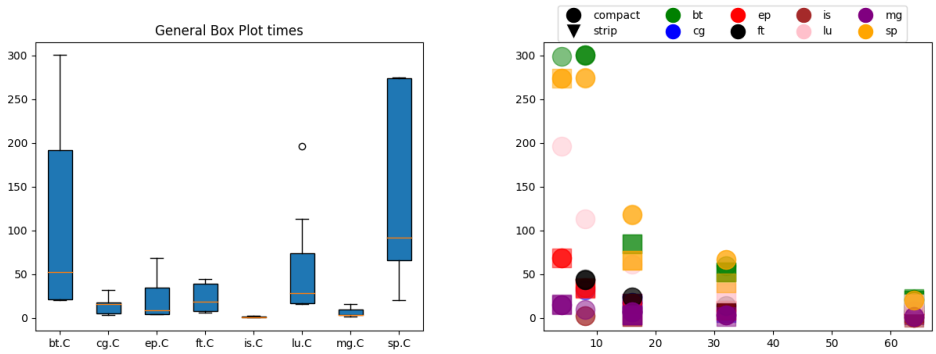


Σχήμα 6.114: *Speed-up Box plot διαγράμματα lu και mg*

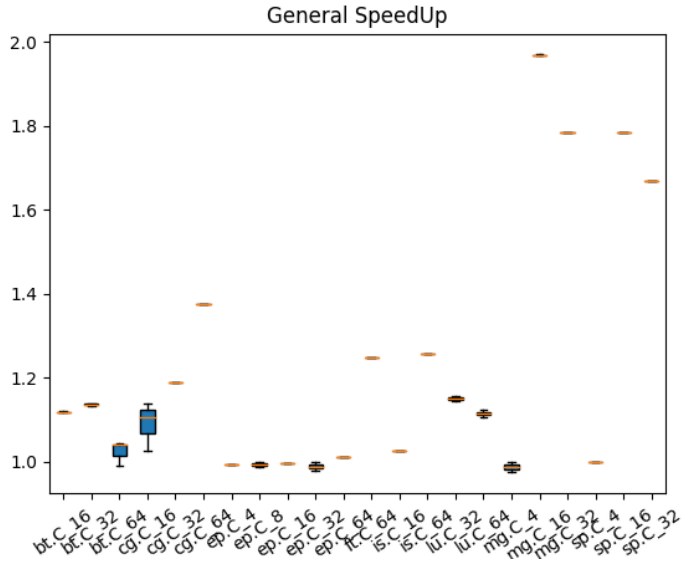


Σχήμα 6.115: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.116: *General Box plot και Scatter plot διαγράμματα του πειράματος*

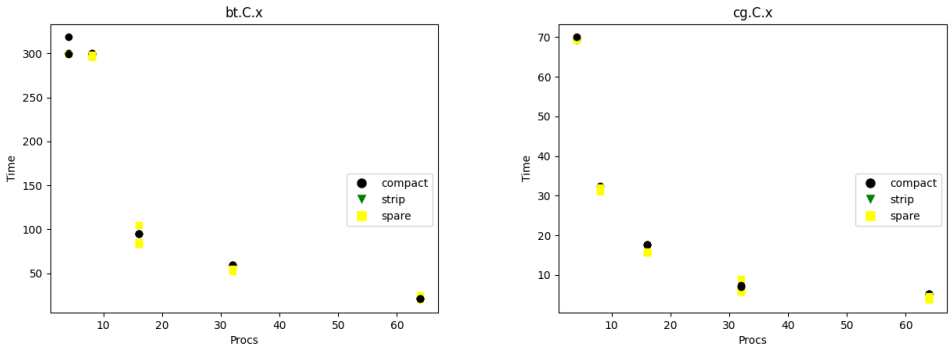


Σχήμα 6.117: *Γενικό Speed-up plot διάγραμμα*

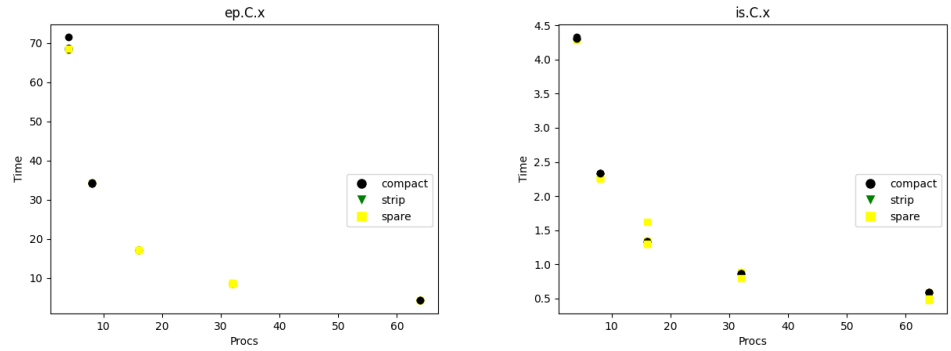
6.5.2.2 250 Εργασίες

Συνολικός χρόνος εκτέλεσης : 4030.6 δευτερόλεπτα (1.1 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 1901.8 δευτερόλεπτα (31 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 63.05 δευτερόλεπτα (1 λεπτό)

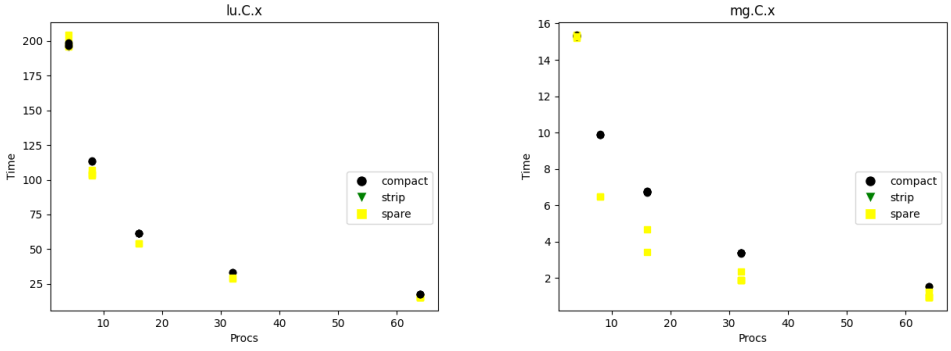
Γραφήματα εργασιών:



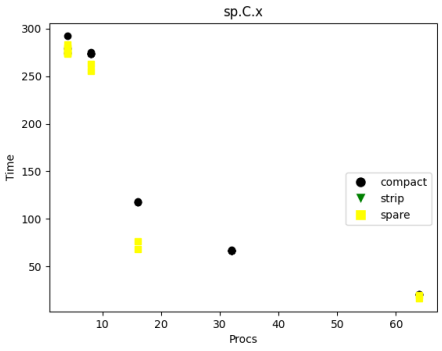
Σχήμα 6.118: *Διαγράμματα bt και cg*



Σχήμα 6.119: *Διαγράμματα ep και is*

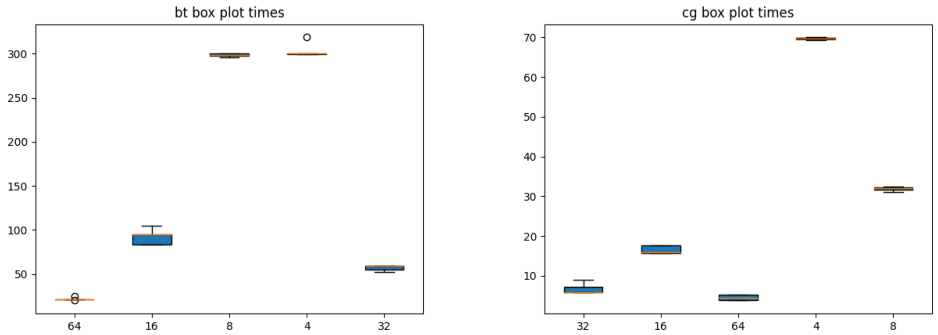


Σχήμα 6.120: *Διαγράμματα lu και mg*

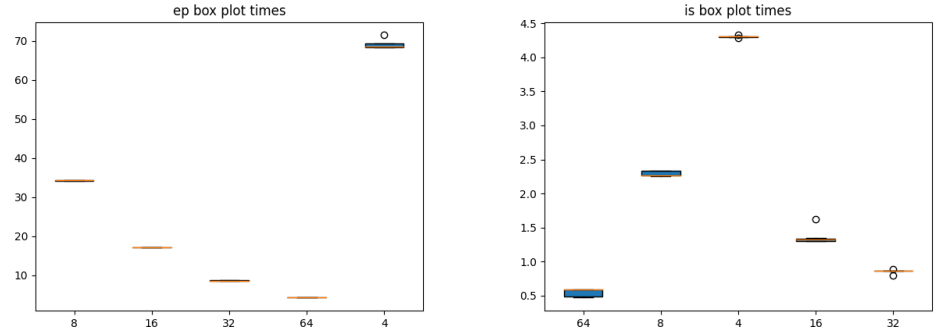


Σχήμα 6.121: *Διάγραμμα sp*

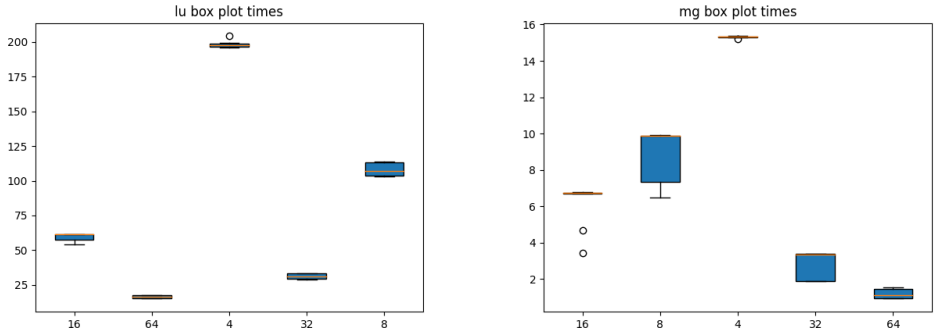
Γραφήματα τύπου Box plot των εργασιών:



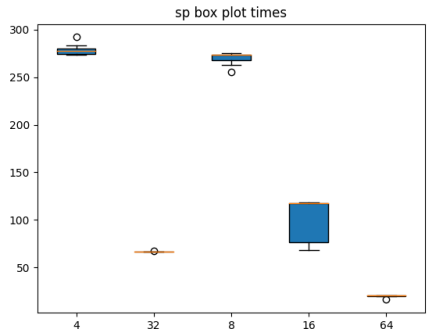
Σχήμα 6.122: *Box plot διαγράμματα bt και cg*



Σχήμα 6.123: *Box plot διαγράμματα ep και is*

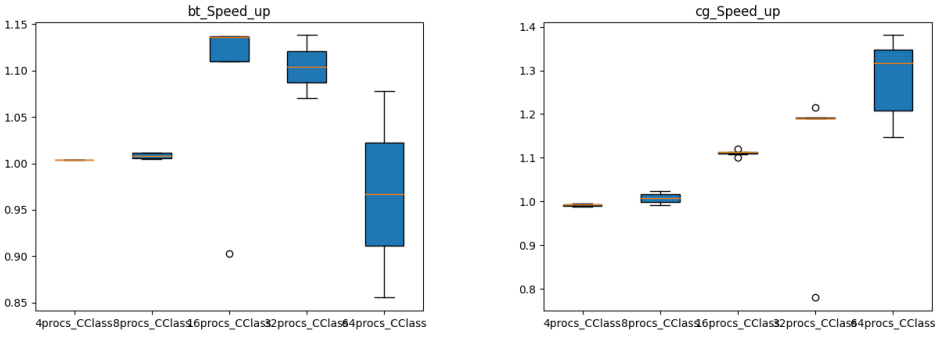


Σχήμα 6.124: *Box plot διαγράμματα lu και mg*

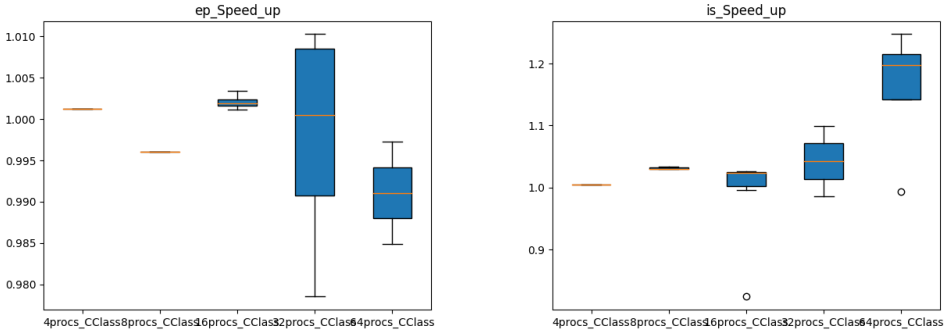


Σχήμα 6.125: *Box plot διάγραμμα sp*

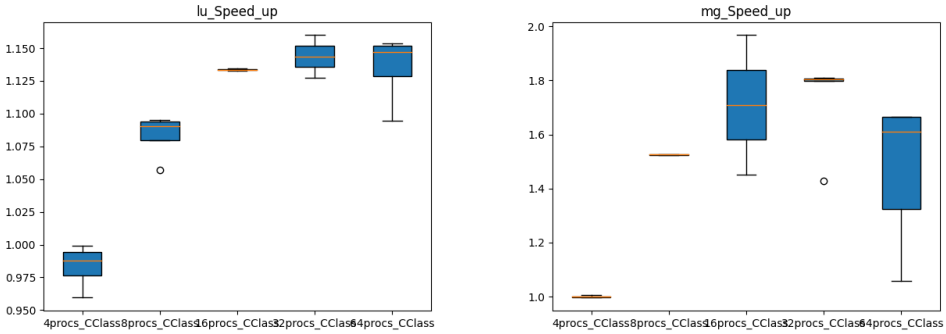
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



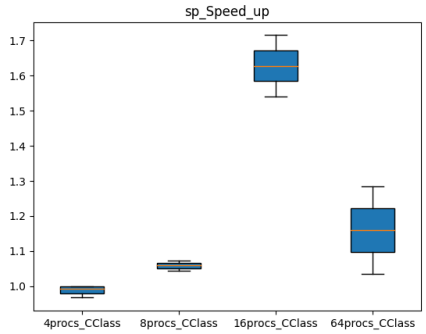
Σχήμα 6.126: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.127: *Speed-up Box plot διαγράμματα ep και is*

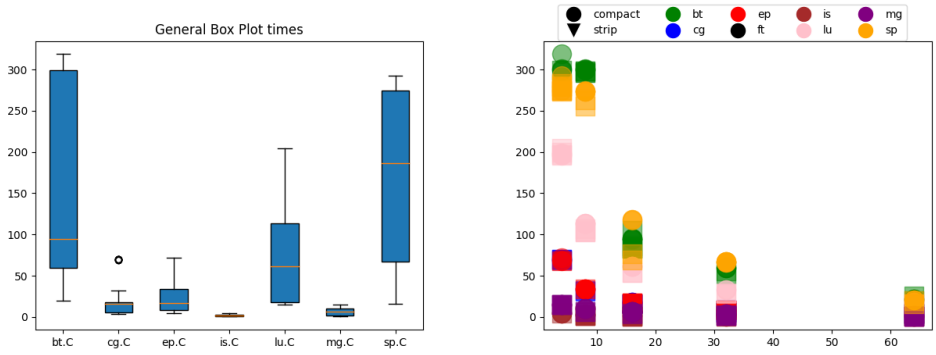


Σχήμα 6.128: *Speed-up Box plot διαγράμματα lu και mg*

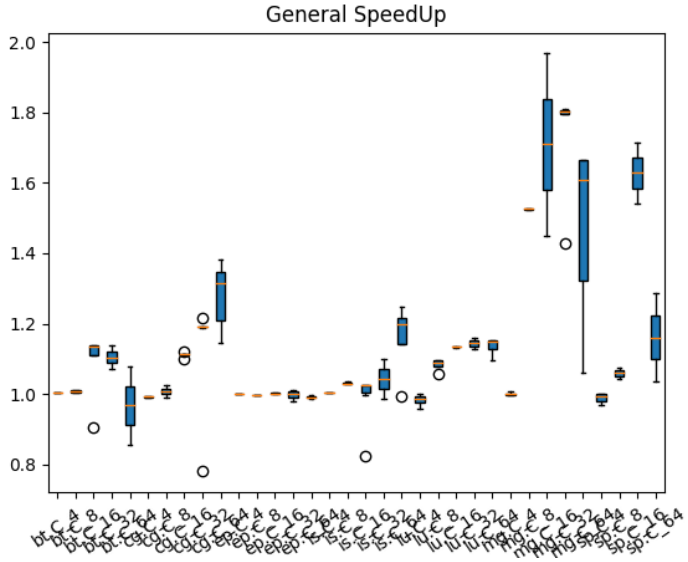


Σχήμα 6.129: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.130: *General Box plot και Scatter plot διαγράμματα του πειράματος*



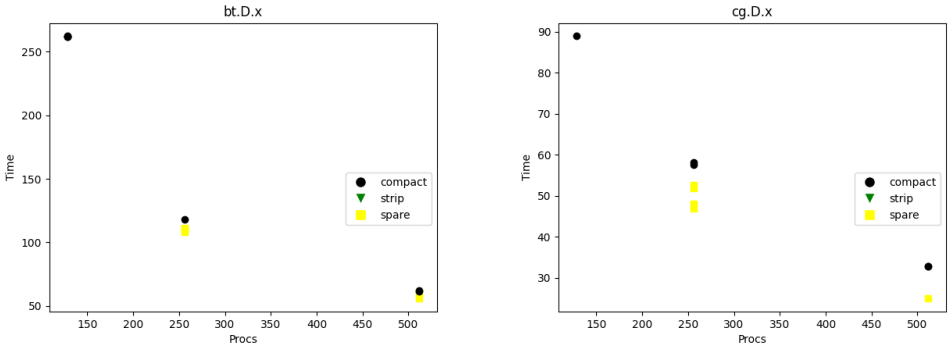
Σχήμα 6.131: *Γενικό Speed-up plot διάγραμμα*

6.5.3 Κλάση D

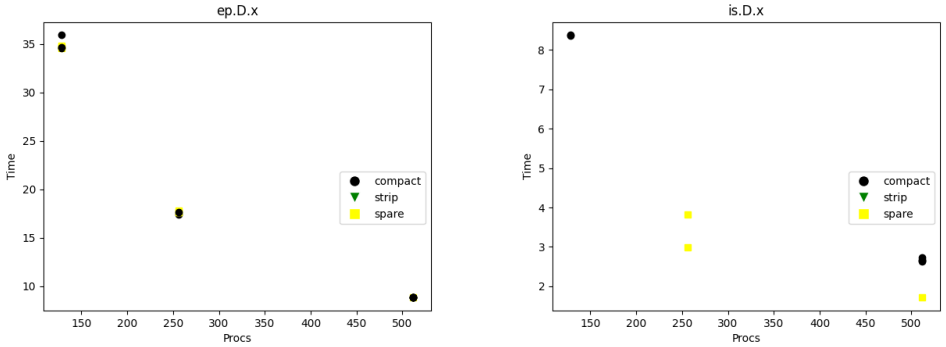
6.5.3.1 100 Εργασίες

Συνολικός χρόνος εκτέλεσης : 2909.6 δευτερόλεπτα (48 λεπτά)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 1351.91 δευτερόλεπτα (22 λεπτά)
Μέσος χρόνος εκτέλεσης μιας εργασία: 66.24 δευτερόλεπτα (1.1 λεπτά)

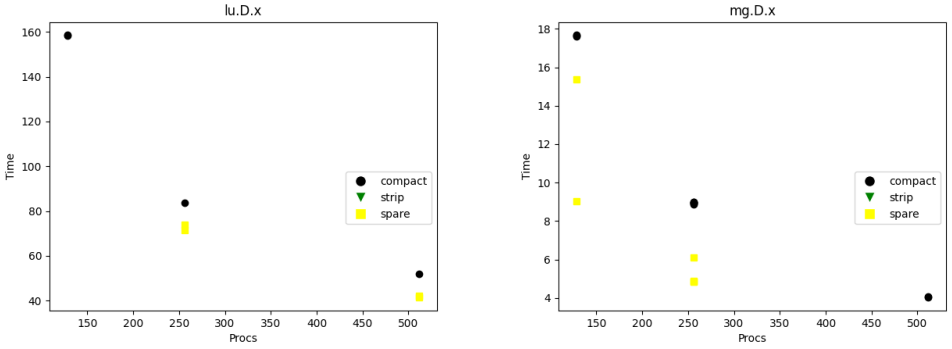
Γραφήματα εργασιών:



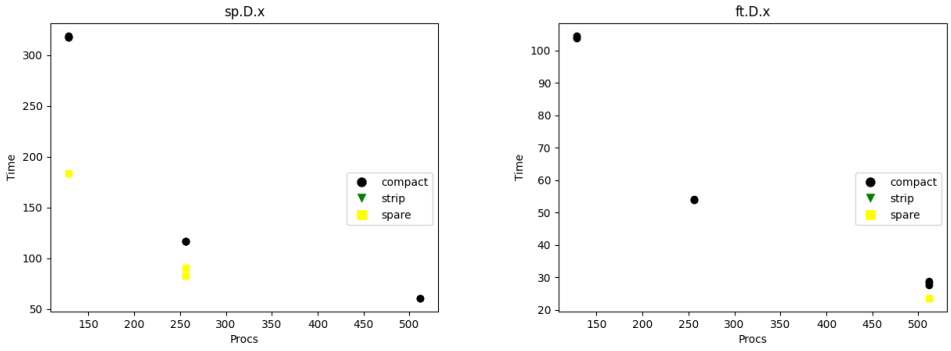
Σχήμα 6.132: *Διαγράμματα bt και cg*



Σχήμα 6.133: *Διαγράμματα ep και is*

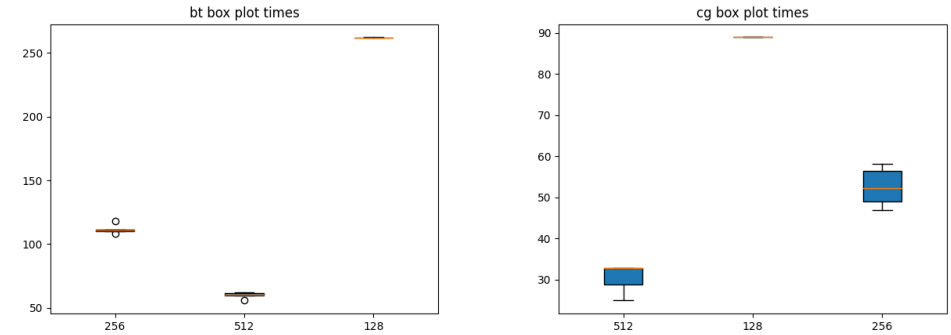


Σχήμα 6.134: *Διαγράμματα lu και mg*

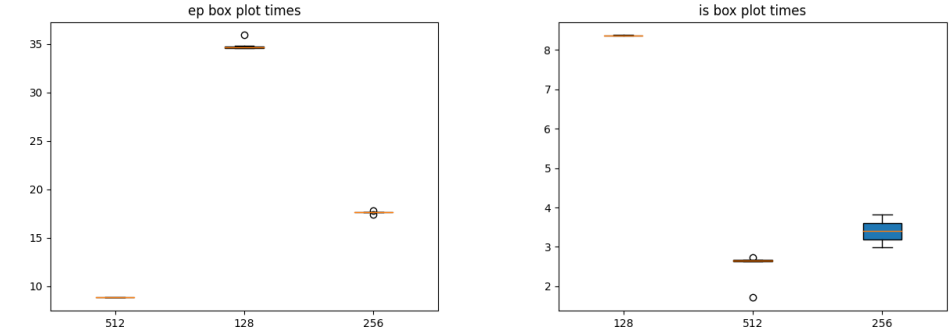


Σχήμα 6.135: *Διαγράμματα sp και ft*

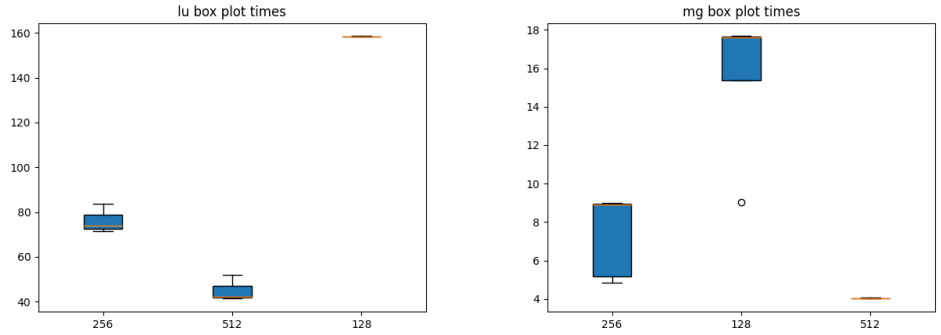
Γραφήματα τύπου Box plot των εργασιών:



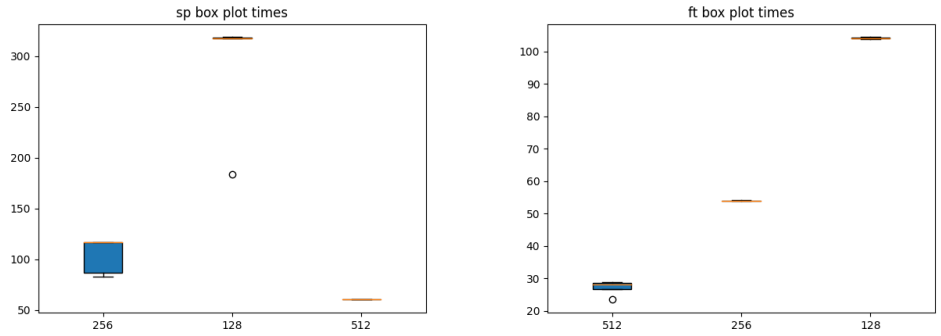
Σχήμα 6.136: *Box plot διαγράμματα bt και cg*



Σχήμα 6.137: *Box plot διαγράμματα ep και is*

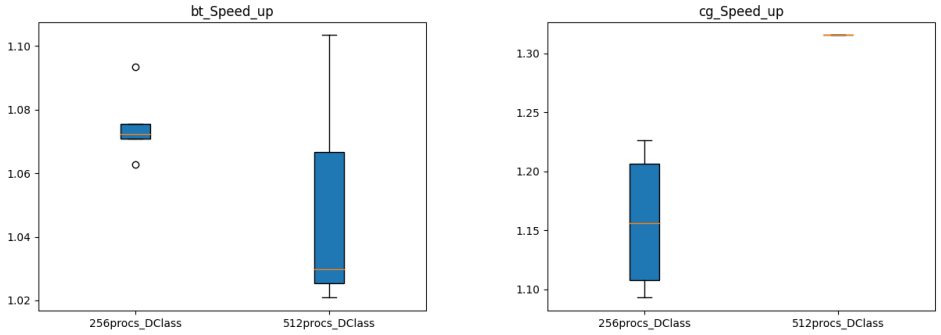


Σχήμα 6.138: *Box plot διαγράμματα lu και mg*

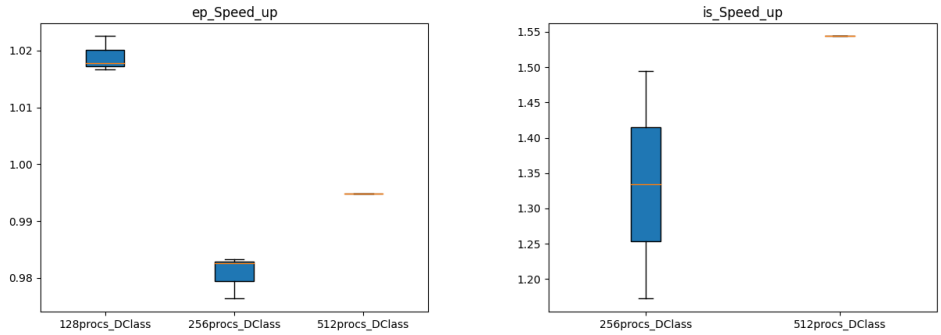


Σχήμα 6.139: *Box plot διαγράμματα sp και ft*

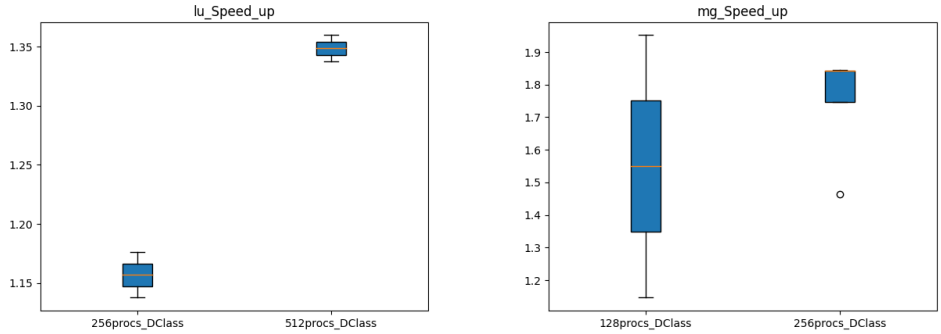
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



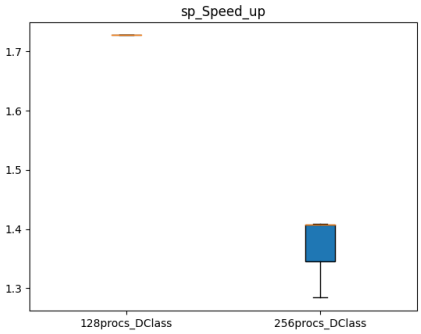
Σχήμα 6.140: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.141: *Speed-up Box plot διαγράμματα ep και is*

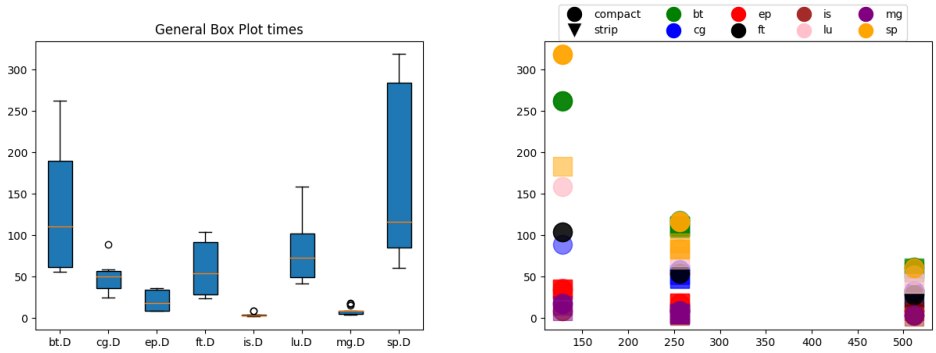


Σχήμα 6.142: *Speed-up Box plot διαγράμματα lu και mg*

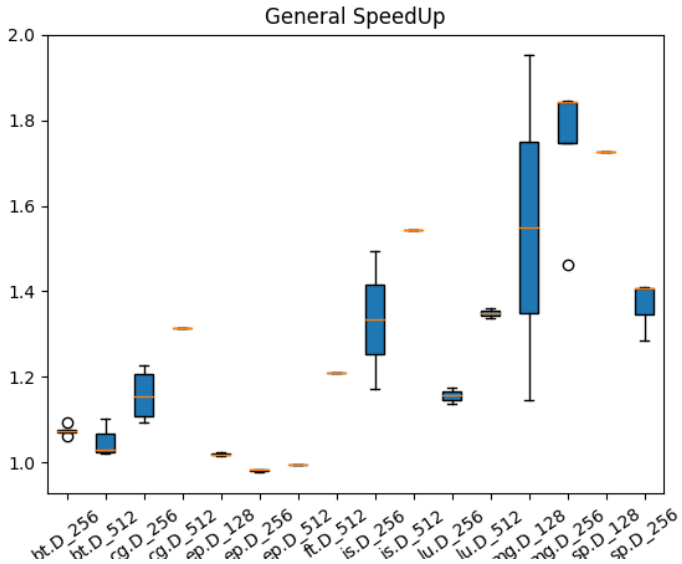


Σχήμα 6.143: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.144: *General Box plot και Scatter plot διαγράμματα του πειράματος*

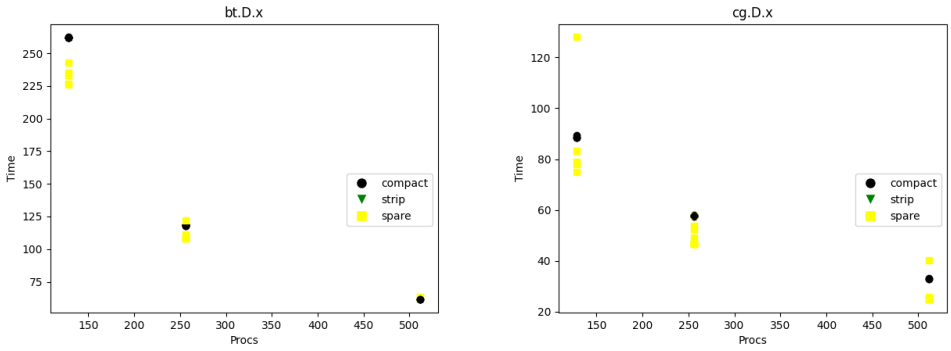


Σχήμα 6.145: *Γενικό Speed-up plot διάγραμμα*

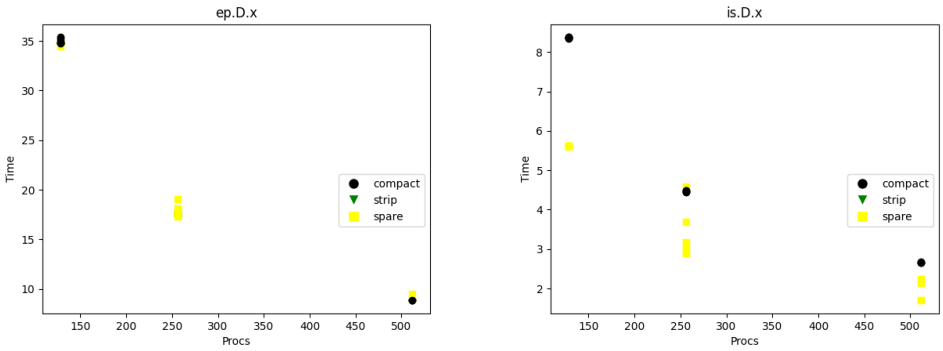
6.5.3.2 250 Εργασίες

Συνολικός χρόνος εκτέλεσης : 7550.0 δευτερόλεπτα (2 ώρες)
Μέσος χρόνος παραμονής μιας εργασίας στην ουρά: 3906.65 δευτερόλεπτα (1 ώρα)
Μέσος χρόνος εκτέλεσης μιας εργασία: 72.64 δευτερόλεπτα (1.2 λεπτά)

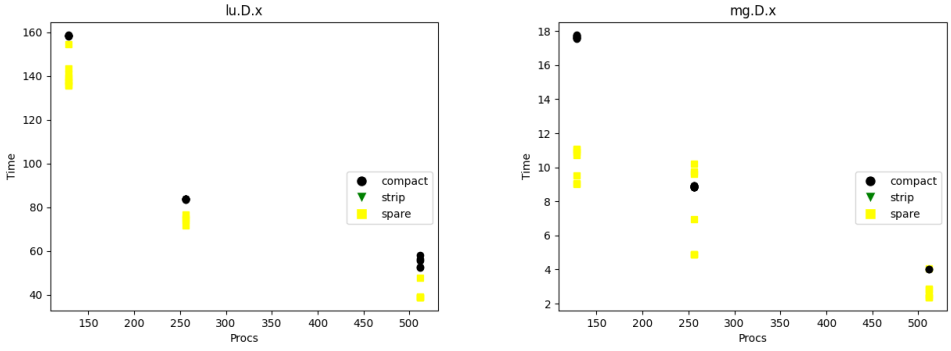
Γραφήματα εργασιών:



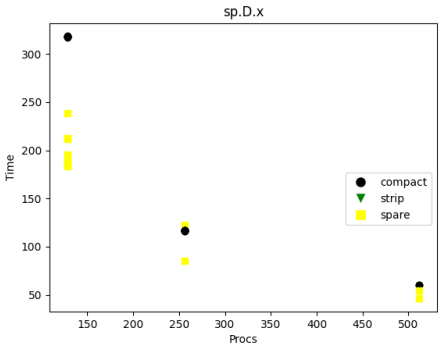
Σχήμα 6.146: *Διαγράμματα bt και cg*



Σχήμα 6.147: *Διαγράμματα ep και is*

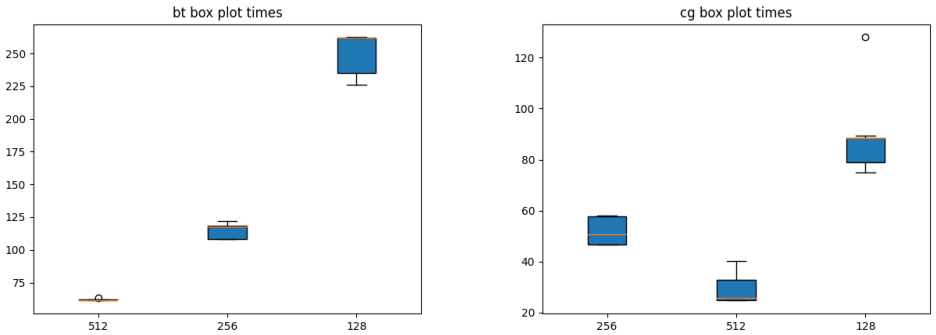


Σχήμα 6.148: *Διαγράμματα lu και mg*

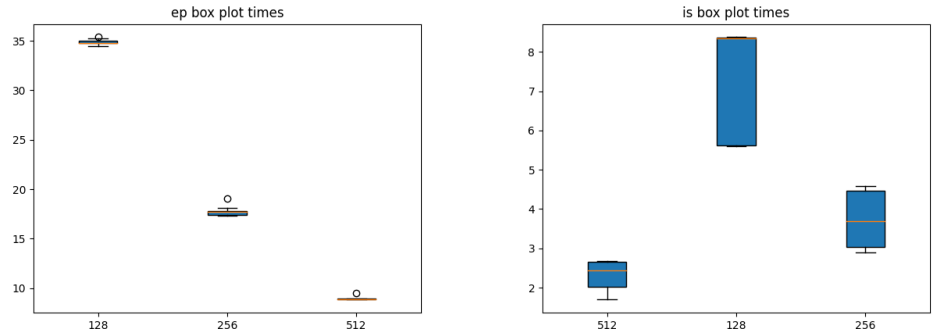


Σχήμα 6.149: *Διάγραμμα sp*

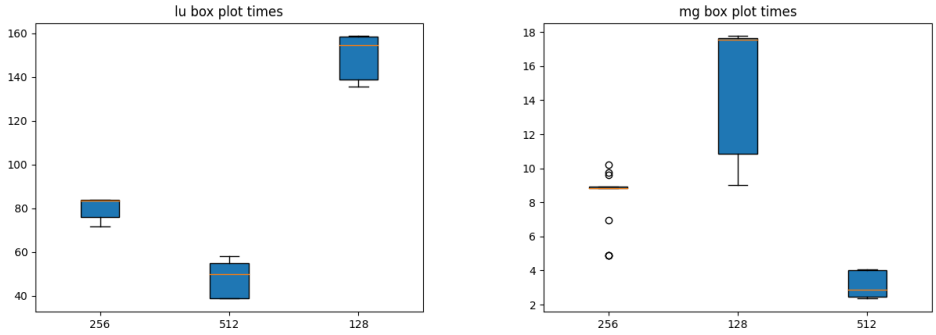
Γραφήματα τύπου Box plot των εργασιών:



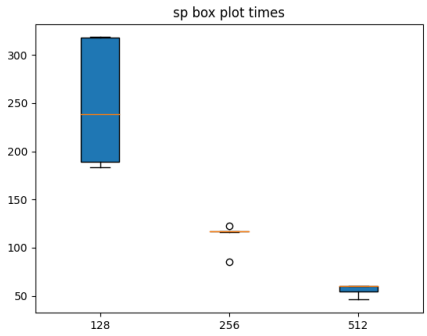
Σχήμα 6.150: *Box plot διαγράμματα bt και cg*



Σχήμα 6.151: *Box plot διαγράμματα ep και is*

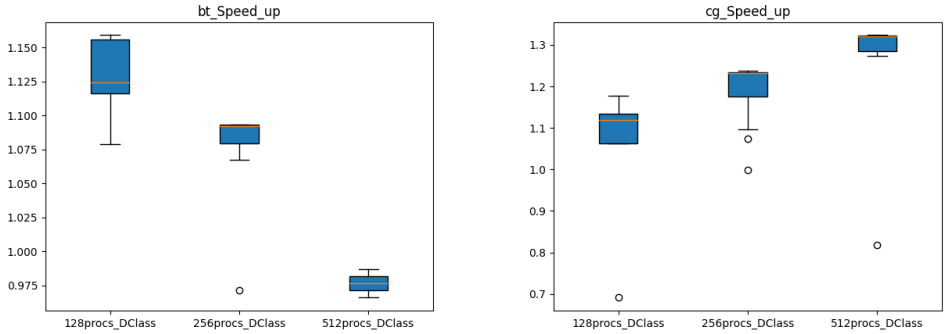


Σχήμα 6.152: *Box plot διαγράμματα lu και mg*

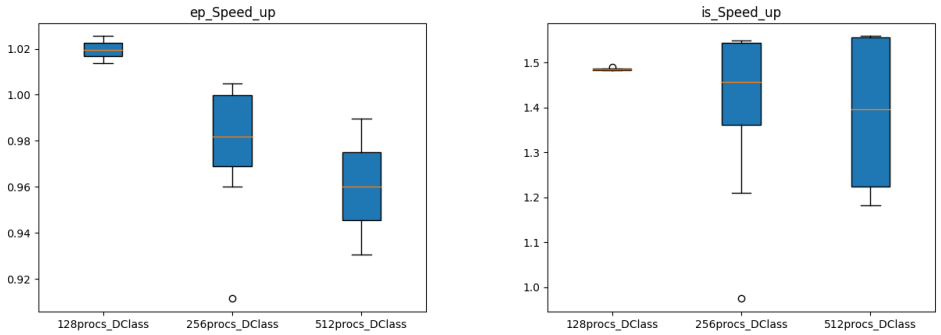


Σχήμα 6.153: *Box plot διάγραμμα sp*

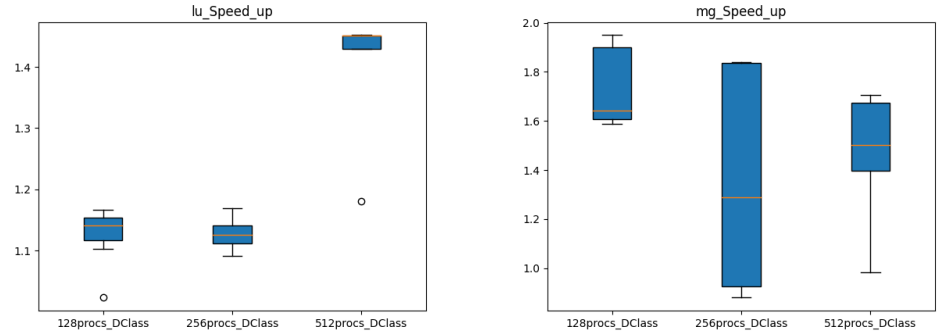
Γραφήματα τύπου Box plot για το speed-up των εργασιών:



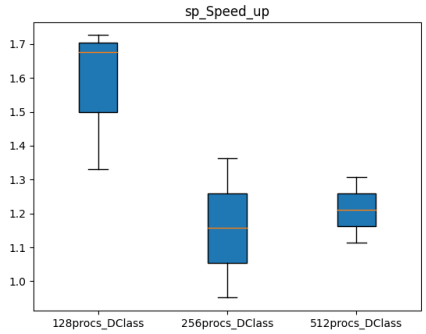
Σχήμα 6.154: *Speed-up Box plot διαγράμματα bt και cg*



Σχήμα 6.155: *Speed-up Box plot διαγράμματα ep και is*

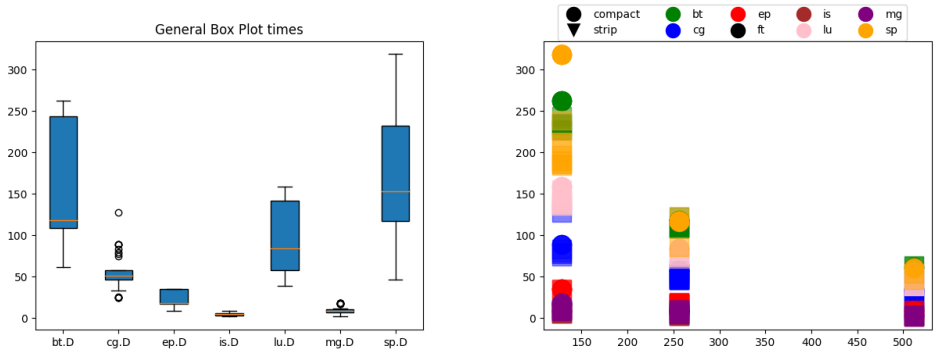


Σχήμα 6.156: *Speed-up Box plot διαγράμματα lu και mg*

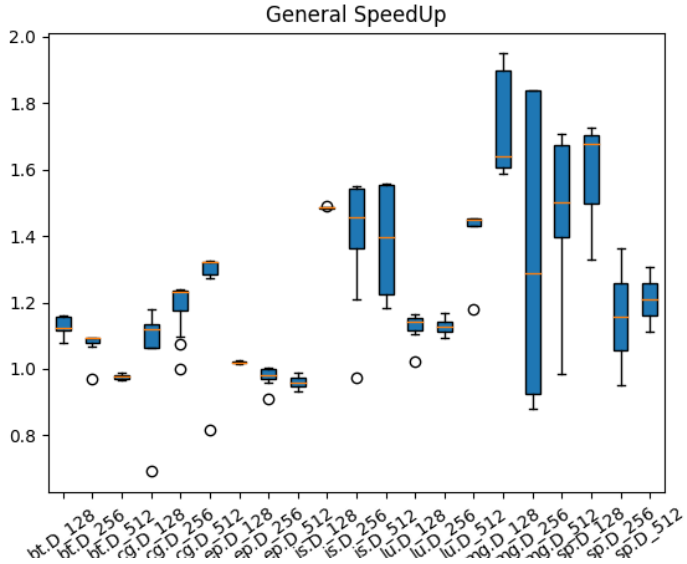


Σχήμα 6.157: *Speed-up Box plot διάγραμμα sp*

Γραφήματα Scatter plot και Box plot για όλο το πείραμα



Σχήμα 6.158: *General Box plot και Scatter plot διαγράμματα του πειράματος*



Σχήμα 6.159: *Γενικό Speed-up plot διάγραμμα*

6.5.4 Σχόλια

Σε αντίθεση με το σύστημα των κλόνων του CSLab, στον ARIS παρατηρούμε μία πιο σαφή εικόνα των αποτελεσμάτων. Συγκεκριμένα βλέπουμε:

- **Γενική βελτίωση της Compact πολιτικής**
- **Το Speed-up παραμένει ως επί των πλείστων ˜1, χωρίς να έχουμε τόσο μεγάλες τιμές (λόγω βελτίωσης του Compact).**
- **Πολύ μεγάλη μείωση στους χρόνους εκτέλεσης κάθε εργασίας, ακόμα και με την διαφορά σε κλάση.**
- **Όλες οι εφαρμογές εμφάνιζαν σχεδόν γραμμική αύξηση της απόδοσης με την αύξηση των διεργασιών της κάθε μίας.**

Η τόσο μεγάλη βελτίωση στην απόδοση κατά πάσα πιθανότητα οφείλεται στην μεγαλύτερη ισχύ του κάθε επεξεργαστή (10 πυρήνες έναντι των 4 στο σύστημα της σχολής) και την αρκετά μεγαλύτερη ταχύτητα διαύλου για την καλύτερη επικοινωνία μεταξύ των μηχανημάτων.

Κεφάλαιο 7

Μελλοντικές επεκτάσεις

Ο κώδικας θα μπορούσε να τροποποιηθεί, για να μπορεί να ανταπεξέρχεται καλύτερα σύμφωνα με τις ιδιαιτερότητες του κάθε μηχανήματος (π.χ. ύπαρξη GPU, διαφορετική μνήμη). Με αυτόν τον τρόπο θα γίνεται μια καλύτερη κατανομή των εργασιών στην περίπτωση αυτή.

Επίσης, η προσθήκη επιπλέον πολιτικών για την κατανομή των πόρων θα ήταν μια σημαντική προσθήκη στην λειτουργίκότητας της εφαρμόγης, δίνοτας στον χρήστη μεγαλύτερη ελευθερία στην εκτέλεση των πειραμάτων.

Τέλος, η συμβατότητα της εφαρμογής με επιπλέον benchmarks εκτός των NPB θα αποτελούσε μία μεγάλη επέκταση στην λειτουργικότητα και στις δυνατότες της.

Παραρτήματα

Παράρτημα Α′

Παράρτημα Data set

Α′.1 Data set τριάντα εργασιών με οκτώ διεργασίες

Αλγοριθμος Α′.1: *30 job, 8 procs*

```
1     cg.B.x 8 70  0 1 0  ep.B.x 8 20  0 1 1
2     ep.B.x 8 20  0 0 0  is.B.x 8 5   0 1 0
3     lu.B.x 8 120 0 1 0  bt.B.x 8 230 0 1 0
4     bt.B.x 8 230 0 0 1  mg.B.x 8 10  0 1 0
5     bt.B.x 8 230 0 1 0  mg.B.x 8 10  0 0 0
6     is.B.x 8 5   0 1 1  bt.B.x 8 230 0 1 0
7     sp.B.x 8 470 0 0 0  is.B.x 8 5   0 1 0
8     lu.B.x 8 120 0 1 0  bt.B.x 8 230 0 0 1
9     ep.B.x 8 20  0 0 0  is.B.x 8 5   0 1 1
10    cg.B.x 8 70  0 0 0  cg.B.x 8 70  0 0 1
11    cg.B.x 8 70  0 0 0  is.B.x 8 5   0 0 1
12    bt.B.x 8 230 0 0 0  sp.B.x 8 470 0 1 0
13    ep.B.x 8 20  0 1 1  ep.B.x 8 20  0 1 0
14    bt.B.x 8 230 0 1 0  sp.B.x 8 470 0 1 0
15    is.B.x 8 5   0 1 0  sp.B.x 8 470 0 1 1
```

Α′.2 Data set σαράντα εργασιών με τριανταδύο διεργασίες

Αλγοριθμος Α′.2: *40 job, 32 procs*

```
1    ep.B.x 32 5   0 1 1  cg.B.x 32 90  0 0 1
2    is.B.x 32 10  0 0 1  cg.B.x 32 90  0 1 0
3    is.B.x 32 10  0 1 0  is.B.x 32 10  0 0 0
4    is.B.x 32 10  0 0 1  ep.B.x 32 5   0 1 1
5    lu.B.x 32 45  0 1 1  bt.B.x 32 90  0 1 1
6    sp.B.x 32 160 0 0 0  lu.B.x 32 45  0 1 1
7    bt.B.x 32 90  0 0 1  sp.B.x 32 160 0 1 0
8    bt.B.x 32 90  0 0 0  is.B.x 32 10  0 0 0
9    lu.B.x 32 45  0 0 1  is.B.x 32 10  0 1 1
10   sp.B.x 32 160 0 1 0  ep.B.x 32 5   0 1 1
11   bt.B.x 32 90  0 0 0  is.B.x 32 10  0 0 0
12   cg.B.x 32 90  0 1 0  mg.B.x 32 10  0 1 1
13   bt.B.x 32 90  0 1 0  sp.B.x 32 160 0 0 0
14   mg.B.x 32 10  0 1 1  lu.B.x 32 45  0 1 0
15   mg.B.x 32 10  0 0 0  mg.B.x 32 10  0 1 0
16   ep.B.x 32 5   0 1 1  is.B.x 32 10  0 1 0
17   mg.B.x 32 10  0 0 1  mg.B.x 32 10  0 1 0
18   sp.B.x 32 160 0 0 0  sp.B.x 32 160 0 1 0
19   is.B.x 32 10  0 1 0  cg.B.x 32 90  0 1 0
20   sp.B.x 32 160 0 0 0  sp.B.x 32 160 0 0 0
```

Α΄.3 Data set πενήντα εργασιών με δεκαέξι διεργασίες

Αλγοριθμος Α΄.3: *50 job, 16 procs*

```
1    ep.B.x 16 10   0 1 1  sp.B.x 16 160 0 1 1
2    cg.B.x 16 60   0 1 1  ep.B.x 16 10  0 0 1
3    mg.B.x 16 10   0 0 0  bt.B.x 16 80  0 0 0
4    is.B.x 16 5    0 1 0  is.B.x 16 5   0 1 0
5    lu.B.x 16 60   0 1 1  lu.B.x 16 60  0 1 0
6    sp.B.x 16 160  0 0 1  mg.B.x 16 10  0 0 0
7    ep.B.x 16 10   0 1 0  cg.B.x 16 60  0 0 0
8    bt.B.x 16 80   0 1 0  is.B.x 16 5   0 1 1
9    lu.B.x 16 60   0 0 1  mg.B.x 16 10  0 1 0
10   bt.B.x 16 80   0 1 1  cg.B.x 16 60  0 0 0
11   bt.B.x 16 80   0 1 1  lu.B.x 16 60  0 1 1
12   lu.B.x 16 60   0 1 0  mg.B.x 16 10  0 0 1
13   lu.B.x 16 60   0 0 1  lu.B.x 16 60  0 0 0
14   lu.B.x 16 60   0 0 1  bt.B.x 16 80  0 1 1
15   is.B.x 16 5    0 0 1  is.B.x 16 5   0 0 1
16   lu.B.x 16 60   0 0 1  bt.B.x 16 80  0 0 1
17   mg.B.x 16 10   0 0 0  ep.B.x 16 10  0 1 1
18   bt.B.x 16 80   0 1 0  lu.B.x 16 60  0 0 0
19   mg.B.x 16 10   0 1 1  cg.B.x 16 60  0 0 1
20   lu.B.x 16 60   0 0 0  sp.B.x 16 160 0 0 1
21   cg.B.x 16 60   0 1 1  bt.B.x 16 80  0 1 1
22   bt.B.x 16 80   0 0 0  cg.B.x 16 60  0 0 1
23   sp.B.x 16 160  0 1 1  mg.B.x 16 10  0 1 1
24   sp.B.x 16 160  0 1 0  is.B.x 16 5   0 1 0
25   cg.B.x 16 60   0 1 0  ep.B.x 16 10  0 1 1
```

Α′.4 Data set εκατό τυχαίων εργασιών με τυχαίο αριθμό διεργασιών

Αλγοριθμος Α′.4: *100 random job queue*

```
1   mg.B.x 32 10  0 1 0  sp.B.x 16 160 0 0 0
2   bt.B.x 32 90  0 1 1  mg.B.x 4  20  0 1 1
3   ft.B.x 64 50  0 1 1  mg.B.x 16 10  0 1 1
4   ep.B.x 64 3   0 0 1  bt.B.x 16 80  0 0 0
5   ft.B.x 64 50  0 0 0  bt.B.x 16 80  0 0 1
6   is.B.x 8  5   0 1 1  ep.B.x 64 3   0 1 1
7   bt.B.x 64 80  0 0 1  ep.B.x 64 3   0 1 0
8   ep.B.x 64 3   0 1 1  cg.B.x 64 150 0 1 1
9   ep.B.x 64 3   0 1 1  lu.B.x 16 60  0 1 0
10  mg.B.x 16 10  0 1 0  mg.B.x 16 10  0 0 1
11  ep.B.x 8  20  0 0 0  lu.B.x 32 45  0 0 1
12  mg.B.x 4  20  0 1 1  ep.B.x 32 5   0 0 1
13  bt.B.x 32 90  0 0 1  sp.B.x 4  470 0 1 1
14  cg.B.x 8  70  0 1 1  sp.B.x 4  470 0 0 0
15  mg.B.x 64 5   0 1 1  is.B.x 64 20  0 1 1
16  is.B.x 16 5   0 0 1  lu.B.x 64 50  0 1 1
17  sp.B.x 8  470 0 1 1  mg.B.x 8  10  0 1 1
18  cg.B.x 64 150 0 1 0  sp.B.x 32 160 0 1 1
19  bt.B.x 8  230 0 1 1  sp.B.x 8  470 0 1 0
20  lu.B.x 64 50  0 0 1  bt.B.x 64 80  0 0 1
21  sp.B.x 16 160 0 1 1  ep.B.x 16 10  0 1 0
22  ft.B.x 16 50  0 1 1  is.B.x 64 20  0 1 1
23  mg.B.x 16 10  0 0 0  lu.B.x 8  120 0 1 0
24  cg.B.x 16 60  0 1 0  ft.B.x 32 60  0 1 0
25  bt.B.x 64 80  0 1 0  ep.B.x 8  20  0 1 0
26  is.B.x 64 20  0 0 1  ep.B.x 16 10  0 0 1
27  mg.B.x 32 10  0 1 0  mg.B.x 4  20  0 0 1
28  lu.B.x 32 45  0 0 1  lu.B.x 64 50  0 0 1
29  ft.B.x 16 50  0 1 1  sp.B.x 64 130 0 1 1
30  sp.B.x 16 160 0 0 1  mg.B.x 4  20  0 0 1
31  bt.B.x 64 80  0 0 1  mg.B.x 32 10  0 1 1
32  cg.B.x 16 60  0 0 1  bt.B.x 4  230 0 1 1
33  ep.B.x 32 5   0 0 0  ft.B.x 8  60  0 1 0
34  cg.B.x 16 60  0 1 1  sp.B.x 32 160 0 0 0
35  lu.B.x 4  170 0 1 0  ep.B.x 8  20  0 0 0
36  is.B.x 8  5   0 1 1  bt.B.x 8  230 0 1 0
37  is.B.x 64 20  0 1 1  sp.B.x 16 160 0 1 0
38  ft.B.x 64 50  0 1 0  ep.B.x 4  40  0 0 0
39  bt.B.x 32 90  0 0 1  ft.B.x 8  60  0 1 1
40  ft.B.x 64 50  0 1 1  ep.B.x 64 3   0 1 0
```

```
    ep.B.x 32 5   0 1 0  cg.B.x 16 60  0 0 0
    ep.B.x 4  40  0 1 0  mg.B.x 64 5   0 1 1
    cg.B.x 16 60  0 0 0  mg.B.x 16 10  0 0 1
    sp.B.x 64 130 0 1 0  bt.B.x 32 90  0 0 0
    ep.B.x 4  40  0 1 0  cg.B.x 32 90  0 0 0
    is.B.x 64 20  0 1 0  is.B.x 16 5   0 0 0
    bt.B.x 8  230 0 1 1  sp.B.x 32 160 0 1 1
    mg.B.x 32 10  0 0 0  sp.B.x 4  470 0 1 1
    mg.B.x 16 10  0 1 1  bt.B.x 64 80  0 1 0
    ft.B.x 8  60  0 1 0  cg.B.x 64 150 0 0 1
```

Α΄.5 Data set διακοσίων πενήντα τυχαίων εργασιών με τυχαίο αριθμό διεργασιών

Αλγοριθμος Α΄.5: *250 random job queue*

```
    lu.B.x 16 60  0 1 0  ep.B.x 8  20  0 0 0
    ep.B.x 16 10  0 0 1  ep.B.x 32 5   0 0 1
    bt.B.x 64 80  0 0 0  mg.B.x 16 10  0 1 0
    mg.B.x 8  10  0 0 1  cg.B.x 32 90  0 0 1
    lu.B.x 16 60  0 1 0  sp.B.x 4  470 0 1 0
    mg.B.x 16 10  0 1 0  cg.B.x 16 60  0 1 0
    cg.B.x 32 90  0 0 1  is.B.x 64 20  0 0 1
    ep.B.x 32 5   0 1 1  lu.B.x 16 60  0 1 0
    lu.B.x 64 50  0 1 0  bt.B.x 64 80  0 1 0
    cg.B.x 16 60  0 1 1  cg.B.x 64 150 0 0 1
    sp.B.x 32 160 0 1 1  cg.B.x 4  90  0 0 1
    lu.B.x 64 50  0 0 1  mg.B.x 16 10  0 1 1
    sp.B.x 4  470 0 1 1  lu.B.x 16 60  0 1 0
    ep.B.x 32 5   0 0 0  mg.B.x 4  20  0 0 1
    bt.B.x 16 80  0 0 1  mg.B.x 4  20  0 0 0
    lu.B.x 4  170 0 1 0  cg.B.x 32 90  0 1 1
    bt.B.x 8  230 0 1 0  is.B.x 64 20  0 1 1
    ep.B.x 64 3   0 1 0  ep.B.x 8  20  0 1 0
    mg.B.x 4  20  0 1 0  ep.B.x 4  40  0 1 1
    cg.B.x 32 90  0 0 1  mg.B.x 16 10  0 1 0
    cg.B.x 32 90  0 0 0  mg.B.x 32 10  0 0 0
    mg.B.x 64 5   0 0 1  sp.B.x 4  470 0 0 0
    lu.B.x 32 45  0 1 1  is.B.x 8  5   0 0 0
    mg.B.x 32 10  0 0 1  is.B.x 8  5   0 1 0
    bt.B.x 16 80  0 0 0  mg.B.x 8  10  0 1 1
    lu.B.x 8  120 0 1 0  ep.B.x 32 5   0 0 0
```

```
1    is.B.x 8  5   0 1 0  cg.B.x 8  70  0 1 1
2    cg.B.x 4  90  0 1 0  bt.B.x 4  230 0 1 0
3    mg.B.x 4  20  0 1 0  bt.B.x 64 80  0 1 0
4    cg.B.x 16 60  0 0 1  sp.B.x 8  470 0 1 1
5    cg.B.x 4  90  0 1 0  sp.B.x 16 160 0 1 1
6    mg.B.x 4  20  0 1 1  lu.B.x 64 50  0 0 0
7    is.B.x 64 20  0 1 0  is.B.x 64 20  0 1 0
8    lu.B.x 8  120 0 0 1  is.B.x 4  5   0 1 0
9    lu.B.x 16 60  0 0 1  mg.B.x 32 10  0 0 1
10   cg.B.x 16 60  0 0 1  mg.B.x 16 10  0 1 1
11   ep.B.x 32 5   0 0 0  cg.B.x 16 60  0 1 0
12   mg.B.x 64 5   0 0 1  bt.B.x 4  230 0 1 1
13   cg.B.x 64 150 0 0 1  sp.B.x 16 160 0 1 1
14   sp.B.x 64 130 0 1 0  bt.B.x 32 90  0 0 1
15   bt.B.x 8  230 0 1 0  bt.B.x 16 80  0 1 0
16   sp.B.x 32 160 0 1 1  bt.B.x 4  230 0 1 1
17   is.B.x 16 5   0 0 1  mg.B.x 32 10  0 1 0
18   is.B.x 32 10  0 0 1  lu.B.x 4  170 0 0 1
19   is.B.x 16 5   0 0 1  cg.B.x 64 150 0 1 1
20   sp.B.x 4  470 0 0 0  sp.B.x 4  470 0 1 1
21   is.B.x 8  5   0 0 1  bt.B.x 8  230 0 0 1
22   sp.B.x 4  470 0 0 0  cg.B.x 64 150 0 1 1
23   sp.B.x 16 160 0 0 0  cg.B.x 32 90  0 0 0
24   mg.B.x 64 5   0 0 1  lu.B.x 4  170 0 0 0
25   lu.B.x 4  170 0 0 0  bt.B.x 64 80  0 0 1
26   cg.B.x 16 60  0 1 1  ep.B.x 8  20  0 1 0
27   mg.B.x 8  10  0 1 0  cg.B.x 64 150 0 0 0
28   ep.B.x 64 3   0 1 0  is.B.x 8  5   0 1 1
29   sp.B.x 64 130 0 1 0  sp.B.x 8  470 0 0 0
30   cg.B.x 8  70  0 0 0  sp.B.x 4  470 0 0 0
31   lu.B.x 64 50  0 0 1  is.B.x 4  5   0 1 0
32   ep.B.x 4  40  0 1 1  sp.B.x 32 160 0 1 1
33   mg.B.x 4  20  0 0 1  is.B.x 8  5   0 0 1
34   lu.B.x 64 50  0 0 0  sp.B.x 8  470 0 1 1
35   mg.B.x 32 10  0 1 0  ep.B.x 16 10  0 0 0
36   is.B.x 4  5   0 1 1  sp.B.x 32 160 0 1 1
37   sp.B.x 4  470 0 0 1  is.B.x 16 5   0 0 1
38   lu.B.x 32 45  0 1 0  bt.B.x 8  230 0 0 0
39   is.B.x 64 20  0 0 0  lu.B.x 64 50  0 1 0
40   bt.B.x 8  230 0 0 1  sp.B.x 32 160 0 1 0
41   is.B.x 64 20  0 1 0  is.B.x 8  5   0 0 1
42   cg.B.x 32 90  0 1 1  lu.B.x 64 50  0 0 0
43   mg.B.x 32 10  0 1 1  mg.B.x 4  20  0 1 1
44   mg.B.x 4  20  0 0 0  sp.B.x 32 160 0 1 0
45   ep.B.x 16 10  0 1 0  mg.B.x 8  10  0 1 0
46   cg.B.x 64 150 0 1 1  lu.B.x 64 50  0 1 1
47   is.B.x 4  5   0 0 0  lu.B.x 4  170 0 0 0
48   sp.B.x 32 160 0 1 0  cg.B.x 32 90  0 0 0
49   lu.B.x 8  120 0 1 0  bt.B.x 8  230 0 1 1
50   ep.B.x 16 10  0 0 1  mg.B.x 32 10  0 1 1
51   lu.B.x 16 60  0 1 1  sp.B.x 64 130 0 1 0
```

```
lu.B.x 64 50  0 1 0  mg.B.x 64 5   0 0 1
sp.B.x 4  470 0 0 0  bt.B.x 16 80  0 0 0
bt.B.x 8  230 0 1 0  mg.B.x 32 10  0 1 0
ep.B.x 8  20  0 1 0  bt.B.x 16 80  0 1 0
ep.B.x 64 3   0 1 1  cg.B.x 16 60  0 0 0
sp.B.x 4  470 0 0 0  is.B.x 8  5   0 0 1
cg.B.x 4  90  0 0 0  is.B.x 32 10  0 1 0
is.B.x 8  5   0 1 1  mg.B.x 32 10  0 0 0
lu.B.x 32 45  0 0 1  lu.B.x 8  120 0 0 1
is.B.x 64 20  0 0 0  mg.B.x 32 10  0 0 1
is.B.x 16 5   0 0 0  cg.B.x 4  90  0 0 0
is.B.x 32 10  0 1 0  ep.B.x 4  40  0 1 1
cg.B.x 64 150 0 0 0  ep.B.x 64 3   0 0 1
ep.B.x 4  40  0 0 1  bt.B.x 4  230 0 0 1
bt.B.x 16 80  0 1 0  lu.B.x 8  120 0 1 1
sp.B.x 64 130 0 0 1  is.B.x 16 5   0 1 0
cg.B.x 16 60  0 0 0  mg.B.x 64 5   0 0 0
sp.B.x 16 160 0 1 0  bt.B.x 16 80  0 0 0
cg.B.x 4  90  0 1 1  bt.B.x 64 80  0 1 0
ep.B.x 8  20  0 1 1  mg.B.x 16 10  0 1 0
cg.B.x 16 60  0 0 0  is.B.x 32 10  0 1 1
cg.B.x 8  70  0 0 0  lu.B.x 4  170 0 1 1
is.B.x 32 10  0 0 1  sp.B.x 64 130 0 0 0
sp.B.x 16 160 0 0 1  bt.B.x 32 90  0 1 1
sp.B.x 8  470 0 1 0  bt.B.x 64 80  0 1 0
lu.B.x 8  120 0 0 1  sp.B.x 8  470 0 1 1
mg.B.x 64 5   0 1 1  mg.B.x 16 10  0 0 0
bt.B.x 32 90  0 1 0  sp.B.x 8  470 0 1 0
is.B.x 4  5   0 1 0  bt.B.x 4  230 0 1 0
mg.B.x 8  10  0 1 0  bt.B.x 16 80  0 1 0
ep.B.x 32 5   0 0 0  ep.B.x 32 5   0 0 0
mg.B.x 4  20  0 1 0  ep.B.x 8  20  0 1 0
is.B.x 16 5   0 0 1  lu.B.x 4  170 0 1 1
is.B.x 64 20  0 0 1  bt.B.x 32 90  0 1 1
mg.B.x 32 10  0 1 1  sp.B.x 8  470 0 0 0
ep.B.x 64 3   0 0 1  bt.B.x 16 80  0 0 1
lu.B.x 32 45  0 0 0  mg.B.x 4  20  0 0 1
mg.B.x 16 10  0 0 0  lu.B.x 8  120 0 0 0
bt.B.x 8  230 0 0 0  ep.B.x 16 10  0 0 1
cg.B.x 64 150 0 0 1  mg.B.x 64 5   0 1 1
mg.B.x 8  10  0 0 0  ep.B.x 64 3   0 1 0
ep.B.x 32 5   0 0 0  cg.B.x 16 60  0 0 1
lu.B.x 4  170 0 1 1  mg.B.x 16 10  0 1 1
bt.B.x 32 90  0 0 0  cg.B.x 32 90  0 1 1
mg.B.x 16 10  0 1 1  mg.B.x 64 5   0 0 1
is.B.x 16 5   0 1 0  cg.B.x 64 150 0 0 0
is.B.x 16 5   0 0 1  cg.B.x 16 60  0 0 0
lu.B.x 16 60  0 0 1  lu.B.x 64 50  0 1 0
```

Παράρτημα Β′

Heatmap

	BT	*CG*	*EP*	*FT*	*IS*	*LU*	*MG*	*SP*
BT	6	5	1	8	4	2	7	3
CG	2	7	1	5	8	3	6	4
EP	4	7	8	5	6	3	2	1
FT	6	3	1	8	4	2	5	7
IS	3	7	1	6	8	2	5	4
LU	2	4	1	8	3	6	7	5
MG	2	4	1	7	6	3	8	5
SP	6	2	1	7	3	4	5	8

Εικόνα Β′.1: *Heatmap*

Βιβλιογραφία

[1] CSLab. *Parallel Processing Systems*. σελίδες 7–12. NTUA, 2015.

[2] MPI. *MPI: A Message-Passing Interface Standard*. https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf, 2021.

[3] Professor David W. Walker Professor Geoffrey C. Fox. *Concurrencyand Computation-Practice and Experience*. https://doi.org/10.1002/cpe.3670, 2016.

[4] ARIS Documentation. *Hardware Overview*. https://doc.aris.grnet.gr/system/hardware/.

[5] GRnet. *Εισαγωγή στους υπερυπολογιστές*. https://hpc.grnet.gr/supercomputer.

[6] Mark Baker και Rajkumar Buyya. *Cluster Computing at a Glance*. σελίδες 1–45. Prentice Hall PTR, 1999.

[7] *Kubernetes*. https://github.com/kubernetes/kubernetes.

[8] Αλέξιος Παπαβασιλείου. *Υλοποίηση εικονικού διαχειριστή πόρων για MPI εφαρμογές σε υπερυπολογιστικά συστήματα*. http://artemis.cslab.ece.ntua.gr:8080/jspui/bitstream/123456789/18315/1/thesis.pdf, 2022.

[9] Adaptive Computing. *Torque Resource Manager*. https://support.adaptivecomputing.com/wp-content/uploads/2021/02/torqueAdminGuide-6.1.3.pdf, 2021.

[10] Wikipedia. *Box plot definition*. https://en.wikipedia.org/wiki/Box_plot.

[11] *Intel® Xeon® Processor E5335*. https://ark.intel.com/content/www/us/en/ark/products/28443/intel-xeon-processor-e5335-8m-cache-2-00-ghz-1333-mhz-fsb.html.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλ.	βλέπε
κτλ.	και τα λοιπά
κ.ο.κ	και ούτω καθεξής
π.χ.	παραδείγματος χάριν
ΚΜΕ	Κεντρική Μονάδα Επεξεργασίας
HPC	High-performance computing
MPI	Message Passing Interface
NPB	NAS Parallel Benchmarks
LLC	logical link control