



**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**  
**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**  
**ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (CSLAB)**

**ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ**

“Σχεδιασμός Middleware για Data-Intensive Εφαρμογές”

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

Γεώργιος Δ. Χαντζηαλεξίου

**Επιβλέποντες : Georgios I. Goumas – Professor NTUA**  
**Shantenu Jha – Professor Rutgers University**

Αθήνα, Οκτώβριος 2017

Αυτή η σελίδα έμεινε σκοπίμως κενή



**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**  
**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**  
**ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (CSLAB)**

**ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ**

“Designing Middleware for High-Performance Data-Intensive Applications”

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

Γεώργιος Δ. Χαντζηαλεξίου

**Επιβλέποντες: Georgios I. Goumas – Professor NTUA**  
**Shantenu Jha – Professor Rutgers University**

Adopted by the three members of the committee, in October 2017:

.....  
Γεώργιος Γκούμας  
Καθηγητής Ε.Μ.Π.

.....  
Νεκτάριος Κοζύρης  
Καθηγητής Ε.Μ.Π.

.....  
Shantenu Jha  
Καθηγητής Rutgers

Αθήνα, Οκτώβριος 2017

Γεώργιος Δ. Χαντζηαλεξίου  
Φοιτητής της σχολής των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του  
Εθνικού Μετσόβιου Πολυτεχνείου

Copyright © Γεώργιος Δ. Χαντζηαλεξίου,  
Με επιφύλαξη παντός δικαιώματος. All rights reserved

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

## **ΕΥΧΑΡΙΣΤΙΕΣ**

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Γεώργιο Γκούμα, που μου επέτρεψε να εργαστώ πάνω στο θέμα που επέλεξα. Επιπλέον θα ήθελα να ευχαριστήσω τον Professor Shantenu Jha που με εμπιστεύτηκε με την συγκεκριμένη εργασία και μου επέτρεψε να την συνεχίζω σαν θέμα της διπλωματικής μου. Οι γνώσεις και οι εμπειρίες που απέκτησα κατά τη διάρκεια ανάπτυξης της διπλωματικής ήταν πολύ σημαντικές και ενδιαφέρουσες. Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου και τις αδερφές μου για την στήριξή τους καθ όλη τη διάρκεια των σπουδών μου. Τέλος θα ήθελα να ευχαριστήσω τους φίλους και συνάδελφους Μήτρο και Όμηρο για τις ώρες που περάσαμε μαζί διαβάζοντας, καθώς το μοναχικό διάβασμα είναι πολύ πιο κουραστικό, αλλά και όλους τους υπόλοιπους φίλους μου για την στήριξή τους και την παρέα τους εδώ και πολλά χρόνια. Μάξιμε, Τζαννέ, Μπιλ, Θανάση, σας ευχαριστώ.

## Περίληψη

Πλατφόρμες υπολογιστικής υψηλής απόδοσης, όπως οι "supercomputers", έχουν παραδοσιακά σχεδιαστεί για να ικανοποιούν τις απαιτήσεις υπολογισμών των επιστημονικών εφαρμογών. Κατά συνέπεια, έχουν σχεδιαστεί ως καθαροί παραγωγοί και όχι ως καταναλωτές δεδομένων. Το οικοσύστημα της Apache εξελίχθηκε για να ικανοποιήσει τις απαιτήσεις των εφαρμογών επεξεργασίας πολλών δεδομένων και έχει αντιμετωπίσει πολλούς από τους παραδοσιακούς περιορισμούς των πλατφορμών H.P.C. Υπάρχει όμως μια κατηγορία επιστημονικών εφαρμογών που χρειάζονται τις συλλογικές δυνατότητες των παραδοσιακών υπολογιστικών περιβαλλόντων υψηλής απόδοσης και του οικοσυστήματος της Apache. Για παράδειγμα, οι επιστημονικοί τομείς της μοριακής δυναμικής, της γονιδιωματικής και της επιστήμης δικτύων πρέπει να ενώσουν τους παραδοσιακούς υπολογιστές με την ανάλυση Hadoop / Spark. Εξετάζουμε το κρίσιμο ερώτημα σχετικά με τον τρόπο παρουσίασης των δυνατοτήτων και των δύο υπολογιστικών περιβαλλόντων σε τέτοιες επιστημονικές εφαρμογές. Ενώ αυτά τα ερωτήματα χρειάζονται απαντήσεις σε πολλαπλά επίπεδα, σχεδιάσαμε ένα middleware διαχείρισης πόρων που θα μπορούσε να υποστηρίξει τις ανάγκες και των δύο. Προτείνουμε την επέκταση στο Pilot-Abstraction του radical pilot έτσι ώστε να παρέχουμε ένα ενοποιημένο επίπεδο διαχείρισης πόρων. Πρόκειται για ένα σημαντικό βήμα προς τη διαλειτουργική χρήση των οικοσυστημάτων HPC και Apache Spark. Επιτρέπει επίσης στις εφαρμογές να ενσωματώνουν στάδια HPC (π.χ. προσομοιώσεις) στην ανάλυση δεδομένων. Πολλά κέντρα υπερυπολογιστών έχουν αρχίσει να υποστηρίζουν επίσημα τα περιβάλλοντα Hadoop, είτε σε ένα αποκλειστικό περιβάλλον είτε σε υβριδικές αναπτύξεις χρησιμοποιώντας εργαλεία όπως το myHadoop. Αυτό συνήθως περιλαμβάνει πολλές εγγενείς λεπτομέρειες για το περιβάλλον που πρόκειται να χρησιμοποιηθεί και συχνά έχουμε ζητήματα όπως: Πώς να διερευνηθούν οι επιλογές όπως data locality έναντι data movement ;

Για το σκοπό αυτό, η πειραματική ανάλυση της απόδοσης είναι απαραίτητη μέσω μιας διαδικασίας παρακολούθησης του συνολικού χρόνου ολοκλήρωσης δύο επιλεγμένων αλγορίθμων και προσεκτικά επιλεγμένου συνόλου δεδομένων. Με αυτά τα αποτελέσματα μπορούμε να κατανοήσουμε την συμπεριφορά του στρώματος πόρων για αλλαγή σε διαφορετικές παραμέτρους απόδοσης των εφαρμογών μας.

Στην παρούσα εργασία αποφασίσαμε να μελετήσουμε τον αλγόριθμο k-means που χρησιμοποιείται για την ομαδοποίηση των λειτουργιών δεδομένων και τη μελέτη ενός αλγορίθμου μοριακής δυναμικής, του οποίου το όνομα είναι leaflet finder, χρησιμοποιώντας το radical-pilot και το pilot-Spark.

Εφαρμόζουμε και τους δύο αλγορίθμους και τρέχουμε πειράματα για να ανακτήσουμε τη μέτρηση που σχετίζεται με τη χρήση υπολογιστικών πόρων και τον συνολικό χρόνο εκτέλεσης για κάθε δοκιμή. Αναλύουμε τα αποτελέσματα των συμπεριφορών των μετρήσεων που εκτελούνται σε τρεις διαφορετικούς υπέρ-υπολογιστές υψηλής απόδοσης, Stampede, Wrangler και Comet. Χρησιμοποιούμε τα δεδομένα που συλλέγουμε για κάθε μέτρηση και εκτέλεση για να αποδείξουμε την ακρίβεια και τη χρησιμότητα του νέου υπολογιστικού στρώματος. Ελέγξτε την επεκτασιμότητα των αλγορίθμων στην κατανεμημένη έκδοση του a και παρατηρήστε την πιθανή βελτίωση του χρόνου. Τέλος προσπαθούμε να συγκρίνουμε και

τα δύο middleware και να σχολιάσουμε την υπεροχή ή όχι της αρχιτεκτονικής του Apache Spark.

**Keywords:** Κ-μέσα, Αναζήτηση Φυλλαδίων, ομαδοποίηση, Apache, Spark, χαρακτηρισμός απόδοσης, μεσαία λογισμικά, κατανεμημένα υπολογιστικά συστήματα, υπολογιστές υψηλών επιδόσεων

# Abstract

High-performance computing platforms such as “supercomputers” have traditionally been designed to meet the compute demands of scientific applications. Consequently, they have been architected as net producers and not consumers of data. The Apache Hadoop ecosystem has evolved to meet the requirements of data processing applications and has addressed many of the traditional limitations of HPC platforms. There exist a class of scientific applications however, that need the collective capabilities of traditional high-performance computing environments and the Apache Hadoop ecosystem. For example, the scientific domains of bio-molecular dynamics, genomics and network science need to couple traditional computing with Hadoop/Spark based analysis. We investigate the critical question of how to present the capabilities of both computing environments to such scientific applications. Whereas this questions needs answers at multiple levels, we designed a resource management middleware that might support the needs of both. We propose extensions to the Pilot-Abstraction of radical-pilot so as to provide a unifying resource management layer. This is an important step towards interoperable use of HPC and Spark. It also allows applications to integrate HPC stages (e. g. simulations) to data analytics. Many supercomputing centers have started to officially support Hadoop environments, either in a dedicated environment or in hybrid deployments using tools such as myHadoop. This typically involves many intrinsic, environment-specific details that need to be mastered, and often swamp conceptual issues like: How to explore runtime trade-offs (data localities vs. data movement)?

For this purpose, the experimental analysis of the performance is necessary through a process of tracking the total time of completion of two selected algorithms, and carefully selected dataset. With these results we can understand the behavior of the resource layer for change in different performance parameters of our applications.

In the present work we decided to study the k-means algorithm used for clustering data operations, and the study of a molecular dynamics algorithm, whose name is leaflet-finder, using the radical-pilot and radical pilot-spark environments.

We implement both algorithms and run experiments to retrieve metric associated with the use of computing resources and time performance of each test execution. We analyze the results of our measurements behaviors executing in three different state of the art High Performance Computers, Stampede, Wrangler and Comet. We use the data collected for each metric and execution to prove the accuracy and the usefulness of the new resource layer. Check the scalability of the algorithms in the distributed version of a and observe probable time improvement. Finally we attempt a comparison of both middlewares that shows us the architecture superiority of Spark.

**Keywords:** K-means, Leaflet Finder, clustering, Apache, Spark, performance characterization, middleware, distributed computing, high performance computing



# Πίνακας Περιεχομένων

Κεφάλαιο 1	Σ.9
1. Εισαγωγή	9
1.1 Κίνητρο	9
1.2 Σκοπός της διπλωματικής εργασίας	10
1.3 Διάρθρωση της διπλωματικής εργασίας	11
Κεφάλαιο 2	11
2.1 Συσταδοποίηση	11
2.1.1 Ανάλυση συσταδοποίησης	11
2.1.2 Αλγόριθμος συσταδοποίησης	14
2.2 Μοριακή Δυναμική	15
2.2.1 Ανάλυση μοριακής δυναμικής	15
2.2.2 Αλγόριθμος Leaflet-Finder	17
2.3 Επιστημονικά εργαλεία	18
2.3.1 Apache Spark	18
2.3.1.1 Εισαγωγή	18
2.3.1.2 Αρχιτεκτονική του Spark	19
1. Executors	20
2. Driver	20
2.3.1.3 Resilient Distributed Datasets	21
2.3.1.4 Caching	22
2.3.2 Radical Cybertools	23
2.3.2.1 Εισαγωγή	23
2.3.2.2 Radical Pilot	24
2.3.2.3 Radical Pilot-Spark	27
2.4 Εργαλεία	28
2.4.1 Map-Reduce	28
2.4.2 Lustre Filesystem	32
2.4.2.1 Εισαγωγή	32
2.4.2.2 Lustre Architecture	32
Κεφάλαιο 3	
3. Εκτέλεση και μοντελοποίηση του αλγόριθμου K-μέσων χρησιμοποιώντας το Radical-Pilot και Pilot-Spark	34
3.1 Περιγραφή της Πειραματικής Διαδικασίας	34
3.2 Κατασκευή Συνόλου Δεδομένων	35
3.3 Αποτελέσματα των Πειραμάτων	36
Κεφάλαιο 4	40
4. Εκτέλεση και μοντελοποίηση του αλγόριθμου leaf-finder χρησιμοποιώντας το Radical-Pilot και Pilot-Spark	40
4.1 Περιγραφή της Πειραματικής Διαδικασίας	40
4.2 Εφαρμογή leaf-finder πάνω στο Radical-Pilot	42

4.3 Αποτελέσματα Πειραμάτων	44
Κεφάλαιο 5	
5. Επίλογος	46
Κεφάλαιο 6	46
6. Cyber-Infrastructure	46
6.1 Stampede	46
6.2 Wrangler	48
6.3 Comet	49
Βιβλιογραφία	52



# Κεφάλαιο 1

## 1. Εισαγωγή

### 1.1 Κίνητρο

Η αφαίρεση Map-Reduce που χρησιμοποιείται ευρέως από το Apache Hadoop έχει χρησιμοποιηθεί με επιτυχία για πολλές εφαρμογές με μεγάλο όγκο δεδομένων σε διάφορους τομείς. Ένας σημαντικός διαφοροποιητής του Hadoop σε σύγκριση με το HPC είναι η διαθεσιμότητα πολλών αφαίρεσης υψηλού επιπέδου και εργαλείων για αποθήκευση δεδομένων, μετασχηματισμούς και προηγμένες αναλύσεις. Αυτές οι αφαιρέσεις επιτρέπουν συνήθως τη συλλογιστική υψηλού επιπέδου σχετικά με τον παραλληλισμό των δεδομένων χωρίς την ανάγκη χειροκίνητης κατάτμησης των δεδομένων, τη διαχείριση των εργασιών επεξεργασίας αυτών των δεδομένων και τη συλλογή των αποτελεσμάτων, η οποία απαιτείται σε άλλα περιβάλλοντα. Μέσα στο οικοσύστημα Hadoop, εργαλεία όπως το Spark έχουν κερδίσει δημοτικότητα υποστηρίζοντας συγκεκριμένες ανάγκες επεξεργασίας δεδομένων και αναλύσεων και χρησιμοποιούνται όλο και περισσότερο στις επιστήμες, την οικονομική ανάλυση των αγορών και την εξαγωγή γνώσεων για την οικοδόμηση επιχειρηματικών μοντέλων, φαρμακείων και πολλά άλλα.

Οι εφαρμογές που έχουν μεγάλο όγκο δεδομένων σχετίζονται με μεγάλη ποικιλία χαρακτηριστικών και ιδιοτήτων. Η πολυπλοκότητα και τα χαρακτηριστικά τους είναι αρκετά διαφορετικά από τις εφαρμογές HPC. Για παράδειγμα, συχνά περιλαμβάνουν πολλαπλά στάδια, όπως τη λήψη δεδομένων, την προεπεξεργασία, την εξαγωγή χαρακτηριστικών και την προηγμένη ανάλυση. Ενώ ορισμένα από αυτά τα στάδια είναι δεσμευμένα για I / O, συχνά με διαφορετικά πρότυπα (τυχαία / διαδοχική πρόσβαση), τα άλλα στάδια είναι υπολογισμός / δεσμευμένα στη μνήμη. Δεν αποτελεί έκπληξη το γεγονός ότι ένα διαφορετικό σύνολο εργαλείων για την επεξεργασία δεδομένων (π.χ. MapReduce, Spark RDDs), πρόσβαση σε πηγές δεδομένων (streaming, συστήματα αρχείων) και μορφές δεδομένων (μορφές επιστημονικών δεδομένων (HDF5), στήλες μορφής ORC και παρκέ), έχουν προκύψει και συχνά πρέπει να συνδυαστούν για να υποστηρίξουν τις τελικές ανάγκες των εφαρμογών.

Ορισμένες εφαρμογές, ωστόσο, αψηφούν την εύκολη ταξινόμηση ως έντασης δεδομένων ή HPC. Στην πραγματικότητα, υπάρχει ιδιαίτερο ενδιαφέρον για μια τάξη επιστημονικών εφαρμογών, όπως η βιομοριακή μοριακή δυναμική, που έχουν ισχυρά χαρακτηριστικά τόσο εντατικής όσο και HPC. Οι βιομοριακές προσομοιώσεις είναι τώρα υψηλής απόδοσης, φτάνουν σε αυξανόμενες χρονικές κλίμακες και μεγέθη προβλημάτων, δημιουργώντας έτσι τεράστια ποσά δεδομένων. Ο κύριος όγκος των δεδομένων σε τέτοιες προσομοιώσεις είναι συνήθως δεδομένα τροχιάς που είναι χρονικά επιλεγμένα σε συντεταγμένων και ταχύτητας. Τα δευτερεύοντα δεδομένα περιλαμβάνουν άλλες φυσικές παραμέτρους που περιλαμβάνουν διαφορετικά ενεργειακά στοιχεία. Συχνά, τα δεδομένα που παράγονται πρέπει να αναλυθούν έτσι ώστε να προσδιοριστεί το επόμενο σύνολο διαμορφώσεων προσομοίωσης. Ο τύπος της ανάλυσης ποικίλλει από τον υπολογισμό των στιγμών υψηλότερης τάξης, στα βασικά συστατικά, σε χρονικά εξαρτώμενες παραλλαγές.

Το MDAnalysis και το CPPTraj είναι δύο εργαλεία που εξελίχτηκαν για να ικανοποιήσουν τις αυξανόμενες απαιτήσεις για την ανάλυση δεδομένων από εφαρμογές μοριακής δυναμικής. Αυτά τα εργαλεία παρέχουν ισχυρές αναλύσεις για συγκεκριμένο τομέα. Μια πρόκληση είναι η ανάγκη να τα κλιμακώσουμε σε μεγάλους όγκους δεδομένων που παράγονται με μοριακές προσομοιώσεις καθώς και στη σύζευξη μεταξύ των τμημάτων προσομοίωσης και αναλύσεων. Αυτό υπογραμμίζει την ανάγκη για περιβάλλοντα που υποστηρίζουν κλιμακούμενη επεξεργασία δεδομένων διατηρώντας παράλληλα την ικανότητα να εκτελούν προσομοιώσεις στην κλίμακα έτσι ώστε να παράγουν τα δεδομένα.

Από όσο γνωρίζουμε, δεν υπάρχει λύση που να παρέχει τις ολοκληρωμένες δυνατότητες του Hadoop και HPC. Για παράδειγμα, η πλατφόρμα αναλύσεων Uray 1 του Cray έχει Hadoop και Spark που τρέχουν σε αρχιτεκτονική HPC σε αντίθεση με τα συνηθισμένα συμπλέγματα, αλλά χωρίς το περιβάλλον και τις δυνατότητες λογισμικού HPC. Ωστόσο, αρκετές εφαρμογές που κυμαίνονται από βιομοριακές προσομοιώσεις σε μοντέλα επιδημιολογίας [7] απαιτούν σημαντικές προσομοιώσεις με δυνατότητες ανάλυσης, όπως η συσσωμάτωση και η ανάλυση γραφημάτων. Με άλλα λόγια ορισμένα στάδια (ή τμήματα του ίδιου σταδίου) μιας εφαρμογής θα χρησιμοποιούσαν ιδανικά περιβάλλοντα Hadoop / Spark και άλλα στάδια (ή τμήματα αυτών) χρησιμοποιούν περιβάλλον HPC.

Τις τελευταίες δεκαετίες, η κοινότητα HPC (High Performance Distributed Computing) έχει σημειώσει σημαντική πρόοδο στην αντιμετώπιση της διαχείρισης πόρων και φόρτου εργασίας σε ετερογενείς πόρους. Για παράδειγμα, υιοθετήθηκε η έννοια του προγραμματισμού πολλαπλών επιπέδων [8], όπως αποκαλύπτεται στο διαχωρισμό της εκχώρησης φόρτου εργασίας από τη διαχείριση των πόρων χρησιμοποιώντας την έννοια των θέσεων εργασίας ενδιάμεσου εμπορευματοκιβωτίου (επίσης αναφέρεται ως Pilot-Jobs [9] Hadoop. Ο προγραμματισμός πολλαπλών επιπέδων είναι μια κρίσιμη δυνατότητα για εφαρμογές έντασης δεδομένων, καθώς συχνά μόνο οι προγραμματιστές επιπέδου εφαρμογής μπορούν να γνωρίζουν τις τοποθεσίες των πηγών δεδομένων που χρησιμοποιούνται από μια συγκεκριμένη εφαρμογή. Αυτό οδήγησε στην επέκταση του Pilot-Abstraction σε Pilot-Data [10] για να αποτελέσει το κεντρικό στοιχείο ενός ενδιάμεσου λογισμικού διαχείρισης πόρων.

## 1.2 Σύντομη περίληψη της διπλωματικής εργασίας

Σε αυτή την εργασία μελετάμε την απόδοση των αλγορίθμων σε HPC περιβάλλον χρησιμοποιώντας δύο διαφορετικούς διαχειριστές πόρων, τον Radical-Pilot, τον Apache Spark, τον οποίο ενσωματώσαμε στο Radical-Pilot. Επιλέξαμε να μελετήσουμε δύο διαχειριστές πόρων χρησιμοποιώντας τους δύο ακόλουθους αλγόριθμους, τον αλγόριθμο συσταδοποίησης k-means και τον leaflet-finder και τρέχουμε τα πειράματά μας σε δύο διαφορετικούς supercomputers. Οι αλγόριθμοι που χρησιμοποιήθηκαν σε αυτή τη μελέτη αναπτύχθηκαν από εμάς χρησιμοποιώντας το framework Radical-Pilot.

Τα παραπάνω πειράματα εκτελούνται για διαφορετικές παραμέτρους που σχετίζονται είτε με τα χαρακτηριστικά του συνόλου δεδομένων εισόδου (πολλαπλότητα περιπτώσεων, διαστάσεων) είτε με τον ίδιο τον αλγόριθμο (επαναλήψεις, συστοιχίες), είτε το περιβάλλον εκτέλεσης (πυρήνες ανά μηχανή, μνήμη ανά μηχανή). Αλγόριθμος ομαδοποίησης. Ομοίως, και για τον leaflet-finder, τα πειράματά μας εκτελούνται χρησιμοποιώντας διαφορετικά

σύνολα δεδομένων, τα οποία είναι διαφορετικά πρωτότυπα σενάρια σε πολλά περιβάλλοντα εκτέλεσης (πυρήνες ανά μηχανή, μνήμη ανά μηχανή). Ο σκοπός αυτών των επιδόσεων είναι να μετρηθεί ο απαιτούμενος χρόνος για τον αλγόριθμο ολοκλήρωσης.

Ο σκοπός της συλλογής τέτοιων δεδομένων είναι ένα πρώτο βήμα για την κατανόηση της συμπεριφοράς εκτέλεσης αυτών των αλγορίθμων και στους δύο διαχειριστές πόρων. Στη συνέχεια, θέλουμε να παρατηρούμε τη συμπεριφορά και την απόδοση και των δύο αλγορίθμων σε διαφορετικές μηχανές που έχουν διαφορετικά χαρακτηριστικά, ώστε να μπορούμε να βρούμε συνθήκες που ευνοούν τη χρήση ενός σε σχέση με τον άλλο διαχειριστή.

Όσον αφορά τη διαδικασία εξαγωγής της μέτρησης του χρόνου, χρησιμοποιούμε τη βιβλιοθήκη του Python `datetime`. Για να υπολογίσουμε το πειραματικό σφάλμα εκτελούμε κάθε πείραμα 2-3 φορές για να ανιχνεύσουμε πιθανές αποκλίσεις και να χρησιμοποιήσουμε ως αποτέλεσμα τη μέση τιμή.

### 1.3 Διάρθρωση της διπλωματικής εργασίας

Στο Κεφάλαιο 2 παρουσιάζουμε το γενικό θεωρητικό υπόβαθρο που σχετίζεται με τον αλγόριθμο `k-means`, το αλγόριθμο `leaflet-finder`, τα επιστημονικά εργαλεία που θα χρησιμοποιήσουμε σε αυτό το έργο, καθώς και τις θεωρητικές γνώσεις των εργαλείων που χρησιμοποιούμε. Αυτά θεωρούνται απαραίτητα για τη μελέτη που θα ακολουθήσει στις επόμενες μονάδες εργασίας.

Στο Κεφάλαιο 3 επικεντρωνόμαστε στη μελέτη των επιδόσεων του αλγορίθμου `k-means` στο `radical pilot` και στον `pilot-spark`. Αρχικά αναφέρει την πειραματική διαδικασία που ακολουθήθηκε σε σχέση με τις επιλεγμένες επιδόσεις που ελέγχθηκαν. Παρουσιάζουμε επίσης τα αποτελέσματα των μετρήσεών μας σε όλες τις εκτελέσεις και αναλύουμε την συμπεριφορά απόδοσης ως μεταβάλλοντας οποιαδήποτε εκτέλεση παραμέτρων.

Στο Κεφάλαιο 4, ακολουθούμε την ίδια διαδικασία όπως στο Κεφάλαιο 3, αυτή τη φορά για το `leaflet-finder` αλγόριθμο μοριακής δυναμικής. Πραγματοποιούμε πάλι πειράματα χρησιμοποιώντας τον `radical pilot` και τον `pilot-spark` και έχουμε αρχικά δεδομένα ως είσοδο στον αλγόριθμο. Στη συνέχεια παρουσιάζουμε τα αποτελέσματα των μετρήσεων μας για κάθε δοκιμασμένο σενάριο.

Στο Κεφάλαιο 5 συνοψίζουμε τα συμπεράσματα που συνάγουμε σε όλη τη μελέτη μας και σχολιάζουμε οποιαδήποτε από αυτές τις επεκτάσεις εργασίας.

Στο Κεφάλαιο 6 αναφέρουμε τις μηχανές που χρησιμοποιήθηκαν σε αυτή τη μελέτη και τα τεχνικά χαρακτηριστικά τους.

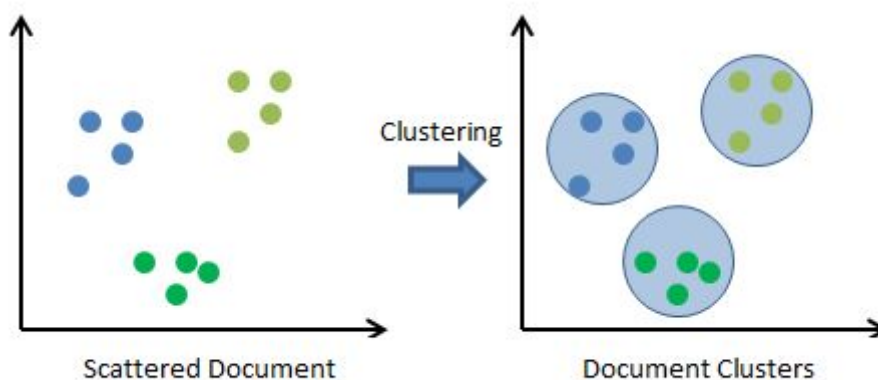
## Κεφάλαιο 2

### 2.1 Συσταδοποίηση

#### 2.1.1 Ανάλυση Συσταδοποίησης

Η ομαδοποίηση [13] είναι ένα πολύ σημαντικό πρόβλημα στη μάθηση χωρίς επίβλεψη. Όπως κάθε πρόβλημα αυτού του είδους, η ομαδοποίηση προσπαθεί να δομήσει μια συλλογή τυχαίων δεδομένων. Ένας χαλαρός ορισμός της ομαδοποίησης θα μπορούσε να είναι: η διαδικασία της οργάνωσης των δεδομένων σε ομάδες με βάση παρόμοια χαρακτηριστικά.

Μπορούμε να το δείξουμε με ένα απλό γραφικό παράδειγμα:



Σχήμα 1: Εικόνα Συσταδοποίησης

Στο σχήμα 1 μπορούμε να προσδιορίσουμε τα 3 σύνολα στα οποία έχει χωριστεί το σύνολο δεδομένων. Το κριτήριο ομοιότητας που χρησιμοποιήθηκε για την ταξινόμηση είναι η απόσταση. Δύο ή περισσότερα στοιχεία ανήκουν στο σύμπλεγμα X εάν η απόσταση τους από το σύμπλεγμα X είναι μικρότερη από την απόσταση από το σύμπλεγμα Y.

Ενώ η συσταδοποίηση αντικειμένων του φυσικού κόσμου που κατέχει το μυαλό των ανθρώπων είναι μια ασυνείδητη και αυτόματη διαδικασία, στον κόσμο των υπολογιστών που έχει τις έννοιες μηχανικής μάθησης που απαιτούνται για να καθορίσουμε μαθηματικά μοντέλα για να τα ταξινομήσουμε σε ομάδες.

Αυτό συνήθως σχετίζεται με την αναπαράσταση αντικειμένων που ενδιαφέρουν σε φορείς σε  $n$ -διάστατο ευκλείδειο χώρο όπου η καρδιανότητα της διάστασης του φορέα αντικατοπτρίζει τον αριθμό των χαρακτηριστικών - ιδιότητες που περιγράφουν το θέμα και τις αντίστοιχες τιμές του.

Ποσοτικοποίηση σε μια επιλεγμένη περιοχή. Από τότε τα αντικείμενα είναι φορείς στην έννοια του χώρου ομοιότητας λογικά χαρτογραφημένο σε ένα πρότυπο ευκλείδειου χώρου.

Μεταξύ των πολλών επιλογών η πιο κοινή ανησυχία ευκλείδειων απόσταση, απόσταση από το Μανχάταν και απόσταση Cosine. Οι υπολογισμοί των εξισώσεων των παραπάνω παρουσιάζονται παρακάτω για τους φορείς X και Y.

$$d_{euclidean}(X, Y) = \|X - Y\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}$$

$$d_{manhattan}(X, Y) = \|X - Y\|_1 = |x_1 - y_1| + |x_2 - y_2| + \dots + |x_n - y_n|$$

$$d_{cosine}(X, Y) = 1 - \frac{X*Y}{\|X\|_2 * \|Y\|_2}$$

Έτσι, ο στόχος της ομαδοποίησης είναι να προσδιοριστεί σε ποια ομάδα ένα σύνολο μη επισημασμένων δεδομένων μπορεί να ταιριάζει καλύτερα. Αλλά πώς μπορούμε να αποφασίσουμε ποιο είναι το καλύτερο κριτήριο επιλογής για την ομαδοποίηση; Δεν υπάρχει επιστημονική απόδειξη για την επιλογή ενός κριτηρίου ομαδοποίησης, οπότε ο χρήστης είναι υπεύθυνος για την παροχή αυτού του κριτηρίου, με τέτοιο τρόπο ώστε το αποτέλεσμα της ομαδοποίησης να ταιριάζει στις ανάγκες του. Για παράδειγμα, θα μπορούσαμε να βρούμε εκπροσώπους για ομοιογενείς ομάδες (μείωση των δεδομένων), να βρίσκουμε "φυσικές συστάδες" και να περιγράψουμε τις άγνωστες ιδιότητες τους ("φυσικούς" τύπους δεδομένων), βρίσκοντας χρήσιμες και κατάλληλες ομάδες ("χρήσιμες" ή στην εξεύρεση ασυνήθιστων αντικειμένων δεδομένων (ανίχνευση εξωστρέφειας).

Οι αλγόριθμοι ομαδοποίησης μπορούν να εφαρμοστούν σε πολλά πεδία. Για παράδειγμα:

**Μάρκετινγκ:** εύρεση ομάδων πελατών με παρόμοια συμπεριφορά δεδομένης μιας μεγάλης βάσης δεδομένων πελατών που περιέχουν τις ιδιότητές τους και προηγούμενα αρχεία αγοράς.

**Βιολογία:** ταξινόμηση φυτών και ζώων με βάση τα χαρακτηριστικά τους.

Βιβλιοθήκες: παραγγελία βιβλίων.

**Ασφάλιση:** προσδιορισμός ομάδων κατόχων ασφαλιστηρίων συμβολαίων αυτοκινήτων με υψηλό μέσο κόστος απαιτήσεων. Εντοπίζοντας τις απάτες.

**Πολεοδομία:** προσδιορισμός ομάδων κατοικιών ανάλογα με τον τύπο κατοικίας, την αξία και τη γεωγραφική τους θέση.

**Σεισμικές μελέτες:** συσσωμάτωση παρατηρημένων σεισμικών επιφανειών για τον εντοπισμό επικίνδυνων ζωνών.

**WWW:** ταξινόμηση εγγράφων. Συσσωρεύοντας δεδομένα weblog για να ανακαλύψετε ομάδες παρόμοιων προτύπων πρόσβασης.

Ανεξάρτητα από τη λειτουργία μοντελοποίησης και επιλογής απόστασης, η ομαδοποίηση κατηγοριοποιείται σε επίπεδα και ιεραρχικά ανάλογα με το τελικό αποτέλεσμα. Συγκεκριμένα, η ομαδική συσσωμάτωση αναφέρεται σε μια πλήρως ανεξάρτητη συστοιχία άρθρων με



μηδενική αλληλεξάρτηση, η οποία στοχεύει να επισημάνει μόνο ομοιότητες αντικειμένων μόνο μέλη του ίδιου συμπλέγματος. Η δεύτερη ομαδοποίηση της κατηγορίας στοχεύει στην τοποθέτηση μπλοκ σε μορφή δέντρων, τη ρίζα που συγκροτούμε περιγράφοντας γενικευμένες ομάδες και ενδιάμεσους κόμβους. Είναι πιο συγκεκριμένες ομάδες. Στην τελευταία κατηγορία εμφανίζονται ιδανικά οι λέξεις δεντρογράμματα. Επιπλέον, η ομαδοποίηση μπορεί να χαρακτηριστεί ολική ή μερική. Στην ολική συσσώρευση του τελικού στόχου είναι ένα από τα στοιχεία εισόδου της διαδικασίας να είναι μέλος ενός συμπλέγματος. Σε αντίθεση με μερικούς μπορεί να είναι και τα άρθρα που δεν περιέχονται σε ένα μπλοκ μετά την ομαδοποίηση. Η τελευταία κατηγορία είναι η λογική επιλογή όταν θέλουμε να απομονώσουμε αντικείμενα στο διάστημα είναι ουσιαστικά διαχωρίσιμα από τα υπόλοιπα και λειτουργούν σαν ένα είδος θορύβου.

Από τα παραπάνω προκύπτει ότι η συσπείρωση εξαρτάται στενά από τον τύπο και τα χαρακτηριστικά όλων των αντικειμένων στα οποία εφαρμόστηκε. Το επακόλουθο αυτών είναι η ύπαρξη ενός πλήθους αλγορίθμων, με πολύ διαφορετικές πολυπλοκότητες. Η γενική αξία και η αναγκαιότητα της διαδικασίας, ωστόσο, είναι αυτή που την καθιστά χρήσιμη σε ένα ευρύ φάσμα επιστημονικών πεδίων, όπως η μάθηση μηχανών, η αναγνώριση προτύπων, η ανάλυση εικόνων και η εξόρυξη δεδομένων βίντεο, η βιοπληροφορική. Σε αυτό το πρόγραμμα θα επικεντρώσουμε το ενδιαφέρον μας σε έναν πολύ γνωστό αλγόριθμο ομαδοποίησης που ονομάζεται n-μέσος όρος (k-means), τον οποίο θα παρουσιάσουμε στο Κεφάλαιο 3.

## 2.1.2 Αλγόριθμος Συσταδοποίησης

Ο k-means αλγόριθμος προτάθηκε πρώτα από τον Stuart Lloyd το 1957 ως μια τεχνική για την κωδικοποίηση παλμικού κώδικα (κωδικοποίηση παλμικού κώδικα) για αυτό και πολλές φορές βρέθηκε και ονομάστηκε αλγόριθμος Lloyd. Ο αλγόριθμος εκτελεί αντικείμενα ομαδοποίησης σε n-διάστατο ευκλείδειο χώρο με τον ευκλείδειο κανόνα που χρησιμοποιείται συχνότερα ως συνάρτηση απόστασης.

Ανήκει στην κατηγορία αλγορίθμων που εκτελούν επίπεδη, ολική συσσώρευση, ενώ η αναπαράσταση των κερκίδων ανήκει σε πρωτότυπο που αναφέρεται στο προηγούμενο τμήμα. Συγκεκριμένα, ο εκπρόσωπος κάθε συστάδας θεωρείται ένα λεγόμενο κεντροειδές (centroid) που υπολογίζεται ως ένας φορέας που είναι η μέση τιμή των διανυσμάτων που αποτελούν τη συστάδα.

Ο αλγόριθμος, δεδομένου ενός συνόλου ή αντικειμένων κατάλληλα διαμορφωμένων ως διανυσμάτων με διαστάσεις  $d$  που αντιπροσωπεύουν τους χαρακτήρες τους, εκτελεί επανειλημμένα τα ακόλουθα τρία βήματα:

Το πρώτο βήμα επιλέγει τα αρχικά κεντροειδή, με την πιο βασική μέθοδο να επιλέγονται δείγματα από το σύνολο δεδομένων. Μετά την αρχικοποίηση, το k-μέσον αποτελείται από βρόχο μεταξύ των δύο άλλων βημάτων. Το πρώτο βήμα αναθέτει κάθε δείγμα στο πλησιέστερο κέντρο. Το δεύτερο βήμα δημιουργεί νέα κεντροειδή, λαμβάνοντας τη μέση τιμή όλων των δειγμάτων που αντιστοιχούν σε κάθε προηγούμενο κεντροειδές. Η διαφορά μεταξύ των παλαιών και των νέων κεντροειδών υπολογίζεται και ο αλγόριθμος επαναλαμβάνει αυτά

τα δύο τελευταία βήματα μέχρις ότου αυτή η τιμή είναι μικρότερη από ένα όριο. Με άλλα λόγια, επαναλαμβάνει έως ότου τα κεντροειδή δεν κινηθούν σημαντικά.

Ο αλγόριθμος μπορεί επίσης να γίνει κατανοητός από την έννοια των διαγραμμάτων Voronoi. Αρχικά το διάγραμμα Voronoi των σημείων υπολογίζεται χρησιμοποιώντας τα τρέχοντα κεντροειδή. Κάθε τμήμα στο διάγραμμα Voronoi γίνεται ξεχωριστό σύμπλεγμα. Δεύτερον, τα κεντροειδή ενημερώνονται με τον μέσο όρο κάθε τμήματος. Στη συνέχεια ο αλγόριθμος επαναλαμβάνει αυτό μέχρις ότου εκπληρωθεί ένα κριτήριο διακοπής. Συνήθως, ο αλγόριθμος σταματά όταν η σχετική μείωση της αντικειμενικής συνάρτησης μεταξύ των επαναλήψεων είναι μικρότερη από τη δεδομένη τιμή ανοχής. Αυτό δεν συμβαίνει σε αυτή την εφαρμογή: η επανάληψη σταματά όταν τα κεντροειδή μετακινούν λιγότερο από την ανοχή.

Η αντικειμενική λειτουργία που προσπαθούμε να ελαχιστοποιήσουμε σε αυτή την περίπτωση είναι μια λειτουργία σφάλματος:

$$Obj = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

όπου :

$\sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$  Είναι ένα επιλεγμένο μέτρο απόστασης μεταξύ ενός σημείου δεδομένων  $x_i^{(j)}$

και του κέντρου συμπλέγματος  $c_j$ , , είναι ένας δείκτης της απόστασης των σημείων δεδομένων η από τα αντίστοιχα κέντρα συμπλέγματος.

Ως αναφορά στην χρονική πολυπλοκότητα του αλγορίθμου, υπολογίζεται εύκολα και είναι  $O(nkd)$ , όπου όπου  $n$  είναι ο αριθμός των σημείων που εμπλέκονται στη διαδικασία,  $k$  ο αριθμός των συστάδων,  $d$  ο αριθμός των διαστάσεων του κάθε σημείου και τέλος με τον αριθμό των επαναλήψεων  $i$ . Στις περισσότερες κοινές εφαρμογές, τα  $k$ ,  $d$  και  $i$  είναι σημαντικά μικρότερα από το  $n$ , ώστε μπορούμε να πούμε ότι ο αλγόριθμος είναι σχεδόν γραμμικός ως ο αριθμός αναφοράς των διανυσμάτων εισόδου. Η χωρική πολυπλοκότητα είναι επίσης εύκολα μετρήσιμη και με δεδομένο το  $O((n k) d)$ , το οποίο αντικατοπτρίζει ακριβώς την ανάγκη του αλγορίθμου να είναι διαθέσιμη στη μνήμη τουλάχιστον όλων των δεδομένων συν το κέντρο της κάθε επανάληψης.

Ο αλγόριθμος  $k$ -μέσων, όπως βλέπουμε, είναι μια εύκολη και αξιόπιστη λύση για το πρόβλημα της ομαδοποίησης και η ελκυστικότητα αυτού είναι ότι έχει οδηγήσει σε πολλές παραλλαγές και ευρεία χρήση σε πολλές διαφορετικές περιοχές. Ακόμη και ένα σημαντικό πλεονέκτημα του αλγορίθμου που αξίζει να αναφερθεί είναι ότι είναι "ευχάριστα" παράλληλο. Συγκεκριμένα, όπως φαίνεται από την προηγούμενη περιγραφή της διαδικασίας υπολογισμού του αλγορίθμου επικεντρώνεται κυρίως στην εκχώρηση κάθε αντικειμένου με το πλησιέστερο κέντρο και κάθε φορά είναι ανεξάρτητη με τους αντίστοιχους υπολογισμούς για κάθε άλλο σημείο. Ότι η ιδιότητα τον καθιστά ιδιαίτερα δημοφιλές σε καταναμημένα συστήματα όπου η είσοδος του συνόλου δεδομένων χωρίζεται σε μικρότερα κομμάτια και οι υπολογισμοί που πραγματοποιούνται από διάφορους υπολογιστικούς κόμβους. Ο μόνος περιορισμός είναι οι απλοί κόμβοι ενημέρωσης κάθε φορά με ένα νέο σύνολο κεντροειδών, αλλά κάτι που γίνεται εύκολα και αποτελεσματικά από τα σύγχρονα καταναμημένα μοντέλα

προγραμματισμού. Αυτός είναι ο λόγος για τον οποίο ο αλγόριθμος k-means αυτών των περιβαλλόντων υπερέρχει σε σχέση με ανταγωνιστικούς αλγορίθμους.

## 2.2 Μοριακή δυναμική

### 2.2.1 Ανάλυση Μοριακής δυναμικής

Η μοριακή δυναμική [16] (MD) είναι μια μέθοδος προσομοίωσης υπολογιστών για τη μελέτη των φυσικών κινήσεων ατόμων και μορίων, και είναι επομένως ένας τύπος προσομοίωσης N-σώματος. Τα άτομα και τα μόρια επιτρέπεται να αλληλεπιδρούν για μια καθορισμένη χρονική περίοδο, δίνοντας μια εικόνα της δυναμικής εξέλιξης του συστήματος. Στην πιο συνηθισμένη εκδοχή, οι τροχιές των ατόμων και των μορίων καθορίζονται με αριθμητική επίλυση των εξισώσεων κίνησης του Νεύτωνα για ένα σύστημα αλληλεπιδρώντων σωματιδίων, όπου οι δυνάμεις μεταξύ των σωματιδίων και των δυναμικών ενεργειών τους υπολογίζονται χρησιμοποιώντας διατομικά δυναμικά ή πεδία δύναμης μοριακής μηχανικής. Η μέθοδος αναπτύχθηκε αρχικά στο πεδίο της θεωρητικής φυσικής στα τέλη της δεκαετίας του 1950, αλλά σήμερα εφαρμόζεται κυρίως στη χημική φυσική, στην επιστήμη των υλικών και στη μοντελοποίηση βιομορίων.

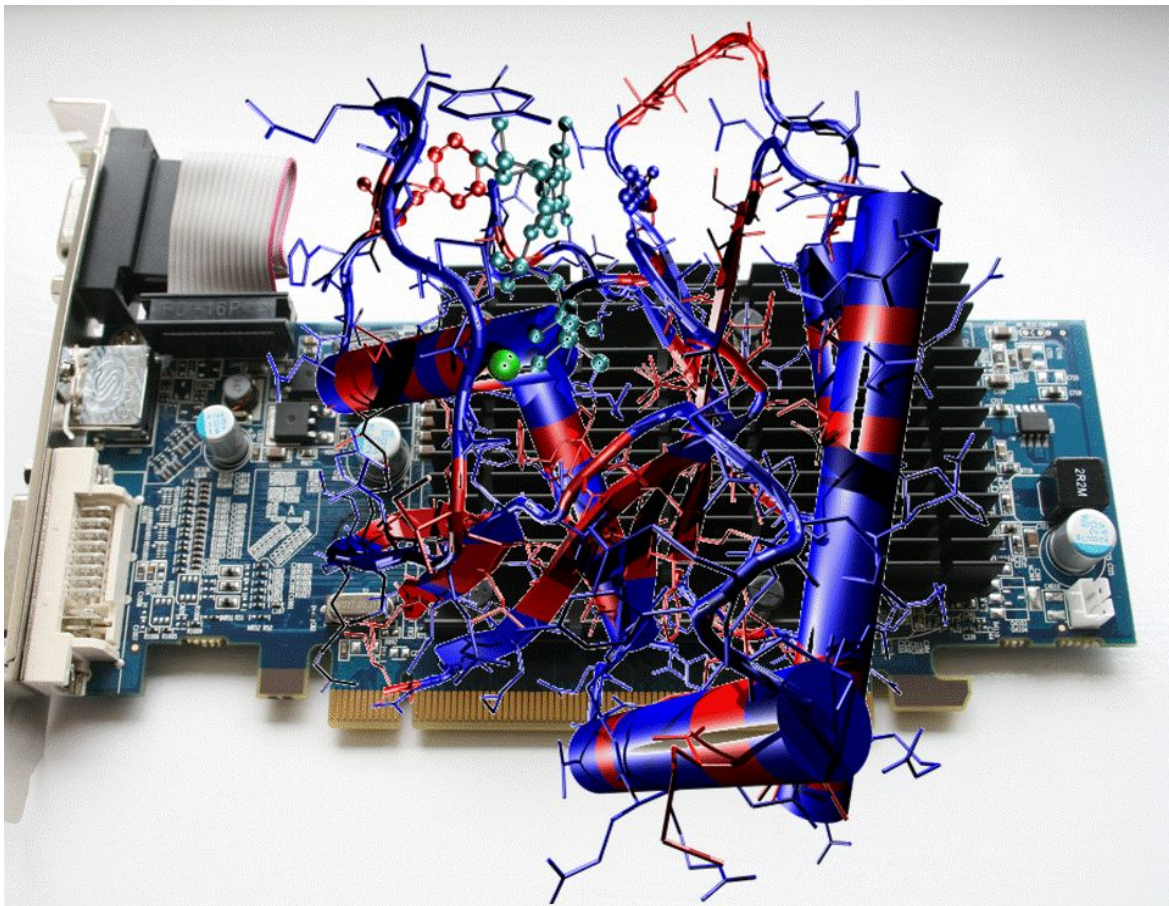
Επειδή τα μοριακά συστήματα τυπικά αποτελούνται από έναν τεράστιο αριθμό σωματιδίων, είναι αδύνατον να προσδιοριστούν αναλυτικά οι ιδιότητες τέτοιων σύνθετων συστημάτων. Η προσομοίωση MD παρακάμπτει αυτό το πρόβλημα χρησιμοποιώντας αριθμητικές μεθόδους. Ωστόσο, μακρές προσομοιώσεις MD είναι μαθηματικά άρρωστοι, δημιουργώντας αθροιστικά σφάλματα στην αριθμητική ολοκλήρωση που μπορούν να ελαχιστοποιηθούν με κατάλληλη επιλογή αλγορίθμων και παραμέτρων, αλλά δεν εξαλείφονται πλήρως.

Για συστήματα που υπακούουν στην εικονολογική υπόθεση, η εξέλιξη μίας μονομοριακής δυναμικής προσομοίωσης μπορεί να χρησιμοποιηθεί για τον προσδιορισμό μακροσκοπικών θερμοδυναμικών ιδιοτήτων του συστήματος: οι μέσοι χρόνοι ενός συστήματος ergodic αντιστοιχούν σε μέσους όρους του μικροκανονικού συνόλου. Η MD έχει επίσης ονομαστεί "στατιστική μηχανική με αριθμούς" και "το όραμα του Laplace για Νευτώνεια μηχανική" για την πρόβλεψη του μέλλοντος με την κίνηση των δυνάμεων της φύσης και την κατανόηση της μοριακής κίνησης σε ατομική κλίμακα.

#### Περιορισμοί

Ο σχεδιασμός μιας προσομοίωσης μοριακής δυναμικής θα πρέπει να λαμβάνει υπόψη την διαθέσιμη υπολογιστική ισχύ. Το μέγεθος προσομοίωσης ( $n$  = αριθμός σωματιδίων), το χρονικό διάστημα και η συνολική χρονική διάρκεια πρέπει να επιλέγονται έτσι ώστε ο υπολογισμός να μπορεί να ολοκληρωθεί μέσα σε εύλογο χρονικό διάστημα. Ωστόσο, οι προσομοιώσεις πρέπει να είναι αρκετά μεγάλες ώστε να είναι σχετικές με τις χρονικές κλίμακες των φυσικών διεργασιών που μελετώνται. Για να γίνουν στατιστικά έγκυρα συμπεράσματα από τις προσομοιώσεις, το προσομοιωμένο χρονικό διάστημα θα πρέπει να ταιριάζει με την κινητική της φυσικής διαδικασίας. Διαφορετικά, είναι ανάλογο να κάνουμε συμπεράσματα για το πώς ένας άνθρωπος περπατάει όταν βλέπει μόνο λιγότερο από ένα βήμα. Οι περισσότερες επιστημονικές δημοσιεύσεις σχετικά με τη δυναμική των πρωτεϊνών

και τα δεδομένα χρήσης DNA από προσομιώσεις που εκτείνονται σε νανοδευτερόλεπτα ( $10^{-9}$  δευτερόλεπτα) σε μικροδευτερόλεπτα ( $10^{-6}$  s). Για να λάβετε αυτές τις προσομιώσεις, απαιτούνται αρκετές ημέρες CPU σε χρόνια CPU. Παράλληλοι αλγόριθμοι επιτρέπουν τη διανομή του φορτίου μεταξύ CPU. Ένα παράδειγμα είναι ο χωροταξικός αλγόριθμος ή ο αλγόριθμος αποσύνθεσης της δύναμης.



Σχήμα 2: Υπολογιστής Μοριακής δυναμικής

## 2.2.2 Αλγόριθμος Leaflet-Finder

Για την παρούσα εργασία αναπτύξαμε έναν αλγόριθμο μοριακής δυναμικής, του οποίου το όνομα είναι leaflet-finder [17]. Αυτός ο αλγόριθμος στοχεύει στην αναγνώριση του άνω και κάτω φύλλου μιας επίπεδης μεμβράνης ή του εξωτερικού και του εσωτερικού φύλλου ενός κυστιδίου, συγκρίνοντας ιστόγραμμα αποστάσεων από το κέντρο της γεωμετρίας.

Ο αλγόριθμος λαμβάνει ως είσοδο τρισδιάστατες καρτεσιανές συντεταγμένες οι οποίες είναι οι συντεταγμένες του φωσφορικού. Το πρώτο βήμα είναι ο υπολογισμός των αποστάσεων όλων των ζευγών όλων των φωσφορικών αλάτων. Το κάνουμε αυτό δημιουργώντας ένα δισδιάστατο πίνακα γειτνίασης  $n \times n$ , όπου  $n$  είναι ο αριθμός των φωσφορικών. Ως ευριστική

μμέθοδος χρησιμοποιούμε τις ευκλείδεια απόσταση Το δεύτερο βήμα είναι να θέσουμε μια μέθοδο χρησιμοποιούμε την ευκλείδεια απόσταση αποκοπής. Εάν μια απόσταση είναι μεγαλύτερη από την αποκοπή, τότε την θέτουμε στο άπειρο. Το δεύτερο βήμα είναι να υπολογίσουμε τα συνδεδεμένα στοιχεία του γραφήματος. Το κάνουμε αυτό χρησιμοποιώντας τη βιβλιοθήκη networkx [11]. Η βιβλιοθήκη networkx λαμβάνει ως είσοδο τον πίνακα γειννίας που δημιουργήσαμε νωρίτερα και επιστρέφει τα συνδεδεμένα στοιχεία του γραφήματος. Το τρίτο βήμα είναι να τα ταξινομήσετε και να επιστρέψετε τα μεγαλύτερα και δεύτερα μεγαλύτερα στοιχεία του γραφήματος που αντιστοιχούν στα απαιτούμενα φυλλάδια.

Ο υπολογισμός των αποστάσεων είναι μια εργασία που απαιτεί υπολογισμό και μνήμη. Ο πίνακας γειννίας απαιτεί  $n^2$  χώρο. Με αυτό είπαμε να υπολογίσουμε τις αποστάσεις παράλληλα χρησιμοποιώντας ριζοσπαστικό πιλοτικό πλαίσιο. Αναθέσαμε σε κάθε μονάδα μέτρησης μικρότερο αριθμό υπολογισμών, ώστε να μπορούμε να έχουμε λιγότερες συγκρίσεις ανά πυρήνα και να χρησιμοποιήσουμε λιγότερη μνήμη, ώστε όλα τα στοιχεία να χωρέσουν στον κριάρι των μηχανών μας.

$$d_{euclidean} = |\hat{x}_1 - \hat{x}_2|, \text{ where } \hat{x} = (x, y, z)$$

---

Αλγόριθμος:

1. Build a graph of all phosphate distances < cutoff
  2. Identify the largest connected subgraphs
  3. Analyse first and second largest graph, which correspond to the leaflets
- 

## 2.3 Επιστημονικά Εργαλεία

### 2.3.1 Apache Spark

#### 2.3.1.1 Εισαγωγή

Το Apache Spark είναι μια υπολογιστική πλατφόρμα ειδικά σχεδιασμένη για cluster υπολογιστών (cluster computing platform) που υλοποιείται στη γλώσσα προγραμματισμού Scala. Κατασκευασμένο για να υποστηρίζει καταναμημένες εφαρμογές γενικού σκοπού γενικά βασίζεται στην επεξεργασία μεγάλων όγκων δεδομένων με υψηλό βαθμό αποτελεσματικότητας και ταχύτητας.

Σε ένα περιβάλλον του Spark είναι ένα μοντέλο MapReduce που αναπτύσσει επέκταση, με την κύρια διαφορά που υποστηρίζει τους περισσότερους τύπους υπολογισμών, όπως

διαλογικές ερωτήσεις (διαδραστικά ερωτήματα) και επεξεργασία δεδομένων ροής δεδομένων (επεξεργασία δεδομένων ροής). Μια άλλη διαφορά του Spark σε σχέση με τις εφαρμογές του MapReduce σε άλλα συστήματα όπως Hadoop, είναι η δυνατότητα να παρέχεται η αποθήκευση δεδομένων στη μνήμη κάθε κόμβου του συμπλέγματος κατά την εκτέλεση της εργασίας. Αυτή η τελευταία ιδιότητα που περιγράφεται ως προσωρινή μνήμη και είναι πιθανώς ένα από τα πιο σημαντικά χαρακτηριστικά που εισήγαγε το Spark στην ανάπτυξη καταμεμημένων συστημάτων και δίνει πλεονέκτημα σε ταχύτητα συγκριτικά με το Hadoop, επιτυγχάνοντας ακόμη και έως 100 φορές την εκτέλεση εφαρμογής.

Η δομή Spark περιγράφεται καλύτερα ως μια ενιαία στοίβα υποσυστημάτων που συνεργάζονται και παρέχουν καθεμία από αυτές τις υπηρεσίες ξεχωριστά. Συγκεκριμένα, η καρδιά του συστήματος είναι ο πυρήνας του Spark. Η βασική ενότητα είναι το υπολογιστικό σύστημα του κινητήρα που είναι υπεύθυνο για τις εφαρμογές δρομολόγησης, διαμοιρασμού και παρακολούθησης, οι οποίες αναλύονται λεπτομερώς στις επιμέρους μονάδες (εργασίες) υπολογιστών και εκχωρούνται στους κόμβους του cluster. Ο πυρήνας είναι επίσης αυτό που παρέχει διασυνδέσεις για την δημιουργία προγραμματικών στοιχείων του συστήματος, όπως για παράδειγμα τα Resilient Distributed Datasets (RDD) που θα αναλύσουμε αργότερα. Είναι επίσης η βάση πάνω στην οποία βασίζονται τα υποσυστήματα Spark Sql, Spark Streaming, MLlib και GraphX. Αναφέρουμε εν συντομία ότι το Spark Sql προσφέρει δυνατότητες συνεργασίας με τα δομημένα δεδομένα του Spark μέσω της γλώσσας Sql και της παραλλαγής που προσφέρει το Apache Hive. Το Spark Streaming είναι αυτό που δίνει τη δυνατότητα αποθήκευσης και ροής δεδομένων σε πραγματικό χρόνο. Το GraphX είναι μια ειδική βιβλιοθήκη που έχει σχεδιαστεί για να παρέχει υποστήριξη για εργασίες και αλγόριθμους που επεξεργάζονται γραφήματα. Τέλος MLlib τη βιβλιοθήκη μια συλλογή από αλγόριθμους μηχανικής μάθησης. Το MLlib έχει επίσης μια εφαρμογή k-means, αλλά σε αυτό το έργο θα χρησιμοποιήσουμε μια έκδοση k-means που εφαρμόσαμε.

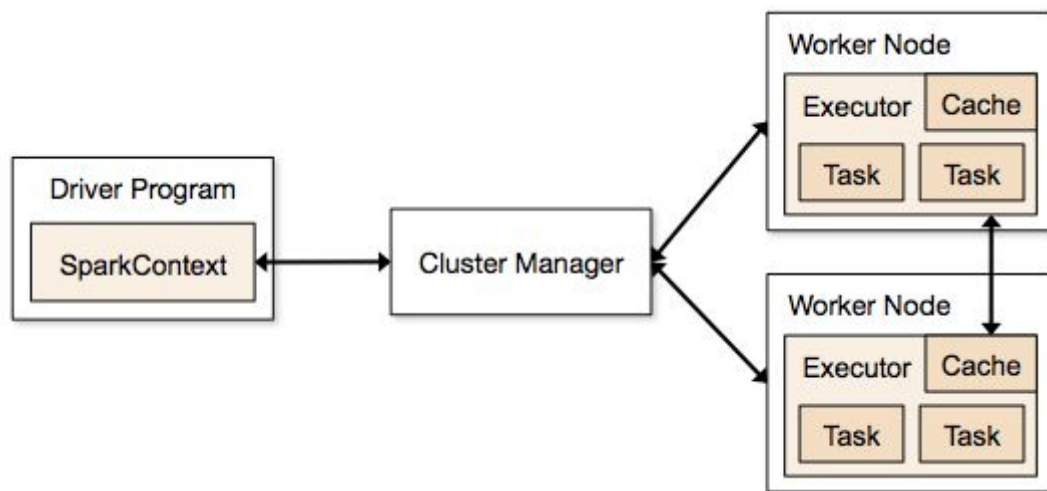
Το Spark, όπως αναφέρθηκε παραπάνω, υλοποιείται εξ ολοκλήρου στη γλώσσα προγραμματισμού Scala, Java ένα πολυγλωσσικό παράδειγμα με ισχυρές ενδείξεις για τον λειτουργικό και αντικειμενοστραφή προγραμματισμό. Χρησιμοποιεί το περιβάλλον JVM για την εκτέλεση των προγραμμάτων της και έχει σχεδιαστεί έτσι ώστε να μπορεί να εισάγει και να χρησιμοποιεί βιβλιοθήκες της Java. Το Spark υποστηρίζει φυσικά την ανάπτυξη εφαρμογών σε οποιαδήποτε από τις γλώσσες Java και Scala και Python. Αυτό καθιστά την πλατφόρμα Spark ανεξάρτητη (πλατφόρμα ανεξάρτητη), ενώ με την Python παρέχει μεγάλη ελευθερία επιλογής από την άποψη του χρήστη.

### 2.3.1.2 Αρχιτεκτονική του Spark

Η αρχιτεκτονική του Spark βασίζεται στο διάσημο προγραμματιστικό μοντέλο MapReduce που ακολουθεί το αρχιτεκτονικό μοντέλο master / slave. Πιο συγκεκριμένα στην ορολογία του Spark υπάρχουν ο master και οι workers (slave) . Όταν ξεκινάτε ένα σύμπλεγμα Spark, εκτελείται μια κύρια διαδικασία στον κόμβο από τον οποίο δίνεται η εντολή start, ενώ οι διεργασίες των workers εκτελούνται σε κόμβους των εργατών.

Για να εκτελέσετε μια εφαρμογή Spark πρέπει να υπολογίσετε τους πόρους από το σύμπλεγμα, να μετατραπούν σε κεντρικούς πυρήνες μνήμης και cpu. Ένας επιτηρητής των διαθέσιμων πόρων του cluster παρέχει στην Spark την επιλογή μεταξύ των διαχειριστών

πόρων των ανεξάρτητων (standalone) , και του Mesos. Το αυτόνομο, το οποίο θα χρησιμοποιήσουμε σε αυτό το έργο, και έναν ολοκληρωμένο διαχειριστή πόρων εφαρμογής που παρέχεται από το Apache Spark. Οι διαδικασίες κυρίων και εργαζόμενων ανήκουν στον ανεξάρτητο διαχειριστή πόρων ο οποίος έχει την εποπτεία όλων των διαδικασιών και των πόρων ενός αποθέματος του συγκεκριμένου συμπλέγματος. Εκτός αυτού, πρέπει να διατηρήσουμε τη συνεχή επικοινωνία μεταξύ των κόμβων των εργαζομένων και του κύριου κόμβου όχι μόνο για την επιτυχή εκτέλεση της εφαρμογής, αλλά και για να εντοπίσουμε τυχόν προβλήματα. Τέλος, αυτές οι διαδικασίες δημιουργούν διακομιστές σε κάθε κόμβο για να παρέχουν χρήσιμες πληροφορίες για τη λειτουργία τους μέσω του πρωτοκόλλου http. Τα βασικά συστατικά μιας εφαρμογής Spark είναι ο οδηγός του εκτελεστή και του διαχειριστή συμπλέγματος.



## 1. Executors

Οι (εκτελεστές) executors [14] είναι διαδικασίες Java που ξεκινούν τους εργαζόμενους στους κόμβους συμπλέγματος. Ο εκτελεστής ξεκίνησε από τον κύριο κόμβο στην αρχή κάθε εργασίας και η διάρκεια ζωής είναι τόσο μεγάλη όσο και η διάρκεια της εργασίας. Οι εκτελεστές είναι αυτοί που συμμετέχουν στον υπολογισμό που απαιτείται για την εργασία. Αυτό γίνεται με την ανάθεση ανεξάρτητων μονάδων υπολογισμού (εργασιών) σε κάθε εκτελεστή. Συγκεκριμένα, η αποστολή κάθε εκτελεστή είναι να δεσμεύσει τους απαραίτητους πόρους, μνήμη και cpu πυρήνες και να εκτελέσει συγκεκριμένες λειτουργίες που καθορίζονται από τον κώδικα της εργασίας σε ένα κομμάτι των δεδομένων που επεξεργάζεται η εργασία. Τα αποτελέσματα της εργασίας τους ποικίλλουν ανάλογα με τον τύπο της εργασίας. Αυτά είτε ανακοινώθηκαν στον οδηγό για να παρακολουθήσουν τη διαδικασία υλοποίησης, όπως θα δούμε είτε αποθηκεύονται στην προσωρινή μνήμη είτε στο δίσκο ως ενδιάμεσα αποτελέσματα που θα καταναλώνονται από μια άλλη λειτουργία σε μια μελλοντική εργασία. Αυτή η αποθήκευση πραγματοποιείται από μια υπηρεσία που ονομάζεται BlockManager και συνεργάζεται με τον εκτελεστή και τον οδηγό. Τέλος, είναι σημαντικό να τονίσουμε την ανοχή σφάλματος στην Αρχιτεκτονική του Spark. Κάθε εργασία

είναι μια ανεξάρτητη υπολογιστική μονάδα, πράγμα που σημαίνει ότι εάν για οποιονδήποτε λόγο μια εργασία αποτύχει να ολοκληρωθεί, αυτό δεν επηρεάζει τη συνεχιζόμενη εκτέλεση της εργασίας μόνο και μόνο επειδή μια άλλη όμοια εργασία θα αρχίσει εκ νέου.

## 2. Driver

Το πρόγραμμα οδήγησης(driver) είναι η διαδικασία που ελέγχει τη μέθοδο `main ()` της εφαρμογής-πελάτη. Σε αυτό υπάρχει το αντικείμενο `SparkContext` που αντιπροσωπεύει και ενσωματώνει όλες τις επιλεγμένες ρυθμίσεις και παραμέτρους για μια συγκεκριμένη εργασία. Οι δύο βασικές υποχρεώσεις του οδηγού μπορούν να συνοψιστούν ως εξής:

Η διαδικασία του οδηγού είναι υπεύθυνη για το σπάσιμο της εφαρμογής σε μικρότερα τεμάχια. Όταν αρχίσει να τρέχει η εφαρμογή, ο οδηγός έχει την ευθύνη να μετατρέψει τον κώδικα σε εργασίες και να του μεταβιβάσει στους εκτελεστές που είναι διαθέσιμοι στους εργαζόμενους του συμπλέγματος.

Στην αρχή, ο οδηγός σχεδιάζει ένα λογικό σχέδιο εργασίας, το οποίο έχει τη μορφή κατευθυνόμενου ακυκλικού γραφήματος (Directed Acyclic Graph - DAG) των διαδικασιών που πρέπει να εκτελεστούν. Μετά από αυτό, το λογικό σχέδιο εκτέλεσης πρέπει να μεταφραστεί σε φυσικό σχέδιο εκτέλεσης. Στο στάδιο αυτό πραγματοποιήθηκαν διάφορες βελτιστοποιήσεις στον τρόπο εκτέλεσης που μπορεί να γίνει από τον οδηγό. Τέλος, μεταφράζει το λογικό σχέδιο υλοποίησης στο φυσικό σχέδιο εκτέλεσης που αποτελείται από πολλαπλά καθήκοντα. Το δεύτερο βήμα είναι να προγραμματίσετε τις εργασίες στους εκτελεστές. Η δρομολόγηση των εργασιών στους εκτελεστές πραγματοποιείται αμέσως μετά την έξοδο από τον οδηγό του σχεδίου φυσικής εκτέλεσης για την εργασία.

Σε αυτή τη φάση, κάθε εκτελεστής έχει κάνει γνωστή την παρουσία του στον οδηγό και αναμένει ότι θα του ανατεθεί μια εργασία για να ξεκινήσει την επεξεργασία. Ο οδηγός κάνει έξυπνες επιλογές που αναθέτουν όσο το δυνατόν περισσότερους κόμβους εργασίας που αποθηκεύονται τοπικά και τα δεδομένα θα χρειαστεί να επεξεργαστούν αυτήν την εργασία. Αυτά μπορεί να αφορούν είτε τα δεδομένα που είναι αποθηκευμένα στον τοπικό δίσκο κόμβων και έχουν πρόσβαση για πρώτη φορά είτε ακόμη και τα δεδομένα που είναι αποθηκευμένα στον κόμβο της προσωρινής μνήμης από προηγούμενη εργασία που εκτελείται εκεί. Σε οποιαδήποτε από τις δύο παραπάνω περιπτώσεις, ο οδηγός προτιμά πάντα τον συγκεκριμένο κόμβο που ικανοποιεί αυτά τα κριτήρια. Αυτό φυσικά δεν είναι πάντα εφικτό, διότι πρέπει μόνο να σκεφτεί την περίπτωση όπου δεν υπάρχουν τέτοιοι κόμβοι στο σύμπλεγμα μας, αλλά αυτή τη στιγμή είναι απασχολημένος να εκτελεί άλλο έργο. Ωστόσο, είναι προτιμότερο να επιλέγεται ένας τέτοιος κόμβος σε σχέση με οποιοδήποτε άλλο, ο σπινθήρας και να παρέχεται σχετική ρύθμιση που καθορίζει τον αριθμό των δευτερολέπτων που μπορούν να αναμένουν ότι ο οδηγός θα απελευθερωθεί ως κόμβος τελικά για να μεταβιβάσει την εργασία.

### 2.3.1.3 Resilient Distributed Datasets (RDD)

Τα ελαστικά καταναμημένα σύνολα δεδομένων (RDD) αποτελούν μια βασική δομή δεδομένων του Spark. Πρόκειται για μια αμετάβλητη καταναμημένη συλλογή αντικειμένων.



Κάθε σύνολο δεδομένων στο RDD χωρίζεται σε λογικά διαμερίσματα, τα οποία μπορούν να υπολογιστούν σε διαφορετικούς κόμβους του συμπλέγματος. Τα RDD μπορούν να περιέχουν οποιοδήποτε τύπο αντικειμένων Python, Java ή Scala, συμπεριλαμβανομένων των κλάσεων που ορίζονται από το χρήστη.

Επισημώς, ένα RDD είναι μια συλλογή αρχείων κατανεμημένων μόνο για ανάγνωση. Τα RDDs μπορούν να δημιουργηθούν μέσω προκαθορισμένων λειτουργιών είτε σε δεδομένα με σταθερή αποθήκευση είτε σε άλλα RDD. Το RDD είναι μια συλλογή στοιχείων ανθεκτικών σε σφάλματα, τα οποία μπορούν να λειτουργούν παράλληλα.

Υπάρχουν δύο τρόποι δημιουργίας RDD - παραλληλισμός μιας υπάρχουσας συλλογής στο πρόγραμμα οδήγησης ή αναφοράς σε ένα σύνολο δεδομένων σε ένα εξωτερικό σύστημα αποθήκευσης, όπως ένα κοινό σύστημα αρχείων, το HDFS, το HBase ή οποιαδήποτε άλλη πηγή δεδομένων που προσφέρει μια μορφή εισαγωγής Hadoop.

Ο Spark κάνει χρήση της ιδέας του RDD για την επίτευξη ταχύτερων και αποδοτικότερων λειτουργιών MapReduce. Ας συζητήσουμε πρώτα πώς πραγματοποιούνται οι διαδικασίες MapReduce και γιατί δεν είναι τόσο αποτελεσματικές.

Η κοινή χρήση δεδομένων είναι αργή στο MapReduce λόγω της αναπαραγωγής, της σειριοποίησης και του δίσκου I/O. Οι περισσότερες από τις εφαρμογές του Hadoop, ξοδεύουν περισσότερο από το 90% του χρόνου, κάνοντας λειτουργίες ανάγνωσης και εγγραφής HDFS.

Αναγνωρίζοντας αυτό το πρόβλημα, οι ερευνητές ανέπτυξαν ένα εξειδικευμένο πλαίσιο που ονομάζεται Apache Spark. Η βασική ιδέα της σπίθας είναι τα ελαστικά κατανεμημένα σύνολα δεδομένων (RDD). Υποστηρίζει τον υπολογισμό επεξεργασίας εντός μνήμης. Αυτό σημαίνει ότι αποθηκεύει την κατάσταση της μνήμης ως αντικείμενο στις εργασίες και το αντικείμενο μοιράζεται μεταξύ αυτών των εργασιών. Η κοινή χρήση δεδομένων στη μνήμη είναι 10 έως 100 φορές ταχύτερη από το δίκτυο και το δίσκο.

### 2.3.1.4 Caching

Όπως έχουμε δει μέχρι στιγμής στο Apache, ο Spark εισάγει στον τομέα των κατανεμημένων υπολογιστικών μηχανημάτων μερικά πολύ ενδιαφέροντα γεγονότα, όπως το RDD που αναλύσαμε στην προηγούμενη ενότητα. Αυτό που γενικά το κάνω πολύ ελκυστικό για εφαρμογές ανάλυσης δεδομένων (εργασίες ανάλυσης δεδομένων) είναι το χαρακτηριστικό που επιτρέπει την αποθήκευση κατά την εκτέλεση των ενδιάμεσων αποτελεσμάτων εργασίας (τα οποία είναι τα ίδια με τη σειρά RDD) στην κύρια μνήμη των κόμβων που ονομάζεται σύμπλεγμα Caching.

Παραδοσιακά μοντέλα υπολογιστικής όπως το MapReduce, τα οποία ακολουθούν μια παρόμοια φιλοσοφία εκτέλεσης, μπορούν να διατηρήσουν μόνο δεδομένα στην κύρια μνήμη μέσα στα οποία επεξεργάζονται από μια συγκεκριμένη εργασία σε δεδομένη χρονική στιγμή. Εάν απαιτούνται τα ίδια δεδομένα από μια επόμενη εργασία που θα εκτελεστεί στον ίδιο κόμβο, τότε αυτά παρέχονται από το δίσκο. Προφανώς δημιουργεί ένα αδύναμο σημείο (συμφόρηση) κατά την εκτέλεση της εργασίας καθώς η ταχύτητα πρόσβασης δεδομένων της κύριας μνήμης σε σύγκριση με το δίσκο είναι υπερπολλαπλασιαστική. Πολύ σημαντικά οφέλη από την προσωρινή αποθήκευση αποκομίζουν τις εργασίες που χρησιμοποιούν επαναληπτικούς αλγόριθμους (επαναληπτικοί αλγόριθμοι), επεξεργάζοντας σε κάθε

επανάληψη το σύνολο δεδομένων που τροφοδοτείται. Τα k-μέσα που ανήκουν σε αυτή την κατηγορία αλγορίθμων και είναι κατάλληλα σε αυτό το τμήμα για να δούμε λίγο πιο λεπτομερώς την έννοια της κρυφής μνήμης και πώς εφαρμόζεται στο Spark.

Όπως έχουμε δει ήδη, οι εκτελεστές είναι εκείνες οι διαδικασίες java που εκτελούνται σε κάθε κόμβο του συμπλέγματος και αποτελούν τον πυρήνα των υπολογισμών κάθε εργασίας. Εφόσον πρόκειται για διεργασίες java που εκτελούνται σε μια εικονική μηχανή Java (JVM) που παρέχει το κομμάτι μνήμης γνωστό ως σωρό, το οποίο είναι το διαθέσιμο περιβάλλον διεργασίας (χώρος εργασίας). Ο σωρός χωρίζεται σε τμήματα μερικής μνήμης που χρησιμεύουν για την αποθήκευση διαφόρων ειδών δεδομένων που σχετίζονται με τη διαδικασία.

Το Spark έχει προεπιλεγμένο μέγεθος 512 MB για κάθε executor, ο οποίος μπορεί να ρυθμιστεί χρησιμοποιώντας τη διαμόρφωση `spark.executor.memory`, με βάση τις ανάγκες μας. Το Spark εκφράζει το μέγεθος μνήμης του βάσει του συνολικού χώρου εργασίας κάθε εκτελεστή JVM. Έχουμε τις εξής περιοχές:

#### **Ασφαλής περιοχή:**

Αυτή η περιοχή αντιπροσωπεύει το 90% του συνολικού σωρού και το λεγόμενο ακριβώς επειδή θέτει ένα ανώτατο όριο το οποίο μπορεί να εμπλέξει τον σωρό για να αποφύγει τις εξαιρέσεις χαμηλής μνήμης. (Σφάλμα εκτός λειτουργίας μνήμης - OOM)

#### **Περιοχή ανακατεύθυνσης:**

Καταλαμβάνει το 20% του ασφαλούς, όπως φαίνεται στο σχήμα και έχει σχεδιαστεί για να αποθηκεύει τα δεδομένα που θα ανακατέψουν. Η διαδικασία πραγματοποιείται όταν τα δεδομένα τυχαίας σειράς παράγουν μια εργασία που πρέπει να καταναλωθεί από έναν άλλο κόμβο που εκτελεί μια άλλη εργασία στο σύμπλεγμα. Συνήθως αυτά τα δεδομένα πρέπει να ταξινομηθούν και έτσι το τμήμα αυτό πληροί τις απαιτήσεις μνήμης για αυτή την ταξινόμηση.

#### **Περιοχή αποθήκευσης:**

Οι λογαριασμοί για το 60% της ασφαλούς περιοχής είναι αυτό που εξυπηρετεί τα καταλύματα αποθήκευσης και το RDD σε έναν κόμβο. Από εδώ μπορούν να έχουν πρόσβαση στα δεδομένα πολύ πιο γρήγορα από οποιαδήποτε εργασία.

#### **Unroll Region:**

Το Spark παρέχει επίσης τη δυνατότητα αποθήκευσης δεδομένων σε σειριακή μορφή στη μνήμη ή στο δίσκο. Αυτή η μορφή δεδομένων απαιτείται από καταναλωμένα συστήματα, όπως το Spark, προκειμένου να μεταφέρονται δεδομένα μέσω του δικτύου. Σε σειριοποιημένα δεδομένα φόρμας δεν μπορούν να χρησιμοποιηθούν αμέσως, αλλά πρώτα πρέπει να γίνει μια διαδικασία αποεπικαλύψεως. Αυτές οι λειτουργίες γεμίζουν την περιοχή ξετυλίγματος που είναι το 20% της περιοχής αποθήκευσης.

## 2.3.2 Radical Cybertools

### 2.3.2.1 Εισαγωγή

Το RADICAL Cybertools [4] είναι μια σουίτα σαφώς καθορισμένων δυνατοτήτων που βασίζονται σε αφαίρεση, οι οποίες είναι σχεδιασμένες για κλιμακούμενες, διαλειτουργικές και βιώσιμες προσεγγίσεις για την υποστήριξη της επιστήμης σε μια σειρά συστημάτων υψηλής απόδοσης και κατανεμημένων υπολογιστών.

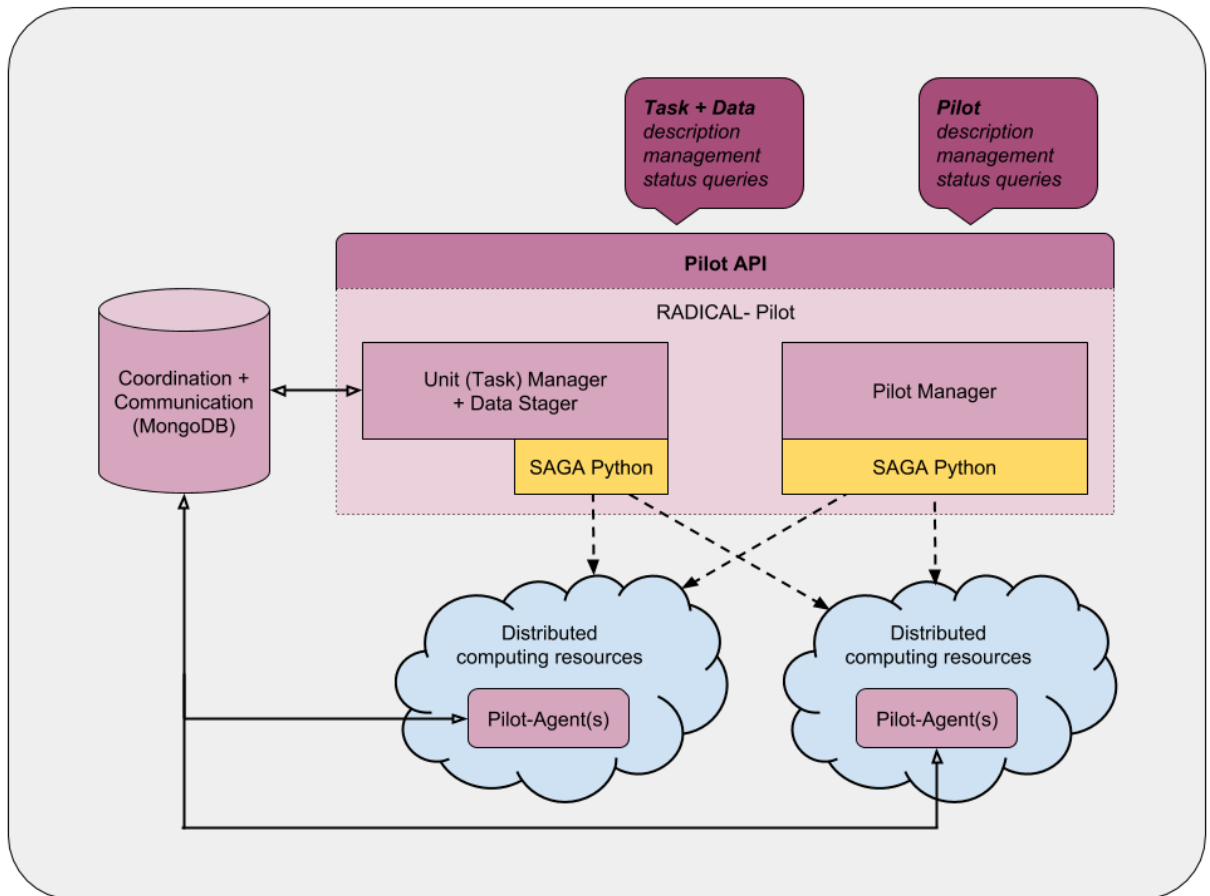
Το RADICAL-SAGA αποτελείται από δύο στοιχεία: RADICAL-Pilot: ένα κλιμακωτό και ευέλικτο σύστημα Pilot-Job που παρέχει ευέλικτες δυνατότητες διαχείρισης πόρων σε επίπεδο εφαρμογών και RADICAL-SAGA: μια ελαφριά διεπαφή που παρέχει μια διαλειτουργικότητα βασισμένη σε πρότυπα σε μια σειρά υπολογιστικών συστημάτων .

Το RADICAL Cybertools βασίζεται σε σημαντικές θεωρητικές εξελίξεις, σε βέλτιστες πρακτικές ανάπτυξης λογισμικού παραγωγής και σε καλά καθορισμένα μοντέλα χρήσης και προγραμματισμού. Το RADICAL Cybertools χρησιμοποιείται σαν μέσο για την έρευνα την ενεργή δομή, που επιτρέπει στους χρήστες της να προσεγγίσουν άψογα το επόμενο επίπεδο κλίμακας και λειτουργικότητας.

Στο έργο αυτό θα ασχοληθούμε κυρίως με το Radical Pilot και το Radical Pilot-Spark.

### 2.3.2.2 Radical Pilot

Το RADICAL-Pilot (παλαιότερα γνωστό ως BigJob) είναι ένα ευέλικτο pilot σύστημα που σε σου λύνει τη διαχείριση εργασίας και δεδομένων για clusters, grids και clouds. Το RADICAL-Pilot γράφεται στη γλώσσα προγραμματισμού της Python και ως εκ τούτου μπορεί εύκολα να αναπτυχθεί στον χώρο των χρηστών. Επιτρέπει τον έλεγχο σε επίπεδο χρήστη των Pilots και υποστηρίζει ένα ευρύ φάσμα τύπων εφαρμογών. Είναι χτισμένο πάνω από το απλό API για εφαρμογές πλέγματος (SAGA), ένα API υψηλού επιπέδου, εύχρηστο για την πρόσβαση σε κατανεμημένους πόρους, που σημαίνει ότι το RADICAL-Pilot μπορεί να δουλέψει σε διάφορα backends όπως PBS, SGE, Amazon EC2, κτλ



Σχήμα 3: Radical-Pilot

- Τι είναι τα Pilot-Jobs?

Τα Pilot-Jobs υποστηρίζουν την αποσύνδεση της υποβολής φόρτου εργασίας από την ανάθεση πόρων. Αυτό έχει ως αποτέλεσμα ένα ευέλικτο μοντέλο εκτέλεσης, το οποίο με τη σειρά του επιτρέπει την κατανεμημένη κλίμακα των εφαρμογών σε πολλαπλούς και ενδεχομένως ετερογενείς πόρους. Τα Pilot-Jobs υποστηρίζουν την χρήση εργασιών με κοντέινερ με εξελιγμένη διαχείριση ροής εργασιών για τον συντονισμό της εκτόξευσης και της αλληλεπίδρασης των πραγματικών υπολογιστικών εργασιών μέσα στο δοχείο. Επιτρέπουν την εκτέλεση εργασιών χωρίς την ανάγκη να περιμένουν κάθε εργασία.

Το κύριο μέλημα του RADICAL-Pilot είναι να είναι ένα ευέλικτο και επεκτάσιμο σύστημα βασισμένο σε pilot systems για την υποβολή εργασιών και τη διαχείριση δεδομένων. Σε αντίθεση με πολλά άλλα συστήματα Pilot-Job, το RADICAL-Pilot υποστηρίζει εγγενώς εργασίες MPI και, λόγω της ενσωμάτωσής του με το saga-python, λειτουργεί σε διάφορα συστήματα backend.

- Πού χρησιμοποιείται το Radical-Pilot?

Το RADICAL-Pilot χρησιμοποιείται από επιστήμονες και μηχανικούς για την επίλυση πραγματικών προβλημάτων, όπως η αναδίπλωση πρωτεϊνών, η παραδοσιακή

δειγματοληψία ομπρέλας μοριακής δυναμικής, η ανταλλαγή αντιγράφων και άλλα προγράμματα που έχουν μεγάλο όγκο δεδομένων / πράξεων.

- Παράμετρος Sweeps
  - Πολλές περιπτώσεις του ίδιου έργου (εφαρμογές σε σύνολο)
  - Ενοποιημένες εργασίες
  - Χαλαρά συζευγμένα, ξεχωριστά καθήκοντα
  - Εργασίες με δεδομένα και / ή εξαρτήσεις υπολογισμού
- Γιατί βοηθάνε τα Pilot-Jobs?

### **Απλή εγκατάσταση, Χιλιάδες εργασίες, λιγότερη ανησυχία**

Η κατανεμημένη κυβερνητική διάρθρωση κατανεμημένη σε επίπεδο παραγωγής έχει σχεδόν πάντα εγκαταστήσει έναν τοπικό διαχειριστή πόρων, όπως ένα σύστημα παρτίδας ουράς. Μια κατανεμημένη εφαρμογή συχνά απαιτεί πολλές εργασίες να παράγουν χρήσιμα δεδομένα εξόδου. Αυτές οι εργασίες έχουν συχνά το ίδιο εκτελέσιμο αρχείο. Ένας παραδοσιακός τρόπος υποβολής αυτών των εργασιών θα ήταν να υποβάλετε μια μεμονωμένη εργασία για κάθε εκτελεστή. Αυτές οι εργασίες (συχνά εκατοντάδες) κάθονται στο σύστημα παρτίδας αναμονής και ενδέχεται να μην ενεργοποιηθούν ταυτόχρονα. Οι παραλλαγές φόρτωσης και προγραμματισμού ενδέχεται να προσθέσουν περιττές ώρες στο συνολικό χρόνο της εργασίας μέχρι την ολοκλήρωσή της λόγω ανεπάρκειας στον προγραμματισμό πολλών μεμονωμένων εργασιών.

Ένα Pilot-Job παρέχει μια εναλλακτική προσέγγιση. Μπορεί να θεωρηθεί ως εργασία εμπορευματοκιβωτίων για πολλές υπο-θέσεις εργασίας. Ένα έργο Pilot-Job αποκτά τους απαραίτητους πόρους για την εκτέλεση των υποδιαιρέσεων (επομένως, ζητά όλους τους πόρους που απαιτούνται για την εκτέλεση των υποδιαιρέσεων, και όχι μόνο μία υποδιόραση). Εάν ένα σύστημα έχει μια παρτίδα παρτίδας, η Pilot-Job υποβάλλεται σε αυτή την ουρά. Μόλις ενεργοποιηθεί, μπορεί να εκτελέσει απευθείας τις υποεργασίες, αντί να περιμένει να θέσει σε ουρά κάθε υποεργασία. Αυτό εξαλείφει την ανάγκη υποβολής διαφορετικής εργασίας για κάθε εκτελέσιμο και μειώνει σημαντικά το χρόνο ολοκλήρωσης.

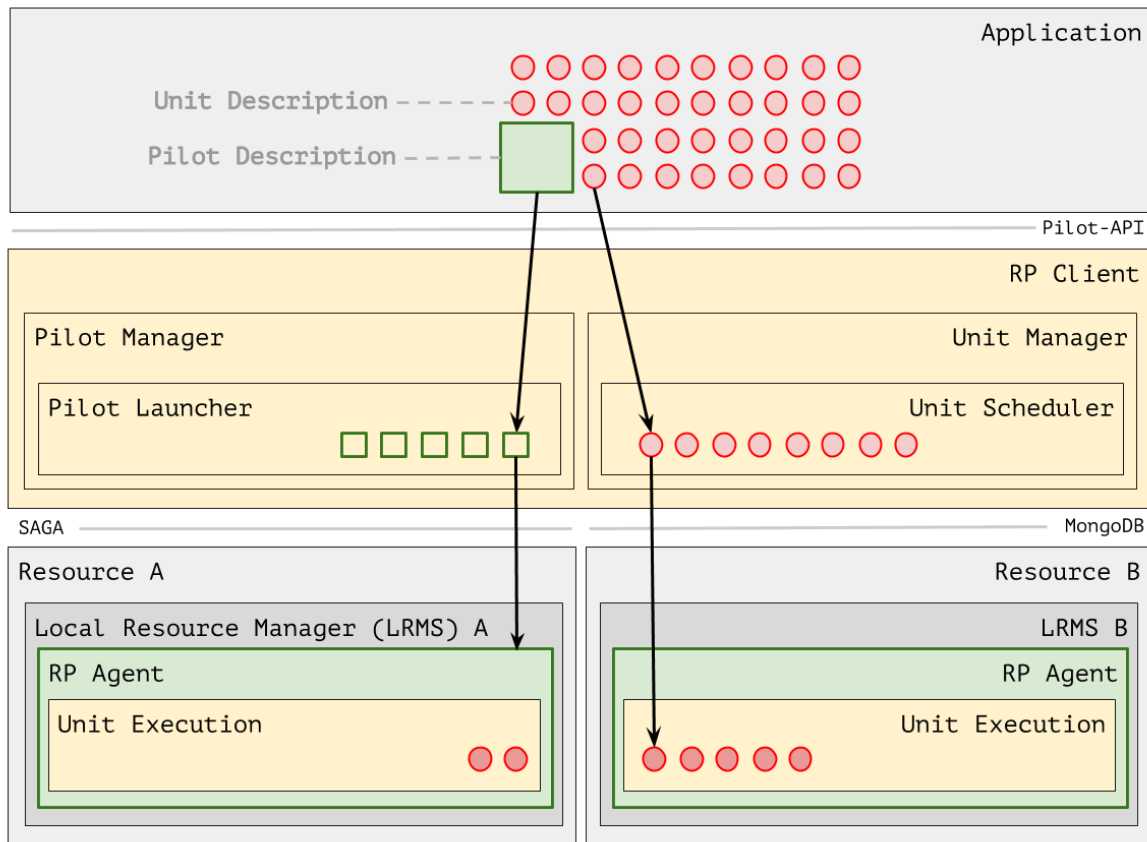
- Αρχιτεκτονική Radical Pilot

Το RADICAL-Pilot (RP) είναι ένα σύστημα Pilot Job που γράφεται στην Python. Επιτρέπει σε έναν χρήστη να εκτελεί μεγάλους αριθμούς υπολογιστικών εργασιών (αποκαλούμενων Compute-Units) ταυτόχρονα σε ένα ή περισσότερα απομακρυσμένα Compute-Pilots που το RADICAL-Pilot μπορεί να ξεκινήσει με διαφάνεια σε πλήθος διαφορετικών κατανεμημένων πόρων, όπως clusters HPC και Clouds.

Σε αυτό το μοντέλο, ένα τμήμα (τεμάχιο) ενός πόρου αποκτάται από την εφαρμογή ενός χρήστη, έτσι ώστε η εφαρμογή να μπορεί να προγραμματίζει απευθείας Υπολογιστικές Μονάδες σε αυτή τη φέτα πόρων, αντί να περάσει από τον προγραμματιστή εργασιών του συστήματος. Σε πολλές περιπτώσεις, αυτό μπορεί να μειώσει δραστικά τον συνολικό χρόνο εκτέλεσης, καθώς οι μεμονωμένες Μονάδες Υπολογιστών δεν χρειάζεται να περιμένουν στην ουρά προγραμματισμού του συστήματος, αλλά μπορούν να εκτελεστούν απευθείας στους υπολογιστές-πιλότους.

Οι μονάδες Compute-Units είναι συχνά εκτελέσιμα με μονόπλευρα / πολλαπλά σπειρώματα, αλλά το RADICAL-Pilot υποστηρίζει επίσης την εκτέλεση παράλληλων εκτελέσιμων αρχείων, για παράδειγμα με βάση το MPI ή το OpenMP.

Το RADICAL-Pilot δεν είναι ένα στατικό σύστημα, αλλά παρέχει στον χρήστη μια βιβλιοθήκη προγραμματισμού ("Pilot-API") που παρέχει αφαιρέσεις για πρόσβαση σε πόρους και διαχείριση εργασιών. Με τη βιβλιοθήκη αυτή, ο χρήστης μπορεί να αναπτύξει τα πάντα από απλά "scripts υποβολής" σε αυθαίρετα περίπλοκες εφαρμογές, υπηρεσίες και εργαλεία υψηλότερου επιπέδου.



Σχήμα 4: Αρχιτεκτονική του RP

Η εικόνα επισκόπησης της αρχιτεκτονικής του RP (Σχ. 4) δείχνει τα κύρια στοιχεία ή τα RP και τις λειτουργικές τους σχέσεις. Το σύστημα RP θα ερμηνεύσει τις πιλοτικές περιγραφές και θα υποβάλει τις αντίστοιχες πιλοτικές παρουσίες στους πόρους-στόχους. Στην συνέχεια θα δεχτεί τις περιγραφές μονάδων και θα υποβάλει αυτούς για εκτέλεση στους πιλότους που δημιουργήθηκαν νωρίτερα.

### 2.3.2.3 Radical Pilot-Spark

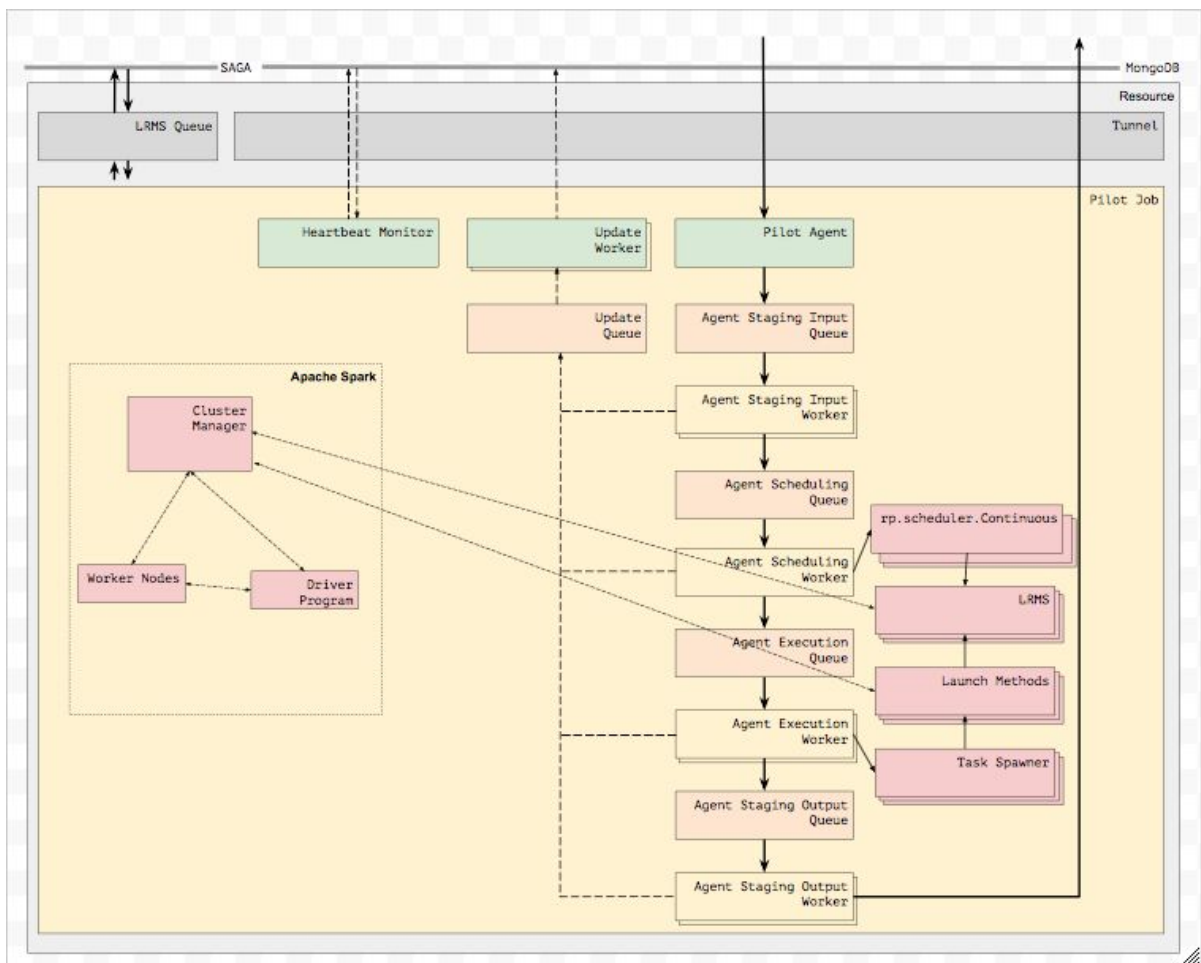
Το Radical-Pilot Spark είναι ένα χαρακτηριστικό που προστέθηκε στο Radical-Pilot. Το Spark έχει ενσωματωθεί στο Radical Pilot για να υποστηρίξει και να εκτελέσει την εργασία

Apache Spark στο διανεμημένο περιβάλλον του κυβερνοεπιχειρησιακού δικτύου που υποστηρίζεται από το Radical-Pilot.

Το Radical Pilot-Spark είναι υπεύθυνο για τη ρύθμιση του περιβάλλοντος στην απαιτούμενη υποδομή ώστε ο χρήστης να μπορεί να εκτελεί εφαρμογές Apache Spark. Μια εργασία Pilot-Spark μπορεί να χωριστεί σε τρεις διαφορετικές κατηγορίες.

Στην αρχή το πιλοτικό σύστημα ελέγχει αν οι απαιτούμενοι πόροι έχουν ήδη εγκαταστήσει το Spark και τις εξαρτήσεις του. Εάν όχι, το πιλοτικό σύστημα εξετάζει αν οι μονάδες μπορούν να φορτωθούν στο μηχάνημα, διαφορετικά θα μεταφορτώσει τα κομμάτια που λείπουν από το Internet. Αυτές είναι η πιο πρόσφατη έκδοση Java, γλώσσα scala και apache spark.

Το επόμενο βήμα είναι η διαμόρφωση του Spark. Αυτό σημαίνει ότι το σύστημα θα ορίσει τις κύριες μεταβλητές που απαιτούνται από το σπινθήρισμα για να αρχίσει να τρέχει. Αυτό θα είναι: JAVA\_HOME, SCALA\_HOME, MASTER\_IP και να προσθέσετε τους κόμβους σκλάβου στο αρχείο διαμόρφωσης σκλάβων. Ο αριθμός των εξαρτημένων κόμβων εξαρτάται από τον απαιτούμενο αριθμό πυρήνων από τη διεπαφή του radical-pilot. Το σύμπλεγμα Spark είναι τώρα έτοιμο να ξεκινήσει να εκτελεί εφαρμογές.



## 2.4 Εργαλεία

## 2.4.1 Map-Reduce

Το map-reduce [12] παρουσιάστηκε από την Google το 2004. Πρόκειται για ένα μοντέλο προγραμματισμού και μια συναφή υλοποίηση για την επεξεργασία και τη δημιουργία μεγάλων συνόλων δεδομένων. Οι χρήστες καθορίζουν μια συνάρτηση mapper που επεξεργάζεται ζεύγος κλειδιών / τιμών για να δημιουργήσει ένα σύνολο ζευγών ενδιάμεσου κλειδιού / τιμής και μια συνάρτηση μείωσης που συγχωνεύει όλες τις ενδιάμεσες τιμές που σχετίζονται με το ίδιο ενδιάμεσο κλειδί. Πολλά καθήκοντα του πραγματικού κόσμου εκφράζονται σε αυτό το μοντέλο.

Τα προγράμματα γραμμένα σε αυτό το λειτουργικό στυλ παραλληλίζονται αυτόματα και εκτελούνται σε ένα μεγάλο σύνολο μηχανών βασικών προϊόντων. Το σύστημα χρόνου εκτέλεσης φροντίζει για τις λεπτομέρειες του διαχωρισμού των δεδομένων εισόδου, του προγραμματισμού της εκτέλεσης του προγράμματος σε ένα σύνολο μηχανών, του χειρισμού των βλαβών του μηχανήματος και της διαχείρισης της απαιτούμενης επικοινωνίας μεταξύ μηχανών. Αυτό επιτρέπει στους προγραμματιστές χωρίς εμπειρία σε παράλληλα και καταναμημένα συστήματα να χρησιμοποιούν εύκολα τους πόρους ενός μεγάλου καταναμημένου συστήματος.

Ο υπολογισμός λαμβάνει ένα σύνολο ζευγών κλειδιού / τιμής εισόδου και παράγει ένα σύνολο ζευγών κλειδιού / τιμής εξόδου. Ο χρήστης της βιβλιοθήκης Map-Reduce εκφράζει τον υπολογισμό ως δύο λειτουργίες: Map and Reduce.

Ο mapper, γραμμένος από το χρήστη, παίρνει ένα ζεύγος εισόδου και παράγει ένα σύνολο από ενδιάμεσα ζεύγη κλειδιού / τιμής. Η βιβλιοθήκη Map-Reduce συγκεντρώνει όλες τις ενδιάμεσες τιμές που σχετίζονται με το ίδιο ενδιάμεσο πλήκτρο  $k$  και τις μεταφέρει στη λειτουργία reduce.

$\text{map}(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$

Η συνάρτηση Reduce, επίσης γραμμένη από το χρήστη, δέχεται ένα ενδιάμεσο πλήκτρο  $k$  και ένα σύνολο τιμών για αυτό το κλειδί. Συνενώνει αυτές τις τιμές για να σχηματίσει ένα πιθανότερο μικρότερο σύνολο τιμών. Συνήθως παράγεται μόνο τιμή μηδέν ή μία τιμή εξόδου ανά μείωση της επίκλησης. Οι ενδιάμεσες τιμές παρέχονται στη λειτουργία μείωσης του χρήστη μέσω ενός iterator. Αυτό μας επιτρέπει να χειριζόμαστε λίστες τιμών που είναι πολύ μεγάλες για να χωρέσουν στη μνήμη.

$\text{reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_2)$



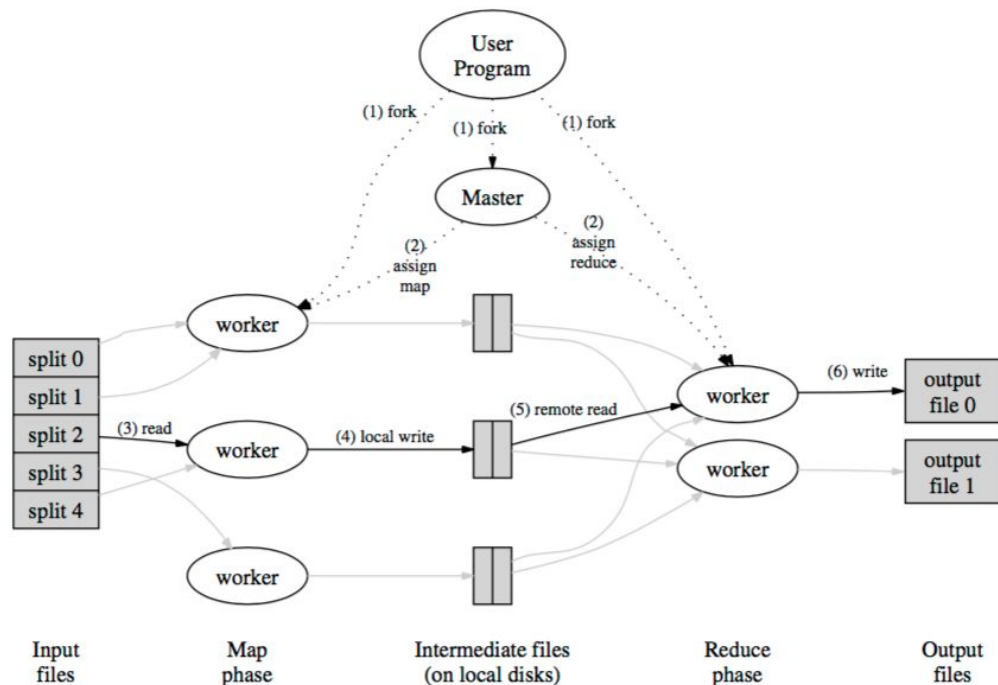


Figure 1: Execution overview

## Fault-Tolerance

Δεδομένου ότι η βιβλιοθήκη MapReduce έχει σχεδιαστεί για να βοηθήσει στην επεξεργασία πολύ μεγάλων ποσοτήτων δεδομένων χρησιμοποιώντας εκατοντάδες ή χιλιάδες μηχανές, η βιβλιοθήκη πρέπει να μπορεί να ανεχθεί αποτυχίες του μηχανήματος.

Ο master δέχεται περιοδικά μηνύματα από κάθε εργαζόμενο. Αν δε ληφθεί απάντηση από έναν εργαζόμενο σε ένα συγκεκριμένο χρονικό διάστημα, ο πλοίαρχος σηματοδοτεί τον εργαζόμενο ως αποτυχημένο. Οποιοσδήποτε εργασίες map που ολοκληρώθηκαν από τον εργαζόμενο επαναφέρονται στην αρχική κατάσταση αδράνειας και επομένως γίνονται αποδεκτές για προγραμματισμό σε άλλους εργαζόμενους. Ομοίως, οποιαδήποτε εργασία map ή μείωση της τρέχουσας εργασίας σε έναν αποτυχημένο εργαζόμενο επαναφέρεται επίσης σε αδράνεια και γίνεται επιλέξιμος για αναδιάταξη.

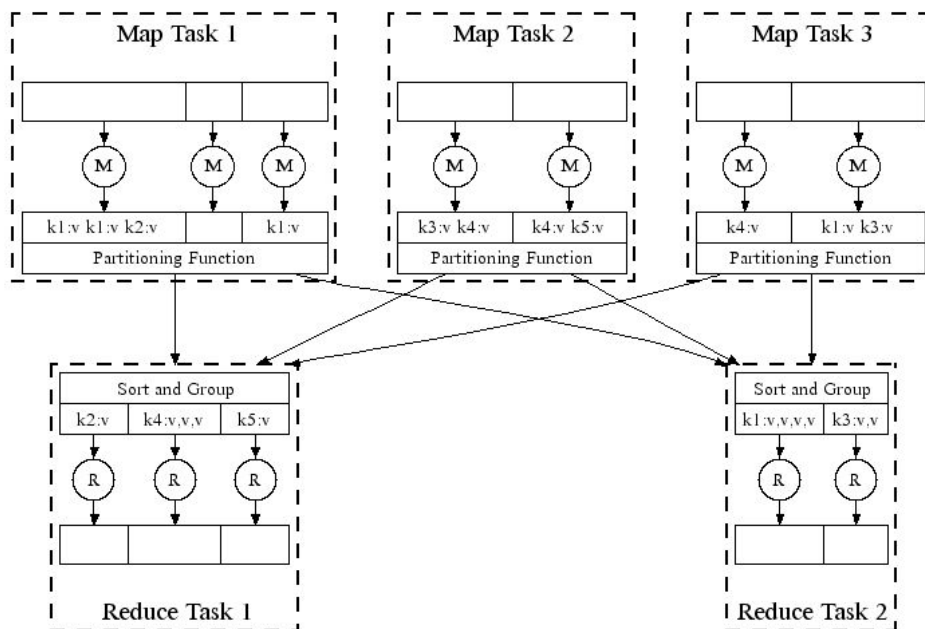
Οι εργασίες map που ολοκληρώθηκαν εκτελούνται εκ νέου, επειδή η έξοδός τους αποθηκεύεται στον τοπικό δίσκο του μη επιτυχημένου μηχανήματος και επομένως είναι απροσπέλαστη. Οι ολοκληρωμένες εργασίες μείωσης δεν χρειάζεται να εκτελούνται εκ νέου, αφού η παραγωγή τους αποθηκεύεται σε ένα παγκόσμιο σύστημα αρχείων.

Όταν μια εργασία map εκτελείται πρώτα από τον εργαζόμενο A και στη συνέχεια εκτελείται αργότερα από τον εργαζόμενο B (επειδή το A απέτυχε), όλοι οι εργαζόμενοι που εκτελούν μειωμένες εργασίες ενημερώνονται για την εκ νέου εκτέλεση. Κάθε εργασία μείωσης που δεν έχει ήδη διαβάσει τα δεδομένα από τον εργαζόμενο A θα διαβάσει τα δεδομένα από τον εργαζόμενο B.

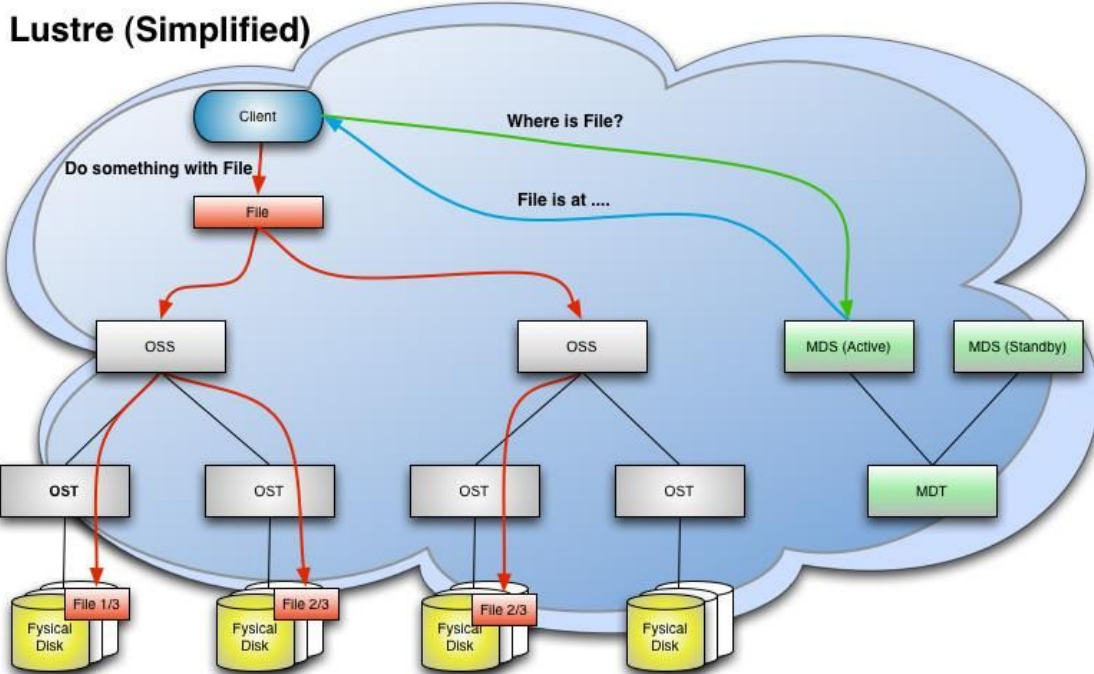
Το MapReduce είναι ανθεκτικό στις μεγάλες αποτυχίες των εργαζομένων. Για παράδειγμα, κατά τη διάρκεια μίας λειτουργίας MapReduce, η συντήρηση δικτύου σε ένα σύμπλεγμα που εκτελούσε προκάλεσε ομάδες 80 μηχανών κάθε φορά για να καταστεί απρόσιτη για αρκετά λεπτά. Ο κύριος MapReduce απλώς επανεξήγησε την εργασία που πραγματοποίησε η μη προσβάσιμη μηχανή εργατών και συνέχισε να κάνει πρόοδο, τελικά ολοκληρώνοντας τη λειτουργία MapReduce.

Είναι εύκολο να κάνετε τα κύρια περιοδικά σημεία εγγραφής των δομών κύριων δεδομένων που περιγράφονται παραπάνω. Εάν πεθάνει η κύρια εργασία, ένα νέο αντίγραφο μπορεί να ξεκινήσει από το τελευταίο checkpointed σημείο. Ωστόσο, δεδομένου ότι υπάρχει μόνο ένας κύριος, η αποτυχία του είναι απίθανη. Σε αυτήν την εφαρμογή, τερματίζεται ο υπολογισμός MapReduce αν αποτύχει ο κύριος. Οι πελάτες μπορούν να ελέγξουν για αυτή την κατάσταση και να ξαναδοκιμάσουν τη λειτουργία MapReduce εάν το επιθυμούν.

Η ιδιαίτερη αξία του MapReduce προκύπτει από το γεγονός ότι προσφέρει ένα απλό αλλά πολύ ισχυρό μοντέλο προγραμματισμού για κατανεμημένες εφαρμογές, αρκεί να εκφραστούν σε φάσεις χαρτών και να μειωθούν. Η συνεισφορά στον τομέα των κατανεμημένων συστημάτων αντικατοπτρίζεται στο γεγονός ότι σχεδόν αμέσως εφάρμοσαν ανταγωνιστική εφαρμογή ανοιχτού κώδικα η οποία περιλαμβάνεται στο οικοσύστημα Hadoop, ενώ το μοντέλο εκτέλεσης παραμένει αμετάβλητο στα νέα συστήματα παράλληλης επεξεργασίας όπως το Apache Spark που ενσωματώσαμε Radical Cybertools.



## 2.4.2 Lustre Filesystem



Σχήμα 5: Lustre Filesystem

### 2.4.2.1 Εισαγωγή

Το Lustre [6] είναι ένα είδος παράλληλου καταμεμημένου συστήματος αρχείων, το οποίο χρησιμοποιείται γενικά για υπολογιστές συμπλεγμάτων μεγάλης κλίμακας. Το όνομα Lustre είναι μια λέξη που προέρχεται από το Linux και το cluster. Το λογισμικό συστήματος αρχείων Lustre είναι διαθέσιμο υπό την Γενική Άδεια Δημόσιας Χρήσης GNU (μόνο έκδοση 2) και παρέχει συστήματα αρχείων υψηλής απόδοσης για συστοιχίες υπολογιστών που κυμαίνονται από μικρά σύνολα ομάδων εργασίας έως μεγάλης κλίμακας συμπλέγματα πολλαπλών τοποθεσιών.

Επειδή τα συστήματα αρχείων Luster διαθέτουν δυνατότητες υψηλής απόδοσης και ανοικτή αδειοδότηση, χρησιμοποιείται συχνά σε supercomputers. Από τον Ιούνιο του 2005, έχει χρησιμοποιηθεί με συνέπεια τουλάχιστον από τα μισά από τα κορυφαία δέκα και πάνω από 60 από τους 100 πρώτους υψηλότερους υπερυπολογιστές παγκοσμίως, συμπεριλαμβανομένων των κορυφαίων supercomputers TOP500 στον κόσμο το 2014, Titan και Sequoia.

Τα συστήματα αρχείων Lustre είναι επεκτάσιμα και μπορούν να είναι μέρος πολλών συμπλεγμάτων υπολογιστών με δεκάδες χιλιάδες κόμβους πελάτη, δεκάδες petabytes (PB) αποθήκευσης σε εκατοντάδες εξυπηρετητές και περισσότερα από ένα terabyte ανά δευτερόλεπτο (TB / s) συνολικού I / O διακίνηση. Αυτό καθιστά τα συστήματα αρχείων Lustre μια δημοφιλή επιλογή για επιχειρήσεις με μεγάλα κέντρα δεδομένων, συμπεριλαμβανομένων εκείνων σε κλάδους όπως η μετεωρολογία, η προσομοίωση, το πετρέλαιο και το φυσικό αέριο, η επιστήμη της ζωής, τα πλούσια μέσα ενημέρωσης και η χρηματοδότηση.

## 2.4.2.2 Lustre Architecture

Ένα σύστημα αρχείων Lustre έχει τρεις κύριες λειτουργικές μονάδες:

Ένας ή περισσότεροι διακομιστές μεταδεδομένων (MDSes) που έχουν έναν ή περισσότερους στόχους μεταδεδομένων (MDT) ανά σύστημα αρχείων Lustre που αποθηκεύει μεταδεδομένα χώρου ονομάτων, όπως ονόματα αρχείων, καταλόγους, δικαιώματα πρόσβασης και διάταξη αρχείων. Τα δεδομένα MDT αποθηκεύονται σε ένα σύστημα αρχείων τοπικού δίσκου. Ωστόσο, σε αντίθεση με τα κατανεμημένα συστήματα αρχείων βασισμένα σε μπλοκ, όπως το GPFS και το PanFS, όπου ο διακομιστής μεταδεδομένων ελέγχει όλη την κατανομή των μπλοκ, ο διακομιστής μεταδεδομένων Lustre εμπλέκεται μόνο σε ελέγχους διαδρομής και αδειών και δεν συμμετέχει σε καμία λειτουργία εισόδου / εξόδου αρχείων, Αποφεύγοντας τα σημεία συμφόρησης κλιμάκωσης I / O στο διακομιστή μεταδεδομένων. Η δυνατότητα πολλαπλών MDT σε ένα σύστημα αρχείων είναι μια νέα δυνατότητα στο Lustre 2.4 και επιτρέπει στις δευτερεύουσες σειρές καταλόγου να βρίσκονται στη δευτερεύουσα MDT, ενώ το 2.7 και αργότερα επιτρέπει στους μεγάλους μεμονωμένους καταλόγους να διανέμονται και σε πολλαπλούς MDTs.

Ένας ή περισσότεροι διακομιστές αποθήκευσης αντικειμένων (OSS) που αποθηκεύουν δεδομένα αρχείων σε έναν ή περισσότερους στόχους αποθήκευσης αντικειμένων (OST). Ανάλογα με το υλικό του διακομιστή, ένα OSS τυπικά εξυπηρετεί μεταξύ δύο και οκτώ OSTs, με κάθε OST να διαχειρίζεται ένα ενιαίο σύστημα αρχείων τοπικών δίσκων. Η χωρητικότητα ενός συστήματος αρχείων Lustre είναι το άθροισμα των δυνατοτήτων που παρέχουν οι OST. Πελάτες που έχουν πρόσβαση και χρησιμοποιούν τα δεδομένα. Το Lustre παρουσιάζει όλους τους πελάτες με έναν ενοποιημένο χώρο ονομάτων για όλα τα αρχεία και τα δεδομένα στο σύστημα αρχείων, χρησιμοποιώντας την τυπική σημασιολογία POSIX και επιτρέπει ταυτόχρονη και συνεπή πρόσβαση ανάγνωσης και εγγραφής στα αρχεία του συστήματος αρχείων.

Το MDT, το OST και ο πελάτης μπορεί να βρίσκονται στον ίδιο κόμβο (συνήθως για σκοπούς δοκιμής), αλλά σε τυπικές εγκαταστάσεις παραγωγής οι λειτουργίες αυτές βρίσκονται σε χωριστούς κόμβους που επικοινωνούν μέσω δικτύου. Το στρώμα Luster Network (LNET) μπορεί να χρησιμοποιήσει διάφορους τύπους διασυνδέσεων δικτύου, συμπεριλαμβανομένων των εγγενών ρήμων InfiniBand, TCP / IP σε Ethernet και άλλων τεχνολογιών ιδιόκτητων δικτύων όπως η διασύνδεση Cray Gemini. Στο Luster 2.3 και νωρίτερα υποστηρίχθηκαν επίσης τα δίκτυα Myrinet, Quadrics, Cray SeaStar και Rapid-Array, αλλά αυτοί οι οδηγοί δικτύου απογοητεύτηκαν όταν αυτά τα δίκτυα δεν ήταν πλέον εμπορικά διαθέσιμα και η υποστήριξη αφαιρέθηκε πλήρως στο Lustre 2.8. Το Lustre θα επωφεληθεί από τις μεταφορές μέσω απευθείας απομακρυσμένης μνήμης (RDMA), όταν είναι διαθέσιμες, για να βελτιώσει τη διακίνηση και να μειώσει τη χρήση της CPU.

Η αποθήκευση που χρησιμοποιείται για τα συστήματα αρχείων υποστήριξης MDT και OST παρέχεται κανονικά από συσκευές RAID υλικού, αν και θα λειτουργήσει με οποιοσδήποτε συσκευές μπλοκ. Από το Lustre 2.4, το MDT και το OST μπορούν επίσης να χρησιμοποιήσουν το ZFS για το σύστημα αρχείων υποστήριξης, επιτρέποντάς τους να χρησιμοποιούν αποτελεσματικά την αποθήκευση JBOD αντί για συσκευές RAID υλικού. Οι διακομιστές Lustre OSS και MDS διαβάζουν, γράφουν και τροποποιούν δεδομένα με τη

μορφή που επιβάλλει το υποστηρικτικό σύστημα αρχείων και επιστρέφουν τα δεδομένα αυτά στους πελάτες. Οι πελάτες δεν έχουν άμεση πρόσβαση στο υποκείμενο αποθηκευτικό χώρο.

Ένα OST είναι ένα αποκλειστικό σύστημα αρχείων που εξάγει μια διεπαφή σε byte εύρους αντικειμένων για λειτουργίες ανάγνωσης / εγγραφής. Ένα MDT είναι ένα αποκλειστικό σύστημα αρχείων που ελέγχει την πρόσβαση στο αρχείο και ενημερώνει τους πελάτες για τη διάταξη των αντικειμένων που συνθέτουν κάθε αρχείο. Τα MDT και τα OST χρησιμοποιούν είτε μια βελτιωμένη έκδοση του ext4 που ονομάζεται `ldiskfs`, είτε το ZFS / DMU για αποθήκευση δεδομένων MDT και OST back-end για την αποθήκευση αρχείων / αντικειμένων χρησιμοποιώντας τη θύρα ZFS-on-Linux ανοιχτής προέλευσης.

Όταν ένας υπολογιστής-πελάτης αποκτά πρόσβαση σε ένα αρχείο, εκτελεί μια αναζήτηση αρχείου στο MDS. Όταν ολοκληρωθεί η αναζήτηση ονόματος αρχείου MDS, είτε η διάταξη ενός υπάρχοντος αρχείου επιστρέφεται στον πελάτη είτε ένα νέο αρχείο δημιουργείται για λογαριασμό του πελάτη, εφόσον ζητηθεί. Για πράξεις ανάγνωσης ή εγγραφής, ο πελάτης ερμηνεύει τη διάταξη σε επίπεδο στρώματος λογικού αντικειμένου (LOV), το οποίο χαρτογραφεί την μετατόπιση και το μέγεθος σε ένα ή περισσότερα αντικείμενα, καθένα από τα οποία βρίσκεται σε ξεχωριστό OST. Ο πελάτης τότε κλειδώνει το φάσμα αρχείων που λειτουργούν και εκτελεί μία ή περισσότερες παράλληλες λειτουργίες ανάγνωσης ή εγγραφής απευθείας στα OST. Με αυτήν την προσέγγιση, τα σημεία συμφόρησης για τις επικοινωνίες client-to-OST εξαλείφονται, έτσι ώστε το συνολικό εύρος ζώνης που είναι διαθέσιμο για τους πελάτες να διαβάζουν και να γράφουν δεδομένα κλιμακώνεται σχεδόν γραμμικά με τον αριθμό των OST στο σύστημα αρχείων. Μετά την αρχική αναζήτηση της διάταξης αρχείου, το MDS δεν συμμετέχει στο αρχείο IO.

Οι πελάτες δεν τροποποιούν άμεσα τα αντικείμενα στα συστήματα αρχείων OST, αλλά, αντίθετα, αναθέτουν αυτήν την εργασία στα OSSes. Αυτή η προσέγγιση εξασφαλίζει δυνατότητα κλιμάκωσης για μεγάλης κλίμακας συμπλέγματα και υπερυπολογιστές, καθώς και βελτιωμένη ασφάλεια και αξιοπιστία. Αντίθετα, τα συστήματα αρχείων που βασίζονται σε block-based συστήματα, όπως το Global File System και το OCFS, πρέπει να επιτρέπουν άμεση πρόσβαση σε όλες τις υποκείμενες αποθηκευτικές εγκαταστάσεις από όλους τους πελάτες του συστήματος αρχείων, που απαιτούν μεγάλο SAN back-end που συνδέεται με όλους τους πελάτες και Αυξάνει τον κίνδυνο της διαφθοράς αρχείων από εσφαλμένους / ελαττωματικούς πελάτες.

## Κεφάλαιο 3

### 3. Εκτέλεση και μοντελοποίηση του αλγόριθμου k-μέσων χρησιμοποιώντας το Radical-Pilot και Pilot-Spark.

## **Επισκόπηση**

Σε αυτό το κεφάλαιο παρουσιάζουμε τον αλγόριθμο k-means, όπως έγινε με τη χρήση της σουίτας του Radical-Pilot System. Αρχίζουμε το κεφάλαιο που περιγράφει ολόκληρη την πειραματική διαδικασία που ακολουθείται για την εξαγωγή των δεδομένων στα οποία βασιζόμαστε για τα στάδια της ανάλυσης και της μοντελοποίησης. Στη συνέχεια αφιερώστε ένα τμήμα στην περιγραφή των συνόλων δεδομένων που χρησιμοποιούν και της κατασκευής τους. Επίσης αφιερώθηκε ένα τμήμα που περιγράφει τον τρόπο με τον οποίο ο αλγόριθμος εκτελείται σε μηχανές TACC, Stampede και Wrangler, ενώ στις δύο τελευταίες ενότητες αυτού του κεφαλαίου αναλύονται και παρουσιάζονται τα αποτελέσματα των πειραμάτων μας. Τέλος, παρουσιάζονται τα μοντέλα που κατασκευάστηκαν και επαληθεύεται η ακρίβειά τους.

### **3.1 Περιγραφή της Πειραματικής Διαδικασίας**

Ο σκοπός της διαδικασίας δοκιμής είναι η εκτέλεση του αλγορίθμου k-means που ποικίλει σε κάθε παράμετρο εκτέλεσης που σχετίζεται με τη λογική του αλγορίθμου, όπως είναι η επιλογή του k ως ο αριθμός των συστάδων, ο αριθμός επαναλήψεων αλγορίθμου και παραμέτρων που περιγράφουν τη μορφή του συνόλου δεδομένων εισόδου δεδομένου ότι είναι ο αριθμός των φορέων / αντικειμένων (παρουσιών) και το μέγεθος (διαστάσεις) κάθε στιγμιότυπου του συνόλου δεδομένων. Παράλληλα είναι ενδιαφέρον το μεταβαλλόμενο περιβάλλον εκτέλεσης από την άποψη της κύριας μνήμης του μηχανήματος που τρέχει τον αλγόριθμο και του αριθμού των διαθέσιμων πυρήνων (πυρήνων). Για κάθε τέτοια εκτέλεση ενδιαφέρεστε για τη συλλογή μετρήσεων απόδοσης, όπως έχουμε ήδη αναφέρει, που θα μας βοηθήσουν σε αυτό το στάδιο να αναλύσουμε τη συμπεριφορά που παρατηρούμε κατά τη μοντελοποίηση.

Συγκρίνεται ο χρόνος ολοκλήρωσης του αλγορίθμου k-means που εκτελείται σε δύο υποδομές σε HPC. Χρησιμοποιούμε τρία διαφορετικά σενάρια: 10.000 σημεία και 5.000 συμπλέγματα, 100.000 σημεία / 500 ομάδες και 1.000.000 σημεία / 50 συμπλέγματα. Κάθε σημείο ανήκει σε έναν τρισδιάστατο χώρο. Οι απαιτήσεις υπολογισμού εξαρτώνται από το προϊόν του αριθμού των σημείων και του αριθμού των συστάδων, συνεπώς είναι σταθερό και για τα τρία σενάρια. Η επικοινωνία στην φάση της ανακατεύθυνσης, ωστόσο, αυξάνεται με το αριθμός σημείων. Για τους σκοπούς αυτού του σημείου αναφοράς, τρέχουμε 2 επαναλήψεις του k-means. Αυτό θα μας προσφέρει την πιο αντιπροσωπευτική εικόνα της απόδοσης του αλγορίθμου και επίσης θα μας εμποδίσει να κάνουμε πυκνή δειγματοληψία που στερείται χρήσιμων πληροφοριών και μελετώντας το νόημα.

Αρχικά περιορίστηκε να χρησιμοποιηθούν μόνο δύο διαφορετικές εκδόσεις για τον συνολικό αριθμό πυρήνων των μηχανών, επειδή, όπως θα δείξουμε στην επόμενη ενότητα, η απόδοση του αλγορίθμου είναι ανεξάρτητη από τους πυρήνες. Ως αναφορά η επιλογή αυτής της μνήμης ήταν μετά από τα αρχικά μας πειράματα όπου παρατηρήσαμε ότι οι εκτελέσεις αλγορίθμου με περισσότερες από 16 GB μνήμης που ορίσαμε ως μέγιστη δεν ήταν πρακτική

αξία μελέτης που ξεπερνούσε την λογική χρήση της μνήμης μιας κεντρικής υλοποίησης και τα αποδεκτά χρονικά πλαίσια αντικατοπτρίζουν ρεαλιστική εφαρμογή.

Χρησιμοποιούμε έως και 3 κόμβους σε Stampede και Wrangler. Στο Stampede κάθε κόμβος έχει 16 πυρήνες και 32 GB μνήμης. Σε Wrangler 24 πυρήνες και 128 GB μνήμης. Εκτελούμε τα πειράματα με τις ακόλουθες διαμορφώσεις: 8 εργασίες σε 1 κόμβο, 16 εργασίες σε 2 κόμβους και 32 εργασίες σε 3 κόμβους.

## 3.2 Κατασκευή Συνόλου Δεδομένων

Όπως αναφέρθηκε παραπάνω στα δεδομένα εισόδου του αλγορίθμου k-means, σε πραγματικές εφαρμογές είναι αντικείμενα των οποίων τα χαρακτηριστικά εμφανίζονται σε διαφορετικές διαστάσεις κάθε στιγμής και καθώς οι τιμές έχουν αναληφθεί για να ποσοτικοποιήσουν το αντίστοιχο χαρακτηριστικό. Με βάση αυτά και εξαιτίας της απουσίας πραγματικών δεδομένων, μπορούμε να αναπτύξουμε μια python εφαρμογή που δημιουργεί συνθετικά σύνολα δεδομένων με συγκεκριμένο αριθμό παρουσιών και διαστάσεων που καθορίζει ο χρήστης ως παραμέτρους εισόδου στο σενάριο.

Η γεννήτρια του συνόλου δεδομένων δημιουργεί τις συντεταγμένες των παρουσιών χρησιμοποιώντας την ομοιόμορφη συνάρτηση από την τυχαία βιβλιοθήκη του python [15]. Ο χρήστης επιλέγει τον αριθμό των παρουσιών που είναι πρόθυμοι να χρησιμοποιήσει και η γεννήτρια δημιουργεί τα ζητούμενα στοιχεία. Αυτά τα δεδομένα εξάγονται σε ένα αρχείο του οποίου το όνομα είναι dataset.in. Τα στοιχεία που δημιουργούνται στο σενάριο είναι τρισδιάστατα αλλά είναι πολύ εύκολο να αλλάξετε τον αριθμό των διαστάσεων των στοιχείων αλλάζοντας μόνο δύο γραμμές στον κώδικα της εφαρμογής.

Ο αριθμός των διαστάσεων και η κατανομή των στοιχείων θα μπορούσαν να είναι σημαντικοί παράγοντες για την ταχύτητα σύγκλισης του αλγορίθμου και συνεπώς για το συνολικό χρόνο ολοκλήρωσης του αλγορίθμου. Δεδομένου ότι ο σκοπός αυτής της εργασίας δεν είναι η απόδοση της απόδοσης του αλγορίθμου k-means, αποφασίσαμε να επιλέξουμε ένα μέγιστο αριθμό δύο επαναλήψεων προκειμένου να απαλλαγούμε από τον παράγοντα τυχαιότητας της ολοκλήρωσης k-means.

## 3.3 Αποτελέσματα των Πειραμάτων

Σε αυτή την υποενότητα παρουσιάζουμε και αναλύουμε τα αποτελέσματα των εκτελέσεων των k-μέσων για τις διάφορες παραμέτρους που είδαμε στην ενότητα 3.1. Εστιάζουμε στο συνολικό χρόνο ολοκλήρωσης του αλγορίθμου και οι βαθμοί ελευθερίας είναι ο αριθμός των πυρήνων cpu και το μέγεθος των συνόλων δεδομένων.

Πραγματοποιούμε πειράματα με 8, 16 και 32 πυρήνες. Όσον αφορά το σύνολο δεδομένων, χρησιμοποιούμε δέκα χιλιάδες μονάδες με 5 χιλιάδες ομάδες, εκατό χιλιάδες σημεία με 500 ομάδες και ένα εκατομμύριο σημεία με πενήντα ομάδες.

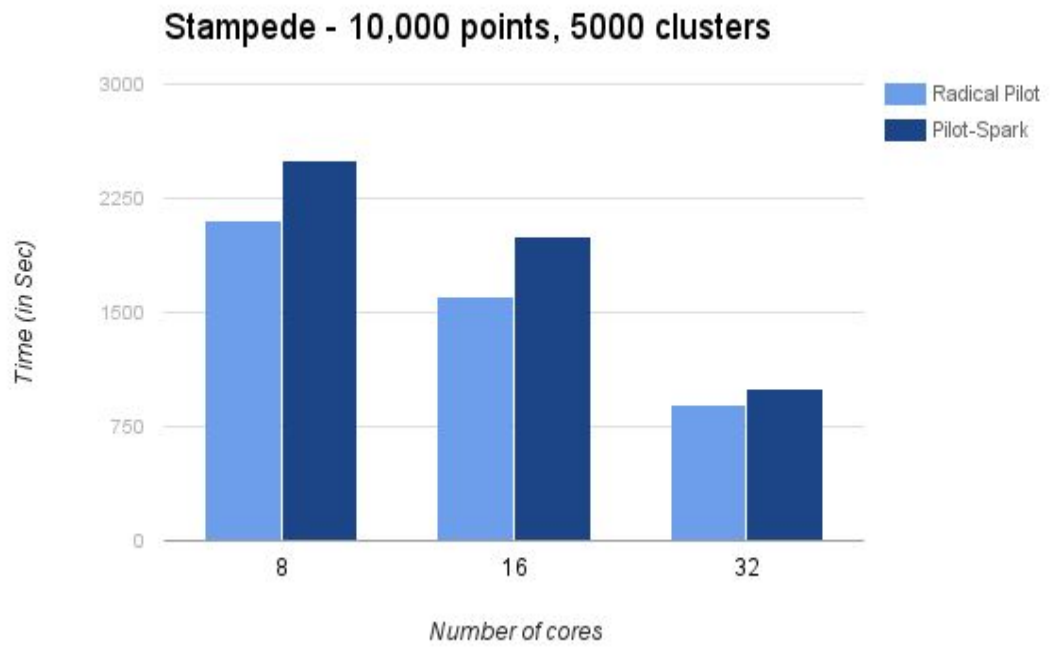
Stampede:

Scenario	Ram Memory (GB)	Number of Cores	Number of Nodes	Main Memory Type
all	30	8	1	HDD
all	30	16	1	HDD
all	60	32	2	HDD

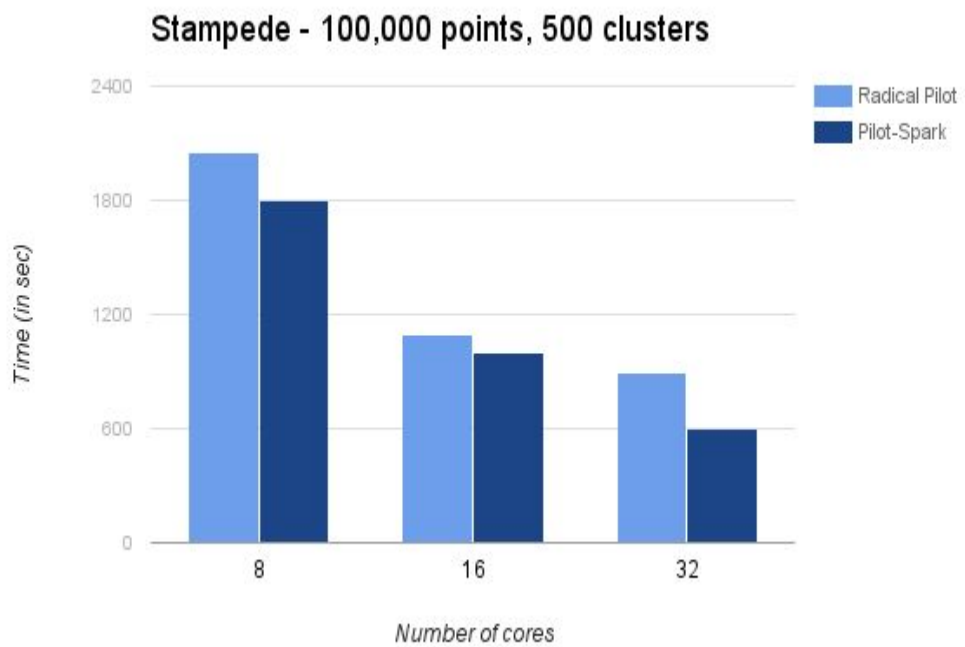
Wrangler:

Scenario	Ram Memory (GB)	Number of Cores	Number of Nodes	Main Memory Type
all	128	8	1	Flash
all	128	16	2	Flash
all	256	32	2	Flash

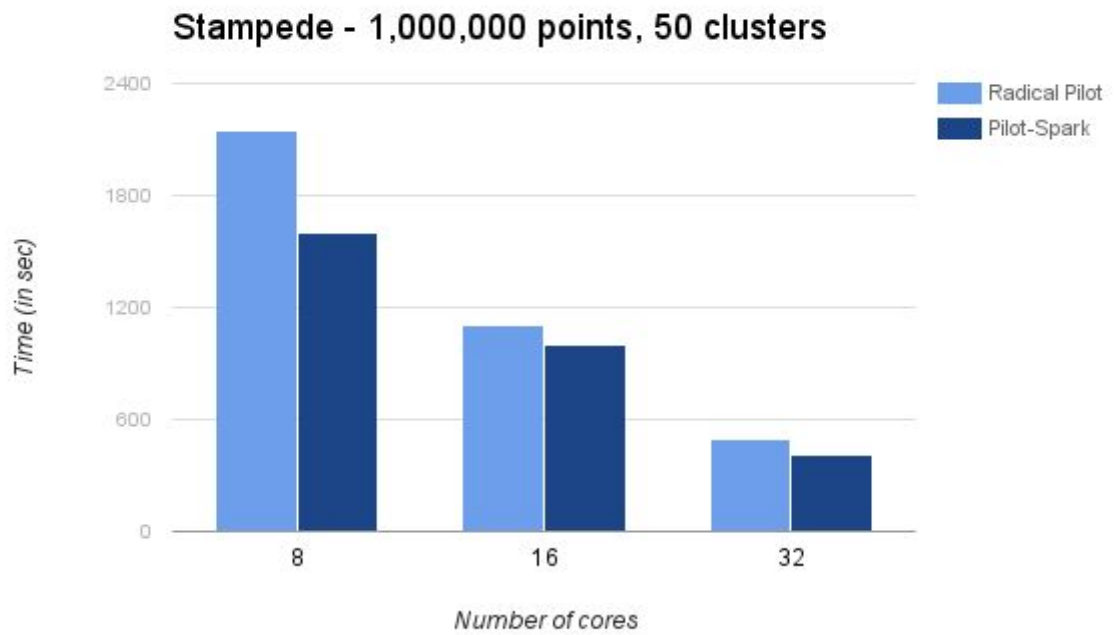




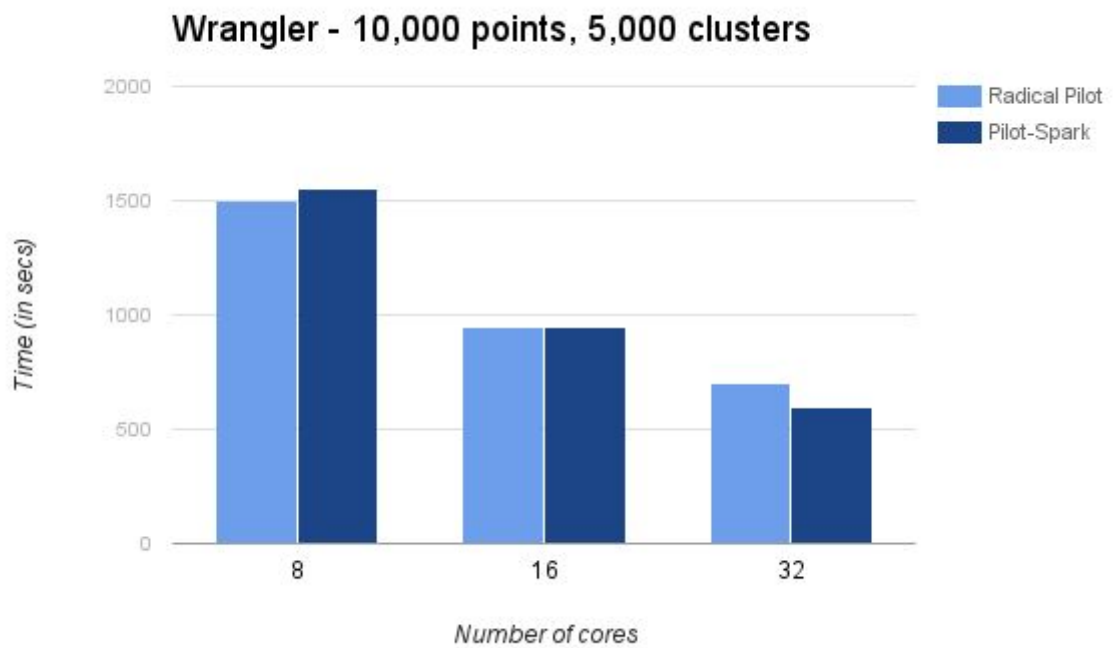
Σχήμα 6: k-means figure 1



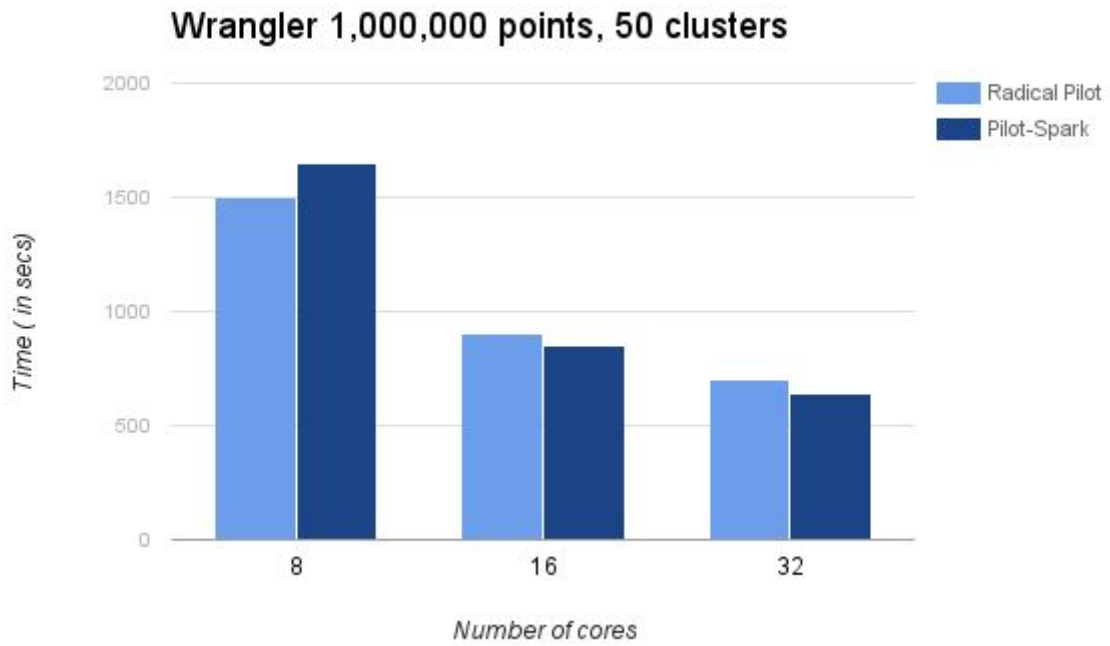
Σχήμα 7: k-means figure 2



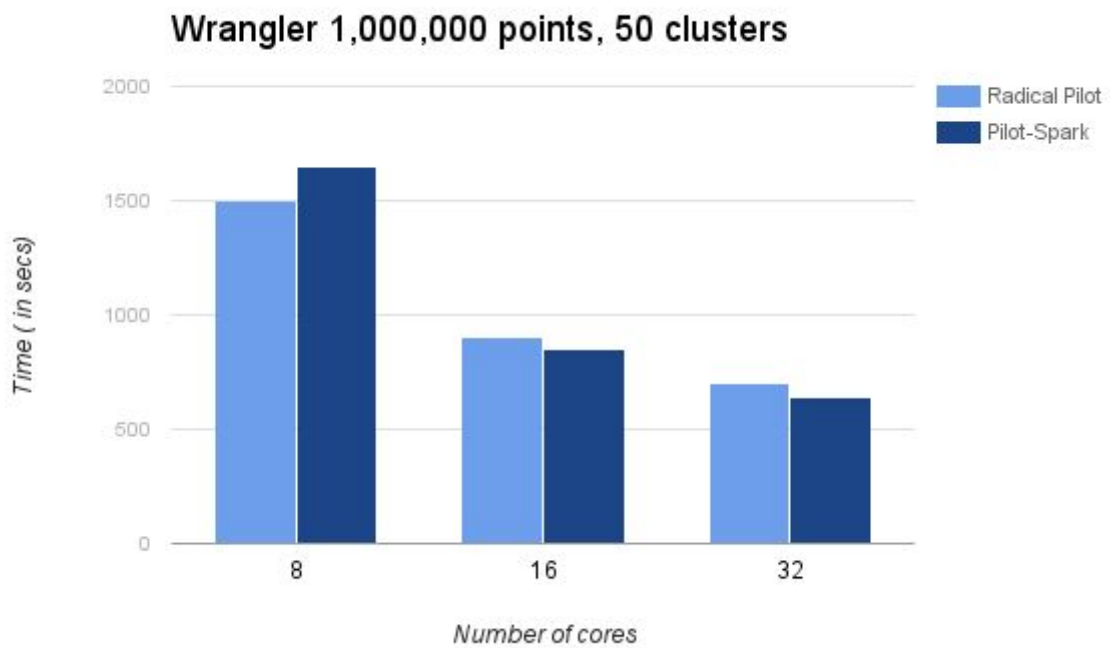
Σχήμα 8: k-means figure 3



Σχήμα 9: k-means figure 4



Σχήμα 10: k-means figure 5



Σχήμα 11: k-means figure 6

Μπορούμε να δούμε από τα στοιχεία ότι ο χρόνος ολοκλήρωσης είναι ταχύτερος για τη μηχανή Wrangler τόσο για το Radical Pilot όσο και για το Pilot-Spark. Ο κύριος λόγος για τη σημαντική βελτίωση της απόδοσης του Wrangler είναι η ποιότητα του υλικού, της CPU και των μνημών. Το Wrangler χρησιμοποιεί δίσκους SSD, οι οποίοι μπορούν να εκτελέσουν πολύ περισσότερες λειτουργίες ανάγνωσης και εγγραφής παράλληλα με τον σκληρό δίσκο Stampede.

Επιπλέον, σε μικρότερες εφαρμογές η απόδοση του radical-pilot είναι καλύτερη από αυτή του Pilot-Spark. Αυτό αναμένεται πλήρως λόγω του χρόνου έναρξης του Pilot-spark. Αν δεν βρεθεί, στον radical-pilot πρέπει να κάνει λήψη των δυαδικών ψηφίων τα οποία ξεπερνούν τα 300 δευτερόλεπτα. Η ώρα εκκίνησης εξαρτάται από την ταχύτητα δικτύου του απομακρυσμένου πόρου.

Παρ' όλα αυτά, μπορούμε να παρατηρήσουμε μια βελτίωση της απόδοσης του συνολικού χρόνου ολοκλήρωσης όταν χρησιμοποιούμε το pilot-spark.

## Κεφάλαιο 4

### 4. Εκτέλεση και μοντελοποίηση του αλγορίθμου leaflet-finder χρησιμοποιώντας το radical-Pilot και Pilot-Spark.

#### Επισκόπηση

Σε αυτό το κεφάλαιο παρουσιάζουμε τον αλγόριθμο leaflet-finder όπως έγινε χρησιμοποιώντας τη σουίτα του Radical Pilot System. Ξεκινάμε πάλι το κεφάλαιο περιγράφοντας όλη την πειραματική διαδικασία που ακολουθήθηκε για την εξαγωγή των δεδομένων στα οποία βασιζόμαστε για τα στάδια της ανάλυσης και της μοντελοποίησης. Στη συνέχεια αφιερώνεται ένα τμήμα στην περιγραφή των συνόλων δεδομένων που χρησιμοποιούν και της κατασκευής τους. Επίσης ένα τμήμα περιγράφει τον τρόπο με τον οποίο ο αλγόριθμος εκτελείται σε μηχανές TACC, Stampede και Wrangler και Comet, ενώ στις δύο τελευταίες ενότητες αυτού του κεφαλαίου αναλύονται και παρουσιάζονται τα αποτελέσματα των πειραμάτων μας. Τέλος, παρουσιάζονται τα μοντέλα που κατασκευάστηκαν και επαληθεύεται η ακρίβειά τους.

#### 4.1 Περιγραφή της Πειραματικής Διαδικασίας

Ο σκοπός της πειραματικής διαδικασίας είναι να εκτελέσει τον αλγόριθμο leaflet-finder χρησιμοποιώντας διαφορετικά πρωτότυπα σενάρια για να μελετήσει την απόδοσή του. Πραγματοποιούμε τα πειράματα τόσο με το radical-pilot όσο και με τον pilot-Spark. Είναι πολύ ενδιαφέρον να δούμε πώς λειτουργεί ο αλγόριθμος κατά την εκτέλεση του με διαφορετικό αριθμό πυρήνων cpu και με διαφορετικές υποδομές. (Διαφορετικές αρχιτεκτονικές, μνήμες, crus). Γι 'αυτό διενεργούμε τα πειράματά μας σε τρία διαφορετικά μηχανήματα, τον Stampede, τον Wrangler και τον Comet, και φτάνουμε μέχρι και 256 πυρήνες.

Συγκρίνουμε το χρόνο μέχρι την ολοκλήρωση του αλγορίθμου Finder-Finder που εκτελείται σε τρεις υποδομές HPC. Χρησιμοποιούμε τρία διαφορετικά σενάρια: Έχουμε αρχικές τροχιές με τρία διαφορετικά μεγέθη, μικρά, μεσαία τα οποία είναι διπλάσια από το μήκος των μικρών και μακρά που είναι τετραπλό με το μήκος των μικρών. Το σύντομο σενάριο έχει περίπου 49 χιλιάδες μονάδες. Κάθε σημείο ανήκει σε έναν τρισδιάστατο χώρο.

Το σύνολο δεδομένων μας έχει έναν πολύ μεγάλο αριθμό σημείων που πρέπει να υπολογίσουμε τις ευκλείδειες αποστάσεις για όλα τα ζεύγη. Με βάση τους υπολογισμούς μας, ο πίνακας γεινίασης μπορεί να φτάσει μέχρι και 80GB RAM ενός συστήματος ~ 150K, και πρέπει να υπολογίσει περίπου 10,6 εκατομμύρια άκρες. Επιπλέον, προσπαθούμε να δούμε πώς ο αλγόριθμος εκτελεί χρησιμοποιώντας διαφορετικές παραμέτρους, όπως ο αριθμός των crus, ο τύπος των μνημών. Όπως μπορεί εύκολα να γίνει κατανοητός από τον πειραματικό χώρο που προσπαθούμε να περιγράψουμε, επεκτείνεται πολύ γρήγορα σε μεγέθη που δεν είναι εύκολα παρατηρήσιμα και διαχειρίσιμα.

Για τον παραπάνω λόγο η καθοδήγηση της διαδικασίας σχεδιασμού μας είναι η προσεκτική επιλογή αυτών των δειγμάτων. Προκειμένου να παρέχουμε την πιο αντιπροσωπευτική εικόνα της απόδοσης του αλγορίθμου και επίσης να αποτρέψουμε την πρόκληση κακής δειγματοληψίας που στερείται χρήσιμων πληροφοριών και μελετώντας το νόημα, αποφασίσαμε να ζητήσουμε πρωτότυπα δεδομένα που παρέχονται από προσομοιώσεις σε πραγματικό χρόνο. Με αυτό είπαμε ότι υπάρχουν πραγματικές τροχιές μοριακής δυναμικής που παράγονται από εργαλεία όπως CHARMM, Gromacs, Amber, NAMD και LAMMPS.

Εκτελούμε τα πειράματα χρησιμοποιώντας RADICAL Pilot και Pilot Spark σε Stampede, Wrangler και Comet με 16, 32, 64, 128 πυρήνες και μετρήστε το Time to Completion για ένα πλαίσιο. Αποφασίσαμε να κλιμακώσουμε έως και 128 πυρήνες εξαιτίας της συμφόρησης των υπολογισμών των συνδεδεμένων εξαρτημάτων. Ενώ ο πίνακας γεινίασης κλιμακώθηκε καλά χτυπήσαμε μια συμφόρηση προσπαθώντας να υπολογίσουμε τα συνδεδεμένα στοιχεία. Περισσότερα από αυτά αργότερα.

Stampede:

Scenario	Ram Memory (GB)	Number of Cores	Number of Nodes	Main Memory Type
all	30	16	1	HDD
all	60	32	2	HDD

all	90	64	4	HDD
all	120	128	8	HDD

Wrangler:

Scenario	Ram Memory (GB)	Number of Cores	Number of Nodes	Main Memory Type
all	128	16	1	Flash
all	256	32	2	Flash
all	1280	64	5	Flash
all	1408	128	6	Flash

Comet:

Scenario	Ram Memory (GB)	Number of Cores	Number of Nodes	Main Memory Type
all	128	16	1	SSD
all	256	32	2	SSD
all	1280	64	5	SSD
all	1408	128	6	SSD

## 4.2 Εφαρμογή Leaflet Finder πάνω στο Radical Pilot

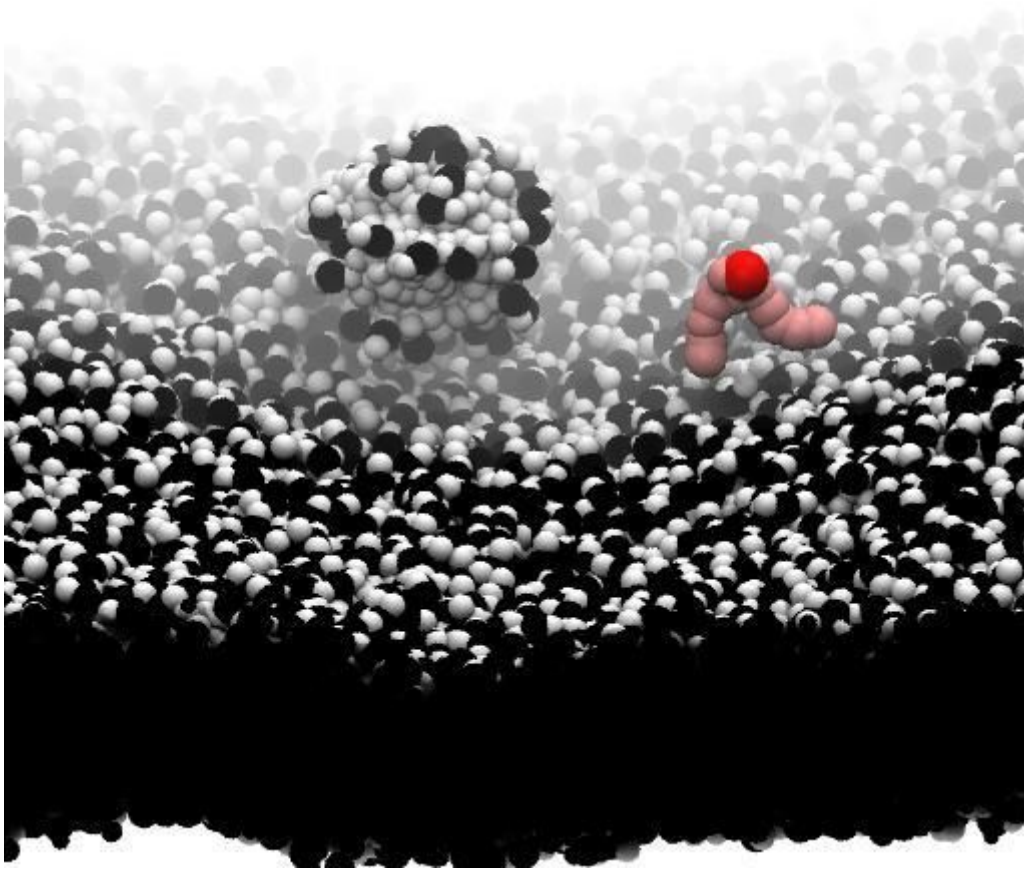
Η εφαρμογή του αλγορίθμου με χρήση πιλοτικού είναι μια σχετικά εύκολη διαδικασία. Το πρόγραμμα που γράψαμε στο rython και χρησιμοποιούσαμε πέρα από τον pilot τα παρακάτω πλαίσια, το πολυάριθμο για εύκολη ανάγνωση του συνόλου δεδομένων και το scikit-μάθαμε να υπολογίζουμε τις ευκλείδειες αποστάσεις μεταξύ των στοιχείων. Τέλος χρησιμοποιήσαμε το networkx για να υπολογίσουμε τα συνδεδεμένα στοιχεία του γραφήματος.

Τα αρχεία που παράγονται από τις τροχιές εξαγωγής που αναφέρονται παραπάνω δεν δίνουν απευθείας τις συντεταγμένες των οποίων χρησιμοποιήσαμε. Για το λόγο αυτό χρησιμοποιήσαμε το εργαλείο MDAnalysis [5] για να εξαγάγουμε τις συντεταγμένες που μελετούμε σε αυτό το έργο. Αυτό το πακέτο μας παρήγαγε αρχεία που δίνουν τη συμβολή στον αλγόριθμό μας. Αυτά τα αρχεία είναι σε μορφή numpy, η οποία είναι πολύ ευέλικτη, καθώς δεν χρησιμοποιεί μεγάλο χώρο αποθήκευσης και παρέχει επίσης πολλές δυνατότητες διαχείρισης αυτού του αρχείου, όπως άμεση εκτίμηση του μεγέθους του αρχείου, εύρεση του αριθμού Συντεταγμένες και πολλά άλλα.

Ο αλγόριθμος μετά την ανάγνωση του αρχείου εισόδου και η απόσταση αποκοπής που επιλέγει ο χρήστης, διανέμει το αρχείο στον απομακρυσμένο μηχάνημα σε κοινό φάκελο πρόσβασης του ριζοσπαστικού πιλότου. Εάν ο χρήστης δεν επιλέξει απόσταση αποκοπής, τότε χρησιμοποιείται η προεπιλεγμένη τιμή η οποία είναι 16. Τότε διαβάζουμε τον αριθμό των διαθέσιμων πυρήνων και αποφασίζουμε πόσες αποστάσεις θα υπολογίσει κάθε Υπολογιστική Μονάδα του Πιλοτικού.

Κάθε υπολογιστική μονάδα διαβάζει το αρχείο εισόδου και ποια στοιχεία πρέπει να υπολογίσει και αρχίζει τον υπολογισμό των αποστάσεων. Όπως αναφέρθηκε προηγουμένως, ο χώρος αποθήκευσης που απαιτείται για τις αποστάσεις των εκατομμυρίων ακρών είναι πολύ μεγάλος και αντισοικονομικός. Από την άλλη πλευρά, ο απώτερος στόχος του αλγορίθμου είναι να υπολογίσει τα φυλλάδια και ενδιαφέρεται για τα δεδομένα άμεσης απόστασης μετά από αυτό το βήμα του αλγορίθμου. Για το λόγο αυτό αποφασίσαμε να αποθηκεύσουμε τις αποστάσεις χρησιμοποιώντας μόνο έναν πίνακα αλήθειας. Εάν η τιμή απόστασης που υπολογίζεται περισσότερο από την απόσταση αποκοπής που έχει επιλέξει ο χρήστης, τότε θα υπολογίσουμε την απόσταση απεριόριστη και θα θέσουμε την τιμή στο μήτρα παρακέντησης στο μηδέν. Διαφορετικά θα ορίσουμε μια τιμή 1. Κάθε Compute-Unit αποθηκεύει τα αποτελέσματα σε ένα αρχείο κειμένου και το στέλνει σε ένα φάκελο στο μηχάνημα που είναι αναγνώσιμος από όλες τις Υπολογιστικές Μονάδες του πιλότου.

Μετά από όλες τις μονάδες Compute-Units να ολοκληρώσουν τους παραπάνω υπολογισμούς, τότε δημιουργήστε μια νέα εργασία η οποία δεσμεύεται να δημιουργήσει το πλήρες adjacency matrix και να υπολογίσει τα συνδεδεμένα στοιχεία του παραπάνω γραφήματος. Αυτό γίνεται με την ανάγνωση των παραπάνω αρχείων κειμένου και τα αποθηκεύει στη σωστή θέση ενός πίνακα NxN numpy. Ο λόγος που χρησιμοποιούμε το numpy matrix είναι ότι είναι αναγνώσιμο από το networkx. Το networkx παίρνει ως είσοδο τον πίνακα γειτνίασης και υπολογίζει όλα τα συνδεδεμένα στοιχεία του γραφήματος. Μας ενδιαφέρουν μόνο τα δύο μεγαλύτερα στοιχεία, ώστε να ταξινομούμε τα αποτελέσματα και να επιστρέψουμε μόνο τα δύο μεγαλύτερα.



Σχήμα 12: 9\_cg\_membrane\_outlier\_micelle1

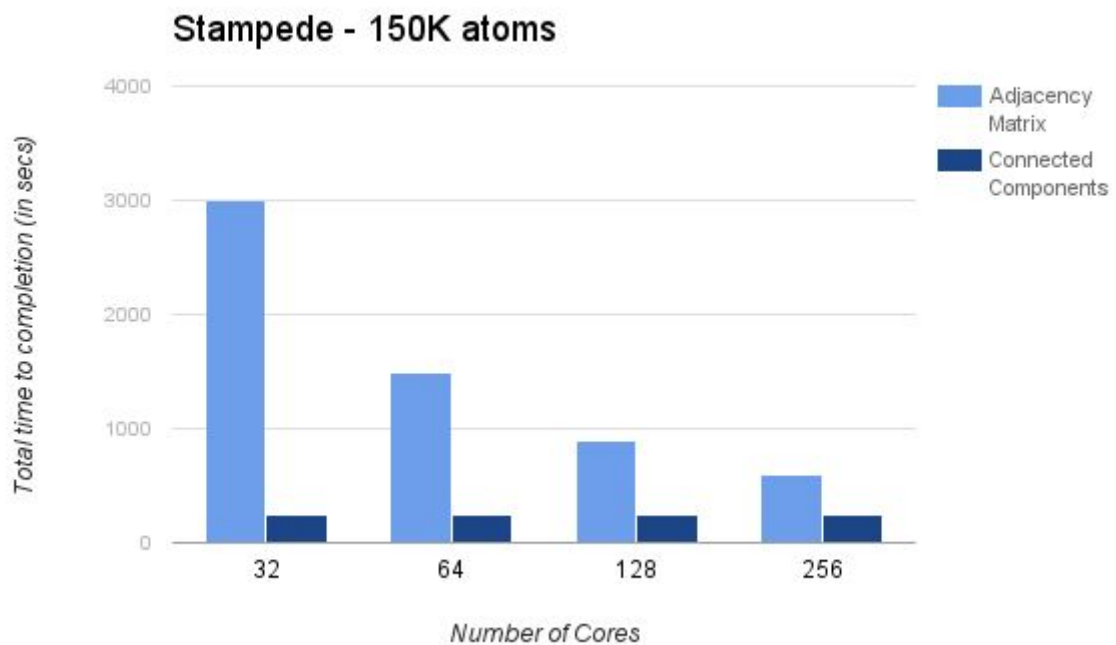
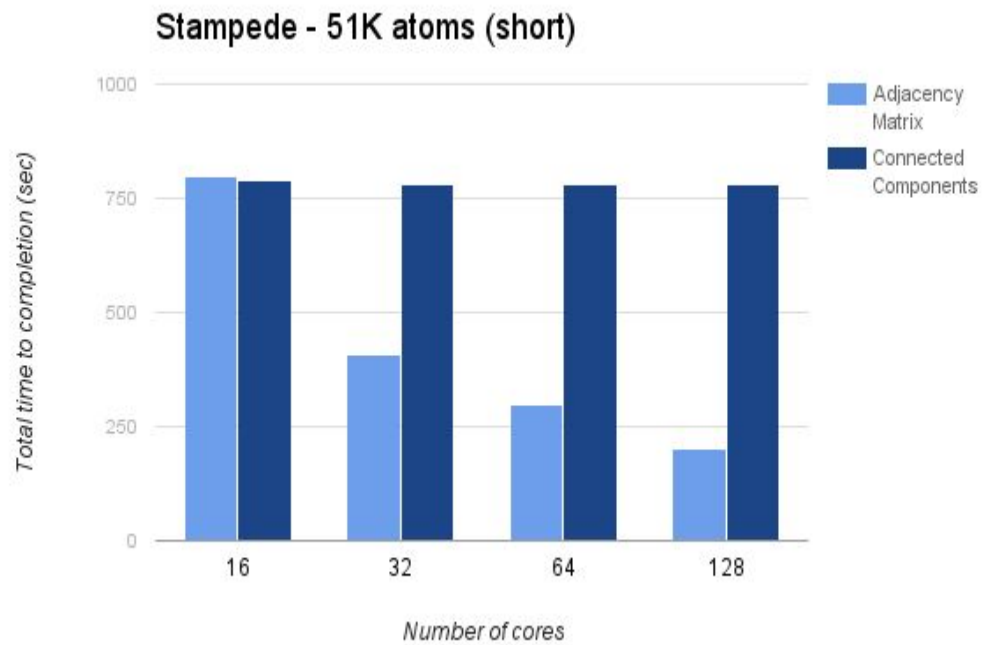
### 4.3 Αποτελέσματα Πειραμάτων

Σε αυτό το υποτμήμα παρουσιάζουμε και αναλύουμε τα αποτελέσματα των εκτελέσεων leaflet-finder για τις διάφορες παραμέτρους που είδαμε στην ενότητα 4.1. Επικεντρωνόμαστε στην παρατήρηση του συνολικού χρόνου ολοκλήρωσης της εφαρμογής τόσο σε ριζοσπαστικό πιλότο όσο και σε pilot-spark.

Εκτελούμε πειράματα με πυρήνες 16, 32, 64, 128 και 256.



Stampede:



# Κεφάλαιο 5

## 5. Επίλογος

Σε αυτή τη διπλωματική εργασία μελετήσαμε την απόδοση του k-μέσου και του αλγόριθμου εύρεσης φυλλαδίων στις υποδομές HPC και τις δοκιμάσαμε με δύο διαφορετικούς προγραμματιστικούς πόρους του radical-pilot . Πρόκειται για τον απρογραμμάτιστο προγραμματιστή του ριζοσπαστικού πιλότου και του προγραμματιστή του Spark, τον οποίο ονομάζουμε πιλότο-σπινθήρα. Διεξάγουμε μερικά πειράματα με διαφορετικές παραμέτρους εκτέλεσης για να κατανοήσουμε το συνολικό χρόνο ολοκλήρωσης των αλγορίθμων μας.

Αναπτύξαμε το σύνολο δεδομένων για τα πειράματά μας για k-means και τα χρησιμοποιούμε σε δύο διαφορετικές υποδομές HPC, Stampede και Wrangler. Περιορίσαμε τον αριθμό των επαναλήψεων του k-means αλγόριθμου σε δύο και τρέχουμε τα πειράματα με διαφορετικά σύνολα δεδομένων και διαφορετικό αριθμό πυρήνων.

Διαπιστώσαμε ότι όταν δεν χρησιμοποιούμε μεγάλο αριθμό πυρήνων, η απλή εφαρμογή του που αναπτύχθηκε πάνω στον radical-pilot είναι καλύτερη από την εφαρμογή που αναπτύχθηκε πάνω στον radical-pilot που συμβαίνει λόγω της αρχικής λειτουργίας του pilot-Spark. Όταν ζυγίζουμε σε 16 ή περισσότερα καθήκοντα τότε παρατηρούμε σημαντική βελτίωση της απόδοσης του pilot-Spark, στο συνολικό χρόνο ολοκλήρωσης που καλύπτει τον χρόνο εκκίνησης του πιλότου. Αυτό είναι αποτέλεσμα του εξαιρετικού σχεδιασμού της αρχιτεκτονικής εκτέλεσης του Spark.

Όσον αφορά το leaflet-finder χρησιμοποιήσαμε αρχικά δεδομένα για τον αλγόριθμο εύρεσης φυλλαδίων που δημιουργήθηκε από εργαλεία προσομοίωσης. Χρησιμοποιήσαμε τρεις διαφορετικές υποδομές HPC για τον leaflet-finder, Stampede, Wrangler και Comet και τρέχουμε τα πειράματα με τρία διαφορετικά σενάρια.

# Κεφάλαιο 6

## 6. Cyber-Infrastructure

### 6.1 Stampede

Το σύστημα TACC Stampede [3] είναι ένα συμπλέγμα Dell PFLOPS (PF) Dell που βασίζεται σε 6400+ κόμβους εξυπηρετητή Dell PowerEdge, εξοπλισμένο με 2 επεξεργαστές Intel Xeon E5 (Sandy Bridge) και έναν CoproProcessor Intel Xeon Phi (MIC Αρχιτεκτονική). Η συνολική

κορυφαία απόδοση των επεξεργαστών Xeon E5 είναι 2 + PF, ενώ οι επεξεργαστές Xeon Phi προσφέρουν πρόσθετη συνολική μέγιστη απόδοση 7 + PF. Το σύστημα περιλαμβάνει επίσης ένα σύνολο κόμβων σύνδεσης, κόμβων μεγάλης μνήμης, κόμβων γραφικών (τόσο για απομακρυσμένη απεικόνιση όσο και για υπολογισμό) και κόμβους διπλού επεξεργαστή. Πρόσθετοι κόμβοι (που δεν είναι άμεσα προσβάσιμοι από τους χρήστες) παρέχουν υπηρεσίες διαχείρισης και αρχείων.

Ένα από τα σημαντικά ζητήματα σχεδίασης του Stampede ήταν η δημιουργία ενός πόρου cyberinfrastructure πολλαπλών χρήσεων, που προσφέρει μεγάλη μνήμη, μεγάλη μεταφορά δεδομένων και δυνατότητες GPU για υπολογιστές με ένταση δεδομένων, επιτάχυνσης ή οπτικοποίησης. Αυξάνοντας μερικούς από τους κόμβους εντάσεως υπολογισμών μέσα στο σύστημα με πολύ μεγάλη μνήμη και GPU, δεν χρειάζεται να μετακινήσετε δεδομένα για υπολογιστές έντασης δεδομένων, απομακρυσμένη οπτικοποίηση και υπολογισμό GPGPU. Για τις καταστάσεις που απαιτούν μεταφορά μεγάλων δεδομένων από άλλους ιστότοπους, έχουν ενσωματωθεί 4 διακομιστές δεδομένων υψηλής ταχύτητας στα συστήματα αρχείων Luster.

Υπολογισμός Κόμβων: Η πλειοψηφία των 6400 κόμβων έχει διαμορφωθεί με δύο επεξεργαστές Xeon E5-2680 και έναν Coprocessor Intel Xeon Phi SE10P (σε κάρτα PCIe). Αυτοί οι υπολογιστικοί κόμβοι έχουν διαμορφωθεί με μνήμη "host" 32GB με πρόσθετη μνήμη 8GB στην κάρτα ομοεπεξεργαστή Xeon Phi. Ένας μικρότερος αριθμός υπολογιστικών κόμβων έχει ρυθμιστεί με δύο Xeon Phi Coprocessors - ο συγκεκριμένος αριθμός κόμβων που διατίθενται σε κάθε διαμόρφωση ανά πάσα στιγμή θα είναι διαθέσιμος από την περίληψη παρτίδας παρτίδας.

Συστήματα αρχείων: Το σύστημα Stampede υποστηρίζει μια παγκόσμια, παράλληλη αποθήκευση αρχείων 14 PB που διαχειρίζεται ως τρία συστήματα αρχείων Lustre. Κάθε κόμβος περιέχει έναν τοπικό δίσκο 250GB. Επίσης, το αρχειακό σύστημα ταινιών TACC Ranch (χωρητικότητα 60 PB) είναι προσβάσιμο από το Stampede.

Διασύνδεση: Οι κόμβοι διασυνδέονται με την τεχνολογία Mellanox FDR InfiniBand σε μια τοπολογία λιπαρών δένδρων δύο επιπέδων (πυρήνες και φύλλα).



## 6.2 Wrangler

Το σύστημα ανάλυσης και αποθήκευσης δεδομένων Wrangler [2] έχει σχεδιαστεί για τις ανάγκες των σύγχρονων ερευνητών δεδομένων. Η μοναδική αρχιτεκτονική της Wrangler χειρίζεται τις πολλές πτυχές του όγκου, της ταχύτητας και της ποικιλίας που μπορούν να καταστήσουν την έρευνα ψηφιακών δεδομένων δύσκολη στην εφαρμογή σε τυποποιημένα συστήματα υψηλής απόδοσης. Το σύστημα σχεδιάστηκε γύρω από ένα σύστημα αποθήκευσης φλας υψηλής ταχύτητας 0,5 PB που μπορεί να χρησιμοποιηθεί για τη διαχείριση των ροών εργασίας ανάλυσης δεδομένων και επεξεργασίας που δεν είναι πρακτικά σε άλλα συστήματα με πιο αργούς δίσκους περιστροφής ή σημαντικά μικρότερες εσωτερικές συσκευές αποθήκευσης SSD.

Το Wrangler παρέχεται δυναμικά από τους χρήστες με διαφορετικούς τρόπους για να χειρίζεται τις διαφορετικές ροές δεδομένων, συμπεριλαμβανομένων των βάσεων δεδομένων (τόσο συστήματα σχεσιακών βάσεων δεδομένων όσο και νεότερες βάσεις δεδομένων τύπου noSQL), ροές εργασίας Hadoop / HDFS (συμπεριλαμβανομένου του MapReduce και Spark) και πιο προσαρμοσμένες ροές εργασίας Βασισμένο στο παράλληλο σύστημα αρχείων. Εκτός από την ανάλυση και επεξεργασία δεδομένων, η Wrangler υποστηρίζει τις ανάγκες διατήρησης, κοινής χρήσης και δημοσίευσης δεδομένων για πολλά έργα δεδομένων.

### Technical Specifications of Wrangler System

COMPONENT	TECHNOLOGY	PERFORMANCE/SIZE
Compute Nodes (TACC)	dual CPU 12 Core Xeon E5-2680-v3	96 nodes/ 2304 cores
Memory (TACC)	Distributed DDR4	12.2 TB
Flash Storage	DSSD PCI attached NAND Flash	0.5 PB (200 TB Shared POSIX, 200 TB HDFS, 100 TB local attached/Object store)
Compute Nodes (IU)	dual CPU 12 core Xeon E5-2680-v3	24 nodes/576 cores
Shared Disk	Lustre, parallel file system	10 PB replicated, 34 OSS, 153 OST
Interconnect	Infiniband Mellanox Switch	FDR 54 Gbit/s Infiniband 40 Gb/s Ethernet

## 6.3 Comet

Ο Comet [1] είναι ο νέος πόρος HPC του SDSC, ένας supercomputer που έχει σχεδιαστεί για να μετασχηματίζει την προηγμένη επιστημονική πληροφορική, διευρύνοντας την πρόσβαση και την ικανότητα μεταξύ παραδοσιακών και μη παραδοσιακών ερευνητικών τομέων. Το αποτέλεσμα ενός βραβείου Εθνικού Ιδρύματος Επιστημών αξίας 21,6 εκατομμυρίων δολαρίων, συμπεριλαμβανομένου του υλικού και των κεφαλαίων εκμετάλλευσης, το Comet είναι ικανό για μια συνολική μέγιστη απόδοση δύο petaflops ή δύο quadrillion λειτουργίες ανά δευτερόλεπτο.



Ο Comet ενώνει τον supercomputer Gordon της SDSC ως έναν άλλο βασικό πόρο στο πρόγραμμα XSEDE του NSF (Extreme Science and Engineering Discovery Environment), το οποίο περιλαμβάνει την πιο προηγμένη συλλογή ολοκληρωμένων ψηφιακών πόρων και υπηρεσιών στον κόσμο.

"Ο κομήτης είναι υπεύθυνος για την παροχή υπολογιστών υψηλής απόδοσης σε μια πολύ μεγαλύτερη ερευνητική κοινότητα - αυτό που αποκαλούμε HPC για το 99 τοις εκατό - και λειτουργεί ως πύλη για την ανακάλυψη", δήλωσε ο Norman, ο κύριος ερευνητής του προγράμματος. "Ο Comet έχει σχεδιαστεί ειδικά για να καλύψει τις ανάγκες των ερευνητών που δεν καλύπτονται σε τομείς που δεν βασίζονται παραδοσιακά σε υπερυπολογιστές για να βοηθήσουν στην επίλυση προβλημάτων, σε αντίθεση με τον τρόπο με τον οποίο έχουν ιστορικά χρησιμοποιηθεί τέτοια συστήματα".

Ο Comet είναι διαμορφωμένος έτσι ώστε να παρέχει μια λύση για αναδυόμενες ερευνητικές απαιτήσεις που συχνά αναφέρονται ως η «μακρά ουρά» της επιστήμης, η οποία περιγράφει την ιδέα ότι ο μεγάλος αριθμός ερευνητικών έργων μέτριου μεγέθους, που βασίζονται σε υπολογισμούς, εξακολουθεί να αντιπροσωπεύει συνολικά μια τεράστια ποσότητα της έρευνας και των συνακόλουθων επιστημονικών επιπτώσεων και εκ των προτέρων. Ο Comet μπορεί να υποστηρίξει χρήστες μικρής κλίμακας σε ολόκληρο το φάσμα των κοινοτήτων NSF, ενώ ταυτόχρονα καλωσορίζει τις ερευνητικές κοινότητες που συνήθως δεν

χρησιμοποιούν πιο παραδοσιακά συστήματα HPC, όπως η γονιδιωματική, οι κοινωνικές επιστήμες και η οικονομία.

Το Comet είναι ένα ολοκληρωμένο cluster της Dell που χρησιμοποιεί την οικογένεια Xeon® Processor E5-2600 v3 της Intel, με δύο επεξεργαστές ανά κόμβο και 12 πυρήνες ανά επεξεργαστή σε 2,5GHz. Κάθε υπολογιστικός κόμβος διαθέτει 128 GB (gigabytes) παραδοσιακής DRAM και 320 GB τοπικής μνήμης flash. Δεδομένου ότι ο Comet έχει σχεδιαστεί για τη βελτιστοποίηση της χωρητικότητας για εργασίες μέτριας κλίμακας, κάθε σχάρα 72 κόμβων (1.728 πυρήνες) έχει μια πλήρη διασύνδεση InfiniBand FDR από το Mellanox, με υπερκάλυψη 4: 1 στα ράφια. Υπάρχουν 27 ράφια αυτών των υπολογιστικών κόμβων, συνολικού ύψους 1.944 κόμβων ή 46.656 πυρήνων.

Επιπλέον, ο Comet έχει τέσσερις κόμβους μεγάλης μνήμης, ο καθένας με τέσσερις υποδοχές 16 πυρήνων και μνήμη 1,5 TB, καθώς και 36 κόμβους GPU, ο καθένας με τέσσερις μονάδες GPU NVIDIA (γραφικές μονάδες επεξεργασίας). Οι μονάδες GPU και οι κόμβοι μεγάλης μνήμης προορίζονται για συγκεκριμένες εφαρμογές όπως οπτικοποιήσεις, προσομοιώσεις μοριακής δυναμικής ή συναρμολόγηση γονιδιώματος de novo.

Οι χρήστες κομητών θα έχουν πρόσβαση στο 7.6 PB του χώρου αποθήκευσης, καθώς το σύστημα παράλληλης αποθήκευσης αρχείων Data Oasis της SDSC αναβαθμίζεται σημαντικά. Το σύστημα διαμορφώνεται με συνδεσιμότητα 100 Gbps (Gigabit ανά δευτερόλεπτο) με το Internet2 και το ESNet, επιτρέποντας στους χρήστες να μετακινούν γρήγορα δεδομένα στο SDSC για ανάλυση και ανταλλαγή δεδομένων και να επιστρέφουν δεδομένα στα ιδρύματά τους για τοπική χρήση. Μέχρι το καλοκαίρι του 2015, το Comet θα είναι το πρώτο σύστημα παραγωγής XSEDE για την υποστήριξη virtualization υψηλής απόδοσης σε επίπεδο συμπλέγματος πολλαπλών κόμβων. Η χρήση Virtue I / O Single Root (SR-IOV) από τον Comet σημαίνει ότι οι ερευνητές μπορούν να χρησιμοποιήσουν το δικό τους περιβάλλον λογισμικού όπως συμβαίνει με το cloud computing, αλλά μπορούν να επιτύχουν την υψηλή απόδοση που αναμένουν από έναν υπερυπολογιστή

System	Performance	Key Features
Comet Summary	~2 Pflop/s peak; 47,776 compute cores; 247 TB total memory; 634 TB total flash memory	<p><b>Compute Nodes (1944 total)</b></p> <p>Intel Xeon E5-2680v3 2.5 GHz dual socket, 12 cores/socket; 320 GB SSD local scratch memory; 120 GB/s memory bandwidth</p> <p><b>GPU Nodes (36 total)</b></p> <p>2 NVIDIA K-80 GPUs per node; dual socket, 12 cores/socket; 128 GB DDR4 DRAM; 120GB/s memory bandwidth; 320 GB flash memory</p>

		<p><b>Large-memory Nodes (4 total)</b></p> <p>1.5 TB total memory; 4 sockets, 16 cores/socket; 2.2 GHz</p> <p><b>Interconnect</b></p> <p>Hybrid Fat-Tree topology; 56 Gb/s (bidirectional) link bandwidth; 1.03-1.97 <math>\mu</math>s MPI latency</p> <p><b>Lustre-based Parallel File System</b></p> <p>Access to Data Oasis</p>
--	--	--

## 7. Βιβλιογραφία

- [1] "HPC Systems - San Diego Supercomputer Center." 2015. 13 Jul. 2016  
<[http://www.sdsc.edu/services/hpc/hpc\\_systems.html](http://www.sdsc.edu/services/hpc/hpc_systems.html)>
- [2] "XSEDE User Portal | TACC Wrangler." 2014. 13 Jul. 2016  
<<https://portal.xsede.org/tacc-wrangler>>
- [3] "XSEDE User Portal | TACC Stampede." 2012. 13 Jul. 2016  
<<https://portal.xsede.org/tacc-stampede>>
- [4] "RADICAL Cybertools." 2014. 13 Jul. 2016 <<https://radical-cybertools.github.io/>>
- [5] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. MDAAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.* **32**(2011), 2319-2327, doi:10.1002/jcc.21787. PMID:PMC3144279
- [6] "Lustre (file system) - Wikipedia, the free encyclopedia." 2011. 14 Jul. 2016  
<[https://en.wikipedia.org/wiki/Lustre\\_\(file\\_system\)](https://en.wikipedia.org/wiki/Lustre_(file_system))>
- [7] A SPACE-EFFICIENT PARALLEL ALGORITHM FOR COUNTING EXACT TRIANGLES IN MASSIVE NETWORKS. Shaikh Arifuzzaman, Maleq Khan and Madhav Marathe. 17th IEEE International Conference on High Performance Computing and Communications (HPCC), New York City, Aug. 2015.
- [8] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In Supercomputing, 1996. Proceedings of the 1996 ACM/IEEE Conference on, pages 39–39, 1996.
- [9] Andre Luckow, Mark Santcroos, Andre Merzky, Ole Weidner, Pradeep Mantha, and Shantenu Jha. P\*: A model of pilot-abstractions. IEEE 8th International Conference on e-Science, pages 1–10, 2012. <<http://dx.doi.org/10.1109/eScience.2012.6404423>. >
- [10] Andre Luckow, Mark Santcroos, Ashley Zebrowski, and Shantenu Jha. Pilot-Data: An Abstraction for Distributed Data. *Journal Parallel and Distributed Computing*, October<sup>1</sup> 2014. <<http://dx.doi.org/10.1016/j.jpdc.2014.09.009>.>
- [11] "Overview — NetworkX." 2013. 21 Jul. 2016 <<https://networkx.github.io/>>
- [12] Dean, J. "MapReduce: Simplified Data Processing on Large ... - Google Research." 2004. <<http://research.google.com/archive/mapreduce-osdi04.pdf>>



- [13] "Clustering - Introduction." 2013. 21 Jul. 2016  
<[http://home.deib.polimi.it/matteucc/Clustering/tutorial\\_html/](http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/)>
- [14] "TutorialsPoint." 2005. 21 Jul. 2016 <<http://www.tutorialspoint.com/>>
- [15] "cpython: 309f3559a729 Lib/random.py - Python.org mercurial ..." 2014. 21 Jul. 2016  
<<https://hg.python.org/cpython/file/2.7/Lib/random.py>>
- [16] "Molecular dynamics - Wikipedia, the free encyclopedia." 2011. 21 Jul. 2016  
<[https://en.wikipedia.org/wiki/Molecular\\_dynamics](https://en.wikipedia.org/wiki/Molecular_dynamics)>
- [17] "Leaflet identification — MDAnalysis.analysis ... - PythonHosted.org." 2014. 21 Jul. 2016  
<[https://pythonhosted.org/MDAnalysis/documentation\\_pages/analysis/leaflet.html](https://pythonhosted.org/MDAnalysis/documentation_pages/analysis/leaflet.html)>