

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ροές Εργασιών σε Κατανεμημένα Συστήματα

Ιωάννης Δ. Καλύβας

Επιβλέπων : **Ιωάννης Ιωαννίδης,** Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ροές Εργασιών σε Κατανεμημένα Συστήματα

Ιωάννης Δ. Καλύβας
Α.Μ.: 1115201000041

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Ιωαννίδης,** Καθηγητής ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Στην σύγχρονη εποχή, η οποία χαρακτηρίζεται ως εποχή της πληροφορίας, η ανάγκη για ανάκτηση, αποθήκευση και επεξεργασία των τεράστιων όγκων δεδομένων που δημιουργούνται ανά πάσα στιγμή καθώς και εκτέλεση αλγορίθμων που απαιτούν μεγάλη υπολογιστική ισχύ οδήγησε στην μετατόπιση των υπολογιστικών συστημάτων από ατομικές μονάδες επεξεργασίας σε συστάδες (clusters) υπολογιστών, στους οποίους ο φόρτος εργασίας διαμοιράζεται σε κάθε κόμβο (node) με σκοπό την συντομότερη περάτωση κάθε διεργασίας μέσω της παραλληλοποίησης.

Σκοπός αυτής της πτυχιακής εργασίας είναι η μελέτη συστημάτων που ελέγχουν, κατανέμουν και επεξεργάζονται αυτά τα δεδομένα και βοηθούν στην επικοινωνία μεταξύ των κόμβων. Τέτοια συστήματα ονομάζονται Μηχανές Ροών Εργασιών (Workflow Engines) .Σε αυτή την πτυχιακή εργασία παρουσιάζονται ορισμένα βασικά παραδείγματα αυτων των συστημάτων όπως το Apache Spark το Apache Hadoop και το Apache Taverna και γίνεται μια βασική σύγκριση μεταξύ τους.

Τέλος παρουσιάζεται η εφαρμογή που αναπτύχθηκε πάνω στο Apache Spark μέσω της οποίας γίνεται μια προσπάθεια να απλοποιηθεί η δημιουργία ερωτημάτων (queries) σε αυτό το σύστημα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ροές Εργασιών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μαζικά δεδομένα ,Apache Spark,HDFS, Hadoop, Κατανεμημένα συστήματα, Νέφη υπολογιστικών συστημάτων, συστάδες

ABSTRACT

In the modern era,which is characterized as the era of information,the need for retrieval,storage and processing of the all increasing data and the execution of complex resource consuming algorithms led to the migration of computing processing from single powerful computer systems to clusters,in which the workload is equally distributed among the nodes to achieve fast results through parallelization.

The goal of this theses is the study of such systems that control,distribute and process data and manage the communication among the nodes.These systems are called Workflow Engines.We present some basic examples of these systems such as Apache Spark,Apache Hadoop and Apache Taverna and we try to compare them.

Finally we present the application which was built on top of Spark where we try to simplify the process of creating and submitting Spark Jobs on a Spark Cluster.

SUBJECT AREA: Workflow Engines

KEYWORDS: Big Data, Apache Spark, HDFS, Apache Hadoop, Distributed Systems, cloud, clusters

ΕΥΧΑΡΙΣΤΙΕΣ

Για την παρούσα πτυχιακή ευχαριστώ πολύ τον κ. Ιωαννίδη Ιωάννη,Καθηγητή του Τμήματος Πληροφορικής και Τηλεπικοινωνιών.

ΠΕΡΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 10

1ΕΙΣΑΓΩΓΗ **11**

2. ΥΠΟΒΑΘΡΟ **12**

2.1 Ροές Εργασιών **12**
2.1.1 Γενικά 12
2.1.2 Μοντέλα Ροών Εργασιών 12
2.1.2.1 Επιστημονικές Ροές Εργασιών 12
2.1.2.2 Η ροή εργασιών του MapReduce 13
2.1.2.3 Ο Κατευθυνόμενος Ακυκλικός Γράφος ως ροή εργασιών 14

2.2 Παραδείγματα Σύγχρονων Συστημάτων Ροών Εργασιών **15**
2.2.1 Taverna 15
2.2.1.1 Πλεονεκτήματα και μειονεκτήματα του Apache Taverna 15
2.2.1.2 Συμπέρασμα 16
2.2.2 Apache Hadoop 16
2.2.2.1 Hadoop Yarn 17
2.2.2.2 Hadoop Distributed FileSystem 18
2.2.2.3 Hadoop MapReduce 19
2.2.2.4 Σύνοψη 19
2.2.3 Apache Spark 20
2.2.3.1 RDD(Resilient Distributed Dataset) 20
2.2.3.2 RDD vs Hadoop MapReduce Ή DAG vs MapReduce 21
2.2.3.3 Spark SQL 24
2.2.3.4 Spark Streaming 25
2.2.3.5 Συμπέρασμα 25
2.2.4 Apache Flink 26
2.2.4.1 Δεδομένα συνεχούς ροής(Streaming Data) στο Flink 26
2.2.4.2 Επίπεδα Αφαίρεσης στο Apache Flink 26
2.2.4.3 Συμπέρασμα 27

3. ΕΦΑΡΜΟΓΗ **28**
3.1 Γενικά 28
3.1.1 Spring Framework 28
3.1.2 BootStrap CSS 28
3.1.3 Thymeleaf 28

3.2 Spark Template Engine **29**
3.2.1 Το οικοσυστημα της εφαρμογης 29
3.2.2 Ο πυρήνας της εφαρμογής 29
3.2.3 Αποτελέσματα ενός ερωτήματος 31
3.2.4 Spark Template Script Language 31

ΣΥΜΠΕΡΆΣΜΑΤΑ **33**

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ **34**

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ **35**

ΑΝΑΦΟΡΕΣ **36**

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1: ΣΧΗΜΑΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΠΟΥ ΑΝΑΠΤΥΧΘΗΚΕ….32

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1:Τυπικό Παράδειγμα μιας MapReduce ροης εργασιων……..13
Εικόνα 2:Ο Κατευθυνόμενος Ακυκλικός Γράφος……14
Εικόνα 3:Το Workbench του Apache Taverna……16
Εικόνα 4:Σχηματικη Απεικόνιση των οντοτήτων του Hadoop Yarn……..17
Εικόνα 5:Ο καταμερισμός και διαχωρισμός του Spark RDD……....22
Εικόνα 6:Ένα παραδειγμα πρότυπου(Template) προγράμματος σε Spark……....33
Εικόνα 7:Μια προτυποποιημενη(Template) συνάρτηση σε Spark……34
Εικόνα 8: Η γραμματική της γλωσσάς για τα Spark Templates……35

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1:Παράδειγμα 1 Lazy Evaluation στο Spark…….23
Πίνακας 2:Αποτέλεσμα Παράδειγματος 1 Lazy Evaluation στο Spark……24
Πίνακας 3:Παράδειγμα 2 Lazy Evaluation στο Spark……....24
Πίνακας 4:Αποτέλεσμα Παράδειγματος 2 Lazy Evaluation στο Spark……25

ΠΡΟΛΟΓΟΣ

Η πτυχιακή αυτή έχει σαν στόχο την μελέτη διάφορων Μηχανών Ροών Εργασιών και την ανάπτυξη μιας διαδικτυακής εφαρμογής που έχει σαν σκοπό την απλοποίηση του Apache Spark μέσω της παροχής προτύπων (templates). Περατώθηκε στην Αθήνα κατά το διάστημα μεταξύ Ιανουαριου 2016 και Ιανουαριου 2017

1 ΕΙΣΑΓΩΓΗ

Στην σύγχρονη εποχή, η οποία χαρακτηρίζεται ως εποχή της πληροφορίας, η ανάγκη για ανάκτηση, αποθήκευση και επεξεργασία των τεράστιων όγκων δεδομένων που δημιουργούνται ανά πάσα στιγμή καθώς και εκτέλεση αλγορίθμων που απαιτούν μεγάλη υπολογιστική ισχύ οδήγησε στην μετατόπιση των υπολογιστικών συστημάτων από ατομικές μονάδες επεξεργασίας σε συστάδες (clusters) υπολογιστών, στους οποίους ο φόρτος εργασίας διαμοιράζεται σε κάθε κόμβο (node) με σκοπό την συντομότερη περάτωση κάθε διεργασίας μέσω της παραλληλοποίησης.

Δεδομένων αυτών των αναγκών ξεκίνησε και η ανάπτυξη συστημάτων που θα μπορούσαν να διαμοιράσουν σωστά τον φόρτο σε κάθε κόμβο ανάλογα με τις δυνατότητες και τους πόρους τους και να αναλάβουν την επεξεργασία των δεδομένων. Τετοια συστήματα όπως το HTCondor, το Taverna και κυρίως τα δυο συστήματα της Apache για αναλύσεις μεγάλου όγκου δεδομένων(big data analytics) το Apache Spark και το Apache Hadoop είναι το αντικείμενο αυτής της πτυχιακής. Ιδίως τα δυο τελευταία θα αποτελέσουν και τα βασικά εργαλεία της εφαρμογής που αναπτύχθηκε σαν μέρος της πτυχιακής.

Αυτά τα συστήματα ονομάζονται Μηχανές Ροών Εργασιών και αναλαμβάνουν την κατανομή των εργασιών ανάλογα με τις δυνατότητες του κάθε κόμβου και την επεξεργασία των δεδομένων .

2. ΥΠΟΒΑΘΡΟ

Στο κεφάλαιο αυτό γίνεται μια προσπάθεια θεωρητικής εισαγωγής στις έννοιες των ροών εργασιών και των συστημάτων διαχείρισης ροών εργασιών και παρουσιάζονται ορισμένα από αυτά τα συστήματα όπως το Apache Taverna και το Apache Flink.Στην συνέχεια δίνεται ιδιαίτερη έμφαση στο Apache Spark και το Apache Hadoop που χρησιμοποιήθηκαν και στην εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας. Τέλος παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την διαδικτυακή υποστήριξη της εφαρμογής και πιο συγκεκριμένα το Spring Framework καθως και το Bootstrap CSS

2.1 Ροές Εργασιών

2.1.1 Γενικά

Μια ροή εργασιών είναι ένα σύνολο βημάτων τα οποία συνδέονται μεταξύ τους

με κάποιο τρόπο και έχουν σκοπό την διεκπεραίωση μιας εργασίας. Αυτά τα βήματα μπορεί επί παραδείγματι να είναι διαδικασίες μετασχηματισμού των δεδομένων , διαδικασίες καταμερισμού δεδομένων ,διαδικασίες παροχής υπηρεσιών και πάρα πολλές άλλες υπηρεσίες που εξαρτώνται με το σύστημα που χρησιμοποιείται κάθε φορά.

Σε πιο αφηρημένο επίπεδο ορίστηκε ως ροή εργασιών από τον Συνασπισμό Διαχείρισης Ροών Εργασιών (Workflow Management Coalition) "Η αυτοματοποίηση μιας διαδικασίας, κατά την οποία έγγραφα ,πληροφορίες ή εργασίες περνούν από έναν παραλήπτη σε έναν άλλον για να πραγματοποιηθούν ενέργειες σύμφωνα με ένα σύνολο διαδικαστικών κανόνων ".

2.1.2 Μοντέλα Ροών Εργασιών

2.1.2.1 Επιστημονικές Ροές Εργασιών

Οι επιστημονικές ροές εργασιών είναι εξειδικευμένες ροές εργασιών σχεδιασμένες να επεξεργάζονται ή να συνθέτουν μεγάλους όγκους δεδομένων ή να εκτελούν αλγορίθμους μεγάλης πολυπλοκότητας που απαιτούν ισχυρή υπολογιστική ισχύ .Τέτοιες ροές εργασιών μπορούν να συνδυαστούν με κατανεμημένα συστήματα υπολογιστών ώστε να φέρουν σε πέρας τις πολύπλοκες αυτές διεργασίες. Αυτό επιτυγχάνεται μέσω της κατανομής των διεργασιών ανάμεσα στους κόμβους του κατανεμημένου συστήματος και τον ορισμό των ενεργειών που θα πραγματοποιήσει κάθε κόμβος για την δημιουργία του αποτελέσματος . Το αποτέλεσμα αυτό μπορεί να είναι το τελικό αποτέλεσμα της διαδικασίας ή η είσοδος δεδομένων ενός επομένου κόμβου στην ροή εργασίας.

Ορισμένα συστήματα τέτοιου είδους όπως το Apache Taverna, που μελετάται στην συνέχεια, παρέχει και οπτική απεικόνιση των ροών και επιτρέπει στους χρήστες την κατασκευή τους μέσω του γραφικού του περιβάλλοντος.Η ικανότητα αυτή στο Apache Taverna βοηθάει τους επιστήμονες από πεδία άσχετα με τον προγραμματισμό να οργανώσουν την εκτέλεση μιας πλήρους και εξειδικευμένης ροής εργασιών χρησιμοποιώντας εργαλεία που βρίσκονται κυρίως στο διαδίκτυο αλλά και στο τοπικό

τους σύστημα.Με αυτό τον τρόπο μπορούν να δημιουργήσουν και να επαναλάβουν ενα πείραμα τους πάνω σε δεδομένα χωρις να ασχολούνται με το προγραμματιστικό τμήμα της ροής εργασιών.

2.1.2.2 Η ροή εργασιών του MapReduce

Το MapReduce είναι ένα προγραμματιστικό μοντέλο για επεξεργασία δεδομένων σε κατανεμημένα συστήματα που αναπτύχθηκε από την Google το 2004.Είναι ένα απλό στην κατανόηση προγραμματιστικό μοντέλο που βασίζεται σε δυο διαδικασίες ,μια που ονομάζεται Map και άλλη μια που ονομάζεται Reduce ,από της οποίες παίρνει και το όνομά του. Παρουσιάζει αρκετές ομοιότητες με παλαιότερα μοντέλα παράλληλης επεξεργασίας δεδομένων όπως το Scatter και Reduce του MPI.

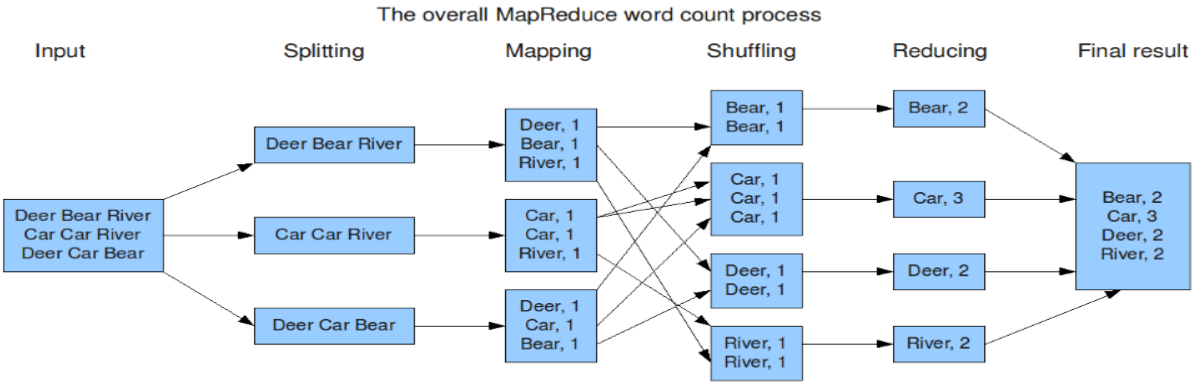
Κατά την διαδικασία Map τα δεδομένα σχηματίζουν ζευγάρια κλειδιού-τιμής (key-value pairs) ενώ κατά την διαδικασία του Reduce συγκεντρώνονται και υπολογίζονται τα αποτελέσματα με χρήση του κλειδιού κάθε ζευγαριού.Το MapReduce μπορεί να πραγματοποιηθεί σε μια μεγάλη ποικιλία δεδομένων,είτε αυτά είναι δομημένα είτε αδόμητα,καθώς είναι αρκετά εύκολο σχεδόν όλα τα είδη δεδομένων να μετασχηματιστούν σε ζευγάρια κλειδιού-τιμής. Σε αυτό βοηθάει ότι δεν υπάρχει κάποιος περιορισμός στο είδους του κλειδιού ή της τιμής καθώς μπορεί να είναι αριθμοί,ακολουθίες χαρακτήρων,άλλοι τύποι δεδομένων ακόμα και άλλα ζευγάρια κλειδιού-τιμής.

Λόγω αυτών των δυνατοτήτων του το MapReduce χρησιμοποιείται ευρέως σαν προγραμματιστικό μοντέλο σε εφαρμογές που λειτουργούν πάνω σε συστάδες (clusters) υπολογιστών.Μια ακόμα δυνατότητα του μοντέλου είναι η ικανότητα του ανακατανέμει τα δεδομένα ,που έχουν περάσει από την διαδικασία του Map, ανάλογα με τα κλειδιά τους. Με αυτή την διαδικασία τα ζευγάρια κλειδιού-τιμής που εχουν κοινα κλειδια συγκεντρωνονται στον ίδιο κόμβο και συνεπώς διευκολύνεται η διαδικασία υπολογισμού μέσω της Reduce διαδικασίας αφού πλέον γίνεται τοπικά σε κάθε κόμβο. Συνοψίζοντας το MapReduce χωρίζεται στα παρακάτω βασικά στάδια :

- **Διαδικασία "Map" (k1,v1)=> (map(k1,v1)=(k2,v2))**: Κάθε κόμβος-εργάτης(worker or slave node) εκτελεί την μέθοδο map() που έχει οριστεί στο τοπικό τμήμα δεδομένων που του ανατίθεται από τον κόμβο-αφέντη (master node) και αποθηκεύει τα δεδομένα σε κάποιο προσωρινό σύστημα αποθήκευσης.Αποτελεί δηλαδή μια μετατροπή των δεδομένων απο μια μόρφη σε μια άλλη χωρίζοντας τα με βάση κάποιο κλειδί.

- **Διαδικασία "Shuffle"**:Ανάλογα με τα κλειδιά των ζευγαριών κλειδιού-τιμής που σχηματίστηκαν στο προηγούμενο στάδιο,οι κόμβοι εργάτες αναδιανέμουν τα δεδομένα έτσι ώστε κάθε ζευγάρι με κοινό κλειδί να βρίσκεται στον ίδιο κόμβο εργάτη
- **Διαδικασία "Reduce" (k2,v2)=>(k2,reduce(list(v2))=(k3,v3)):**Με βάση την μέθοδο που έχει οριστεί κάθε κόμβος επεξεργάζεται τα δεδομένα που του έχουν ανατεθεί παράλληλα ανάλογα με τα κλειδιά τους μεταφέροντας τα δεδομένα που έχουν ίδια κλειδία σε ίδιους κόμβους και εφαρμόζοντας πάνω στα δεδομένα με το ίδιο κλειδί

την συνάρτηση που έχει οριστεί .Εν ολίγοις τα δύο στάδια shuffle και reduce μπορούν να θεωρηθούν σαν ένα.



Εικόνα 1:Τυπικό Παράδειγμα μιας MapReduce ροης εργασιων

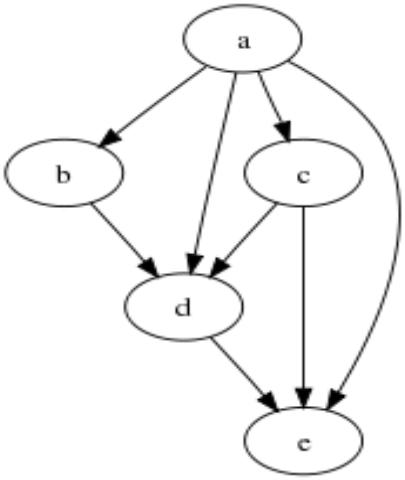
Στην εικόνα 2.1 φαίνεται ένα από τα χαρακτηριστικότερα παραδείγματα μια ροής εργασιών MapReduce το πρόγραμμα WordCount στο οποίο μετριέται η εμφάνιση λέξεων σε ένα κειμένου.Αρχικά (επίπεδο Splitting) το κειμένου που δίνεται σαν είσοδος στο MapReduce διασπάται ανα γραμμή και ανατίθεται σε κάθε κόμβο της συστάδας υπολογιστών (cluster).Στην συνέχεια (επίπεδο Mapping) κάθε κόμβος επεξεργάζεται την γραμμή κειμένου που του έχει ανατεθεί και δημιουργεί ζευγάρια κλειδιού-τιμής με κλειδί την λέξη και τιμή τον αριθμό ένα .Στο επόμενο επίπεδο (επίπεδο Shuffling) οι κόμβοι ανακατανέμουν μεταξύ τους τα ζευγάρια που ο κάθε ένας έχει φτιάξει με τέτοιο τρόπο ώστε ζευγάρια με ίδιο κλειδί να βρίσκονται στον ίδιο κόμβων συνέχεια (επίπεδο Reduce) οι κόμβοι εφαρμόζουν την μέθοδο Reduce που τους έχει δοθεί και προσθέτουν τα πεδία τιμών στα ζευγάρια ανάλογα με το εκάστοτε κλειδί και σχηματίζονται τα τελικά αποτελέσματα με τον αριθμό εμφάνισης της κάθε λέξης σε ένα κείμενο.Τέλος (final result) τα δεδομένα συγκεντρώνονται στον κόμβο αφέντη(master node) και είναι έτοιμα για συνολική παρουσίαση .

2.1.2.3 Ο Κατευθυνόμενος Ακυκλικός Γράφος ως ροή εργασιών

Στα μαθηματικά και στην πληροφορική ο Κατευθυνόμενος Ακυκλικός Γράφος είναι ένας γράφος με συγκεκριμένη κατεύθυνση από έναν κόμβο σε έναν άλλον και με την προϋπόθεση ότι δεν υπάρχει μονοπάτι που να ξεκινάει από έναν κόμβο και να επιστρέφει στον ίδιο.Έτσι ο Κατευθυνόμενος Ακυκλικός Γράφος μπορεί να χρησιμοποιηθεί στις ροές εργασιών καθώς μπορεί να μοντελοποιήσει την μετάβαση από το ένα στάδιο μια διεργασίας στο επόμενο.

Στα σύγχρονα συστήματα διαχείρισης ροών εργασιών εφαρμόζονται τεχνικές με χρήση του Κατευθυνόμενου Ακυκλικού Γράφου σαν ένα επίπεδο αφαίρεσης του Map Reduce.Πιο συγκεκριμένα τα στάδια των μετασχηματισμών των δεδομένων μπορούν να θεωρηθούν σαν κόμβοι του γράφου ενώ οι ακμές του είναι οι συναρτήσεις που μετασχηματίζουν τα δεδομένα.Με την χρήση του Κατευθυνόμενου Γράφου για την

έκφραση της ροής των εργασίων επιτρέπεται πλέον στα σύγχρονα συστήματα να πετύχουν βελτιστοποιήσεις επιλέγοντας το βέλτιστο μονοπάτι στον γράφο.



Εικόνα 2: Ο Κατευθυνόμενος Ακυκλικός Γράφος

2.2 Παραδείγματα Σύγχρονων Συστημάτων Ροών Εργασιών

2.2.1 Taverna

Το Taverna είναι ένα σύστημα διαχείρισης Ροών Εργασιών ελεύθερου λογισμικού γραμμένο σε Java.Ανήκει πλέον στο Apache Incubator,τον προθάλαμο δηλαδή του Apache Software Foundation που αποτελεί την σημαντικότερη κοινότητα για συστήματα ελεύθερου λογισμικού.Είναι ένα λογισμικό που χρησιμοποιείται αρκετά στην επιστημονική κοινότητα για την εκτέλεση ροών εργασιών από επιστήμονες με ελάχιστο υπόβαθρο στον προγραμματισμό

2.2.1.1 Πλεονεκτήματα και μειονεκτήματα του Apache Taverna

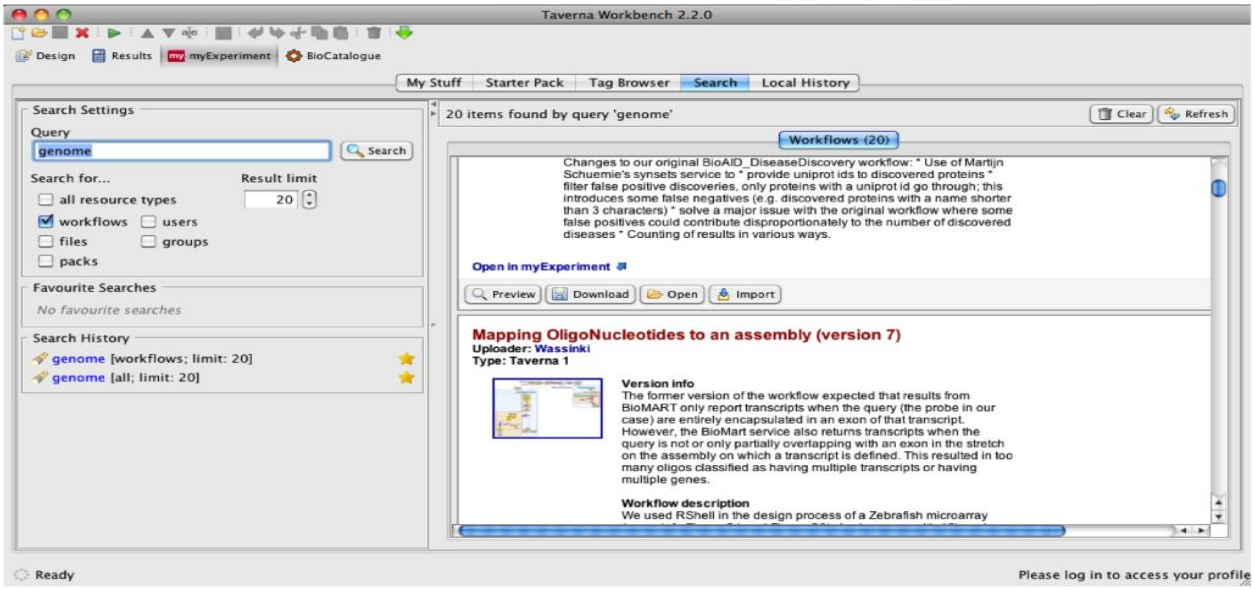
Ένα από τα βασικά πλεονεκτήματα του Apache Taverna είναι η ικανότητα που δίνει στον χρήστη να σχεδιάζει τις ροές εργασιών μέσα από ένα γραφικό περιβάλλον (εικόνα 2.2.1.1).Με αυτό τον τρόπο γίνεται εύκολο για τον μέσο χρήστη με ελάχιστη ή μηδενική γνώση προγραμματισμού να σχεδιάσει και να εκτελέσει αναλύσεις πάνω σε επιστημονικά δεδομένα με ελάχιστο κόπο.Επίσης η δυνατότητα του Apache Taverna να επιτρέπει την χρήση διαφόρων πρωτοκόλλων υπηρεσιών διαδικτύου (Web Services), όπως το Rest και τα WDSL-style πρωτόκολλα,προσφέρει στον χρήστη τα εργαλεία να χτίσει πολύπλοκες Ροές Εργασιών χρησιμοποιώντας υπηρεσίες οπουδήποτε στο διαδίκτυο.Με αυτόν τρόπο ο χρήστης μπορεί να εκτελέσει Ροές Εργασιών πάνω σε δεδομένα χωρίς να χρειαστεί να γράψει κώδικα απλά γνωρίζοντας την είσοδο και την έξοδο των υπηρεσιών διαδικτύου που έχει σκοπό να χρησιμοποιήσει

Από την άλλη ένα από τα βασικά πλεονεκτήματα του Apache Taverna είναι και βασικό του μειονέκτημα ικανότητα του Apache Taverna να συνδυάζει διαφορετικά είδη υπηρεσιών (τοπικές ή διαδικτυακές εργασίες) μέσα σε μια Ροή Εργασιών και το γεγονός ότι η επιλογή αυτών των εργασιών εναποτίθενται στον χρήστη δεν επιτρέπει στο σύστημα

να μπορεί να κατανείμει τον φόρτο των βημάτων της Ροής Εργασιών σε ένα κατανεμημένο σύστημα

2.2.1.2 Συμπέρασμα

Το Apache Taverna είναι ένα δυνατό εργαλείο για την επιστημονική κοινότητα που μπορεί να συνδυάσει και να αυτοματοποιήσει διαδικασίες που αλλιώς θα γίνονταν "χειροκίνητα".Είναι πολύ χρήσιμο στο να δημιουργηθούν ροές Εργασιών από έναν χρήστη που δεν έχει μεγάλο προγραμματιστικό υπόβαθρο και θέλει να μεταφέρει δεδομένα και να τα επεξεργαστεί από ένα σημείο Α σε ένα Β.Δεν συνιστάται όμως για προβλήματα ανάλυσης μεγάλου όγκου δεδομένων γιατί θα πρέπει να επιλεχθούν από τον χρήστη τα διάφορα βήματα της ροής εργασιών και η σειρά που θα εκτελεστούν και δεν υπάρχει προεγκατεστημένο σύστημα κατανομής των δεδομένων έτσι ώστε αυτό να γίνει με μεγαλύτερη ταχύτητα.΄Ετσι αυξάνεται ο χρόνος και η πολυπλοκότητα στην δημιουργία αναλύσεων σε μεγάλους όγκους δεδομένων.



Εικόνα 3. Το Workbench του Apache Taverna προσφέρει την δυνατότητα δημιουργίας Workflows μέσα από ένα γραφικό περιβάλλον

2.2.2 Apache Hadoop

Εμπνευσμένο από δυο δημοσιεύσεις της Google (το Google File System Paper και το Map Reduce:Simplified Data Processing On Large Clusters) το Apache Hadoop είναι ένα λογισμικού ανοικτού κώδικα για αποθήκευση και ανάλυση μεγάλου όγκου δεδομένων γραμμένο κατά κύριο λόγο σε Java.Η δημιουργία του ξεκίνησε το 2006 με την πρώτη ολοκληρωμένη έκδοση του να δημοσιεύεται το 2011.Από τότε αποτελεί μια από τις βασικές λύσεις για προβλήματα αναλύσεων μεγάλου όγκου δεδομένων χρησιμοποιώντας συστάδες υπολογιστών.Ο πυρήνας των υπολογισμών του Hadoop βασίζεται στο μοντέλο

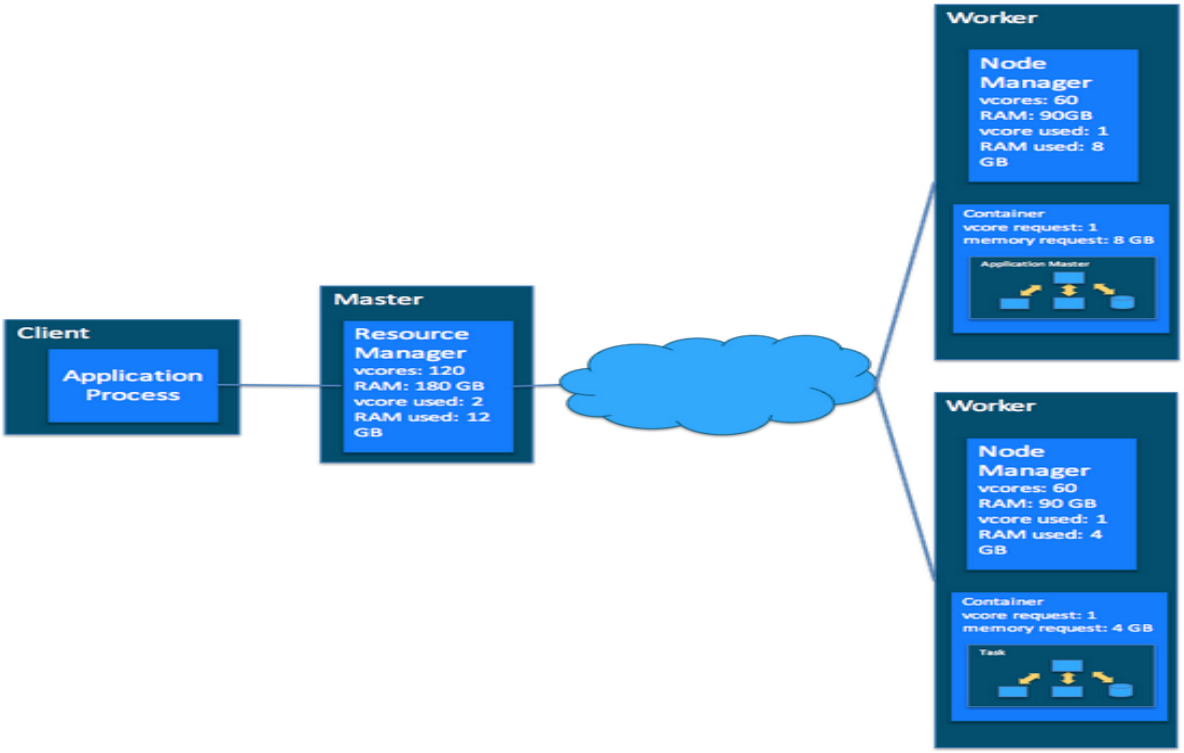
του MapReduce.Επίσης το Hadoop προσφέρει ένα πολύ ικανό σύστημα αποθήκευσης δεδομένων,το HDFS (Hadoop DIstributed File System), και ένα σύστημα διαχείρισης των προσφερόμενων πόρων μιας συστάδας υπολογιστών το Yarn.

2.2.2.1 Hadoop Yarn

Το Yarn είναι μια πλατφόρμα διαχείρισης πόρων σε μια συστάδα υπολογιστών που αναλαμβάνει το τρόπο που θα μοιραστούν αυτοί οι πόροι σε κάθε εφαρμογή που ζητάει πρ'οσβαση.Χωρίζει τους κόμβους σε μια συστάδα υπολογιστών σε δυο κατηγορίες,τον ResourceManager που έχει την έννοια του master και τους NodeManagers που έχουν την έννοια των slaves.Ο ResourceManager αναλαμβάνει την επικοινωνία με την εκάστοτε εφαρμογή που ζητάει πόρους,ελέγχει την επάρκεια τον πόρων στην συστάδα και αναθέτει σε κάθε κόμβο (NodeManager) την αντίστοιχη εργασία που πρέπει να φέρει εις πέρας.Ο NodeManager αναλαμβάνει την διαχείριση της εργασίας που ξεκινάει στον υπολογιστή που βρίσκεται και επιστρέφει τα αποτελέσματα της στον ResourceManager. Κάθε εφαρμογή που θέλει να τρέξει σε έναν Yarn Cluster στέλνει ένα αίτημα στον ResourceManager με τους πόρους που θέλει να χρησιμοποιήσει. Στην συνέχεια ο

ResourceManager του Yarn αποφασίζει πότε και σε ποιον ή ποιους NodeManagers θα αναθέσει την εκτέλεση της εφαρμογής

Το Hadoop Yarn μπορεί να χρησιμοποιηθεί πάνω από το Hadoop Distributed FIleSystem και υποστηρίζει διάφορες εφαρμογές αναζήτησης εκτός του Hadoop MapReduce όπως το Apache Spark,το ElasticSearch και άλλα .



Εικόνα 4. Σχηματικη Απεικόνιση των οντοτήτων του Hadoop Yarn

2.2.2.2 Hadoop Distributed FileSystem

Το HDFS είναι ένα από τα στοιχεία κλειδιά του Είναι το βασικό αποθηκευτικό μέσο του Hadoop και υποστηρίζει καταμερισμό των δεδομένων στους κόμβους ενός Hadoop Cluster.Ο μεγαλύτερος Hadoop Cluster υποστηρίζει 4500 κόμβους με δυνατότητα αποθήκευσης στο HDFS 2ΟΟ PB δεδομένων Αποτελείται από έναν κεντρικό κόμβο που ονομάζεται Namenode που διαχειρίζεται τα μεταδεδομένα του HDFS cluster και πολλούς κόμβους που ονομάζονται Datanodes και χρησιμοποιούνται για να αποθηκευτούν τα δεδομένα.Τα αρχεία με τα δεδομένα χωρίζονται σε μεγάλα κομμάτια(blocks) και μοιράζονται ανάμεσα στους κόμβους(Datanodes) του HDFS.

Επειδή το HDFS σχεδιάστηκε για να χρησιμοποιείται σε Εμπορικούς (Commodity) Servers που είναι επιρρεπής σε σφάλματα υλικού κάθε block δεδομένων αποθηκεύεται σε παραπάνω από έναν κόμβους.Με αυτό τον τρόπο εξασφαλίζεται ότι αν ένας κόμβος σταματήσει να λειτουργεί τα δεδομένα που υπάρχουν σε αυτόν θα μπορούν να ανακτηθούν από κάποιον άλλον κόμβο του Cluster.Ο παράγοντας που ορίζει πόσα αντίγραφα κάθε block θα αποθηκεύονται σε άλλους κόμβους ονομάζεται Replication Factor και ορίζεται στο αρχείο ρυθμίσεων του Όταν ένας κόμβος δεδομένων αποτυγχάνει

ο κεντρικός κόμβος Namenode αντιγράφει τα δεδομένα σε έναν άλλο κόμβο ώστε ο Replication Factor να παραμένει σταθερός.Έτσι εξασφαλίζεται ότι ένα HDFS Cluster θα είναι ανεκτικός σε σφάλματα υλικού

Οι κομβοι ενός HDFS Cluster επικοινωνουν μεταξυ τους για να ανακατανεμουν τα δεδομενα όταν δουν ότι σε καποιο κομβο υπαρχει ελευθερος χωρος.Ετσι επιτυγχανεται η ισομερης κατανομη των δεδομενων αναμεσα στους κομβους του HDFS.Η επικοινωνια αναμεσα στους κομβους γινεται μεσω του TCP/IP.

Για την ανάκτηση,την ανάγνωση και την επεξεργασία δεδομένων σε ένα HDFS το

Hadoop προσφέρει μια διεπαφή προγραμματισμού εφαρμογών σε Java (Java API) καθώς και ένα σύστημα εντολών κελύφους σαν αυτό του UNIX Filesystem.

Το Hadoop Distributed FileSystem αποτελεί ίσως το πιο δυνατό χαρτί αυτή την στιγμή για το Apache Hadoop καθώς είναι το καλύτερο κατανεμημένο σύστημα αποθήκευσης δεδομένων(datastore).Γι' αυτό τον λόγο οι περισσότερες εφαρμογές για αναλύσεις μεγάλου όγκου δεδομένων στηρίζονται στο HDFS σαν αποθηκευτικό μέσο ανεξάρτητα από το αν θα χρησιμοποιήσουν οποιοδήποτε άλλο από τα εργαλεία του Hadoop(Yarn, MapReduce).

2.2.2.3 Hadoop MapReduce

Πάνω από το FileSystem και το Yarn σε ένα Hadoop Cluster τρέχει το MapReduce του Hadoop.Το Hadoop MapReduce είναι ένα πλαίσιο λογισμικού(Software Framework) που υλοποιεί τον κλασσικό προγραμματιστικό μοντέλο του MapReduce.Τα δεδομένα χωρίζονται σε μεγάλα κομμάτια και επεξεργάζονται από τις Map διαδικασίες έξοδοι αυτών των διαδικασιών στην συνέχεια γίνονται είσοδοι σε Reduce διαδικασίες είσοδοι και οι έξοδοι αποθηκεύονται στο Filesystem.

Για να πετύχει μεγαλύτερη ταχύτητα πάνω στις αναζητήσεις το framework του MapReduce προσπαθεί να πετύχει τυπικότητα δεδομένων μέσω του Master κόμβου του που ονομάζεται JobTracker.Ο JobTracker αναλαμβάνει την επιλογή των κόμβων στους οποίους θα τρέξουν τα Map και Reduce Tasks.Στην επιλογή αυτή προτιμά είτε τον κόμβο στον οποίο βρίσκονται τα δεδομένα είτε κάποιον άλλο κόμβο που βρίσκεται στο ίδιο rack αν αυτός δεν είναι διαθέσιμος.Έτσι σε πρώτο πλάνο επιδιώκει την τοπικότητα των δεδομένων και σε δεύτερο την χαμηλή επιβάρυνση του δικτύου που υποστηρίζει τον cluster.

Παρά τις προσπάθειες που κάνει για να πετύχει μεγαλύτερη ταχύτητα το Hadoop MapReduce έχει ξεπεραστεί από πιο σύγχρονα συστήματα ανάλυσης δεδομένων.Το γεγονός ότι για κάθε στάδιο του Map και του Reduce διαβάζει και γράφει στο filesystem καθώς επίσης και το γεγονός ότι βλέπει κάθε Map και Reduce ξεχωριστά και αφού ολοκληρωθούν συνεχίζει στα επόμενα δεν του επιτρέπουν να συναγωνιστεί άλλες τεχνολογίες που έχουν αναπτυχθεί λαμβάνοντας υπόψιν αυτούς του παράγοντες.Αποτέλει όμως μια αξιόπιστη λύση για ένα γηγενές σύστημα μέσα στο ευρύτερο οικοσύστημα του Apache Hadoop(HDFS,Yarn)

2.2.2.4 Σύνοψη

Το Apache Hadoop είναι μια από τις πιο δυνατές προτάσεις στον τομέα της ανάλυσης μεγάλου όγκου δεδομένων.Τουλάχιστον ένα τμήμα από αυτά που το αποτελούν χρησιμοποιείται από πάρα πολλές εταιρείες ανα τον κόσμο.Παρά το γεγονός ότι πλέον το MapReduce που εφαρμόζει το Hadoop ίσως θεωρείται ξεπερασμένο,το HDFS και το

Yarn είναι δυο πολύ ικανές λύσεις για αυτό που προσφέρουν και χρησιμοποιούνται από τους "αντίπαλους" του MapReduce σαν αποθηκευτικά μέσα και διαχειριστές συστάδων υπολογιστών αντίστοιχα

2.2.3 Apache Spark

Το Apache Spark ξεκίνησε να αναπτύσσεται από το AMPLab του πανεπιστημίου του Berkeley και στην συνέχεια δωρίστηκε στο Apache Software Foundation.Ειναι ένα λογισμικό ελεύθερου κώδικα για αναλύσεις δεδομένων σε συστάδες υπολογιστών.Αποτελεί αυτή την στιγμή την βασική επιλογή για αναλύσεις μεγάλου όγκου δεδομένων και αλγορίθμων που τρέχουν σε συστάδες.Το βασικό στοιχείο που το έχει φέρει στην κορυφή είναι η επιλογή του να μεταφέρει τα δεδομένα από τον δίσκο στην μνήμη και με αυτόν τον τρόπο να γλιτώσει από την καθυστέρηση της εγγραφής/ανάγνωσης δεδομένων από τον δίσκο

Η βασική δομή που χρησιμοποιεί το Spark για να μεταφέρει τα δεδομένα από τον δίσκο στην μνήμη είναι τα RDDs(Resilient Distributed Datasets).Είναι αμετάβλητες(immutable) δομές που μπορούν να διαμοιραστούν παράλληλα στους κόμβους μιας συστάδας υπολογιστών και περιέχουν δεδομένα σε διάφορες μορφές

Αυτή την στιγμή το Apache Spark μέσω των διαφόρων συστατικών του όπως το GraphX,SparkSql ,SparkMLlib και SparkStreaming χρησιμοποιείται για να προσφέρει λύσεις σε πολλά προβλήματα όπως η επεξεργασία γράφων ,οι machine learning αλγόριθμοι και τα ζωντανά δεδομένα

Σε αντίθεση με το Hadoop,το Spark δεν προσφέρει ολοκληρωμένη λύση για διαχείριση μεγάλου όγκου δεδομένων και απαιτεί κατά ελάχιστον ένα ξεχωριστό κατανεμημένο σύστημα αποθήκευσης δεδομένων.Στην περίπτωση της διαχείρισης των πόρων μια συστάδας υπολογιστών υποστηρίζει δικό του σύστημα (Spark Standalone) αλλά είναι επίσης συμβατό με το Hadoop Yarn και το Apache Mesos.Στην ουσία είναι μια βελτιστοποίηση του Hadoop MapReduce.

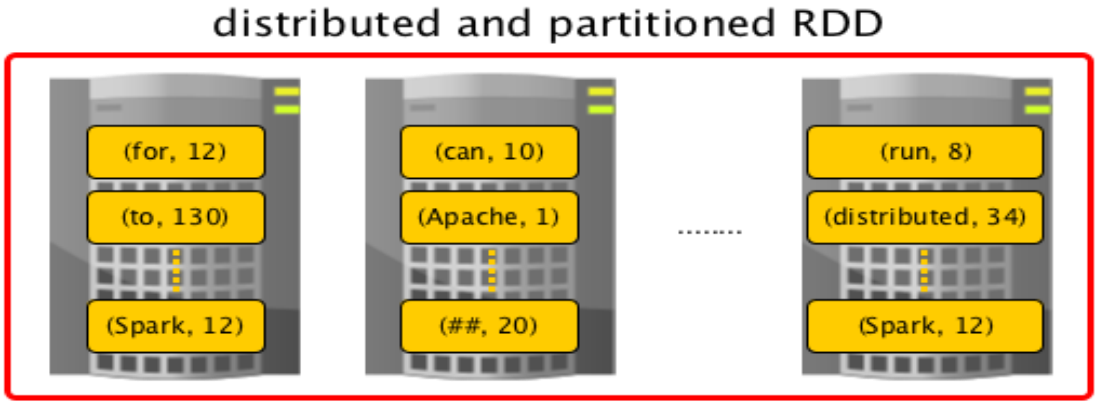
2.2.3.1 RDD(Resilient Distributed Dataset)

Το RDD είναι η κεντρική δομή του Apache Spark και κατοικεί για όσο το δυνατόν περισσότερο στην μνήμη.Είναι το βασικό μέσο αφαίρεσης δεδομένων του Spark.Το όνομα του προέρχεται από τις βασικές αρχές που το διέπουν.Είναι **Resilient** δηλαδή ανθεκτικό σε σφάλματα (fault tolerant).Για να πετύχει αυτή την ανθεκτικότητα το RDD χρησιμοποιεί την έννοια του γράφου ιεραρχίας(lineage graph) που θα δούμε στο 2.2.3.2.Καθε RDD κρατάει τα βήματα στον γράφο από τα οποία έχει προέλθει και έτσι αν σε κάποιο σημείο ένα RDD περιέχει κατεστραμμένα δεδομένα μπορεί να ανακατασκευαστεί ακολουθώντας τον γράφο ιεραρχίας.Ειναι **Distributed** (κατανεμημένο) καθώς τα δεδομένα ενός RDD μπορούν να βρίσκονται σε πολλούς διαφορετικούς κόμβους μέσα σε μια συστάδα υπολογιστών.Τέλος είναι **Dataset** δηλαδή συλλογή δεδομένων που η επιλογή του είδους των δεδομένων μέσα στο Dataset επαφίεται στον χρήστη.Έτσι ένα RDD μπορεί να περιέχει String,Key-Value τιμές και άλλα αντικείμενα.

Ένα επίσης βασικά στοιχείο των RDDs είναι ότι είναι αμετάβλητα(immutable).Έτσι ο χρήστης δεν μπορεί να προσθέσει δεδομένα σε ένα RDD αλλά μπορεί να εφαρμόσει σε αυτό ενέργειες και μετατροπές και να σχηματίσει νέα RDD χρησιμοποιώντας ένα παλιό.Οι μετατροπές(transformations) που υποστηρίζονται σε ένα RDD είναι αδρομερής όπως maps,filters και joins που εφαρμόζουν την ίδια λειτουργία σε ένα μεγάλο όγκο δεδομένων

ενέργειες(actions) από την άλλη είναι λειτουργίες που επιτρέπουν την εξαγωγή πληροφοριών από ένα Εκτός από αυτές της λειτουργίες το RDD προσφέρει δυο ακόμα πολύ σημαντικές,την ικανότητα να αποθηκεύσει στην μνήμη ή στον δίσκο RDD τα οποία θα επαναχρησιμοποιηθούν (persist) και την ικανότητα να ελέγξει ο χρήστης πως αυτά τα δεδομένα θα διασπαστούν στους κόμβους ενός συμπλέγματος υπολογιστών(partitioning).

Στον πυρήνα του ένα RDD απαρτίζεται από 5 βασικές δόμες.Μια λίστα με τους διαμερισμός(partition) του RDD μέσα σε ένα σύμπλεγμα υπολογιστών ώστε κάθε RDD να ξέρει κάθε φορά που βρίσκονται τα τμήματα του. Μια συνάρτηση η όποια θα εφαρμοστεί πάνω σε κάθε εγγραφή ενός RDD.Μια λίστα αλληλεξαρτήσεων από προηγούμενα RDD μέσω της οποίας σχηματίζεται ο γράφος κληρονομικότητας ενός Προαιρετικά έναν διαμοιραστεί που αποφασίζει στα Key-Value ζευγάρια πως κατακερματίζονται τα κλειδιά και πως οι διαμερισμοί μοιράζονται τα ζευγάρια αυτά και μια λίστα από προτεινόμενες περιοχές για να τοποθετηθούν οι διαμοιρασμό



Εικόνα 5: Ο τρόπος που κατανέμονται και διαμοιράζονται τα στοιχεία ενός RDD μέσα σε ένα σύμπλεγμα υπολογιστών σε εγγραφές με μορφή ζευγαριών Key-Value

2.2.3.2 RDD vs Hadoop MapReduce Ή DAG vs MapReduce

Ο βασικός λόγος που το Apache Spark είναι η πρώτη επιλογή ανάμεσα στα συστήματα που αναλύουν μεγάλους όγκους δεδομένων είναι ότι κατάφερε να υπερκεράσει το Hadoop MapReduce.Για να το καταφέρει αυτό βασίστηκε σε δυο βασικές ιδέες.Πρώτον μετέφερε τα δεδομένα από τον δίσκο στην μνήμη.Διαχείριση δεδομένων στην μνήμη απαιτεί μεγάλο όγκο πόρων σε RAM αλλά είναι πολύ πιο γρήγορα προσπελάσιμη από τον δίσκο.Δεύτερον μπορεί να εφαρμόσει βελτιστοποιήσεις στον τρόπο που γίνονται οι μετατροπές των δεδομένων καθώς με την εισαγωγή των RDDs το Spark γνωρίζει ανά πάσα στιγμή τον γράφο της ροής των εργασιών πάνω στα δεδομένα

Για να πετύχει τις αναλύσεις στην μνήμη το Spark φορτώνει σε αυτήν όσα περισσότερα δεδομένα για όσο περισσότερο διάστημα μπορεί.Επειδή όπως είναι φυσικό ο δίσκος χωράει πολύ περισσότερα δεδομένα από την μνήμη όταν η μνήμη γεμίζει κάποια κομμάτια των δεδομένων γράφονται στον δίσκο και ανακτώνται αργότερα όταν η μνήμη αδειάσει εγγραφές και οι αναγνώσεις όμως αυτές στον δίσκο δεν είναι προτεραιότητα στο Spark και έτσι συμβαίνουν τόσο συχνά όσο είναι απαραίτητο από το σύστημα σε αντίθεση με το Hadoop MapReduce που έχει σαν προτεραιότητα την έγγραφη και ανάγνωση στον δύσκολο βασικότερο στοιχείο όμως του Apache Spark είναι η ικανότητα του να γνωρίζει όλα τα προηγούμενα βήματα στις μετατροπές των δεδομένων.Βασικά σε ένα πρώτο επίπεδο το να γνωρίζει τα βήματα είναι το μόνο που κάνει τεχνική που χρησιμοποιεί το RDD ονομάζεται οκνηρός υπολογισμός(lazy evaluation) .Αυτό σημαίνει ότι όλα τα transformation που εφαρμόζονται στον γράφο των RDD δεν υπολογίζονται παρά μόνο

ορίζονται οι γονείς και οι εκάστοτε μετατροπές που θα γίνουν στα RDD.Μόνο όταν κληθεί κάποια ενέργεια (action) πάνω σε κάποιο RDD αποφασίζεται ποια RDD χρησιμοποιήθηκαν σε προηγούμενα στάδια για να φτάσουμε στην έξοδο του τελικού RDD.Τις αποφάσεις αυτές τις αναλαμβάνει το τμήμα του Spark που ονομάζεται DAGScheduler και έχει επίσης την ευθύνη να επιλέξει σε ποια σημεία της συστάδας υπολογιστών θα τρέξουν τα transformations και τα actions για να επιτευχθεί καλύτερη τοπικότητα στα δεδομένα.Αντίθετα το Hadoop MapReduce δεν γνωρίζει τα προηγούμενα βήματα του γράφου και εκτελεί κάθε στάδιο Map και Reduce που συναντάει.Αυτό ίσως να μην αποτελεί μεγάλο πρόβλημα σε μικρο προβλήματα με λίγα και ανεξάρτητα στάδια Map και Reduce αλλά είναι μεγάλος πονοκέφαλος σε επαναληπτικούς αλγόριθμους και προγράμματα που απαιτούν επαναχρησιμοποίηση των προηγούμενων σταδίων

Ο τρόπος που γίνεται η βελτιστοποίηση αυτή φαίνεται στο παρακάτω παράδειγμα:

Έστω ότι έχουμε ένα dataset με δεδομένα από εργαζόμενους σε διάφορες εταιρείες και θέλουμε να εκτελέσουμε ένα απλό ερώτημα πάνω σε αυτά ώστε να βρούμε το σύνολο των μισθών σε μια εταιρεία.Το ερώτημα θα ήταν όπως στην εικόνα 2.2.3.2.1 όπου διαχωρίζεται σε ένα στάδιο Map,ένα Filter και ένα Reduce.Για τις ανάγκες του παραδείγματος δεν μας ενδιαφέρει τι κάνουν οι συναρτήσεις που μπαίνουν σαν ορίσματα στις μετατροπές αλλά να ξέρουμε ότι εμπεριέχονται δυο μετρητές στις συναρτήσεις του Map και του Filter οι οποίοι τυπώνονται στο τέλος μετά την ενέργεια (action) πάνω στο

RDD salaries που ονομάζεται collect.Επίσης να τονιστεί ότι το ερώτημα αυτό εκτελέστηκε σε τοπικό μηχάνημα και όχι σε συστάδα υπολογιστών

Πίνακας 1

```
JavaRDD<String> file = context.textFile("~/csv/TechCrunchcontinentalUSA.csv");
JavaPairRDD<String,Integer> paircompanywithmoney=file.mapToPair(WORDS_MAPPER);
JavaPairRDD<String,Integer> reducecompanies=paircompanywithmoney.filter(WORDS_FILTER);
JavaPairRDD<String,Integer> salaries=reducecompanies.reduceByKey(WORDS_REDUCER);;
System.out.println(salaries.collect());
System.out.println("Number of Filters:"+numberoffilters);
System.out.println("Number of Maps:"+numberofmaps);
```

Μετά την εκτέλεση του ερωτήματος το πρόγραμμα τυπώνει τα παρακάτω στοιχεία

Πίνακας 2

```
[(facebook,495700000)]
Number of Filters:1461
Number of Maps:1461
```

Άρα σε αυτήν την περίπτωση έτρεξε 1461 φορές την συνάρτηση Map πάνω στα δεδομένα και άλλες τόσες την συνάρτηση Filter για να φέρει τα δεδομένα στην μορφή που ήθελε πριν εφαρμόσει πάνω τους το Reduce και στην συνέχεια το Collect.

Ας κάνουμε τώρα κάτι απλό.Θα ξανακαλέσουμε τα ίδια transformations σε RDD με άλλα ονόματα άλλα χωρίς να καλέσουμε το collect σε αυτά.

Πίνακας 3

```
JavaRDD<String> file = context.textFile("~/csv/TechCrunchcontinentalUSA.csv");
JavaPairRDD<String,Integer> paircompanywithmoney=file.mapToPair(WORDS_MAPPER);
JavaPairRDD<String,Integer> reducecompanies=paircompanywithmoney.filter(WORDS_FILTER);
JavaPairRDD<String,Integer> salaries=reducecompanies.reduceByKey(WORDS_REDUCER);
JavaPairRDD<String,Integer> paircompanywithmoney2=file.mapToPair(WORDS_MAPPER);
JavaPairRDD<String,Integer> reducecompanies2=paircompanywithmoney.filter(WORDS_FILTER);
JavaPairRDD<String,Integer> salaries2=reducecompanies.reduceByKey(WORDS_REDUCER);
System.out.println(salaries.collect());
System.out.println("Number of Filters:"+numberoffilters);
System.out.println("Number of Maps:"+numberofmaps);
```

Το αποτέλεσμα είναι το ίδιο με ακριβώς τον ίδιο αριθμό Filters και Maps.Το Spark γνωρίζοντας τον γράφο των RDD αγνόησε πλήρως τα transformations στα RDDs που δεν συμμετέχουν στο τελικό αποτέλεσμα.Ένα σύστημα το οποίο δεν έχει την ικανότητα να δει τον γράφο της ροής εργασιών όπως το Hadoop MapReduce δεν θα απέφευγε τα επιπλέον Map,Filter και Reduce και θα τα υπολόγιζε κανονικά συν το επιπλέον κόστος από το διάβασμα-γράψιμο στον δίσκο

Αλλά ας δούμε και άλλο ένα παράδειγμα.Έστω ότι στο πρόγραμμα που είχαμε παραπάνω δεν μας ενδιέφερε το τελικό Reduce και θέλαμε απλώς να εφαρμόσουμε μια συνάρτηση Filter πάνω στα δεδομένα και δειγματοληπτικά να πάρουμε τα 5 πρώτα

στοιχεία.Αντί για την ενέργεια Collect λοιπόν θα εφαρμόσουμε μια άλλη που ονομάζεται take και παίρνει σαν όρισμα τον αριθμό των πρώτων στοιχείων από ένα RDD.

Πίνακας 4

```
JavaRDD<String> file = context.textFile("~/csv/TechCrunchcontinentalUSA.csv");
JavaPairRDD<String,Integer> paircompanywithmoney=file.mapToPair(WORDS_MAPPER);
JavaPairRDD<String,Integer> reducecompanies=paircompanywithmoney.filter(WORDS_FILTER);
System.out.println(reducecompanies.take(5));
System.out.println("Number of Filters:"+numberoffilters);
System.out.println("Number of Maps:"+numberofmaps);
```

Το Spark θα τυπώσει τα παρακάτω αποτελέσματα αγνοώντας πλήρως τις υπόλοιπες εγγραφές και υπολογίζοντας μόνο τις 5 πρώτες που συναντάει και καλύπτουν τις προδιαγραφές του Filter.Επισης βλέπουμε ότι το προηγούμενο στάδιο από αυτό που

καλέσαμε το action εκτελέστηκε πάνω σε όλα τα δεδομένα αλλά μόνο σε αυτά που ενδιέφεραν το action και κάλυπταν τις προδιαγραφές των transformations.

```
Number of Filters:17
Number of Maps:17
```

Με αυτά τα δυο πολύ απλά παραδείγματα δείξαμε πως το Spark μπορεί να πετύχει βελτιστοποιήσεις στην ταχύτητα αγνοώντας αχρείαστα στάδια και αγνοώντας αχρείαστα δεδομένα

2.2.3.3 Spark SQL

Το Spark SQL είναι ένα τμήμα του Spark που υποστηρίζει επεξεργασία δομημένων δεδομένων.Επιτρέπει στον χρήστη να τρέξει ερωτήματα σε SQL μορφή πάνω σε τεράστιους όγκους δεδομένων που μπορούν να βρίσκονται σε διάφορα αποθηκευτικά μέσα και σε διάφορες μορφές

Για να το πετύχει αυτό το Spark εισήγαγε μια νέα δομή,το DataFrame.Το DataFrame είναι μια δομή που μπορεί να κατανεμηθεί στους κόμβους μιας συστάδας υπολογιστών και περιέχει δεδομένα σε μορφή στηλών.Μπόρουμε να φανταστούμε ότι το DataFrame είναι για το Spark ότι ένα Table είναι για την κανονική SQL.Τα DataFrames μπορούν να δημιουργηθούν απο διαφόρων ειδών πηγές που περιέχουν κάποια δομή στα δεδομένα τους όπως αρχεία CSV,JSON,Tables στο Apache Hive,δεδομένα σε NoSQL βάσεις ακόμα και αλλά RDD.

Με την εισαγωγή του Spark SQL η ομάδα του Spark κατάφερε να έρθει ένα ακόμα βήμα πιο κοντά στο να δημιουργήσει ένα σύστημα που να παρέχει τις δυνατότητες για πολλών ειδών αναλύσεις σε μεγάλους όγκους δεδομένων.Πρόσθεσε δηλαδή την δυνατότητα να πραγματοποιηθούν αναζητήσεις σε δεδομένα τα οποία είναι δομημένα(structured) σε κατανεμημένα συστήματα βάσεων δεδομένων προσφέροντας ένα API οικείο στους μεγαλύτερο ποσοστό των χρηστών που ασχολούνται με διαχείριση δεδομένων.Χρησιμοποιώντας δομημένα δεδομένα υπάρχει πλέον η δυνατότητα για

εφαρμογή δηλωτικών ερωτημάτων τα οποία αυτοματοποιούν σε ακόμα μεγαλύτερο βαθμό τις βελτιστοποιήσεις

2.2.3.4 Spark Streaming

Εκτός από την δυνατότητα του να διαχειριστεί σωρευμένα (batch) δεδομένα το Apache Spark περιλαμβάνει και ένα σύστημα διαχείρισης ζωντανών δεδομένων ροής (streaming data).Το γεγονός αυτό περιλαμβάνεται στην γενικότερη προσπάθεια που κάνουν οι δημιουργοί του Spark για να παρέχουν ένα πλήρες περιβάλλον για αναλύσεις σε δεδομένα μεγάλου όγκου ανεξάρτητα από την μορφή που παρέχονται στο σύστημα

Τα δεδομένα ροής(streaming data) είναι συνεχόμενα δεδομένα που παράγονται από συστήματα όπως αισθητήρες που ελέγχουν ζωντανά την κίνηση ή σε πιο χαρακτηριστικό παράδειγμα δεδομένα που προέρχονται από τα κοινωνικά δίκτυα από την αλληλεπίδραση των χρηστών σε αυτά. Ένα πρόβλημα διαχείρισης δεδομένων ροής είναι η χρήση του trending στο twitter που εμφανίζει σε ένα χρήστη τα γεγονότα που συζητάνε πιο πολύ στο κοινωνικό δίκτυο άλλοι χρήστες μια δεδομένη στιγμή. Για να αντιμετωπιστούν αυτά τα προβλήματα το Spark προσέφερε λύση μέσω της πλατφόρμας του Spark Streaming.

Για να διαχειριστεί τα ζωντανά δεδομένα το Spark Streaming τα αντιμετωπίζει σαν μια συνεχή ροή από σωρευμένα δεδομένα. Τα δεδομένα αυτά μετατρέπονται σε RDD’s που επιτρέπουν στο Spark να τα διαχειριστεί με κατανεμημένο και ανεκτικό σε σφάλματα τρόπο χρησιμοποιώντας τις βασικές ιδέες στις οποίες στηρίζεται το RDD.Επίσης με αυτό τον τρόπο δημιουργείται η δυνατότητα επεξεργασίας των δεδομένων με υψηλού επιπέδου μοντέλα επεξεργασίας όπως το MapReduce.Τέλος είναι συμβατό με διάφορες πηγές παροχής ζωντανών δεδομένων όπως το Apache Kafka,το Apache Flume άλλα και απλά TCP sockets.

Παρά τις δυνατότητες όμως που διαθέτει το Spark Streaming,αποτελεί παράγοντα διαμάχης στην κοινότητα των συστημάτων αναλύσεων μεγάλου όγκου δεδομένων όσο αναφορά την απόφασή του να επεξεργάζεται τα ζωντανά δεδομένα σαν μια σειρά από σωρευμένα δεδομένα. Πολλοί επιστήμονες διαφωνούν με αυτή την αντιμετώπιση και για αυτό τον λόγο έχουν δημιουργηθεί άλλα συστήματα που αφορούν τα σωρευμένα δεδομένα.

2.2.3.5 Συμπέρασμα

Συνοψίζοντας το Apache Spark αποτελεί ίσως την πιο πλήρη λύση στην κοινότητα αυτή την στιγμή σχετικά με τα συστήματα ανάλυσης μεγάλου όγκου δεδομένων. Μέσω της μεταφοράς των δεδομένων στην μνήμη και της δυνατότητας αποφάσεων για τους μετασχηματισμούς των δεδομένων λόγω του γράφου ιεραρχίας πετυχαίνει μεγάλες διαφορές στην ταχύτητα κατά την ανάλυση των δεδομένων. Διαθέτει μια πολύ μεγάλη κοινότητα που συνεισφέρει διαρκώς νέα πράγματα στο Ανεξάρτητα από την διαφωνία για την επάρκεια του Spark Streaming το Spark διαθέτει ένα πλήρες πακέτο για αναλύσεις σε μεγάλους όγκους δεδομένων ,γράφους και ζωντανά δεδομένα μέσω μια εύκολης στην χρήση προγραμματιστικής διεπαφής που μπορεί να χρησιμοποιηθεί σε 4 διαφορετικές γλώσσες(Scala,Java,Python,R).Είναι ,επίσης ,συμβατό με μια σειρά από διαφορετικά συστήματα όπως διαφορετικές βάσεις δεδομένων (Cassandra,Apache Hive,HDFS) και διαφορετικούς cluster managers (Yarn,Messos) προσφέροντας ακόμα και γηγενή

επιλογή σε αυτούς. Για τους παραπάνω λόγους αποφασίστηκε και η χρησιμοποίηση του στην εφαρμογή που υλοποιήθηκε αλλά και σαν βασικό στοιχειό της πτυχιακής γενικότερα

2.2.4 Apache Flink

Το Apache Flink είναι η νέα μεγάλη παρουσία στον χώρο των αναλύσεων μεγάλου όγκου δεδομένων.Είναι ένα σύστημα που βασίζεται κυρίως στην ανάλυση ζωντανών δεδομένων αλλά παρέχει και δυνατότητα για επεξεργασία σωρευμένων δεδομένων.Γραμμένο σε Java και Scala προσφέρει ένα εύχρηστο προγραμματιστικό περιβάλλον και αποτελεί μια βελτιστοποίηση του Hadoop MapReduce,μεταφέροντας και αυτο τα δεδομένα στην μνήμη

Σε αντίθεση με το Spark Streaming το Apache Flink δεν αντιμετωπίζει τα δεδομένα σαν μια σειρά από σωρευμένα δεδομένα σε ένα χρονικό διάστημα αλλά αντιθέτως σαν μια συνεχή ροή από δεδομένα,όπως δηλαδή είναι πραγματικα.Λόγω αυτού του γεγονότος το Apache Flink θεωρείται μια πιο επαρκή λυση για τα ζωντανά δεδομένα

2.2.4.1 Δεδομένα συνεχούς ροής(Streaming Data) στο Flink

Στο Apache Flink τα δεδομενα έχουν δυο μορφές.Μια μορφή είναι τα πεπερασμένα δεδομένα που έχουν κάποιο όριο και η άλλη είναι τα συνεχή δεδομένα χωρίς κάποιο οριο.Στην ουσία όμως ενώ υπάρχει αυτός ο θεωρητικός διαχωρισμός το Flink αντιλαμβάνεται τα πεπερασμένα δεδομένα σαν συνεχή δεδομένα ροής που κάποια στιγμή σταμάτησαν

Προγραμματιστικά το Flink αποτελείται από δυο βασικές έννοιες στον κορμό του.Τις ροές(Streams) και τους μετασχηματισμούς(transformations).Η ροή(stream) είναι μια συνεχόμενη παροχή δεδομένων στο σύστημα ενώ ένας μετασχηματισμός είναι η μετατροπή μιας ροής σε μια αλλη.Πανω σε αυτές τις δυο βασικές έννοιες χτίζεται όλη η προγραμματιστική διεπαφή του Flink που προσφέρει μεγάλη αφαίρεση μέσω της Apache Flink SQL αλλά και συλλογές δεδομένων για πραγματικού χρόνου και σωρευμένα δεδομενα.Την στιγμή που ξεκινάει να εκτελείται ένα πρόγραμμα στο Flink μεταφράζεται σε ένα πλάνο μια ροής δεδομένων που φέρει ομοιότητες με το πλάνο εκτέλεσης εργασιών στο Apache Spark αλλά αφορά συνεχή ροή δεδομένων

2.2.4.2 Επίπεδα Αφαίρεσης στο Apache Flink

Το Apache Flink παρέχει 4 προγραμματιστικές διεπαφές που περικλείουν τις βασικές ιδέες του Flink (Streams και Transformations).Οι προγραμματιστικές διεπαφές προσφέρουν διάφορα επίπεδα αφαίρεσης πανω στο τρόπο που μπορούμε να διαχειριστούμε τα δεδομενα.Αυτα είναι το Dataset,το DataStream,τα Tables και η SQL.

Το **DataStream** ειναι μια ροή δεδομένων που δημιουργείται από πηγες όπως sockets,αρχεία και άλλα συστήματα παροχής ροών δεδομενων.Υποστηριζει μετατροπές όπως map,join,reduce,filter που μετατρέπουν ένα DataStream σε ένα άλλοτε DataStreams μπορούν μετα την επεξεργασία να αποθηκευτούν σε διάφορα συστήματα αποθηκευσης.Ειναι το βασικό μέσο ανάλυσης και επεξεργασίας δεδομένων συνεχούς ροής χωρις τέλος στο Flink.

Από την άλλη το **DataSet** είναι το βασικό μέσο επεξεργασίας σωρευμένων δεδομενων.Ειναι μια συλλογή δεδομένων ίδιου τύπου και υποστηρίζει και αυτό τις ίδιες σχεδόν μετατροπές με το DataStream.Στον πυρήνα τους τα DataSets και τα DataStreams

είναι παρόμοια καθώς το Flink βασίζεται στην ιδέα οτι ένα σύνολο σωρευμένων δεδομένων είναι μια ροή δεδομένων που έχει σταματήσει

Τέλος το **Table** και το **SQL** api είναι δυο προγραμματιστικές διεπαφές που επιτρέπουν την εκτέλεση SQL ερωτημάτων πάνω στις ροές δεδομένων προσφέροντας μια υψηλού επίπεδου αφαίρεση στον τρόπο που μπορεί ο χρήστης να αναζητήσει και να επεξεργαστεί δεδομένα στις ροές

2.2.4.3 Συμπέρασμα

Το Apache Flink είναι ενα ανερχόμενο σύστημα για αναλύσεις μεγάλου όγκου δεδομένων σε κατανεμημένα συστηματα.Προσπαθει να διορθώσει κάποιες από τις γενικότερες αστοχίες του Spark όντας μεταγενέστερο και κατά πολλούς προσφέρει μια καλύτερη λύση στον τομέα τον ζωντανών δεδομένων ροης.Βεβαια στον τομέα των σωρευμένων δεδομένων που είναι το δυνατό χαρτί του Apache Spark ακόμα είναι πιο πισω άλλα παρουσιάζει πολλά περιθώρια βελτιωσης.Ισως οι αποφάσεις που έχουν πάρει τα δυο συστήματα για τον τρόπο που θα μπουν στα “χωράφια¨ του άλλου να μην είναι οι ενδεικτικές

Συνοψίζοντας το Flink ειναι ένα συνεχώς αναπτυσσόμενο λογισμικό με την τελευταία του έκδοση να έχει βγει τον Φεβρουάριο του 2017 που στοχευει να αυξήσει το μερίδιο του στις εταιρείες που χρησιμοποιούν συστήματα ανάλυσης δεδομένων

3. ΕΦΑΡΜΟΓΗ

3.1 Γενικά

Στο πλαίσιο της πτυχιακής αναπτύχθηκε μια εφαρμογή που είχε σαν σκοπό την απλοποίηση του Apache Spark για τον μη έμπειρο χρήστη-προγραμματιστή παρέχοντας ένα επίπεδο αφαίρεσης στο προγραμματιστικό κομμάτι του Spark μέσω της παροχής templates.Μέσω αυτών των templates οι χρήστες μπορούν να εκτελέσουν απλά και σύνθετα ερωτήματα παρέχοντας στο σύστημα τον λιγότερο δυνατό κώδικα από πλευράς τους χρησιμοποιώντας ήδη έτοιμα templates που περιέχουν γενικευμένα ερωτήματα πάνω στο Spark.Μέσω αυτού του συστήματος γεννήθηκε η δυνατότητα για ένα ακόμα επίπεδο αφαίρεσης με την εισαγωγή μια script "γλώσσας" που απαιτεί ακόμα λιγότερο προγραμματιστικό υπόβαθρο από τον χρήστη .Η εφαρμογή αναπτύχθηκε μέσω του πλαισίου ανάπτυξης διαδικτυακών εφαρμογών Spring και χρησιμοποιήθηκε το BootStrap CSS για την οπτική απεικόνιση των ιστοσελίδων και το Thymeleaf σαν μέσο προτυποποίησης της σέλιδας.Πατάει πάνω σε ένα Standalone Spark Cluster και χρησιμοποιεί το HDFS σαν αποθηκευτικό χώρο

3.1.1 Spring Framework

Το Spring είναι ένα πλαίσιο λογισμικού που αναπτύχθηκε με σκοπό την δημιουργία συγχρόνων και απαιτητικών εφαρμογών σε Java εύκολα και γρήγορα.Εκατομμύρια χρήστες ανά τον κόσμο χρησιμοποιούν το Spring για να δημιουργήσουν εφαρμογές οποιουδήποτε είδους αν και το βασικό πλεονέκτημα του βρίσκεται στην enterprise Java.Στον τομέα αυτό έχει καταφέρει να προσπεράσει την Java 2EE και κατά βάση τα Enterprise JavaBeans.Εκτος από τα διαδικτυακά πλαίσια λογισμικού που προσφέρει το Spring είναι διάσημο και για την παροχή τεχνικών αναστροφής ελέγχου (Inversion of Control).

Λόγω της δημοτικότητας του και του γεγονότος ότι είναι ένα πλαισίου λογισμικού ανοιχτού κώδικα,από τον πυρήνα του Spring έχουν ξεπηδήσει διάφορα πλαίσια λογισμικού όπως το Spring MVC ,το Spring Boot και το Spring Security.Τα τρία αυτά πλαίσια λογισμικού χρησιμοποιούνται και στην ανάπτυξη της εφαρμογής που δημιουργήθηκε στα πλαίσια της πτυχιακής

3.1.2 BootStrap CSS

Το Bootstrap CSS είναι ένα δωρεάν πλαίσιο λογισμικού για την ανάπτυξη του μπροστά μέρους μιας εφαρμογής.Του τμήματος δηλαδή που επικοινωνεί με τον χρήστη.Παρέχει έτοιμα πρότυπα βασισμένα σε HTML και CSS όπως κουμπιά,υποδοχείς κείμενου,πίνακες και άλλα.Επίσης χρησιμοποιεί JavaScript για να προσδώσει σε κάποια από αυτά λειτουργικότητα

Αναπτύχθηκε από δυο προγραμματιστές στο Twitter και αργότερα δημοσιεύθηκε σαν ελεύθερο λογισμικό στο Github το 2011.Το 2014 ήταν το νούμερο 1 project στο GitHub.Ειναι σχεδιασμένο με πρωταρχικό στόχο να είναι συμβατό στα κινητά και είναι συμβατό με όλους τους συγχρόνους διαδικτυακούς φυλλομετρητες.

3.1.3 Thymeleaf

Είναι ένα σύστημα παροχής πρότυπων για Java συμβατό με XML,XHTML και HTML5 που βρίσκεται στην πλευρά του διακομιστή.Χρησιμοποιείται για να το οπτικό

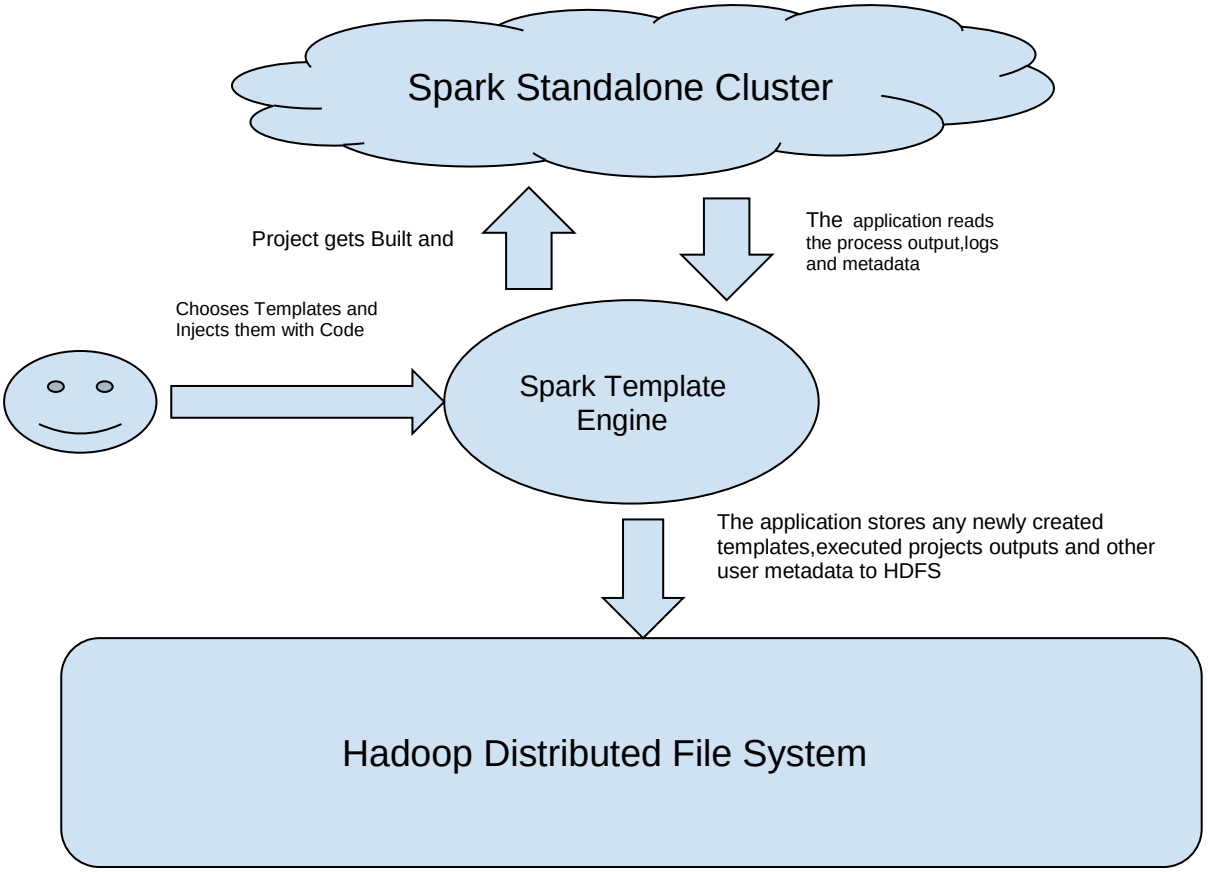
κομμάτι(View Layer) σε εφαρμογές MVC(Model,View,Controller).Αναπτύχθηκε με στόχο να αντικαταστήσει την τεχνολογία JSP(Java Server Pages) της Java EE.Είναι εύκολο στην χρήση και λιγότερο "φλύαρο" από το JSP δίνοντας την δυνατότητα να σχεδιαστούν μικρότερα,πιο εύχρηστα και πιο επεκτάσιμα templates.

Έχει δημιουργηθεί με στόχο την ενσωμάτωση του σε εφαρμογές που βασίζονται στο οικοσύστημα του Spring άλλα είναι συμβατό και με άλλες τεχνολογίες.Γενικά πάντως σε συνδυασμό με το Spring δίνει τα καλύτερα δυνατά του αποτελέσματα και για αυτό τον λόγο προτιμήθηκε

3.2 Spark Template Engine

3.2.1 Το οικοσυστημα της εφαρμογης

Το συστημα παροχης και επεξεργασια προτυπων που δημιουργηθηκε στα πλαίσια της εφαρμογής στηρίζεται και αλληλεπιδρά με το Apache Hadoop και το Apache Spark.Συγκεκριμενα χρησιμοποιεί το Hadoop Distributed File System για να αποθηκεύει τα πρότυπα,τα αποτελέσματα των ερωτημάτων που εχει δημιουργήσει ο χρήστης και αλλά μεταδεδομένα που έχουν να κάνουν με το ιστορικό των ερωτηματων.Επισης χρησιμοποιεί το Apache Spark σαν την μονάδα στην οποία εκτελούνται τα ερωτήματα πάνω στα δεδομενα.Για την διαχείριση των συστάδων υπολογιστών χρησιμοποιήθηκε ο Standalone Cluster του Apache Spark.Παρακατω φαινεται πως η εφαρμογή επικοινωνεί με όλο το οικοσύστημα



Σχημα 1: Σχηματική απεικόνιση του οικοσυστήματος της εφαρμογής

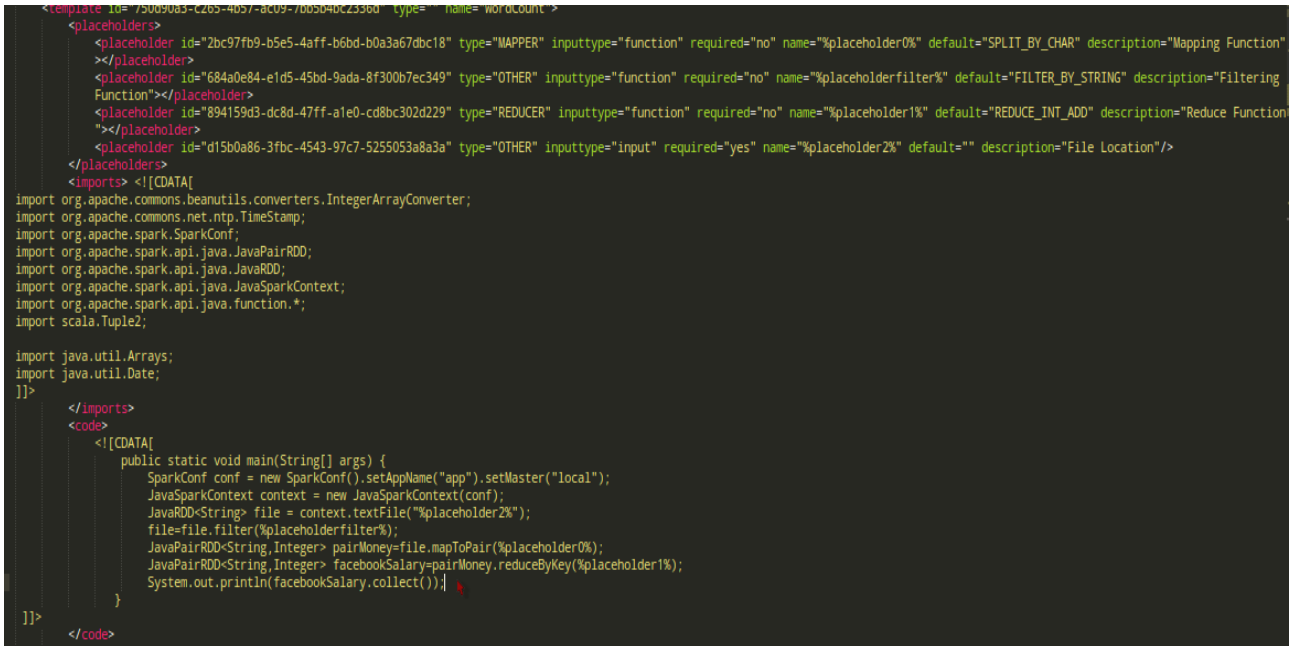
3.2.2 Ο πυρήνας της εφαρμογής

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής ειναι ενα σύστημα παροχής πρότυπων για το Apache Spark (template engine).Μέσω της εφαρμογής ο χρήστης θα μπορει να χρησιμοποιήσει έτοιμα πρότυπα ροών εργασιών στο Apache Spark και να

ανεβάσει νέα δικά του ώστε να τα χρησιμοποιήσει έτοιμα αυτός ή κάποιος άλλος χρηστης.Αυτα τα πρότυπα ειναι προγράμματα σε Spark που έχουν υποδοχείς κωδικα στους οποίους εισάγει σε συγκεκριμένα σημεία δίκια του κομμάτια κώδικα ή άλλα επίσης προτυποποιημένα κομμάτια κώδικα

Με αυτό τον τρόπο ενας χρήστης γνωρίζοντας τι κάνει ένα πρότυπο και τι στοιχεία περιμένει από τον χρήστη,μπορεί να τρέξει εξειδικευμένα ερωτήματα πάνω στο Spark χωρίς να γνωρίζει πως λειτουργεί το Spark ή έστω γνωρίζοντας πολύ λίγα πράγματα

Μόλις περάσει από αυτό το στάδιο ο χρήστης και μετατρέψει το πρότυπο που επέλεξε σε μια πλήρη Java κλάση το πρότυπο αποθηκεύεται στο τρέχοντα εκτελέσιμο φάκελο του χρήστη όπου μπορεί να συνθέσει ένα πρόγραμμα που αποτελείται από πολλά προτυπα.Επισης ο χρήστης μπορεί να ανεβάσει στον εκτελέσιμο φάκελο του κάποιο δικό



Εικόνα 7 : Ένα πρότυπο προγράμματος σε Spark όπως αποθηκεύεται στο σύστημα

του αρχείο Java ή να επεξεργαστεί περαιτέρω προσθέτοντας επιπλέον κωδικά σε κάποιο από τα αρχεία που εχει ετοιμάσει μέχρι εκείνη την στιγμή μέσω των προτύπων

Το σύστημα κρατάει ένα ιστορικό με όλες τις επιλογές που έχει κάνει ο χρήστης πάνω σε ένα πρότυπο σε διαφορετικές ημερομηνιες.Με αυτόν τρόπο μπορεί να επαναφέρει ένα πρότυπο που είχε επεξεργαστεί παλιότερα και να το επεξεργαστεί ξανά αλλάζοντας τα κομμάτια του κώδικα που είχε χρησιμοποιησει.Ιστορικο υπάρχει και σε επίπεδο ολόκληρο προγράμματος όπου ένας χρήστης μπορεί να επαναφέρει ένα παλιό πρόγραμμα που είχε στείλει για εκτέλεσή στο Spark και να επεξεργαστεί κάποια κλάση του,να την αφαιρέσει ή να την αντικαταστήσει με κάποια άλλη

Τέλος ένας χρήστης μπορεί να μοιραστεί με έναν άλλο χρήστη εκτός από τα πρότυπο που έχει φτιάξει και ενα ολόκληρο προγραμμα που έχει εκτελέσει παλιότερα στο Spark.Με αυτό τον τρόπο μπορούν χρήστες με κοινά ενδιαφέροντα ή συνεργάτες ή κάποιος χρήστης που έχει προσλάβει κάποιον προγραμματιστή να αναπαράγουν ερωτήματα στο Spark που εχει φτιάξει ένας άλλος χρήστης και να αλλάξουν μικρά σημεία στον κωδικα

όπως παραδείγματος χάριν το αποθηκευτικό μέσο που βρίσκονται τα δεδομένα ή που θα αποθηκευτούν τα αποτελέσματα.

```
<function type="MAPPER" id="d73969ef-18af-493c-8dbd-680b6656c0d7" name="EXTRACT_TO_OBJ" >
    <placeholders>
        <placeholder inputtype="input" type="OBJECT_TYPE" id="af2aac86-6bd3-4bd8-9014-c9f7d6405ac5" name="%placeholder0%" required="yes" default='' description="Type of Object to convert
        String"></placeholder>
        <placeholder inputtype="input" type="CONVERSION" id="407a31d9-fdbd-4ef5-9541-4bd3c0ba3b8c" name="%placeholder1%" required="yes" default='' description="Code to convert String
        Array to Object"></placeholder>
    </placeholders>
    <code>
        <![CDATA[ private static final Function<String, %placeholder0%> EXTRACT_TO_OBJ =
                new Function<String, %placeholder0%>() {
                    public %placeholder0% call(String s) throws Exception {
                       String [] strarray=s.split(",");
                        %placeholder1%
                    }
                };]]>
    </code>
    <externallibraries type="" location=""/>
    <description><![CDATA[A Mapper function reading a String and converting it to some other value]]></description>
</function>
```

Εικόνα 8: Μια προτυποποιημενη συνάρτηση που μπορ ει να εισαχθεί σε κάποιον υποδοχέα(Placeholder) ενός πρότυπου

3.2.3 Αποτελέσματα ενός ερωτήματος

Μέσω της εφαρμογής ενας χρήστης μπορει να δει τα αποτελέσματα της εφαρμογής που ετρεξε.Τα αποτελέσματα που μπορει να δει ενας χρήστης ειναι η έξοδος ενός προγράμματος που έτρεξε αρα για να δει αποτελέσματα πρέπει ο κώδικας του να εκτυπώνει δεδομένα στην ροη εξόδου του προγραμματος.Επισης στον χρήστη παρουσιάζεται και η ροη σφάλματος ενός προγράμματος στην όποια το Spark εκτυπώνει τα σταδια που περνάει το προγραμμα του χρήστη άλλα και οτι σφάλμα έχει προκυψει.Με αυτό τον τρόπο ο χρήστης μπορει να ελέγξει τυχόν λάθη που έχει κανει στο ερώτημα που ετρεξε.Εκτος από αυτά αποθηκεύεται και το Spark logfile που ειναι το ιδιο αρχείο που χρησιμοποιεί το Spark για την οπτική απεικόνιση των ερωτημάτων

Τα αποτελέσματα αποθηκεύονται στο HDFS.Μολις ο χρήστης επιλέξει να κατεβάσει τα αποτελέσματα ενός ερωτήματος που έτρεξε το προ γραμμα ανατρέχει στον φάκελο του χρήστη στο HDFS,βρίσκει τα αποτελέσματα και τα περνάει μέσα σε ενα αρχείο zip.Στην συνέχεια τα δεδομένα αποθηκεύονται στον τοπικό φάκελο του χρήστη

3.2.4 Spark Template Script Language

Στην προσπάθεια να κανουμε το Spark ακόμα πιο απλό για τον μεσο χρηστη και να αφαιρέσουμε περαιτέρω την πολυπλοκότητα από ένα Java πρόγραμμα σε Spark, δημιουργήθηκε στα πλαίσια της πτυχιακής μια απλή Script γλώσσα στην όποια ο χρήστης χρησιμοποιεί Java μόνο άμα το θελήσει

Μέσω αυτής της γλωσσάς,γράφοντας ενα απλό Script ο χρήστης μπορ ει εύκολα να κατασκευάσει ενα Spark πρότυπο χωρις να δει την μορφή των προτυπων αλλα ξέροντας μονο τι κάνουν και τι απαιτούν από τον χρήστη να εισαγει.Επειδη τα πρότυπα

υποστηρίζουν και προκαθορισμένες τιμες,τα script του χρήστη μπορει να χρειαστούν τιμές μονο στα σημεία που δεν έχουν προκαθορισμένες τιμές ή αν το επιλεξει να αντικαταστήσει (override) αυτές τις προκαθορισμένες τιμές

```
void Goal() :{}
{
  Invoke()
  WhereStatement()
  <EOF>
}

void Invoke() :{}
{
  "INVOKE" Identifier()
}

void Identifier() :{}
{
  <IDENTIFIER>
}

void Code() :{}
{
  <CODE>
}

void WhereStatement() :  {}
{
  "WHERE"  QueryStatements()
}

void QueryStatements() :  {}
{
  PlaceHoldersStatement() "=" PlaceHoldersAssignment() (AdditionalPlaceholdersStatements())*
}

void PlaceHoldersStatement() :  {}
{
  Identifier()"."Identifier()
}

void PlaceHoldersAssignment() :  {}
{

  "{" Code() "}"
}

void AdditionalPlaceholdersStatements() :  {}
{
  "AND" QueryStatements()
}
```

Εικόνα 9: Η γραμματική της γλωσσάς για τα Spark Templates

Με αυτό τον τρόπο γράφοντας ενα απλό script οπως παραδείγματος χαριν “INVOKE MapReduceExample WHERE MapReduceExample.FILE_LOCATION={/*” hdfs://hadoopmaster:9000/file” */}" θα κληθει το πρότυπο Spark με αυτό το όνομα και θα εισαχθεί στον υποδοχέα με ταυτότητα FILE_LOCATION η τιμη που επιλέξαμε και στους υπόλοιπους υποδοχείς θα τοποθετηθούν οι προκαθορισμένες τιμες.Με λιγα λόγια μέσω αυτό του απλού script δημιουργήσαμε μια ολοκληρωμένη Java κλάση που αν δεν θέλει ο χρήστης δεν χρειάζεται να την δει.

ΣΥΜΠΕΡΆΣΜΑΤΑ

Τα τελευταία χρόνια που το ίντερνετ αναπτύσσεται με ραγδαίους ρυθμούς οι ανάγκες μας για μεγαλύτερη δυνατότητα αποθήκευσης δεδομένων αλλά και για πιο γρήγορη και ευέλικτη αναζήτηση στα δεδομένα εχει οδηγήσει τους επιστήμονες του τομέα της Πληροφορικής στην δημιουργία συγχρόνων συστημάτων ανάλυσης μεγάλου όγκου δεδομενων.Συμφωνα με την IBM κάθε μερα σε ολο τον κόσμο παράγονται 2.5 πεντακισεκκατομυρια bytes από δεδομενα.Ενω ο ρυθμός που παράγονται τα δεδομένα αυξάνεται και αυτός συνεχως.Εν ολιγοις σύμφωνα με έρευνες το 90% των δεδομένων που υπάρχουν αυτή την στιγμή στον κόσμο εχει δημιουργηθεί τα τελευταία 2-3 χρόνια

Λόγω των παραπάνω έχουν αναπτυχθεί μια σειρά από πολλά συστήματα που εφαρμόζουν διάφορες ροες εργασιων.Απο απλές που επαφίενται στον χρήστη οπως αυτές του Apache Taverna μέχρι πιο συνθέτες οπός το MapReduce.Καθε τεχνική ροής εργασιών σχεδιάζεται από τους επιστήμονες για να αντιμετωπίσει τα εκάστοτε προβλήματα και να βελτιώσει τις λύσεις που των προκατόχων δημιουργώντας ανάγκες για περισσότερες βελτιστοποιησεις.Ετσι για αυτολυς τους λόγους ο τομέας των αναλύσεων μεγάλου όγκου δεδομένων ειναι ραγδαία αναπτυσσόμενος με χιλιάδες επιστήμονες να ασχολούνται σε αυτόν

Αυτή την στιγμή κανενα από τα συστήματα ροων εργασιών δεν αντιμετωπίζει όλα τα προβλήματα του κόσμου των Big Data,άλλα αυτό ίσως και να ει ναι αδυνατο.Υπαρχει ομως μια μεγάλη κοινότητα και ενα σύνολο από συστήματα μέσα στο οικοσύστημα της Apache που καλύπτουν όλες τις ανάγκες συνολικα.Ετσι συνδυάζοντας τις διάφορες επιλογές από συστήματα που εχει ένας προγραμματιστής μπορει να αναπτύξει μια εφαρμογή που να καλύπτει τα εκάστοτε προβλήματα που εχει κλιθεί να αντιμετωπισει.Ειτε αυτά ειναι τεράστιοι όγκοι δεδομένων για ανάλυση σε ενα αποθηκευτικό μεσο ειτε ζωντανά δεδομένα οπός τα likes στο Facebook ή τα trends στο Twitter η λύση ειναι εκει εξω.

Μέσα σε αυτό το πλαίσιο χρησιμοποιήθηκαν σε αυτή την εργασία δυο από τα διασημότερα συστήματα στον τομέα των Big Data Analytics,το Apache Spark και το Apache Hadoop.Συνδυαζοντας τα δυο δυνατά στοιχεία του καθενος.Στο Πράτο το κατανεμημένο σύστημα αρχείων του και στο δεύτερο την εξαιρετική του ταχύτητα στις αναλύσεις δεδομενων.Πανω σε αυτά τα δυο συστήματα χτίσαμε μια εφαρμογή που απλουστεύει για τον χρήστη την δημιουργία αναλύσεων πανό στα δεδομένα και των διευκολύνει στην επανεκτέλεση του σε μια συστάδα υπολογιστων.Με αυτό τον τρόπο διευκολύνουμε τον οποιονδήποτε σχετικό ή άσχετο με τον προγραμματισμό να αναζητήσει τα δεδομένα του και να κανει μελέτες πανό σε αυτά παρέχοντας του ενα πλήρες οικοσύστημα

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Template	Προτυπο
Function	Συναρτηση
Key-Value pair	Ζευγαρι Κλειδιου-Τιμης
API	Προγραμματιστικη διεπαφη
Software Framework	Πλαισιο Λογισμικου

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Dataset
TCP/IP	Transmission Control Protocol/ Internet Protocol
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
YARN	Yet Another Resource Negotiator

ΑΝΑΦΟΡΕΣ

[1] S. Ghemawat, H. Gobioff, and S.T. Leung, The Google File System, *Google*.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica,Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing.University of California, Berkeley

[3] J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*,Google, Inc..

[4] M. Armbrust , R. S. Xin, C. Lian, Y. Huai , D. Liu , J. K. Bradley , X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi , M. Zaharia,Spark SQL: Relational Data Processing in Spark,AMPLab, UC Berkeley.

[5] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, I. Stoica, Discretized Streams: Fault-Tolerant Streaming Computation at Scale,University of California, Berkeley.

[6] P. Carbone A. Katsifodimos S. Ewen ,V. Markl S Haridi K. Tzoumas.,*Apache Flink™: Stream and Batch Processing in a Single Engine,TU Berlin & DFKI*.

[7] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica,Spark: Cluster Computing with Working Sets,University of California, Berkeley