ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ
Τμήμα Πληροφορικής

Πτυχιακή Εργασία

Ανίχνευση εισβολών με αρχιτεκτονική Lambda.

Ευφροσύνη Καλτιριμίδου

Επιβλέπων:
Αθηνά Βακάλη
Καθηγήτρια

Θεσσαλονίκη 2016

Περιεχόμενα

Κεφάλαιο 1

Αρχιτεκτονική Lambda

Ένα από τα πιο σύνθετα προβλήματα που χρειάζεται να αντιμετωπίζουν τα Big Data συστήματα είναι η εύρεση λύσης/απάντησης σε πραγματικό χρόνο.Κατά την αρχική τους σχεδίαση, δεν υπήρχε καμία πρόβλεψη για την αντιμετώπιση αυτού του είδους των προβλημάτων και φαινόταν έξω από την σφαίρα των δυνατοτήτων τους. Με την υποστήριξη και την ανάπτυξη που έλαβαν τα χρόνια μετά την εμφάνιση τους, βοήθησαν να αναπτυχθούν και να εδραιωθούν σε όλο και περισσότερους τομείς της πληροφορικής. Ο τομέας των real-time analytics αποδείχθηκε ένα αρκετά μεγάλο πρόβλημα, λόγω του όγκου της πληροφορίας που αποθηκεύεται σε ένα τέτοιο σύστημα. Η λύση ακόμα δεν έχει δοθεί από ένα μόνο συγκεκριμένο σύστημα αλλά έχει περιγραφτεί μια αρχιτεκτονική συστημάτων που παράγει το σωστό αποτέλεσμα, με μερικούς περιορισμούς.Η αρχιτεκτονική αυτή ονομάζεται Lambda και οι βασικές της αρχές θα αναλυθούν σε αυτό το κεφάλαιο.

Κεφάλαιο 2

Data Model

Ο πυρήνας της αρχιτεκτονικής Lambda είναι τα δεδομένα της και συγκεκριμένα το master dataset που αποθηκεύεται στο batch layer. Το master dataset αποτελεί την πηγή της "αλήθειας" όπως έχει καταγραφεί και αποθηκευτεί. Ακόμα και αν όλα τα views στο speed και serving layer χανόντουσαν, το σύστημα μπορεί να τα ξαναδημιουργήσει μέσα σε μερικές ώρες από το master dataset. Είναι προφανές οτι το master dataset είναι το μοναδικό κομμάτι της αρχιτεκτονικής Lambda το οποίο πρέπει να προφυλαχτεί από data corruption, disk failure, και άλλα σφάλματα που οδηγούν στη μη αναστρέψιμη απώλεια δεδομένων.

1. Ορολογία

Είναι σημαντικό να καθορίσουμε την ορολογία που χρησιμοποιείται σε κάθε θεματικό επίπεδο ώστε να γίνει σαφές το περιεχόμενο του.

Information Είναι οι πληροφορίες που έχουν σχέση με την εφαρμογή που εκτελείται στο σύστημα.

Data Οι πληροφορίες που δεν μπορούν να εξαχθούν/υπολογιστούν από άλλα δεδομένα που κατέχονται. Αποτελούν το master dataset. Θα αναφέρονται ως δεδομένα.

Views Πληροφορίες που έχουν εξαχθεί/υπολογιστεί από τα Data.

Queries Ερωτήσεις που υποβάλλονται στο σύστημα και απαντώνται από τα Views.

2. Ιδιότητες των δεδομένων

Υπάρχουν τρεις ιδιότητες που προτείνεται να έχουν τα δεδομένα του master dataset και αναφέρονται ως rawness, immutability, perpetuity ή όπως αποδίδονται σε μία έκφραση "eternal trueness of data". Η κάθε μία ιδιότητα θα αναλυθεί και θα εξηγηθεί για να γίνει σαφές ο λόγος που προτείνεται.

2.1 Rawness

Οι πληροφορίες που αποθηκεύονται στο master dataset και αποτελούν τα δεδομένα πρέπει να είναι στην πιο ακατέργαστη μορφή τους, κατά προτίμηση στη μορφή σχεδόν που τα λαμβάνουμε χωρίς καμία τροποποίηση ή διαγραφή. Το σύστημα είναι ένα Query Answering σύστημα με κύρια πηγή πληροφοριών το master dataset. Όσο πιο ακατέργαστα είναι τα δεδομένα που κατέχει τόσο περισσότερες ερωτήσεις μπορεί να απαντήσει. Αυτό οφείλετε στο γεγονός ότι οι μέθοδοι κατεργασίας των δεδομένων για αποθήκευση εφαρμόζουν αλγόριθμους για serialization πάνω στα δεδομένα κατά την οποία μπορεί να διαγραφούν δεδομένα που δεν έχουν προβλεφθεί κατά την σχεδίαση των αλγορίθμων ή να μετατραπούν. Επειδή δεν είναι γνωστά από την αρχή όλα τα ερωτήματα που θα υποβληθούν στο σύστημα είναι προτιμότερο να κατέχει όλη την πληροφορία

στην αρχική της μορφή ώστε να μπορεί να απαντήσει όποιο ερώτημα προκύψει.

2.2 Immutability

Η επόμενη ιδιότητα των δεδομένων είναι η αμεταβλητότητα τους, η μόνη ενέργεια που επιτρέπεται στο master dataset είναι η Add και δεν υπάρχει Update. Η Delete είναι μια ειδική περίπτωση ενέργειας και επιτρέπεται κάτω από συγκεκριμένες περιπτώσεις. Α αμεταβλητότητα των δεδομένων εμφανίζει δύο πλεονεκτήματα.

Ασφάλεια από Ανθρώπινα λάθη Ένας από τους λόγους απώλειας δεδομένων σε πολλές περιπτώσεις οφείλετε σε λάθος πράξεις του ανθρώπινου παράγοντα. Με την αμεταβλητότητα, κανένας χρήστης ή διαχειριστής δεν μπορεί να αλλοιώσει τα ήδη υπάρχοντα δεδομένα, αλλά μόνο να προσθέσει καινούργια. Αν αυτά τα καινούργια είναι λανθασμένα μπορούν εύκολα να διαγραφούν και το σύστημα να συνεχίσει να δουλεύει δημιουργώντας εκ νέου τα Views του με τα σωστά δεδομένα.

Simplicity Η αποθήκευση δεδομένων που μπορούν να μεταβληθούν στο μέλλον απαιτεί την ύπαρξη τεχνικών εύρεσής συγκεκριμένων δεδομένων. Απαιτείται δηλαδή κάποιο indexing. Αντίθετα η εγγραφή αμετάβλητων δεδομένων προϋποθέτει μόνο έναν μηχανισμό για Append στο master dataset το οποίο είναι πολύ πιο απλό στο σχεδιασμό και υλοποίηση του συστήματος.

Ένα από τα μειονεκτήματα του immutable dataset είναι οι μεγαλύτερες απαιτήσεις σε αποθηκευτικό χώρο καθώς δεν γίνεται κάποιος χωρισμός των δεδομένων σε πίνακες, πολλά στοιχεία μπορεί να αποθηκεύονται ξανά και ξανά με κάθε εγγραφή.

2.3 Perpetuity

Η συγκεκριμένη ιδιότητα επιβεβαιώνει αυτό που θεωρούνταν ως τώρα αυτονόητο, τα δεδομένα είναι και θα είναι για πάντα αληθινά. Data is eternally true. Το immutability δεν μπορεί να υποστηριχθεί χωρίς να ικανοποιείται το perpetuity. Η ιδιότητα αυτή καλύπτεται από την προσθήκη timestamp στις πληροφορίες που αποθηκεύονται στο master dataset, από την στιγμή που υπάρχει χρόνος αναφοράς για τα δεδομένα είναι εύκολο να ελεγχθεί και αν όντως ισχύουν.

2.4 Delete

Αναφέρθηκε ήδη μια περίπτωση όπου η διαγραφή των immutable δεδομένων είναι επιτρεπτή και μάλιστα απαραίτητη για την σωστή λειτουργία του συστήματος. Θα αναφερθούν ακόμα δύο περιπτώσεις στις οποίες το Delete μπορεί να εφαρμοστεί στο master dataset. Και στις δύο αυτές περιπτώσεις, η διαγραφή δεν έχει σχέση με την ορθότητα των δεδομένων αλλά με την αξία των δεδομένων.

Garbage collection Ο όρος είναι γνωστός και αναφέρεται στην συλλογή και εκκαθάριση άχρηστων δεδομένων. Παρομοίως μπορεί να εφαρμοστεί και στο master dataset για να μειώσει τον αποθηκευτικό χώρο που απαιτεί με το να διαγραφούν τα δεδομένα, ή τα κομμάτια των δεδομένων που έχουν την χαμηλότερη πληροφοριακή αξία.

Κανονισμοί Σε αρκετές περιπτώσεις ευαίσθητων και ιδιωτικών δεδομένων, απαιτείται η διαγραφή τους είτε μετά είτε κατά την συλλογή τους.

Σε όλες τις περιπτώσεις διαγραφής θα πρέπει να ληφθούν μέτρα για να ολοκληρωθεί η διαδικασία σωστά και χωρίς επιπλέον απώλειες δεδομένων. Συνιστάται η αντιγραφή ολόκληρου του master dataset, η εφαρμογή των αλγορίθμων για την διαγραφή των κατάλληλων δεδομένων και ο έλεγχος έχουν απομείνει τα σωστά δεδομένα χωρίς απώλειες. Το αντίγραφο μπορεί να διαγραφεί με ασφάλεια αφού δημιουργηθούν τα καινούργια Views και δοκιμαστούν με επιτυχία.

3. Αναπαράσταση των δεδομένων

Μετά τον ορισμό των ιδιοτήτων των δεδομένων μας, θα αναλυθεί ο προτεινόμενος τρόπος αναπαράστασης τους μέσα στο master dataset. Γενικά υπάρχουν πολλοί τρόποι αναπαράστασης των δεδομένων εκτός των σχεσιακών βάσεων δεδομένων όπως, XML, JSON και άλλα.
Για την αρχιτεκτονική Lambda προτείνετε η αποθήκευση και αναπαράσταση των δεδομένων σε Fact-based Model. Σε αυτό το μοντέλο,η πληροφορία χωρίζετε σε κομμάτια όπου το καθένα αποτελεί ένα Fact. Το κάθε fact πρέπει ικανοποιεί δύο προϋποθέσεις για να μπορεί να εισαχθεί στο master dataset.

Atomic Η πρώτη προϋπόθεση αναφέρεται στο γεγονός ότι το fact δεν μπορεί να διαιρεθεί σε μικρότερα facts. Είναι το ελάχιστο κομμάτι πληροφορίας που μπορούμε να συλλέξουμε.

Timestamped Κάθε fact συνοδεύεται από μια χρονική σήμανση που δηλώνει την χρονική στιγμή που συλλέχτηκε/δημιουργήθηκε.

Unique Κάθε fact πρέπει να είναι μοναδικό να εύκολα διαχειρίσιμο από τα υπόλοιπα. Για πολλούς λόγους μπορούν να προκύψουν διπλότυπες εγγραφές κατά την συλλογή των πληροφοριών. Θα πρέπει να υπάρχει ένα πεδίο που να είναι μοναδικό για το κάθε fact. Το timestamp δεν μπορεί να χρησιμοποιηθεί διότι σε περιπτώσεις που τις πληροφορίες τις συλλέγουν περισσότεροι από έναν agent, μπορεί να προκύψουν δύο facts με ίδια στοιχεία. Επίσης ακόμα και στην περίπτωση που την συλλογή την κάνει μόνο ένας agent, δικτυακά προβλήματα μπορεί να προκαλέσουν την αποστολή του ίδιου fact για εγγραφή παραπάνω από μία φορά. Προτείνετε η χρήση ενός ειδικού field ID που θα είναι μοναδικό για κάθε fact.

Ένα πολύ καλό παράδειγμα fact-based model αποτελούν τα logs που δημιουργούνται από servers και monitor services. Σε αυτά καταγράφετε η κάθε ενέργεια που εκτελείται στην εφαρμογή με την ακριβή χρονική στιγμή που εκτελέστηκε. Θα αναλυθούν τα πλεονεκτήματα που προσφέρει το fact-based model για να γίνει πιο σαφής η επιλογή του ως μοντέλο αναπαράστασης του master dataset.

- Μπορούν να ανακτηθούν δεδομένα με βάση την χρονική στιγμή. Δηλαδή οι ερωτήσεις που θα υποβληθούν μπορούν να προσδιορίζουν συγκεκριμένη χρονική στιγμή ή περίοδο για την οποία επιθυμούν την απάντηση.
- Υπάρχει προστασία από ανθρώπινα λάθη που προσθέτουν λανθασμένες πληροφορίες στο dataset. Τα συγκεκριμένα facts μπορούν εύκολα να βρεθούν και να διαγραφούν από το dataset.
- Δεν υπάρχει καθορισμένο σχήμα που πρέπει να ακολουθούν τα δεδομένα, συνεπώς δεν υπάρχει ανησυχία για πεδία με τιμή NULL που μπορεί να υπάρξουν και πως θα γίνει ο χειρισμός τους.

4. Σύνοψη

Τα δεδομένα που αποθηκεύονται στο master dataset αποτελούν την βάση της αρχιτεκτονικής Lambda. Οι αποφάσεις που θα καθορίσουν τον τρόπο αναπαράστασης τους και των ιδιοτήτων που θα έχουν, θα καθορίσουν σε μεγάλο βαθμό τις δυνατότητες του συστήματος. Ανάλογα με τα δεδομένα που θα υπάρχουν στο σύστημα και το πως είναι, θα δημιουργηθούν οι αλγόριθμοι ανάλυσης.

Το σημαντικό σε κάθε επιλογή που γίνεται είναι να λαμβάνεται πάντα υπόψη ότι τα δεδομένα όπως λαμβάνονται μπορεί να αλλάξουν με το πέρασμα του χρόνου και οι τεχνικές που θα αναπτυχθούν, καθώς και το μοντέλο που τα αναπαριστά πρέπει να είναι ευέλικτα ώστε να μπορούν να προσαρμοστούν.

Κεφάλαιο 3

Master Dataset Store

Αυτό το κεφάλαιο επικεντρώνετε στο σύστημα αποθήκευσης του master dataset που θα χρησιμοποιεί η αρχιτεκτονική Lambda. Η επιλογή ενός τέτοιου τρόπου θα επηρεάσει σε μεγάλο βαθμό την απόδοση όλου του συστήματος. Τα δεδομένα που αναμένετε να αποθηκευτούν απαιτούν πολύ περισσότερο αποθηκευτικό χώρο από αυτό που μπορεί να προσφέρει αποδοτικό ένας μόνο server. Συνεπώς απαιτείται η χρήση ενός κατανεμημένου συστήματος αποθήκευσης. Το σύστημα αποθήκευσης θα πρέπει επίσης να είναι σε θέση να χειριστεί τις συνεχείς προσθήκες στο master dataset χωρίς να απαιτεί πολλούς χειρισμούς από τους διαχειριστές.

1. System Requirements

Υπάρχουν τρεις απαιτήσεις που χρειάζεται να καλύπτει το σύστημα αποθήκευσης. Αυτές αναφέρονται στις πράξεις που θα πρέπει να γίνονται πάνω στα δεδομένα. Τα δεδομένα μας θα είναι immutable και eternal. Το οποίο σημαίνει ότι δεν θα υπάρχει κανένα Update πάνω στα υπάρχοντα δεδομένα. Το σύστημα θα πρέπει να υποστηρίζετε μόνο την Add και τη Delete σε πολύ συγκεκριμένες περιπτώσεις, απαιτήται επίσης να έχει πολύ καλή απόδοση στην εγγραφή καινούργιων δεδομένων σε μια μεγάλη συλλογή δεδομένων. Η πράξη της διαγραφής θεωρείται ότι θα συμβαίνει τόσο σπάνια που η απαίτηση να είναι optimal δεν είναι μεγάλης προτεραιότητας.
Η δεύτερη πράξη που θα εκτελεί το σύστημα είναι η ανάγνωση των δεδομένων από τους αλγορίθμους που θα κατασκευάζουν τα Views στο Batch layer. Το σύστημα θα πρέπει να έχει πολύ καλή απόδοση στην ανάγνωση πολλών δεδομένων, ολόκληρου του dataset στην χειρότερη περίπτωση. Τέτοια συστήματα περιγράφονται από το ακρωνύμιο WORM (Write Once Read Many) και είναι συστήματα αποθήκευσης δεδομένων που αναπτύσσονται για την εξυπηρέτηση Big Data αλγόριθμων.
Οι απαιτήσεις φαίνονται πιο αναλυτικά στον παρακάτω πίνακα.

Πίνακας 3.1: Απαιτήσεις Συστήματος

Πράξη	Απαιτήσεις	Σχόλια
Εγγραφή	Αποδοτική προσθήκη δεδομένων	Η μόνη πράξη εγγραφής εκτελείται για να προστεθούν δεδομένα στο master dataset. Δεν υπάρχει διαδικασία για Update.
	Scalable storage	Το master dataset, ανάλογα την περίπτωση και το θέμα της εφαρμογής, αναμένεται να χρησιμοποιεί από πολλές δεκάδες Gigabytes έως Pettabytes για μεγάλες εφαρμογές. Θα πρέπει το σύστημα να μπορεί αποδοτικά να προσθέτει συνεχώς δεδομένα στο master dataset.
Ανάγνωση	Υποστήριξη παράλληλης ανάγνωσης	Η δημιουργία των Views στο Batch Layer θα γίνετε με την χρήση ολόκληρου του master dataset. Συνεπώς χρειάζεται υποστήριξη της παράλληλης επεξεργασίας για να μπορέσει να διαχειριστεί τον όγκο των δεδομένων αποδοτικά.
Εγγραφή και Ανάγνωση	Ευελιξία στην διαχείριση του κόστους αποθήκευσης/επεξεργασίας	Η αποθήκευση τέτοιου μεγέθους δεδομένα είναι πολύ κοστοβόρα. Μία λύση αποτελεί η συμπίεση των δεδομένων για εξοικωνόμηση χώρου και η αποσυμπίεση τους για την επεξεργασία. Θα πρέπει να προσφέρετε η δυνατότητα συμπίεσης των δεδομένων και ανάκτησης τους όταν επιθυμεί ο χρήστης.
	Αμεταβλητότητα δεδομένων	Θα πρέπει να ληφθούν όλοι οι δυνατοί έλεγχοι και συνθήκες για να τηρηθεί η αμεταβλητότητα του master dataset.

Η τρίτη απαίτηση είναι εξίσου, αν όχι περισσότερο, σημαντική με τις προηγούμενες καθώς διασφαλίζει την ασφάλεια των δεδομένων και λαμβάνει όλες τις δυνατές ενέργειες για να μην υπάρχει καμία απώλεια. Βεβαίως κανένα σύστημα δεν μπορεί να εγγυηθεί ότι καλύπτει κάθε περίπτωση απώλειας δεδομένων με απόλυτη ακρίβεια. Η σωστή επιλογή ενός αρκετά αποδοτικού fault-tolerant συστήματος με συνεχείς ελέγχους και ένα μηχανισμό backup μπορεί να καλύψει το μεγαλύτερο κομμάτι των περιπτώσεων και να κρατήσει τα δεδομένα ασφαλή.

2. Επιλογή συστήματος

Σύμφωνα με αυτές τις απαιτήσεις θα γίνει η επιλογή του κατάλληλου κατανεμημένου συστήματος αποθήκευσης.Επειδή υπάρχουν πολλές κατηγορίες συστημάτων για αποθήκευση δεδομένων θα επιλεχθεί αεχικά ο τύπος του συστήματος και στη συνέχεια θα γίνει πιο εύκολη και η επιλογή ενός συγκεκριμένου.

2.1 Σχεσιακές βάσεις δεδομένων

Οι σχεσιακές βάσεις δεδομένων δεν ταιριάζουν ως σύστημα αποθήκευσης για τέτοιου είδους δεδομένα όπως έχει ήδη αναλυθεί. Χρειάζονται μεγάλο σχεδιασμό και χωρισμό των δεδομένων και των πεδίων τους κάτι το οποίο δεν είναι επιθυμητό. Επίσης οι βάσεις δεδομένων παρέχουν πολλούς μηχανισμούς για τροποποίηση των δεδομένων, το οποίο είναι άχρηστο και επιβλαβές στη συγκεκριμένη περίπτωση.

2.2 Key-value βάσεις δεδομένων

Οι συγκεκριμένες βάσεις δεδομένων αποθηκεύουν τις πληροφορίες ως με έναν διαχωρισμό σε key και value, όπου key είναι ένα αναγνωριστικό για την μονάδα των πληροφοριών που αποθηκεύεται κάθε φορά, και value οι πληροφορίες. Ένα τέτοιο σύστημα μπορεί να χρησιμοποιηθεί από την αρχιτεκτονική για το master dataset όπως έχει περιγραφεί. Δεν προτείνετε για τους λόγους ότι περιέχει πολλές επιπλέον υπηρεσίες που δεν χρειάζονται, όπως indexing των δεδομένων, απαιτεί μια σχεδίαση για τον καταμερισμό των δεδομένων και παρέχει από μόνο του μηχανισμούς για Update οι οποίοι θα πρέπει να απενεργοποιηθούν με κάποιο τρόπο.

2.3 Filesystem

Οι βάσεις δεδομένων οποιασδήποτε μορφής απαιτούν μια εξοικείωση του χρήστη με τον τρόπο λειτουργίας, τους μηχανισμούς και το σχήμα της βάσης για να μπορέσει να τις χρησιμοποιήσει. Αντίθετα το Filesystem είναι ένα σύστημα αποθήκευσης με το οποίο όλοι είναι εξοικειωμένοι ως μέρος κάθε λειτουργικού συστήματος. Επίσης ένα Filesystem αποτελεί μια πολύ απλή λύση στο πρόβλημα της αποθήκευσης καθώς τα πάντα γράφονται σε αρχεία και αποθηκεύονται σε φακέλους. Οι μέθοδοι εγγραφής είναι εύκολοι και γρήγοροι, απλά επισυνάπτονται δεδομένα σε αρχεία ή δημιουργούνται νέα. Το μέγεθος των δεδομένων που αντιμετωπίζουμε όμως δεν μπορεί να το χειριστεί ένας υπολογιστής με τα μέσα που διαθέτει, συνεπώς η αρχιτεκτονική Lambda απαιτεί ένα Distributed Filesystem.

3. Distributed Filesystem

Το πρόβλημα με το Filesystem, όπως έχει αναφερθεί είναι ότι ανήκει σε ένα μόνο μηχάνημα. Αυτό θέτει περιορισμούς στο μέγεθος του αποθηκευτικού χώρου και στην απόδοση του συστήματος. Τα προβλήματα αυτά έχουν ήδη επιλυθεί με την υλοποίηση των Distributed Filesystem, δηλαδή ενός Filesystem το οποίο αποτελείτε από πολλά διαφορικά μηχανήματα συνδεδεμένα μεταξύ του μέσω δικτύου. Οι λεπτομέρειες για το πως ακριβώς δουλεύει ένα τέτοιο σύστημα δεν είναι κομμάτι αυτής της εργασίας. Θα αναλυθούν όμως μία μία οι απαιτήσεις που προαναφέρθηκαν και πως τις καλύπτει το Distributed Filesystem.

3.1 Immutability

Οι ενέργειες που επιτρέπονται σε ένα Distributed Filesystem είναι σχεδόν ίδιες με αυτές ενός απλού Filesystem, με κάποιες μικρές διαφορές, μία από αυτές, η τροποποίηση αρχείων.Πολλές υλοποιήσεις των Distributed Filesystem απαγορεύουν την αλλαγή αρχείων που έχουν αποθηκευτεί. Ο μόνος τρόπος προσθήκης δεδομένων είναι με την δημιουργία και αποθήκευση νέων αρχείων. Συνεπώς αυτές οι υλοποιήσεις, έχουν ήδη διαφυλάξει την αμεταβλητότητα των αρχείων. Για κάθε εισαγωγή δεδομένων θα δημιουργείται και ένα νέο αρχείο.

3.2 Scalability-Parallelism

Τα αρχεία δεν είναι αποθηκευμένα σε ένα μηχάνημα, αλλά βρίσκονται διαχωρισμένα σε όλα τα μηχανήματα του συστήματος μας. Ο αποθηκευτικός χώρος είναι πολύ μεγαλύτερος από αυτόν που μπορεί να προσφέρει ένα μηχάνημα και η αύξηση του είναι πλέον πολύ απλή ενέργεια, με την εισαγωγή νέων μηχανημάτων. Επίσης τέτοια Distributed συστήματα παρέχουν έτοιμες παράλληλες τεχνικές για όλα τις πράξεις που πρέπει να εκτελεστούν ώστε να είναι αποδοτικά.

3.3 Data Loss

Τα Distributed Filesystem παρέχουν μηχανισμούς για την διαφύλαξη των δεδομένων από hardware failures, network failures κτλ. Το κάθε σύστημα έχει διαφορετικούς τρόπους για να το επιτύχει αλλά όλοι είναι αρκετά αποδοτικοί. Φυσικά κανένα σύστημα δεν μπορεί να είναι τελείως ασφαλές και να υπάρχει πρόβλεψη για κάθε πρόβλημα που μπορεί να προκύψει, αλλά με τα μέτρα που λαμβάνει το Distributed Filesystem, συνεχείς ελέγχους και περιοδικά backup του master dataset, η απώλεια δεδομένων αποτελέι πολύ σπάνια περίπτωση.

3.4 Υλοποιήσεις

Όπως αναφέρθηκε, υπάρχουν πολλές έτοιμες υλοποιήσεις Distributed Filesystem. Κάποιες από αυτές, που παρέχουν και προστασία από απώλεια δεδομένων, ονομαστικά είναι:

- BeeGFS
- Ceph
- GFS (Google Inc.)
- Windows Distributed File System (DFS) (Microsoft)
- GlusterFS (Red Hat)
- HDFS (Apache Software Foundation)
- LizardFS (Skytechnology)
- MooseFS (Core Technology / Gemius)

- OneFS (EMC Isilon)
- OrangeFS (Clemson University, Omnibond Systems)
- ObjectiveFS
- Panfs (Panasas)
- RozoFS (Rozo Systems)
- Parallel Virtual File System (Clemson University, Argonne National Laboratory, Ohio Supercomputer Center)
- XtreemFS

4. Σύνοψη

Η επιλογή του σωστού συστήματος αποθήκευσης έχει επίσης μεγάλη συνεισφορά στην απόδοση του συστήματος. Η σωστή ανάλυση των απαιτήσεων και των συστημάτων που προσφέρονται αποτελεί ένα αναγκαίο βήμα στον σχεδιασμό της κάθε αρχιτεκτονικής. Οι κυριότερες απαιτήσεις που πρέπει να καλυφθούν αφοράν τα δεδομένα και την διαφύλαξη τους με κάθε δυνατό μέσο. Άλλες απαιτήσεις που μπορούν να βοηθήσουν στην επιλογή του κατάλληλου συστήματος είναι η ευχέρεια και η γνώση των διαχειριστών με κάποιο/α από αυτά τα συστήματα, ο κώδικας του συστήματος και αν είναι open source, οι χρηματικοί πόροι. Αλλά αυτές οι απαιτήσεις είναι πολύ ειδικές και σχετίζονται με την κάθε εφαρμογή ξεχωριστά, συνεπώς δεν θα γίνει η ανάλυση τους.
Για τη συγκεκριμένη αρχιτεκτονική έχει επιλεχθεί το σύστημα HDFS.

Κεφάλαιο 4

HDFS

Το σύστημα HDFS (Hadoop Distributed File System) είναι ένα κατανεμημένο σύστημα αρχείων σχεδιασμένο για να λειτουργεί σε μεγάλο πλήθος μηχανημάτων. Έχει πολλές ομοιότητες με τα υπάρχοντα κατανεμημένα συστήματα αρχείων, αλλά και πολύ σημαντικές διαφορές.Το HDFS είναι εξαιρετικά ανεκτικό σε σφάλματα και έχει σχεδιαστεί για να λειτουργεί σε υλικό χαμηλού κόστους. Παρέχει υψηλό thoughput για τα δεδομένα της εφαρμογής και είναι κατάλληλο για εφαρμογές που έχουν μεγάλα σύνολα δεδομένων.Μερικές απαιτήσεις POSIX παραβλέπονται για να επιτραπεί και στις υπηρεσίες streaming τη πρόσβαση στα δεδομένα του συστήματος. Το HDFS αρχικά αναπτύχθηκε ως υποδομή για το έργο της μηχανής αναζήτησης Ιστού Apache Nutch. Τώρα είναι ένα κομμάτι του project Apache Hadoop.

1. Αρχιτεκτονική

Η υλοποίηση του HDFS που θα χρησιμοποιηθεί είναι η έκδοση που περιλαμβάνετε στο Hadoop 2, το οποίο θα αναλυθεί στα επόμενα κεφάλαια. Ακολουθεί την αρχιτεκτονική Master-Slave που είναι και η αρχιτεκτονική ολόκληρου του Hadoop. Οι δύο κύριες υπηρεσίες του είναι το Namenode και το Datanode. Αυτές οι δύο αποτελούν τον κορμό του συστήματος αλλά θα αναλυθούν και άλλες υπηρεσίες και ο ρόλος τους.

1.1 Namenode

Το Namenode είναι μια υπηρεσία master που εκτελείται σε ένα μηχάνημα του cluster και αποθηκεύει την κατάσταση των των αρχείων και καταλόγων σε μια ιεραρχία. Τα αρχεία αυτά περιέχουν στοιχεία όπως χαρακτηριστικά των αποθηκευμένων δεδομένων, όπως τα δικαιώματα, τροποποιήσεις και τους χρόνους πρόσβασης, ονομάτων και τον χώρο στο δίσκο. Τα δεδομένα που αποθηκεύονται στο HDFS είναι χωρισμένα σε blocks (συνήθως 128 megabytes, και κάθε block του αρχείου είναι αντιγραμένο σε πολλαπλά DataNodes. Το NameNode διατηρεί το namespace και την χαρτογράφηση του κάθε block στα DataNodes. Είναι μια πολύ σημαντική υπηρεσία χωρίς την οποία το σύστημα δεν μπορεί να λειτουργήσει. Για τον λόγο αυτό υπάρχει και μια υπηρεσία που ονομάζεται SecondaryNameNode η οποία τρέχει σε διαφορετικό μηχάνημα από το Namenode και κρατάει ένα πιστό όλων των δεδομένων του Namenode.Σε περίπτωση κατάρρευσης του Namenode, το σύστημα ειδοποιεί το SecondaryNamenode να αναλάβει την ευθύνη του Namenode μέχρι να επανέλθει σε σωστή λειτουργία.

1.2 DataNode

Η υπηρεσία Datanode εκτελείται στα slave μηχανήματα του cluster. Αναλαμβάνουν την αποθήκευση των blocks και των metadata που σχετίζονται με κάθε block.Τα metadata περιλαμβάνουν checksums για τον επιβεβαίωση της ακεραιότητας των δεδομένων και το generation stamp που δηλώνει την ακριβή ημερομηνία και ώρα εισαγωγής τους στο HDFS. Κατά την εκκίνηση, κάθε DataNode συνδέεται με το NameNode και εκτελεί ένα handshake. Ο σκοπός του είναι να εξακριβώσει την ταυτότητα του namespace και την έκδοση του λογισμικού του DataNode. Εάν είτε δεν ταιριάζει με αυτό του NameNode, η DataNode κλείνει αυτόματα.Μετά το handshake, το DataNode, εγγράφετε στον κατάλογο του NameNode και στέλνει ένα report με όλα τα blocks που κατέχει και κάποια metadata. Με αυτό τον τρόπο το Namenode γνωρίζει την τοποθεσία του κάθε block και σε πόσα διαφορετικά Datanodes βρίσκεται. Κατα την λειτουργία του Datanode, στέλνει σήματα στο Namenode ανα τακτά χρονικά διαστήματα, για να επιβεβαιώσει την σωστή λειτουργία του.Εάν κάποιο Datanode σταματήσει να στέλνει σήματα, θεωρείται εκτός λειτουργίας, το Namenode θα αναφέρει το σφάλμα και θα συνεχίσει την λειτουργία του με τα υπόλοιπα, τα blocks που βρίσκονται αποθηκευμένα σε αυτό το Datanode δεν είναι προσβάσιμα και αν ζητηθούν, θα χρησιμοποιηθούν τα αντίγραφα τους από άλλα Datanodes.

1.3 HDFS Client

Η πρόσβαση των εφαρμογών στα δεδομένα γίνεται μέσω του HDFS client. Είναι μια βιβλιοθήκη που παρέχει μεθόδους βασικά για την ανάκτηση και την εγγραφή δεδομένων από τον HDFS. Αυτή η υπηρεσία θα κάνει τις απαραίτητες αιτήσεις στο Namenode για κάποιο αρχείο ή μια λίστα με αρχεία. Δεν χρειάζετε να γνωρίζει το πως είναι διανεμημένα και αποθηκευμένα στον cluster, συνεπώς παρέχει ένα επίπεδο abstraction μεταξύ της εφαρμογής και της αρχιτεκτονικής του HDFS.

Παράρτημα Α΄

Ορολογία