

# LOW COST BIG DATA ANALYTIC SOLUTION FOR PMT

Submitted By

**BHUMI B PATEL**

15MCEC17



DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

MAY 2017

# LOW COST BIG DATA ANALYTIC SOLUTION FOR PMT

**Project**

Submitted in partial fulfillment of the requirements

For the degree of

**Master of Technology in Computer Science & Engineering**

Submitted By

**BHUMI B PATEL**

**15MCEC17**

Guided By

**Prof. Kruti Lavingia**



**DEPARTMENT OF COMPUTER ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**

**MAY 2017**

## Certificate

This is to certify that the Project entitled "LOW COST BIG DATA ANALYTIC SOLUTION FOR PMT" submitted by BHUMI B PATEL (15MCEC17), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science & Engineering of Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Kruti Lavingia  
Guide & Assistant Professor,  
Department of Computer Engg,  
Institute of Technology,  
Nirma University, Ahmedabad

Dr. Priyanka Sharma  
Professor & Coordinator (M.Tech - CSE),  
Department of Computer Engg,  
Institute of Technology,  
Nirma University, Ahmedabad

Dr. Sanjay Garg  
Professor and Head,  
Department of Computer Engg,  
Institute of Technology,  
Nirma University, Ahmedabad

Dr. Alka Mahajan  
Director,  
Institute of Technology,  
Nirma University, Ahmedabad

## Certificate

This to certify that BHUMI B PATEL (15MCEC17), a student of M.Tech in Computer Science Engineering, Institute of Technology, Nirma University, Ahmedabad has been working in this organization since the 5th of July, 2016 and has carried out her project work titled "LOW COST BIG DATA ANALYTIC SOLUTION FOR PMT". She is working as an intern under the supervision of Mr. Bharath Gandhi (Mentor), and Mr. Sharath kalyan (Manager). she has successfully completed the assigned work and is allowed to submit her project report. The results embodied in this project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mr. Bharath Gandhi  
Senior Technical Specialist,  
Radiology Solutions,  
Philips Innovation Campus,  
Bangalore

Mr. Sharath Kalyan  
Verification Manager,  
Radiology Solutions,  
Philips Innovation Campus,  
Bangalore

## Statement of Originality

I, **Bhumi Bharkumar Patel**, Roll. No. **15MCEC17**, give undertaking that the Major Project entitled "**LOW COST BIG DATA ANALYTIC SOLUTION FOR PMT**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

---

Signature of Student

Date:

Place:

Endorsed by  
Prof. Kruti Lavingia  
(Signature of Guide)

## Acknowledgements

First and foremost, sincere thanks to my mentor, **Mr. Bharath Gandhi**, (Technical Specialist, Radiology Solutions). I enjoyed his valuable guidance and inputs and owe him lots of gratitude for having a profound impact on this report. I would like to thank my teammate Chikkanna Vasu for their vast support, without which, this project work would never have been completed. It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Kruti Lavingia** for giving me an opportunity to work under her guidance. I would also like to thank my manager **Mr. Sharath Kalyan** for his immense support and encouragement.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Honble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr Alka Mahajan**, Honble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work. I also owe my friends and colleagues at Radiology Solutions Department, Philips Innovation Campus, special thanks for helping me on this path and for making this internship more enjoyable.

I would like to thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work. At last I would like to thank Nirma University and Philips Healthcare for providing resources and quality environment for research and development.

- **Bhumi B Patel**

**15MCEC17**

## Abstract

*There are so many departments in hospital. Radiology is one of the department. It mainly works with the medical imaging technology. Radiology machines are used to perform scans on human body. This scans are performed by technologists. Results of this scans are stored as a DICOM image in PACS. Radiologist takes this DICOM images, observe them and then prepare a report which is used by the physicians. Radiology is one of the most revenue generating department of the hospital and resource utilization is one of the big challenge for this department. PMT is one which helps this department in resource utilization. PMT provides a UI which displays some default dashboards and give user privilege to create dashboards based on their roles. Philips is using many third party tools for PMT. BI tool, which philips is using, performs well for adhoc queries but the only problem with this tool is, it is very costly. So, our aim is to find a low cost alternative with the same performance. This BI tool uses in-chip technology which is new and no other tools are using it. Other BI tools like Tableau, QlikView uses in-memory technology. But still this tools are also costly. Low cost big data analytic solution for PMT is given in this report. For the demo purpose we will use any public dataset, for the data processing we will use spark SQL, for data caching we will use tachyon.*

## List of Abbreviations

CT	Computed Tomography
MR	Magnetic Resonance
PIC	Philips Innovation Campus
RS	Radiology Solution
HIS	Hospital Information System
RIS	Radiology Information System
PACS	Picture Archiving and Communication System
EHR	Electronic Health Record
OLAP	Online Analytical Processing
RDBMS	Relational DataBase Management System
BI	Business Intelligence
HDFS	Hadoop Distributed File System
YARN	Yet Another Resource Negotiator



# Contents

<b>Certificate</b>	<b>iii</b>
<b>Certificate</b>	<b>iv</b>
<b>Statement of Originality</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
List of Abbreviations . . . . .	viii
List of Figures . . . . .	xii
List of Tables . . . . .	xiii
<b>1 Introduction</b>	<b>1</b>
1.1 Objective of the Project . . . . .	2
<b>2 Theoretical Background &amp; Literature Survey</b>	<b>3</b>
2.1 Radiology Workflow . . . . .	4
2.1.1 HL7 . . . . .	5
2.1.2 DICOM . . . . .	5
2.1.3 HIS . . . . .	6
2.1.4 RIS . . . . .	6
2.1.5 PACS . . . . .	6
2.2 Data Complexity In Healthcare . . . . .	6

2.3	Database technologies . . . . .	8
2.3.1	OLAP Cubes . . . . .	8
2.3.2	In-Memory Database . . . . .	10
2.3.3	In chip memory . . . . .	13
2.4	BI tool that uses In chip . . . . .	14
2.5	Problem Statement . . . . .	17
2.6	Hadoop . . . . .	18
2.6.1	HDFS . . . . .	18
2.6.2	Map Reduce . . . . .	19
2.6.3	YARN . . . . .	20
2.7	Spark . . . . .	20
2.7.1	Resilient Distributed Datasets . . . . .	21
2.7.2	Spark SQL . . . . .	21
2.8	Tachyon . . . . .	23
2.8.1	Features . . . . .	23
2.8.2	Architecture . . . . .	25
<b>3</b>	<b>Proposed Approach</b>	<b>27</b>
3.1	Plan for theoretical analysis . . . . .	28
3.2	Metrics to be used . . . . .	28
3.3	Proposed Tools to be used . . . . .	28
<b>4</b>	<b>Implementation &amp; Results</b>	<b>29</b>
4.1	Hadoop Installation . . . . .	29
4.2	Spark Installation . . . . .	36
4.3	Tachyon Installation . . . . .	38
4.3.1	Alluxio Master Web Interface . . . . .	39
4.3.2	Alluxio Workers Web Interface . . . . .	42
4.4	Tachyon(Alluxio) + Spark . . . . .	44
4.4.1	General setup . . . . .	44

<i>CONTENTS</i>	xi
4.4.2 Alluxio as Input and Output . . . . .	45
4.5 Results . . . . .	45
<b>5 Conclusion &amp; Future Work</b>	<b>48</b>
5.1 Conclusion . . . . .	48
5.2 Future Work . . . . .	48

# List of Figures

2.1	Radiology Workflow . . . . .	4
2.2	OLAP cube example . . . . .	9
2.3	BI Application Data Flow . . . . .	16
2.4	Alluxio Layer . . . . .	25
2.5	Alluxio Architecture . . . . .	26
3.1	Proposed Approach . . . . .	27
4.1	Performance Matrix for Text Files . . . . .	46
4.2	Performance Matrix for CSV Files . . . . .	46

# List of Tables

I	Comparison of Disk and RAM . . . . .	11
II	Comparison of RDBMS, In-Memory technology and In-Chip technology	14

# Chapter 1

## Introduction

Hospitals has many departments like accident emergency, cardiology, critical care, gynaecology, neurology, oncology, radiology, urology etc. Radiology is the department which works with the digital imaging. It contains different modalities like CT, MR, Ultrasound, X-Ray etc. These modalities are handled by different modality managers. Scans are performed on the human body through this modalities by technologists. Technologists are the person who perform the scans and analyzes the results. Once the scans are done for a particular patient, radiologist examines the results and prepare a report and send it to a doctor/physician. All this data are stored and transferred in DICOM and HL7 format. Resource utilization is the challenging task for this department. Each and every modality cost is really high and better utilization of them is required to increase the revenue.

Philips Innovation Campus (PIC) is driving a big data analytics product named as PMT to enable hospital personnel to improve operational efficiency. The PMT is a new software framework/platform which provides the information, analysis, reporting and improvement engine which will be sold as a component of a larger imaging services solution. The main emphasis of the PMT is to facilitate quality and efficiency improvements in operational, clinical, and financial performance for the customers.

PMT includes many third party components. BI tool (RS currently using) is selected for PMT because of its remarkable speed and end-to-end BI capability. Reason behind the remarkable speed of this tool is the in-chip technology. All most all the BI tools make use of in-memory technology but this tool is the first one to exploit the possibilities of Intel x86 CPU. The only problem with this tool is its cost. So, aim is to find a low cost alternative for PMT.

## 1.1 Objective of the Project

The objective of this project is to create

- Cost effective data warehousing solution
- Low latency querying on TB scale
- Consistent caching mechanism across entire data warehouse pipeline

## Chapter 2

# Theoretical Background & Literature Survey

As mentioned in the chapter 1, Resource utilization is one of the problems faced by radiology department in hospitals. As radiology department is the most revenue generating department in the hospital, problems faced by this department need to be addressed. In US, one rule has been passed according to which government will pay only for those medical scans which reveals the defects. So, radiology department of hospitals needs some means by which they can analyze and monitor the activities of the department. This is needed to make sure that no unwanted scans happens for any patient and all the medical devices and staff are utilized properly as it will affect the revenue of the hospital. Philips Innovation Campus (PIC) is driving a big data analytics product named as PMT to enable hospital personnel to improve operational efficiency. The PMT is a new software framework/platform which provides the information, analysis, reporting and improvement engine which will be sold as a component of a larger imaging services solution. The main emphasis of the PMT is to facilitate quality and efficiency improvements in operational, clinical, and financial performance for the customers.



## 2.1 Radiology Workflow

Patient arrives at hospital and registration happens. All the patient information get stored into HIS. Referring physician orders required radiology procedure. Once the order is placed, required patient data is stored into the RIS and order is scheduled. Modality fetch the data from RIS and if there are any relevant prior studies has been done then it will be fetched from PACS. Before exam starts, prework is done. Once the modality and patient is ready, acquisition starts. After the completion of the acquisition, results are stored in the PACS and if required, results (i.e. images) are printed. Radiologist fetch those result from PACS. Radiologist does the analysis to find defects and prepare a report which will be stored into report repository. Physician fetch those results and prescribes medicine or surgery accordingly. Figure 2.1 shows the pictorial representation for the same.

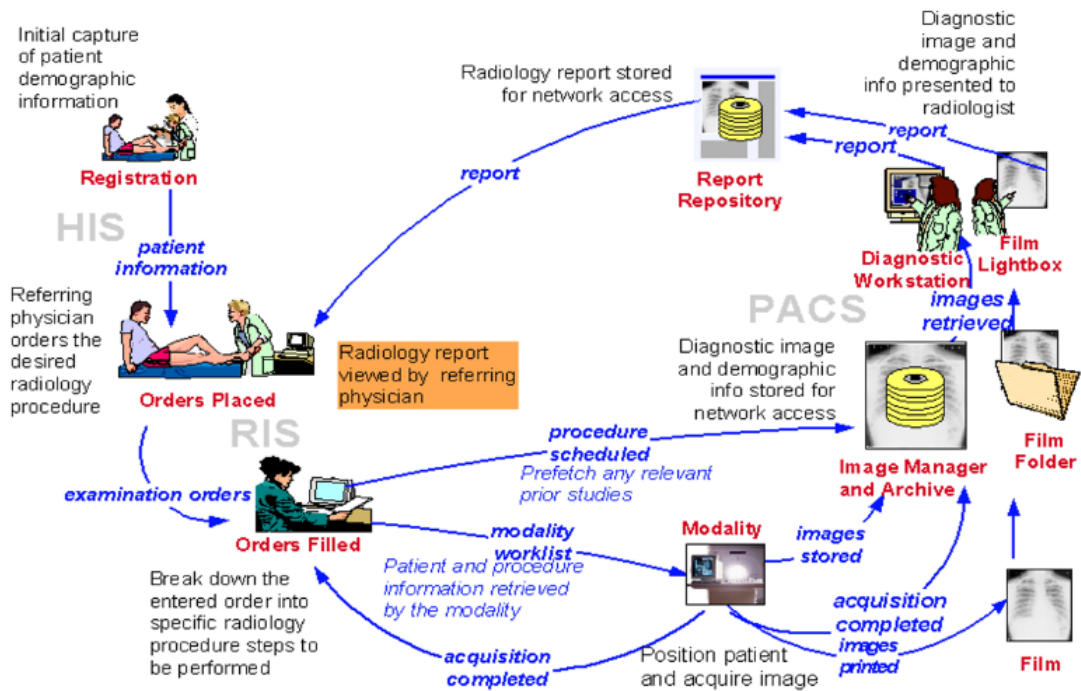


Figure 2.1: Radiology Workflow

### 2.1.1 HL7

Each hospital stores and share information in different ways. Whenever health information needs to move across organization boundaries, it hits the incompatible standards roadblock [2]. Someone has convert the format and pass the information. So we need a system which standardize the process of storing and sharing the health information. An EHR (electronic health record) was the initial solution. EHRs have been very successful in eliminating the problem of integrating systems within an organization and they continue to be one of the cornerstones of the healthcare IT structure [?]. It solved the problem within the organization but sharing of information outside the organization still faces a problem. Orgaization need to share patient information with insurance companies and send patient care information to the government. So, EHRs were a limited solution and there is a need of better solution. HL7 was that better solution. HL7 is an ANSI accredited, OSI level 7, application layer protocol for exchanging clinical and administrative data between healthcare systems [2]. It defines the rule for sharing the health data between applications.

### 2.1.2 DICOM

HL7 is generally used for health record which includes patient information while DICOM is used for storing medical images (i.e. scan results). DICOM incorporates standards for imaging modalities such as radiography, ultrasonography, computed tomography (CT), magnetic resonance imaging (MRI), and radiation therapy [3]. Image formats like JPEG have embedded tags from which we can identify the image. Similarly DICOM also groups the information into data sets. For example, a file of a x-ray image of a patient also includes the patient id. Data object of DICOM contains many attributes like patient name, patient id, etc. and a special attribute which contains image pixel data.

### 2.1.3 HIS

A hospital information system ( HIS ) is essentially a computer system that can manage all the information to allow health care providers to do their jobs effectively [4]. It mainly focuses on administrative needs. It is designed to manage hospitals operation like administrative, financial, legal issues etc.

### 2.1.4 RIS

A radiology information system (RIS) is a networked software system for managing medical imagery and associated data [5]. It is useful in tracking information related to billing and orders. With PACS, it manage record-keeping, billing and image archives.

### 2.1.5 PACS

In medical imaging, electronic picture archiving and communication systems (PACS) have been developed in an attempt to provide economical storage, rapid retrieval of images, access to images acquired with multiple modalities, and simultaneous access at multiple sites [6].

## 2.2 Data Complexity In Healthcare

Two things need to be considered while choosing the hardware and software for big data analysis

- What type of data has to be process and for what purpose?
- What kind of processing has to perform for analysis and how quickly we need to get the results?

It takes time to prepare data due to data indexing, data modeling tasks, aggregation of data from various sources before they can be queried for analytical purposes. Throughout the analytical process, Specified skills and resources needed. Each time

changes are introduced to the data model, manpower cost is amplified in order to investigate new analytic paths. Often additional third party tools are required as well in order to alleviate some of the pains associated with this process [7]. License fees are required for analytical tools, as well additional data warehouse and data preparation tools [7]. More costs are added by the skills required to maintain and integrate different tools from different providers that may or may not work effectively together.

BI tool that RS department is currently using, provides healthcare organizations with business analytics software that was built specifically to simplify the analytical process for complex data, optimizing speed to handle larger data sets and unifying the process of analyzing data, thereby eliminating many steps in data preparation and easing the burden of preparing complex data for analysis in healthcare and other organizations [7].

This is achieved with the help of two core technology: In-Chip technology and Single-Stack architecture. In-Chip technology, optimized for processing speed by taking maximum advantage of modern CPU technology. This means the system can process 100s of millions of rows using low cost, commodity hardware [8]. A Single-Stack architecture joins all the elements of a healthcare analytics solution, from data preparation to data management, querying and visualization, into a single, efficient software solution. The combination of these two core technologies solves the bulk of the problems associated with complex data.

Healthcare organizations use this technology to combine data from the many disparate systems they work with. Using this technology they can create operational dashboards which medical and administrative staff can use for immediate answers to questions that arise in their day-to-day work, and healthcare analysts can dive deeper into the data to reach new insights and suggest data-driven courses of action.

## 2.3 Database technologies

Database technologies have been evolved rapidly in last two decades. OLAP has gained popularity in 1990s but at the start of 21st century, in-memory databases have gained that popularity. However, the requirements of modern business intelligence have set a challenge that in-memory databases will have a very difficult time responding to [8]. To overcome the problems faced by in-memory databases, new technology i.e. in-chip technology has been introduced.

### 2.3.1 OLAP Cubes

This technology was first developed in the late 1960s, but it gained widespread commercial use in the 1990s. As computer hardware was not much powerful at that time compared to today, OLAP was advanced. For analysts, OLAP has introduced a better way to do multidimensional analysis on large volumes of data. To build an OLAP cube, table datasets are converted into multi-dimensional arrays to optimize data retrieval and querying. For analysis, Users can access particular dimensions of the data after building cubes.

For a simplified example, lets think of a chain of pet stores that tracks sales of various items across cities and over time. It might track these figures in a series of spreadsheets such as these:

2012			
	MR Scans	CT Scans	X-Ray
Bangalore	201	298	311
Surat	150	148	203
Ahmedabad	190	158	207

2013			
	MR Scans	CT Scans	X-Ray
Bangalore	199	230	303
Surat	148	160	214
Ahmedabad	187	179	276

2014			
	MR Scans	CT Scans	X-Ray
Bangalore	178	293	298
Surat	154	139	190
Ahmedabad	196	170	201

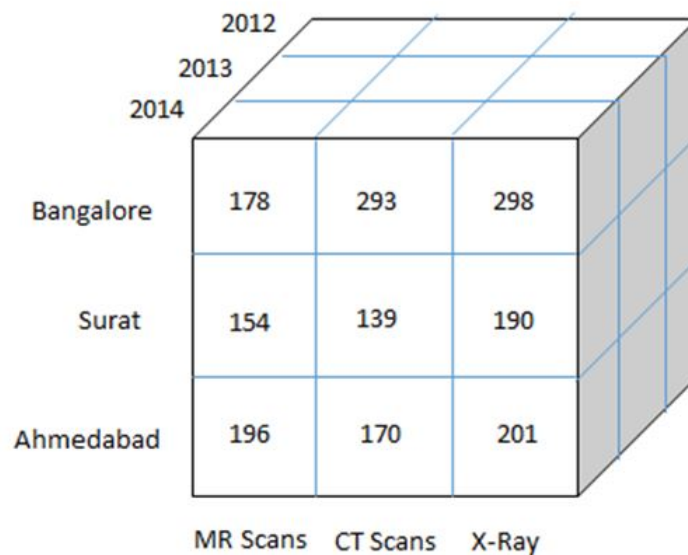


Figure 2.2: OLAP cube example

To answer queries, some basic operations like roll-up, drill-down, slice, dice and pivot are performed on OLAP cubes. These operations contain aggregated data which are previously calculated when the system is not being used by users (i.e. at rest). So, once a query is made, the answer is already within the data cube and recovered directly.

But OLAP cubes have some disadvantages. Each query requires a new dimension which is added to the cube. OLAP databases quickly become resource intensive when it comes to data storage and management [8]. Each new build takes a relatively long time to produce because every cell of the data is requires aggregating data for the CPU to process.

For pre-designed queries, OLAP cubes are very fast but data for new query is not already calculated and requires other dimensions to be added to the cube which is a long process.

**Advantages:**

- Centralized data integration
- Fast data retrieval for specific queries

**Disadvantages:**

- limited support for ad-hoc queries
- Long build times

### 2.3.2 In-Memory Database

In-memory databases became popular in the start of the 21st century with the creation of 64-bit PCs which are cheap and widely available and the adoption of columnar databases as an alternative to the row-based systems which were the basis for OLAP cubes [2]. More RAM on a PC allow more data to be quickly queried. If preparation and processing a million rows of data on a machine with only 2GB of RAM was a tedious job, users could now add more GB of RAM to their PCs and store data in relational databases which could be queried much faster than before. In-memory databases have become much more prominent in recent years. However OLAP-based

solutions can still be found in massive organization-wide implementations [8].

A computer have mainly two types of data storage tools

- Disk and
- RAM (random access memory).

The main differences between Disk and RAM are defined in the following table:

DISK	RAM
Abundant	Limited
Slow	Fast
Cheap	Expensive
Long term	Short term

Table I: Comparison of Disk and RAM

Modern computers have more disk storage than the RAM. Reading Data from disk storage is slow than reading data from RAM and cost of RAM is more than cost of disk. Two things that creates more disk operations in a disk-based relational database which result in poor performance are:

- **Table Scans:** Transferring tables from Disk to RAM for calculations
- **Complex Data:** Query needs data from different tables which requires joins

In-memory technology takes care of both these issues. It first loads the complete database into RAM and then transfer data to the CPU from RAM for data fetch and calculation. All In-memory technologies share the same premise: that it is simply much faster to perform calculations over data that is stored in RAM than it is when that same data is stored in a table on a disk [8]. As the 64-bit computers are considered commodity hardware In-memory technology takes advantage of it. Even it is relatively cheaper now to add more RAM to both commodity and proprietary hardware. In-memory technology performs well, at small scales. When datasets are small and simple, it enables faster development compared to a solution which is built



on top of an RDBMS. However, scalability is the issue which it faces. When the RAM is used to store the raw data, it be likely to run out quickly that is the major challenge related to RAM. As storage size increases, RAM is small and many data sets now a days are too large to fit. Additionally, each query to the database uses RAM for intermediate calculations. Complex situations still require that data be widely modified and loaded into a data warehouse, before being loaded into the memory-based storage. This situation can arise when there are many users who are querying the database simultaneously or data sets are complex.

The fact is, data sets are getting larger and larger, with companies creating more information than continuously both from internal sources and from external ones which business executives look to in order to gain a competitive advantage [8]. Though the RAM prices came down with the exponential growth in the size of data, its still comparatively costly storage which cannot be scaled indeterminately. In memory technology works well for small amount of data but it does not promise reasonable performance for large amount and difficulty of the data which is currently being aggregated, analyzed and gathered by modern businesses.

**Advantages:**

- Fast data retrieval
- Support for ad-hoc queries

**Disadvantages:**

- Expensive to implement and maintain
- Scalability issues

### 2.3.3 In chip memory

ElastiCubes In-Chip Technology is growing rapidly as an alternative solution because of the limitations of traditional OLAP database technologies. ElastiCube is the result of thoroughly analyzing the strengths and weaknesses of both OLAP and in-memory technologies. The aim is to provide a true alternative to OLAP technology, without compromising the speediness of the development cycle and query response times for which in-memory technologies are praised [8]. In-Chip technology is the new generation of in-memory technology used for business analytics and it is fast as well as scalable. The name ElastiCube comes from the databases unique ability to stretch beyond the hard limitations imposed by older generation technologies [8].

This technology uses a disk-based columnar database for storage. It is to provide fast disk reads and is able to load data from disk to RAM when it is required. The queries are managed in-memory without reads from the disk. And most fundamentally, RAM limitations are not big issue compare to in-memory technology because no need to keep entire data in RAM on a eternal basis. It is succeeded through innovative compression and credentials of the database which are not used on a constant basis and can be gone at rest usually the data businesses are collect around 80 percent.

This Technology also has a different way of creating joins from different tables. It uses columnar algebra to join between fields Instead of joining tables. This approach, the join process can be handled only in the CPU cache.

The table below compares between Relational DBMS technology, In-Memory technology and In-Chip Technology by several technical features:

Feature	RDBMS	In-Memory Asso- ciative	In-Chip Technol- ogy
Columnar Storage	Some	No	Yes
In Memory Query Processing	No	Yes	Yes
Performance Upon Installation	Slow	Fast	Fast
Data Capacity	Unlimited	Limited (by size and RAM)	Unlimited
Scalability Level	Large scale	Small scale	Small / Large scale

Table II: Comparison of RDBMS, In-Memory technology and In-Chip technology

## 2.4 BI tool that uses In chip

Speed is the most important factor for any application as users do not like to use the application which takes too much time to respond. Speed of any application depends on the available resources and how application makes great use of those resources. Very few BI products are taking advantage of the possibilities of the Intel x86 CPU. BI tool (which RS using) uses Intel x86 CPU and it makes full use of that CPU which gives it a speed advantage. BI applications are data heavy they need to process a fairly large amount of data to produce a result [9]. BI applications first collect the data from a database, then they perform a set of analytical calculations and display the result for the user, in chart or dashboard form. Data are continuously transferred from disk to memory and then into the CPU till the analysis has not been completed and the results are not displayed on screen.

An x86 CPU has three layers of in-chip memory. Data are stored in that layers before they are processed. This layers are cache named L1, L2 and L3. In multiprocessor environment, each CPU has individual L1 and L2 cache while L3 cache is shared between all cores. The caches have different capacities. L1 Cache has a capacity of 32Kb, L2 has 256Kb and L3 has 8Mb to 20Mb [9].

Consider that core takes  $x$  time to process the data which is already available in core. So if the data is available in L1 cache then fetching the data from L1 cache to core, it will take  $3x$  time. Further if data is in L2 cache then fetching the data from L2 cache to core, it will take  $3.3x$  time. If the data is available in L3 cache then it will take  $3.5x$  time. Fetching the data from memory will take nearly  $10x$  time and fetching data from disk is thousands time slower. So if the data is in L1 cache, latency will be less and if data is in disk, latency will be more. So, while developing the BI software, this data flow needs to be handled.

BI tool (which RS using) uses a columnar database. It stores the data in to disk and not in memory. Whenever data comes, it will first compress all the data and then save it to the disk. Data will be in compressed form until it comes to L1 cache. This mechanism will improve the I/O performance because it will reduce the transfer time. Data will be decompressed in L1 cache. This saves memory bandwidth. This is called cache aware decompression. Because of the cache aware decompression, more RAM will be available for other resources. If the CPU is kept as busy as possible and it is used efficiently, the software runs much faster no matter how much memory is available [9].

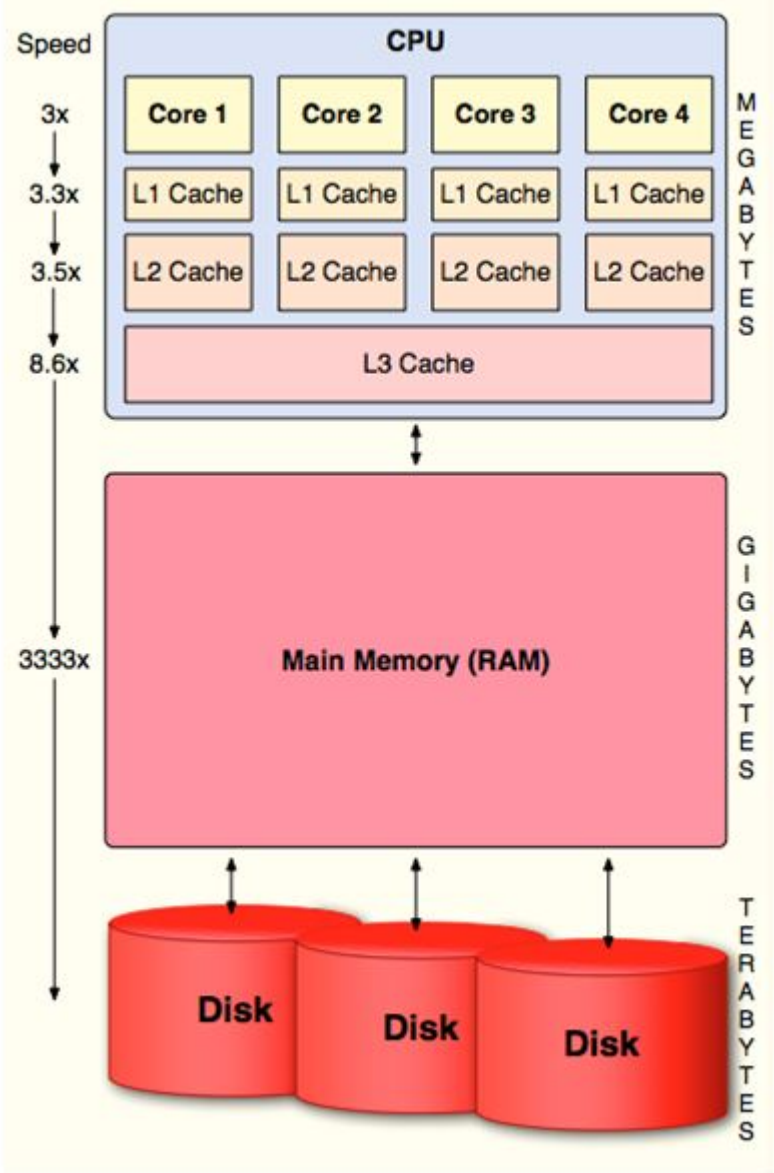


Figure 2.3: BI Application Data Flow

As mentioned above, different cache has different storage capacity but that is not more than 20 Mb. Disk can store TB of data. This BI tool will keep track of the location of data. Because of the ElastiCube technology, large volume of data can be queried from database.

Basically, this BI tool, decompose the queries into subqueries. Many of this subqueries

are repetitive. BI tool can take advantage of this pattern. This BI tool has included a learning algorithm which learns characteristics of the various requests (queries) that it has to satisfy so that it can optimize for them [9]. Subqueries that are decomposed from main query are pre-loaded into L1 cache and is in compressed form. Then it will be decompressed and image of the same are also moved to L2 and L3 caches to make the read and write operation fast.

Subquery results are pre-loaded into L1 Cache as compressed data, making extremely economic use of this very fast but limited resource. Later decompressed images of that same data are moved to the larger, but slower, L2 and L3 Caches. So decompression operations read from and write to cache, and thus are extremely fast. As more queries are processed, BI tool can reuse those results which will improve the performance. So higher workload wont slow down the tool but it will improve the speed. Processing data in chip is at least 100 times faster than processing it in-memory [9].

## 2.5 Problem Statement

BI tool which philips is using for PMT is a complete end-to-end BI tool. It also performs well for adhoc queries. But the only problem with this tool is, it is very costly. So, our aim is to find a low cost alternative with the same performance. In-chip technology is new, no other tools are using it. Other BI tools like Tableau, QlikView uses in-memory technology. But still this tools are also costly.

As this is the big data solution, we have started with hadoop and spark. Some basic concepts about hadoop and spark are covered in section 2.6 and 2.7.

## 2.6 Hadoop

Hadoop was designed to handle large dataset. It is an open source framework which is designed to process and store large amount of data which cant be store in a single disk space.

### 2.6.1 HDFS

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware [10].

#### **Very large file:**

File size is in GB or TB or more which cant be stored in a single disk.

#### **Streaming data access:**

The idea behind Building a HDFS is that the write all the efficient pattern once and then read them multiple times. Before performing any analysis on the datasets, they are fetched from different souces first.

#### **Commodity hardware:**

Commodity hardware is the one which is commonly available and it can be obtained from multiple vendors. Haddop is designed in such a way that it can run on clusters with commodity hardwares i.e. expensive hardwares are not required for hadoop. So, there will be a high chances of failure of nodes.

#### **Blocks:**

In general filesystem block size is in KBs and disk block size is 512 bytes. But default HDFS block size is 128 MB [10]. Reason behind this large block size is to reduce the cost of seeks. To make sure that the data will not be lost, replication for each block is done. Replication factor here is typically three.

**Namenodes and Datanodes:**

HDFS cluster has two type of nodes: 1) Namenodes and 2) Datanodes. Namenode is responsible for maintaining metadata of all the directories and files. This information is maintained in two forms: namespace image file and edit log file [10]. Namenode knows on which datanode, blocks for a particular file is located but it does not store location of block because location information is restored after the system starts from datanodes. A client accesses the filesystem through namenode and datanodes. Datanodes store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing [10]. As namenode store the metadata of directories and files, without the namenode filesystem cant be used. If the machine that is running the namenode is destroyed, all the files stored on the filesystem would be lost because there will be know way of reconstructing the files from the blocks. This is the reason why namenode should be resilient to failure. This can be achieved in two ways. The first way is to back up the files that make up the persistent state of the filesystem metadata. Hadoop can be configured so that the namenode writes its persistent state to multiple filesystems [10]. Another option is to maintain a secondary name node. It will not act as a name node but it will simply merge the namespace image with the edit log which will prevent edit log from becoming very large.

**2.6.2 Map Reduce**

Hadoop processing happens in two phase i.e map phase and reduce phase. Input and output of each phase is key value pair. MapReduce divides the dataset and this datasets are being processed parallely. Each outputs of map task become the input to reduce task.



**Example:**

Consider five files, each having two columns i.e. city and temperature. We want to find a maximum temperature for each city. For this, each file will be given to different map task. Map task will go through that file which it is processing and find maximum temperature for each city. After this result of each map task is given to reduce task which will combine this result and find the maximum temperature for each city.

**2.6.3 YARN**

In MapReduce 1, there are two types of daemon that control the job execution process: a job tracker and one or more task trackers [10]. The job tracker takes care of the scheduling the tasks for task trackers. The task tracker executes the tasks and sends the report to job tracker. Whenever task fails, job tracker reschedules it. So basically job tracker manages both scheduling of a task and monitoring task progress. By contrast, in YARN these responsibilities are handled by separate entities: the resource manager and an application master (one for each MapReduce job) [10].

**2.7 Spark**

Apache Spark is developed to overcome the I/O limitations and an alternative for Hadoop MapReduce. It has emerged as the de facto standard for big data analytics after Hadoop's MapReduce [11]. It is newest parallel computing engine working with Hadoop. Spark allows the data to be cached in memory and thus removes disc overhead and increases the processing speed by 100 times [12]. As a framework, it combines a core engine for distributed computing with an advanced programming model for in-memory processing [11]. It is emerged using Scala which supports both object-oriented and functional programming. It has fault tolerance capabilities and linear scalability same as MapReduce. Hadoop uses disk based programming model while spark uses multistage in-memory programming model. Having an advanced model, Apache Spark is easy to use and faster. Apache Spark leverages the memory of a

computing cluster to reduce the dependency on the underlying distributed file system, leading to dramatic performance gains in comparison with Hadoops MapReduce [13]. It is also called a general-purpose engine because set of applications combine all the computations like iterative algorithms, job batches and interactive queries which required separated distributed systems.

### 2.7.1 Resilient Distributed Datasets

Spark created a unique data structure called Resilient Distributed Datasets (RDD [14]), which allows Spark application to keep data in memory, while MapReduce relies on HDFS to keep data consistent and provides an efficient data sharing between computations [15]. An RDD is a partitioned collection of records, read-only. RDDs provide fault-tolerant, parallel data structures that let users store data explicitly on disk or in memory, control its partitioning and manipulate it using a rich set of operators [15]. An RDD can be generated either from other RDDs or external data sources. RDD supports coarse grained transformation and logging them to provide fault tolerance [16]. RDD avoids replication of data by the graph of operations which were created it. When data is lost on failure then it can efficiently re-compute. When the user want to use multiple times then RDDs should be explicitly cached. Spark provide the RDD abstraction by using a simple programming interface. Every RDD is a common interface with five sections of information: an iterator, partitions, data placement, metadata and dependencies about its separating schema. This representation can efficiently express several cluster computing models that previously required separate frameworks [17].

### 2.7.2 Spark SQL

Spark SQL is a component of Apache Spark that supports SQL-like processing of structured data [18]. Spark SQL is evolved by the Shark [18], and reducing the dependence on the Hive. Spark SQL enables users to combine declarative syntax of

SQL and the concise with the control of procedural programming languages. Spark SQL absorbs the memory storage (In-Memory Columnar Storage) of Shark, compatibility of Hive; it has a great development in terms of data compatibility, performance optimization, components extension [memory computing]. Spark SQL involves three modules as follows:

- Core: It processes I/O data which is getting from multiple sources like JSON, Parquet, RDD, etc. then the query results are represented as schema RDD
- Catalyst: It executes the queries during the process with including optimization, parsing, physical plans, binding.
- Hive: Hive provides CLI and JDBC / ODBC interfaces for Hive data processing [19].

In these modules, Catalyst is the core part of the merits and its performance will affect the overall performance [19].

#### **Advantages:**

- Data compatibility: It is not only compatible to hive but also achieved from the JSON and RDD files.
- Performance optimization: It presents Cost Model to evaluate dynamic query and get the best physical plan because spark SQL uses In-Memory storage and different optimization techniques.
- Component extension: To redefine and expand the parser, analyzer and optimizer of SQL [19].

Spark SQL is the main project of the Spark. The optimization is most important for Spark SQL which is utilized maximum performance of distributed parallel system and hardware resources.

## 2.8 Tachyon

Alluxio, also known as Tachyon. It is the first memory speed virtual distributed storage system in the world. It combine bridges computation framework and data access with primary storage system. Applications only need to connect with Alluxio to access data stored in any underlying storage systems [20]. In addition, Alluxio has memory-centric architecture which permits data access orders much faster than existing solutions. In the big data system, Alluxio is between computation jobs or frameworks, such as Apache Flink, Apache MapReduce, or Apache Spark, and different storage systems, such as GlusterFS, Amazon S3, HDFS, Google Cloud Storage, Alibaba OSS, or OpenStack Swift. Tachyon brings significant performance improvement to the ecosystem; for example, Baidu uses Alluxio to improve speedup the throughput of their data analytics pipeline 30 times [20]. Apart from performance, Tachyon associations new workloads with data stored in old storage systems. Users can use Tachyon by using standalone cluster mode, such as on Google Cloud, Amazon EC2 using its standalone cluster mode, for example on, Google Cloud, or launch Tachyon with Apache Yarn or Apache Mesos. Alluxio is compatible to Hadoop. MapReduce and Spark programs can run on top of tachyon without change in code. It is one of the fastest developing open source projects under Apache License 2.0 which is set up at many companies.

### 2.8.1 Features

- Tiered Storage – Tachyon can achieve HDDs and SSDs apart from memory it allows to store large datasets in Tachyon. Data will automatically be managed between the different tiers, keeping hot data in faster tiers. Custom policies are easily pluggable, and a pin concept allows for direct user control.
- Lineage – Using lineage, all the applications write output into memory, it check checkpoints periodically in an asynchronous way. If it is fails, it can recom-

putation restore the lost files. Tachyon can generate high throughput without compromise fault-tolerance using lineage.

- Flexible File API – By providing `InputStream` and `OutputStream` interfaces than get the best performance and support for memory I/O because Tachyon API is similar to one of the java file class. Instead of HDFS, we can use tachyon with Spark and Hadoop MapReduce because Tachyon is Hadoop compatible.
- Unified Namespace – Tachyon has transparent naming which make sure that directory hierarchy and file names for object generated in Tachyon when these objects to the primary storage system.
- Web UI & Command Line – Using the web UI, user can use the file systems easily. Administrators can access all the information (checkpoint path, locations, etc.) of every file in debug mode and user can also access `/bin` file to interrelate to Tachyon.

Alluxio has a unique place in the big data system because it has central point of access and also memory centric design. For computation frameworks and user applications, It manages fast storage and data access, enabling locality and data sharing between jobs, apart from whether they run on same computation machine or not. For big data applications, As a output, Alluxio can get an order of magnitude speed up because it provides a common interface for data access. Since Alluxio hides the integration of under storage systems to applications, any under storage can back all the applications and frameworks running on top of Alluxio [21]. Alluxio can work as a combining layer for all various data sources.



Figure 2.4: Alluxio Layer

### 2.8.2 Architecture

Alluxio has a standard master-slave architecture similar to HDFS and GFS (see Figure 2.4) except that: (1) Alluxio uses the RAM instead of local hard drive for storing data; (2) For achieve data fault-tolerance, Alluxio uses lineage-based recovery instead of data replication which is used by HDFS, to improve the write throughput. Alluxio has a single master with multiple workers. At high level, Alluxio is divided into three components, the master, workers, and clients. Alluxio servers make up by using master and workers and it is manage and maintain by system admin. The applications are clients, for example Alluxio command-line users or MapReduce or Spark. Users will required to connect with client component of Alluxio.

#### Master

The master is responsible for managing the global metadata of the system, for example, the file system tree [21]. If clients want to modify or read this metadata then

need to interact with master. Additionally, all workers are also connect to master to maintain participation in the system. If any components want to communicate with the master then they can through requests.

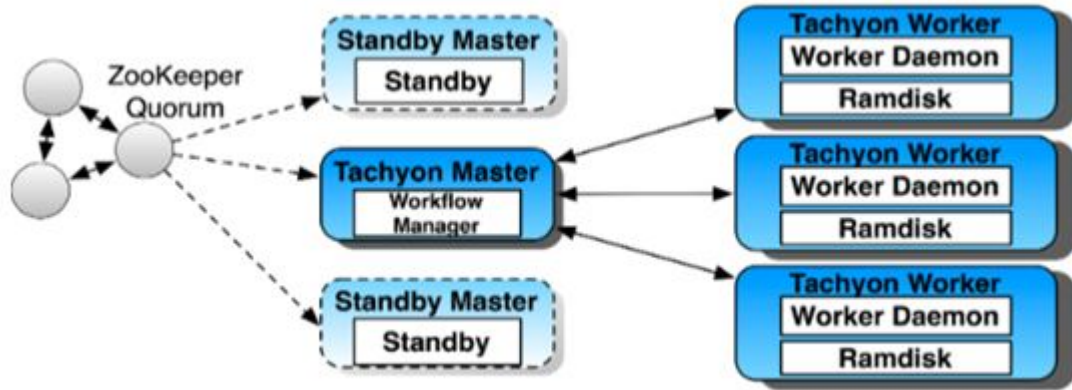


Figure 2.5: Alluxio Architecture

## Worker

The workers are responsible for managing local resources allocated to Alluxio [21]. These resources could be hard disk, local memory and user configurable. Workers are storing data as block and also respond the requests which are created by clients for read and write data. However, the worker is only responsible for the data in these blocks; the actual mapping from file to blocks is only stored in the master [21].

## Client

The Alluxio client is responsible for user access to interact with the servers. Client communicate with the master for metadata operations and also with workers for read and write operations on data which is present in Alluxio. If data is present in the under storage but not in Alluxio, accessed by using client who are under storage.

# Chapter 3

## Proposed Approach

Currently in Philips propriety technology is used to achieve low latency querying at TB scale, which is expensive. The implementation of proposed architecture will provide alternate means at lower cost. Proposed approach is shown in the figure.

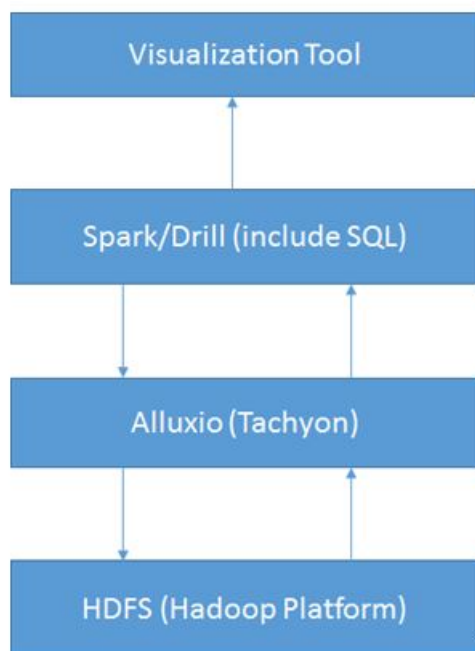


Figure 3.1: Proposed Approach



### 3.1 Plan for theoretical analysis

For the data processing either to use spark sql or apache drill, some research needs to be done. Also the visualization tool is not final yet. According to the requirement we can use one of the already available tools or can build one by our own. This decision is still pending.

### 3.2 Metrics to be used

For the testing purpose we are planning to use any public health dataset.

### 3.3 Proposed Tools to be used

**Data Input:** csv files

**Data Processing:** Spark SQL

**Data caching:** Tachyon (Alluxio)

**Data retrieval:** Apache Drill/Spark

**Data visualization:** Not yet decided

# Chapter 4

## Implementation & Results

### 4.1 Hadoop Installation

#### Step 1:

Create a dedicated system user and then login with root. Then create hadoop group and add user to that group.

```
$ su root
$ sudo su
$ sudo addgroup hadoop (create group hadoop)
$ sudo adduser --ingroup hadoop huser (add huser to hadoop group)
```

Add user to the sudo group.

```
$ usermod aG sudo huser
```

Now give permissions to huser for which add below lines in sudoers file.

```
huser ALL=(ALL)ALL
```

#### Step 2 :

Update the source list and Check if java is installed in your ubuntu (inside VM).

```
$ sudo apt get update  
$ java version
```

**Step 3 :**

Configure SSH

All the communication in HDFS happens over TCP/IP and for the data movement HTTP is used.

To launch the processes on slave nodes, Shell (SSH) is used by Hadoop core. SSH connection needs to be password-less.

Install SSH server using below command.

```
$ sudo apt get install openssh server
```

**Step 4 :**

Generate SSH public/private key for communication. Share this public/private key pair with different hadoop users to authenticate them.

As it is required to unlock the key without any interaction, we will create a RSA key with empty password. In real scenario, empty password is not recommended but here you do not want to type password every time i.e whenever Hadoop node interaction happens.

```
$ ssh keygen
```

**Step 5 :**

Copy public key to authorized key file & edit permissions. Enable SSH access to your local machine with this newly created key.

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

**Step 6 :**

Change permission of file authorized\_key to have all permissions.

```
$ chmod 700 ~/.ssh/authorized_keys
```

**Step 7 :**

Start SSH.

```
$ sudo /etc/init.d/ssh restart
```

**Step 8 :**

Use below command to test the SSH setup.

```
$ ssh localhost
```

Press Enter key. Here it should not ask for any password. If it asks for any password use below command.

```
$ ssh-add
```

**Step 9 :**

Disable IPV6 as Apache Hadoop has only been tested and developed on IPv4 stacks and it is not supported on IPv6 networks. So, Hadoop needs IPv4 to work, and only IPv4 clients can talk to the cluster. Because of this limitation of hadoop, if your organisation moves to IPv6 only, you will encounter problems.

To disable IPv6 add below lines in sysctl.conf file at the end.

```
#Disable IPV6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

```
$ sudo sysctl -p
$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

It should display 1 (It means ipv6 has been successfully disabled). Reboot the system if it is not 1.

**Step 10 :**

Download hadoop from <http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz>

**Step 11 :**

Move this downloaded file to /usr/local/ directory.

```
$ cd /<download_dir>
$ mv hadoop-2.7.3.tar.gz /usr/local/
```

**Step 12 :**

Unzip this file and give permissions to user.

```
$ sudo gunzip Hadoop 2.7.3.tar.gz
$ sudo tar xvzf Hadoop 2.7.3.tar
$ sudo rm -rf Hadoop 2.7.3.tar
$ sudo ln -s Hadoop 2.7.3 hadoop
$ sudo chown R huser:hadoop Hadoop 2.7.3
$ sudo chown R huser:hadoop hadoop
$ sudo chmod 777 hadoop 2.7.3
```

**Step 13 :**

Set java path in hadoop-env.sh

**Step 14 :**

Update the bashrc file with below information

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_PREFIX=/usr/local/hadoop
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export HADOOP_YARN_HOME=${HADOOP_HOME}
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop

export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX}/lib/native
export HADOOP_OPTS="-Djava.library.path=${HADOOP_PREFIX}/lib"

export JAVA_HOME=/usr/local/java
```

Set PATH variable.

```
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$SCALA_HOME/bin:$SPARK_HOME/bin:$M3_HOME/bin
```

Open a new terminal to make all the changes in affect.

### Step 15 :

Create a temp directory used as base location for HDFS.

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown R huser:hadoop /app/hadoop/tmp
$ sudo chmod R 777 /app/hadoop/tmp
```

### Step 16 :

Update yarn-site.xml

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

</configuration>
```

### Step 17 :

Update core-site.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
</property>

<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>

```

**Step 18 :**

Update mapred-site.xml

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>

</configuration>

```

Now create the dirs required for hadoop

```

sudo mkdir -p /usr/local/hadoop/yarn_data/hdfs/namenode
sudo mkdir -p /usr/local/hadoop/yarn_data/hdfs/datanode
sudo chmod 777 /usr/local/hadoop/yarn_data/hdfs/namenode
sudo chmod 777 /usr/local/hadoop/yarn_data/hdfs/datanode
sudo chown -R huser:hadoop /usr/local/hadoop/yarn_data/hdfs/namenode
sudo chown -R huser:hadoop /usr/local/hadoop/yarn_data/hdfs/datanode

```

**Step 19 :**

Update hdfs-site.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>dfs.replication</name>
<value>1</value>
</property>

<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop/yarn_data/hdfs/namenode</value>
</property>

<property>
<name>dfs.datanode.name.dir</name>
<value>file:/usr/local/hadoop/yarn_data/hdfs/datanode</value>
</property>

</configuration>

```

**Step 20 :**

Format the HDFS filesystem via Namenode

```
$ hadoop namenode format
```

**Step 21 :**



To start single node cluster follow below steps.

```
$ cd /usr/local/hadoop/bin
$ sbin/start-dfs.sh
$ sbin/start-yarn.sh
$ jps
```

All the nodes are started as below.

```
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin
17/03/13 12:37:11 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
17/03/13 12:37:11 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
17/03/13 12:37:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
17/03/13 12:37:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
17/03/13 12:37:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache
17/03/13 12:37:11 INFO util.GSet: VM type = 64-bit
17/03/13 12:37:11 INFO util.GSet: 0.0299999999329447746% max memory 966.7 MB = 297.0 KB
17/03/13 12:37:11 INFO util.GSet: capacity = 2^15 = 32768 entries
17/03/13 12:37:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-890586504-127.0.1.1-1489388831850
17/03/13 12:37:12 INFO common.Storage: Storage directory /usr/local/hadoop1/yarn_data/hdfs/namenode has been successfully formatted.
17/03/13 12:37:12 INFO namenode.FSImageFormatProtobuf: saving image file /usr/local/hadoop1/yarn_data/hdfs/namenode/current/fsimage.chkpt_9000000000000000000000 using no compression
17/03/13 12:37:12 INFO namenode.FSImageFormatProtobuf: Image file /usr/local/hadoop1/yarn_data/hdfs/namenode/current/fsimage.chkpt_0000000000000000000000 of size 352 bytes saved in 0 seconds.
17/03/13 12:37:12 INFO namenode.MNStorageRetentionManager: Going to retain 1 images with txid => 0
17/03/13 12:37:12 INFO util.ExitUtil: Exiting with status 0
17/03/13 12:37:12 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at bhumi-VirtualBox/127.0.1.1
*****/
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin$ start-dfs.sh
17/03/13 12:37:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.7.3/logs/hadoop-huser-namenode-bhumi-VirtualBox.out
localhost: datanode running as process 30569. Stop it first.
Starting secondary namenodes [0.0.0.0]
0.0.0.0: secondarynamenode running as process 30764. Stop it first.
17/03/13 12:37:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin$ jps
31298 NameNode
30569 DataNode
30764 SecondaryNameNode
31725 Jps
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop1/logs/yarn-huser-resourcemanager-bhumi-VirtualBox.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.7.3/logs/yarn-huser-nodemanager-bhumi-VirtualBox.out
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin$ jps
31298 NameNode
31782 ResourceManager
30569 DataNode
31980 Jps
30764 SecondaryNameNode
31935 NodeManager
huser@bhumi-VirtualBox: /usr/local/hadoop1/sbin$
```

## 4.2 Spark Installation

Step 1 : Check the Java version.

```
$ java -version
```

Step 2 : Download the Scala using wget.

```
$ sudo mkdir /usr/local/src/scala
$ cd /usr/local/src/scala
$ sudo wget http://www.scala-lang.org/files/archive/scala-2.12.1.tgz
```

Step 3 : Extract the downloaded file.

```
$ sudo tar -xvzf scala-2.12.1.tgz
```

**Step 4 :** Update bashrc file.

```
$ sudo vi .bashrc
export SCALA_HOME=/usr/local/src/scala/scala-2.12.1
export PATH=$SCALA_HOME/bin:$PATH
wq! (Save and quit)
```

**Step 5 :** Restart bashrc.

```
$ . .bashrc
```

**Step 6 :** Verify if scala is installed successfully or not.

```
$ scala -version
Or Type scala (It will go to interactive shell)
```

**Step 7 :** Install git. Spark build depends on git.

```
$ sudo apt-get update
$ sudo apt-get install git
```

**Step 8 :** Download Spark and extract downloaded file.

```
$ sudo wget http://d3kbcqs49mub13.cloudfront.net/spark-1.4.1-bin-hadoop2.6.tgz
$ sudo tar -xvzf spark-1.4.1-bin-hadoop2.6.tgz
```

**Step 9 :** Update bashrc file.

Make sure below lines are there in bashrc file.

```
export SCALA_HOME=/usr/local/scala
export SPARK_HOME=/usr/local/spark
export M3_HOME=/usr/local/sbt
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$SCALA_HOME/bin:$SPARK_HOME/bin:$M3_HOME/bin
```

**Step 10 :** Start all the services of Hadoop

```
$ cd /usr/local/spark/conf
$ sudo cp spark-defaults.conf.template spark-defaults.conf
```

**Step 11 :** Start spark shell

```
$ cd /use/local/spark/conf
$ sudo cp spark-defaults.conf.template spark-defaults.conf
```

Spark is installed successfully

### 4.3 Tachyon Installation

Alluxio can be configured in a variety of modes. The simplest setup for new users is to run Alluxio locally. Below are the steps to install locally. Before installing Alluxio, make sure you have installed Java in your system (JDK 7 or above versions).

**Step 1 :** Download the Alluxio 1.0.0

```
$ wget http://alluxio.org/downloads/files/1.0.0/alluxio-1.0.0-bin.tar.gz
$ tar xvfz alluxio-1.0.0-bin.tar.gz
$ cd alluxio-1.0.0
```

**Step 2 :** Create alluxio-env.sh file in conf folder using alluxio-env.sh.template file.

```
$ cp conf/alluxio-env.sh.template conf/alluxio-env.sh
```

To run Alluxio in standalone mode, update ALLUXIO\_UNDERFS\_ADDRESS in alluxio-env.sh file. Set ALLUXIO\_UNDERFS\_ADDRESS to a tmp directory in the local filesystem (e.g., export ALLUXIO\_UNDERFS\_ADDRESS=/tmp).

**Step 3 :** Format Alluxio first and then start it locally.

```
$ ./bin/alluxio format
$ ./bin/alluxio-start.sh local
```

**Note:** When we start alluxio locally, user will need to enter root password multiple times because Alluxio needs to setup RAMFS. To avoid the need to repeatedly input the root password, you can add the public ssh key for the host into

/.ssh/authorized\_keys.

Verify that Alluxio is running by visiting `http://localhost:19999`.

### 4.3.1 Alluxio Master Web Interface

The Alluxio master serves a web interface to help manage the system. The default port for the Alluxio master web interface is 19999, so the web interface can be viewed by visiting `http://MASTER IP:19999`. For instance, if you started Alluxio locally, the master web interface can be viewed by visiting `localhost:19999`.

The Alluxio master web interface contains several different pages, described below.

#### 4.3.1.1 Home Page

The Alluxio master home page looks something like below:

The screenshot shows the Alluxio Master Web Interface Home Page. The browser address bar displays `localhost:19999/home`. The page features a navigation menu with options: Overview (selected), Browse, Configuration, Workers, In-Memory Data, Logs, Metrics, and Enable Auto-Refresh. The main content is divided into three summary sections:

- Alluxio Summary:**

Master Address:	localhost/127.0.0.1:19998
Started:	03-13-2017 18:40:50:802
Uptime:	2 day(s), 22 hour(s), 5 minute(s), and 4 second(s)
Version:	1.4.0
Running Workers:	0
Startup Consistency Check:	COMPLETE
- Cluster Usage Summary:**

Workers Capacity:	0.00B
Workers Free / Used:	0.00B / 0.00B
UnderFS Capacity:	29.94GB
UnderFS Free / Used:	23.16GB / 6.78GB
- Storage Usage Summary:**

Storage Alias	Space Capacity	Space Used	Space Usage

Home Page includes below sections:

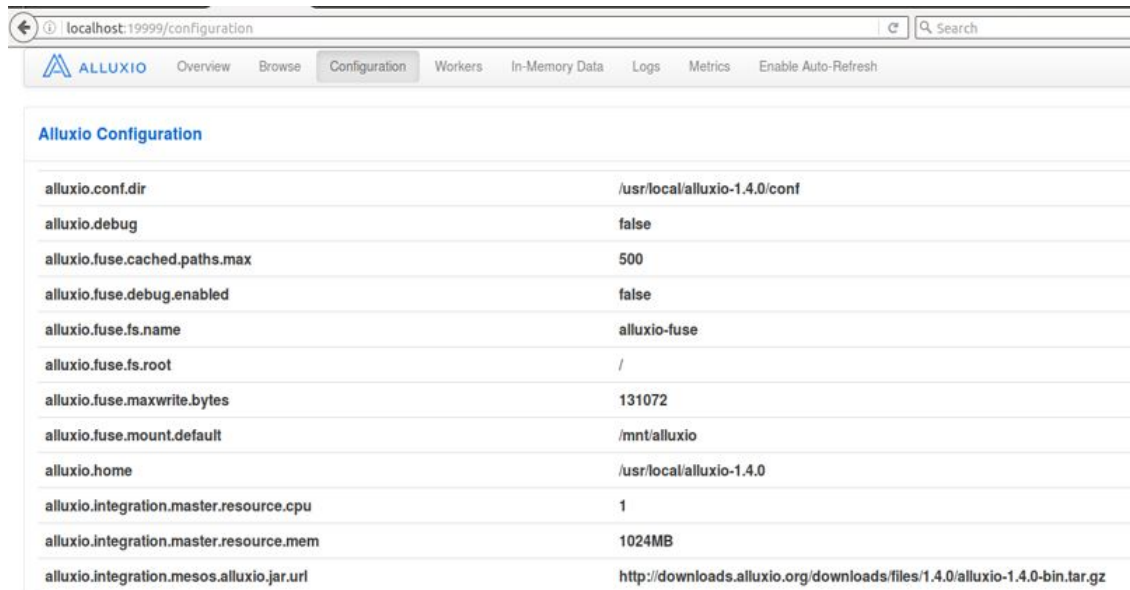
**Alluxio Summary:** Alluxio system level information.

**Cluster Usage Summary :** Alluxio storage information as well as under storage information. Alluxio storage utilization can be near 100%, but under storage utilization should not approach 100%.

**Storage Usage Summary:** Alluxio tiered storage information which gives a breakdown of amount of space used per tier across the Alluxio cluster.

### 4.3.1.2 Configuration Page

System Configuration page will give information about current system configuration.



Alluxio Configuration	
alluxio.conf.dir	/usr/local/alluxio-1.4.0/conf
alluxio.debug	false
alluxio.fuse.cached.paths.max	500
alluxio.fuse.debug.enabled	false
alluxio.fuse.fs.name	alluxio-fuse
alluxio.fuse.fs.root	/
alluxio.fuse.maxwrite.bytes	131072
alluxio.fuse.mount.default	/mnt/alluxio
alluxio.home	/usr/local/alluxio-1.4.0
alluxio.integration.master.resource.cpu	1
alluxio.integration.master.resource.mem	1024MB
alluxio.integration.mesos.alluxio.jar.url	http://downloads.alluxio.org/downloads/files/1.4.0/alluxio-1.4.0-bin.tar.gz

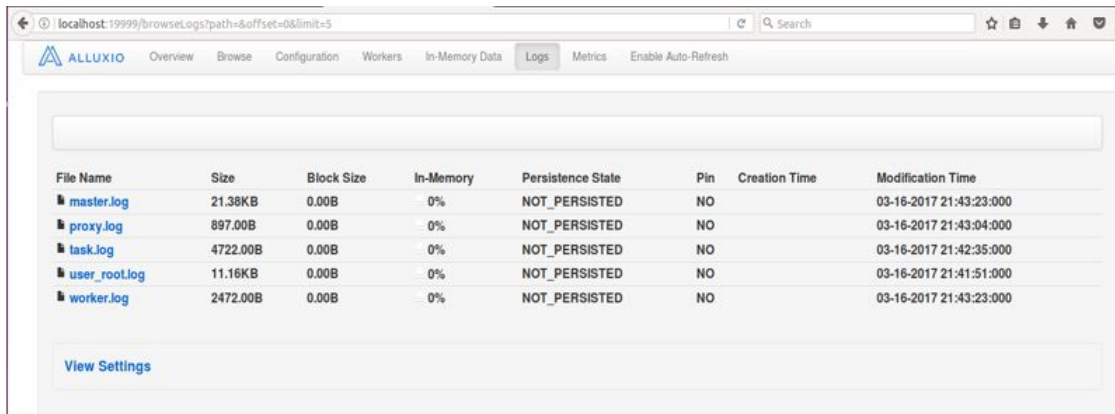
The configuration page has two sections:

**Alluxio Configuration:** A map of all the Alluxio configuration properties and their set values.

**White List:** Contains all the Alluxio path prefixes eligible to be stored in Alluxio. A request may still be made to a file not prefixed by a path in the white list. Only whitelisted files will be stored in Alluxio.

### 4.3.1.3 Log File System Page

You can see the logs for Alluxio file system through the UI. When selecting the "Logs File System" tab in the navigation bar, you will see something like this:



The screenshot shows the Alluxio web interface with the 'Logs' tab selected. A table lists several log files with their respective sizes, block sizes, in-memory usage, persistence states, and timestamps.

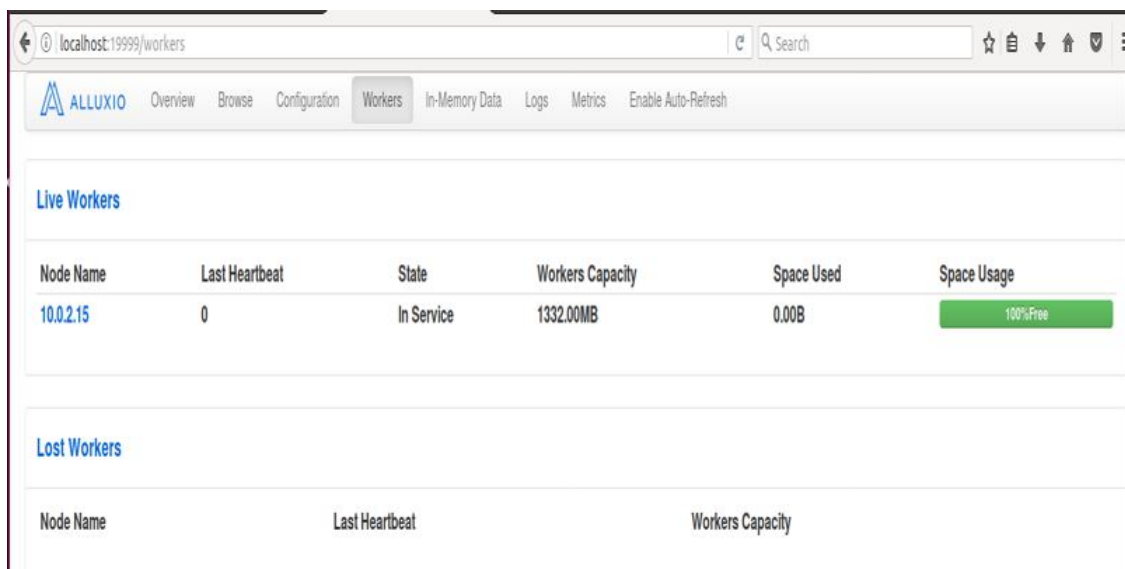
File Name	Size	Block Size	In-Memory	Persistence State	Pin	Creation Time	Modification Time
master.log	21.38KB	0.00B	0%	NOT_PERSISTED	NO	03-16-2017 21:43:23:000	03-16-2017 21:43:23:000
proxy.log	897.00B	0.00B	0%	NOT_PERSISTED	NO	03-16-2017 21:43:04:000	03-16-2017 21:43:04:000
task.log	4722.00B	0.00B	0%	NOT_PERSISTED	NO	03-16-2017 21:42:35:000	03-16-2017 21:42:35:000
user_root.log	11.16KB	0.00B	0%	NOT_PERSISTED	NO	03-16-2017 21:41:51:000	03-16-2017 21:41:51:000
worker.log	2472.00B	0.00B	0%	NOT_PERSISTED	NO	03-16-2017 21:43:23:000	03-16-2017 21:43:23:000

#### 4.3.1.4 Browse In-Memory Files Page

This section lists all the in-memory files with details like name of the file, its size, size for each block, its creation time and modification time.

#### 4.3.1.5 Workers Page

The master also shows all known Alluxio workers in the system and shows them in the "Workers" tab.



The screenshot shows the Alluxio web interface with the 'Workers' tab selected. It displays two sections: 'Live Workers' and 'Lost Workers'.

Node Name	Last Heartbeat	State	Workers Capacity	Space Used	Space Usage
10.0.2.15	0	In Service	1332.00MB	0.00B	100% Free

Node Name	Last Heartbeat	Workers Capacity
-----------	----------------	------------------

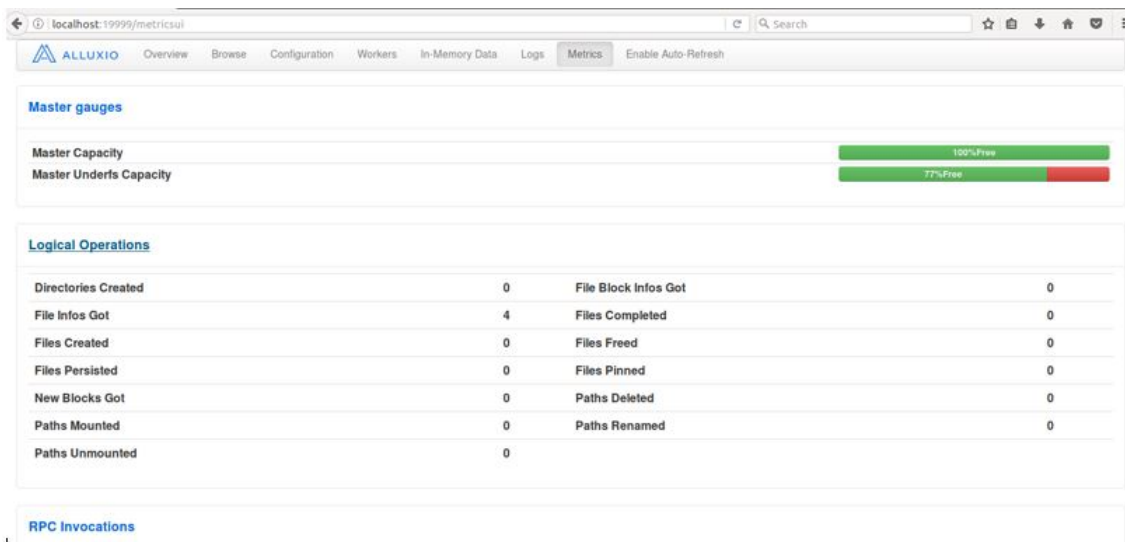
The workers page gives an overview of all Alluxio worker nodes divided into two sections:

**Live Nodes:** A list of all the workers currently serving Alluxio requests. Clicking on the worker name will redirect to the workers web UI.

**Dead Nodes:** A list of all workers proclaimed as dead by the master, usually due to a long timeout waiting for the worker heartbeat. Possible causes include system restart or network failures.

#### 4.3.1.6 Master Metrics

To Access master metrics section, click on the Metrics tab in the navigation bar.



This section shows all master metrics. It includes the following sections:

**Master Gauges:** Overall measures of the master.

**Logical Operation:** Number of operations performed.

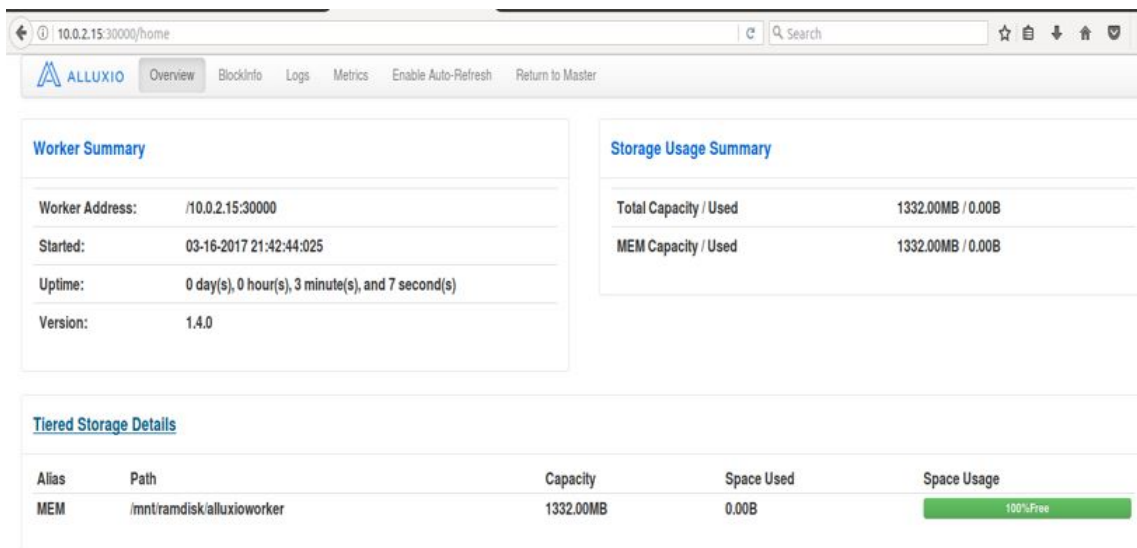
**RPC Invocation:** Number of RPC invocations per operation.

### 4.3.2 Alluxio Workers Web Interface

Each Alluxio worker also serves a web interface to show worker information. The default port for the worker web interface is 30000 so the web interface can be viewed by visiting `http://WORKER IP :30000`. For instance, if you started Alluxio locally, the worker web interface can be viewed by visiting `localhost:30000`

### 4.3.2.1 Home Page

The home page for the Alluxio worker web interface is similar to the home page for the Alluxio master, but shows information specific to a single worker. Therefore, it has similar sections: Worker Summary, Storage Usage Summary, Tiered Storage Details.



The screenshot shows the Alluxio worker web interface. The browser address bar displays '10.0.2.15:30000/home'. The navigation bar includes the Alluxio logo and several tabs: Overview (selected), BlockInfo, Logs, Metrics, Enable Auto-Refresh, and Return to Master. The main content area is divided into three sections:

- Worker Summary:** A table with the following data:
 

Worker Address:	/10.0.2.15:30000
Started:	03-16-2017 21:42:44:025
Uptime:	0 day(s), 0 hour(s), 3 minute(s), and 7 second(s)
Version:	1.4.0
- Storage Usage Summary:** A table with the following data:
 

Total Capacity / Used	1332.00MB / 0.00B
MEM Capacity / Used	1332.00MB / 0.00B
- Tiered Storage Details:** A table with the following data:
 

Alias	Path	Capacity	Space Used	Space Usage
MEM	/mnt/ramdisk/alluxioworker	1332.00MB	0.00B	100% Free

### 4.3.2.2 BlockInfo Page

In the "BlockInfo" page, you can see the files on the worker, and other information such as the file size and which tiers the files is stored on. Also, if you click on a file, you can view all the blocks of that file.

### 4.3.2.3 Worker Metrics

To Access worker metrics section, click on the Metrics tab in the navigation bar.

This section shows all worker metrics. It includes the following sections:

**Worker Gauges:** Overall measures of the worker.

**Logical Operation:** Number of operations performed.



## 4.4 Tachyon(Alluxio) + Spark

### 4.4.1 General setup

Alluxio cluster has been set up previously.

For tachyon with spark configuration, modify spark-defaults.conf file by adding below lines.

```
spark.driver.extraClassPath /pathToAlluxio/core/client/target/Alluxio core client 1.4.0 jar with dependencies.jar
spark.executor.extraClassPath /pathToAlluxio/core/client/target/Alluxio core client 1.4.0 jar with dependencies.jar
```

If Tachyon is running on Hadoop cluster, create core-site.xml in conf folder of spark and add below lines to it.

```
<configuration>
  <property>
    <name>fs.alluxio.impl</name>
    <value>alluxio.hadoop.FileSystem</value>
  </property>
</configuration>
```

If Tachyon is running in fault tolerant mode with zookeeper, add the following lines to previously created core-site.xml file.

```
<property>
  <name>fs.alluxio-ft.impl</name>
  <value>alluxio.hadoop.FaultTolerantFileSystem</value>
</property>
```

Modify spark-defaults.conf by adding below lines to it.

```
spark.driver.extraJavaOptionsDalluxio.zookeeper.address=zookeeperHost1:2181,zookeeperHost2:2181 Dalluxio.zookeeper.enabled=true
spark.executor.extraJavaOptionsDalluxio.zookeeper.address=zookeeperHost1:2181,zookeeperHost2:2181 Dalluxio.zookeeper.enabled=true
```

## 4.4.2 Alluxio as Input and Output

This section explain how to use Alluxio as input and output sources for Spark applications.

### 4.4.2.1 Use Data Already in Alluxio

First, copy local data to the Alluxio file system. Put the file LICENSE into Alluxio, assuming Alluxio is up:

```
bin/alluxio fs copyFromLocal LICENSE /LICENSE
```

Open spark-shell and Run the following commands, assuming Alluxio Master is running on localhost:

```
> val s = sc.textFile("alluxio://localhost:19998/LICENSE")  
> val double = s.map(line => line + line)  
> double.saveAsTextFile("alluxio://localhost:19998/LICENSE2")
```

Go to <http://localhost:19999/browse>. You will find a file named LICENSE2 containing double lines of file LICENSE.

### 4.4.2.2 Use Browse Data in Alluxio

Download any CSV file and put that file in Alluxio using below command, assuming that Alluxio and spark is running up:

```
cd /usr/local/alluxio-1.4.0  
sudo bin/alluxio fs copyFromLocal /home/bhumi/Downloads/newfile.csv /newfile.csv
```

## 4.5 Results

In the first experiment we calculated time required to count number of lines of a file. We did this for different size of text files. As we can see that the spark alone perform better when the file size is small but for larger file size alluxio with spark is better.

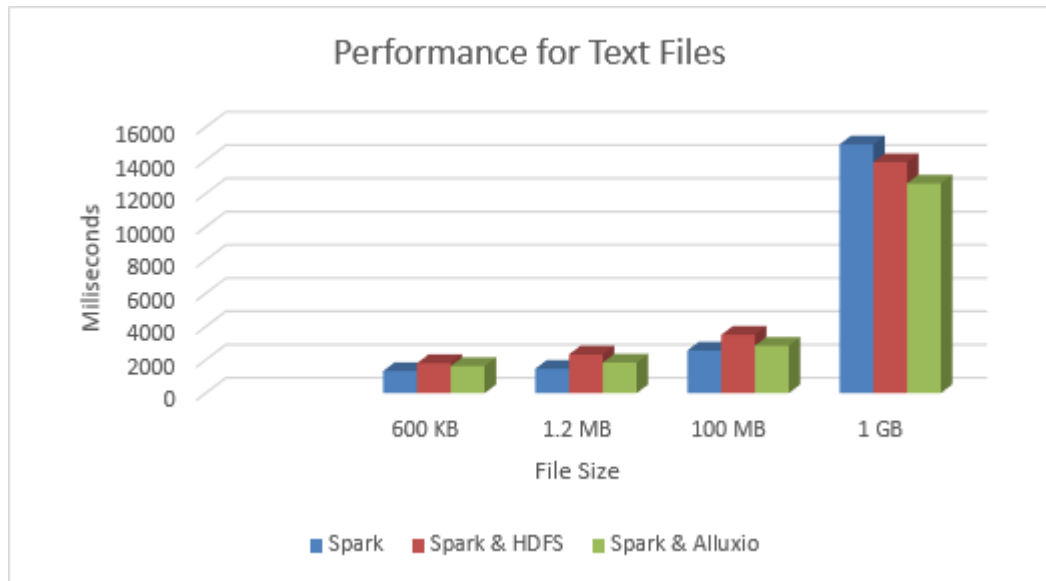


Figure 4.1: Performance Matrix for Text Files

We did the same experiment for the csv files. As we can see that spark and aluuxio combination perform far better than spark and combination of spark and HDFS.

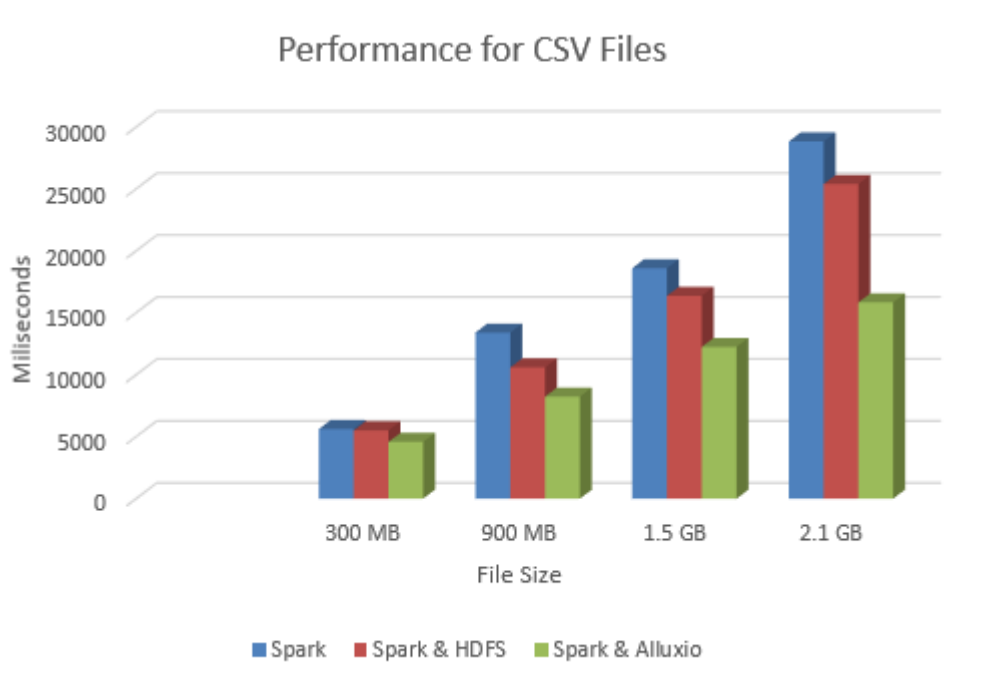


Figure 4.2: Performance Matrix for CSV Files

Results of experiment carried out to decide the alternative are shown in figure

4.1 and 4.2. Even the subsequent file operations will take less time than the first one. This is the big advantage of alluxio. Files that it will load in to the RAM are MRU based i.e most recently used file will be dynamically loaded into RAM.

# Chapter 5

## Conclusion & Future Work

### 5.1 Conclusion

PMT will help hospital personnel in improving their performance and user satisfaction. BI tool which is currently used for PMT is using in chip technology which gives it a speed advantage. This is the new technology and no other tools are using it. The problem with this BI tool is, it is expensive. As an alternative we can use other tools like Tableau or Qlikview but they are also expensive. So If we are targeting for the multinational hospitals we can use those tools but small hospitals won't be able to afford PMT. So, there was a need to identify an architecture for PMT which is cost effective.

Apache Spark can be used in the PMT architecture as it provides in memory processing but it has some performance limitation. After analysis, we came to conclusion that Alluxio which is a memory file system can be used as an alternative for PMT and we don't require any costly BI tools for our application.

### 5.2 Future Work

Currently Alluxio doesn't give any security features. As PMT deals with the hospital data, there is a need of secure environment. So, now we will work towards making

this architecture secure. Including Apache Mesosphere in the architecture is also part of future work as it will be helpful in managing PMT.