Technical University of Berlin (TU Berlin)

Master Thesis

# An Analysis of Side Effects of Hadoop's Optimization in Data Center

*Author:*
Khwaja Zubair Sediqi

*Supervisor:*
Prof. Anja Feldmann
Dr. Marco Canini

*A thesis submitted in fulfilment of the requirements for the degree of Master of Computer Science*

*in the*

Intelligent Network Group(FG INET)
Department of Telecommunication Systems

August 2013

# Declaration of Authorship

I, Khwaja Zubair Sediqi, declare that this thesis titled, 'An Analysis of Side Effects of Hadoop's Optimization in Data Center' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

*"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."*

Dave Barry

TECHNICAL UNIVERSITY OF BERLIN (TU BERLIN)

# *Abstract*

Faculty of Electrical Engineering and Computer Science
Department of Telecommunication Systems

Master of Computer Science

**An Analysis of Side Effects of Hadoop's Optimization in Data Center**

by Khwaja Zubair Sediqi

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

# *Acknowledgements*

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **LAH** | **L**ist **A**bbreviations **H**ere |

# Physical Constants

# Symbols

| | | |
|---|---|---|
| $a$ | distance | m |
| $P$ | power | W (Js$^{-1}$) |
| | | |
| $\omega$ | angular frequency | rads$^{-1}$ |

*For/Dedicated to/To my…*

# Chapter 1

# Related Work

Today's most popular applications are interned based applications such as, social media, e-commerce, etc. For user interact with these applications , various data such as click-stream data,crawled web documents, web requests, logs etc, are generated. As the applications serves millions of users around the globe, so the amount generated data is also huge or so called Big Data. This Big Data is is potential gold mine for the companies to understand access pattern and ad revenue of the company. click-stream data for user actions are the main sources for developers and operators to diagnose problems in production.

The Authors in Google implemented many special-purpose computation paradigm in the past years. The purpose of these special-purpose computation was to process large amount of raw data such as crawled web documents, web requests, logs, etc. The process of large data helps Google to compute various graph of derived data, such as inverted indices, various graph representation of web documents, summaries of number of pages crawled per host, the set of most frequent queries in a data , etc.

## 1.1 Background

### 1.1.1 MapReduce

MapReduce is a programming model and associated implementation for process large data sets in parallel. The mapreduce program reads input key/value pairs and generates output key/value pairs. A mapreduce program consist of Map and Reduce phases. The map and reduce phases can be defined as map and reduce functions written by programmer. The map function read input data as key/value pairs and generates intermediate values.Output of map function si processed by mapreduce platform to The reduce function reads intermediate data generated from map function(s) and merges all values associated with same intermediate key. Mapreduce is designed to run jobs that lasts minutes or hours on dedicated hardware in single data center, with very high bandwidth interconnects.

*** Map Reduce Example, page 21, Hadoop Definitive Guide, 3rd Edition.

## 1.2 Hadoop

Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library[3].The name hadoop is not an acronyme; it is made-up name by kid of Doug Cutting calling his yellow elephant toy hadoop.

Job: MapReduce job is unit of work that need to be processed by nodes. It consist of input data, job configuration information and MapReduce program. To run the jobs, Hadoop divides it into smaller pieces called tasks. There are two types of tasks: map tasks and reduce tasks.

Job execution are controlled by two components of Hadoop called JobTracker and Task-Tracker. Jobtracker is responsible to run all jobs on system. It coordinates job execution by scheduling tasks on tasktracker. The jobtracker maintains record about status of each job. The tasktracker executes tasks on nodes and sends progress report to jobtracker. In case if task execution failed, the jobtracker can rescheduler the task on same or different tasktracker.

Job Creation: Usually the data to be process by Hadoop is very large data set. Hadoop divides this large input data to small fixed-size "input split or split. Each split is called MapReduce job. The splits are fed as input to user defined map function. Map functions read each record of input split and process it. Having many small jobs mean that the execution time for each job is smaller comparing to large input. So if we run the small jobs in parallel on Hadoop cluster, the total time of processing all small jobs will be smaller than total time to process large input data set. [1]

If the job size is very small , then the job creation and map creation time will dominante the over all execution of job. Therefore usually the size of MapReduce jobs are the same as the HDFS block size , which is 64 MB. Such job size is good for Rack Locality Feature of Hadoop.

### 1.2.1 Hadoop Distributed File System (HDFS)

HDFS is the file system component of Hadoop which stores large data sets across cluster of computers in reliable and distributed manner. HDFS is designed to stream data in high bandwidth to user applications. HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware [1] [**?** ].

**Very Large Files** The very large in this context refers to files in size of hundreds of megabytes, gigabytes or terabytes.

**Streaming Data** The efficient data processing idea behind HDFS design was based on write-once and read many times. Usually data set is generated or copied from source and performance evaluation is executed on large proportion, if not all, of data set. Therefore it is the time to read the whole file is more important than reading the first record [1].

**Commodity Computers** HDFS does not require expensive high-available and reliable hardware. It is designed to run over cluster of commonly available hardware from multiple vendors or so called commodity hardware where the failure chances of hardware is very high. For case of hardware, HDFS is designed to continue process and work without noticing the user application from hardware failure [1]. HDFS will not work so well for low latency applications and small file .

**Low-latency data access** As mentioned earlier HDFS was designed for high throughput of data, loading files may lead to delay. Thus, applications that requires low latency in tens of milliseconds, will not work well using HDFS.

**Lots of small data** The inode data and list of blocks belonging to each file is called metadata. The namenode stores the filesystem metadata in random access memory (RAM) [2]. The number of files in namenode is limited by memory size. Each file, directory, and block takes 150 bytes in memory, so, it is feasible to have millions of files, but billions is beyond the capacity of this hardware [1].

### 1.2.2 HDFS Blocks

The minimum amount data or sequence of bytes (or bits) that disk can read or write is called disk block size. The size of disk block is usually 530 Bytes [1]. To read and write data into blocks, filesystem blocks which are in size of kilo bytes are created on top of disk blocks. Generally, the disk blocks are transparent to filesystem. HDFS also has the concept of block; it is default block size is 64 megabyte. The size of HDFS block can be modified to larger number for example 128 megabytes or 512 megabytes.

HDFS breaks the large file into fixed size chunks called HDFS blocks. Each block is stored independently in the cluster. For small file chunks, full capacity of HDFS is not occupied by HDFS. The time to read and write on disk depends on two factors called seek time and data transfer rate of the disk. The time needed to move HEAD (data reader or write component of disk) to the block from where it should read or write is called seek time. The amount of data that disk can read and transfer in second is called disk transfer rate which is usually calculated as megabytes per second. HDFS blocks are large compared to disk block size, this is to reduce the seek time of the disk. For large blocks, the data transfer time is significantly bigger compared to seek time to move to beginning of the block, thus time to transfer multiple files is equal to disk transfer rate. It is not necessary that all blocks of the file to be placed on same single disk. So, an advantage of the block structure is that if a file is larger than available capacity of single disk in the cluster, it can be stored across multiple disks in the cluster.

### 1.2.3 Namenode and Datanode

Hadoop stores meta-data and application data separately. The Meta data is stored in server machine so called Name Node. The user application data is stored on other servers so called Data Node(s). All the servers are connected and communicate with each other using TCP-based protocols. The Data Node does not use mechanism like RAIDi to durable the data, instead stores copy of data across multiple Data Nodes in the cluster to ensure reliability. The advantage of this strategy is that data transfer is multiplied and there are more chances to locate computation near the data (Data Locality) [2].

## 1.3 Hadoop Optimizations

The optimization relevant to hadoop scheduler is covered in this section.As explained in section one of this chapter, hadoop use mechanism to distribute across cluster of computers. Hadoop use scheduler to assign the jobs to datanodes, thus scheduler has key rule in hadoop's performance. Many literature suggested approaches to optimize hadoop scheduler performance.

### 1.3.1 Capacity Scheduler

Though organizations can have their own private compute resources with sufficient capacity to execute jobs on, but such private resources may cost expensive and lead to low utilization of resources.Capacity Scheduler is pluggable MapReduce scheduler that is designed to allow multiple-tenants to share large Hadoop cluster securely and to maximize utilization of the cluster.It is cost effective for the organization to share clusters and run jobs comparing to having their own private cluster.

Using capacity scheduler , the organizations fund the Hadoop cluster collectively, and obtains their share.The available resource in the Hadoop cluster is partitioned and guaranteed minimum share for each organization is provided. In addition, organization can access more then limited share only when the cluster resource is not used by other organization. This mechanism provides elasticity for organization and maximizes the cluster utilization. It means, if other organizations does not use the cluster, then any organization can use full cluster resources(not limited minimum) for its computational jobs as long as their only one organization submitting jobs to the cluster.

As the cluster is shared among organization, this leads to strong cooporation for multi-tenancy to guarantee minimum limit of each organization. To avoid more then limit consume of resources by single job or user that affects other organization's share, capacity scheduler provides safe-guards that limits the user or job access to its share.The capacity scheduler , manages users and jobs in queue structures and resource share is provided to each queue based on their economical share of the cluster. Typically the queues are setup by administrator.

As mentioned in above paragraph, capacity scheduler is pluggable Hadoop mapreduce job scheduler which is available as JAR file in the hadoop tarball under contrib/ capacity-scheduler directory.It is also possible to build scheduler from source by executing ant package, and source package is available under build/contrib/capacity-scheduler

#### Features

#### Capacity Guarantee

Capacity scheduler supports multiple queues,the organization submit jobs to their relevant queue(s). The cluster resources is allocated between all the queues based on their economical share of the cluster. It mean that certain capacity of the cluster at disposal of jobs of its queue. There could be soft or hard limit between queues configured by administrator. Where soft limit means that queue can access more then limited capacity if the others does not use the resource. Hard limit refers to situation where organization can use only their own share of the resource not more.

#### Security

In order to prevent unauthorized user to submit jobs to queue, strict access control lists are applied to each queue. The safe-guards can be used to ensure that users can not view or modify jobs from other users.

**Elasticity**

It is not must for queue of jobs to use only its share portion of resources.Queues can demand beyond their capacity, if there is chance in future point in tasks, then after scheduled tasks executed and completed on resource, these free resources can be allocated to queue which has demand. This mechanism maximizes resource utilization and ensures that resource are available in elastic manner.

**Multi-tenancy**

To ensure that cluster resource is not monopolized by single,job,user and queue limits are provided which ensures that the system or in particular jobTracker is not overwhelmed by too many tasks or jobs.

**Operability**

User and administrators can view current allocation of the queues of the system through console. It is also possible to change queue modification during run time without disruption to users.

**Resource-based Scheduling**

Resource intensive jobs are those that requires or can demand for higher-requirement then default. Capacity Scheduler can accomodate applications or in particular jobs with different resource requirement. Memory is the only resource currently supported by Capacity Scheduler.

**Job Priorities**

Though a running job can not be preempted by any other job, but it is possible to assign higher priority to a job within the queue. Jobs that have higher priority will have access to queue's share of resources faster than jobs with lower priority. By default, priority is disabled in capacity scheduler.[**?** ]

### 1.3.2 Fair-Share Scheduler

Fair-share scheduler is pluggable mapreduce Hadoop scheduler that maintains separate queues for user groups (pools). Resources are allocated to jobs, in a way that on average every job gets fair share of resources over time. If there is single job running, then, it can utilize full resources of the cluster.For new submitted jobs, the slots that become free will be allocated, this mechanism provides opportunity that each job consume on average roughly same amount of resources. Unlike default Hadoop scheduler that maintain queue of jobs, fair-share scheduler lets short jobs to complete in reasonable time and also it does not starve long jobs. [**?** ] [**?** ]

The fair-share scheduler maintains jobs into pools, initial fair distribution of resources across multiple pools are assigned to these pools in order to limit their access to resources.In addition to provision of fair share of resources, fair share scheduler can provide minimum guranteed amount of resources( or computer time of CPU) to each pool to ensure that each user,pool gets sufficient amount of resources. If a pool completed its jobs and does not need the resources,excess resources is evenly distributed among other pools.[**?** ] By default there is one queue per user so that all users can get equally same fraction of total resources.The setup of job pool is possible based on Unix user groups or any other jobconf property.Within each queue, jobs can be scheduled as first-in-first-out (FIFO) or fair share schedule. Fair-share can support job priority, where priority is weight that identifies fraction of total compute time for each job. In addition, inter-queue job priority is also supported by fair-share scheduler.[**?** ]

**Task Preeption**

In case if minimum share of a pool is not provided, after waiting for certain period of time, the scheduler may kill a task from other pool(s) to provide minimum share to pool.Killing task of other jobs is called task preemption.It is also possible that preemption happen if a pool is below its half share for configurable timeout period.Usually timeout value is higher than minimum share timeout preemption.[**?** ] [**?** ]

In both cases of above preemption , the fair share scheduler kills most-recently-launched tasks from over-allocated jobs, to minimize wasted computation.Since Hadoop jobs are tolerated to losing tasks, killing tasks does not cause jobs to fail but causes them to take longer to finish.[**?** ][**?** ]

### 1.3.3 Speculative Execution

The goal of speculative execution is to reduce the job completion time by speculating the tasks from straggler machines. The tasks are categorized into below 3 categories. If there is free slot on a node, then a task is selected according to the category number from one these categories.

1 - Failed Tasks: If a task fails multiple time , due to a bug and stop the job, such task is marked as "failed task" and given highest priority.

2 - Non-Running Tasks:These are fresh tasks that has not being executed on any node yet. For maps, data-locality is considered and tasks that are closer to the node is performed first.

3 - Speculative Tasks:To find speculative task,the progress of task is monitored by hadoop with a progress score between 0 and 1.Map progress depends on input data read and its progress score is fraction of input read data. The reduce phase compromise three sub phases where each sub-phase is counted as 1/3 of progress report. The three sub-phase of reduce phase is explained as a,b,c bullet points bellow.

a - Fetching of map outputs, also called copy phase. b - Sorting of map outputs by key, also called sort phase. c - Applying user-defined function to the list of map outputs with each key, also called reduce phase.

In each sub-phase of reduce, the score is fraction of data processed. For example, a task halfway through the copy phase has a progress score of 1/2 * 1/3 = 1/6 , while a task halfway through the reduce phase scores 1/3 + 1/3 +(1/2 * 1/3)= 5/6.

**Straggler**

For map tasks and reduce tasks average of their progress score is defined as threshold by hadoop.If progress score for a task is less then the the threshold of its category (maps or reduces) minus 0.2, and it has run for at least 2 minutes , it is marked as straggler. All the tasks below,beyond? the threshold are considered as slow and the scheduler runs at least one speculative copy of these tasks at time.

**source:Improving MapReduce Performance in Heterogeneous Environments Matei Zaharia, page:31 on the document**

### 1.3.4 LATE Scheduler

"Hadoops scheduler can cause severe performance degradation in heterogeneous environments. We design a new scheduling algorithm, Longest Approximate Time to End (LATE), that is highly robust to heterogeneity. LATE can improve Hadoop response times by a factor of 2 in clusters of 200 virtual machines on EC2"[source:Improving mapreduce ...]. LATE is another speculative task scheduler that behave well in real environment[source:Improving mapreduce ...].The task's finish time is estimated and for those tasks that are believed to finish farthest time into the future are speculatively executed. Because the speculative copy are execute in faster nodes so they have better chance to overtake the original copy and reduce job response time. The heuristic to estimate time left for task completion is as follow: The ıprogress rate for each task is estimated as ProgressScore/T, where T is the amount of time task has been running for. Based on ıprogress rate the time to completion is estimated as (1 - ProgressScore/ProgressRate). The assumption in this estimation is that task has the same progress rate. The execution of speculative tasks on fast nodes provides better opportunity that a speculative copy of task overtake the original task. Nodes that are below SlowNodeThreshold are considered as fast nodes and speculative tasks can only be executed on fast nodes no stragglers.Sum of progress scores for all succeded and in progress tasks on the cluster is estimated as speed, and SlowNodeThreshold is defined as percentile of speed below which node will be considered as slow node.This avoids unnecessary speculation when only fast tasks are running. The total number of speculative tasks that can be executed in one time, is defined as ıSpeculativeCap.If less then SpeculativeCap speculative is running, and an free node asks for task then the task assignment decision is made as follow: - If node is slow(total progress is lower than SlowNodeThreshold) then ignore the request. - Estimate the time left for running tasks that are not speculated and rank them for speculative execution. - Run the copy of task with highest rank (lowest progress rate comparing to SlowTaskThreshold).

"In practice, we have found that a good choice for the three parameters to LATE are to set the SpeculativeCap to 10% percent of available task slots and set the SlowNodeThreshold and SlowTaskThreshold to the 25th percentile of node progress and task progress rates respectively [source matie zahrie]."

### 1.3.5 Advantages of LATE

The native Hadoop scheduler mechanism is to consider any task that is below the fixed threshold as slow task and treat them equally for speculative execution. While, LATE relaunches the slowest task and small number(at maximum as SpeculativeCap) of tasks to limit contention for shared resources.LATE mechanism is to prioritize among slow tasks on how much they hurt job response time and rank them for speculation priority. Hadoop native scheduler assumes that nodes are homogeneous and any candidate node for task execution is likely to be a fast node. In contrast, LATE takes into account node heterogeneity by ranking some nodes as slow node(nodes below SlowNodeThreshold are marked as slow node)and assigns new tasks only to fast nodes. Hadoop native scheduler focuses on progress rate and speculativly executes any slow task.LATE focuses on estimated time left and speculatively executes only tasks that will improve job response time. For example,if task A is 5x slower than the mean but has 90 percent progress, and task B is 2x slower than the mean but is only at 10 percent progress, then task B will be chosen for speculation first, even though it is has a higher progress rate, because it hurts the response time more. Therefore , LATE provides opportunity for slow nodes to be utilized as long as this does not hurt job response time which is unlike of progress rate base scheduler that always re-executes task from slow nodes.

### 1.3.6 References

The '`natbib`' package is used to format the bibliography and inserts references such as this one [? ]. The options used in the '`Thesis.tex`' file mean that the references are listed in numerical order as they appear in the text. Multiple references are rearranged in numerical order (e.g. [? ? ]) and multiple, sequential references become reformatted to a reference range (e.g. [? ? ? ]). This is done automatically for you. To see how you use references, have a look at the '`Chapter1.tex`' source file. Many reference managers allow you to simply drag the reference into the document as you type.

Scientific references should come *before* the punctuation mark if there is one (such as a comma or period). The same goes for footnotes[1]. You can change this but the most important thing is to keep the convention consistent throughout the thesis. Footnotes themselves should be full, descriptive sentences (beginning with a capital letter and ending with a full stop).

To see how LaTeX typesets the bibliography, have a look at the very end of this document (or just click on the reference number links).

### 1.3.7 Figures

There will hopefully be many figures in your thesis (that should be placed in the 'Figures' folder). The way to insert figures into your thesis is to use a code template like this:

```
\begin{figure}[htbp]
  \centering
    \includegraphics{./Figures/Electron.pdf}
    \rule{35em}{0.5pt}
  \caption[An Electron]{An electron (artist's impression).}
  \label{fig:Electron}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the electron that you can see in the figure below.

[1]Such as this footnote, here down at the bottom of the page.

FIGURE 1.1: An electron (artist's impression).

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have labels just in case you need to refer to them (such as in Figure 1.1). The '`\caption`' command contains two parts, the first part, inside the square brackets is the title that will appear in the 'List of Figures', and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The '`\rule`' command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

The LaTeX Thesis Template is able to use figures that are either in the PDF or JPEG file format.

### 1.3.8 Typesetting mathematics

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The "Not So Short Introduction to LaTeX" (available here) should tell you everything you need to know for most cases of typesetting mathematics. If you need more information, a much more thorough mathematical guide is available from the AMS called, "A Short Math Guide to LaTeX" and can be downloaded from:
`ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

There are many different LaTeX symbols to remember, luckily you can find the most common symbols here. You can use the web page as a quick reference or crib sheet and because the symbols are grouped and rendered as high quality images (each with a downloadable PDF), finding the symbol you need is quick and easy.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
  \label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{1.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, just put the command, '`\nonumber`' immediately after the equation.

## 1.4 Sectioning and Subsectioning

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the '`\chapter{}`', '`\section{}`' and '`\subsection{}`' commands you write in the source.

The table of Contents should only list the sections to three (3) levels. A '`\chapter{}`' is level one (1). A '`\section{}`' is level two (2) and so a '`\subsection{}`' is level three

(3). In your thesis it is likely that you will even use a '`\subsubsection{}`', which is level four (4). Adding all these will create an unnecessarily cluttered table of Contents and so you should use the '`\subsubsection*{}`' command instead (note the asterisk). The asterisk (*) tells LaTeX to omit listing the subsubsection in the Contents, keeping it clean and tidy.

## 1.5 In Closing

You have reached the end of this mini-guide. You can now rename or overwrite this pdf file and begin writing your own '`Chapter1.tex`' and the rest of your thesis. The easy work of setting up the structure and framework has been taken care of for you. It's now your job to fill it out!

Good luck and have lots of fun!

Guide written by —
Sunil Patel: www.sunilpatel.co.uk

# Bibliography

[1] Tom White. *Hadoop:The Definitive Guide*, volume Third Edition. May 2012. URL `http://link.aip.org/link/?RSI/72/4477/1`.