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

Speed of Light	c	$=$	$2.997\ 924\ 58 \times 10^{8}\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

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

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Introduction

Today's most popular applications are interned based applications such as, social media, e-commerce, etc. For user interacts 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 the 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.

1.2 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.

1.3 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 TaskTracker. 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.

1.3.1 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 [2] .

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].

1.3.2 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.

1.3.3 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].

1.3.4 Folders

This template comes as a single Zip file that expands out to many files and folders. The folder names are mostly self-explanatory:

Appendices – this is the folder where you put the appendices. Each appendix should go into its own separate '`.tex`' file. A template is included in the directory.

Chapters – this is the folder where you put the thesis chapters. A thesis usually has about seven chapters, though there is no hard rule on this. Each chapter should go in its own separate '`.tex`' file and they usually are split as:

- Chapter 1: Introduction to the thesis topic
- Chapter 2: Background information and theory

- Chapter 3: (Laboratory) experimental setup
- Chapter 4: Details of experiment 1
- Chapter 5: Details of experiment 2
- Chapter 6: Discussion of the experimental results
- Chapter 7: Conclusion and future directions

This chapter layout is specialised for the experimental sciences.

Figures – this folder contains all figures for the thesis. These are the final images that will go into the thesis document.

Primitives – this is the folder that contains scraps, particularly because one final image in the 'Figures' folder may be made from many separate images and photos, these source images go here. This keeps the intermediate files separate from the final thesis figures.

1.3.5 Files

Included are also several files, most of them are plain text and you can see their contents in a text editor. Luckily, many of them are auxiliary files created by LaTeX or BibTeX and which you don't need to bother about:

Bibliography.bib – this is an important file that contains all the bibliographic information and references that you will be citing in the thesis for use with BibTeX. You can write it manually, but there are reference manager programs available that will create and manage it for you. Bibliographies in LaTeX are a large subject and you may need to read about BibTeX before starting with this.

Thesis.cls – this is an important file. It is the style file that tells LaTeX how to format the thesis. You will also need to open this file in a text editor and fill in your own information (such as name, department, institution). Luckily, this is not too difficult and is explained in section 1.4 on page 7.

Thesis.pdf – this is your beautifully typeset thesis (in the PDF file format) created by LaTeX.

Thesis.tex – this is an important file. This is the file that you tell LaTeX to compile to produce your thesis as a PDF file. It contains the framework and constructs that tell LaTeX how to layout the thesis. It is heavily commented so you can read exactly what each line of code does and why it is there. After you put your own information into the '`Thesis.cls`' file, go to this file and begin filling it in – you have now started your thesis!

vector.sty – this is a LaTeX package, it tells LaTeX how to typeset mathematical vectors. Using this package is very easy and you can read the documentation on the site (you just need to look at the '`vector.pdf`' file):
`http://www.ctan.org/tex-archive/macros/latex/contrib/vector/`

lstpatch.sty – this is a LaTeX package required by this LaTeX template and is included as not all TeX distributions have it installed by default. You do not need to modify this file.

Files that are *not* included, but are created by LaTeX as auxiliary files include:

Thesis.aux – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main '`.tex`' file.

Thesis.bbl – this is an auxiliary file generated by BibTeX, if it is deleted, BibTeX simply regenerates it when you run the main tex file. Whereas the '`.bib`' file contains all the references you have, this '`.bbl`' file contains the references you have actually cited in the thesis and is used to build the bibliography section of the thesis.

Thesis.blg – this is an auxiliary file generated by BibTeX, if it is deleted BibTeX simply regenerates it when you run the main '`.tex`' file.

Thesis.lof – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main '`.tex`' file. It tells LaTeX how to build the 'List of Figures' section.

Thesis.log – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main '`.tex`' file. It contains messages from LaTeX, if you receive errors and warnings from LaTeX, they will be in this '`.log`' file.

Thesis.lot – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main '`.tex`' file. It tells LaTeX how to build the 'List of Tables' section.

Thesis.out – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main '`.tex`' file.

So from this long list, only the files with the '`.sty`', '`.bib`', '`.cls`' and '`.tex`' extensions are the most important ones. The other auxiliary files can be ignored or deleted as LaTeX and BibTeX will regenerate them.

1.4 Filling in the ‘`Thesis.cls`’ File

You will need to personalise the thesis template and make it your own by filling in your own information. This is done by editing the ‘`Thesis.cls`’ file in a text editor.

Open the file and scroll down, past all the ‘`\newcommand...`’ items until you see the entries for ‘`University Name`’, ‘`Department Name`’, etc....

Fill out the information about your group and institution and ensure you keep to block capitals where it asks you to. You can also insert web links, if you do, make sure you use the full URL, including the ‘`http://`’ for this.

The last item you should need to fill in is the Faculty Name (in block capitals). When you have done this, save the file and recompile ‘`Thesis.tex`’. All the information you filled in should now be in the PDF, complete with web links. You can now begin your thesis proper!

1.5 The ‘`Thesis.tex`’ File Explained

The `Thesis.tex` file contains the structure of the thesis. There are plenty of written comments that explain what pages, sections and formatting the LaTeX code is creating. Initially there seems to be a lot of LaTeX code, but this is all formatting, and it has all been taken care of so you don’t have to do it.

Begin by checking that your information on the title page is correct. For the thesis declaration, your institution may insist on something different than the text given. If this is the case, just replace what you see with what is required.

Then comes a page which contains a funny quote. You can put your own, or quote your favourite scientist, author, person, etc... Make sure to put the name of the person who you took the quote from.

Next comes the acknowledgements. On this page, write about all the people who you wish to thank (not forgetting parents, partners and your advisor/supervisor).

The contents pages, list of figures and tables are all taken care of for you and do not need to be manually created or edited. The next set of pages are optional and can be deleted since they are for a more technical thesis: insert a list of abbreviations you have used in the thesis, then a list of the physical constants and numbers you refer to and finally, a list of mathematical symbols used in any formulae. Making the effort to fill these tables means the reader has a one-stop place to refer to instead of searching the internet and references to try and find out what you meant by certain abbreviations or symbols.

The list of symbols is split into the Roman and Greek alphabets. Whereas the abbreviations and symbols ought to be listed in alphabetical order (and this is *not* done automatically for you) the list of physical constants should be grouped into similar themes.

The next page contains a one line dedication. Who will you dedicate your thesis to?

Finally, there is the section where the chapters are included. Uncomment the lines (delete the '`%`' character) as you write the chapters. Each chapter should be written in its own file and put into the 'Chapters' folder and named '`Chapter1`', '`Chapter2`, etc... Similarly for the appendices, uncomment the lines as you need them. Each appendix should go into its own file and placed in the 'Appendices' folder.

After the preamble, chapters and appendices finally comes the bibliography. The bibliography style (called '`unsrtnat`') is used for the bibliography and is a fully featured style that will even include links to where the referenced paper can be found online. Do not under estimate how grateful you reader will be to find that a reference to a paper is just a click away. Of course, this relies on you putting the URL information into the BibTeX file in the first place.

1.6 Thesis Features and Conventions

To get the best out of this template, there are a few conventions that you may want to follow.

One of the most important (and most difficult) things to keep track of in such a long document as a thesis is consistency. Using certain conventions and ways of doing things (such as using a Todo list) makes the job easier. Of course, all of these are optional and you can adopt your own method.

1.6.1 Printing Format

This thesis template is designed for single sided printing as most theses are printed and bound this way. This means that the left margin is always wider than the right (for binding). Four out of five people will now judge the margins by eye and think, "I never noticed that before.".

The headers for the pages contain the page number on the right side (so it is easy to flick through to the page you want) and the chapter name on the left side.

The text is set to 11 point and a line spacing of 1.3. Generally, it is much more readable to have a smaller text size and wider gap between the lines than it is to have a larger text size and smaller gap. Again, you can tune the text size and spacing should you want or need to. The text size can be set in the options for the '`\documentclass`' command at the top of the '`Thesis.tex`' file and the spacing can be changed by setting a different value in the '`\setstretch`' commands (scattered throughout the '`Thesis.tex`' file).

1.6.2 Using US Letter Paper

The paper size used in the template is A4, which is a common – if not standard – size in Europe. If you are using this thesis template elsewhere and particularly in the United States, then you may have to change the A4 paper size to the US Letter size. Unfortunately, this is not as simple as replacing instances of '`a4paper`' with '`letterpaper`'.

This is because the final PDF file is created directly from the LaTeX source using a program called '`pdfTeX`' and in certain conditions, paper size commands are ignored and all documents are created with the paper size set to the size stated in the configuration file for pdfTeX (called '`pdftex.cfg`').

What needs to be done is to change the paper size in the configuration file for `pdfTeX` to reflect the letter size. There is an excellent tutorial on how to do this here:
`http://www.physics.wm.edu/~norman/latexhints/pdf_papersize.html`

It may be sufficient just to replace the dimensions of the A4 paper size with the US Letter size in the `pdftex.cfg` file. Due to the differences in the paper size, the resulting margins may be different to what you like or require (as it is common for Institutions to dictate certain margin sizes). If this is the case, then the margin sizes can be tweaked by opening up the `Thesis.cls` file and searching for the line beginning with, '`\setmarginsrb`' (not very far down from the top), there you will see the margins specified. Simply change those values to what you need (or what looks good) and save. Now your document should be set up for US Letter paper size with suitable margins.

1.6.3 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).

1.6.4 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.

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 2.1). 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

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

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

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.

1.6.5 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.

1.7 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.

1.8 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

Chapter 2

Related Work

2.1 Background

Today's most popular applications are interned based applications such as, social media, e-commerce, etc. For user interacts 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 the 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.

2.1.1 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.

2.2 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 TaskTracker. 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.

2.2.1 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 [2] .

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].

2.2.2 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.

2.2.3 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].

2.3 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.

2.3.1 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

2.3.2 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]."

2.3.3 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.

2.3.4 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).

2.3.5 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}
```

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

```
\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.

FIGURE 2.1: 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 2.1). 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.

2.3.6 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:

$$E = mc^2 \tag{2.1}$$

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.

2.4 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.

2.5 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

Chapter 3

Experimental Environment

3.1 Introduction

All the experiments are tested in simulated environment.The environment consist of physical resources,virtual operating system,hadoop software,open stack and tools/scripts to process and analyse the log data. To run the experiments , a set of Ubuntu 12.10 Virtual machine is spawned and installed on top of physical computers. Then, the process of hadoop-snap-shot-3 installation master and tata node configuration happens. After completion of hadoop setup, using terasort workload is generated and stored on datanodes.The masternode does not act as data node in our experiments. Once, the workload generation is completed, the terasort starts process the data using hadoop's default capacity scheduler. The openstack software is used to manage the VMs. I also use scripts to process and analyze the logs(see appendix).

3.1.1 Physical Resources

A total number of seven(7) computer machines connected through central switch is used to run the experiments.All the computers are connected using Gigabit ethernet port to the switch. Each computer has sixteen(16)GB of RAM(Random Access Memory).The computers are equipped witch eight(8) CPU(Central Processing Unit), where the speed of each CPU is aproximately 2,3 GHZ.The system uses 10 GB of disk space to store the virtual machine and hadoop software. Additional mounted hard disk space of seventy two(72) GB is provided as NFS(Network File System) storage to each computer.

3.1.2 Terasort

Teragen - Generates the random data that is used as input data for terasort.The data is generated in rows and the format of row is ”¡10 bytes key¿¡10 bytes rowid¿¡78 bytes filler¿”. The keys are random characters from the set .. , rowid is justified row id as a int and the filler consists of 7 runs of 10 characters from A to Z.Teragen divides the number of rows by the desired number of tasks and assigns set of rows to each map.**??**

TeraSort - It is implemented as a MapReduce sort job with a custom partitioner that uses a sorted list of n-1 sampled keys that define the key range for each reduce.

3.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Chapter 4

Chapter Title Here

4.1 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

4.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

4.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

4.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Appendix A

Appendix Title Here

Write your Appendix content here.