Byzantine Fault-Tolerant and Locality-Aware Scheduling MapReduce

Andrea Graziani
andrea.graziani93@outlook.it
Università Degli Studi di Roma Tor Vergata
Rome, Italy

ABSTRACT

MapReduce is often used to run critical jobs such as scientific data analysis. However, evidence in the literature shows that arbitrary faults do occur and can probably corrupt the results of MapReduce jobs; Moreover, ignoring data locality during task scheduling can lead to performance degradation and a pointless bigger network traffic.

We present an original MapReduce algorithm capable to tolerate arbitrary or Byzantine faults experienced by worker nodes and to resolve master node single point of failure problem; moreover, recognizing input data network locations and sizes, our algorithm performs a locality aware task scheduling, improving performance and diminishing network traffic.

Although the execution of a job with our algorithms uses more resources respect to other implementations, like Hadoop, we believe that this cost is acceptable for critical applications that require that level of fault tolerance.

KEYWORDS

MapReduce, Fault tolerance, Arbitrary failure, Data locality

ACM Reference Format:
Andrea Graziani. 2019. Byzantine Fault-Tolerant and Locality-Aware Scheduling MapReduce. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Various data-intensive tasks, like seismic simulation, natural language processing, machine learning, astronomical data parsing, web data mining and many other, require a processing power that exceeds the capabilities of individual computers; this fact imposes the use of *distributed computing*. Nowadays many famous distributed applications use thousands of computers and hundreds of other devices like network switches, routers and power units in order to provide their services to an increasing number of users in every part of the world, moving consequently an huge amount of data between computers and server. *MapReduce*, a framework developed by Google, represents a solution for processing large data sets in a distributed environment.

However, as many studies confirm, *hardware component failures are frequent* and they will probably happen more often in the future owing to the increasing number of computer and server connected to internet. Is been documented that in the first year of a cluster at Google there were 1000 individual machine failures and thousands of hard drive failures. A recent study of DRAM errors in a large number of servers in Google data-centres for 2.5 years concluded that these errors are more prevalent than previously believed, with more than 8% DIMM affected by errors yearly, even if protected by error correcting codes (ECC) [?]. A Microsoft study of 1 million consumer PCs shown that CPU and core chipset faults are also frequent. [?] Moreover moving large amount of data repeatedly to distant nodes is becoming the bottleneck owing to an increased network traffic causing performance degradation.

These are the reasons why to construct a distributed system in such a way it can provide its services even in the presence of failures is become so critical; consequently, to provide a *fault tolerant* cloud application represents an important goal in distributed-systems design. Moreover exploiting data locality, in order to mitigate network traffic and delay, becomes very important to improve performance.

Then the goal of this paper is to describe an *Arbitrary Fault-Tolerant Locality-Aware* (AFTLA) *MapReduce runtime system* capable to mitigate problems described above.

, ,
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 ARBITRARY FAULT-TOLERANT LOCALITY-AWARE MAPREDUCE

Arbitrary fault tolerance

As known academic literature describes many type of failure, like *crash failures*; however the most serious are known as *arbitrary failures* or *Byzantine failures*, according to which a server may produce arbitrary responses at arbitrary times which cannot be detected as being incorrect.

Redundancy represents the key technique used to manage these kind of failure, according to which,

The key approach to tolerating a faulty process is to organize several identical processes into a group.

Our BFT MapReduce follows the approach of executing each task more than once, similarly to the works mentioned above. The chal-

Process groups are part of the solution for building fault-tolerant systems. In particular, having a group of identical processes allows us to mask one or more faulty processes in that group. In other words, we can replicate processes and organize them into a group to replace a single (vulnerable) process with a (fault tolerant) group.

An important issue with using process groups to tolerate faults is how much replication is needed. To simplify our discussion, let us consider only replicated-write systems. A system is said to be k-fault tolerant if it can survive faults in k components and still meet its specifications. If the components, say processes, fail silently, then having k + 1 of them is enough to provide k-fault tolerance. If k of them simply stop, then the answer from the other one can be used.

On the other hand, if processes exhibit arbitrary failures, continuing to run when faulty and sending out erroneous or random replies, a minimum of 2k + 1 processes are needed to achieve k-fault tolerance. In the worst case, the k failing processes could accidentally (or even intentionally) generate the same reply. However, the remaining k + 1 will also produce the same answer, so the client or voter can just believe the majority.

3 SYSTEM ARCHITECTURE

In order to properly describe our MapReduce algorithm, including our arbitrary failures management system and how we exploit data locality in order to improve performance, it is necessary to describe our system architecture.

Assumptions

Our system is composed by a set of distributed processes, every of which run on *different* hosts.

We assume that our system runs *asynchronously*, that is no assumptions about process execution speeds or message delivery times are made; therefore we can normally use timeouts to conclude that a process has crashed but, occasionally, such conclusion is false. However, all processes are connected by *reliable channels*, so no messages are lost, duplicated or corrupted; that feature is guaranteed by the use of TCP connections.

Clients are always correct, because if they are not there is no point in worrying about the correctness of our system's output.

Finally we assume the existence of an hash function that is *collisions-resistant*, for which it is infeasible to find two inputs that produce the same output.

Our system is composed by a set of distributed processes:

Client Process the clients that request the execution of jobs composed by map and reduce tasks

Leader Primary Process It manages the execution of word-count jobs received from clients coordinating Worker Nodes

Backup Primary Process It manages the execution of word-count jobs received from clients coordinating Worker Nodes

Worker Process A Worker Process executes map and reduce task scheduleted by current Leader Primary Process. In order to achieve fault tolerance, any Worker Process must be run indepently on different host. In our implementation, each process run on independent Amazon EC2 server

Worker Group All system's nodes in which worker process are running are logically split into several *Groups*, that is sets of equal worker processes, each of which execute the same commands using same input data in the same order. In a group all worker processes run independently on different host and they do not interact with each other in any way. Current Leader Primary Process can interact with groups members using a push-based approach in order to schedule map or reduce tasks. Although, for performance reasons, not always happen, when a task is sent to the group itself, all members of the group receive it.
The key property that all groups have is that when a message is sent to the group itself, all members of the group receive it.
primary coordinates all write operations
In other words, we can replicate processes and organize them into a group to replace a single (vulnerable) process with a (fault tolerant) group.
When a task is for work is generated, either by an external client or by one of the workers, it is sent to the coordinator.
as a set of Task- Trackers that execute tasks

We assume that clients are always correct, because if they are not there is no point in worrying about the correctness of the job's output.

The host where current master process is running may fail, for example by crashing or by losing network connectivity. Therefore, to ensure system availability in such a way that it can keep on providing its services, multiple master process copies of run on different host, and that a backup copy is promoted to become the new leader when the previous leader fails.

In order to decide when the current leader master process has failed and to elect a new leader, we have used a coordination service like Apache ZooKeeper.

The Mesos master stores information about the active tasks and registered frameworks in memory: it does not persist it to disk or attempt to ensure that this information is preserved after a master failover. This helps the Mesos master scale to large clusters with many tasks and frameworks. A downside of this design is that after a failure, more work is required to recover the lost in-memory master state.

Mesos consists of a master daemon that manages agent daemons running on each cluster node, and Mesos frameworks that run tasks on these agents.

The Mesos master stores information about the active tasks and registered frameworks in memory: it does not persist it to disk or attempt to ensure that this information is preserved after a master failover. This helps the Mesos master scale to large clusters with many tasks and frameworks. A downside of this design is that after a failure, more work is required to recover the lost in-memory master state.

System model. The system is composed by a set of distributed processes: the clients that request the execution of jobs composed by map and reduce tasks, the JobTracker that manages the execution of a job as explained, a set of TaskTrackers that execute tasks, the NameNode that manages access to data stored in HDFS, and a set of DataNodes where HDFS stores file blocks. We say that a process is correct if it follows the algorithm, otherwise we say it is faulty. We also use these two words to denominate a task (map or re- duce) that, respectively, returns the result that corresponds to an execution in a correct TaskTracker (correct) or not (faulty). Processes run in servers in a datacenter.

The algorithm

In order to achieve A simplistic solution to make MapReduce Byzantine fault-tolerant given the system model would be the following. First, the JobTracker starts 2f + 1 replicas of each map task in different servers and TaskTrackers. Second, the JobTracker starts also 2f + 1 replicas of each reduce task. Each reduce task fetches the output from all map replicas, picks the most voted results, processes them and stores its output in HDFS. In the end, either the client or a special task must make the vote of the outputs to pick the most voted. An even more simplistic solution would be to run a consensus, or Byzantine agreement between each set of map task replicas and reduce task replicas. This would involve even more replicas (typically 3f + 1) and more messages exchanged.

Crash failure detection

Workers nodes crash faults are detected using ZooKeeper's coordination service. As known, ZooKeeper allows users to store persistently coordination data into several hierarchically grouped nodes, called *znode*. However, ZooKeeper has the notion of *ephemeral nodes* too; these special znodes exists as long as the session that created the znode is active, that is when the session ends the znode is deleted. A ZooKeeper client establishes a session with the ZooKeeper service

At session expiration the cluster will delete any/all ephemeral nodes owned by that session and immediately notify any/all connected clients of the change

In this way, is very easy to keep check the status of worker nodes by current primary process. If any of worker node crash, ZooKeeper automatically delete znode associated to crashed server and notifies current leader.

Deferred execution

As known, arbitrary faults are very hard to detect and manage

Deferred execution. Crash faults are detected by the previously existing Hadoop mechanisms, and arbitrary faults are uncommon, so there is no point in always executing $2f + 1$ replicas to usually obtain the same result.

By default, current leader primary process starts only $f + 1$ replicas of the same task, then wait results checking if they all return the same result. If a timeout elapses, or some returned results do not match, more replicas (up to f) are started, until there are $f + 1$ matching replies.

In the best case, without Byzantine faults, only f + 1 replicas are started. If arbitrary faults are uncommon, we have a < f + 1 replica started reducing the overhead

Digest outputs

$f + 1$ matching outputs of a given task (maps or reduces) have to be received to be considered correct. However, tasks output can have considerable size be large therefore to move data from workers to leader is useless. can increse uselessly network traffic, worsening performance.

Digest outputs. f + 1 matching outputs of maps or reduces have to be received to be considered correct. These outputs tend to be large, so it is useful to fetch one output from some task replica and compare just digests (hashes). This way, it is still possible to validate the output without generating much additional network traffic.

Template Styles

The primary parameter given to the "`acmart`" document class is the *template style* which corresponds to the kind of publication or SIG publishing the work. This parameter is enclosed in square brackets and is a part of the `documentclass` command:

```
\documentclass[STYLE]{acmart}
```

Journals use one of three template styles. All but three ACM journals use the `acmsmall` template style:

- `acmsmall`: The default journal template style.
- `acmlarge`: Used by JOCCH and TAP.
- `acmtog`: Used by TOG.

The majority of conference proceedings documentation will use the `acmconf` template style.

- `acmconf`: The default proceedings template style.
- `sigchi`: Used for SIGCHI conference articles.
- `sigchi-a`: Used for SIGCHI "Extended Abstract" articles.
- `sigplan`: Used for SIGPLAN conference articles.

Template Parameters

In addition to specifying the *template style* to be used in formatting your work, there are a number of *template parameters* which modify some part of the applied template style. A complete list of these parameters can be found in the *LaTeX User's Guide.*

Frequently-used parameters, or combinations of parameters, include:

- `anonymous,review`: Suitable for a "double-blind" conference submission. Anonymizes the work and includes line numbers. Use with the `\acmSubmissionID` command to print the submission's unique ID on each page of the work.
- `authorversion`: Produces a version of the work suitable for posting by the author.
- `screen`: Produces colored hyperlinks.

This document uses the following string as the first command in the source file:

```
\documentclass[sigchi]{acmart}
```

4 MODIFICATIONS

Modifying the template — including but not limited to: adjusting margins, typeface sizes, line spacing, paragraph and list definitions, and the use of the `\vspace` command to manually adjust the vertical spacing between elements of your work — is not allowed.

Your document will be returned to you for revision if modifications are discovered.

5 TYPEFACES

The "`acmart`" document class requires the use of the "Libertine" typeface family. Your TeX installation should include this set of packages. Please do not substitute other typefaces. The "`lmodern`" and "`ltimes`" packages should not be used, as they will override the built-in typeface families.

6 TITLE INFORMATION

The title of your work should use capital letters appropriately - https://capitalizemytitle.com/ has useful rules for capitalization. Use the `title` command to define the title of your work. If your work has a subtitle, define it with the `subtitle` command. Do not insert line breaks in your title.

If your title is lengthy, you must define a short version to be used in the page headers, to prevent overlapping text. The `title` command has a "short title" parameter:

```
\title[short title]{full title}
```

7 AUTHORS AND AFFILIATIONS

Each author must be defined separately for accurate metadata identification. Multiple authors may share one affiliation. Authors' names should not be abbreviated; use full first names wherever possible. Include authors' e-mail addresses whenever possible.

Grouping authors' names or e-mail addresses, or providing an "e-mail alias," as shown below, is not acceptable:

```
\author{Brooke Aster, David Mehldau}
\email{dave,judy,steve@university.edu}
\email{firstname.lastname@phillips.org}
```

The `authornote` and `authornotemark` commands allow a note to apply to multiple authors — for example, if the first two authors of an article contributed equally to the work.

If your author list is lengthy, you must define a shortened version of the list of authors to be used in the page headers, to prevent overlapping text. The following command should be placed just after the last `\author{}` definition:

```
\renewcommand{\shortauthors}{McCartney, et al.}
```

Omitting this command will force the use of a concatenated list of all of the authors' names, which may result in overlapping text in the page headers.

The article template's documentation, available at https://www.acm.org/publications/proceedings-template, has a complete explanation of these commands and tips for their effective use.

8 RIGHTS INFORMATION

Authors of any work published by ACM will need to complete a rights form. Depending on the kind of work, and the rights management choice made by the author, this may be copyright transfer, permission, license, or an OA (open access) agreement.

Regardless of the rights management choice, the author will receive a copy of the completed rights form once it has been submitted. This form contains LaTeX commands that must be copied into the source document. When the document source is compiled, these commands and their parameters add formatted text to several areas of the final document:

- the "ACM Reference Format" text on the first page.
- the "rights management" text on the first page.
- the conference information in the page header(s).

Rights information is unique to the work; if you are preparing several works for an event, make sure to use the correct set of commands with each of the works.

9 CCS CONCEPTS AND USER-DEFINED KEYWORDS

Two elements of the "acmart" document class provide powerful taxonomic tools for you to help readers find your work in an online search.

The ACM Computing Classification System — https://www.acm.org/publications/class-2012 — is a set of classifiers and concepts that describe the computing discipline. Authors can select entries from this classification system, via https://dl.acm.org/ccs/ccs.cfm, and generate the commands to be included in the LaTeX source.

User-defined keywords are a comma-separated list of words and phrases of the authors' choosing, providing a more flexible way of describing the research being presented.

CCS concepts and user-defined keywords are required for all short- and full-length articles, and optional for two-page abstracts.

10 SECTIONING COMMANDS

Your work should use standard LaTeX sectioning commands: `section`, `subsection`, `subsubsection`, and `paragraph`. They should be numbered; do not remove the numbering from the commands.

Simulating a sectioning command by setting the first word or words of a paragraph in boldface or italicized text is **not allowed.**

11 TABLES

The "`acmart`" document class includes the "`booktabs`" package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide.*

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

12 MATH EQUATIONS

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

Display Equations

A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered

Table 2: Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_{0}^{\pi+2} f \qquad (2)$$

just to demonstrate LaTeX's able handling of numbering.

13 FIGURES

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Figure 1: 1907 Franklin Model D roadster. Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons. (https://goo.gl/VLCRBB).

Your figures should contain a caption which describes the figure to the reader. Figure captions go below the figure. Your figures should **also** include a description suitable for screen readers, to assist the visually-challenged to better understand your work.

Figure captions are placed *below* the figure.

The "Teaser Figure"

A "teaser figure" is an image, or set of images in one figure, that are placed after all author and affiliation information, and before the body of the article, spanning the page. If you wish to have such a figure in your article, place the command immediately before the `\maketitle` command:

```
\begin{teaserfigure}
 \includegraphics[width=\textwidth]{sampleteaser}
  \caption{figure caption}
  \Description{figure description}
\end{teaserfigure}
```

14 CITATIONS AND BIBLIOGRAPHIES

The use of BibTeX for the preparation and formatting of one's references is strongly recommended. Authors' names should be complete — use full first names ("Donald E. Knuth") not initials ("D. E. Knuth") — and the salient identifying features of a reference should be included: title, year, volume, number, pages, article DOI, etc.

The bibliography is included in your source document with these two commands, placed just before the `\end{document}` command:

```
\bibliographystyle{ACM-Reference-Format}
\bibliography{bibfile}
```

where "`bibfile`" is the name, without the "`.bib`" suffix, of the BibTeX file.

Citations and references are numbered by default. A small number of ACM publications have citations and references formatted in the "author year" style; for these exceptions, please include this command in the **preamble** (before "`\begin{document}` of your LaTeX source:

```
\citestyle{acmauthoryear}
```

Some examples. A paginated journal article [?], an enumerated journal article [?], a reference to an entire issue [?], a monograph (whole book) [?], a monograph/whole book in a series (see 2a in spec. document) [?], a divisible-book such as an anthology or compilation [?] followed by the same example, however we only output the series if the volume number is given [?] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [?], a chapter in a divisible book in a series [?], a multi-volume work as book [?], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [?], a proceedings article with all possible elements [?], an example of an enumerated proceedings article [?], an informally published work [?], a doctoral dissertation [?], a master's thesis: [?], an online document / world wide web resource [? ? ?], a video game (Case 1) [?] and (Case 2) [?] and [?] and (Case 3) a patent [?], work accepted for publication [?], 'YYYYb'-test for prolific author [?] and [?]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [?]. Boris / Barbara Beeton: multi-volume works as books [?] and [?]. A couple of citations with DOIs: [? ?]. Online citations: [? ? ?]. Artifacts: [?] and [?].

15 ACKNOWLEDGMENTS

Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research and the preparation of the work should be included in an acknowledgment section, which is placed just before the reference section in your document.

This section has a special environment:

```
\begin{acks}
...
\end{acks}
```

so that the information contained therein can be more easily collected during the article metadata extraction phase, and to ensure consistency in the spelling of the section heading.

Authors should not prepare this section as a numbered or unnumbered `\section`; please use the "`acks`" environment.

16 APPENDICES

If your work needs an appendix, add it before the "`\end{document}`" command at the conclusion of your source document.

Start the appendix with the "`appendix`" command:

```
\appendix
```

and note that in the appendix, sections are lettered, not numbered. This document has two appendices, demonstrating the section and subsection identification method.

17 SIGCHI EXTENDED ABSTRACTS

The "`sigchi-a`" template style (available only in LaTeX and not in Word) produces a landscape-orientation formatted article, with a wide left margin. Three environments are available for use with the "`sigchi-a`" template style, and produce formatted output in the margin:

- `sidebar`: Place formatted text in the margin.
- `marginfigure`: Place a figure in the margin.
- `margintable`: Place a table in the margin.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

A RESEARCH METHODS

Part One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.

Part Two

Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper. Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.

B ONLINE RESOURCES

Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.