Multi-tenant Big Data Analytics at Scale: Challenges and Opportunities [*]

[Extended Abstract] [†]

Ben Trovato [‡]
Institute for Clarity in Documentation
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

G.K.M. Tobin [§]
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Lars Thørväld [¶]
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Delphi theory

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

1.1 Big Data Brings Multi-tenancy

Big data requires moving computation to data. That mean, s applications need to be run where the data resides. That naturally brings multi-tenancy: multiple applications sharing same set of resources.

1.2 Multi-tenancy Breeds Heterogeneity

Heterogeneity along many dimensions:

1. Workloads (ad-hoc, repeated, iterative, batch Vs. interactive etc.)
2. Application engines (MapReduce, Tez, Spark, etc.)
3. Data (structured, semi-structured, poly-structured, etc.)
4. People (computer scientists, data scientists, non-engineers, etc.)
5. Needs (SLAs, tension between throughput and latency, tension between speed of innovation and tight-ship control)

(Note: It should not come out as Teradata has solved all problems)

1.3 Gatekeepers of Multi-tenancy

Hence, dealing with multi-tenancy is a complex problem. Many gate-keepers:

- Admission control

[*](Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

[†]A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

[‡]Dr. Trovato insisted his name be first.

[§]The secretary disavows any knowledge of this author's actions.

[¶]This author is the one who did all the really hard work.

3^{rd} *Biennial Conference on Innovative Data Systems Research (CIDR)* January 7-10, 2007, Asilomar, California, USA.

- Scheduler
- Resource Manager

Ultimately all of these act based on configurations supplied by humans.

1.4 Roadmap

- Section 2 will show—based on operational experiences from very large clusters at Rocket Fuel—challenges that operators of multi-tenant clusters face.
- Section 3 discusses a solution, called Colossal, that we have started to build to address these challenges.
- Section 4 outlines a research agenda.

2. OPERATIONAL CHALLENGES IN MULTI-TENANT CLUSTERS FOR BIG DATA ANALYTICS

The Hercules cluster is a 642-node petabytes-level data processing platform at Rocket Fuel, serving multiple systems, such as Hadoop, Hive, HBase, and Hue[]. Hercules is shared among users from different teams and installed the Hadoop fair scheduler [] to maintain map and reduce slots among those users. Inside the fair scheduler, the cluster is divided into six pools, namely analyst, default, engineer, mobile, modeling, and prod, representing the main types of workload with unique characteristics. For example, the analyst pool is composed of jobs generated by Hive queries, modeling pool involves CPU-intensive machine learning jobs, and prod pool runs recurring ETL production jobs.

Furthermore, there are several roles participate in the resource management process, and generally serve conflicting interests. For example, the cluster operators, whose objective is to maximize the cluster performance while securing the business SLAs. The users such as data analysts, on the other hand, wish to minimize the latency of their jobs and show little interest in the overall performance of the cluster. Another role is the coordinator, who collects and coordinates the resource requirements of users and ensures steady and robust resource provisioning among all users.

This multi-tenant scenario at Rocket Fuel has become increasingly prevalent in most Big Data companies, and has brought about unprecedented operational challenges due to the intricate contention, dependency, and interference between systems, jobs, work-flows, and users. Even a slight change in these elements can result in unpredictable consequences. For example, it is frequently observed that a 5% increase in the map demand of analyst pool tends to fail 10% jobs in modeling pool, and simple Hive queries from analyst pool can significantly undermine the overall cluster performance by decreasingly the effective utilization by up to 40%. We believe that deep understanding of these problems has tremendous practical value to many Big Data companies, and is of central importance in multi-tenant resource management. We summarize some of these frequently faced problems at Rocket Fuel, which motivate our work.

2.1 Understanding the real cluster performance

One type of challenge lies in understanding the real performance of the cluster.

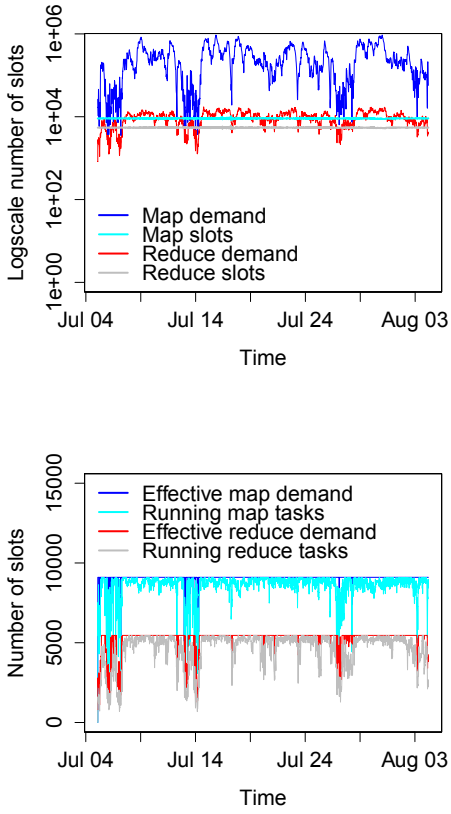

Figure 1: Overview of the cluster utilization

2.1.1 *Cluster utilization and effective utilization*

One frequently asked question involves the cluster utilization, which we restrict to the map and reduce slot use. Fig. 1 shows the demand (the number of ready tasks) and the number of running tasks over the period between July. 5 and Aug. 4. according to OpenTSDB[]. As can be seen, the demand is almost always higher than the total number of slots. This implies that there are almost always sufficient ready tasks for the cluster to run at full capacity. However, one must also notice the otherwise: the number of running tasks is generally less than the corresponding demand and is also less than the corresponding number of slots. This discrepancy can be explained by various reasons, such as preemption (actually implemented as killing in Hadoop), delay in the cluster etc. Let us then consider the effective utilization, which accounts for only those successful tasks. That is, a task and its belong jobs both finished with status SUCCESS. These tasks are effective in terms of contributing to completion of the job. We first define the effective map demand Dm(t) and the effective reduce demand Dm(t) at instant t as follows:

$$D_m(t) = \min\{S_m(t) - C_m(t), \#\text{MAP\_SLOTS}\}$$

Here, #MAP_SLOTS is the total number of map slots on the cluster. Similarly, the definition of D_r also follows. Sm(t) and Cm(t) denote the number of runnable map tasks and completed map tasks, respectively, by time t. Similarly, Sr(t) and Cr(t) are defined for reduce tasks. The values

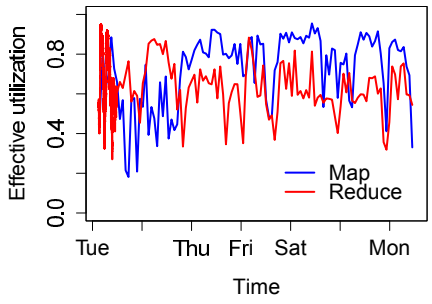

Figure 2: Effective map and reduce utilization

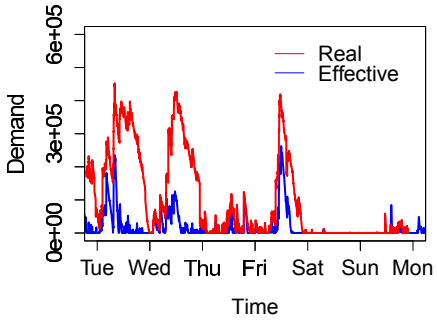

Figure 3: Map demand of analyst pool

Table 1: Task status statistics during July 22 and 28

	Percentage (%)
Failed jobs	3
Killed jobs	3.11
Succeeded jobs	93.89
Failed task attempts	0.32
Killed task attempts	18
Succeeded task attempts	81.68

of the above variables can be easily obtained from the job history log and Hadoop configurations.

The effective demand implies the maximum amount of work, in number of tasks, the cluster can handle at any instant. Ideally, the number of running tasks should be equal to the corresponding effective demand when peak performance is reached. Naturally, we can define the effective utilization U(t) of the cluster as the ratio:

$$U_m(t) = \frac{R_m(t)}{D_m(t)}$$

where Rm(t) and Rr(t) are the number of successful running map and reduces tasks, respectively, at instant t. Fig. 2 depicts the Um(t) and Ur(t) over the week between July 29 and August 4. The overall effective utilization for map and reduce is 72.5% and 67.3%, respectively, over this period. While the demand is above the cluster capacity, the cluster does not run at its peak performance. One might want to investigate the reasons that are responsible for the decreased effective utilization.

2.1.2 *Possible reasons for low effective utilization*

There are several factors that can undermine the effective utilization of a cluster. Killed jobs, for example, decrease the effective utilization by wasting the cluster resource spent on them. Speculative execution also reduces the effective utilization: a task attempt will be killed if it is an speculative task and the original task has succeeded or the either way. Table 1 gives the task status statistics by status. As can be seen that there are considerable amount of unsuccessful task attempts 18.32%. In addition, we have observed that the analyst pool, which involves jobs mainly from Hive, suffers most from this issue. The reason is that jobs in analyst pool typically entail reading large input files, resulting in a significant number of maps. These jobs tend to run longer and have higher chances to be killed by inpatient users or cluster operators.

Figure. 2 shows the map demand of analyst pool during July 29 to August 4. The demand consists of those running tasks, speculative tasks, and tasks scheduled, but are waiting, for execution. The demand shown as R̈eal̈accunts for all launched tasks, and the effective demand consists of only successful tasks of successful jobs, which really contributed to the effective utilization.

Another affecting factor is the preemption in fair scheduler. Preemptions occur when a pool has been below its minimum share or half fair share for certain period of time. In this case, the fair scheduler will kill most recently launched tasks in pools that have more tasks running than their fair shares. Clearly, the ongoing work of these tasks gets lost once they are killed.

Furthermore, the effective utilization only measures whether or not the maps and reduces are effectively allocated to tasks. It may still be an optimistic assessment: even if slots are allocated to tasks, the tasks may not necessarily turly make progress. For example, the slow start parameter of Hadoop controls when should the reduces start sorting; however, reduces are often launched much earlier, waiting for the completion of map phase. While these reduce tasks have acquired reduce slots, they may not be doing any real work.

2.2 Workload Tuning

Another type of challenges has something to do with workload tuning. In particular, cluster operators are especially interested in knowing the overall impact of tuning the workload in certain pool. Due to the shared nature of the cluster, it is difficult to isolate the impact of the change in workload of a pool. For example, we have observed that there is considerable proportional of queries in analyst pool that have failed due to long latency, as is shown in Figure 2. However, some of these queries can be identified and avoided in advance.

2.3 Configuration Optimization

A third type of challenges is stemmed from optimizing the configurations.

2.3.1 *Should preemption be turned on?*

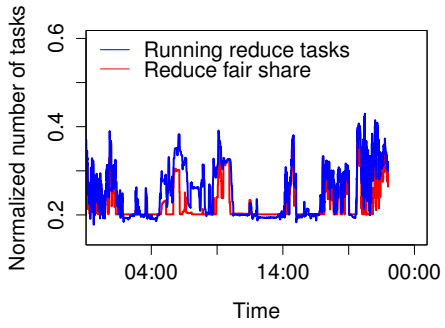

Figure 4: Long running reduces in modeling pool

The cluster operators are often faced with whether or not to enable preemptions in fair the scheduler. On one hand, turning on preemptions enforces the SLAs of performance critical pools, such as jobs in the production pool. Whether preemption is needed to guarantee SLAs really depends on the workload and cluster capacity. Fig. 3 shows the reduce fair share and number of running reduce tasks of modeling pool over one day. As can be seen, modeling pool uses more reduce slots than its reduce fair share in a five-hour interval between 4:00 and 9:00 the fair share. According to the computation of fair shares, there were other pools that suffered from starvation during this period. Thus, turning preemption is needed to guarantee the SLAs of those starved pools.

On the other hand, the preemption in fair scheduler is actually killing, which leads to reduced effective utilization. Thus, turning off the preemption can improve the effective utilization, if preemptions become a performance issue. We use Colossal to compare the effective utilization and average job latency for both cases.

2.3.2 *Resource planning in advance*

The job of the coordinator of a pool involves foreseeing the demand growth of the pool, and planning for the resources in advance. Once the cluster operators informed of the need, they must evaluate the impact of the change. For instance, the cluster operator are sometimes bewildered by questions such as ?What will be the impact of a 5% increase in the workload in the analyst queue?? These questions have no easy answers, and any cursory decisions or avoiding these questions can cause catastrophic consequences to production jobs.

2.3.3 *Loosening the admission control in a queue*

The admission control of a pool in Hadoop restricts the concurrency level of the pool. Loosening the admission control of a pool allows more jobs to run concurrently. In particular, an loosened admission control only effect when there are enough jobs submitted. It should also be noted that job size distributions are different in each pool, which adds to the complication of the impact of admission control.

2.3.4 *Impact of data formats*

The table format, such as ORC, can also make a difference in overall performance of a cluster. Fixing the workload, a different table format will result in different start time, completion time, and duration of tasks, jobs, and workflows.

3. A COLOSSAL ATTEMPT TO MANAGE MULTI-TENANT BIG DATA ANALYTICS AT SCALE

We developed Colossal, a comprehensive solution framework to answer questions as described in Section 2. Colossal consists of four main components: 1) a log parser which extracts workload traces from Hadoop job history, 2) a workload model which is trained using the extracted workload traces and then can be used to generate similar workload, 3) a scheduler simulator works in an accelerated virtual timeline, and 4) a performance inspector which evaluates the performance in user-specified metrics, such as effective utilization, throughput, and average job latency. The implementation of Colossal has several advantages:

- Efficiency: Produce the performance metrics and schedule of one week?s workload (55762 jobs and 35260434 tasks) in just few minutes on a single commodity machine.
- Flexibility: Allow for easy abstraction, modification, and generation of workload of varied time period lengths.
- Modularity: The workload model, simulator, and other components are pluggable. Each can be easily replaced with one fits the practice.

The Colossal serves as a Ẅhat-Ifëngine to answer questions as described in Section 2. These questions are first translated and then effected on the workload model. The simulator processes the workload generated by the workload model and outputs the schedule and performance metrics for evaluation. The Colossal framework also allows for optimization in terms of user-specified performance metrics. The optimization is done by searching over the possible parameter space, and can be further accelerated by parallelizing with MapReduce and using heuristics to narrow down the search.

3.1 An example workload model

To answer performance questions regarding the Hercules cluster, we first developed a workload model which can generate similar workload of varied time period lengths. We observed that the job arrival follows a Poisson process with a relatively constant arrival rate for each pool. In addition, the map and reduce task duration follows a lognormal distribution, which was also shown in []. The task count of a job is also approximated using a lognormal distribution for each pool. In particular, the SETUP task is considered as a MAP and CLEANUP task is considered as a REDUCE. Thus, every job has a positive map task count and reduce task count. A fitted workload model for analyst pool is given by

Here, we used maximum likelihood estimation of parameters[]. logmean and logsd are the mean and standard deviation of the logarithm of the observed value, respectively. Workload models for other pools were also derived in a similar manner.

3.2 Colossal-based solutions

Overall, Colossal supports 1) simulating the performance of the cluster under given workload and configuration, 2)

Table 2: Workload model of analyst pool (Time is in seconds)

Parameter	Description	Estimated Value
job_arrival_rate	Number of job arrivals per second	0.00282392166
logmean_maps_per_job	Mean of the logarithm of job map count	4.860108
logmean_reduces_per_job	Mean of the logarithm of job reduce count	1.720507
logsd_maps_per_job	Standard deviation of the logarithm of job map count	3.202215
logsd_reduces_per_job	Standard deviation of the logarithm of job reduce count	2.540193
logmean_map_duration	Mean of the logarithm of job task duration	3.765196
logmean_reduce_duration	Mean of the logarithm of reduce task duration	5.154496
logsd_map_duration	Standard deviation of the logarithm of map task duration	0.8837955
logsd_reduce_duration	Standard deviation of the logarithm of reduce task duration	2.043326

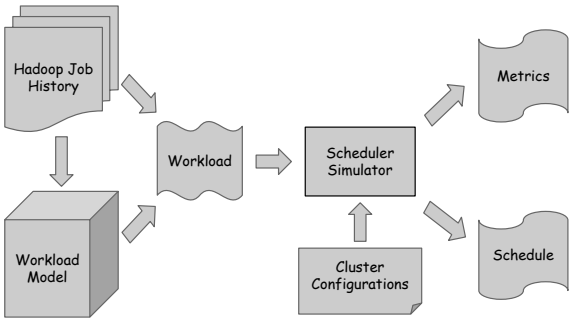

Figure 5: Overview of Colossal

modeling the workload and allowing for modifications, and 3) automatic parameter tuning based on given performance metrics. Specifically, 1) is achieved using the pluggable scheduler simulator in Colossal, 2) is done by fitting a statistical model to historical workload traces, and 3) relies on 1) to perform exhaustive or heuristic search over the parameter space.

Colossal is designed to answer ẄWhat-Ifq̈uestions concerning the cluster performance, workload, and parameters. The general steps to pose a ?What-If? question are shown in Fig 5. First, workload is either extracted from Hadoop job history or generated using the workload model. Second, the simulator inputs the workload and cluster configurations, and then outputs a performance metrics file, which is similar to Hadoop metrics, as well as the schedule.

3.2.1 Posing a question in Colossal

To pose performance related questions, one simply needs to visualize the metrics time series in the metric file, or alternatively to analyze the schedule. Colossal provides several built-in performance metrics, such as effective utilization as described in the previous section and the average job latency for each pool. To pose questions concerning the workload, the statistical workload model introduced in 3.1 is readily available for modifications. As for automatic parameter tuning, one can fix the objective, e.g., average job latency of analyst pool, and then repeat the simulation against different parameter combinations. In this case, constraints can also be specified by the user in a similar manner. This is useful to guarantee the SLAs of production jobs.

More specifically, users define an objective function F(M,S) using Colossal APIs, where M and S are the output metrics and schedule. F(M,S) defines the performance metrics for evaluation, and Colossal handles the optimization of F(M,S). In Colossal, users can specify the parameters(free variables) to optimize, e.g., minShare of a given pool. Further, Colossal also allows for heuristics to speedup the optimization. So far, Colossal provides three APIs for writing an objective: 1) AvgJobLatency(S), which computes the average job latency of each pool, 2) Throughput(S), which computes the throughput (number of jobs per second) of each pool, and 3) EffUtil(S), which computes the defined effective utilization. Table 3 summarizes the procedures to pose questions described in Section 2.

4. PROPOSED RESEARCH AGENDA

Spectrum that involves: better visualization and control tools for operators, to truly automatic decision making by non-human gatekeepers of multi-tenancy.

5. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

6. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

Table 3: Procedures to pose questions in Colossal

Question	Procedures
What is the effective utilization of the cluster?	Use the simulator to generate the schedule files. Then use built-in function EffUtil(S) to obtain the effective utilization based on S.
What is the impact of a change in workload?	Reflect the change in the workload model, and then run Colossal to generate metrics and schedule files. Next, compute the performance metrics of interest based on these two files.
Should preemptions be turned off?	First, visualize the metrics file generated by Colossal for the present workload. Then identify any long-running tasks problem in any pools. Next, compare the performance before and after turning off the preemptions. Finally, make a decision according to above results.
How to evaluate the impact of a 5% increase in the workload of analyst pool?	First, run Colossal to obtain baseline performance using any user-defined metrics. Then increase the job arrival rate of analyst pool by 5% and re-evaluate the performance. Finally, compare the results above.
How loosening the admission control of modeling pool impacts the performance?	Another module is needed to map the original workload to an adjusted one according to the loosened admission control. Then use Colossal to evaluate the performance using the adjusted workload.
What difference does ORC format make?	Likewise, another module is needed to produce the adjusted workload and use Colossal to evaluate the performance of the adjusted workload.

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

A.2.2.1 Inline (In-text) Equations.

A.2.2.2 Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LaTeX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References