18 October 2013

Estimating job runtime for CMS analysis jobs

Igor Sfiligoi

Abstract

The basic premise of pilot systems is to create an overlay scheduling system on top of leased resources. And by definition, leases have a limited lifetime, so any job that is scheduled on such resources must finish before the lease is over, or it will be killed and all the computation wasted. In order to effectively schedule jobs to resources, the pilot system thus requires the expected runtime of the users jobs. Past studies have shown that relying on user provided estimates is not a valid strategy, so the system should try to make an estimate by itself. This paper provides a study of the historical data obtained from the CMS Analysis Operations submission system. Clear patterns are observed, suggesting that making prediction of an expected job lifetime range is achievable with high confidence level in this environment.

Presented at *CHEP2013 Computing in High Energy Physics 2013*

Estimating job runtime for CMS analysis jobs

I Sfiligoi

University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA

isfiligoi@ucsd.edu

Abstract. The basic premise of pilot systems is to create an overlay scheduling system on top of leased resources. And by definition, leases have a limited lifetime, so any job that is scheduled on such resources must finish before the lease is over, or it will be killed and all the computation wasted. In order to effectively schedule jobs to resources, the pilot system thus requires the expected runtime of the users' jobs. Past studies have shown that relying on user provided estimates is not a valid strategy, so the system should try to make an estimate by itself. This paper provides a study of the historical data obtained from the CMS Analysis Operations submission system. Clear patterns are observed, suggesting that making prediction of an expected job lifetime range is achievable with high confidence level in this environment.

1. Introduction

In recent years, the pilot paradigm has become the dominant way of using widely distributed computing resources for the scientific communities, an example pilot product being the glideinWMS[1]. Its separation of resource provisioning from user job scheduling has proven to be very suitable for Grid infrastructures, like the Open Science Grid (OSG)[2][3] and the European Grid Initiative (EGI)[4].

By definition, dynamic resource provisioning operates on a notion of leases. Each provisioned resource is expected to have a limited lifetime. Furthermore, Grid computing, and more generally batch computing are traditionally job based, so the lease lifetime limit is set at job execution start time, and cannot be extended. Any provisioned resource thus has to take the remaining lease time in consideration when picking the next user job to run. Which, by extension, requires the knowledge of that job's runtime, or at the very least, a good estimate of its max runtime. This would be especially important for multi-core pilots, where end-of-life costs are substantial, as shown in figure 1.

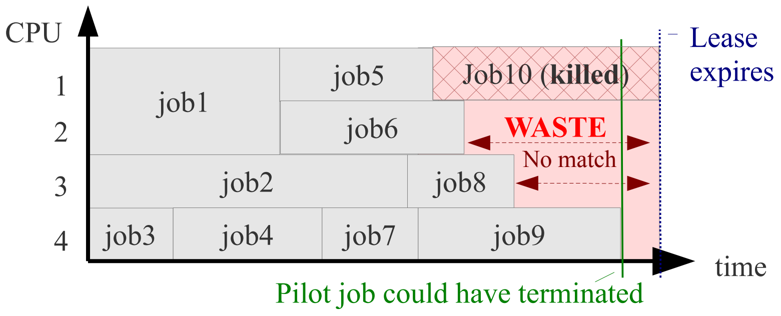


Figure 1. Pilot lifetime and job scheduling

Expecting users to provide an accurate estimate is however unrealistic; studies have shown that even in presence of tangible rewards users will not provide a good estimate[5]. On the other hand, automated prediction of job runtimes also seems not to be feasible in the general case[6]. As a result, most scheduling systems end up using the worst case scenario for scheduling purposes.

In this paper we report on an exploratory study we conducted of the historical data for user analysis jobs of the CMS Experiment at the Large Hadron Collider[7][8]. The analysis reveals clear patterns that could be used to effectively predict the job runtime in a narrow value range, with the upper bound being most of the time significantly lower than the worst case assumption. As a result, using this technique could allow for significant gains in pilot resource utilization.

2. The CMS analysis environment

The CMS analysis operations infrastructure aims at enabling CMS physicists to perform their analysis. It is based on glideinWMS[1] as the underlying Workload Management System (WMS) and CRAB[9] as the submission mechanism. Due to the nature of the WMS, CMS has several submission nodes, and CRAB distributes the user submissions randomly among them, for load balancing purposes, as shown in figure 2.

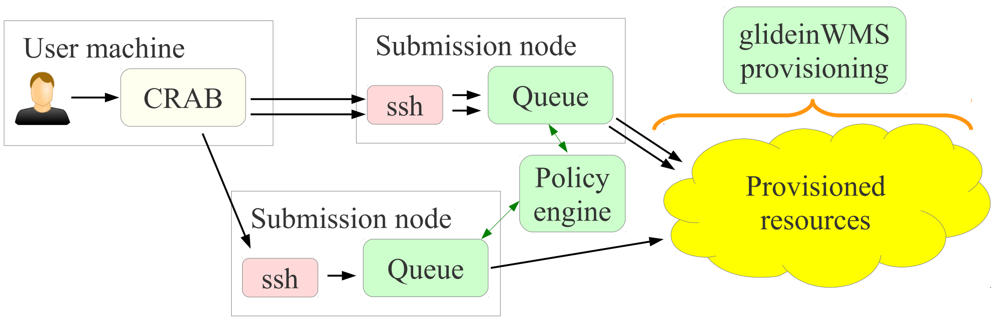


Figure 2. An overview of the CMS analysis operations infrastructure

The data being analyzed by the CMS users is composed of collision readout events, properly transformed and/or augmented to enable physics analysis. The events are stored in files of approximately the same size, and the files are then logically grouped into datasets.

Each user submission is referred to as **a task**. The user provides an executable bundle and the dataset it wants this executable to be run over. CRAB analyzes the dataset, splits the task into a **set of jobs**, each analyzing **approximately the same number of events**, and then submits them to glideinWMS for scheduling and execution. All jobs belonging to a single task are submitted to a single submission node.

The CMS analysis operations is currently using a fixed runtime estimate for all of the users' jobs, which is at present set to 8h. The glideinWMS system allows for a flexible matching, so this optimistic estimate is used only when matching jobs that have never started. If a job gets killed for any reason, the system will try to find another pilot to run it in. To avoid multiple terminations, the matchmaking will at this point use the worst case runtime value, which is currently set to 22h. Users are allowed to set both of those numbers on a task-by-task basis, but very few users ever do.

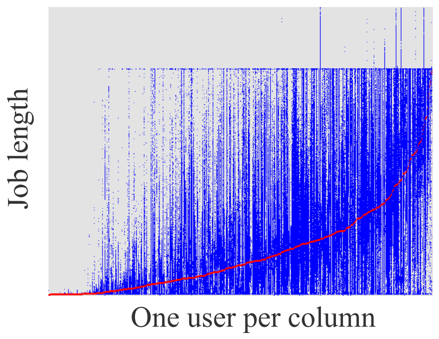
3. Analysis of the historical data

The data being analyzed covers 5 months worth of submissions to a subset of the CMS analysis operations infrastructure, namely the period of Apr to Aug 2013 and the UC San Diego submitters only. The data contains 18k tasks from 608 users, for a total of 1.4M jobs.

3.1. Qualitative analysis

We first attempted to establish a correlation between users and their job runtimes. As can be seen in figure 3a, such a correlation indeed seems to exist, with the average job runtime of certain users being an order of magnitude lower that those of others. We also checked mean per-user job times over different submission time windows, and found they did not seem to substantially change over time.

Given the observed correlation for the mean job runtimes, we next looked for a usable threshold for making predictions. A good pick seems to be a cut on the 80% of the jobs closest to the user's mean job runtime. As can be seen in figure 3b, for most users, job runtimes stay below the 2x mean threshold. No good lower limit has instead been observed at this point; most users seem to have a significant fraction of very short jobs.



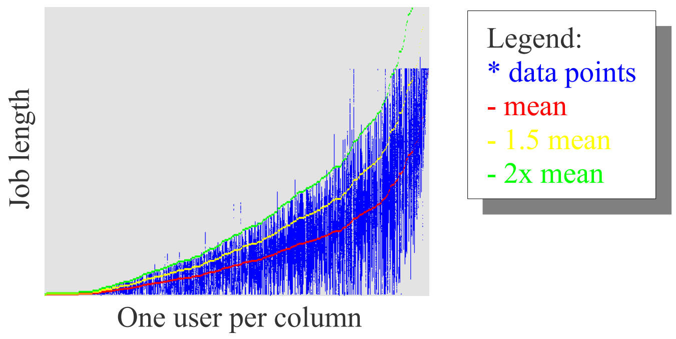
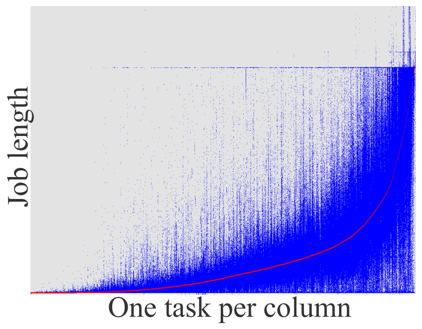


Figure 3a. Job runtimes by user, sorted by per-user mean job runtime

Figure 3b. Job runtimes by user, sorted by per-user mean job runtime. Limited to 80% of jobs closest to the user's job runtime mean.

Next, we looked at the runtimes of the jobs belonging to a single task. Since each task belongs to a single user, the correlation must be at least as good, but the available historical data shows that it seems to be significantly better, as shown in figure 4a. Again, we looked for a good threshold to use for prediction purposes, and settled for a 90% cut. As shown in figure 4b, jobs from most tasks stay below 2x mean in this selection. Again, no usable lower bound have been observed at this point.



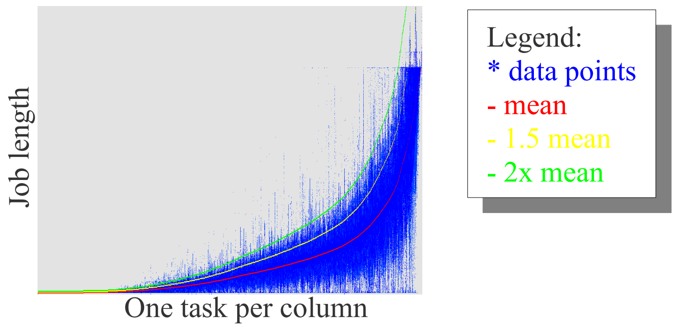


Figure 4a. Job runtimes by task, sorted by per-task mean job runtime

Figure 4b. Job runtimes by task, sorted by per-task mean job runtime. Limited to 90% of jobs closest to the tasks's job runtime mean.

Finally, it should be noted that in both per-user and per-task figures, for most columns the 2x mean threshold is significantly lower than worst case scenario, which can be inferred from the plateau in the figures.

3.2. Quantitative analysis

Having established that there likely exist thresholds useful for making predictions, we analyzed the average effectiveness over the whole range of users and/or tasks. We also calculated the **average per-user and per-task mean job runtime, and in both cases it turns to be just over 4h; 5x lower than the 22h worst case limit.**

The data shows that the per-user mean job runtime is likely a pretty decent predictor of the user's max job runtime. As can be seen from table 1, a threshold of 2x mean includes almost 90% of the jobs. And if we settled for 3x mean, we could get over 96% of the jobs covered. As expected, though, the per-user mean does not have a usable lower bound threshold; only 66% of job runtimes are above half the per-user mean job runtime. Again, we checked the values over different time intervals in the data, and found them to be quite stable in time.

Table 1. Per-user statistics

(a) Fraction of jobs below treshold

Treshold	Fraction
1.5x mean	81%
2x mean	89%
3x mean	96%

(b) Fraction of jobs above treshold

Treshold	Fraction
0.67x mean	56%
0.5x mean	66%
0.33x mean	76%

When analyzing the per-task statistics, almost all jobs sit below the 2x mean job runtime threshold, as can be seen from table 2. And even setting the man job runtime threshold at 1.25x mean covers almost 90% of the jobs. Moreover, unexpectedly, the per-task mean job has also usable lower bound thresholds. Over 96% of the jobs have runtimes that are above half the mean threshold. This makes it feasible to look for intervals that might be usable for prediction, seeing that over 92% of the jobs were in the [0.5x,1.5x] range around the per-task mean.

Table 2. Per-task statistics

(a) Fraction of jobs below treshold

Treshold	Fraction
1.25x mean	89%
1.5x mean	95%
2x mean	99%

(b) Fraction of jobs above treshold

Treshold	Fraction
0.8x mean	82%
0.67x mean	91%
0.5x mean	96%

(c) Fraction of jobs in an interval around the mean

Interval	Fraction
[0.67x,1.5x]	87%
[0.5x,1.5x]	92%
[0.5x,2x]	95%

3.3. Making preditions at runtime

The discovered per-user job max runtime thresholds could be used for prediction purposes, since the per-user mean does not seem to change significantly over time.

The extracted per-task thresholds cannot however be used for prediction. With very few exceptions, each submitted task is unique. And knowing the task's mean job runtime after all the jobs of that task are done, is worthless. We thus decided to analyze how early in the lifetime of a task can we extract thresholds that can be used to make useful predictions. We chose to use the termination time of the first N jobs submitted that belong to a specific task as the measuring point. This allows us to have a uniform and unbiased measurement.

The results obtained by the first N jobs approach are very positive. As can be seen in table 3, on average, even just looking at the runtime of the first submitted job gives a pretty decent prediction, definitely better than relying on per-user prediction alone. And as seen in table 4, by the time the first 10 jobs are done, one already gets numbers that are close to the ones one would get after the fact.

Table 3. Statistics after 1st job in task terminates

(a) Fraction of jobs below treshold

Treshold	Fraction
1.5x mean	90%
2x mean	94%

(b) Fraction of jobs above treshold

Treshold	Fraction
0.67x mean	83%
0.5x mean	90%

Table 4. Statistics after first 10 jobs in task terminate

(a) Fraction of jobs below treshold

Treshold	Fraction
1.5x mean	94%
2x mean	97%

(b) Fraction of jobs above treshold

Treshold	Fraction
0.67x mean	85%
0.5x mean	92%

The analysis also showed that 92% jobs belonged to tasks with 20 or more jobs. So, as expected, most of the jobs are still alive at prediction time; 82% after the first job and 63% after the first 10. This allows us to make a useful prediction, thus having a major impact on the scheduling decisions.

4. Conclusions

This exploratory analysis of the historical data from CMS analysis operations shows that there are clear patterns that could be used for predicting job runtimes within a narrow range, and with a good confidence level, with the upper bound being significantly lower than the worst case assumption most of the time.

This could be used to efficiently schedule those jobs in the pilot environment of the CMS analysis operations. And since there are already mechanisms in place to recover from a wrongly predicted job runtime, a narrower prediction window can easily offset a small mis-prediction rate.

It should however be noted that the obtained results cannot really be directly applied to any other environment. But they do show that effective job runtime prediction is feasible in at least some pilot environments, and should not be discounted a-priori.

Acknowledgements

This paper would not be possible without the help from E Arias-Castro, and from the CMS analysis operations team, in particular S Belforte and J Letts. This work was partially sponsored by the US National Science Foundation Grant PHY-1120138.

References

[1] Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Würthwein F 2009 The pilot way to grid resources using glideinWMS *Comp. Sci. and Info. Eng., 2009 WRI World Cong. on* **2** pp 428-432 doi:10.1109/CSIE.2009.950

[2] Pordes R et al. 2007 *J. Phys.: Conf. Ser.* **78** 012057 doi:10.1088/1742-6596/78/1/012057

[3] Sfiligoi I, Würthwein F, Dost J M, MacNeill I, Holzman B and Mhashilkar P 2011 Reducing the human cost of grid computing with glideinWMS *Proc. Cloud Computing 2011* (Rome, Italy) pp 217-221 ISBN 978-1-61208-153-3 http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2011_8_40_20068

[4] Kranzlmüller D, Marco de Lucas J and Öster P 2010 *Remote Instr. and Virt. Lab.* pp 61-66 doi:10.1007/978-1-4419-5597-5_6

[5] Bailey Lee C, Schwartzman Y, Hardy J and Snavely A 2005 *Job Scheduling Strategies for Parallel Processing L. N. in Comp. Sci.* **3277** pp 253-263 doi:10.1007/11407522_14

[6] Tsafrir D, Etsion Y and Feitelson D G 2007 *Parallel and Dist. Sys., IEEE Trans. on* **17 6** pp 789-803 doi:10.1109/TPDS.2007.70606

[7] Chatrchyan S et al. 2008 *J. of Instr.* **3** S08004 doi:10.1088/1748-0221/3/08/S08004

[8] Bradley D et al. 2010 *J. Phys.: Conf. Ser.* **219** 072013 doi:10.1088/1742-6596/219/7/072013

[9] Belforte S et al. TBD Using ssh as portal, the CMS CRAB over glideinWMS experience *Proc. CHEP 2014* (Amsterdam, NL)