

A History-based Estimation for LHCb job requirements

Nathalie Rauschmayr

CERN - European Organization for Nuclear Research

E-mail: nathalie.rauschmayr@cern.ch

Abstract. The main goal of a Workload Management System (WMS) is to find and allocate resources for the given tasks. The more and better job information the WMS receives, the easier will be to accomplish its task, which directly translates into higher utilization of resources. Traditionally, the information associated with each job, like expected runtime, is defined beforehand by the Production Manager in best case and fixed arbitrary values by default. In the case of LHCb's Workload Management System no mechanisms are provided which automate the estimation of job requirements. As a result, much more CPU time is normally requested than actually needed. Particularly, in the context of multicore jobs this presents a major problem, since single- and multicore jobs shall share the same resources. Consequently, grid sites need to rely on estimations given by the VOs in order to not decrease the utilization of their worker nodes when making multicore job slots available. The main reason for going to multicore jobs is the reduction of the overall memory footprint. Therefore, it also needs to be studied how memory consumption of jobs can be estimated.

A detailed workload analysis of past LHCb jobs is presented. It includes a study of job features and their correlation with runtime and memory consumption. Following the features, a supervised learning algorithm is developed based on a history based prediction. The aim is to learn over time how jobs' runtime and memory evolve influenced due to changes in experiment conditions and software versions. It will be shown that estimation can be notably improved if experiment conditions are taken into account.

1. Introduction

In the case of the LHCb experiment, estimation of job requirements is done by a Production Manager and therefore the decisions are based on user experience. In order to avoid that jobs are interrupted by a grid site, requirements are in general overestimated. Providing better estimations can allow improvements in many respects. For instance, knowing in advance whether a job might be critical in terms of memory footprint allows to submit it directly to a high memory queue. Giving better runtime estimates allows the batch system of grid sites to improve scheduling and therefore to increase efficiency and throughput. This becomes essential in the context of multicore jobs, where jobs with different core requirements shall run on the same resources [1]. If there are too many jobs with overestimated runtimes then it can easily lead to bad system utilization: the cores are released earlier than expected but the next job in the batch system queue might not be able to start earlier because it requires more cores than available at that time. These are just a few examples which show how important it is to better estimate job requirements. Runtime estimation is actually a well studied problem in the context of High Performance Computing [7],[8],[9],[10]. They often approach this problem by focusing on some



generic job parameters and comparing it with past jobs. There have been also some recent studies in the HEP community [5],[6], where they analyse user estimates and the impact of high and low luminosity data on job performance.

This paper will go a step further and first analyse which job features have the largest impact and the studies will not only focus on runtime but also on memory footprint of jobs. The LHCb experiment stores a lot of information about each job in a database [3]. For instance, it is known on which hardware (workernode, CPU model, RAM size etc.) a job has been executed. It contains information about the input file, like size and name of file, number of events and in which LHCb run the data has been recorded. The latter feature allows to obtain physics conditions for the corresponding LHCb run [2], like pile-up, trigger configuration, luminosity etc. This is important since the higher the pile-up the more complex is the reconstruction of events. Taking such correlations into account, can help to improve estimation.

This paper proposes to predict job requirements based on information about past jobs. Therefore, it starts first with a workload analysis in section 2. Section 3 evaluates the most important features and section 4 presents a supervised learning algorithm.

2. Workload Analysis

In a first step, LHCb workloads from 2011 and 2012 have been analyzed in order to evaluate how the increase of LHC beam energy changed the requirements of jobs. Normalized time per event can be used as metric, in order to compare runtime of jobs running on different worker nodes and processing different number of events. This metric can be computed as:

$$\frac{\text{HEPSPEC Value} \cdot \text{CPU time}}{\text{Number of Events}} \quad (1)$$

2.1. Runtime

Fig. 1 shows that the distribution of runtime per event values can be approximated by a Gaussian curve. Therefore, a maximum likelihood estimation can be applied in order to obtain the parameters of this distribution like the standard deviation and the maximum likelihood. These values can then be used in order to predict future jobs. Fig. 1 shows that the majority of reconstruction jobs in 2011 required about 11.71 HS06.s, while in 2012 this has increased to 17.91 HS06.s. Also the amount of data has significantly increased. This is caused by the change of experiment conditions in 2012. Due to larger data rate, different trigger conditions and higher luminosity more complex events have been recorded. The standard deviation is about 2.1 in the year 2011 and about 2.7 in 2012. This means that runtime is on average wrongly estimated by 2.1 respectively 2.7 HS06.s. The LHCb experiment provides luminosity leveling, which ensures a relatively stable luminosity value during a run. That's the main reason why a Gaussian distribution can be observed. This is in contrast to other experiments like CMS or ATLAS, where high luminosity data cause significant outliers in the runtime of jobs.

Stripping is the processing step which takes place after the reconstruction. It sorts the reconstructed events into different output streams. The workload analysis is shown in Fig. 2. A slight increase from 5.26 HS06.s per event to 5.87 HS06.s can be observed. This is mainly caused by different stripping options in the corresponding year. The characteristics of events do not influence the computational complexity of the jobs since the jobs only classify events and put them into different output streams. It can be seen that more data has been processed by stripping jobs in the year 2012 compared to 2011.

2.2. Memory

The same analysis has been applied for the memory footprints of jobs from 2011 and 2012. These values clearly do not follow a Gaussian distribution any longer as shown in Fig. 3. On

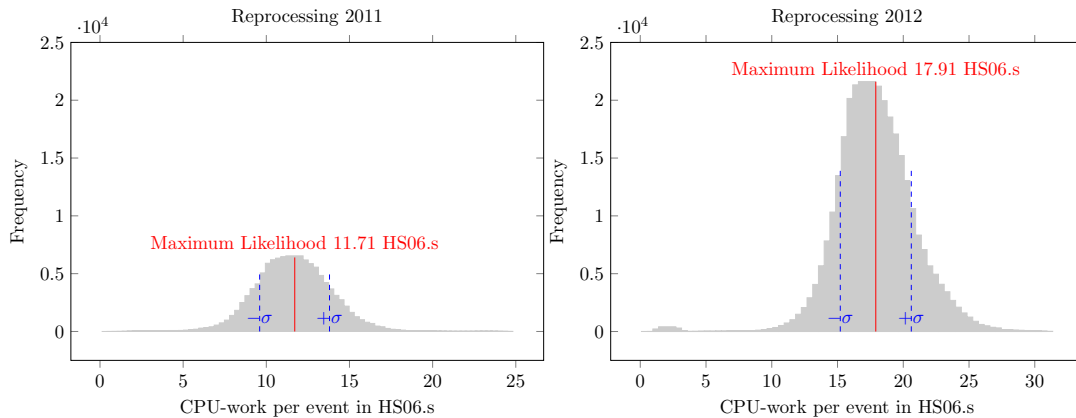


Figure 1. Required time per event for reconstruction jobs [4]

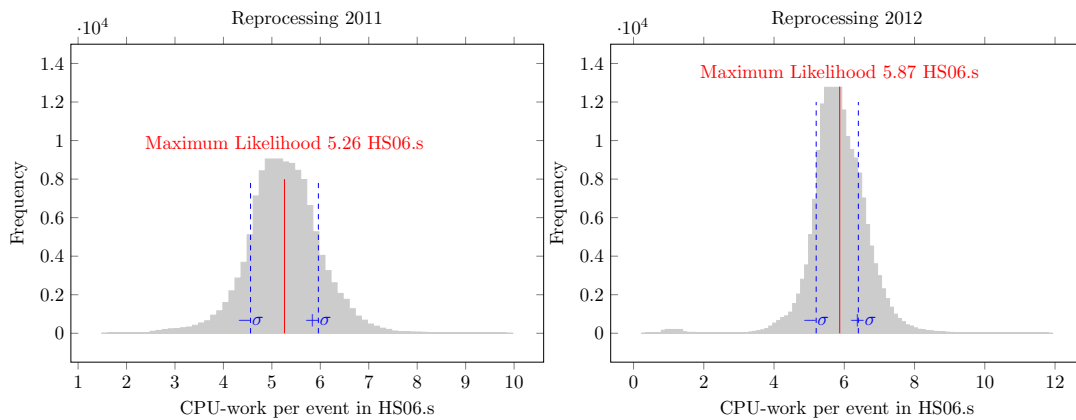


Figure 2. Required time per event for stripping jobs [4]

the other side, the parameter memory is not as critical as runtime, because an overestimation will not have any negative impact. Reconstruction jobs required on average 1.6 GB in 2011 and 1.76 GB in 2012. In the case of stripping jobs a significant increase from 2.2 to 3.4 GB can be observed (see Fig. 4). This is due to the fact that in 2012 the number of stripping lines has been increased.

3. Detecting the Most Important Features

Having analyzed the workloads from the past, distributions have been obtained which can be used to predict future jobs. This section will analyze whether a better prediction can be given when more input features are taken into account rather than taking only the simple mean of prior job runtimes and memory footprints. Therefore, the correlations between runtime, memory footprint and the different job meta data have to be measured. Well known correlations are for instance:

- The higher the pile-up the more complex will be the reconstruction in terms of memory and runtime
- The higher the beam energy, the more particles will be generated and therefore the larger will be an event

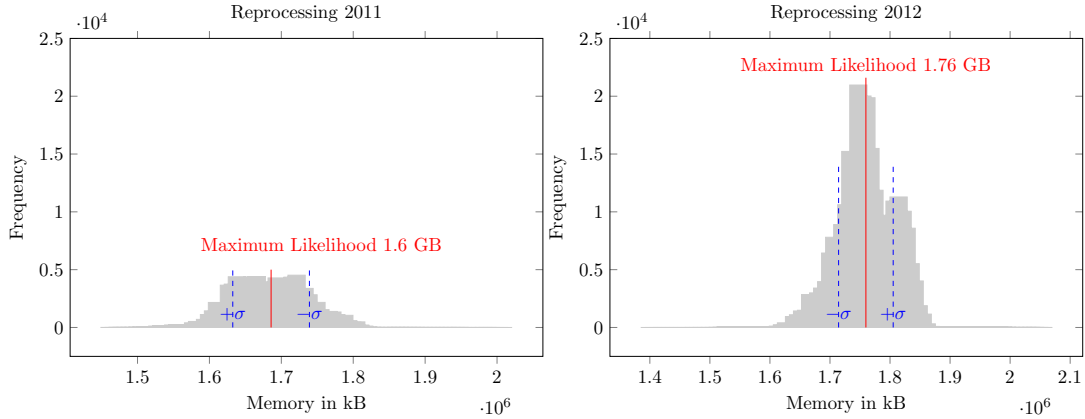


Figure 3. Memory requirements for reconstruction jobs [4]

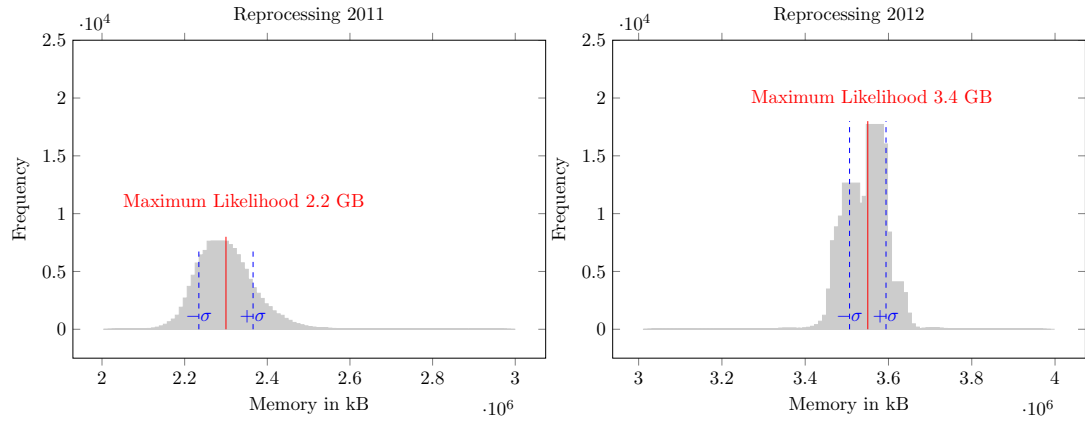


Figure 4. Memory requirements for stripping jobs [4]

In order to study the impact of these parameters on the runtime and memory footprint of jobs, the feature space has been normalized. Afterwards different prediction models have been applied. It appeared that linear regression (LR) was the model to which the data responded the best. After having found the best θ values for the corresponding model, it can be evaluated which features do not have a significant impact on the output features. Fig. 5 shows that the features avg. event size and avg. multiplicity have a low weight and can therefore be neglected. This is because these features are redundant. Taking this subset of features and estimating the runtime of jobs results in a 22% better estimation compared to a naive estimator like Maximum Likelihood Estimation (MLE).

In the case of stripping jobs, it appears that physics conditions like avg. luminosity and multiplicity do not have any impact on the runtime of these jobs (Fig. 6). This is expected, since these jobs sort reconstructed events into different output streams. Therefore, for the runtime it is more relevant, how many events have to be processed and how big the input files are. Taking such features into account results in a 25% improvement in prediction.

Taking into account more input features for predicting memory consumption of jobs, reached only slight improvements of about 2-6% compared to a naive estimator. This is most probably related to the fact that only the virtual memory of jobs is stored in the database. Since jobs in general request much more memory than they actually need, virtual memory is not the right parameter to consider.


| Normalize | Find best Θ | Remove small Θ | Evaluate RMSE |
|-------------------|--------------------|-----------------------|---|
| File Size | 0.80 | 1.05 | 2.16 |
| Avg. Event Size | 0.19 | x | |
| HEPSPEC | -0.97 | -0.97 | |
| Number Of Events | -1.33 | -1.55 | |
| Avg. Luminosity | 0.67 | 0.59 | |
| Avg. Multiplicity | -0.08 | x | |
| | | |  22% better than naive estimator like MLE |

Figure 5. Prediction model for runtime of reconstruction jobs


| Normalize | Find best Θ | Remove small Θ | Evaluate RMSE |
|-------------------|--------------------|-----------------------|---|
| File Size | -0.19 | -0.18 | 0.54 |
| Avg. Event Size | 0.70 | 0.62 | |
| HEPSPEC | 0.19 | 0.19 | |
| Number Of Events | 0.23 | 0.27 | |
| Avg. Luminosity | -0.08 | x | |
| Avg. Multiplicity | 0.003 | x | |
| | | |  25% better than naive estimator like MLE |

Figure 6. Prediction model for runtime of stripping jobs

4. Supervised Learning

This section evaluates how the different prediction models perform when only a small training set is available. When a new production is created, then similar jobs must be first found. The false prediction will be high in the beginning, because of effects which have not been seen in the training data. The supervised learning algorithm has the following steps:

- (i) Find similar jobs which have already run
- (ii) Predict requirements for the next k jobs using either (MLE/LR)
- (iii) When k jobs have finished, update the prediction formula with the new results obtained
- (iv) Repeat steps 2 and 3 until all jobs have finished

In the evaluation k has been set to 100 and jobs from the whole experiment year 2012 have been chosen. This means that the supervised learning algorithm is updated after each 100th job. Fig. 7 shows the accumulated error for the prediction of reconstruction jobs. As expected the false prediction is relatively high in the beginning. In the case of runtime it goes up to 5 HS06.s. But over time the error quickly decreases. Assuming that the model captures all influencing factors, then the accumulated error should either decrease or converge. This is clearly not the case for the runtime prediction as shown in Fig. 7, where the error obviously increases again. During an experiment year the trigger configuration is changed which leads to the fact that events are recorded which differ from previously recorded events. This can lead to an increased false prediction. For instance, Fig. 7 shows that by the end of the experiment year (after 250k jobs) a significant increase in the runtime prediction occurs in case of MLE. This happened because the jobs processed events which have been recorded under a minimum bias trigger. These events differ significantly from events recorded during the rest of the year. However, the linear regression model could predict the runtime of these jobs relatively well since it takes the physics conditions into account. In general taking more input features into account can deliver better estimates than MLE.

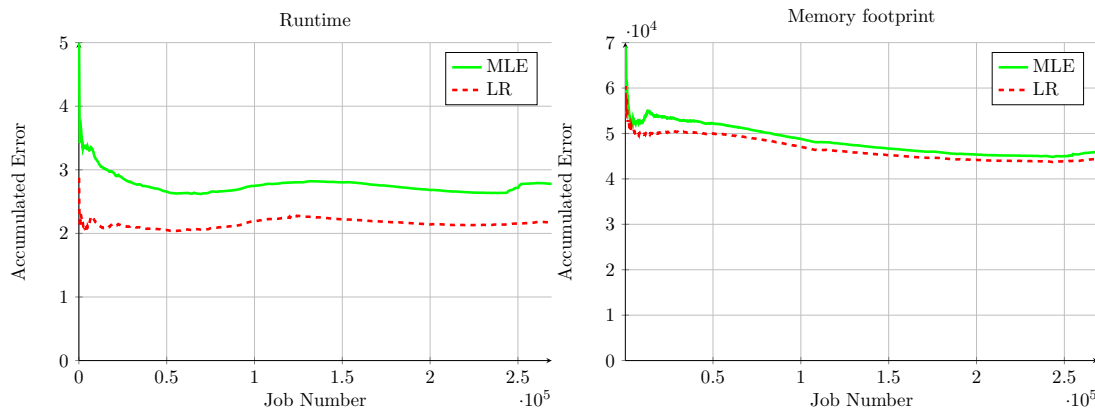


Figure 7. Accumulated error for the prediction of reconstruction job requirements (Update after $k = 100$ jobs) [4]

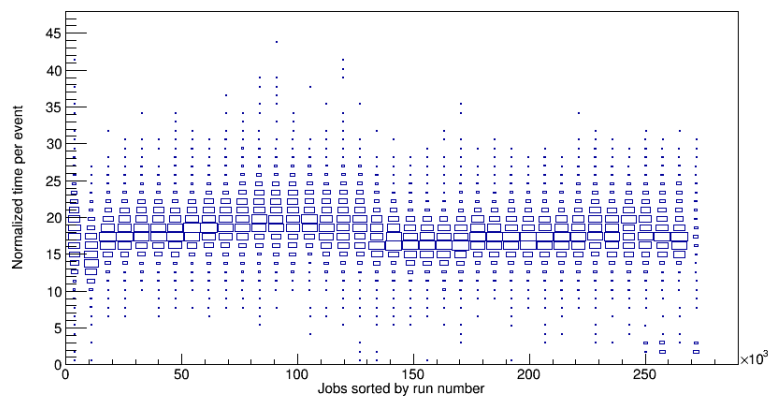


Figure 8. Normalized CPU Time per event of all reconstruction jobs from 2012 sorted by run number

Fig. 8 shows reconstruction jobs sorted by run number and the corresponding runtimes per event are grouped into smaller distributions. Large squares indicate the center of these distributions, smaller ones present the outliers. It becomes clear that the center of these distributions is fluctuating a lot during an experiment year. This is mainly due to different trigger configurations.

In case of memory footprint both models deliver similar results (Fig. 7). As discussed previously, this might be related to the fact that LHCb stores only the virtual memory of jobs in the database.

Fig. 9 shows the results obtained for stripping jobs. Runtime prediction follows a similar curve as in the case of reconstruction jobs. For the memory footprint a significant outlier can be observed. This is due to huge combinatorics which has lead to a memory footprint several times larger than what is used to be the average. Even taking into account meta data like file size and number of events, does not allow to foresee such a memory increase. Consequently, this outlier can be seen in both models.

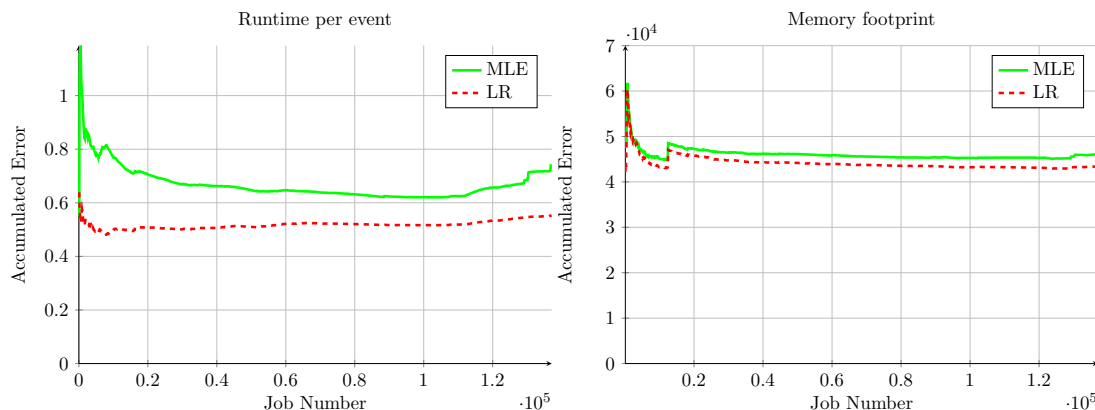


Figure 9. Accumulated error for the prediction of stripping job requirements (Update after $k = 100$ jobs)

5. Conclusion

This paper has shown how job requirements can be estimated at the example of LHCb. The main idea is to retrieve information about jobs that have already successfully finished and to create an estimator. Section 2 has shown a workload analysis from LHCb reprocessing productions of 2011 and 2012. Based on the derived distributions, a naive estimator has been proposed in order to predict future jobs. Section 4 and 3 have shown that taking job meta data like file size, number of events, avg. multiplicity etc. can help to significantly improve the prediction of job requirements up to 25% compared to a naive estimator. The prediction model must be able to regularly update itself. Therefore, a supervised learning algorithm has been presented that starts with a small set of training data and regular updates ensure that it can learn over time from changing conditions. Both models can be easily integrated in the Workload Management System and can support the Production Manager in the decision making.

References

- [1] Multicore Task Force. <https://indico.cern.ch/event/296031/material/slides/0?contribId=0>, 2014.
- [2] RunDB: Run 103127. <https://lbrundb.cern.ch/rundb/run/103127/>, 2014.
- [3] Zoltan Mathe. Feicim: A browser and analysis tool for distributed data in particle physics. PhD thesis, U. Coll., Dublin, May 2012.
- [4] Nathalie Rauschmayr. Optimisation of LHCb Applications for Multi- and Manycore Job Submission. PhD thesis, Institute of Technologie, Karlsruhe, October 2014.
- [5] Samir Cury Siqueira. Event processing time prediction at the CMS Experiment of the Large Hadron Collider. Technical Report CMS-CR-2013-375. CERN-CMS-CR-2013-375, CERN, Geneva, Oct 2013.
- [6] Igor Sfiligoi. Estimating job runtime for CMS analysis jobs. Technical Report CMS-CR-2013-340. CERN-CMS-CR-2013-340, CERN, Geneva, Oct 2013.
- [7] Warren Smith, Ian T. Foster, and Valerie E. Taylor. Predicting application runtimes using historical information. In Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, IPPS/SPDP 98, pages 122142, London, UK, UK, 1998. Springer-Verlag.
- [8] Richard Gibbons. A historical application profiler for use by parallel schedulers. In Dror G. Feitelson and Larry Rudolph, editors, Job Scheduling Strategies for Parallel Processing, volume 1291 of Lecture Notes in Computer Science, pages 5877. Springer Berlin Heidelberg, 1997.
- [9] Byoung-Dai Lee and Jennifer M. Schopf. Run-time prediction of parallel applications on shared environments. In Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, pages 487491, 2003.
- [10] Engin Ipek, Bronis R. Supinski, Martin Schulz, and Sally A. McKee. An approach to performance prediction for parallel applications. In Jose C. Cunha and PedroD. Medeiros, editors, Euro-Par 2005 Parallel Processing, volume 3648 of Lecture Notes in Computer Science, pages 196205. Springer Berlin Heidelberg, 2005.