System Performance and Cost Modelling in LHC computing

Catherine Biscarat[1], *Tommaso* Boccali[2], *Daniele* Bonacorsi[3], *Concezio* Bozzi[4,5], *Raul* Cardoso Lopes[6], *Davide* Costanzo[7], *Dirk* Duellmann[4], *Johannes* Elmsheuser[8], *Eric* Fede[9], *José* Flix Molina[10], *Domenico* Giordano[4], *Costin* Grigoras[4], *Jan* Iven[4], *Michel* Jouvin[13], *Yves* Kemp[12], *David* Lange[14], *Helge* Meinhard[4], *Michele* Michelotto[15], *Gareth Douglas* Roy[16], *Andrew* Sansum[17], *Andrea* Sartirana[18], *Markus* Schulz[4], *Andrea* Sciabà[4], *Oxana* Smirnova[19], *Graeme* Stewart[4], *Andrea* Valassi[4], *Renaud* Vernet[20], *Torre* Wenaus[8], and *Frank* Wuerthwein[21]

[1]Institut National Polytechnique de Grenoble and CNRS/IN2P3, Grenoble, France
[2]INFN Sezione di Pisa, Pisa, Italy
[3]INFN Sezione di Bologna, Università di Bologna, Bologna, Italy
[4]CERN, Geneva, Switzerland
[5]Università e INFN, Ferrara, Ferrara, Italy
[6]Brunel University, London, United Kingdom
[7]Department of Physics and Astronomy, University of Sheffield, Sheffield, United Kingdom
[8]Physics Department, Brookhaven National Laboratory, Upton, NY, USA
[9]Laboratoire d'Annecy de Physique des Particules (LAPP) and CNRS/IN2P3, Annecy, France
[10]Centro de Investigaciones Energéticas Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[11]School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom
[12]Deutsches Elektronen-Synchrotron, Hamburg, Germany
[13]LAL, Université Paris-Sud and CNRS/IN2P3, Orsay, France
[14]Princeton University, Princeton, NJ, USA
[15]INFN Sezione di Padova, Università di Padova, Padova, Italy
[16]SUPA - School of Physics and Astronomy, University of Glasgow, Glasgow, United Kingdom
[17]STFC Rutherford Appleton Laboratory, Didcot, United Kingdom
[18]Laboratoire Leprince-Ringuet, Ecole Polytechnique, CNRS/IN2P3, Université Paris-Saclay, Palaiseau, France
[19]Lunds Universitet, Fysiska Institutionen, Avdelningen för Experimentell Högenergifysik, Box 118, 221 00 Lund, Sweden
[20]Centre de Calcul de l'IN2P3, Villeurbanne, Lyon, France
[21]University of California, San Diego, La Jolla, CA, USA

Abstract. The increase in the scale of LHC computing expected for Run 3 and even more so for Run 4 (HL-LHC) over the next ten years will certainly require radical changes to the computing models and the data processing of the LHC experiments. Translating the requirements of the physics programmes into computing resource needs is a complicated process and subject to significant uncertainties. For this reason, WLCG has established a working group to develop methodologies and tools intended to characterise the LHC workloads, better understand their interaction with the computing infrastructure, calculate their cost in terms of resources and expenditure and assist experiments, sites and the WLCG project in the evaluation of their future choices. This working group

started in November 2017 and has about 30 active participants representing experiments and sites. In this contribution we expose the activities, the results achieved and the future directions.

1 Introduction

The computing infrastructure for the LHC experiments, managed via the WLCG project [1], has been successfully operating since 2008, with a good match between the resources needed by the experiments and those made available by the funding bodies. In the last few years though it became increasingly clear that Run 3 (for ALICE and LHCb) and Run 4, or HL-LHC (for ATLAS and CMS) will determine very significant increases in the scale of computing, which will not be simply accommodated by technological improvements in the likely scenario of a flat budget evolution. Simply extrapolating the current software performance to the expected trigger rates and average pile-up produces a $O(10)$ discrepancy between the needed and the affordable levels of computing and storage capacity. For this reason, "revolutionary" changes in the software and the computing models of the experiments will be absolutely necessary.

The need of a joint working group between WLCG and the HSF [2] dedicated to the study of performance and cost of computing became apparent and it finally started at the end of 2017, with a long term roadmap that extends to the start of HL-LHC. About thirty members, from experiments, sites and IT and software experts participate to the group activities. The initial focus has been on improving the understanding of current workloads and to establish methodologies and tools to analyse their performance; however, some thought has already been given to the exploration of future scenarios and to try to quantify the gains that could be achieved through paradigm changes. Currently the most important areas of work are:

- the collection of reference workloads from each experiment, to conduct performance studies in controlled and repeateable conditions;
- the definition of the metrics that best characterise the applications and the implementation of tools to measure them;
- the adoption of a common framework for estimating resource needs:
- the adoption of a common process to evaluate the cost of an infrastructure as a function of the experiment needs;
- preliminary studies to explore possible savings in different areas.

2 Workload characterisation and metrics

To model the behaviour of the workloads of the LHC experiments it is essential to understand how the different capabilities that a site provides impact the throughput of a specific workload. To limit the complexity of a performance model, it is desirable to minimise the number of performance characteristics; this requires to find a set of "orthogonal" metrics to which the workload throughput is sensitive. This knowledge can then be used by software developers to avoid bottlenecks by balancing resource usage, and by site managers to optimise their expenditure. The balance between the amount of memory per core, the disk performance and the speed of the memory can be optimised on the basis of the characterisation of the relevant workloads. Figure 1 illustrates this relationship.

Finding such a metric set if far from trivial, given the complexity of workloads and their production environments. We started from a representative collection of reference workloads

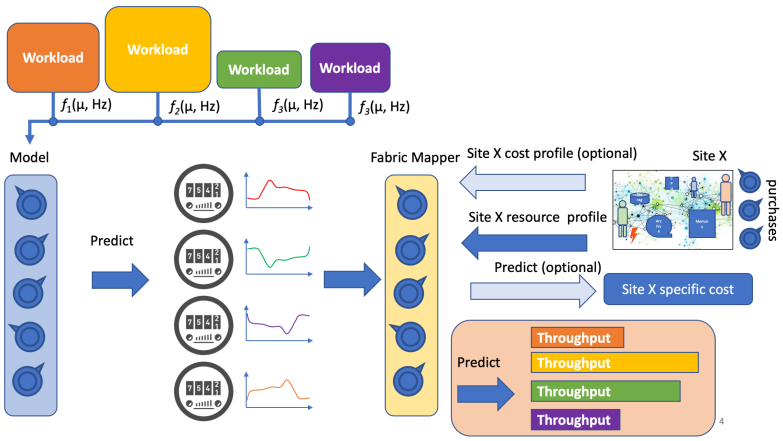

Figure 1. Workload resource requirements of the workloads depend on the pileup and the trigger rate. With this information the model predicts the demands on the different resources. The Fabric Mapper abstracts the characteristics of a fabric and produces estimates for throughputs. Optional sites can add their local cost structure to the model

Figure 2. Memory consumption over the execution of an ATLAS Digi-Reco job; the different processing steps are clearly visible. The periods with very low memory consumption are due to the merging of intermediate files

for each experiment. Then, we listed several metrics, grouped in various categories (CPU, I/O, memory, storage, swap and network) and assessed their relevance. Finally we focused on those that can be measured without significant interference with the running workload on nodes. A tool, PrMon [3], originally developed by ATLAS, has been generalised and allows to measure time series of a large set of parameters. Prmon uses information collected by the operating system and does not require any instrumentation. Figure 2 shows, as an example, measurements of the memory usage during the execution of a workload that performs digitisation with pile-up data and reconstruction for Monte Carlo events. Prmon allows to estimate which capabilities of a system limit the throughput and how efficiently multi-process and

multi-threaded version of the programs use their allocated cores. To study the dependencies on latency, bandwidth and memory restrictions, another tool has been developed that runs workloads repeatedly with increasingly restricted bandwidth, memory and increased latency, measuring the throughput for each configuration.

Another tool, Trident [4], was developed to provide a convenient access to the information contained in CPU hardware counters. These counters provide information on the level of parallelism exploited by a workload, memory access, utilisation of caches and vector capabilities, etc. in an abstract and CPU-model dependent way. This gives the developer an estimate on how much improvement is at best possible in exploiting the available resources and quantitatively understand the limiting factors. The site managers can use the insights gained from Trident in decisions concerning trade-offs between different capabilities, like faster memory vs. size or system disk speed and network.

2.1 Next steps

While the time series measured by Prmon give valuable insights, they cannot be directly used as input for modeling the behaviour of the workloads. Work on this process has started to extract parameters from metric time series, as processing can be described as sequence of steps, each one looping over events. During each of these steps the resource usage can be described by a small number of parameters and the number of processed events. This, in combination with the measured dependency on latency, bandwidth and memory, will allow to predict the throughput of the current workloads under different conditions.

For the resource modelling under future running conditions the dependency on the average pile-up has to integrated into the model. For data size and simulation time, the dependency is at most linear. For reconstruction, an extrapolation to HL-LHC levels is currently very uncertain and it would be exponential with the current algorithms; ATLAS and CMS planned detector upgrades will help to address this problem. ATLAS by introducing tracking layers that are closely spaced, so that they provide precise starting vectors and CMS by high precision time resolution that allows to distinguish between hits from different collisions.

3 Resource estimation

3.1 Motivations

In order to sensibly plan their activity and estimate their costs, experiments need to translate the requirements of their physics programmes into computing resources needs. They thus need a modelling tool that takes as input the details of the physics activity and the features of the computing model and outputs the amount and characteristics of the computing resources required.

Currently, all LHC experiments have their own tools for this task, often some complex spreadsheets. Our purpose is to provide a common framework that could standardise, generalise and possibly improve what each experiment does, in its own specific way, today. This framework should be as generic and customable as possible, it should define a schema for the parameters that describe the expected activity and the characteristics of the computing model, and provide some standard calculations on these parameters. Besides providing the experiment with a standardised way to compute their needs, such framework will allow us to easily play with different computing model scenarios and explore potential gains.

3.2 Current Status

A first version of the framework [7] was obtained by forking, refactoring and slightly generalising some code used by CMS to estimate HL-LHC requirements [8]. It is a set of modules and scripts that reads some inputs files and produces plots and tables with the required storage and CPU resources. Inputs are loaded hierarchically so that we can easily overwrite a generic set of definitions with different special scenarios. This is handled by the classes and functions defined in the *ResourceModel* module. Parameters include:

- LHC parameters: trigger rates, live fractions, shutdown years, etc.
- Computing model: event sizes and processing times, software improvement factors, etc.
- Storage model: numbers of versions, replicas, etc.
- Infrastructure: capacity model, T1 disk and tape, etc.

The framework computes the number of data and Monte Carlo events per year (*EventModel* module) using the LHC parameters and some physics information like the required montecarlo/data ratio. The functions defined in *CPUModel* and *StorageModel* modules, then, translate the number of events into CPU and Storage requirements for the different activities (reconstruction, montecarlo, analysis). The module *ModelOut* defines functions to create plots and tables.

3.3 Future Work

This first version of the framework elicited strong interest from other LHC experiments ad has been agreed as a common basis for future development. Generalising it to all LHC experiments will require rewriting some of the most CMS-specific parts and making the parameters schema more flexible. Even if a common framework was not finally adopted by all LHC experiments, having them to adopt a similar approach will greatly benefit the ability to produce consistent and flexible resource estimates.

Besides the inclusion of all LHC experiments, other lines of development have been proposed. For example, the model should be allowed to have generic time granularity (it currently only deals with yearly data) and should include an estimation of network resources.

4 Site cost estimation

4.1 Context

The purpose of site cost estimation is to understand and measure, at a global scale, what the data centres typical expenses are, and predict what they may be in years from now. The approach chosen in this context is to model those expenses, taking into account the diversity of national contexts across sites in terms of funding, procurement procedures, and local market conditions. These results will help experiments plan new computing strategies that will improve the cost-effectiveness of their resource usage.

4.2 First results

Some of the first elements to address are the diversity of expenses across sites, and the definition of what these expenses are. A simple and quick exercise was made by four candidate sites, aiming to get a first estimate of the financial cost to run a given workflow and to store a given amount of data, in terms of IT resources and power consumption. The answers to this

exercise happened to be significantly different from site to site, up to a factor of 2. The reasons were found to originate from different aspects: the intrinsic variety of costs, the measurement method, and the understanding of what a given metric means.

Those results showed clearly the need of a consolidated and common model and method to measure costs across WLCG data centres. Here are a few examples to illustrate this point. One may consider the cost of a server providing a given capacity, including (or not) the equipement that is shipped with it (rack, switch, adaptor etc.). One may also consider that one capacity unit of tape storage includes (or not) the investments made in library, drives, disk cache etc. that are needed for such system to work properly. Finally, the measurement of an electrical consumption may include (or not) the Power Usage Effectiveness of the data centre in order to take into account UPS or HVAC system contributions to the final power bill.

Consequently, it becomes fundamental in the context of site cost estimation to establish precise definition of the cost-related metrics in order to build a reliable model.

4.3 Next steps

A preliminary attempt to address data centre resource TCO through cost modeling was shown in reference [9]. This model assumes that a data centre invests year after year a constant budget in the following assets: batch system, disk storage and tape system capacities. If this hypothesis is satisfied (even roughly), one can show that budget (B) and available capacity (K) over time are bound together by a quantity (c^*) that depends on hardware cost evolution and lifetime:

$$B(t) = K(t) \times c^*(t) \tag{1}$$

In the case where hardware costs by unit of capacity decrease exponentially over time with a rate r (e.g. 0.2 for a 20% yearly decrease) and hardware is replaced after τ years, one can show that

$$c^*(t) = c(t)\frac{r}{1-(1-r)^\tau} \tag{2}$$

and the time evolution of the site capacity for a flat budget turns out to be exponentially increasing.

This model however does not address all the components of a TCO, like manpower. Site cost studies may leverage this model to estimate the financial impact of a variation in the usage that experiments make of data centre IT resources. But the quantity c^* may differ significantly from site to site, so an extension of this model at global scale will have to take into account as many as site-dependent parameters as possible to establish c^*.

In order to get a better idea of the variations of the expenditure at different sites, a survey is being conducted at all Tier-1 sites, although the results are not yet available.

5 HL-LHC and areas of improvement

Since the start of the LHC the community constantly improved the throughput of the main workflows; however, studies conducted by the Understanding Performance Team and the Cost and this working group still found several areas in which potential improvements could be achieved [**?**].

5.1 Compiler and software improvements

WLCG sites provide a variety of CPU models, and almost all code is built to run on the oldest available architectures in the system. This and recent advancements in compiler techniques motivated further studies.

From studies conducted on current GEANT simulation, reconstruction and NLO generator code, gains by compiler and link-based optimisations are around 20-25%. This includes the compilation of the code for individual target CPUs, the use of Intel's commercial compiler and the use of feedback-directed optimisation (AutoFDO from Google and Intel). Attempts of compiler-based vectorisation of current production codes showed none or minimal improvements. Reducing the overhead of shared libraries by building large libraries resulted in significant gains on older architectures (e.g., a 10% improvement on Ivy Bridge for the ATLAS simulation code). Comparisons between the Intel compiler and current gcc versions showed no clear differences. It has to be noted that the use of feedback-directed optimisation requires that the objects are build statically, which for some of the LHC codebase is not trivial.

From profiling the code and detailed analysis of the dynamic use of memory allocation, it is known that the current code spends up to 25% of the time on memory/object management, due to the frequent creation and destruction of small objects [5]. A 60-90% of allocations exists for less than 100μs and are smaller than 64 Bytes. By using more adequate techniques and strategies for object management, such as object pools, libs for large and static vectors, this could be reduced to less than 10% with relative minor code refactorisation. At the same time the data structure layout can be improved for more efficient utilisation of caches and improved memory access.

The current HEP code executes on modern cores significantly less than 2 instructions per cycle (0.8-1.5) . With the large vector registers provided by current cores the theoretical limit can be above 20 instructions/cycle for code that can be completely vectorised. However, tuned complex code for HPC systems can reach values of about 4. This can be seen as an upper limit for our code base. Approaching this level of utilisation requires at least a change of the used data structures and a re-implementation/factorisation of the algorithms used in the HEP code base. This is therefore best taken into consideration when new algorithms are designed and implemented.

Gains from new algorithms and the impact of new detector components can't be treated here. As the development of the new, cellular automaton based, HLT track reconstruction code for the ALICE HLT has shown, large factors O(100) can in some cases be achieved [**?**].

5.2 Storage

The LHC community has recently agreed to a scenario where managed storage is consolidated at a few, very large sites (the "data lake") as the most promising to achieve significant cost savings.

For what concerns operational effort, the 2015 WLCG site survey [6] showed that on average Tier-1 sites require 2.5 FTEs for operating storage and Tier-2 sotes 0.75 FTEs, with a weak dependence on the amount of storage (+15% FTEs for doubling the storage). By concentrating managed storage at a few sites and using only disk caches (much simpler to manage) at most sites, one can estimate a decrease of the overall number of FTE for storage operations from around 100 to around 60, that is, a 40% reduction.

The vast majority of disk (80%) is used for data formats used for analysis. Data popularity studies show that on average datasets exist at about 2 sites and are accessed less than 10

Table 1. Summary of Cache and Latency Studies

Workload	Condition	Latency (ms)	Method	Rel. time
ATLAS Digi-Reco	data on node	0	running local	1
ATLAS Digi-Reco	data remote	25	running local	1.9
ATLAS Digi-Reco	data remote, empty cache	25	Xcache	1.08
ATLAS Digi-Reco	data remote, pop. cache	25	Xcache	1.04
ATLAS Derivation	data on node	0	running local	1
ATLAS Derivation	data on EOS	< 1	running local	1.02
ATLAS Derivation	data remote	25	running local	8.3
ATLAS Derivation	data remote, empty cache	25	Xcache	1.05
ATLAS Derivation	data remote, pop. cache	25	Xcache	1.03
CMS Digi-Reco	data local	0	added latency	1
CMS Digi-Reco	data local	5	added latency	1.01
CMS Digi-Reco	data local	10	added latency	1.04
CMS Digi-Reco	data local	20	added latency	1.11
CMS Digi-Reco	data local	50	added latency	1.24

times over a period of six months, with most accesses happening in the first month. We can argue that data needs to be stored only once at the large storage sites, and just be temporarily cached as needed depending on the client load. Less popular data may be completely purged from disk and kept on tape.

How much saving can be expected depends on the retention strategy and the use of hierarchical storage (with tape costing about 1/4 than disk storage). The savings from replacing some amount of disk with tape are partially offset by the need of more tape drives and added complexity in the data migration (the latter assumed not to be substantial due to the sophistication of the current data management systems).

Again, analysis of popularity data and storage system monitoring indicates that only a small fraction of the produced data is active at any time and a significant fraction (15-20%) of the data could be moved to a different storage layer.

The concentration of data at a few sites requires to limit the impact of bandwidth limitations and latency on the throughput of applications. We conducted measurements of the impact of latency to the throughput of various workloads leading to the understanding that for latencies up to 25ms the reduction of throughput can be limited to 5% when using Xcache as an additional layer. Some workloads already manage latency on the client level very well. The results are summarised in table 1.

Data is also replicated within storage systems to ensure a high degree of reliability, either by replication (more performant but expensive) or some form of error encoding (cheaper but less performant). Even larger savings can be achieved by not using storage redundancy at all and re-staging from tape, or possibly even regenerating, any lost data.

Based on the observed disk failure rate of an individual disk in the CERN EOS system, which is about 1% per year, the relative total cost of storage and computing at CERN (around 4 HS06/TB), the amount of CPU time to generate AOD events ($\sim$ 850 HS06$\times s$) and their size ($\sim$ 400 kB), one can naively estimate that the computing cost to re-generate the AOD data lost to disk failures is $\sim$ 20% of the cost to make the storage that contained it fully redundant.

In reality, most of the times the lost data would be replicated elsewhere, although it would not be the case for intermediate data sets; one can conservatively estimate that 30-50% of the disc costs can be saved. For this approach to be efficient and effective the process of recreating

Table 2. Estimated potential gains and associate efforts

Change	Effort: Sites	Effort: Users	Potential Gain
Managed storage only at sites	some on large sites	little	-40% ops effort
Reduced data redundancy	some on larger sites	some	-30-50% disk cost
Scheduling and site inefficiencies	some	some	+10-20% CPU
Reduced job failure rates	little	some - massive	+5-10% CPU
Compiler and build improvements	none	little - some	+15-20% CPU
Improved memory usage	none	some	+10-15% CPU
Exploiting modern CPU arch.	none	massive	+100% CPU

individual files has to be automated, which is highly desirable since other failure modes lead to data loss on a comparable scale.

5.3 Gains from improvements in operations

Scheduling inefficiencies in WLCG arise from many reasons. Either due to a mismatch between cores in a system and memory requirements, mismatch between requested cores and cores grouped on nodes (the tessellation problem), batch system inefficiencies, pilot service inefficiencies, delays due to data staging, I/O waits etc.. These inefficiencies are different for different sites and workloads. Site managers have identified some of these problems and estimates range from 20% - 30% of resources being lost due to one or another scheduling inefficiency. It has been shown that with advanced backfilling and more complex job placement strategies, efficiencies above 90% can be reached, at the expense of higher complexity of site and experiment workload management systems. To efficiently use short usage windows the granularity of workloads has to be increased. Another source of scheduling inefficiencies stems from breaking the processing chains from raw data to data analysis objects into several individual steps that exchange data via files. Experiments have started to chain these steps, but often still rely on intermediate local files. Especially when parallel processing threads/processes write individual files, the merging steps create inefficiencies, as they are single threaded. By using shared writers these inefficiencies can be reduced. The overall impact is difficult to measure, but from ATLAS pileup/digitisation/reconstruction chains activity logs it can be estimated to be about 5%.

Losses due to intermittent job failures are for complex workflows unavoidable. Currently most of the steps can recover from failure with the help of automated retry and failover mechanisms. Nevertheless complete jobs are sometimes run up to 20 times before successful completion. The overall loss in walltime due to job failures is between 10 and 15%. Improved procedures to make jobs more resilience or fail very early can reduce these losses, but will not eliminate them.

Table 2 summarises the different identified potential gains.

6 Conclusions

Conclusions.

$\leq$ 0.5 pages.

References

[1] http://wlcg.web.cern.ch/

[2] https://hepsoftwarefoundation.org/
[3] https://github.com/HSF/prmon
[4] S. Muralidharan, D. Smith, *Trident: A three pronged approach to analysing node utilisation*, EPJ Web of Conferences
[5] S. Kama and N. Rauschmayr, J. Phys. Conf. Ser. **898** (2017) no.7, 072031
[6] M. Alandes Pradillo *et al*, J. Phys.: Conf. Ser. **664** 032025
[7] https://github.com/sartiran/resource-modeling/tree/wlcg-wg-new
[8] https://github.com/kenbloom/resource-modeling
[9] Renaud Vernet, Journal of Physics: Conference Series, **664**, 052040 (2015)