LHC Computing: past, present and future

Philippe Charpentier[1]

[1] CERN, EP Department, 1211 Genève 23, Switzerland

Abstract. Although the LHC experiments have been designed and prepared since 1984, the challenge of LHC computing was only tackled seriously much later, at the end of the '90s. This was the time at which the Grid paradigm was emerging, and LHC computing had great hopes that most of its challenges would be solved by this new paradigm. The path to having functional and efficient distributed computing systems was in the end much more complex than anticipated. However, most obstacles were overcome, thanks to the introduction of new paradigms and a lot of manpower investment from the experiments and from the supporting IT units (for middleware development and infrastructure setup). This contribution is briefly outlining some of the biggest hopes and disillusions of these past 20 years, and gives a brief outlook to the coming trends.

1 Introduction

The LHC computing challenges only started to be considered seriously long after detectors had been designed and their construction initiated. It was then realised that the model used until then at CERN (including for the LEP experiments) could not apply to the LHC due to the much larger datasets collected by the experiments. Namely it was realised that centralised data processing and analysis mostly located at CERN would not be possible, even using dedicated infrastructures as were used by the LEP experiments.

A review of the computing needs for LHC experiments had been conducted at the end of the '90s that concluded that the computing resources (both CPU and storage) required to handle such large datasets were far beyond what could be provided at a single location and funded by a single agency [1].

Software development for the LHC experiments had started much before, and it was clear at that time that new programming languages would have to be used. Fortran that had been the language widely used in HEP experiments until then was no longer suitable for handling the complexity of these new experiments, and the turn had been taken in the mid-90's to move to object-oriented programming. This was not an easy move but despite some resistance, the LHC experiments converged to programming mostly in C++ language.

As for every change of cultural background, the early years were difficult. Great hopes had been put into using object databases for persistency of the C++ objects [2], but other experiments like Babar that had to enter into production at that same time encountered the limitations of such an approach. In the end, the pioneering work of René Brun and co-workers on ROOT [3] was recognised as the most appropriate solution, not only for the data persistency, but also for providing physicists with analysis tools.

The processing and analysis of the petabytes of data produced every year by the LHC experiments remained though the biggest challenge.

2 The MONARC model

A model for distributed computing had to be designed, and in the late '90s a tiered model emerged under the name of MONARC [4].

As the LHC experiments are all located at CERN, this site is obviously playing a central and particular role, being the source location of the data to be processed. Its infrastructure needs to be tailored to the requirements of recording and storing the very precious data collected by the experiments. Any data loss or failure to record real raw data represents an inefficiency in using the LHC collider and its experiments. CERN was thus assigned the Tier0 central position in the MONARC model.

The MONARC model was defining several levels of tiers. The Tier1s were meant to be mostly national centres with large capacity of storage, in particular custodial (i.e. tape) storage. They must have a very high level of availability and reliability as they would hold also precious data: a second replica of the raw data as a backup to the Tier0 storage, but also master replicas of derived datasets (reconstructed data, and any further derived dataset used for physics analysis). Tier1s were also supposed to be supporting the Tier2s in their region.

Tier2s were smaller regional centres providing disk storage and CPU capacity. In the MONARC model, they were supposed to be mostly used by physicists to analyse the data that are replicated from the supporting Tier1s. CPU resources at all tiers were planned to also be used for simulation activities that constitute by far the largest usage of CPU resources in HEP experiments.

3 The Grid paradigm

About at the same time, I.Foster and C.Kesselman proposed a model to implement distributed computing that they called *The Grid* [5].

The basic idea is brilliant and elegant: the Grid is supposed to look to its users like a single large computing centre, in which storage and compute resources are seamlessly accessed without the users having to wonder about their location or their peculiarities. The contributing computing centres are interconnected using powerful networks, and they are powered with a layer of software that allows this seamless access as well as their interoperability (the Grid middleware).

The HEP community picked up on this idea in 2000 at a launching meeting just after the CHEP conference in Padova. At the time some middleware had already been developed in the framework of the Globus project [6], but much more was necessary for the LHC computing. Therefore, an R&D program was launched, sponsored by the European Union: the European DataGrid (EDG) [7]. This program was due to tackle all aspects of the Grid paradigm; therefore, several work-packages were created to cover this broad spectrum of needs. This middleware development was further consolidated in the EGEE [8] and EMI [9] European projects.

4 The Worldwide LHC Computing Grid

In order to coordinate the developments made in the EU DataGrid project, in the LHC experiments and to involve the computing centres that had expressed interest in participating in the adventure, the LCG (LHC Computing Grid) initiative was launched in 2002. Its current status and structure can be found in [10].

In the US, several early Grid projects were launched at about the same time. They led to the Open Science Grid (OSG) [11] that has coordinated the development of various Grid middleware in the US and the participation of US laboratories and universities since 2004.

In Europe the LCG, initially led by Les Robertson, was considered at the CERN management level as yet-another collaboration, at the same level as the experiments' collaborations. Therefore, the LCG is overseen by the LHC Committee, and it has its own internal organisation (Collaboration Board, Project Execution Board now Management Board, Grid Deployment Board, Software and Computing Committee…).

The LCG has been organised around 4 large "areas": middleware, deployment, fabric and applications. The LCG is much more a coordinating body, as middleware was supposed to be developed in the external projects (EU DataGrid to start with, soon followed by EGEE (Enabling Grid for E-sciencE), then EMI (European Middleware Initiative)); deployment is ensured by the sites themselves, but clearly this had to be coordinated such that all sites efficiently cooperate; the fabric area was meant to cover in particular the CERN fabric, due to its central role where data are produced; the applications area was coordinating the experiments' efforts as well as common software developments such as ROOT or other packages used by several of the experiments.

The LCG was envisioned to be built on top of underlying Grid infrastructures, each handling the coordination of its participating sites. In Europe and partner regions the sites started being operated under the auspices of EGEE (now EGI) and NorduGrid projects, while in the US and beyond sites joined the already mentioned OSG. To underscore the importance of this global partnership, the LCG was renamed Worldwide LCG (WLCG).

The LHC experiments expressed their initial set of requirements to the Grid under one of the DataGrid work-packages in a document called HEPCAL (High Energy Physics Common Application Layer) [12]. Not only providing input from the experiments, it was jointly written with early testers of the Grid middleware and Grid experts. This document, produced in 2003 and updated in March 2004 was presented to the developers' work-packages. Although considered as an interesting document, the requirements it contained were not fully considered for influencing the middleware developments that, at the time, had already been rather advanced [13].

In parallel with the middleware developments, experiments had started to gain experience in using the Grid infrastructure and the available middleware. Several drawbacks had been identified and in particular two of the experiments (ALICE with AliEn and LHCb with Dirac) had developed a Grid access layer based on a set of services, following more or less the same architectural baselines. A working group on further assessment of requirements had been spawned by the WLCG, called ARDA (A Roadmap to Distributed Analysis) [14]. Based on the architecture of AliEn [15] and DIRAC [16], the ARDA requirement document was proposing an architecture for implementing the HEPCAL requirements. Despite this document being considered even more interesting than HEPCAL, it was never considered by developers for re-orienting their middleware developments.

It is interesting though to notice that the current Grid implementations used by the LHC experiments fulfil those requirements from HEPCAL and are based on the architecture outlined in the ARDA document.

5 The dreams and the nightmares

5.1 Workload management

The dream of Grid computing is that from a user's perspective it looks like a very large homogeneous farm of computers. The internal heterogeneity between the various centres is hidden by the middleware layer. The heterogeneity is due to the fact that despite the fact that the computing centres all agree to participate in the collective effort of the Grid, each has the possibility to decide on many topics: CPU hardware type (subject to procurement rules), batch system, operating system version etc. Very often the choice in these matters is historical and related to the knowledge and preferences of the staff as well as to the fact that these centres serve other purposes than the Grid, and they don't want to change the other users' habits.

One of the main challenges when submitting a task (a.k.a. a job) to the Grid is to make a brokering decision on where this job is going to run. The brokering depends on several factors: if the job needs to access data files as input, the natural choice is to run the job at a CPU resource close to the storage where the files are stored; but when there are multiple choices, jobs should be directed to sites where the expected completion time is the shortest, i.e. where the number of queuing jobs is minimal.

Taking brokering decisions requires thus to know where files are located, and to know at any point in time the status of the CPU resources at all sites. File location is achieved by means of a replica catalog, that contains all existing files for a given VO and their location. The replica catalog needs to be able to handle millions of files with several replicas each, and the service must scale such that brokering can be done quickly. But the most complex constraint is to know at any point in time the state of the computing farm (number of running and waiting jobs in each queue for example). This knowledge cannot be instantaneous, and in practice, sites update the information in a central information service every couple of minutes only. During that time, the state of the resources may have completely changed, as many jobs may have been submitted. This leads to wrong brokering decisions being taken: very often when a site has free resources, the tendency is that all jobs will be submitted to that site, thus creating a large backlog of waiting jobs.

Another problem that soon appeared is that the brokering of jobs was consuming a fair amount of time, and the "resource brokers" were not able to handle more than a few thousand jobs each, since they were in charge of supporting them from submission to completion. The solution was to use a large set of brokers, but as they were not communicating with each other, they were all seeing the same state of sites, thus tending to broker jobs all to the same site until the status of the site was updated in the information system.

Despite substantial improvements brought in the EGEE Workload Management Systems (WMS) over the initial Resource Broker (developed in EU DataGrid), the brokering problem was one of the major concerns for job submission.

This brokering problem was overcome initially by ALICE and LHCb when they first introduced the concept of "pilot jobs" in their Grid access layer (AliEn and DIRAC, already mentioned for ARDA). Pilot jobs are regular Grid jobs that deploy a simple script (the *pilot*), for example via a WMS service, or any other means on computing resources. Pilots can find out the capabilities of the worker node on which they are running (operating system, location, available memory, local storage…), and use this information for matching a real job (called "payload") that is stored in a central task queue.

The deployment of pilots creates an overlay of the computing resources and the late assignment of jobs ensures that the job can immediately be executed, as a pilot only matches jobs that fulfil the capabilities of the worker node. This change of paradigm from a "push" model to a "pull" model was a major breakthrough that improved dramatically the efficiency of the jobs. It was then first adopted by ATLAS with the introduction of PanDA in 2005 [17], and also by CMS as well several years later in their GlideinWMS system.

Also, pilots can be deployed not only as jobs, i.e. through computing elements and batch systems, but can also be launched by virtual machines at instantiation time, or simply as a script on any computer, for example to run a test.

Today all LHC experiments make use of pilots for running jobs on their distributed computing infrastructure, and for example WMS services have been decommissioned and are no longer used by any of these experiments. There is also a tendency to move to simpler Computing Elements as their task is limited to deploying pilots on behalf of the experiments.

Some generic pilot factories using virtual machines have been developed for small cloud-like sites, namely VAC and VCycle [18], that are able to deploy pilots on behalf of multiple VOs, including applying a fair share.

5.2 Data management

The Grid was also supposed to offer a seamless access to data, wherever they are replicated. In addition, as was mentioned for job brokering, it is necessary to know where files are replicated in order to broker jobs close to where the input data is stored.

However, in this domain as well, sites have their history and their preferences, and were free to choose whatever storage technology they wanted. Two main contenders were available in the early 2000's: Castor (at the time the first version of it) and dCache.

Concerning replica catalogs, it quickly turned out that the Globus replica catalog could not scale up to the required number of files, as experiments were expecting millions of them. It is thus in a hurry that a new replica catalog was derived from the Castor name server: the LCG File Catalog (LFC) [19]. This was made available very quickly thanks to a heavy reuse of the Castor code, and became quite popular for LHCb and ATLAS. The LFC demonstrated a very good scaling, and although initially experiments were anticipating the need for several distributed services, it turned out that a single service could cope with the load. In fact, the LFC service was distributed on several servers sharing the same database, and one server was devoted to writing while the others were read-only, but there was no need to have multiple synchronised databases.

Smaller sites were indeed struggling with the deployment of storage as both Castor and dCache are meant for rather large systems, and include tape storage, which was not required for Tier2s. Thus, another derivation from Castor was made for handling pure disk storage systems: the Disk Pool Manager (DPM), that was further developed and is quite robust and easy to deploy and configure, even for smaller sites.

In 2006, it was realised that experiments needed to be able to use different service classes for storage: some files should be stored on disk, some others should be migrated to tape, and some may be permanently on disk but also on tape. Due to the diversity of storage systems, implementing experiments' middleware layers was not easy. Therefore, after some discussion at a workshop in Mumbai (preceding CHEP2006), it was proposed to use a storage interface abstraction that was just being defined at the time: the Storage Resource Manager (SRM). Many discussions had to take place between storage developers, SRM proponents and experiments in order to define the minimum set of functionalities that was required and that could be implemented. Despite SRM often being considered rather heavy (since it is relying on services interacting with the actual storage), one can say it was a reasonable success. It allowed to define different types of service classes, and still is the most popular way to handle tape storage (although ALICE uses *xroot*).

What was felt by the experiments as a great improvement was the de-facto standardisation of file access protocol on *xroot*, although some sites also provide direct POSIX access (for example at CNAF). It is then up to the user to select the most appropriate protocol. One advantage of *xroot* is that all sites allow it on the WAN, which means that files can be accessed remotely. This is used in various ways by experiments, either through the implementation of an *xroot* federation (via redirectors) or as a failover backup for accessing files that are locally unavailable.

Despite some initial thrust, the usage of the *https* protocol for file access and transfer doesn't seem to be widely used yet. However, *https* federation is very handy for checking the existence of files on storage and thus checking the consistency of replica catalogues.

Concerning replica catalogues, those are still quite widely used for job brokering since, unless one allows jobs to run anywhere and access data through the WAN (with potential large latencies), brokering jobs close to their input data is still favoured by all experiments.

6 Commonalities

From the great initial hopes that experiments would only have to provide a thin layer of software for accessing Grid resources, while most of the interface would be ensured by common middleware, the current reality is that all experiments have implemented quite sophisticated Distributed Computing systems.

There are several reasons why this happened. Firstly, the way to access and use computing resources very much depends on the computing model, which for good or bad reasons is quite different from one experiment to another. Secondly because experiments needed to have working systems at times when the middleware was not providing the required reliability, they started to develop their own solutions targeting their specific needs.

In many cases, the systems developed by experiments are being used by other HEP experiments and much beyond in other scientific communities like astrophysics, life or earth sciences… This is the case for DIRAC and PanDA for example.

There are still however several common services left. Some of them were part of the initial plans (e.g. VOMS for authorisation; Computing Elements are still there although sites tend to move to simpler implementations like HTCondor or ARC).

A widely used common service is the File Transfer Service (FTS3) [20] that is based on the Grid File Access Library (gfal2) [21]. This middleware was re-engineered after a first rather successful version had been used for many years. The new system is much more versatile and is able to handle many more protocols than the initial version that was bound to SRM.

As described earlier, for file access, if POSIX access is not available, *xroot* is used. This was a development that took place outside the Grid middleware projects but was adopted because it was fulfilling the experiments' requirements.

Another very interesting tool used by all experiments is the CernVM File System (CVMFS) [22] that is de-facto standard used by all experiments for deploying their applications and beyond (like conditions database, small ancillary files etc). As CVMFS is based on the *http* protocol and the deployment of squid services, it is much simpler now than the initial local installation of software that was requiring long and painful deployment campaigns whenever a new release was done.

7 Brief outlook

The landscape for large data computing is evolving very fast, much faster than the concepts and paradigms used by experiments and sites can cope with.

A lot of the concepts for distributed computing were developed having in mind the fact that applications were single-threaded, processing events sequentially. However, since the increase in CPU power of processors is achieved by increasing the number of cores rather than the speed of individual cores, while the cost of memory is increasing, several experiments have redesigned their frameworks for being multi-threaded (or are currently doing it). The goal is to keep performance and efficiency by decreasing the memory footprint per core. This puts big constraints on the applications (as they should remain efficient, i.e. use 100% of the computing power available) and on the sites (for providing efficient multi-processor scheduling).

Another very active domain in the development of applications is to make best use of vectorisation. The counterpart of this is that applications have to be provided for multiple platforms as all sites are not providing necessarily the same type of processors.

Experiments are also considering very seriously the usage of accelerators and coprocessors (most popular being GPGPUs). It is not yet clear whether this is neither easy nor efficient, although demonstrators for specific parts of the applications show drastic performance improvements. Although these are certainly extremely useful when running applications in a well-controlled environment (for example in the high-level trigger farms of the experiments that they control entirely), it is less clear whether they will ever be of any help for large scale distributed computing, due to the diversity of these devices and the fact that applications would have to be validated for each and every type of them.

With the evolution of hardware and operating systems, it also becomes often quite difficult to execute few-years-old applications since they are no longer compatible. Using virtual machines was considered as a possibility to alleviate that problem and run legacy applications many years later. Lately, even more appealing alternatives have appeared, in particular the use of containers that are light-weight and easy to deploy. Most experiments are working on integrating the use of containers in their distributed computing systems. They offer many other advantages like containment of jobs. Many sites are now providing and even sometimes requesting to use containers for running applications.

In the past few years, experiments have been diversifying a lot the provision of computing resources. From the initial institutional computer farms that were the main source of resources in the early days, experiments are now using all types of Cloud resources (provided by scientific computing centres as alternatives to batch farms, or by commercial cloud providers). It also turns out that many other fields of science rely on High Performance Computers (HPC), very often for running MPI jobs. Although this type of job is not very popular nor suitable for HEP, all experiments are targeting those computing resources, as basically they can also be used as batch farms. Additional complexity comes from the fact that quite frequently those HPC systems are much more complex to access (no direct internet connectivity for example), but they potentially can provide very large computing capacity, which will be very welcome in the coming years.

Commercial cloud providers can also offer very interesting possibilities for extending temporarily the available computing resources in order to absorb short-term urgent requirements. Storage other than temporary on clouds does not seem however to be favoured as requiring too much operational load and also being rather expensive.

As experiments are now capable of using in a seamless manner any type of computing resource by deploying their pilots, any available computing resource should be easily usable. Volunteer computing (for example using BOINC) is also a very good example of how to extend the available computing resources, despite the complex security issues associated with it.

8 Conclusions

When one looks back to the past 18- or 20-years' evolution of the concepts used for the LHC computing, one can notice that there were really many changes. Those changes are probably a good example of Darwinian evolution: many ideas and paradigms were tried out, and in the end only survived those that were the most efficient (and very often as well the simplest). A lot of software/middleware components were developed that were in the end deprecated, although these products probably were an important part of the evolution (even if to demonstrate that the underlying ideas were not the most appropriate). Many "species" appeared in the past few years, and many others are still to come.

It is of course extremely hard to predict what the landscape of HEP computing will be in the next decade. Who would have predicted 15 years ago that Moore's law would be achieved by increasing the number of cores, that new paradigms like containers will exist, that HPC systems would be available at the current scale, that cloud providers would be able to offer enormous computing resources (although at some cost)? What one can consider as granted is that the landscape will be different from what we know today and even from what we may extrapolate. HEP is no longer defining the standards. In this context it is absolutely vital that experiments, sites and funding agencies remain agile enough to be able to adapt quickly to the changes. Their agility in the past 15 years has not been good enough to capture all the changes that took place, since in many aspects one still processes data today as one was doing 20 or 25 years ago!

References

1. S. Bethke (Chair), M. Calvetti, H.F. Hoffmann, D. Jacobs, M. Kasemann, D. Linglin, *Report of the Steering Group of the LHC Computing Review*, http://lhc-computing-review-public.web.cern.ch/lhc-computing-review-public/Public/Report_final.PDF
2. J.Shiers, *Object Database Management Group*, LCB meeting in Barcelona 1998, https://slideplayer.com/slide/8007452/
3. ROOT, https://root.cern.ch
4. M.Adelholz et al., *Models of Network Analysis at Regional Centres for LHC experiments,* https://monarc.web.cern.ch/MONARC/docs/phase2report/Phase2Report.pdf
5. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann, 1998)
6. Ian Foster and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit,* Intl J. of Supercomputer Applications, 11(2):1997, pp. 115-128.
7. *The DataGrid Project*, http://cern.ch/eu-datagrid/
8. *EGEE, Enabling Grid for E-sciencE,* http://cern.ch/eu-egee-org/
9. *EMI – European Middleware Initiative,* http://www.eu-emi.eu/
10. *WLCG web site*, http://wlcg.web.cern.ch
11. *OSG – The Open Science Grid,* https://opensciencegrid.org
12. F. Carminati et al., *Common Use Cases for a HEP Common Application Layer – HEPCAL*, Technical Report, LHC Computing Grid Project, https://uscms.org/s&c/lcg/ARDA/docs/RTAGs/RTAG4-finalreport.pdf (2002)
13. J. Templon, *EDG Response to HEPCAL Use Cases*, https://indico.cern.ch/event/409235/contributions/987300/attachments/823459/1133052/LCG-HEPCAL-sept09.pdf (2003)
14. Ph. Charpentier*, ARDA report to the SC2*, https://indico.cern.ch/event/415165/contributions/1873786/attachments/848086/1181534/2003-09-ARDA-finalreport.pdf (2003)
15. P. Buncic et al, *The AliEn system, status and perspectives*, Proceedings of CHEP'03 San Diego, http://inspirehep.net/record/621164 (2003)

16. A. Tsaregorodtsev et al., *DIRAC: Distributed Infrastructure with Remote Agent Control,* Proceedings of CHEP'03 San Diego, http://inspirehep.net/record/621162 (2003)
17. K. De, *PanDA : Production and Distributed Analysis system for ATLAS*, CHEP'06 Mumbai, https://indico.cern.ch/event/408139/contributions/979616/attachments/815461/1117369/panda-chep06.pdf (2006)
18. A.McNab et al., *Managing Virtual Machines with VAC and VCycle,* J.Phys.Conf.Ser. **664** (2015) 022031
19. J-P. Baud and S. Lemaitre, *The LCG File Catalog,* https://indico.in2p3.fr/event/15112/attachments/41865/51883/Presentation_LFC_visio_conf_France_10_05_2005.pdf (2005)
20. M. Salichos, *FTS3,* WLCG GDB presentation, https://indico.cern.ch/event/251190/contributions/560046/attachments/436655/605917/gdb_fts3.pdf (2013)
21. *Gfal2, Grid File Access Library,* https://dmc.web.cern.ch/projects/gfal-2/home
22. *The CernVM File System*, https://cernvm.cern.ch/portal/filesystem