

Ten years of European Grids: What have we learnt?

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2011 J. Phys.: Conf. Ser. 331 062003

(<http://iopscience.iop.org/1742-6596/331/6/062003>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 95.172.225.198

The article was downloaded on 23/01/2012 at 08:23

Please note that [terms and conditions apply](#).

Ten years of European Grids: What have we learnt?

Stephen Burke¹

¹ Rutherford Appleton Laboratory, Didcot, OX11 0QX, UK

E-mail: stephen.burke@stfc.ac.uk

Abstract. The European DataGrid project started in 2001, and was followed by the three phases of EGEE and the recent transition to EGI. This paper discusses the history of both middleware development and Grid operations in these projects, and in particular the impact on the development of the LHC Computing Grid. It considers to what extent the initial ambitions have been realised, which aspects have been successful and what lessons can be derived from the things which were less so, both in technical and sociological terms. In particular it considers the middleware technologies used for data management, workload management, information systems and security, and the difficulties of operating a highly distributed worldwide production infrastructure, drawing on practical experience with many aspects of the various Grid projects over the last decade.

1. Introduction

Ten years (less a few months) is a long time by computing standards, so this seems like a good time to look back, especially with the startup of the Large Hadron Collider (LHC) and the transition to the new EGI Grid project. However, it is not possible to cover ten years of history in any detail in a short paper, so this is just a selection of comments and observations about particular issues of interest to the author. In particular this paper only discusses the EU-funded projects, i.e. the European DataGrid (EDG) [1], Enabling Grids for E-science (EGEE) [2] and the European Grid Infrastructure (EGI) [3], and their relation to the LHC Computing Grid (LCG) project [4]; it does not discuss the Open Science Grid (OSG) [5] or Nordic DataGrid Facility (NDGF) [6] which also provide resources to the Worldwide LCG collaboration. It should be emphasised that the opinions expressed in this paper are purely those of the author, and do not represent the views of any project or organisation.

This paper first reviews the history of the various Grid projects, and then discusses some lessons learnt from a selection of areas in middleware development and Grid deployment and operations. It concludes with an attempt to summarise the experience of the last decade, and draw some general conclusions for the future.

2. The history of the European Grid projects

EDG ran for a little over three years from January 2001. It had work packages to develop middleware in several specific areas, and also to collect requirements and assess the usefulness of the middleware from the perspective of three user communities including High Energy Physics.

The LCG project started in January 2002; the first Grid deployment was in March 2003, based on the EDG testbed.

EGEE ran from April 2004 to April 2010. The main objectives were to operate a large-scale Grid as a production service, and to develop Grid middleware. This was formally constituted as three separate two-year projects, and this was not purely an administrative division, there were some formal

changes in the project structure which impacted middleware development and deployment. EGEE shared its Grid infrastructure with LCG, and the Grid operations activity within EGEE had a substantial overlap with LCG operations.

EGEE was succeeded by EGI in May 2010. This largely provides a continuation of the EGEE deployment and operations activities, although with a more regionalised structure. However, the EGEE-developed gLite middleware has been transferred to the European Middleware Initiative (EMI) [7] which also incorporates several other middleware distributions.

2.1. EDG ambitions compared with EGEE outcomes

The formation of EDG marked the start of the development of large-scale European Grids, and it seems useful to consider the things the project hoped to achieve and compare them with the actual outcome. To briefly summarise the EDG middleware work packages, the products they produced and the corresponding products in the current production Grid:

- The Workload Management work package produced a job submission service and a connected logging and bookkeeping service. The current Grid has equivalent services which are direct descendants of the EDG products.
- The Data Management work package produced a data movement service, a file cataloguing system and a set of client tools. The current Grid has similar products, but these were written from scratch and are only indirectly influenced by the EDG products.
- The Information and Monitoring work package produced two separate information systems. Direct evolutions of both of these are still in use, although one is in the process of being phased out.
- The Fabric Management work package produced two independent tools to help with configuration and deployment. One of these is currently in use at some large sites, while it is generally considered to be too complex for smaller sites who usually use a simpler tool developed by EGEE.
- The Mass Storage work package produced a prototype storage system. This did not directly survive, but experience with it influenced the development of the Storage Resource Manager (SRM) protocol [8] in use today.
- EDG did not operate a production Grid service, but it did have a work package dedicated to providing a Grid testbed, intended to be an environment where the functionality and performance of the middleware developed within the projects could be evaluated. However, in practice this evolved into something like a production service, in that it ran continuously and provided access to fairly significant resources.
- Security-related middleware was only added to the EDG proposal at the last minute. Developed products included a Virtual Organisation (VO) membership service and various tools to manage authorisation decisions within Grid services. Direct descendants of these products are still in use, soon to be supplemented with a new authorisation service.
- Finally, EDG had a Networking work package which developed a set of network monitoring servers and associated tools. These have no direct equivalent today, perhaps reflecting the fact that networks have been much less problematic than we expected a decade ago.

The middleware development work packages were intended to develop a discrete set of products in particular areas which were not otherwise well-served by the middleware available at the time; there was no particular intention to produce a complete distribution which could serve all the needs of a production service. However, EDG did in fact produce a distribution which could be deployed to produce an operational Grid, albeit with somewhat limited functionality. This was a significant achievement, but had the consequence that the subsequent EGEE project continued developing a complete middleware distribution in-house, which limited its ability to benefit from developments elsewhere. By contrast the EGI project has little in-house middleware development, but expects to select its middleware from external projects according to its needs and the capabilities offered.

3. Lessons learnt

This section examines various aspects of Grid middleware, deployment and operations, and considers some of the more significant lessons that may be learnt.

3.1. Interfaces to computing systems

Grids need an interface to a computing system to allow jobs to be submitted. EDG simply took the gatekeeper service provided by the Globus [9] toolkit and this is still in use, although it has been heavily re-engineered by LCG over the years to be more robust.

The CREAM service, developed in EGEE, is now also robust and has more functionality, and is likely to take over as the primary interface in 2011. This has been in development for around five years, which is not untypical for major new services, and contrasts with typical project funding cycles of 2-3 years. As a result there have sometimes been unrealistic timescales for deployment, and also a lack of certainty that funding would last long enough to see a development through. The lesson is to be realistic from the start about how long it takes to develop a new service to production quality.

3.2. Workload management

As mentioned above, the Workload Management Service (WMS) has been in continuous development for a decade, and the performance is now very good. One feature of the WMS is that the way in which jobs are distributed to sites is almost entirely under user control; while this has obvious advantages it can also cause difficulties because many users fail to understand how it works. Indeed, the details of how jobs get distributed over the Grid can be difficult even for experts to understand or predict because there is a strong coupling with the details of the information system, and there has been surprisingly little research into optimisation of job distribution algorithms.

Also, in practice the LHC experiments have preferred to develop their own workload management systems, generally based on the concept of “pilot jobs”, which pull the real jobs from a central queue. The biggest advantage of this approach is that in practice compute nodes are often broken in some way, and a pilot job can validate its environment before starting the real job. Such a system also allows late binding, i.e. jobs are not queued at sites and can therefore be re-ordered up to the time they start execution. There is also the perceived advantage that such a system puts control over scheduling in the hands of the VO rather than the site or Grid.

3.3. Mass storage

It can be argued that the SRM protocol is the biggest success story in Grid middleware, in that it has achieved all the stated goals: an internationally-standardised web service protocol with many compliant and interoperating implementations and which in practice has been very widely adopted.

However, there are also some disadvantages to SRM. It is a rather complex protocol, perhaps more so than is required, but conversely it fails to reflect many important features of real-world storage systems. Also, SRM is typically layered on top of other storage technologies, and it often fails to mesh well with them due to mismatches in the underlying concepts.

Another question is whether the objective of standardising the server protocol, which has been the general paradigm for all Grid projects for many years, is in fact the most useful approach. In practice users nearly always use some client-side library or command-line tool rather than interacting with the web service directly, so it may in practice be more useful to standardise that API instead.

A final point is that SRM does not cover some aspects of storage systems, including such important areas as file access protocols and quotas, and this remains an active area of development.

3.4. Data management

In contrast to workload management, the data management middleware has been re-designed and re-written from scratch several times. It is often tempting to think that it will be possible to do a better job if starting again in the light of experience, but there are also major downsides with this approach. Existing middleware will have evolved functionality and incorporated bug fixes which embed a large

amount of tacit knowledge which can easily be lost in a re-write. One result of this is that the current generation of data management tools has substantially less functionality than we had in EDG, albeit that what we have has much better performance.

The focus on re-writing the lowest-level middleware has also inhibited the production of higher-level tools; EGEE originally had plans for such services, but they did not in practice materialise. This led the LHC experiments to develop their own data management systems, which are now mature and embed a lot of experience. However, these are strongly coupled to the experiments' own software systems, and it is not clear if any of this experience will be fed back into the generic middleware.

3.5. *Information systems*

EDG produced a service known as the BDII (Berkeley Database Information Index) as a short-term hack while an alternative was developed to the point where it could take over. This had a more elegant design and much more functionality than the BDII, but despite several years of work its performance did not improve sufficiently. The BDII performance was also not adequate as the Grid expanded, but it proved possible to make targeted improvements which allowed it to cope each time a bottleneck was encountered. The lesson here seems to be that focussing work purely on finding quick solutions to problems which are vital to practical use may be more productive than trying to find an elegant design which may eventually provide a general solution, but which works less well in the interim.

The quality of the published information is vital; if even a few sites or services provide bad information it can disrupt the operation of the whole Grid, or render summary information, e.g. the total amount of data stored in the Grid, meaningless. Responsibility for the information has been highly diffuse since it comes from many different services and relies on the configuration at many sites. Over the years the information providers and configuration tools have been improved substantially and the information is now fairly reliable, although by no means perfect. However, this improvement has often been driven by particular individuals, sometimes outside their official job descriptions, rather than by any concerted effort by the Grid projects. One major improvement is the introduction of the GStat monitoring tool [10], which provides an overview of the published information and also validates it as far as possible and raises alarms if it detects inconsistencies.

3.6. *Security services*

Security is never popular with either users or (most) middleware developers since its purpose is to prevent people from doing things unless properly authenticated and authorised. It is however essential – so far there have been no attacks directed against the Grid as such, but it would be unwise to assume that this will continue indefinitely. During the life of EGEE there was a concerted effort to document and solve any known vulnerabilities, and this process is continuing in EGI.

The VO Membership Service (VOMS) has been a success, and is now widely used. However, its integration with individual Grid services has been less successful; in the absence of any general specification developers have each made their own implementation choices, and the results have often been inconsistent. For example, services may use the VOMS information to *select* service functionality rather than just authorising it, which means that different services may conflict in the requirements they place on VOMS credentials.

3.7. *Architecture*

Both EDG and the first phase of EGEE attempted to design an overall architecture to define how the various Grid services should interact, but in practice this had little impact and was largely abandoned. A large factor in this was that middleware development is product-based, with rather little interaction between the product teams, which made it difficult to make progress on issues which fall across multiple products or outside them.

One of the principal architectural rules in EDG was intended to be that there should be no single points of failure, with all services being distributed. In practice this has proved difficult and we now have many single service instances whose failure can stop much of the activity of one or more VOs.

However, this has generally not proven to be a major problem if the services are well-managed, and in fact the recent trend in LCG has been to centralise more services at CERN.

In addition to the general architecture we have also been lacking in strategies for the deployment of services within the production Grid, i.e. questions like the number of service instances to have, where they should be located and what the failover strategies should be. In practice this has usually been determined by ad-hoc decisions which may or may not be optimal. This is partly the result of the separation of middleware development from Grid deployment – something which is increasing with the transition to EGI and EMI.

3.8. Middleware development

The middleware (known as gLite) relies on a large number of open source software tools which often have poor version control, i.e. even minor changes in version can lead to unexpected differences in behaviour. The result is a “dependency hell” wherein the middleware is locked to specific, and sometimes incompatible, versions of those tools. This has made it very hard to port the middleware to different OS versions – even the relatively minor evolution from version 4 to version 5 of the Scientific Linux distribution has taken a few years (and is still not complete); the often-stated goal of porting to other Linux distributions has essentially never been achieved.

The other major lesson has been that the time needed for testing and debugging is always radically underestimated. One consequence is that early attempts to set deadlines for software releases were usually ineffective. EGEE moved to an incremental model where each piece of new code moved at its own pace and was released as it became ready. However, EMI has moved back to something more like a scheduled release model, and it remains to be seen how well this works.

3.9. Error handling

Dealing with errors in Grid services requires a major change in attitude by software developers. A single computer works most of the time, and if it fails it usually fails completely. However, a Grid with hundreds of sites and tens of thousands of individual machines will always have many things broken – but conversely it always has many more things working. This means that exceptions are not exceptional; they arise constantly in normal operation. To be effective, middleware therefore needs intelligent failover strategies which can retry operations if possible, but not to the extent that the retries themselves create more problems. It also implies that error messages and logging need to be comprehensive and clear, to allow users and system managers to understand the cause of any failure.

Unfortunately, the middleware development process has tended to militate against this. Project milestones and deliverables relate to functionality, so there is relatively little incentive to improve error handling at the expense of other work. Also, developers usually work with small, well-controlled test systems, so they are often unaware of many of the problems that exist in the real production environment – this contrasts with the developers in the LHC experiments, who generally work directly on the production systems and interact directly with users. A final point is that the middleware has many layers, and if the lower layers have poor error handling there is little the higher layers can do. Error handling has improved somewhat over the years, but usually in response to specific bugs or requests rather than as part of a general strategy.

3.10. Operating a production Grid

Given the comments above, monitoring the state of the Grid and fixing problems as they arise are vital. EGEE and LCG jointly developed a large set of tools and processes which have led to a huge improvement in reliability at the cost of a large amount of manual intervention. One caveat is that EGI is devolving this to a regionalised structure, and it will be vital to do this without losing coherence.

A constant operational problem has been the question of how to test and deploy new middleware; there is a tension between fast upgrades which risk damage if the new software is buggy, and slow upgrades which lead to a wide variety of deployed versions across the Grid. EGEE originally had a pre-production Grid which was intended to test new releases, but in practice neither sites nor users had

enough spare resources to make this work. The consequence is that new middleware is now deployed directly into production, which increases the risk of unreliability. In any case, the practical experience is that some sites can take a very long time to upgrade, and the overall process is such that even trivial bug fixes take many months to be deployed. This contrasts with the services run by the LHC VOs, where bugs can be fixed and new versions deployed within a few days.

4. Conclusions

Following the detailed discussions above, there are some general observations:

- Adjusting an existing product to deal with specific, critical problems as quickly as possible seems to be more effective than designing an elegant new service from scratch.
- The computing aspect of Grids is much easier than data management – even after a decade the latter is still an active area of research and development, although the current middleware has proved sufficient to deal with the initial dataflow from the LHC.
- Networking seems to be even easier – a decade ago we worried that we would have severe network bottlenecks and dedicated middleware solutions would be required. In reality this has not been the case.
- Documentation is always difficult; almost no-one enjoys writing it, and the pace of change makes it rapidly outdated. Also most users tend not to read documentation and prefer more direct support. Finally, most documentation has migrated from PDF documents to web pages and then to wiki pages, and while this may make life easier for the author it usually results in a tangled network of pages which is almost impossible to read in any coherent way.
- Most people in the Grid projects are experts in a specific area without much view of the wider picture. However, it is useful to have at least a few non-specialists who can see the connections between the different areas.
- The LHC experiments have in the end usually got what they wanted, and the Grid has worked very well during the first year of LHC operation. However, this has often been a result of the experiments implementing their own middleware with their own manpower, which is rather far from the vision of a general-purpose universal Grid which we had at the start of EDG.

The LHC is expected to be running during most of the next decade, and probably the one after that. We now have a large-scale Grid that in general is working well, so the priority is to evolve what we have without breaking anything. Experience suggests that developing a completely new service to production quality takes several years, but this is short compared with the expected life of the LHC and may well be worthwhile.

Ten years is still a long time in computing. The general technological changes in the last decade have if anything been less than we might have expected, but the landscape may be quite different in 2021. The current hot topics include clouds, multicore CPUs and virtualisation, but only time will tell if they still look like the big stories a decade from now. The only certainty is change, and LCG will have to change with it – but after a decade of development it starts from a solid foundation.

References

- [1] EDG (European DataGrid) <http://eu-datagrid.web.cern.ch/eu-datagrid/>
- [2] EGEE (Enabling Grids for E-science) <http://www.eu-egee.org/>
- [3] EGI (European Grid Infrastructure) <http://www.egi.eu/>
- [4] LCG (LHC Computing Grid) <http://www.cern.ch/lcg>
- [5] OSG (Open Science Grid) <http://www.opensciencegrid.org/>
- [6] NDGF (Nordic DataGrid Facility) <http://www.ndgf.org/>
- [7] EMI (European Middleware Initiative) <http://www.eu-emi.eu/>
- [8] Storage Resource Management Working Group <http://sdm.lbl.gov/srm-wg/>
- [9] The Globus Alliance <http://www.globus.org/>
- [10] GStat <http://gstat-prod.cern.ch/gstat/summary/>