A Grid of Grids using Condor-G

R. Walker, M. Vetterli, Simon Fraser University, Burnaby, Canada
A. Agarwal, R. Sobie, D. Vanderster, University of Victoria, Canada
M. Grønager, University of Copenhagen, Denmark

Abstract

The Condor-G meta-scheduling system has been used to create a single Grid of GT2 resources from LCG and GridX1, and ARC resources from NorduGrid. Condor-G provides the submission interfaces to GT2 and ARC gatekeepers, enabling transparent submission via the scheduler. Resource status from the native information systems is converted to the Condor ClassAd format and used for matchmaking to job Requirements and Rank by the Condor Negotiator. The use of custom external functions by the Negotiator during matchmaking, provides versatility to develop job placement strategies. For example, a function exist to use a matrix of CE-to-SE bandwidths, together with data location information, to make 'network closeness' available to both Requirements and Rank expressions. Other examples where such flexibility can be applied are in implementing a feedback loop to dynamically prefer successful or fast resources, and block matching to blackholes. The Condor-G Grid of LCG resources has produced 180,000 jobs during recent ATLAS productions, which matches the number produced by the LCG Workload Management System in the same period. GridX1 resources were used for ATLAS production in this way starting Autumn 2005. Simple jobs have been matched and ran on the full Grid federation, including NorduGrid resources, and work is underway to make use of advanced ARC features allowing Condor-G submission of ATLAS production on all resource flavours.

INTRODUCTION

We describe the use of Condor-G as a complete Workload Management System(WMS) for grid resources. This includes the matchmaking of jobs to resources based on requirements and preferences for static or dynamic resource attributes, and also allows the use of external data such as a data location catalogue. A scalable architecture of multiple Schedulers is discussed in comparison to the LCG WMS which scales with multiple Resource Brokers(RB). The system has been deployed on the Canadian HEP Grid, GridX1, and parasitically on LCG to include over 100 sites.

CONDOR AND CONDOR-G

Condor-G is an extension of the popular batch system to the grid world. In the Condor batch system(BS) each worker node(WN) publishes its capabilities and status to a *Collector* in the form of a ClassAd, a list of attribute-value pairs. A user job is also represented by a ClassAd, and is also sent to the Collector. It contains the *Requirements* and *Rank* expressions which specify requirements and preferences of the job in terms of the WN capabilities, e.g. OS or minimum RAM. A *Negotiator* periodically processes the job and resource ClassAds to match jobs to the most appropriate WN.

Zooming out from the batch system to consider distributed processing clusters. The batch nodes are replaced by remote gatekeepers to these clusters, and the WN ClassAd is replaced by a description of the *cluster* capabilities and status, e.g. the number of running and waiting jobs, or the total number of cpus. The architecture and the Condor software components are unchanged, but now the job dispatch is from a local Scheduler to a remote gatekeeper. For this purpose, Condor-G provides a client implementation of the GRAM protocol.

The architecture including the three Condor services is shown in Fig.1. Scalability is achieved by having multiple Schedulers, in contrast to LCG WMS where there are multiple Resource Brokers(RB). It should be noted that if RBs do not communicate, and they don't, then having multiple instances prevents any intelligent resource brokerage whatsoever. The most attractive resource is attractive to all N RB's and is thus N times over-subscribed. Scaling at the Scheduler level, duplicates the busiest component of the WMS without compromising the resource brokerage. Of course this does not scale indefinitely as a single Negotiator has its limits but 2-3 Negotiators is preferable to 20-30 RB's (by some estimates based on current performance).

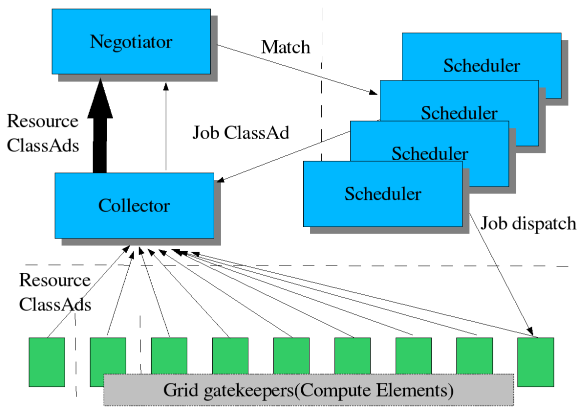

Figure 1: The service architecture of Condor-G with matchmaking, showing the scalability via multiple Schedulers

Matchmaking

The matching of jobs to gatekeeper resources is performed by the *Negotiator* in the same way that jobs are matched to WNs in the Condor BS. First, the resource and job ClassAds are used to evaluate requirements expressions for both the resource and the job. The job typically contains constraints on the resource capabilities such as OS or RAM. Also the resource ClassAd can contain constraints, e.g.no job starts between 08:00-17:00. The job Rank expression is evaluated only for job-resource pairs which pass the requirements. The expressions are mostly formed by simple logic of the attributes in the job and resource ClassAds. However there is the possibility to include arbitrary functions which will be evaluated by the Negotiator in custom shared object libraries. These enable external information, not expressible in the ClassAds, to be folded into the matchmaking process. The classic example is in using the location of any required input data in deciding where to send the job - so called data co-location. Such a function can be incorporated into both the Rank and Requirements expressions, to add constraints and preferences, and is most powerful when used in conjunction with a table of inter-site bandwidths. This is used for ATLAS production whereby a minimum bandwidth to the input data is required and sites with higher bandwidth to the input data are strongly preferred. The LCG WMS only allows an absolute requirement that the input data is on *the* close SE, which is very inflexible, and thus largely unused.

Input and Output sandboxes

The input sandbox typically consists of steering scripts and configuration files, and as such should be only a few MB. Larger files such as binaries or input data have different routes to the Worker Node. Condor-G and GRAM arrange for the user executable to get to the WN, and for the standard output and error to get back to the submission host. We take advantage of this by embedding the input and output sandboxes inside the executable and stdout respectively.

The user executable (usually a script) and any other input files are wrapped into a self-extracting tarball and this is sent as the executable. On arriving at the WN and being run, the tarball unwinds and the actual user executable is run.

Similarly, the actual stdout from the job and any (small) output files are wrapped into a tarball which is passed back as the stdout. A simple script unpacks this when the user wishes to see the actual output files. The stderr is untouched and serves to report any problems in the input or output sandboxing. Although Condor-G does offer mechanisms for moving arbitrary files to and from WNs, this was found to be the only method which worked consistently regardless of which jobmanager was met on the gatekeeper. In particular, the custom LCG jobmanagers that enable non-shared home areas on the WNs, were found to disrupt such Condor-G mechanisms.

DEPLOYMENT AND OPERATION

GridX1

GridX1 is the Canadian HEP grid consisting of 4 clusters and over 2000 cpus. The clusters are shared with other disciplines and have no manpower to install LCG middleware. However they all have gatekeepers and could be integrated into a Condor-G grid by periodically publishing a ClassAd. The WMS proved to be highly performant and stable during the ATLAS DC2 production[1], where 20,000 LCG jobs were ran on GridX1 by means of an interface.

LCG

The LCG Compute Element(CE) is deployed on over 100 sites. In order to use the Condor-G WMS, each CE would need to produce one ClassAd per queue, to describe the resource capabilities and status. This would involve considerable effort and cooperation from the LCG deployment team. Alternatively, there already exists an information system in the form of the central BDII which contains all the required information. A perl script was created to query this BDII and convert the information into one ClassAd per CE queue. This required no deployment to LCG CEs. The extra component is shown in Fig.2. In this way,

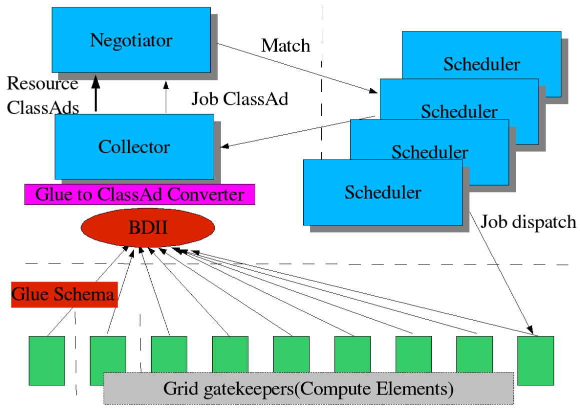

Figure 2: The Condor-G architecture for LCG showing the extra component to convert the BDII information into ClassAds.

any manipulation of the information by either the site managers, such as setting the queue status to 'draining', or by VO managers, perhaps removing sites which fail critical tests, was automatically passed on to the Condor-G WMS.

The large scale operation, as an alternative to the LCG WMS, during DC2 showed a dramatic improvement in the production rate. This was due to the faster submission rate, which was able to fill more of the available cpus. A calculation based on the number of cpus and typical length of job reveals that to keep the resources full a job must be submitted every 6 seconds. This includes the time to check its status several times, and get the output when finished. It was found that the LCG WMS submission took 15-60s, which, together with non-negligible times for status requests and getting the output, meant several submission instances needed to run in parallel. This increased the required manpower. The Condor-G instance submitted a job in 0.1s and had negligible status and get-output times. This is because the Scheduler is local and dedicated. A single instance was able to fill the available resources and double the previous production rate. The actual dispatch time, from the Scheduler to the remote gatekeeper, is comparable for a single Scheduler and a single RB - it is in fact the same Condor-G component in both cases. The observed Condor-G gain is mainly due to the synchronous submit stage returning quickly. In other words there would be a similar throughput for systems consisting of an equal number of Schedulers and RBs. The strength of the Condor-G approach is that the scaling is at the Scheduler level, thus allowing meaningful resource brokerage.

FEDERATING GRIDS

The GridX1 resources have GT2 gatekeepers but are not associated with LCG, and are not in the LCG information system. Nevertheless, they can be included in the Condor-G WMS, and appear just like any LCG CE. This is a form of federation of Grids, and these resources are currently used in precisely this way for ATLAS production.

Condor-G also provides clients to several other gatekeeper types, notably the ARC compute element of NorduGrid. Some 10% of ATLAS resources are accessible only via ARC CEs, so a system capable of submitting to both GT2 and ARC is of great interest. As part of the LCG interoperability project, a BDII was populated with information from the ARC CEs. This can easily be converted to ClassAds, and tests have been performed to do matchmaking and submission across GridX1, LCG and NorduGrid. This will be extended to submit real ATLAS jobs in the future.

CONCLUSIONS

A fully functional WMS has been built from off-the-shelf components provided by the Condor group. After successful tests on GridX1, the system was deployed on LCG by converting information from the native information system. This proved to be highly performant during the ATLAS DC2 production, and continues to be the mainstay of ATLAS production on LCG. Recent developments to include data location metrics in both the Requirements and the Rank have been particularly important in reducing data handling failures due to distant or overloaded storage elements.

The rapid submission and status requests are a result of having a local Scheduler and is a response taken for granted by users of a local batch system. We believe that this is a highly desirable feature to the user, and a strength of the Condor-G WMS.

The upcoming GLite compute element is based on Condor-C - the movement of jobs between Condor Schedulers. Therefore it is anticipated that the Glite CE will be compatible with the existing Condor-G framework, with minimal changes. This will be tested on the pre-production testbed in order to maintain the Condor-G WMS on LCG after GLite deployment.

ACKNOWLEDGMENTS

The authors wish to thank the Condor team for the stable software and excellent support. The original concept and development was driven by the SAMGrid team at FNAL. We thank LCG for their deployment expertise and also for not preventing this unplanned access to resources.

REFERENCES

[1] A. Agarwal et al., GridX1: A Canadian Particle Physics Grid, CHEP 2006, Mumbai, Feb 2006.