Integrating Xgrid into the HENP distributed computing model

L Hajdu[1], A Kocoloski[2], J Lauret[1], M Miller[2]

[1] Brookhaven National Laboratory, Upton, NY 11973, USA

[2] Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

E-mail: `kocolosk@mit.edu`

Abstract. Modern Macintosh computers feature Xgrid, a distributed computing architecture built directly into Apple's OS X operating system. While the approach is radically different from those generally expected by the Unix based Grid infrastructures (Open Science Grid, TeraGrid, EGEE), opportunistic computing on Xgrid is nonetheless a tempting and novel way to assemble a computing cluster with a minimum of additional configuration. In fact, it requires only the default operating system and authentication to a central controller from each node. OS X also implements arbitrarily extensible metadata, allowing an instantly updated file catalog to be stored as part of the filesystem itself. The low barrier to entry allows an Xgrid cluster to grow quickly and organically. This paper and presentation will detail the steps that can be taken to make such a cluster a viable resource for HENP research computing. We will further show how to provide to users a unified job submission framework by integrating Xgrid through the STAR Unified Meta-Scheduler (SUMS), making tasks and jobs submission effortlessly at reach for those users already using the tool for traditional Grid or local cluster job submission. We will discuss additional steps that can be taken to make an Xgrid cluster a full partner in grid computing initiatives, focusing on Open Science Grid integration. MIT's Xgrid system currently supports the work of multiple research groups in the Laboratory for Nuclear Science, and has become an important tool for generating simulations and conducting data analyses at the Massachusetts Institute of Technology.

1. Xgrid Overview

Xgrid[1] is a distributed computing platform for Macintosh computers developed by the Advanced Computation Group at Apple. Xgrid is designed to simplify cluster configuration and maintenance, and its availability in every copy of Mac OS X 10.4 is a significant aid in that endeavor. Adding a Macintosh worker node to an Xgrid cluster is as simple as specifying the IP address of the controller and flipping a switch. Job submission is accomplished using a command-line client or a native Cocoa[2] API and features a job description language (JDL) with a variety of configurable options. Full details of the Xgrid JDL can be found in the manual pages for the command-line client[3].

Figure 1 illustrates the workflow of a typical Xgrid job submission. A detachable client authenticates to the controller and submits a job description consisting of one or more tasks. Agents advertise their presence to the controller and indicate whether they are currently willing to run tasks. Agent administrators have the option to always accept Xgrid tasks or to only accept them after a period of inactivity. The controller disburses tasks to the available agents in a FIFO manner, using a simple algorithm to select the fastest available agent at the time. The

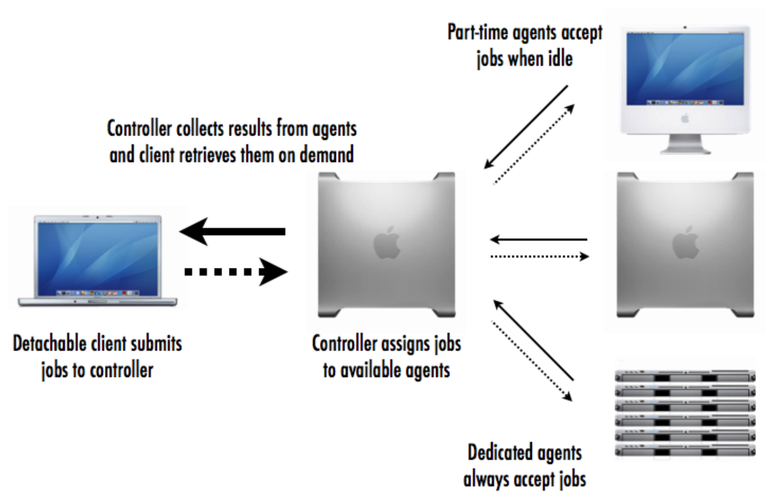
Part-time agents accept jobs when idle
Controller collects results from agents and client retrieves them on demand
Detachable client submits jobs to controller
Controller assigns jobs to available agents
Dedicated agents always accept jobs

Figure 1. Schematic of Xgrid job submission workflow

agents execute the tasks and return all results to the controller. The results are then stored by the controller until the client reconnects and asks to retrieve them.

All communications between elements of an Xgrid cluster are conducted using a lightweight XML protocol built on top of the Blocks Extensible Exchange Protocol (BEEP)[4]. BEEP allows for multiple channels of communication across a single socket. As a result, no ports need to be opened on client or agent machines, while the controller listens on the IETF-specified port 4111. Xgrid offers the ability to stage an arbitrary set of input files in a job's working directory on an agent, and it automatically transfers any output files created there to the controller upon completion. Data transfers handled by Xgrid in this manner are as secure as all other Xgrid communications, and no special steps are required to configure the transfer. On the other hand, the protocol is not ideal for very high-volume data transfers since a) encoding binary data in Base64 implies at least a 33 percent overhead in network I/O and b) all data must be routed through the controller, even if the final destination is a completely separate filesystem.

2. An Xgrid cluster for HENP computing

Over the past several years scientists making personal computer hardware purchases at the Massachusetts Institute of Technology's Laboratory for Nuclear Science have increasingly favored the Macintosh platform. An Xgrid cluster was proposed in early 2006 as a low-maintenance way to harvest the spare cycles of these new machines. The lack of centralized administrative control over the worker nodes favored a cluster configuration with a low barrier to entry – no additional requirements were placed on the agents beyond the one-time authentication needed to connect to the cluster. As a result, the cluster grew quickly to include several dozen machines with a minimal time investment on the part of the maintainers.

On the other hand, scientists wishing to use this cluster for High Energy and Nuclear Physics research needed access to a larger set of software libraries than those guaranteed to be available on a base installation of OS X 10.4. To further complicate matters, there is no officially supported Fortran compiler on the OS X platform, and the worker nodes at MIT included a mix of big-endian PowerPC and little-endian x86 architectures. The open-source Fink[5]

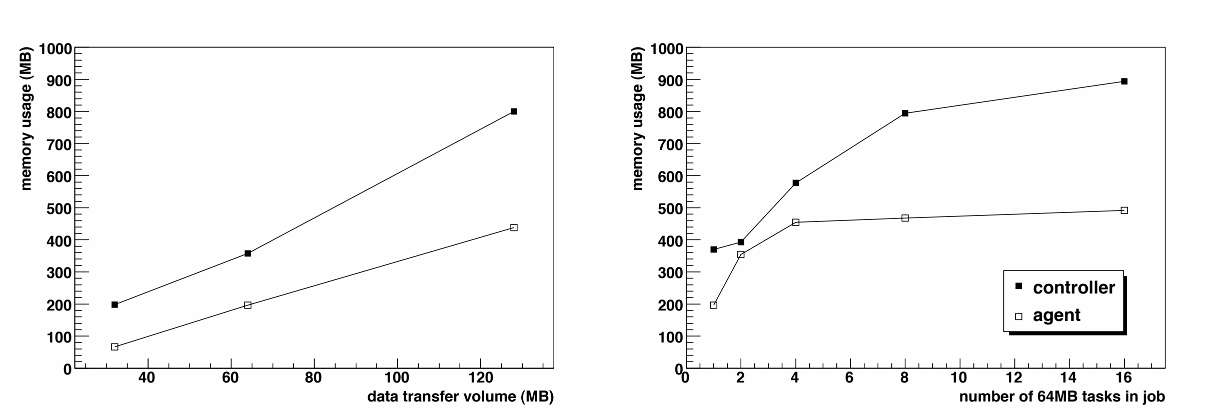

Figure 2. Memory allocation statistics for data transfers using the Xgrid protocol. The left panel plots the peak memory allocation on the controller and an agent for single-task jobs of various sizes. The right panel considers multi-task jobs where each task requires a 64MB transfer, and up to 4 tasks are executed simultaneously on the agent.

package management system allowed us to overcome these obstacles in a flexible and robust manner. Fink simplified the process of porting a wide variety of UNIX software to OS X in both its PowerPC and x86 flavors. Maintaining Fink deployments for both architectures in NFS allows users to link against the same set of libraries that are available at runtime on the Xgrid cluster. A build recipe using Xcode[6] was also made available to automatically generate so-called "Universal Binaries" containing object code for both PowerPC and x86, with the native code being selected automatically by the agent at run-time. The Xgrid cluster at MIT currently uses Fink to support ROOT[7], GEANT4[8], PYTHIA[9], and many other packages that are not HENP-specific.

3. Data, Metadata, and Cataloging

The large datasets generated by the current generation of HENP experiments frequently tax the capabilities of distributed computing systems designed to analyze them. The MIT cluster uses a basic xrootd[10] server to provide reliable read-only access to input ROOT datasets. The XML-based Xgrid transfer protocol is used to stage in macros, executables, and control files on the worker nodes, and to return some output datasets back to the controller. This protocol can be valuable since it requires zero additional configuration and works in situations when alternatives such as a shared filesystem would not be possible. However, Xgrid's methods for translating data to and from Base64 encoding are memory-intensive, and the simultaneous completion of multiple jobs with large output datasets has been the primary source of instabilities in the MIT cluster. Figure 2 plots the peak memory allocation on the controller and an agent for a variety of job types. The left panel indicates a linear relationship between the size of the dataset to be transferred and the memory required on the agent. The agent uses almost 3 MB of memory for every MB that is transferred, while the controller requires approximately double that amount. Splitting a job up into smaller tasks as shown in the right panel offers some improvements, but the controller's memory usage continues to grow. Furthermore, the controller process in the current release of Xgrid will crash and restart if it encounters a job requiring a 1GB memory allocation, effectively limiting the volume of data that can be transferred using this protocol to less than 150MB per task.

Given the limitations of the current incarnation of Xgrid's file transfer mechanism, finding alternative methods for getting output data off the worker nodes in a flexible, efficient and secure manner is an important task. Shared access to centralized storage is often used for this purpose in computing clusters, and the MIT Xgrid relies on such a system for managing larger datasets. However, without a fully Kerberized Xgrid cluster, tasks on the worker nodes are owned by an unprivileged nobody user. Allowing such a user write access to centralized storage may be inappropriate in some deployments. Currently it doesn't appear possible to declare a single best practice for returning output datasets; the authentication requirements and network topology of the cluster, the amount of data being transferred, and the level of trust accorded the agents must all be taken into account when deciding on a protocol.

A catalog of metadata about individual files is an essential tool for assembling datasets satisfying some set of physics requirements. OS X offers an elegant option for implementing such a catalog. The HFS+ filesystem in OS X supports the storage of an arbitrary set of metadata in the extended attributes of files, and the operating system maintains a searchable catalog of a subset of these metadata which is automatically updated at the kernel level. Furthermore, the system exposes an API that allows users to customize the set of metadata imported into the catalog for particular filetypes. As a result, one can produce an instantly updated file catalog on each machine by writing a plugin that defines the physics metadata specification for data files of interest. This file catalog could be a natural complement to a HENP Xgrid cluster; details of the implementation are currently under study.

4. STAR Unified Meta-Scheduler

Today's HENP scientists have access to a very wide variety of evolving systems for doing distributed computing. Mastering the workflows and syntaxes of each of the systems one encounters on a regular basis can be a daunting task that gets in the way of actual research. The STAR experiment's response to this challenge is the STAR Unified Meta-Scheduler (SUMS)[11]. SUMS provides a common front-end to the compute elements used by STAR and simplifies the management of the large numbers of batch jobs needed to analyze a given dataset. SUMS supports Grid submissions using Condor-G as well as direct submissions to clusters running LSF, SGE, Condor, and now Xgrid. The ability to use SUMS to drive the MIT Xgrid cluster led to an immediate increase in the number of users and the number of jobs being submitted. Over the course of time LNS scientists involved in research unrelated to STAR joined the user base as well, and in each case used SUMS to submit their jobs. Researchers working on experiments as varied as SNO, KATRIN, CMS, and even the ILC have at various times used SUMS on MIT's Xgrid cluster to generate simulations or analyze data. The cluster now offers access to spare cycles from nearly 100 cores in the Laboratory for Nuclear Science, the Department of Urban Studies and Planning, and the School of Architecture + Planning. This unique combination of diverse resource contributions and broad research group adoption is a compelling demonstration of the promise of a campus grid.

5. Globus Toolkit Integration

The Globus Toolkit[12] enables grid job submission to a variety of Local Resource Managers (LRMs) using the GRAM framework. LSF, PBS, and Condor are supported directly in a standard Globus installation, and other Local Resource Managers such as SGE can be configured using third-party packages. We are working on adding Xgrid to the list of LRMs supported by GRAM with a view towards future integration of an Xgrid compute element into the Open Science Grid. Like much of the Toolkit, GRAM exists in a web services flavor (WS-GRAM) and a pre-web services one (pre-WS GRAM). pre-WS GRAM uses LRM-specific Perl scripts to handle most of the details of job submission workflows; in WS-GRAM many of the duties that used to be handled by that Perl script are now handled by other parts of the toolkit. In particular, WS-GRAM uses compiled scheduler-event-generator (SEG) modules to monitor the logfiles of LRMs for job state changes instead of querying the LRM for that information directly.

An Xgrid job manager script and SEG for GRAM have been implemented and are currently being tested at MIT. Xgrid's workflow requires an explicit request to export logfiles and data out of the controller once a job has finished. The job manager script's stageOut method can make this request, but stageOut is only used in pre-WS GRAM. In the current version of WS-GRAM it may be necessary to specify a hold on each Xgrid job that takes effect once the SEG reports the job as Finished (that is, the results have been transferred back to the controller), and then submit a Fork job to export the Xgrid results. Trunk sources for WS-GRAM feature new functionality that should eliminate the need for this workaround. Other planned improvements include support for Kerberos[13] authentication through automatic generation of ticket granting tickets from Grid Security Infrastructure certificates.

6. Summary

This contribution presented a unique integration of Xgrid distributed computing technology in HENP. Xgrid is an attractive choice for resource-harvesting clusters due to its simple configuration and flexible communication protocol. An Xgrid cluster at MIT's Laboratory for Nuclear Science currently enables scientists from a variety of research groups in the Laboratory to generate simulations and analyze data using the spare cycles of computers throughout the Institute. The STAR Unified Meta-Scheduler provides a front-end for evolving distributed computing and grid technologies; Xgrid support in SUMS allows physicists to make optimal use of the MIT cluster without needing to learn the specifics of this new batch system. Finally, an ongoing project to support Xgrid in the Globus Toolkit GRAM will allow Xgrid-based compute elements to fully participate in the HENP Grid.

7. Acknowledgements

This work was supported by the Offices of Nuclear Physics and High Energy Physics within the U.S. Department of Energy Office of Science, and the U.S. National Science Foundation. Apple Inc. provided technical insights and feedback through the Worldwide Developers Conference. We would like to thank the organizers of the CHEP 2007 conference.

8. References

[1] URL `http://www.apple.com/macosx/features/xgrid/`
[2] URL `http://developer.apple.com/cocoa/`
[3] URL `http://developer.apple.com/documentation/Darwin/Reference/ManPages/man1/xgrid.1.html`
[4] Rose M 2001 The Blocks Extensible Exchange Protocol Core RFC 3080 (Proposed Standard) URL `http://www.ietf.org/rfc/rfc3080.txt`
[5] URL `http://finkproject.org`
[6] URL `http://developer.apple.com/tools/xcode/`
[7] Brun R and Rademakers F 1997 *Nucl. Instrum. Meth.* **A389** 81–86
[8] Agostinelli S *et al.* (GEANT4) 2003 *Nucl. Instrum. Meth.* **A506** 250–303
[9] Sjostrand T, Mrenna S and Skands P 2006 *JHEP* **05** 026 (*Preprint* `hep-ph/0603175`)
[10] Hanushevsky A *et al.* 2004 *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP 2004)* URL `http://xrootd.slac.stanford.edu`
[11] Lauret J *et al.* 2004 *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP 2004)*
[12] Foster I 2005, *IFIP International Conference on Network and Parallel Computing* (Springer-Verlag LNCS 3779) pp 2–13
[13] Kohl J and Neumann B 1993 The Kerberos Network Authentication Service (V5) RFC 1510 URL `http://www.ietf.org/rfc/rfc1510.txt`