FERMILAB-CONF-04-470-CD

# SAMGRID EXPERIENCES WITH THE CONDOR TECHNOLOGY IN RUN II COMPUTING

A. Baranovski, L. Loebel-Carpenter, G. Garzoglio, R. Herber, R. Illingworth, R. Kennedy, A. Kreymer, A. Kumar, L. Lueking, A. Lyon, W. Merritt, I. Terekhov, J. Trumbo, S. Veseli, S. White, FNAL, Batavia, IL 60510, USA, FNAL, Batavia, IL 60510, USA
R. St. Denis, University of Glasgow, Glasgow G12 8QQ; United Kingdom
S. Jain, A. Nishandar, University of Texas at Arlington, Arlington, TX 76019, USA

*Abstract*

SAMGrid is a globally distributed system for data handling and job management, developed at Fermilab for the D0 and CDF experiments in Run II. The Condor system is being developed at the University of Wisconsin for management of distributed resources, computational and otherwise. We briefly review the SAMGrid architecture and its interaction with Condor, which was presented earlier. We then present our experiences using the system in production, which have two distinct aspects. At the global level, we deployed Condor-G, the Grid-extended Condor, for the resource brokering and global scheduling of our jobs. At the heart of the system is Condor's Matchmaking Service. As a more recent work at the computing element level, we have been benefiting from the large computing cluster at the University of Wisconsin campus. The architecture of the computing facility and the philosophy of Condor's resource management have prompted us to improve the application infrastructure for D0 and CDF, in aspects such as parting with the shared file system or reliance on resources being dedicated. As a result, we have increased productivity and made our applications more portable and Grid-ready. Our fruitful collaboration with the Condor team has been made possible by the Particle Physics Data Grid.

## INTRODUCTION

The SAMGrid project at Fermi National Accelerator Laboratory has been developing solutions for data handling, as well as related job and information management, for the Run II experiments, D0 and CDF. Started as a data handling system primarily for D0, it grew to embrace recent Grid computing developments, see [1] and references therein. The Job and Information Management (JIM) is a principal SAMGrid component co-sponsored by the Particle Physics Data Grid (PPDG), a US Grid initiative that brings together Computer Scientists and Grid Application projects such as SAM, in order to develop end to end solutions for HEP experiments. Our primary stakeholder experiment is D0, with CDF getting increasingly more involved into SAMGrid computing.

The first and foremost benefit for JIM, and more generally SAMGrid, in PPDG and Grid has been our collaboration with the Condor team at the University of Wisconsin. Specifically, we have been studying, adapting and enhancing the Condor technology. We view Condor as a system for management of distributed resources, where “distributed” may mean anything from a cluster to a world-wide system, and “resources” may mean anything from individual computers (or CPUs therein) to whole participating sites.

We have found two distinct applications for the Condor technologies, and the paper is correspondingly organized in two main sections following an overview of our architecture. First, we describe the global-level deployment of Condor-G. Second, we show how our experiences with Condor-managed local computing cluster at Wisconsin have helped us better schedule and manage Run II jobs. Throughout the presentation, we emphasize the design solutions that we have developed.

## OVERVIEW OF SAMGRID ARCHITECTURE

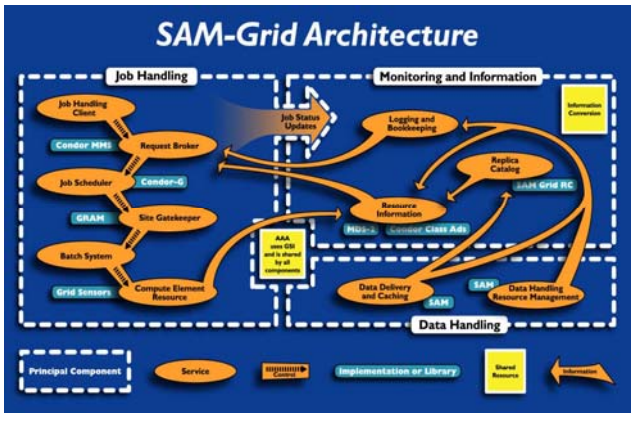


**Figure 1. SAMGrid architecture.**

There are three principal components in the SAMGrid architecture in Figure 1. These are data handling[2], job services and information management. Data handling is the flagship component, see [3,4], yet we artificially reduce its size in the above picture to highlight the more recent developments.

## JOB MANAGEMENT AT THE GRID LEVEL WITH CONDOR-G

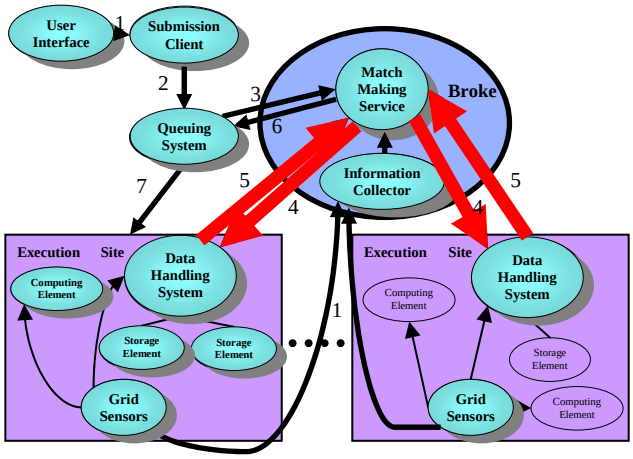


**Figure 2. Job Management in SAMGrid.**

At the high (Grid) level, user jobs are described logically as requests; for example, a job of *type* Monte-Carlo has MC Request ID, specification of the D0 release D0 version, data input (including minimum-bias mix-in) as SAM dataset(s), any other control parameters and, lastly, the *size* of the job such as the total number of events desired. The job is presented to the Grid scheduler through extremely thin user interface. The scheduler (queuing system) communicates with the Request Broker to determine the Grid site for the job to run, see Figure 2.

We emphasize the hierarchical structure of the job, which is unique to SAMGrid A single Grid job is mapped onto many local (i.e. materialized in the batch system of the site) jobs. In our opinion, this provides a clear, hierarchical view of the jobs where the Grid-level job management deals only with high-level jobs, easily understood by user scientists, and detailed decomposition of the job into executable (in the batch system) tasks is left for the Fabric-resident services. This "divide and conquer" paradigm therefore facilitates job management and *scales* well with the workload increase.

Strictly speaking, our job structure is such that a Grid job is mapped (decomposed) onto one or more *cluster* jobs, each cluster job being scheduled at one site; it is the cluster job that is decomposed into a collection of local jobs. As of the time of wring this document, however, Grid job corresponds to only one cluster job, and in the remainder of the document we use cluster job and Grid job interchangeably.

Consider the mapping of our job management components onto Condor Components. In SAMGrid, the Request Broker is embodied as the Condor-G Matchmaking Service (MMS). Although MMS was a part of the classic Condor system, it was absent from the Condor-G technology due to initial implementation limitations. We have decided to reuse the MMS concept at the Grid level and forgo implementation of our own parochial Broker.

Next, we enhanced the MMS with the ability to dynamically retrieve additional information from (resource) advertisers. We accomplished this through introduction of *functions* into ClassAds. Previously, the ClassAd system manipulated with constants, variables and expressions; see [5] for more details. We applied our extension in querying the advertising sites for the input data available for the job being matched, thus linking, for the time in Grid history, of scheduling of Grid jobs with the data handling system, see [1].

Next, we decoupled the client (User Interface in the Figure) from the submission/queuing machines. The former has become easier to install (i.e. no longer requires root privilege, no daemons running, etc) and will probably eventually become a Web-enabled client. The latter is a full-fledged spooling and scheduling server (Condor Schedd) running on a "dedicated" machine.

Our miscellaneous extensions for Condor-G included, among other features, the rematch possibility. This allowed the system to recover from "mis-matched", e.g., routing a job to a site with outdated **gridmapfile** or other fatal obstacle to accepting the job.

## GRID JOB INSTANTIATION AT A LARGE LOCAL CONDOR CLUSTER

In the course of SAMGrid deployment, we have used the large Condor cluster at the University of Wisconsin, the homeland of the Condor project. Obviously, this cluster is not dedicated to D0, and our usage thereof has been made possible exclusively by virtue of Grid collaboration of SAM. (Specifically, this is D0-Condor collaboration under the auspices of the Particle Physics Data Grid consortium). What is important for the purposes of this paper is that the Wisconsin cluster has been our first, and foremost true grid cluster. In practice, this means that we received a good share of resources for D0 computations and we have enjoyed the support of the resource owners, but we never owned any piece of the cluster.

Such an environment has been ideal for SAMGrid for understanding and solving the problem of Grid job instantiation at the Fabric, which we have done in the context of running D0 Monte-Carlo jobs. In this section, we define the "job instantiation" problem and then describe how the Wisconsin cluster configuration imposes constraints and thereby forces one to think in terms of proper abstractions. We then outline our solutions.

*Job instantiation* at the site physically means submission of multiple *local jobs* to the *batch system*, including preparation of all the necessary *non-Physics data* as "input" and subsequent retrieval of the *small output* (i.e. output files such as logs that are not destined for a full-fledged data handling system such as SAM). The movement of non-Physics input data is done from/to SAMGrid submission site, where the (Grid-level) jobs are spooled, and which in turn is typically close to the SAMGrid client site from where the user submits the Grid

jobs. It happens at the *Grid to Fabric boundary* and in SAMGrid is carried out by the Grid to Fabric Interface Services, see [6].

Any member of HEP computing community is familiar with the application-imposed complexities in the instantiation and management of a real Run II physics. Hundreds, often thousands of small files must be supplied with the job in a manner that is efficient enough so as not to break local file transfer mechanisms. The jobs shouldn't interfere even when several of them are scheduled on a single node. The number of local jobs running in parallel must be determined so as to maximize the probability of job completion (within the batch system imposed boundaries) yet not to have too many small jobs producing too many small output files.

The problem of job instantiation is made more difficult by heterogeneity of participating clusters, in terms of directory structures, shared file systems, conventions for naming standard output/diagnostic files, designated mechanisms of *intra-cluster* small file transfers, etc, which is not managed by any "standard" Grid (or not) software. Consequently, pre-SAMGrid solutions typically made a number of simplifying assumptions, of which the following two are the most profound:

1. Some experiment-specific software is pre-installed cluster-wide. The software may be something as conspicuous as the experiment code or something as subtle as the Python language interpreter.

2. There is a utopistic "no-cost", transparent and efficient shared file system, epitomized by the *home area* concept, which the jobs and/or their wrapping scripts may use.

Obviously, these assumptions hinder severely our move towards Grid computing. Particularly flagrant is the "home area" where the jobs can in fact deposit files expected to stay there even after job completion. Overload of (e.g. NFS) shared file system server by uncontrolled, implicit file transfers, or overflow of the user "quota" are only some of the problems. What is more, home area is a ramification of a static "account" concept which cannot possibly make its way into modern Grid computing – imagine jobs from different users from the same VO colliding on a file with a name like "~/seed".

The Wisconsin cluster has helped us enormously by breaking these unrealistic assumptions. First, our Grid jobs were mapped to a non-existent local username such as "nobody" which didn't have a writeable "home" on the worker node. Second, instead of implicit file transfers in a shared file system, Condor provides mechanisms for explicit pre-staging of the job's small (i.e. non-data) files. Third, jobs run in dynamically created scratch space and are required to "carry away" all the small files they produce (i.e. log files) by themselves upon completion.

In SAMGrid, we strongly believe that the services provided by this local Condor cluster (scratch management, explicit intra-cluster small file transfers) and the overall cluster configuration (no shared "home" or other file system) are the correct approach towards making a true Grid Fabric from computing clusters. The SAMGrid job instantiation at the Grid-Fabric has been inspired by this configuration and our solution worked immediately on other (non-Condor) clusters, which made the ongoing SAMGrid deployment possible.

## *SAMGrid solutions*

Due to space constraints, we merely sketch our solutions for the problem of Grid job instantiation at the Fabric, see Figure 3.

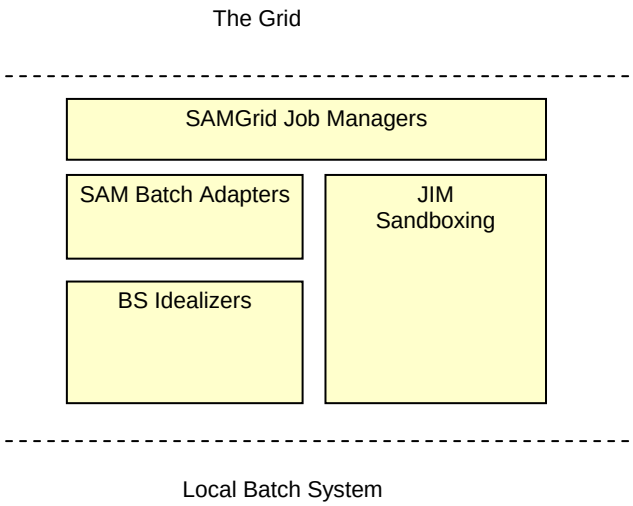


**Figure 3. SAMGrid services at the Grid-Fabric boundary.**

**Batch System Idealizers**. We adopted (but not invented) a term that despite its appearance is intended to represent a serious concept. These, as the name implies, "idealize" the batch systems to make their interactions with Grid machinery easier, by "mitigating" any imperfections and adding any "missing" features. Mitigation includes:

- retries in lookup commands for certain batch systems,
- generation of easy to parse output (batch system commands return output that's usually too terse or too verbose),
- compensation for confusing exit status from batch system commands.

Added features include:

- grouping of jobs for all batch systems by an attribute such as generalized "project" (i.e. in SAMGrid rather than SAM sense),

- local scratch management on the worker nodes, i.e. setup and cleanup of the scratch space before/after user job execution,
- (optional) explicit preference/avoidance of nodes that are /are not well suited for the grid job(s) in question

**Batch System adapters**. These were originally intended for use by the "sam submit" command, which in turn was intended to provide the correct interface to submit SAM analysis job to a batch system, and performing actions such as starting/stopping of a SAM project. In the expanded job management scheme, provided by the JIM and other SAMGrid tools, this package serves as the configuration tool for the job submission/lookup/kill commands, implemented in the aforementioned idealizers, i.e. the adapters for jobs coming from the Grid must be configured to use the idealizer appropriate for the local batch system.

The difference between "adapters" and "idealizers" is that the former provide uniform interface to the batch system, whereas the latter provide the scripts that actually correctly implement these interfaces. For example, the adapter concept contains an interface to lookup a job in the batch system, and an idealizer will actually perform the lookup, handle some of the errors, and return a complex, multi-line string that is nevertheless easy to parse. In a broader sense, "adapters" include "idealizers".

**JIM Sandboxing**. This service is provided within the jim_sandbox software package and is documented therein. "Sandboxing" in SAMGrid refers to the ability to transfer and initialize all the relevant input files for the user job, as well as correct collection and return of "small output. For input sandboxing, a staged *bootstrapping* process is used whereby each subsequent stage uses results of the previous stage, the last and most advanced stage being retrieval of a small input dataset through the SAM data handling system.

**SAMGrid job managers**. These implement the services of grid job instantiation at the execution site, by means of mapping a logical grid job definition (with details provided by e.g. SAM Monte-Carlo request system) to set of local jobs submitted to the batch system. Our job managers come with the jim_job_managers software package and are installed into the Globus job manager area. When activated, they receive the job request via the standard GRAM protocol and perform multiple creation, submission, lookup and kill of the local jobs comprising the Grid job. In addition, they allow for XMLDB-based monitoring of Grid jobs which is at the heart of JIM Grid job monitoring.

## CONCLUSIONS

Through the collaboration with the Condor team, the SAMGrid project benefitted from the Condor technology in two ways, Grid-level and cluster-level.

At the Grid level, we have been able to deploy an enhanced matchmaking service instead of developing VO-specific (or even SAMGrid-specific) resource broker

At the cluster level, we have enjoyed dealing with the proper abstractions that present a true-Grid, non-dedicated computing cluster with minimal assumptions to our system, which helped factorize our local architecture and simplify deployment elsewhere.

## ACKNOWLEDGEMENTS

The SAM-Grid team would like to thank our Condor collaborators and especially Alain Roy, Todd Tannenbaum and Peter Couvares, who worked on all the aspects ranging from design to support of our users. We acknowledge the Particle Physics Data Grid consortium which made this collaboration possible. Finally, we thank all the D0 and CDF collaborators whose continued use and critique of the SAMGrid system make our efforts worthwhile!

## REFERENCES

[1] I. Terekhov, et al. "Grid Job and Information Management for the FNAL Run II Experiments", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003.

[2] I. Terekhov, et al. "Distributed Data Access and Resource Management in the D0 SAM System", in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, Aug. 2001. See also SAM articles in these Proceedings.

[3] G. Garzoglio, et al. "The SAM-GRID project: architecture and plan.", in Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 423 – 425.

[4] I. Terekhov et al., "Meta-Computing at D0"; in Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 402 – 406.

[5] A. Baranovski, *et al*. "Management of Grid Jobs and Data within SAMGrid", Cluster Computing 2004, Sep. 2004, La Jolla, CA, to appear in Proceedings.

[6] G. Garzoglio, et al. "The SAM-Grid Fabric services", talk at the IX International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-03), Tsukuba, Japan; to appear in Nuclear Instruments and Methods in Physics Research, Section A