# USE OF CONDOR AND GLOW FOR CMS SIMULATION PRODUCTION

D. Bradley, S. Dasu, M. Livny, V. Puttabuddhi, S. Rader, W.H. Smith
University of Wisconsin, WI 53706, USA

*Abstract*

The University of Wisconsin distributed computing research groups developed a software system called Condor for high throughput computing using commodity hardware. Condor allows building an enterprise level grid. Several UW departments have Condor computing pools that are integrated in such a way as to flock jobs from one pool to another where resources may be available. An interdisciplinary team of UW researchers recently built a new distributed computing facility, the Grid Laboratory of Wisconsin (GLOW) forming a new pool for campus research computing. We have built a scalable job submission and tracking system called Jug using Python and mySQL that enabled us to scale to run hundreds of jobs simultaneously. Jug also ensured that the data generated is transferred to US Tier-I center at Fermilab. In this paper we discuss our experience and observations regarding the use of opportunistic resources at UW for CMS simulation production in the past year.

## INTRODUCTION

The Large Hadron Collider (LHC), which is under construction at the CERN laboratory, is designed to definitively explore the higgs sector and the electro-weak symmetry breaking mechanism (EWSB). The LHC will collide protons on protons, at 14-TeV center of mass energy, at high luminosities. The Standard Model processes and any new physics phenomena that may be responsible for EWSB, e.g. Super Symmetry (SUSY), will be studied in the resultant TeV scale parton-parton interactions observed by the CMS detector [1], which is also under construction now. Unfortunately, both the rate of the strong interaction physics at electro-weak scale, and per event hadron mulitiplicity are very large. At design luminosity, an average of 17 collisions occur for each 25 ns crossing time, leading CMS to witness a billion proton-proton interactions per second. The LHC events have an average charge particle multiplicity of about 1000. Extraction of signals, particularly those of the higgs decays, from the profusely produced Standard Model background requires exploration of low branching fraction leptonic or photonic modes with good energy resolution. Therefore, CMS is designed with good resolution and high degree of segmentation, and is capable of withstanding a very high rate environment. The trigger and event filter farm discard the well understood background, while retaining the interesting high energy physics of 1-MB events at the rate of 100 Hz. Simulations of physics events, both signal and QCD background are critical not only to characterize the physics performance of the detector and trigger systems, but also for the preparation of sophisticated reconstruction software and analysis tools. To that end, CMS has set itself a challenge to produce 50 million events in Fall 2003 and reconstruct them at the rate of 25 Hz during Spring 2004 [2]. This challenge required far more CPU power than is available at CERN. Collaborating CMS institutions from around the world were invited to help in this simulation production. Our group at University of Wisconsin has taken this challenge to use our facilities to produce data for CMS. These data are available to all CMS physicists including our group.

## CONDOR

The Condor Project [3] at the University of Wisconsin developed, implemented, and deployed, mechanisms and policies that support High Throughput Computing on large collections of distributively owned commodity-computing resources. The Condor software provides a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management.

While providing functionality similar to that of a more traditional batch queueing system, Condor's novel architecture allows it to succeed in areas where traditional scheduling systems fail. Its unique mechanisms enable Condor to effectively harness wasted CPU power from idle resources, e.g., when a particular research group is otherwise busy and not using its resources, Condor can flock jobs from other research groups to it. Condor can preempt the lower priority jobs when a higher priority (e.g., owner's job) is submitted. Upon preemption, Condor is able to transparently produce a checkpoint and migrate a job to a different machine when it becomes available if compiled specially with Condor provided system software libraries (The Standard Universe). For normal jobs, Condor reruns the job automatically (The Vanilla Universe) until it completes successfully or terminates due to user code errors. Jobs that are able to resume processing where left off at the time of preemption, i.e., user level checkpointing, can use the resources very efficiently by integrating run periods that are shorter than the complete job.

Further, Condor does not require a shared file system across machines - if no shared file system is available, Condor can transfer the job's data files on behalf of the user, or Condor may be able to transparently redirect all the job's I/O requests back to the submit machine. As a result, Condor can be used to seamlessly combine all of an organization's computational power into one resource.

Condor provides an extremely flexible and expressive framework for matching resource requests (jobs) with resource offers (machines). Jobs can easily state both job requirements and job preferences. Likewise, machines can specify requirements and preferences about the jobs they are willing to run. These requirements and preferences can be described in powerful expressions, resulting in Condor's adaptation to nearly any desired policy. Condor also implements a mechanism to chain a group of jobs using a Directed Acyclic Graph (DAGman) [4]. DAGman ensures that previous stages of jobs are completed before proceed to the next.

Condor can be used to build Grid-style computing environments that cross administrative boundaries. Condor's "flocking" technology allows multiple Condor compute installations to work together. Condor incorporates many of the emerging Grid-based computing methodologies and protocols. For instance, Condor-G [5] is fully interoperable with resources managed by Globus [6]. Although we only describe our work with pure Condor here, we have also integrated our resources in US Grid3 [7] effort.

## CONDOR & GLOW RESOURCES

There are several Condor pools on the UW campus. The largest of those is operated by the Department of Computer Science with about 800 1 GHz Intel Pentium-III processors. The High Energy Physics group has about 100 CPUs, bulk of which are 2.4 GHz Intel Xeon processors. The Grid Laboratory of Wisconsin (GLOW) [8] is a new inter-departmental research computing initiative with about 800 2.8 GHz Intel Xeon processors being commissioned. These facilities are jointly owned by High Energy Physics (CMS), Astrophysics (IceCube), Medical Physics, Chemical Engineering and Biochemistry research groups. The racks of computers located in these departments are connected using a 1 Gbps campus network backbone. Additional Condor pools from other research groups can be included in this project. We have used the Condor software to configure these pools such that jobs submitted from the High Energy Physics resources are seamlessly flocked to any available resource on the campus. Therefore, in addition to our dedicated resources we have access over a thousand CPUs currently scaling to twice that within a year.

In order to utilize this large CPU power effectively we have also assembled a storage facility for HEP group with an aggregate of about 12 TB RAID5 disk. The GLOW project will increase the storage capacity to about 70 TB.

## CMS APPLICATIONS

The CMS event simulation involved multiple steps; physics simulation (CMKIN [9]), detector simulation (CMSIM [10] or OSCAR [11]) and electronics signal simulation (ORCA [12]). The FORTRAN GEANT3 [13] based CMSIM program was replaced with C++ GEANT4 [14] based OSCAR program partway through this event production challenge.

The CMKIN and CMSIM programs could be compiled with special Condor libraries and are capable of running in the Standard Universe. The OSCAR and ORCA programs use multi-threading and dynamic libraries and are capable of running in the Vanilla Universe only. At the time of this effort the OSCAR and ORCA programs were not able to self check-point their progress, and are required to restart from the very first event upon preemption.

The simulation consisted of several physics datasets comprising of hundreds of thousands of events each. Each dataset was divided into bunches of events termed as assignments which are handed to various computing sites participating in the production. The assignments are further broken into jobs of the size of few hundred events each. Depending on the dataset type and the stage of processing and the machines selected to run the jobs, they can last from an hour to several days each. Often the job running also involved staging in data for input and staging out data after completion over the wide area network. Recovering from job transfer failures and ensuring integrity of the data transferred posed a non-trivial additional burden.

## JUG

Managing hundreds of jobs simultaneously keeping the integrity of several steps required an automated tool. Therefore, we developed software called Jug [15] in Python, to run on top of any underlying batch system, in our case Condor. The object-oriented scripting provided by Python was especially suited for rapid development. Jug used a MySQL [16] database to keep track of a chained bunch of jobs persistently. New stages of processing could be added dynamically if needed. Jug essentially provided a persistent DAG. The usage of the database was especially important because we could restart essentially any component of the system without causing complete shutdown of processing. Jug uses Master-Worker paradigm to delegate either data stage-in, processing or data stage-out to storage and processing workers. The number of storage and processing workers could be scaled as needed. The storage workers used Stork and grid-ftp to transfer data to FNAL dCache/Enstore [17] facility. The processing workers submitted jobs to Condor. The jobs submitted to Condor flocked to HEP, CS or GLOW Condor pools. Jug ensured that jobs progressed to the next stage only upon successful completion of the previous stage. Failed jobs moved to the rear of the job queue so that pathologically persistent failures do not stall the system usage. The operator could attend to those failed jobs at a convenient time.

## RESULTS

By exploiting special features of Condor such as check-pointing and remote IO we have generated over 9 million fully simulated CMSIM events, which is about a quarter of all the data produced by CMS groups at that time. We were able to harness about 260 CPU-days per day for a period of 2 months over the operational period of late fall 2003. We have used 500 CPUs concurrently when opportunity arose to exploit unused resources in laboratories on our campus. At that time GLOW facility was not yet commissioned. The checkpoint capability of CMSIM was especially important at this time to opportunistically use the facilities belonging to CS department.

We changed to use OSCAR for detector simulation and ORCA for electronics simulation in December 2003. Unfortunately, check-pointing of OSCAR and ORCA was not feasible at that time. Therefore, we were forced to accept large job failure rate due to preemption on non-owned resources. We also suffered from difficulties in data transfer. Jug was especially important to track the jobs to successful completion. In spite of these difficulties we were able to produce 1 million OSCAR events in time for April 2004 CMS data challenge at CERN. Since then, we are using both Condor and GLOW to produce data for CMS and local use. We have produced over 3 million OSCAR events and 6 million ORCA events. ORCA production required installation and commissioning of dCache storage management system.

In summary, during the past year our contribution to the CMS data production challenges is one of the largest. More importantly we were able to accomplish this with minimal investment from CMS funds by exploiting resources opportunistically. Development of Jug allowed us to minimize the personnel effort devoted to production management.

With the expertise gained from our production efforts, especially in dCache usage, we have adapted our environment to provide analysis resources. We are able to run about a 100 analysis jobs simultaneously using dCache served from about 12 storage pools.

## CONCLUSIONS

From our experience with the use of Condor and GLOW shared computing resources on the UW campus we conclude that:

- Condor software allows us to build and efficiently use computer pools that are shared by groups across a campus.
- Pooling resources together allows all groups to benefit. Because there are idle resources in one or the other pool at any time, it provides every group an opportunity to exploit more than their fair share. The aggregate usage efficiency is very high.
- Groups with robust, checkpointable software gain the most. This is especially suitable for high energy physics simulations which can run in the background and exploit idle resources.
- By combining resources on campus, small research groups can use idle resources in collaborating pools opportunistically.
- With appropriate agreements of priority and burst utilization policies, a group can command large level of resources that are usually available only at national laboratories

We believe that implementation of shared computing facilities such as GLOW will ultimately lead to democratization of computing access. This will present an opportunity for scientists at Universities to test their innovative ideas promptly.

## ACKNOWLEDGEMENTS

This work is supported by the US National Science Foundation, the US Department of Energy and the Wisconsin Alumni Research Foundation.

## REFERENCES

[1] http://cmsinfo.cern.ch
[2] David Stickland, Plenary Talk, These Proceedings.
[3] http://www.cs.wisc.edu/condor
[4] http://www.cs.wisc.edu/condor/dagman/
[5] http://www.cs.wisc.edu/condor/condor-g/
[6] http://www.globus.org/
[7] http://www.ivdgl.org/grid3/
[8] http://www.cs.wisc.edu/condor/glow
[9] http://cmsdoc.cern.ch/cms/generators/
[10] http://cmsdoc.cern.ch/cmsim/cmsim.html
[11] http://cmsdoc.cern.ch/oscar/
[12] http://cmsdoc.cern.ch/orca/
[13] http://wwwasdoc.web.cern.ch/wwwasdoc/geantold/GEANTMAIN.html
[14] http://wwwasd.web.cern.ch/wwwasd/geant4/
[15] http://www.hep.wisc.edu/cgi-bin/cms/JugMaster.cgi/doc/index.html
[16] http://www.mysql.org
[17] http://www.dcache.org