Campus Grids

Derek Weitzel
Computer Science and Engineering
University of Nebraska–Lincoln
Lincoln, NE 66588-0115
dweitzel@cse.unl.edu

November 3, 2010

Abstract

At many universities, each department maintains a independent cluster for their researchers. Many of these clusters run under utilized. In this paper we will describe a framework and technologly that where used to link departmental clusters such that submission at one cluster could lead to execution on another. This framework is then further expanded to operating on a national production grid.

1 Introduction

The problem we have solved

- Department clusters waste power by being under utilized significant portions of time.
- Researchers have peaks in usage (think paper writing) and need overflow capacity.
- Overflow capacity for their local cluster.

Why the problem is not already solved or other solutions are ineffective in one or more important ways

- A common solution to linking clusters is condor flocking. Flocking requires every cluster to run condor daemons on their nodes.
- Another solution is single vendor/software solution. Both PBS and SGE can create a grid of clusters. This again requires buy-in from all departments on campus.
- Placing globus gatekeepers on each cluster would allow jobs to be submitted to each cluster without modifying the underlying batch system. But, this would require a higher layer of abstraction over the globus gatekeepers to optimally balance load between clusters. Also, it does not provide a method for transparent execution on other clusters.

Why our solution is worth considering and why is it effective in some way that others are not

- The framework described in this paper is designed as a modular framework that will allow clusters to overflow onto each other and to the grid. This framework requires running condor on only one node in the cluster. Each job that comes from another cluster will go through the default scheduler, whether it's PBS, SGE, or LSF.

How the rest of the paper is structured

The rest of this paper first discusses related work in Section 2, and then describes our implementation in Section 3. Section 4 describes how we evaluated our system and presents the results. Section 5 presents our conclusions and describes future work.

2 Related Work

Other efforts that exist to solve this problem and why are they less effective than our method

- Globus is a translation layer between the globus specific Resource Specification Language, and the local resource manager. It has been very successful in that is has a large install base. Globus also has deep integration with standard grid credentials.
- Condor Flocking

Other efforts that exist to solve related problems that are relevant, how are they relevant, and why are they less effective than our solution for this problem

- Diagrid
- GlideinWMS
- Panda

3 Implementation

What we (will do | did): ***Our Solution***

- Created a campus grid integrating 3 clusters on a campus into a grid. Submission to any of the clusters could overflow to the other 2.
- Overflow to the grid
- Using offline ads to efficiently match jobs to glideins on the non-condor cluster.

How our solution (will | does) work

- Campus Factory is the heart of the operation. It provides all throttling and logic to the submit of glidein jobs to the non-condor cluster.
- Glideins are submitted to the non-condor cluster.
- Flocking is enabled between the condor clusters, and the non-condor clusters. The non-condor clusters run the condor daemons on one node that will handle flocking and submission of glidein jobs. This node can also serve as the local condor queue for users to submit to.
- Condor & BLAHP provides the interface to the local batch system. It is developed as a part of glite, and is included in the standard condor distribution.
- OfflineAds are used to efficiently match jobs to potential slots on a machine. These were developed because we do not always want 5 idle jobs running on a cluster. We would rather have a few jobs submitted per day, and only submit more glideins if jobs will run there.

4 Evaluation

How we tested our solution

- Production jobs on Firefly

How our solution performed, how its performance compared to that of other solutions mentioned in related work, and how these results show that our solution is effective

- Faster job submission and start than globus. Should be easy to show.

Context and limitations of our solution as required for summation

- Therefore, my solution is awesome.

5 Conclusions and Future Work

The problem we have solved

- The framework that I have described here creates a grid of clusters that can overflow to each other, and out to the other grids such as the Open Science Grid, or other campuses.

Our solution to the problem

- Combination of Condor, Glidein, and BLAHP.

Why our solution is worthwhile in some significant way

- This solution keeps the administrator in control of priorities and access policies. It can also follow's the campuses security model.

- Provides a cookie cutter, well packaged solution for campuses to deploy a campus grid.

What we will (or could) do next

- Scale this solution to other universities and institutions. Already doing this with the OSG Campus Grid Initiative.

References

[1] Anderson, J., Ramamurthy, S., Jeffay, K., "Real-Time Computing with Lock-Free Shared Objects", *Proceedings of the 16th IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, December 1995, pp. 28-37.

[2] Baruah, S., Howell, R., Rosier, L., "Algorithms and Complexity Concerning the Preemptively Scheduling of Periodic, Real-Time Tasks on One Processor," *Real-Time Systems Journal*, Vol. 2, 1990, pp. 301-324.

[3] Goddard, S., Jeffay, K. "Analyzing the Real-Time Properties of a Dataflow Execution Paradigm using a Synthetic Aperture Radar Application," *Proc. IEEE Real-Time Technology and Applications Symposium*, June 1997, pp. 60-71.