



Advance Reservations

A Theoretical and Practical Comparison of GUR & HARC

Yixin Wu*, Maria Cristina Tugurlan†, Gabrielle Allen‡

Louisiana State University, Baton Rouge

ywu@cct.lsu.edu, ctugur@math.lsu.edu, gallen@cct.lsu.edu



*Department of Computer Science, †Department of Mathematics, ‡Department of Computer Science & CCT

Introduction

New generations of scientific applications involve interactive visualization of large simulations, real-time data exchange between multi-model codes, complex workflows of distributed components, and interplay between live experimental devices and simulations. All of these activities require the ability to co-schedule or advance reserve compute resources, along with networks, data archives, and experimental or visualization equipment.

This poster reports on an investigation of two new tools for co-allocation, the Grid Universal Remote (GUR) and Highly-Available Resource Co-allocator (HARC), which have been deployed on the resources of the TeraGrid and LONI. We compare these components using seven different metrics, ranging from their system architecture, to their usability, and summarize their applicability for real application case studies.

Use-case scenario

In the GENIUS project, users perform brain blood flow simulations in support of clinical neurosurgery. Computational resources are used to simulate various cerebral vascular systems. There is a need for tools and policies to reserve and co-reserve high performance resources, and to geographically distribute single large scale simulations across multiple resources.

Users need to reserve a number of resources at the same time (Figure 1a) or multiple resources for different time periods (Figure 1b)

a) Meta-computing job

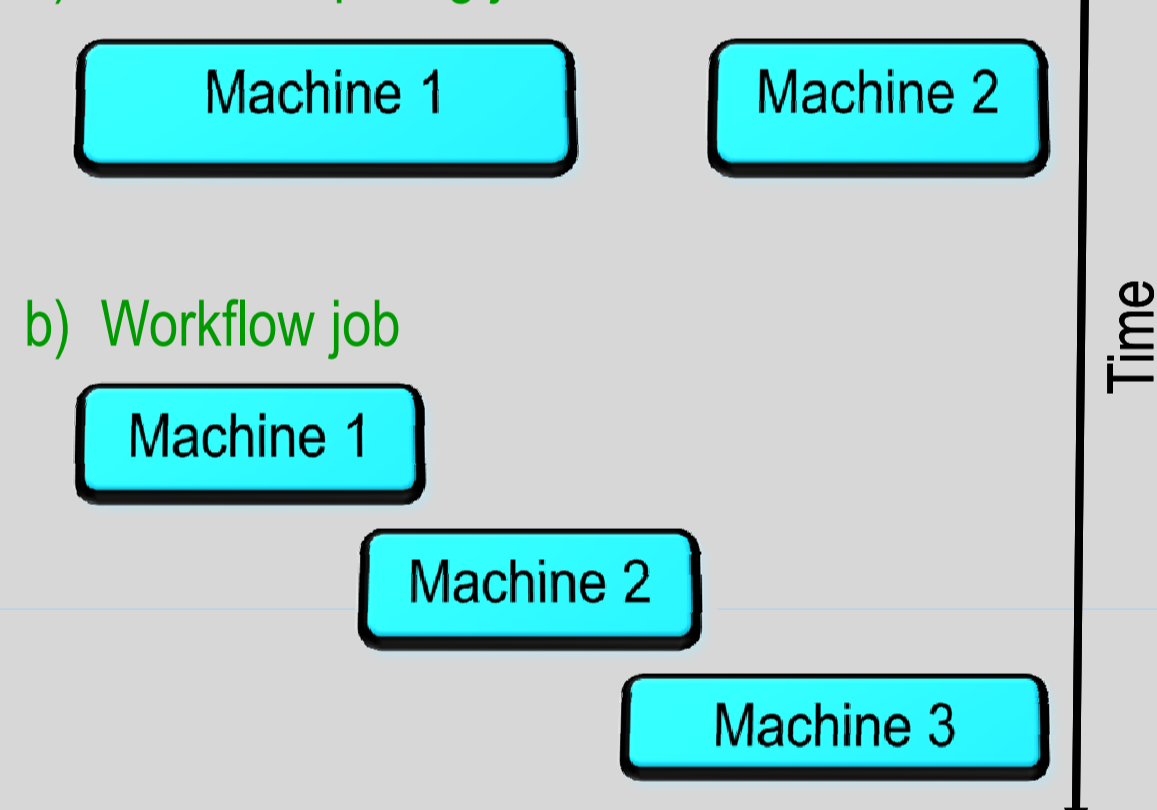


Figure 1

The two system that we are investigating were developed to substitute manual reservations. Users need the ability to create, manipulate or cancel a reservation. The basic steps are:

1. Book the resources;
2. Submit the jobs to the reservations;
3. Monitor the state of the reservations;
4. Cancel the reservations (if time remains).

Applications and Deployment

HARC used in:

- GENIUS project- Clinical Application requires reservations of high performance resources at the same or different time
- iGrid 2005 demo for distributed visualizations
- Cactus - Visualizations: simulations run on one machine and visualized the results on others
- HPC class at LSU – reserve light-path

GUR used in:

- SC'04: co-scheduling reservations across three platforms (SDSC DataStar, SDSC Linux, Purdue SP)
- Co-scheduling between SDSC and other sites

Infrastructure	Machine	HARC	GUR
TeraGrid	SDSC IA-64	Deployed	Deployed
	NCSA IA-64	Deployed	Deployed
	LONI QueenBee x86	Deployed	Deployed
LONI	Bluedawg IBM 575	Deployed	No
	Ducky IBM 575	Deployed	No
	Zeke IBM 575	Deployed	No
	Eric x86	In progress	No

Comparison of Systems

HARC

Architecture

HARC consists of (Figure 2) [1]:

- ✦ Client requests resource co-allocations via
- ✦ Acceptors (replicated TM) which make reservations by talking to
- ✦ Resource Managers which talk to local schedulers on each resource.

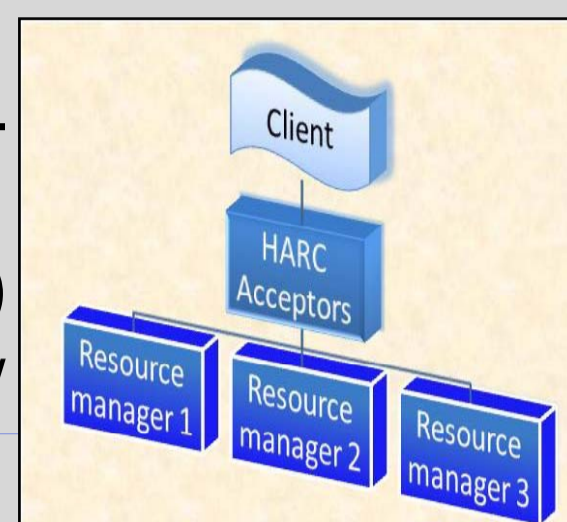


Figure 2

Phased Commit Protocol allows multiple resources to be booked in an all-or-nothing fashion (atomically) and guarantees a consistent response for clients and RMs, regardless of which Acceptor they talk to. HARC is designed to be extensible since new types of Resource Manager can be used without requiring changes to the Acceptor code [2] - Acceptors (install and maintain) - Additional layer

Installation

Java client API and commands

Platform: UNIX

Prerequisite: Java 1.5.0 or later, JRE, CoG JGlibus 1.4, X.509 credential

Components:

- ✦ Client (V1.9.3) (install and configure -trivial)
- ✦ Acceptors (install and configure - hard) (infrastructure)
- ✦ RM (compute RM or network RM) (Perl) harder to install than Client, but much easier than Acceptors

Security

- ✦ Based on X.509 credential
- ✦ Messages are all XML over HTTPS (encrypted)

Portability

- ✦ Designed for Unix platform
- ✦ Java Code
- ✦ Under active development and not in production stage yet
- ✦ Local resource manager works with:
 - ✦ PBSPro
 - ✦ Torque/Maui, Torque/Moab, Torque/Catalina
 - ✦ LoadLeveler
 - ✦ SunGridEngine 6.2
 - ✦ LSF

Functionality

- ✦ Reserve different types of resources
 - ✦ Metacompute (single or multiple clusters)
 - ✦ Network
 - ✦ Workflow
- ✦ Submit jobs on single cluster or multiple clusters
- ✦ Both date and time can be included in the reservations
- ✦ Command line only (HARC is easy to use because of its self-explanatory parameters)

Reliability and Usability

- ✦ When part of the reserved resources becomes unavailable for some reasons, HARC won't cancel the whole reservation, nor notify the user
- ✦ The overall system functions normally if a majority of Acceptors remain in a working state [2]:

$$MTTF_{2F+1} \approx \left(\frac{1}{2F} \right) \left(\frac{MTTF}{2F+1} \right) \left(\frac{MTTF}{MTTR} \right)^F$$

where MTTF is the Mean-Time-To-Failure and MTTR is Mean-Time-To-Repair of an Acceptor F

- ✦ HARC script is easy to understand, commands are self-explanatory

- ✦ Canceling reservations is easy

GUR

GUR is an intuitive analogy of travel industry (Figure 3) [3]:

- ✦ user creates meta job file
- ✦ GUR requests coordinated reservations on clusters
- ✦ local schedulers respond to GUR requests
- ✦ coordinated reservations for nodes are created.

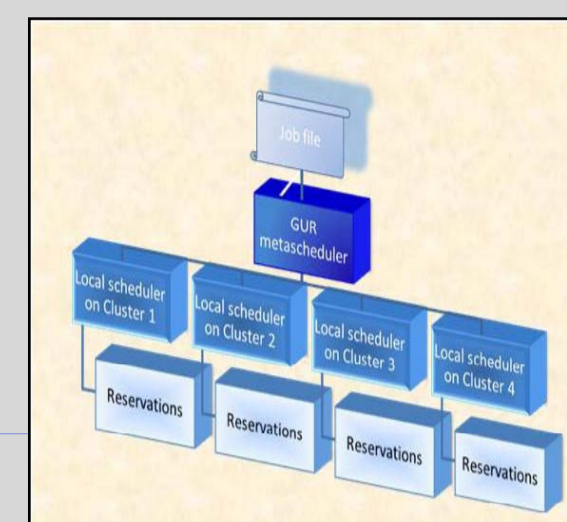


Figure 3

There is no middle layer for GUR.

GUR goes from the users directly to the resources one after another and makes reservations. Users act as travel agent and reserve co-scheduled jobs [4]. If all the local resource managers can meet the request, the reservation is done and GUR will get a reservation ID. Otherwise, the reservation is failed.

Python script

Platform: UNIX

Prerequisite: local scheduler

Components:

- ✦ GUR server package
 - ✦ user commands to set, query and cancel reservations
 - ✦ wrapper script to format input and output to local resource manager commands
- ✦ GUR client package - configure gur.py and gur.config files.

- ✦ Use "myproxy-logon" to get credential
- ✦ Use the ssh and scp commands

✦ Designed for Unix platform

✦ Python Code

✦ Under active development and not in production stage yet

✦ Local resource manager works with:

- ✦ Catalina
- ✦ Moab
- ✦ LSF
- ✦ PBSPro

- ✦ Reserve different types of resources
 - ✦ Metacompute (single or multiple clusters)
 - ✦ Workflow - through GLOBUS2GUR tool [4]
- ✦ Submit jobs on single cluster or multiple clusters with or without reservation id
- ✦ Both date and time can be included in the reservations
- ✦ Command line
- ✦ Web Portal can make reservations only on one cluster

- ✦ When part of the reserved resources becomes unavailable for some reasons, GUR won't cancel the whole reservation, nor notify the user
- ✦ GUR command line and portal are connected
- ✦ GUR script is easy to understand
- ✦ The Web Portal is an advantage, and makes advance reservations very easy
- ✦ Canceling reservations is easy

Conclusions

- ✦ A TeraGrid User survey pointed out that 20% of users are using advance reservations and that 25% are planning to use them in the future. This means that almost half of users will use advance reservations.
- ✦ From the architecture point of view, HARC is more flexible than GUR, because of its service-based design.
- ✦ From the reservation point of view, HARC has the advantage of reserving network resources, which is increasingly important for the current communication intensive grid applications.
- ✦ From end user's point of view, usability might be the most important. GUR's web portal is a good starting point.
- ✦ In the future, we might have to scrutinize the relationship between local schedulers and co-allocation tools to improve efficiency and throughput. With global schedulers, which can schedule jobs on a bunch of clusters, better optimization might be achieved. In some sense, co-allocation tools are transitional.

Summary

Aspects of Comparison	HARC	GUR
Applications	Metacompute Workflow Network	Metacompute Workflow - GLOBUS2GUR
Deployment	TeraGrid, LONI	TeraGrid
Architecture	Paxos Commit Protocol Additional layer acceptors - more flexible	Travel agent analogy No additional layer
Installation	HARC client	Server and client packages
Security	X.509	Myproxy-logon, ssh, scp
Portability	Unix	Unix
Functionality	Command line	Command line Web Portal
Reliability and Usability	MTTF, MTTR, Self-explanatory commands Good Script	Web Portal Easy to understand script

References

1. J. MacLaren, "HARC: The Highly-Available Resource Co-allocator", to appear in *Proceedings of GADA'07*, LNCS 4804 (OTM Conferences 2007, Part II), Springer-Verlag
2. J. MacLaren, "Co-allocation of Compute and Network resources using HARC", in *Proceedings of "Lighting the Blue Touchpaper for UK e-Science: closing conference of ESLEA Project"*. PoS(ESLEA)016, 2007
3. K. Yoshimoto, P. Kovatch, P. Andrews, "Co-scheduling with User-Settable Reservations", 11th Workshop on Job Scheduling Strategies for Parallel Processing, 2005
4. D. Marcusiu, M. Margo, K. Yoshimoto, P. Kovatch, "Automatic Co-Scheduling on the TeraGrid", <http://teragrid.org/library/tg06-gur-mmargo-pk.pdf>

Acknowledgements

We want to thank Dr. Jon MacLaren for all his information, help and support. Also we want to thank to Dr. Kenneth Yoshimoto and Dr. Martin Margo for all the valuable information, and to all the people from TeraGrid and LONI who granted us access to the grid resources.