B-OAR : Extending the scope of Resource Management through Volunteer Computing

Under the guidance of
Prof Olivier Richard
Prof Derrick Kondo
MESCAL, INRIA Rhone-Alpes
{olivier.richard, dkondo}@imag.fr

Project Report
Summer Internship 2008
Submitted By
Nagarjun Kota
Indian Institute of Technology, Kharagpur
nagarjun.kota@gmail.com
Motivation

• What kind of resources participate in grid computing?
• What tasks do the state-of-the-art resource managers do?
• Is it enough?
Classical Grids vs Desktop Grids

• Classical grids such as clusters and multiprocessors are dedicated to the kind of jobs they run; Desktop grids are not

• Classical grids involve a limited number of high-end, dedicated resources; Desktop grids involve a huge number of light-weight and volatile resources

• Desktop Grid computing is much more cost-efficient than classical grid computing in terms of the cost of hardware and setup [4]
The B-OAR integrated grid
Outline

- BOINC vs OAR
- BOINC Single job submission
- OAR job execution mechanism
- The B-OAR interface
- B-OAR Virtual Resource Manager
- Steps to get B-OAR running
- DEMO
- Call for applications

Envisioning ‘The Grid’. Courtesy: GridCafe, CERN
BOINC vs OAR

- BOINC uses an API; OAR doesn’t
- OAR assumes specific execution platform; BOINC doesn’t
- OAR applies admission rules before accepting jobs; BOINC uses mechanisms for validation, security, failure-handling and credit assignment
- OAR assumes and requires dedicated resources; BOINC resources are in fact, volatile
- Check-pointing is more easily implemented with BOINC than with OAR
- BOINC resources cannot communicate with each other; OAR resources can
BOINC Single Job Submission

- BOINC wrapper allows applications that are not based on the BOINC API to run as legacy applications
- Submit such jobs to the BOINC server using “boinc_submit”

Illustration of the BOINC wrapper functionality. Courtesy: BOINC Wiki [4]
OAR job execution mechanism

- User submits job request using “oarsub”
- OAR parses job parameters, applies admission rules
- Job is admitted to OAR DB
- Scheduler allocates resources to job based on its requirements
OAR job execution mechanism

![Diagram of OAR job execution mechanism]

Courtesy: OAR - A batch scheduler with high-level components [1]
The B-OAR interface

- Couple OAR with BOINC by grabbing certain kinds of OAR jobs and running them on BOINC.
- Add “virtual” resources to B-OAR using the B-OAR virtual resource manager.
- If OAR scheduler allocates job to “virtual resources”, grab job parameters, download input files and launch on BOINC using boinc_submit.
- On completion of job, BOINC sends a termination signal to OAR after which results are uploaded.
Flowchart depicting execution of OAR jobs on BOINC resources. Boxes marked * belong to the B-OAR interface.
Features of the B-OAR interface

• Multiple OAR jobs can be made to run on BOINC “virtual” resources

• Maintains a list of jobs run on the BOINC server as the tuple:
  `<OAR_Host_ID, OAR_job_ID, BOINC_Host_ID, BOINC_job_ID>`

• This info is used by the B-OAR Virtual Resource Manager for accounting of B-OAR jobs and resources
B-OAR Virtual Resource Manager

- Automate addition/removal of BOINC virtual resources to OAR DB
- Monitor status of OAR jobs running on BOINC
- Monitor status of BOINC virtual resources, and provide a mapping between them and actual BOINC hosts
- To provide approximate statistics related to the performance of BOINC virtual resources over OAR jobs
B-OAR Virtual Resource Manager

**Modular structure of the B-OAR virtual resource manager**

- **DB updater**
  - B-OAR Virtual Resource Manager
    - B-OAR automated resource adder/remover
  - B-OAR job stats

**B-OAR Automated Resource Adder/Remover**

1. **START**
2. Get the current number of hosts connected
3. If \( \geq \) old number of hosts?
   - Set \(<\text{new hosts} - \text{old hosts}>\) number of nodes to Alive in OAR database
   - Sleep \(<\text{time period}>\)
4. **STOP**
Source code for B-OAR

- Available at the oar svn repository, in the ...
.../branches/boar directory
- Full documentation provided along with the source code
Steps to get B-OAR running

- Set up BOINC server and configure for single job submission
- Set up OAR server
- Download and compile B-OAR source code on both machines
- Modify certain settings in oar.conf
- Connect to the desired OAR server from BOINC using “connection_request” module
- Launch “resource_manager” on BOINC server as a daemon
- Test using “oarsub” from OAR server; use “cosystem” type
DEMO

• B-OAR interface prototype is installed on inaja.imag.fr (accessible from the navajo server via ssh)

• OAR and BOINC servers installed on the same node for simplicity
Possible Uses AND Future Work

• Reinforce OAR by providing it more resources
• A mechanism for load balancing and fault tolerance in the integrated OAR-BOINC grid
• Use B-OAR interface for predicting characteristics like availability of Desktop grid resources
• CALL FOR APPLICATIONS:
  • Scientific, compute-intensive applications consisting of independent tasks
  • Applications for running experiments over the Internet, or that use the B-OAR system as a testbed, like PlanetLab
Acknowledgements

Project supported by EGIDE via Prof Bruno Gaujal and INRIA DRI.

Would like to thank Joseph Emeras, Pierre Neyron, Bruno Bzeznik, Yiannis Georgiou, Christian Seguy for helping me out in the initial phase of the project.

Grateful to Prof Olivier Richard and Prof Derrick Kondo for their guidance and support.

Special thanks to the MESCAL team and the people at LIG, ENSIMAG for providing me a wonderful atmosphere for work and learning.
Related Work

• Lattice Project
• PlanetLab
• Integrating OpenPBS with desktop grid resource managers
References


Thank You

QUESTIONS ??
SUGGESTIONS ?