

BOINC as part of a comprehensive, multi-resource grid computing system

Michael P. Cummings and Adam L. Bazinet
Center for Bioinformatics and Computational Biology
Department of Biology and Institute for Advanced Computer Studies
University of Maryland

models of grid computing

- large, tightly coordinated efforts
 - system-centric, very often “build it, they will come”
 - complex administrative structures
 - policy enforcement mechanisms
 - homogeneous resources
- variable size, loose federations
 - end-user-centric, driven by pragmatism
 - fully autonomous local control
 - no enforcement mechanisms
 - heterogeneous resources

fundamental differences among middleware models

- basic orientation
- user(s)
- projects
- applications
- knowledge about resources
- reliability of resources
- exclusivity of resource use
- push/pull

Globus

- largely oriented toward large clusters and Condor pools
- Grid Security Infrastructure (GSI) provides for strong, distributed authentication of mutually-distrustful parties
- Community Authorization Service (CAS) provides robust authorization capabilities
- Grid Resource Allocation and Management (GRAM) service provides an abstraction layer that allows jobs to be submitted to computational resources without prior knowledge of the underlying job submission and queuing systems
- push model: work is sent from some submitting node to some computational resource, which then accepts and processes the job, returning the results to the submitter

BOINC

- oriented to single user/project, therefore not meant to provide any of the functions which one would expect a normal queuing system to provide
- can automatically match work to be processed with hosts suitable to execute it, taking into account estimated memory and disk requirements as well as architecture and operating system constraints
- includes support for redundant computing
- pull model: BOINC clients (i.e., desktop PCs) contact a server that acts as a central repository of work to retrieve jobs to execute

building a Globus-BOINC scheduler adapter

- selecting the BOINC application you want to run
- transforming a Globus specification of program input files and arguments into BOINC specification (work templates)
- checking job status (currently fulfilled by the BOINC scheduler event generator (SEG) which makes mysql database calls)
- returning the results, which means copying output from where it is placed by BOINC (in some cache hierarchy) to where Globus expects it (the scratch directory)
- transforming file names

benefits of joining Globus and BOINC

- on the Globus side: grid users may gain access to a much wider pool of potential resources than was previously possible
- on the BOINC side: grid users may gain a more full-featured system (e.g., multiple users, multiple applications, authentication, authorization)
- references

Bazinet, A. L., and M. P. Cummings. 2008. The Lattice Project: a Grid research and production environment combining multiple Grid computing models. In Weber, M. H. W. (Ed.) *Distributed & Grid Computing — Science Made Transparent for Everyone. Principles, Applications and Supporting Communities*. Rechenkraft.net, Marburg.

Myers, D. S., A. L. Bazinet and M. P. Cummings. 2008. Expanding the reach of Grid computing: combining Globus- and BOINC-based systems. Pages 71-85. In Talbi, E.-G. and A. Zomaya (Eds.) *Grids for Bioinformatics and Computational Biology*, Wiley Book Series on Parallel and Distributed Computing. John Wiley & Sons, New York.

basic philosophy

- user just want my jobs done so they can do their science
 - finding resources is a barrier
 - applying to use resources is a barrier
 - wait for approval is a barrier
 - learning new system is a barrier
 - grid-enabling applications is a barrier
- heterogeneous computing needs are best met by heterogeneous computing resources
- the computers are already here (and there, and over there), and we just need to figure out how to use them
- makes sense economically, environmentally, socially, practically

heterogeneous needs, heterogeneous resources

- atomization of jobs
 - parameter sweeps
 - stochastic algorithms
 - bootstrap analyses
- pleasingly parallel (only embarrassing on poorly matched resources)
 - often do not need low latency, high-bandwidth interconnects
- sometimes best performance on desktops rather than clusters
 - constantly advancing hardware available
 - advanced GPUs on desktops, rarely on clusters
- makes “big iron” more valuable by using it appropriately

The Lattice Project

- grid development and production system
- motivated solely to satisfy the needs for computing resources
- strives to lower the barriers for both users and developers
- combines multiple resource types
- hardware, operating system, queuing system, and middleware neutral (don't have a dog in the fight)
- ran legacy code using "compatibility library", which overwrote functions using linker tricks, and on Windows used DLL injection, long before BOINC wrapper came on the scene
- perhaps the most comprehensive grid computing system

a few example applications running via BOINC

- HMMProfam: protein sequence analysis (modified source code)
 - periodic updating of database using new data or new models
 - single user
 - closest to typical BOINC project
- MARXAN: design of conservation networks (executable only)
 - single user
 - parameter sweep using empirical and simulated data sets
 - multiple domain-science projects
- GARLI: large-scale phylogenetic analysis (developer involved)
 - multiple users
 - multiple domain-science projects

keeping BOINC users happy

- BOINC users like
 - shorter workunits
 - workunits of known length
 - workunits of consistent length
 - constant supply of workunits
- reality: many workunits have long runtimes, are indeterminate in length, span a range runtimes, and arise irregularly
- consequence: overhead in managing a BOINC project is significantly increased over typical project
- partial fix (planned): decomposing problem into regularized workunits

<http://lattice.umiacs.umd.edu>