Advanced Resource Brokering on ARC

Erik Elmroth & Johan Tordsson
Department of Computing Science and HPC2N
Umeå University
Outline

• Scenario and motivation

• 1st generation
  - Built-in component of the ARC submission interface
  - Estimating the Total Time to Delivery
    • Benchmark-based execution time estimates
    • File transfer time predictions
  - Advance reservations
  - Basic adaptation

• 2nd generation
  - General stand-alone GT4-based broker/job submission service
  - Standards-based with compatibility plugins for specific middlewares
  - Co-allocation
Objective and scenario

- Broker works on behalf of the user - strives to find optimal resources *from the user's point of view*

(Optimization of resource utilization is a task of local resource managers and their support systems)

Current objective:
- stand-alone broker and job submission service
- Grid middleware independent
- scenario typified by HPC users
- follow appropriate standards in order to facilitate component integration
1st generation

- NorduGrid ARC-specific broker
- Built-in component of the ngsub submission interface

Highlights:
- Resource selection based on TTD estimates
  - Benchmark-based execution time estimates
  - Network performance predictions
- Advance reservations
- Basic (batch queue) adaptation mechanism

Overall brokering/job submission algorithm

Process input xRSL spec. Create a list of individual jobs
Identify available clusters via GIIS server(s)
Retrieve static/dynamic resource info from GRISes
For each job:
5. Select cluster:
   a. Filter out non-feasible clusters (user not authorized, cluster not fulfilling requirements on memory, disk space, architecture etc)
   b. Estimate TTD for each remaining resource
      If requested, perform resource reservation
   c. Select the cluster with the shortest predicted TTD.
6. Submit the job to the selected resource.
7. Release any reservations made to non-selected clusters.
8. Perform adaptation
Estimating the Total Time to Delivery

Identify and allocate resources that (from avail. info) gives the minimum estimated Total Time to Delivery (TTD)

• Estimate TTD components, i.e., time for
  - Stage in
  - Queueing
  - Execution
  - Stage out
Stage in/stage out

- Time for input/output/executable transfer to/from storage locations and resources considered (but no storage brokering)

- Estimates currently based on
  - Actual size of input data and executable
  - User-estimated size of output data
  - NWS-prediction of network performance

- NWS: Network Weather Service (UCSD & U. Tenn.)
  - Network performance forecasts (statistical models and historical data)

- Take into account that files may be stored on different resources

- Vital not to underestimate input transfer time

  (outputfiles= (results gsiftp://host3/my_program.results)
   (data gsiftp://host3/my_program.data)
   (logfile gsiftp://host4/my_program.log))

  (outputfilessizes= (results 230MB)
   (data 5GB))
Queueing time & Reservation capability

- Diverse/weak support for queue time estimates from batch systems
- #jobs in queue, #cpus, mean-values for #cpus/job and time/job may give rough estimates (still without taking scheduling algorithm into account!)

- Reservation capability gives
  - Guaranteed waiting time
  - Support for co-allocation

- Current broker prototype supports:
  - Reservations for Maui-based systems
  - Rough estimates for other systems

- Basic steps:
  - Request reservation at LRMS via GridFTP
  - Resource stores mapping user proxy - reservation ID
  - Submit job, including reference to reservation
Estimating the execution time

• Benchmark-based time estimates

• The user specifies
  - Names of benchmarks relevant for the application
  - Benchmark results for system X (X unspecified)
  - Expected execution time on system X

• Broker prototype
  - Estimates execution time component of TTD for each system of interest
  - Sets an appropriate time requirement for job submitted

&(executable = my_app)(stdin = my_app.in)(stdout = my_app.out)
Queue adaptation

• Information is necessarily old - load and availability of resources change rapidly

• Resource selection may prove to be a bad decision, based on old information

• Periodically search for better resources after job submission. Migrate job to another queue if a better cluster is found.
  
    - No support for migration of executing jobs
Component interactions
2nd generation resource broker

• General with respect to underlying Grid middleware

• Based on Globus Toolkit 4
  - the broker/job submission module as a Grid service
  - enables lightweight submission interfaces

• Features
  - Resource selection based on TTD-estimates
    • Benchmark-based execution time estimates
    • File transfer time predictions
  - Advanced reservations
  - Co-allocation

• Adaptation features left outside (moves to monitoring)
“Standards” based

• Component interactions based on proposed “standards”
  - **JSDL**: XML-based job submission language
  - **WS-agreement**: XML language for specifying an agreement between a resource/service provider and a consumer
    + a protocol for creation of an agreement using templates
  - **GLUE**: information service schema used in GT4

• Plugins for
  - invocation from different submission interfaces
  - submitting jobs to resources with different middleware
  (Initially, support for ARC and vanilla GT4 at both ends)
Component interactions
Concluding remarks

• 1st generation prototype: ARC ngsub built-in
  - Resource selection based on computer benchmarks and network performance
  - Advance reservations
  - Queue adaptation

• 2nd generation: general stand-alone broker service
  - Based on GT4
  - Standards-based (JSDL, WS-agreements, GLUE, ...)
  - Plugins for specific middlewares (submission + resource)