Towards Advanced Resource Brokering in UNICORE 6

Bastian Demuth
Jülich Supercomputing Centre
b.demuth@fz-juelich.de

UNICORE Summit 2011, Torun
Overview: UNICORE Workflow System
Goals

• Improve performance and brokering quality
  • Advocate indexing mechanisms
  • Strategies broker multiple incoming jobs at the same time
  • Deal with site failure more gracefully
  • Support job priority
• More flexibility and configurability
  • Re-usability of brokering as a library, e.g. for clients
  • Pre-defined brokering strategies, selectable by job
  • Hot deployment of strategies, Job defines strategy (scripting language)
  • Admin should be able to control this
• Improve reporting when matchmaking fails
• Better abstraction from UNICORE
• Use site specific resources during matchmaking
Zoom In: The Service Orchestrator

Workflow System

Service Orchestrator

Persistence

Resource Brokering

Job Execution

Job Management

WA Management

WA Submission

submit Jobs

consult

collect Attributes

forward

write

Zoom In: Resource Brokering

Service Orchestrator

- Matchmaking
  - forward job + sites
  - delegate job + rating

- Rating
  - forward job + sites
  - delegate job + rating

- Selection

- Matchmaker
  - delegate

- Site Index
  - read
  - write

- Attribute Cache
  - read
  - write

- Attribute Collector
  - read

- Execution Site

- Site Indexer
  - read
  - write

- Matchmaking
  - read
  - delegate job

- Rating Strat.
  - read
  - delegate

- Sel. Strat.
  - read
  - delegate

- Rating Strat.

- Execution Site
Explanation: Resource Brokering

- Removes unsuitable sites
- Delegates to multiple matchmakers
- Matchmakers query site indexes
- One index per job attribute (e.g. OS, #CPUs)
- Indexes built previously => fast removal of invalid sites
- Supports white lists, black lists, grey lists

- Assigns normalized scores to sites
- Delegates to a rating strategy
- Different strategies correspond to different brokering criteria (e.g. data proximity, load balancing, cost-effectiveness)
- Relies on previously collected site attributes
- Ratings can be combined (weighted sums)

- Chooses a site per job by looking at the rating
- Delegates to a selection strategy
- Different flavours, e.g. highest score, stochastic
- Rather simple step

Attribute collectors and matchmakers can be hot-deployed
Rating/selection strategies can be hot-deployed and embedded in the job
Indexing

- Applications, OSes, CPU Arch:
  - Inverted indexes

- JSDL RangeValueTypes – Interval sets:
  - Interval skip lists $\Rightarrow O(\log n + k)$

- Naive approach is actually faster in practice up to very large n
New Web Services

• Resource Broker
  • Just find a target site, don't submit and monitor
• Component deployment (strategies, attribute collection, indexing)
  • Can be controlled by admin
Reusability and Configurability

- Abstract interfaces (e.g. no strict binding to JSDL => 2 modules)
- Things must be handled differently in different clients (e.g. site attributes)
- Dependency injection for all major components
- Spring framework
- Drawback: lot of configuration, hard to find relevant bits
Future work

- Sandbox groovy scripts properly
- Use library in URC, UCC
- Advanced brokering requires additional info about sites
- Strategy predicting job runtime at sites
  - Queueing time (difficult!)
  - Time for stage-ins (feasible, problem: data sparseness)
  - Runtime of the executable (from JSDL)
  - Time for stage-outs (similar to stage-ins)
- Strategy predicting energy consumption (Fit4Green)
  - Requires runtime prediction
- Systematic comparison of strategies
  - Larger test Grid
  - Simulation