GridBot: Execution of Bags of Tasks in Multiple Grids

Jan Stienstra

March 11, 2012
Bags of Tasks

Figure: Bags of Tasks
Outline

- Challenges in BOT scheduling
- Grid Overlay Constructor
- Work-Dispatch Logic
- Results
- Conclusion
Challenges

- Execution of BOTs on multiple grids
- Different grid characteristics
  - Resource availability
  - Failure rates
  - Queuing times
Resource Availability

Figure: Resource availability on different grids
## Failure Rates

<table>
<thead>
<tr>
<th>Grid</th>
<th>#Jobs</th>
<th>Preempted (%)</th>
<th>Failed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UW Madison</td>
<td>96938</td>
<td>20</td>
<td>1%</td>
</tr>
<tr>
<td>OSG</td>
<td>60648</td>
<td>10</td>
<td>1%</td>
</tr>
<tr>
<td>EGEE</td>
<td>16437</td>
<td>7</td>
<td>2%</td>
</tr>
<tr>
<td>Technion</td>
<td>42411</td>
<td>2</td>
<td>0.2%</td>
</tr>
<tr>
<td>Superlink @Technion</td>
<td>241364</td>
<td>(♣)0.2</td>
<td>13%</td>
</tr>
</tbody>
</table>

**Figure:** Reliability on different grids
Queuing Times

Figure: Queuing times on different grids
The General Idea

GridBot is integrated into BOINC to:

- Unify different grids
- Support job execution policies
- Support Multi-BOT prioritization policies
Architecture

Figure: The GridBot Architecture
We differentiate two types of grids:

- **Collaborative Grids**
  - Local queuing system
  - Resources hidden

- **Community Grids**
  - Independent execution clients
  - Client initiated job execution
Grid Overlay Constructor

- Constructs overlay of execution clients
- Monitors resource demand
- Determines set of suitable grids
- Requests resources from collaborative grids
- Submits execution clients to grids
A Unified Grid

- Community grid clients can directly contact the communication frontend
- Collaborative grid / dedicated cluster hosts receive an execution client from GOC
- Execution client are identical to community grid clients
Work-Dispatch Sequence

1. Execution client requests job
2. GridBot generates candidate jobs
3. GridBot assigns a number of jobs to the client
Scheduling pseudocode

Instantiate classad for $h$, BOT and queue

Foreach job $j$ in the job queue

Instantiate classad for $j$

Evaluate $\text{Concurrency}_j$, $\text{Deadline}_j$, $\text{JobRequirements}_j$ and $\text{Rank}_j$

If $\text{JobRequirements}_j = \text{true}$

Add $j$ to candidate set $J$

End

Order the jobs in $J$ by $\text{Rank}_j$

$\text{Concurrent}\leftarrow\text{Concurrent}$ of a job with maximum $\text{Rank}$

Foreach job $j \in J$

$\text{Concurrent} \leftarrow \min(\text{Concurrent}, \text{Concurrent}_j)$

If $\text{Concurrent} < \text{assigned} + \# \text{running jobs on } h$

$\text{Deadline}_j \leftarrow \text{Deadline}_j$

Assign job $j$ to host $h$

assigned $\leftarrow$ assigned + 1

End

End

Figure: Scheduling; upon job request from host $h$
Replication is performed in two situations:

- When a deadline has expired
  - A job is deemed failed
  - The job has to be restarted (and is thus replicated)

- To reduce turnaround time
  - Failures are common
  - Replication increases the chance of correct execution
Comparison with ‘naive’ BOINC

Figure: Execution of a BOT on multiple grids
Effectiveness of Replication Policy

![Figure: Turnaround time for medium sized BOTs](image)

R1: Unlimited replication, all hosts
R2: Unlimited replication, replicas in grids
R3: Constrained replication, replicas in grids
R4: Constrained replication, 'Tail' not used

**Figure:** Turnaround time for medium sized BOTs
Scalability

Figure: Turnaround time for different number of enqueued BOTs
GridBot provides:

- Grid unification
- Policy driven work-dispatch logic
- A scalable architecture for multi-BOT execution
Remember!

GridBot is an efficient, flexible and scalable architecture for execution BOTs on multiple grids