Deadline-based Grid Resource Selection for Urgent Computing

Nick Trebon

6/18/08
University of Chicago
**MS Committee**

- Dr. Ian Foster
- Dr. Pete Beckman (ANL)
- Dr. Rick Stevens

- Special Thanks to Dr. Rich Wolski and Dan Nurmi (UCSB)
Outline

- Introduction
  - Introduction to SPRUCE
  - Problem Statement
  - Literature Review
- Methodology
  - Bounding Total Turnaround Time
  - Other Considerations
- Case Study (6 Experiments)
  - Results and Analysis
- Conclusions and Contributions
- Future Work
Introduction: SPRUCE

- Urgent Computing: SPRUCE
  - Urgent computations
    - Defined by a strict deadline -- late results are useless
    - Must be in “warm standby” -- ready to run when and where needed
  - Provide token–based, priority access for urgent computing jobs
    - Elevated batch queue priority
    - Resource reservations (HARC – prototype implemented)
    - Elevated network bandwidth priority (NLR – prototype implemented)
  - Resources respond to urgent requests based on local policy
Problem Statement

- Given an **urgent computation** and a **deadline**, how does one select the “best” resource?
  - “Best”: Resource that provides the *configuration* most likely to meet the deadline
    - Configuration: a specification of the runtime parameters for an urgent computation on a given resource
      - Runtime parameters: # cpus, input/output repository, **priority**, etc.
  - Cost function (priority):
    - Normal --> No additional cost
    - SPRUCE --> Token + intrusiveness to other users
Literature Review

• Grid Resource Selection
Literature Review, cont.

• **Queue Delay**

• **File Transfer Delay**

• **Performance Modeling**
Outline

- Introduction
  - Introduction to SPRUCE
  - Problem Statement
  - Literature Review
- Methodology
  - Bounding Total Turnaround Time
  - Other Considerations
- Case Study (6 Experiments)
  - Results and Analysis
- Conclusions and Contributions
- Future Work
Probabilistic Bounds on Total Turnaround Time

- Input Phase: \((I_Q, I_B)\)
- Resource Allocation Phase: \((R_Q, R_B)\)
- Execution Phase: \((E_Q, E_B)\)
- Output Phase: \((O_Q, O_B)\)

- If each phase is independent, then:
  - Overall bound = \(I_B + R_B + E_B + O_B\)
  - Overall quantile \(\geq I_Q \times R_Q \times E_Q \times O_Q\)
$I_B + R_B + E_B + O_B$: File Staging Delay

- Methodology is the same for input/output file staging
- Utilize Network Weather Service to generate bandwidth predictions based upon probe
  - Wolski, et al.: Use MSE as sample variance of a normal distribution to generate upper confidence interval
- Problems?
  - Predicting bandwidth for output file transfers
  - NWS uses TCP-based probes, transfers utilize GridFTP
**Deadline-Based Grid Resource Selection**

$I_B + R_B + E_B + O_B$: Resource Allocation Delay

- Normal priority (no SPRUCE)
  - Utilize the existing Queue Bounds Estimation from Time Series (QBETS)

- SPRUCE policies
  - Next-to-run
    - Modified QBETS – Monte Carlo simulation on previously observed job history
  - Pre-emption
    - Utilize Binomial Method Batch Predictor (BMPB) to predict bounds based upon preemption history
  - Other possibilities: elevated priority, checkpoint/restart
Resource Allocation Concerns

- Resource allocation bounds are predicted entirely on historical data and not current queue state
  - What if necessary nodes are immediately available?
  - What if it is clear that the bounds may be exceeded?
    - Example: Higher priority jobs already queued

- Solution: Query Monitoring and Discovery Services framework (Globus) to return current queue state
$I_B + R_B + E_B + O_B$: Execution Delay

- Approach: Generate empirical bound for a given urgent application on each warm standby resource
  - Utilize BMBP methodology to generate probabilistic bounds
    - Methodology is general and non-parametric
Improving the Probability

- Composite quantile quickly decreases
  - \(0.95^4 = 0.815\)
  - \(0.75^4 = 0.316\)

- Overlapping phases
  - E.g., stage input files while job is queued

- Speculative execution
  - Initiate two *independent* configurations
  - \(\Pr(C_1 \text{ or } C_2) = \Pr(C_1) + \Pr(C_2)(1 - \Pr(C_1))\)
Overlapping Phases Example

- **Input file staging**
  - Pr(files staged < 10 minutes) = 0.90
  - Pr(files staged < 20 minutes) = 0.95
  - Pr(files staged < 30 minutes) = 0.99

- **Queue delay**
  - Pr(job begins in < 10 minutes) = 0.75
  - Pr(job begins in < 20 minutes) = 0.85
  - Pr(job begins in < 30 minutes) = 0.95

- **Serial:** Pr(job begins in < 30 minutes) ≤ 0.90 * 0.85 = 0.765

- **Overlap:** Pr(job begins < 30 minutes) ≤ 0.95 * 0.99 = 0.94
Calculating the Probability

- Goal: to determine a probabilistic upper bound for a given configuration

- Approximate approach: query a small subset (e.g., 5) of quantiles for each individual phase and select the combination that results in highest composite quantile with bound < deadline
Outline

- Introduction
  - Introduction to SPRUCE
  - Problem Statement
  - Literature Review
- Methodology
  - Bounding Total Turnaround Time
  - Other Considerations
- Case Study (6 Experiments)
  - Results and Analysis
- Conclusions and Contributions
- Future Work
Case Study

- Application: flash2
- File staging requirements:
  - Modeled after LEAD workflow
  - Input file: 3.9 GB
  - Input repository: Indiana and UCAR
  - Output file: 3.9 GB
  - Output repository: Indiana
# Case Study Resources

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU</th>
<th>Total Nodes</th>
<th>Total CPUs</th>
<th>Req. Nodes</th>
<th>Req. PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>Itanium 2 (1.3/1.5 GHz)</td>
<td>62</td>
<td>124</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Mercury</td>
<td>Itanium 2 (1.5 GHz)</td>
<td>631</td>
<td>1262</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Tungsten</td>
<td>Intel Xeon (3.5 GHz)</td>
<td>1280</td>
<td>2560</td>
<td>32</td>
<td>64</td>
</tr>
</tbody>
</table>

- UC/ANL supports normal, next-to-run and preemption priorities
- Mercury and Tungsten support normal priority
NWS Probes v. GridFTP Transfers

NWS Probes for Indiana to Mercury
(Probe size: 2048 KB)

GridFTP transfers for Indiana to Mercury
(File size: 171 MB)
GridFTP Probe Framework

- Purpose: to generate probe measurements whose behavior matches the behavior seen in large GridFTP transfers
- Probes:
  - Tuned for each source, destination pair
  - Sent every 15 minutes
  - Initiated as third-party transfers
  - Measurements stored directly into NWS
- Results:
  - More similarity between probe and transfer behavior
  - Correctness and accuracy of predicted bounds discussed later
Experiment 1

• Purpose: Validate methodology for bounding GridFTP transfers
• Methodology:
  ✷ For each source, destination pair generate a bound and then initiate a transfer
  ✷ Complete at least 100 trials and evaluate success of bounds
### Experiment 1 Results

#### Table 1: Success rate for predicted bounds on GridFTP transfers

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th># Trials</th>
<th>% &gt; bound (b/w)</th>
<th>Avg % error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indiana</td>
<td>UC/ANL</td>
<td>99</td>
<td>100</td>
<td>8.1</td>
</tr>
<tr>
<td>Indiana</td>
<td>Mercury</td>
<td>100</td>
<td>95</td>
<td>32.4</td>
</tr>
<tr>
<td>Indiana</td>
<td>Tungsten</td>
<td>243</td>
<td>99.6</td>
<td>24.9</td>
</tr>
<tr>
<td>UCAR</td>
<td>UC/ANL</td>
<td>112</td>
<td>100</td>
<td>8.5</td>
</tr>
<tr>
<td>UCAR</td>
<td>Mercury</td>
<td>113</td>
<td>91</td>
<td>4.9</td>
</tr>
<tr>
<td>UCAR</td>
<td>Tungsten</td>
<td>364</td>
<td>96</td>
<td>25.9</td>
</tr>
<tr>
<td>UC/ANL</td>
<td>Indiana</td>
<td>115</td>
<td>94</td>
<td>11.9</td>
</tr>
<tr>
<td>Mercury</td>
<td>Indiana</td>
<td>100</td>
<td>98</td>
<td>12.5</td>
</tr>
</tbody>
</table>
Experiment 1 Conclusions

• The GridFTP probes combined with NWS forecasting allow for both correct and accurate bounds

• The trials originating from UCAR are omitted from latter experiments
Experiment 2

• Purpose:
  1. Validate composite bounding methodology for normal priorities
  2. Evaluate performance of BMBP methodology for bounding execution delay

• Methodology:
  • Generate individual phase bounds followed by the execution of the actual configuration
  • For each of the three resources utilizing the Indiana input and normal priority, complete approximately 100 trials
  • Compare predictions with actual performance
Experiment 2 Results

**Graphs:**
- **UC/ANL-IA64:**
  - Workflow Delay (s) vs. Run #
  - Delays (red) and Bounds (green)

- **Mercury:**
  - Workflow Delay (s) vs. Run #
  - Delays (red) and Bounds (green)

- **Tungsten:**
  - Workflow Delay (s) vs. Run #
  - Delays (red) and Bounds (green)
## Table 2: Success rate for individual and composite phases

<table>
<thead>
<tr>
<th>Resource</th>
<th># Trials</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>99</td>
<td>100%</td>
<td>97%</td>
<td>100%</td>
<td>83%</td>
<td>97%</td>
</tr>
<tr>
<td>Mercury</td>
<td>100</td>
<td>99%</td>
<td>99%</td>
<td>93%</td>
<td>86%</td>
<td>100%</td>
</tr>
<tr>
<td>Tungsten</td>
<td>100</td>
<td>88%</td>
<td>99%</td>
<td>96%</td>
<td>77%</td>
<td>99%</td>
</tr>
</tbody>
</table>

## Table 3: Percent overprediction for each phase

<table>
<thead>
<tr>
<th>Resource</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>8%</td>
<td>2,379%</td>
<td>7%</td>
<td>22%</td>
<td>65%</td>
</tr>
<tr>
<td>Mercury</td>
<td>10%</td>
<td>4,468%</td>
<td>4%</td>
<td>5%</td>
<td>1,417%</td>
</tr>
<tr>
<td>Tungsten</td>
<td>11%</td>
<td>4,811%</td>
<td>13%</td>
<td>-5%</td>
<td>1,842%</td>
</tr>
</tbody>
</table>
Experiment 2 Conclusions

- Utilizing BMBP methodology for execution bounds performs well
- Composite bounds are correct, but not accurate

Table 3: Average delay and bounds

<table>
<thead>
<tr>
<th>Resource</th>
<th>Avg. Delay</th>
<th>Avg. Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>10,833</td>
<td>17,869</td>
</tr>
<tr>
<td>Mercury</td>
<td>8,696</td>
<td>131,951</td>
</tr>
<tr>
<td>Tungsten</td>
<td>7,631</td>
<td>148,233</td>
</tr>
</tbody>
</table>
Experiment 3

- Purpose: Validate composite bounding methodology for **next-to-run** priority
- Methodology:
  - Generate individual bounds followed by the execution of the actual configuration
  - Conduct approximately 100 trials utilizing the UC/ANL resource, Indiana input and next-to-run priority
  - Emulate execution delay
  - Compare predictions with actual performance
Experiment 3 Results

Predicted bounds and actual workflow delays for UC/ANL-IA64 with Next-To-Run priority

Workflow Delay (s)

Run #
Experiment 3 Results, cont.

Table 4: Success rate of individual and composite phases

<table>
<thead>
<tr>
<th>Resource</th>
<th># Trials</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>98</td>
<td>100%</td>
<td>92%</td>
<td>91%</td>
<td>92%</td>
<td>94%</td>
</tr>
</tbody>
</table>

Table 5: Percent overprediction for each phase

<table>
<thead>
<tr>
<th>Resource</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>8%</td>
<td>46%</td>
<td>5%</td>
<td>41%</td>
<td>11%</td>
</tr>
</tbody>
</table>
Experiment 3 Conclusions

- Next-to-run bounds predictor performs well
- Composite bounds are much tighter than in the normal priority cases
  - Average delay: 11,562 seconds
  - Average bound: 12,966 seconds
Experiment 4

- **Purpose:** Validate composite bounding methodology for *preemption* priority
- **Methodology:**
  - Generate individual phase bounds followed by execution of actual configuration
  - Conduct approximately 100 trials utilizing UC/ANL resource, Indiana input and preemption policy
  - Emulate execution delay
  - Preempt “dummy” job
  - Compare predictions with actual performance
Experiment 4 Results

Predicted bounds and actual workflow delays for UC/ANL-IA64 with Preempt priority

Workflow Delay (s)

Run #
Experiment 4 Results, cont.

Table 6: Success rate for individual and composite phases

<table>
<thead>
<tr>
<th>Resource</th>
<th># Trials</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>109</td>
<td>98%</td>
<td>94%</td>
<td>93%</td>
<td>85%</td>
<td>95%</td>
</tr>
</tbody>
</table>

Table 7: Percent overprediction for each phase

<table>
<thead>
<tr>
<th>Resource</th>
<th>Input</th>
<th>Queue</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC/ANL-IA64</td>
<td>6%</td>
<td>75%</td>
<td>4%</td>
<td>7%</td>
<td>6%</td>
</tr>
</tbody>
</table>
Experiment 4 Conclusions

- Bounding methodology for preemption priority is both accurate and correct.
- Average overall bounds are tighter than in next-to-run experiment.
  - Average delay: 10,350 seconds
  - Average bound: 11,019 seconds
Experiment 5

• Purpose: Determine significance of savings from overlapping phases
• Methodology:
  ✷ Generate individual phase bounds followed by execution of configuration
  ✷ Conduct approximately 100 trials utilizing Mercury resource, Indiana input and normal priority
  ✷ Emulate execution
  ✷ 0.95 quantile for queue, execution and output phases
  ✷ 0.9995 quantile for input
  ✷ Compare predictions with actual performance
Experiment 5 Results

Predicted bounds and actual workflow delays for Mercury resource with normal priority and overlapping input and queue phases.
Experiment 5 Results, cont.

Table 8: Success rate for individual and composite phases

<table>
<thead>
<tr>
<th>Resource</th>
<th># Trials</th>
<th>Input</th>
<th>Queue</th>
<th>I &amp; Q</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>102</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>96%</td>
<td>92%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 9: Percent overprediction for each phase

<table>
<thead>
<tr>
<th>Resource</th>
<th>Input</th>
<th>Queue</th>
<th>I &amp; Q</th>
<th>Exec.</th>
<th>Output</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>19%</td>
<td>15,117%</td>
<td>6,480%</td>
<td>3%</td>
<td>4%</td>
<td>1,863%</td>
</tr>
</tbody>
</table>
Experiment 5 Conclusions

- The quantile for the overlapped input and queue phases decreased by more than 50%
- The composite overprediction of composite bound still driven by overprediction of normal priority queue bounds
  - Future work: Demonstrate savings with elevated priority configurations
Experiment 6

• Purpose: Demonstrate the Advisor algorithm which determines the approximately optimal probability that a configuration will meet a given deadline

• Methodology:
  ✷ Query the following quantiles for each phase:
    ▪ Input: [0.9995, 0.999, 0.9975, 0.995, 0.99, 0.975, 0.95, 0.90]
    ▪ Queue: [0.95, 0.75, 0.50]
    ▪ Execution: [0.99, 0.90, 0.80, 0.70, 0.60, 0.50]
    ▪ Output: [0.9995, 0.999, 0.9975, 0.995, 0.99, 0.975, 0.95, 0.90]
  ✷ Deadline: 6 hours
Experiment 6 Results
Experiment 6 Results, cont.
Experiment 6 Results, cont.

Highest quantile that meets 6 hour deadline for various configurations

- mercury
- uc-normal
- uc-ntr

Quantile

Query #
Experiment 6 Conclusions

- In the normal priority conditions, lowering the targeted quantile for the queue bound significantly decreases composite bound
- Elevated priorities offer lower and less variable bounds
- User may have option of selecting a less invasive priority that provides a similar probability of meeting deadline
Outline

• Introduction
  ✷ Introduction to SPRUCE
  ✷ Problem Statement
  ✷ Literature Review
• Methodology
  ✷ Bounding Total Turnaround Time
  ✷ Other Considerations
• Case Study (6 Experiments)
  ✷ Results and Analysis
• Conclusions and Contributions
• Future Work
Conclusions

- The queue bounds predictions for normal priority jobs are correct, but not very accurate
  - Overprediction of normal priority drives overprediction of composite bound
- The composite bounding methodology was correct in all experiments, and accurate in the SPRUCE experiments
  - Individual bounding methodologies were correct and accurate, for the most part
Contributions

• First research to utilize next-to-run bounds predictor in a resource selection context
• First research to apply the BMBP methodology to bound preemption delay and execution delay
• First research to generate composite turnaround time bounds comprised of individual phase bounds generated by distinct methodologies
Outline

• Introduction
  ✷ Introduction to SPRUCE
  ✷ Problem Statement
  ✷ Literature Review

• Methodology
  ✷ Bounding Total Turnaround Time
  ✷ Other Considerations
  • Case Study (6 Experiments)
  ✷ Results and Analysis

• Conclusions and Contributions

• Future Work
Future Work

- Port next-to-run predictors to other resources
- Conduct overlapping phases experiments on resource with SPRUCE
- Automate and standardize warm standby mechanism
- Incorporate resource reservations
- Network bandwidth reservations
- Adapt methodology to Grid workflows
Extra Slides
NWS Probes v. GridFTP Transfers cont.

NWS Probes for Indiana to Tungsten
(Probe size: 2048 KB)

Delay (s)

Run #

0 25 50 75 100 125 150 175

GridFTP transfers for Indiana to Tungsten
(File size: 2048 KB)

Delay (s)

Run #

0 25 50 75 100 125 150 175