Improving Karnak’s Wait Time Predictions

Jungha Woo
Research Computing at Purdue University
Shava Smallen
San Diego Supercomputer Center
J.P. Navarro
Argonne National Laboratory
Overview

• Functionality
• Predictors of the wait times
• Prediction algorithm – decision tree
• Analysis on prediction accuracy
• Improvement ideas
• Current status and future work
Karnak Prediction Service

• Functionality
  • Predict start time of hypothetical jobs
  • Predict start time of queued jobs
  • Provide information about current and recent jobs

• Interfaces
  • REST (XML and plain text), HTML
    • http://karnak.xsede.org
  • Command line programs
  • Java client library

• Available in XSEDE user portal

Slide modified from Warren Smith, with permission
Predicting Waiting Jobs

- A little information is presented about each job
- Jobs are presented in queue order
  - May not be schedule order
- This is accessible to anyone
  - Does not display
    - user names
    - project names

Slide modified from Warren Smith, with permission
Wait Time Prediction

- Two components in the prediction
  - Predicted wait time
  - A confidence interval
    - Provides information about the expected accuracy of the prediction

---

**Job 3165260 on comet.sdsc.xsede.org**

<table>
<thead>
<tr>
<th>Submit Time (CDT)</th>
<th>Processors</th>
<th>Requested Wall Time (hours:minutes:seconds)</th>
<th>Predicted Wait Time (hours:minutes:seconds)</th>
<th>90% Confidence (hours:minutes:seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/22 16:57:35</td>
<td>6</td>
<td>48:00:00</td>
<td>00:36:43</td>
<td>±01:21:15</td>
</tr>
</tbody>
</table>

*Slide modified from Warren Smith, with permission*
Potential Job

- Describe a potential job
  - Systems & queues
  - Processing cores
  - Wall time
  - Confidence level
Prediction for a Potential Job

- Prediction provided for each system/queue
- N/A if the job can’t be submitted to a queue

**Wait Time Predictions**

Prediction for a job that will use 1 processing cores for 1 hours and 0 minutes.

<table>
<thead>
<tr>
<th>System</th>
<th>Queue</th>
<th>Predicted Wait Time (hours:minutes:seconds)</th>
<th>90% confidence (hours:minutes:seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>darter.nics.xsede.org</td>
<td>batch</td>
<td>93:57:27</td>
<td>±43:31:49</td>
</tr>
<tr>
<td>gordon.sdsc.xsede.org</td>
<td>default</td>
<td>41:16:27</td>
<td>±02:57:43</td>
</tr>
<tr>
<td>gordon.sdsc.xsede.org</td>
<td>normal</td>
<td>41:16:27</td>
<td>±02:57:43</td>
</tr>
<tr>
<td>stampede.tacc.xsede.org</td>
<td>large</td>
<td>12:41:34</td>
<td>±04:56:08</td>
</tr>
<tr>
<td>stampede.tacc.xsede.org</td>
<td>normal</td>
<td>30:27:22</td>
<td>±09:50:54</td>
</tr>
</tbody>
</table>

Actual wait time percentile for comet.sdsc.xsede.org for 8 weeks (As of 06/15/2016)

<table>
<thead>
<tr>
<th>Wait time(seconds)</th>
<th>Percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>0</td>
<td>25%</td>
</tr>
<tr>
<td>3</td>
<td>50%</td>
</tr>
<tr>
<td>11</td>
<td>75%</td>
</tr>
<tr>
<td>782645</td>
<td>100%</td>
</tr>
</tbody>
</table>

(=9.05 days)}

---

*Slide modified from Warren Smith, with permission*
Alternative Prediction Formats

• Until now, the predictions are all numeric values
• Users may want simpler answers
  • Color: Blue, Yellow, Red,…
  • Speed: Fastest, medium, slow
  • Grouping: returns grouped results with group numbers such as 0,1,2,3,4,…
  • Raw: Original format
• Good for fair utilization of systems
### Group and LabeledGroup

#### Wait Time/Start time Predictions

<table>
<thead>
<tr>
<th>Group</th>
<th>Queue@System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><a href="mailto:systest@lonestar4.tacc.xsede.org">systest@lonestar4.tacc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:normal-2mic@stampede.tacc.xsede.org">normal-2mic@stampede.tacc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:serial@lonestar4.tacc.xsede.org">serial@lonestar4.tacc.xsede.org</a></td>
</tr>
<tr>
<td>2</td>
<td><a href="mailto:default@gordon.sdsc.xsede.org">default@gordon.sdsc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:normal@stampede.tacc.xsede.org">normal@stampede.tacc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:debug@gordon.sdsc.xsede.org">debug@gordon.sdsc.xsede.org</a></td>
</tr>
<tr>
<td>3</td>
<td><a href="mailto:compute@comet.sdsc.xsede.org">compute@comet.sdsc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:gpu@comet.sdsc.xsede.org">gpu@comet.sdsc.xsede.org</a></td>
</tr>
</tbody>
</table>

#### Wait Time/Start time Predictions

<table>
<thead>
<tr>
<th>Group</th>
<th>Queue@System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td><a href="mailto:systest@lonestar4.tacc.xsede.org">systest@lonestar4.tacc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:serial@lonestar4.tacc.xsede.org">serial@lonestar4.tacc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:normal-2mic@stampede.tacc.xsede.org">normal-2mic@stampede.tacc.xsede.org</a></td>
</tr>
<tr>
<td>medium</td>
<td><a href="mailto:default@gordon.sdsc.xsede.org">default@gordon.sdsc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:debug@gordon.sdsc.xsede.org">debug@gordon.sdsc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:normal@stampede.tacc.xsede.org">normal@stampede.tacc.xsede.org</a></td>
</tr>
<tr>
<td>slow</td>
<td><a href="mailto:gpu@comet.sdsc.xsede.org">gpu@comet.sdsc.xsede.org</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:compute@comet.sdsc.xsede.org">compute@comet.sdsc.xsede.org</a></td>
</tr>
</tbody>
</table>
Job and Queue Updates from XSEDE Information Service (http://info1.dyn.xsede.org/)
Factors That May Affect Wait Times

- How busy are the target queues?
- How busy are the target systems?
- How demanding is your job?
  - How many CPUs did it ask for?
  - How long was the requested wall time?

- Assumptions
  - Karnak does not know systems’ scheduling policy
  - A node can belong to multiple queues
  - A hypothetical job’s wait time does not have user name
Wait Time Predictors

• Two key concepts
  • countAhead: number of pending jobs ahead of me
  • workAhead: remaining work of pending jobs ahead of me

• Compute them for various group of jobs
  • Pending jobs: countAhead, workAhead
  • Running jobs: countRunning, workRunning
  • Same user jobs
  • Same queue jobs, other queue jobs
  • Same allocation jobs
  • Smaller work jobs
# workAhead Computation

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Submit time</th>
<th>State</th>
<th># CPU</th>
<th>Requested time</th>
<th>Start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>6/11 3PM</td>
<td>Running</td>
<td>2</td>
<td>6 hrs</td>
<td>6/19 2 am</td>
</tr>
<tr>
<td>15</td>
<td>6/16 1PM</td>
<td>Running</td>
<td>3</td>
<td>2 hrs</td>
<td>6/19 3 am</td>
</tr>
<tr>
<td>16</td>
<td>6/18 4PM</td>
<td>Pending</td>
<td>5</td>
<td>1 hr</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>6/19 3AM</td>
<td>Pending</td>
<td>1</td>
<td>2 hrs</td>
<td></td>
</tr>
</tbody>
</table>

If a new job were submitted at 6/19 4am, its workAhead would be:

\[
\text{workAhead} = \text{Sum of CPU hours} = 1 \text{ CPU } \times 2 \text{ Hrs} + 5 \text{ CPU } \times 1 \text{ Hr} = 7 \text{ CPU Hrs}
\]
## Features For Predicting Wait Times

- All features are computed from job and queue updates
  - Input features
    - 26 features for queued job prediction
    - 16 features for unsubmitted job prediction
  - Output feature : waitTime

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queue</td>
<td>Name of queue job submitted to</td>
</tr>
<tr>
<td>processors</td>
<td>Number of CPUs requested by job</td>
</tr>
<tr>
<td>requestedWallTime</td>
<td>User-specified maximum job execution time</td>
</tr>
<tr>
<td>countAhead</td>
<td>Number of jobs pending ahead of this job</td>
</tr>
<tr>
<td>workAhead</td>
<td>Amount of work requested by countAhead jobs</td>
</tr>
<tr>
<td>countRunning</td>
<td>Number of jobs running ahead of this job</td>
</tr>
<tr>
<td>processorsRunning</td>
<td>Sum of requested processors of currently running jobs</td>
</tr>
<tr>
<td>workRunning</td>
<td>Number of remaining hours of running jobs</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>countAheadQueue</td>
<td>countAhead for the jobs submitted to same queue</td>
</tr>
<tr>
<td>workAheadQueue</td>
<td>workAhead for the jobs submitted to same queue</td>
</tr>
<tr>
<td>countAheadOtherQueues</td>
<td>countAhead for the jobs submitted to other queues</td>
</tr>
<tr>
<td>workAheadOtherQueues</td>
<td>workAhead for the jobs submitted to other queues</td>
</tr>
<tr>
<td>countAheadLessEqualProcs</td>
<td>countAhead for the jobs asked for less or equal processor than this job</td>
</tr>
<tr>
<td>workAheadEqualProcs</td>
<td>workAhead for the jobs asked for less or equal processor than this job</td>
</tr>
<tr>
<td>countAheadLessEqualWork</td>
<td>countAhead for the jobs whose remaining work is less than equal to this job</td>
</tr>
<tr>
<td>workAheadLessEqualWork</td>
<td>workAhead for the jobs whose remaining work is less than equal to this job</td>
</tr>
</tbody>
</table>
# Features For Predicting Wait Times

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>countAheadUser</td>
<td>countAhead for the jobs submitted by same user</td>
</tr>
<tr>
<td>workAheadUser</td>
<td>workAhead for the jobs submitted by same user</td>
</tr>
<tr>
<td>countRunningUser</td>
<td>countRunning for the jobs submitted by same user</td>
</tr>
<tr>
<td>processorsRunningUser</td>
<td>processorsRunning for the jobs submitted by same user</td>
</tr>
<tr>
<td>workRunningUser</td>
<td>workRunning for the jobs submitted by same user</td>
</tr>
<tr>
<td>countAheadProject</td>
<td>countAhead for the jobs having same project number</td>
</tr>
<tr>
<td>workAheadProject</td>
<td>workAhead for the jobs having same project number</td>
</tr>
<tr>
<td>countRunningProject</td>
<td>countRunning for the jobs having same project number</td>
</tr>
<tr>
<td>processorsRunningPro-</td>
<td>processorsRunning for the jobs having same project number</td>
</tr>
<tr>
<td>ject</td>
<td></td>
</tr>
<tr>
<td>workRunningProject</td>
<td>workRunning for the jobs having same project number</td>
</tr>
</tbody>
</table>

Features in this box can only be computed for queued jobs

User-specific

Project specific
Karnak Uses Decision Trees

• Decision tree
  • Decision tree builds regression or classification models in the form of a tree structure
  • Breaks down a dataset into smaller and smaller subsets having similar wait times
  • At each depth, choose the best feature that reduces the standard deviation
  • Final result is a tree with decision nodes and leaf nodes

• How to use
  • Construct a tree using training data (= past experiences) per system
  • Traverse the tree using a query, leaf node contains the prediction
Decision Tree

• Advantages
  • Simple to understand, and easy to visualize
  • Little data preparation needed
  • The cost of using tree is logarithmic in the number of data points
  • Handle both categorical and numerical data

• Disadvantages
  • Overfitting ( = do not generalize data well)
  • Decision tree can be unstable with small variations in the data
  • Learning optimal decision tree is NP-Complete
A decision tree for comet.sdsc.xsede.org with 4 hours of window size jobs on 06/07/2016

A job satisfying
- processorRunning > 35896
- workAhead < 545313600
- workAheadOtherQueues < 772156800
is expected to start in 2423 seconds
Empirical Wait Time Distributions for Various Windows
Fitting Wait Times Into Sum Of Exponentials

Fitting observed wait times to exponential distribution fitting

Stampede.tacc.xsede.org

Simulated wait times effect
Bagging

- Make a prediction by averaging predictions of multiple decision tree
- Karnak builds five decision trees per system
  - Different lookback window size for experiences
    - 4 hours
    - 1 day
    - 1 week
    - 4 weeks
    - 8 weeks
  - Their experiences’ distributions are not independent
Why Are Predictions So Big?

- Huge wait times
- Decision trees built upon larger dataset tend to output bigger predictions
- Creating multiple experiences per observation inflates the predictions of other leaves
Larger # of Experiences Lead to Larger Predictions

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>processors</td>
<td>777</td>
</tr>
<tr>
<td>requestedWallTime</td>
<td>240</td>
</tr>
<tr>
<td>countAhead</td>
<td>257</td>
</tr>
<tr>
<td>workAhead</td>
<td>1193362560</td>
</tr>
<tr>
<td>countAheadQueue</td>
<td>256</td>
</tr>
<tr>
<td>workAheadQueue</td>
<td>1193333760</td>
</tr>
<tr>
<td>workRunning</td>
<td>1367840807</td>
</tr>
<tr>
<td>countAheadOtherQueue</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window size</th>
<th>Prediction (seconds)</th>
<th>Confidence interval (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 hours</td>
<td>31097</td>
<td>(27154, 35040)</td>
</tr>
<tr>
<td>1 day</td>
<td>70848</td>
<td>(67312, 74384)</td>
</tr>
<tr>
<td>1 week</td>
<td>181974</td>
<td>(170804, 193143)</td>
</tr>
<tr>
<td>4 weeks</td>
<td>163391</td>
<td>(157357, 169425)</td>
</tr>
<tr>
<td>8 weeks</td>
<td>470820</td>
<td>(391119, 550521)</td>
</tr>
</tbody>
</table>

Query/prediction results for Gordon.sdsc.xsede.org on 06/23/2016
Simulated Wait Times

- We know the submit time and start time of Job 1
- Can create arbitrary number of experiences satisfying
  - Submitted between (submit, start) of Job 1
  - Then it will start at the same time as Job 1
- Karnak creates two additional experiences of wait times $x/3, 2x/3$ out of a wait time $x$
Simulated Wait Times Increase Predictions

Decision tree built without simulated wait times

Decision tree built with simulated wait times

Simulated wait times

Node in which average wait time is inflated

33%

66%
Discussion

• Selecting best features
  • workAhead may not be accurate
    • Not all job run for maxWallTime
    • Hard-coded maxWallTime
  • Proxy variables for scheduling policy
    • Policy updates

• Limitation of decision tree
  • Each feature has equal weight vs. variable weights on linear regression coefficients
  • Weak to outliers
Current Status and Future Work

• What remains to be done
  • Implementation
    • Give up ensemble method
    • Do not create simulated wait times under exponential wait time distribution
  • Back-testing and bench marking: comparing error rates between old and new decision trees

• Potential future work
  • Improving the confidence interval
  • Multiple linear regression
  • Neural networks
Questions?

Jungha Woo
wooju@purdue.edu
Backup Slides
Service Interfaces

- REST-style interactions
- HTTP is the base protocol
- A few options for encoding data
  - HTML
    - Simple web interface described previously
  - Plain text
    - Supports thin command line clients
  - XML
    - Supports integration with tools
Command Line Programs

• Roughly follows the web pages
  • ksystems – system summary
  • ksystem – jobs in queues on a system
  • kqueue – current and historical job information
  • kjobs – info about waiting jobs
  • kwait, kstart – predict waiting job
  • kwouldwait, kwouldstart – predict potential job

• Implemented in Python
• Provides output as HTML, XML, or text
• Downloadable via the XSEDE Karnak web page
  • http://karnak.xsede.org/karnak/index.html

Slide modified from Warren Smith, with permission
Potential Job Performance

• Post-analysis of performance
  • Event driven simulation
  • Job data recorded from XSEDE systems

• Workload has 2 event types
  • Request a prediction for each job using system state just before it is submitted
  • Insert information about a job wait time as the job starts

• Process:
  • Perform insertions only for X days
  • Perform insertions and predictions for the next Y days
  • Output performance

• Optimized prediction parameters

Slide modified from Warren Smith, with permission
Example GLUE2 Job message
Experience

• Describes something that has happened in the past
• Consists of input features and output features
  • Input features describe conditions
  • Output features describe what happened
  • Each feature has a name and value
  • Values can be linear (numbers) or nominal (strings)
• For example, a job that ran on a cluster yesterday
  • Input features: username, queue, #CPUs
  • Output feature: run time
Implementation

- glue2 provider generates job information
  - Periodic snapshots of queued and running jobs
  - XML in a XSEDE realization of the GLUE 2 schema
- RabbitMQ picks up this information
- No GLUE 2 job information, no predictions

Slide modified from Warren Smith, with permission
Implementation

- GLUE 2 job information published to centralized RabbitMQ
  - Only authenticated access
  - Short list allowed to access

*Slide modified from Warren Smith, with permission*
Implementation

• Archiver process
  • Subscribes RabbitMQ events for job, and queue information
  • Stores job ordering
  • Stores job information (cores, req. wall time, user, …)
  • Generates events for each job (submit, start, complete)

Slide modified from Warren Smith, with permission
Implementation

- Experience Generator
  - Performs scheduling simulations and stores results
  - Constructs experiences
    - job description, queue position, simulated start time

Slide modified from Warren Smith, with permission
Implementation

- Web service interacts with users
  - Job information from database
  - Embedded predictor to form predictions

---

Slide modified from Warren Smith, with permission
Creating a Query to Get Prediction

Submitted job:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Submit time</th>
<th>State</th>
<th># CPU</th>
<th>Requested time</th>
<th>Start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>6/11 3PM</td>
<td>Running</td>
<td>2</td>
<td>6 hrs</td>
<td>6/19 2am</td>
</tr>
<tr>
<td>15</td>
<td>6/16 1PM</td>
<td>Running</td>
<td>3</td>
<td>2 hrs</td>
<td>6/19 3am</td>
</tr>
<tr>
<td>16</td>
<td>6/18 4PM</td>
<td>Pending</td>
<td>5</td>
<td>1 hr</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>6/19 3AM</td>
<td>Pending</td>
<td>1</td>
<td>2 hrs</td>
<td></td>
</tr>
</tbody>
</table>

Unsubmitted job:

<table>
<thead>
<tr>
<th># CPU</th>
<th>Requested walltime</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1200</td>
</tr>
</tbody>
</table>

Feature:

- queue
- processors
- requestedWallTime
- countAhead
- workAhead
- countRunning
- processorsRunning
- workRunning
- countAheadUser
- workAheadUser
- countRunningUser
- processorsRunningUser
- workRunningUser
Creating Experiences to Build Trees

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Submit time</th>
<th>Status</th>
<th># CPU</th>
<th>Request time</th>
<th>Start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>6/8 3PM</td>
<td>Done</td>
<td>24</td>
<td>3 hrs</td>
<td>6/12 2 am</td>
</tr>
<tr>
<td>13</td>
<td>6/10 1PM</td>
<td>Done</td>
<td>3</td>
<td>4 hrs</td>
<td>6/10 3 pm</td>
</tr>
<tr>
<td>14</td>
<td>6/11 3PM</td>
<td>Running</td>
<td>2</td>
<td>6 hrs</td>
<td>6/19 2 am</td>
</tr>
<tr>
<td>15</td>
<td>6/16 1PM</td>
<td>Running</td>
<td>3</td>
<td>2 hrs</td>
<td>6/19 3 am</td>
</tr>
</tbody>
</table>

Only started jobs have wait times
How To Build a Decision Tree

• Top-down, greedy way to build a decision tree
  1. Start from the root
  2. Compute the standard deviation for all data
  3. Split the dataset with each feature. Repeat for 26 features
  4. Choose the feature that reduces standard deviation most
  5. Divide the dataset based on the selected feature
  6. Keep splitting if there is standard deviation reduction
  7. Run recursively on the non-leaf nodes, until all data is processed

An excellent decision tree example is available at http://www.saedsayad.com/decision_tree_reg.htm
Example of Karnak decision tree