Resource Models: Batch Scheduling

- Last Time
  » Cycle Stealing Resource Model
    – Large Reach, Mass Heterogeneity, complex resource behavior
    – Asynchronous Revocation, independent, idempotent tasks
  » Resource Sharing in Utilities
    – What is possible: lots of sharing!
    – Statistical models of performance and sharing
- Today
  » Batch Scheduling Resource Model
  » Batch Queue Wait Prediction
- Reminders/Announcements
  » Project Writeups back today

Today’s Readings

- Portable Batch System Web Site
  [http://wwwpbsprocom/tech_overviewhtml](http://www.pbspro.com/tech_overview.html)
- Brett Bode, David M. Halstead, Ricky Kendall, and Zhou Lei, The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters
- Optional Reading
Resource Management Problem

• Application Perspective:
  » Given my application, find and bind an appropriate (“best”, “acceptable”, “best below acceptable”) set of resources
  » => Optimize for application quality or performance

• System Perspective:
  » For a set of resources, identify a set of applications which make good use of the resources (“best”, “acceptable”, “high utilization”, etc.)
  » => Optimize for system utilization or “total value”

Resource Management Models

• “Cycle Stealing”
  » Volatile, Asynchronous Preemption

• Dedicated Resource Scheduling TODAY
  » Batch schedulers, dedicated resources, advanced reservation

• Time-sharing
  » Slice schedulers, proportional share, guaranteed/best effort

• Objective: Understand implications for distributed application performance and motivations/advantages of various models (reach, local)
Batch Scheduler

Q’s: Job size, Priority, Resource Type, etc.

Job: #procs, memory, runtime, other

Batch Scheduler

PBS, LSF, Maui, SGE, NQS, etc.

Large Cluster #1

Large Cluster #2

Large Cluster #3

CSE225 – Lecture #10

Batch Scheduling Idea

• Resources are expensive
• Value in achieving high resource utilization
  » Get more work done
  » Allow more users access
• Queue a set of Jobs
  » Allows Choice of “best” next job
• Choose jobs based on
  » Priority
  » Resource Requirement
  » Achieve a Performance Goal
• Stage files in, run, stage files out

CSE225 – Lecture #10
Batch Scheduler for a Single Resource

- Submit, wait, run, complete
- Stage files in and out
- "interactive mode"

What is the scheduling discipline?
- FIFO – simplest
- Many others possible
- Optimize for different metrics

Comparing Scheduling Disciplines

- Scheduling policy makes a big difference in perceived performance
FIFO Scheduling

- Average Delay: 26/7 => 3.5

<table>
<thead>
<tr>
<th>Arrive</th>
<th>LvQ</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Shortest First Scheduling

- Average Delay: 10/7 => 1.5

<table>
<thead>
<tr>
<th>Arrive</th>
<th>LvQ</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Other Scheduling Policies

- Random
- Earliest Deadline First
- Combined Aging and Priority
- Min-Max
- Suffrage
- …

Gaming the system…

- Suppose you were in a shortest job first system
  » How would you get high throughput?

- Suppose you were in a FIFO system
  » How would you get high throughput?

- Suppose you needed to run a distributed/grid job involving resources from multiple batch scheduled resources, what would you do?
Complications: Resource Constraints

- Minimum Memory
- CPU speed
- Operating System
  - => thins the pool of candidates
  - => complicates the choices (matching)
- Increases the complexity of scheduling
- More constraints, the longer you wait

Complications: Parallel Jobs

- Require contiguous partition (interconnect topology: security, performance)
- Becomes a bin-packing problem
- More you ask for, the longer you wait
Complications: Advanced Reservations

- Request a set of resources
  - 64 nodes, >1GB memory, runtime = 1 hour
  - At 5pm PST on Friday, May 28, 2008

- Why?
  - Coordinate a distributed job – for a grid experiment, use of an instrument, etc.

- How does this affect the schedule?
  - Block all use of a specific set of nodes in the period
  - Prohibit schedules that “might” run into the period
  - Reduces efficiency

Intelligent Scheduling

- There are holes in the schedule
  - Jobs run shorter, greedy bin packing limits, Advanced Reservations
- Backfilling: allow lower priority jobs to fill the holes in the schedule
- Maui Scheduler
  - Determines start times for large jobs
  - Backfills smaller jobs
- PBS is a “pluggable scheduler framework”
  - Explore backfill performance
  - Explore scheduler performance
Experimental Parameters

- NASA Workload (lots of real parallel computing)
- 64 node cluster, 256MB memory, flat network
- PBS w/ Maui Scheduler
- Job Mix
  - Large (30%), medium(40%), small(20%), debug and failed (<5%)
  - Range of processor numbers (correlated with size)
- Randomized delay for submission (random order)
  - Same order for each experiment

FIFO vs. Maui
Overall Performance and Resource Utilization

- Intelligent Scheduling improves performance by about 10%
- Theoretical Minimum is another 20% lower
- User information is poor (runtime estimates)
- Overall resource utilization is pretty good

SLA for Grid Application

- Request resources
- Queue...
- Queue...
- Queue... Get Resources!!!!
- Run .. Run .. Run ..
- Relinquish resources
- Start all over again

- SLA: Request and wait for variable time (can be days), run for known quantity of resources, start over
- Asynchronous available after long delay, predictable quantity
Batch Queue Wait Time

- Problem: We want to know how long individual jobs will wait before they will acquire the resources then need
  » Perceived execution time is really affected by wait times
- Goal: Rigorous confidence bounds on the amount of time a specific job will wait in a batch queue before it is scheduled on a cluster or parallel machine.
  » Statistical nature implies that a quantifiable confidence range is necessary
  » Need an answer that applies to an individual job
- Better Goal: estimate percentiles and quantified confidence bounds
  » Statistical certainty at specified confidence levels
  » "At most how long will I have to wait before my job runs with 95% confidence?"

How Well Does it Work?

- Examine the batch queue logs that record wait time
- Choose a quantile and a confidence level
  » 0.95 quantile with 95% confidence
- For each job
  » Calculate the upper limit on the quantile
  » Observe whether job’s wait time is less than that limit
- For the entire trace, record the percentage of job wait times that are less than the prediction
  » Value should be less than quantile if method is working
- 5 sites and machines (NERSC, LANL, LLNL, SDSC, TACC)
- 9 years (96 through 05)
- 1,200,000+ jobs
Quantifiable Confidence

Percentage of Wait Times Correctly Predicted with 95% Confidence

- NERSC SP: 3/01 - 3/03
- LANL O2K: 12/99 - 4/00
- SDSC SP: 4/98 - 4/00
- TACC LS: 1/04 - 5/04
- SDSC PGN: 1/96 - 1/97
- Datastar: 4/04 - 1/05

CSE225 – Lecture #10

Capturing Dynamics

Datastar Queue Times and Predictions
April 2004 to March 2005 (96.1% correct)
Choosing the Best Worst Case
TACC and Datastar Upper 95% Predictions
Thursday February 24, 2005

Choosing the Best Number of Processors
Datastar 95% Predictions
June 2004, 1-4 and 17-64 Processors
Prediction Summary

• Combinations of quantiles provide a qualitative way to evaluate resources
  » If median and 95th percentile are lower, chances are job will start sooner
• Quantiles provide a quantitative way to predict possible outcomes
  » 45% chance that a job will start between the median and the 95th percentile

Discussion: Open Questions and Ideas

• How could you include resources such as these in a distributed application?
• What are the implications of having an indeterminate wait?
• What are the implications of having longer queueing times for large resource requests?
• What are the local implications of Advanced Reservations?
• How can you do co-allocation?
• How does batch queue prediction help?
Basic Queueing Theory

- As resource efficiency (or load) goes to 1, delay goes to infinity
- Fundamental tradeoff between resource utilization and latency?

Summary

- Batch Scheduling Resource Model
  » High Resource Efficiency, but Indeterminate Wait
- Complications
  » Requirements; Parallel Jobs
  » Advance Reservations
- Batch Queue Wait Prediction
  » Early results are promising
  » What to do about them? Are they stable?