Motivation

- Many workstations have idle cycles wasted
- Users need more processing power than their own workstation can provide
- If only … a user could use some idle workstation without disturbing the performance of that system
- Let's use CONDOR!!!

Introduction

- Condor is a “specialized workload management system for compute-intensive jobs.”
- Compilation of 3 areas of research
  - Analysis of workstation usage patterns
    - Who are using these workstations?
    - How often are they free?
  - Remote capacity allocation algorithms
    - How do we allocate jobs?
    - Who gets pushed to the top of the queue?
  - Development of remote execution facilities
    - What is the best way to run a job on a remote machine?
    - How can we stop a job midway when the machine is no longer idle?

System Design

- Placement of background jobs should be transparent
  - User doesn’t need to know where job is being executed
- Guaranteed job completion
  - If job fails on one node, it must be restarted somewhere else
- Users expect access to cycles when wanted
  - Users put their idle workstation out there, expect the same
- Local implementation consume very little capacity
  - Users won’t want it if it interferes with local activity

Scheduling Structure

- Condor approach - Centralized/Distributed Hybrid
  - One central node decides which workstation(s) are available
  - Each workstation has its own local scheduler for its assigned jobs
- Workstation Structure
  - Local scheduler
    - Schedules remote jobs on own system
    - Checks every 1/2 minute to see if user has resumed using station
  - Background queue
    - Jobs user submits are placed here
    - Coordinator (on exactly one workstation)
      - Polls every two minutes for available workstations
      - Can be easily moved to another workstation

Scheduling Structure

- Centralized vs. Distributed
- Centralized
  - Assign background jobs to remote workstations
  - Know status of all jobs and where they were executing
  - Disadvantages: Not easily extendible and subject to failure
- Distributed
  - Negotiations among nodes to resolve contentions
  - Not subject to failure if one node fails
  - Disadvantages: Less efficient with workstation allocations
Scheduling Structure

![Diagram of the Scheduling Structure](image)

Figure 1: The Condor Scheduling Structure.

The Remote Unix(RU) Facility

- Turns idle workstations into cycle servers
- Invoking RU launches shadow process that runs locally as surrogate of remote process
- Any system call made on remote machine communicates with shadow process

Checkpointing

- Saving state of program for restarting execution
- State of RU is:
  - Text - executable code
  - Data - initialized variables
  - BSS - uninitialized variables
  - Stack segments
  - Registers
  - Status of open files
  - Any messages sent to shadow where reply hasn’t been received

Fair Access to Remote Cycles

- Different Kinds of Users
  - Heavy - Consume all capacity for long periods
  - Light - Consume remote cycles occasionally
- Manage capacity with Up-Down Algorithm
  - Enables heavy users to maintain steady access
  - Provides fair access to cycles for light users
  - Each workstation is given a schedule index
    - Remote capacity granted, index increased
    - Remote capacity denied, index decreased
    - Index controls scheduling priority
    - Jobs sometimes preempted for higher priority jobs

Performance

- Based on observing 23 workstations for one month

<table>
<thead>
<tr>
<th>User</th>
<th># of Jobs</th>
<th>% of Jobs</th>
<th>Demand/Job</th>
<th>Demand</th>
<th>% Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>690</td>
<td>75</td>
<td>6.2</td>
<td>4278</td>
<td>90</td>
</tr>
<tr>
<td>B</td>
<td>138</td>
<td>15</td>
<td>2.6</td>
<td>345</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>39</td>
<td>4</td>
<td>1.7</td>
<td>19</td>
<td>0.4</td>
</tr>
<tr>
<td>D</td>
<td>40</td>
<td>4</td>
<td>0.7</td>
<td>28</td>
<td>0.6</td>
</tr>
<tr>
<td>E</td>
<td>11</td>
<td>1</td>
<td>1.7</td>
<td>19</td>
<td>0.4</td>
</tr>
<tr>
<td>Total</td>
<td>918</td>
<td>100</td>
<td>5.2</td>
<td>4771</td>
<td>100</td>
</tr>
</tbody>
</table>

Performance - Service Demand

- Most users had average job of longer than 1 hour
- Mean demand - 5 hours
- Median demand - 3 hours
  - Shorter jobs more common

![Graph of Service Demand](image)
Performance - Queue Length
- Jobs arrived in batches
- Difference between total and light is heavy

![Queue Length](image3.png)

Performance - Wait Ratio
- Amount of time job waits versus service time
- Most cases users didn’t wait much time
- Up-down algorithm skews wait time for heavy users

![Average Wait Ratio](image4.png)

Performance - Remote Utilization
- 12438 hours available
- 4771 hours consumed
  - Condor rocks!
- 25% local on average

![Utilization of Remote Resources](image5.png)

Performance - Checkpointing
- Placing and checkpointing times
  - 5 seconds per megabyte
  - Average checkpoint file is .5 megabytes
  - Average cost is 2.5 seconds
- When to checkpoint
  - Location becomes unavailable for remote execution
  - Process bumped by higher priority
    - Happens when no other workstation idle

![Rate Of Checkpointing](image6.png)

Performance - Leverage
- Metric to compare effort local workstation must have to benefit remote execution
- Remote capacity consumed divided by local capacity
- Local capacity
  - Support placement
  - Checkpointing
  - System calls

![Remote Execution Leverage](image7.png)

Discussion
- Disk Space issues
  - Remote - job has to be placed on remote station’s disk
  - Local - Checkpoint files add up with multiple jobs
- Do not place or checkpoint several jobs simultaneously
  - Each job has impact on local node and network
- Alternate approach of periodic checkpoints
  - This way when remote machine becomes unavailable process is immediately killed
**Condor Now**

- Still going strong
- Project website
  - [www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor)
- Has developed Grid compatibility
  - Condor-G allows you to submit jobs to resources accessible through Globus interface
- Available platforms
  - HP PA-RISC, Sun SPARC, MIPS, x86 (Linux & Windows), ALPHA, PowerPC (Mac OS & AIX), Itanium (IA64)

**Questions?**