

Condor - A Hunter of Idle Workstations

Michael J. Litzkow,
Miron Livny,
And Matt W. Mutka

Presented by Nick Rutar

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
- Lets 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

- 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

- 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

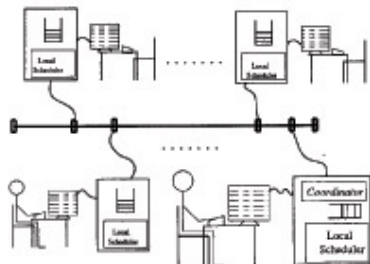


Figure 1: The Condor Scheduling Structure.

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

User	# of Jobs	% of Jobs	Demand/Job	Demand	% Demand
A	690	75	6.2	4278	90
B	138	15	2.5	345	7
C	39	4	2.6	101	2
D	40	4	0.7	28	0.6
E	11	1	1.7	19	0.4
Total	918	100	5.2	4771	100

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

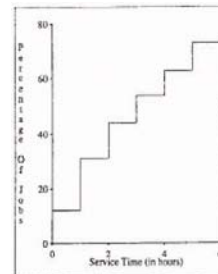


Figure 2: Profile Of Service Demand.

Performance - Queue Length

- Jobs arrived in batches
- Difference between total and light is heavy

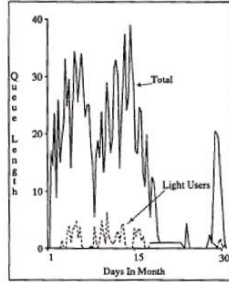


Figure 3: Queue Length.

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

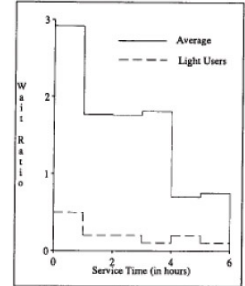


Figure 4: Average Wait Ratio.

Performance - Remote Utilization

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

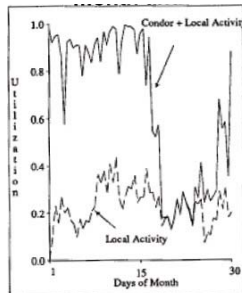


Figure 5: Utilization of Remote Resources.

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

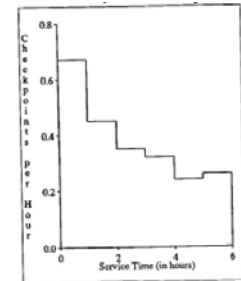


Figure 8: Rate Of Checkpointing.

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

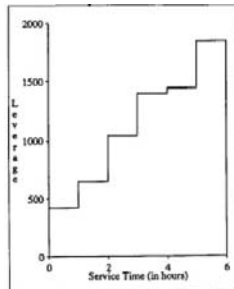


Figure 9: Remote Execution Leverage.

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
- 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?

