CSci 5105

Introduction to Distributed Systems

Distributed Scheduling
Last Time

• Recovery

• Classic Design
  – E2E
  – Lampson Hints
Scheduling Jobs in DS

- Jobs are unrelated
- Two approaches
  - E&L: little is known about the job/resource
  - Condor: user/owner provides information
- Objective function is throughput, average response, waiting time, turnaround time
- E&L => turnaround time (response time)
- Condor => throughput
Eager et al Paper

- **Load sharing**: all processors are busy if a job is waiting
  - not the same as *load balancing*
- Queuing model of sender-initiated load sharing in homogeneous DS
  - Jobs are homogeneous
  - Machines are homogeneous
  - Each job takes unit time
- Fully decentralized
Eager et al Paper (cont’d)

• *Transfer policy*: when to process job locally (threshold) based on queue length
  – What is the tradeoff?

• *Location policy*: where to send if non-local
Eager et al Paper (cont’d)

• *Control* policy: prevent thrashing (hop-limit)
  – E.g. limit of k hops
  – $k^{th}$ host must accept or original node
What Location Policies to Use?

• Simple
  – random
  – probe a few neighbors
    • probe-threshold (probed node is asked whether new job would put it above threshold)

• Complex: shortest queue in the system (higher overhead)

• Belief: complex would be best
Results

- What does this tell us?

No load sharing (k queues)
random
threshold/shortest

Perfect load sharing (1 queue)

average response time
Conclusions

• Simple policies work well under low-med load
  – Given their assumptions

• In general, sender-initiated schemes work well in lightly loaded system
Receiver Initiated

• Pull model
  – Lightly-loaded receivers “pull” jobs from overloaded processors

• Transfer policy: use queue length
  – When queue length < threshold, locate a loaded processor

• Use probes to identify heavily loaded neighbor, random neighbor, etc…
Receiver Initiated (cont’d)

• Problem: which job do we transfer from the sender?
  – Not an issue with sender-initiated
  – Prefer a job not yet running

• More stable at high load
  – Less comm. and job transfers than sender-init.
  – Why?
Receiver Initiated (cont’d)

![Graph showing response time vs. system load with curves for Sender-initiated and Receiver initiated]
Hybrid Scheme

• Ideas?
Condor (HTCondor)

• **Widely-used** distributed platform
  – http://research.cs.wisc.edu/htcondor/

• Cycling harvesting

• Compute centric

• Some centralization

• Two stakeholders
  – machine owner, job owner

• Open Science Grid
  http://display.grid.iu.edu/
What is Condor?

• Condor converts a collection of unrelated workstations into a high-throughput computing facility

• Condor uses matchmaking to ensure that jobs and resources match up
  – environment is highly heterogeneous in both jobs and resources
What is High-Throughput Computing?

• High-performance: CPU cycles/second under ideal circumstances
  – “How fast can I run simulation X on this machine?”

• High-throughput: CPU cycles/day (week, month, year?) under non-ideal circumstances
  – “How many times can I run simulation X in the next week using all available machines?”
What is High-Throughput Computing?

• Condor will execute your job under numerous machine “failures”
  – crash!
  – disconnection
  – disk space exhausted
  – nodes removed or added from the Condor machine pool
  – nodes are shared
What is Matchmaking?

• Condor uses Matchmaking to make sure that work gets done within the constraints of both users and owners

• Users (jobs) have constraints:
  – “I need an Sun Sparc with 256 MB RAM”

• Owners (machines) have constraints:
  – “Only run jobs when I am away from my desk and never run jobs owned by Bob.”
Remote Execution

• Job needs to execute on a remote machine
  – What are the issues?
    • System calls are redirected back to initiating machine
    • Security sandboxing
Machine States

• Most machines will be:
  – **Owner:**
    • The machine’s owner is busy at the console, so no Condor jobs may run
    • Condor will seamlessly migrate jobs via checkpointing* (rerun on identical machine)
  – **Claimed:**
    • Condor has selected the machine to run jobs for other users
Machine States

• Only a few should be:
  – **Unclaimed:**
    • The owner is gone, but Condor has not yet selected the machine
  – **Matched:**
    • Between claimed and unclaimed
  – **Preempting:**
    • Condor is busy removing/migrating a job
Steps to Running a Job

• Establish Condor pool (sys admin)
• Re-link your job for Condor
• Submit the job
• Watch the progress
• Receive email when done
Example Job

Compute the nth Fibonacci number.
Fib(40)

% ./fib 40
fib(40) = 102334155
#include <stdio.h>
#include <stdlib.h>

int fib( int x )
{
    if( x<=0 ) return 0;
    if( x==1 ) return 1;
    return fib(x-1) + fib(x-2);
}

int main(int argc, char *argv[])
{
    int n;
    n = atoi(argv[1]);
    printf("fib(%d) = %d\n", n, fib(n));
    return 0;
}
Re-link for Condor

• Normal compile:
  • gcc -c fib.c -o fib.o

• Normal link:
  • gcc fib.o -o fib

• Use the same command, but add condor_compile:
  • condor_compile gcc fib.o -o fib
Watch the Progress

% condor_q

```
-- Submitter: axpbo8.bo.infn.it : <131.154.10.29:1038> :

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>thain</td>
<td>6/21 12:40</td>
<td>0+00:00:15</td>
<td>R</td>
<td>0</td>
<td>2.5</td>
<td>fib 40</td>
</tr>
</tbody>
</table>
```

Each job gets a unique number.

Status: Unexpanded, Running or Idle

Size of program image (MB)
Receive E-mail When Done

This is an automated email from the Condor system on machine "axpbo8.bo.infn.it". Do not reply.

Your condor job
/tmp_mnt/usr/users/ccl/thain/test/fib 40
exited with status 0.

Completed at:        Wed Jun 21 14:36:36 2000

Real Time:           0 00:11:54
Run Time:            0 00:06:52
Committed Time:      0 00:01:37
...
Matchmaking: ClassAds

• ClassAds are a simple language for describing both the properties and the requirements of jobs and machines
ClassAd for a Machine

- condor_status -l caesar.cs.umn.edu

```plaintext
MyType = "Machine"
TargetType = "Job"
Name = "caesar.cs.umn.edu"
START = TRUE
VirtualMemory = 342696
Disk = 28728536
Memory = 160
Cpus = 1
Arch = "Sparc"
OpSys = "Solaris"
```
ClassAd for a Job

• condor_q -l 9.49

MyType = "Job"
TargetType = "Machine"
Owner = "jon"
Cmd = "/tmp_mnt/usr/users/jon/test/fib"
Out = "fib.out.49"
Args = "49"
ImageSize = 2544
DiskUsage = 2544
Requirements = (Arch == "Sparc") && (OpSys == "Solaris") &&
(Disk >= DiskUsage) && (VirtualMemory >= ImageSize)
Condor Architecture

- User
- Problem Solver
  - (DAGMAN)
  - (Master-Worker)
- Agent
  - schedd
- Resource
  - Central Manager
- Matchmaker
- Shadow
  - Shadow
- Sandbox
  - Starter
- Job
Next Time

• Distributed File Systems
• Read Chapter 11 TVS