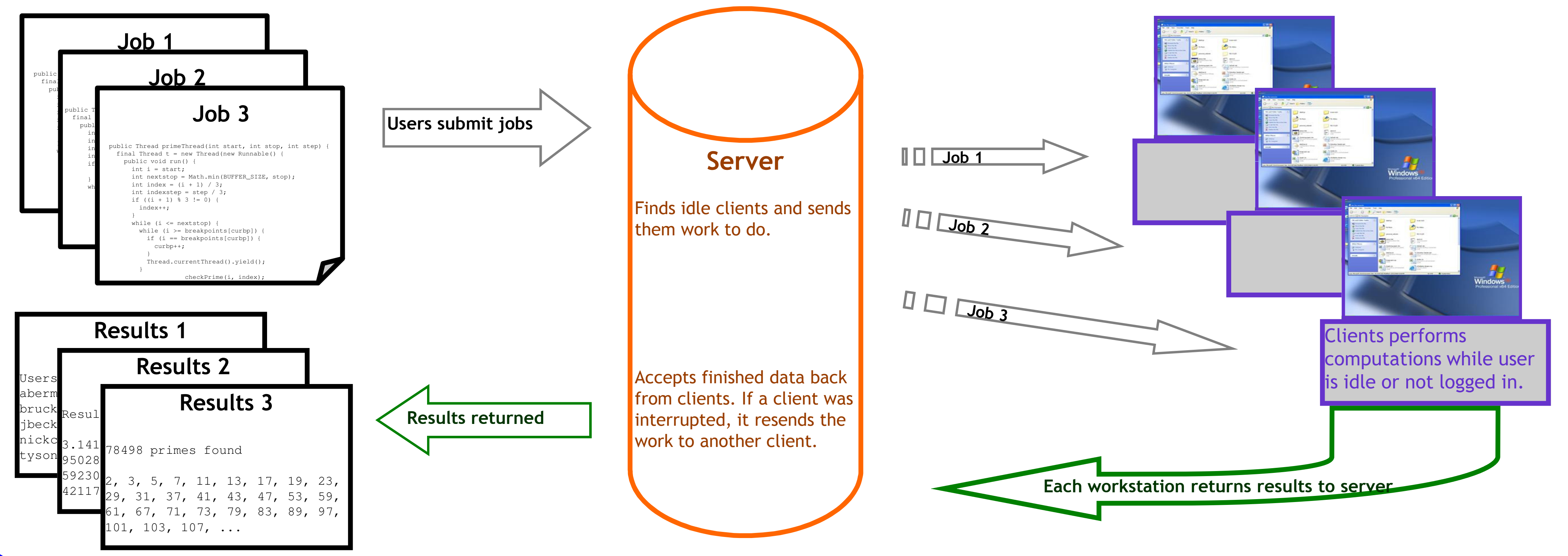
Preparing a Cycle-Scavenging Cluster

A School of Computing project by Andrew Bermudez

The Problem

Parallel programming is among the most important and exciting new areas in computer science courses. It is difficult to gain experience in parallel programming without access to a cluster for program execution. By providing a system to design and test resource-intensive software, a powerful, flexible cluster would enhance the learning experience of the students at the School of Computing. My previous study of options for creating such a cluster utilizing the Advanced Lab computers concluded that it would be wise to use the cycle-scavenging solution presented by Condor to provide these capabilities. I installed and configured the Condor clustering solution on computers in the computer labs.

How Cycle-Scavenging Works


Job 1
Job 2
Job 3
Users submit jobs
Server
Finds idle clients and sends them work to do.
Accepts finished data back from clients. If a client was interrupted, it resends the work to another client.
Job 1
Job 2
Job 3
Clients performs computations while user is idle or not logged in.
Each workstation returns results to server
Results returned
Results 1
Results 2
Results 3
78498 primes found
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, ...

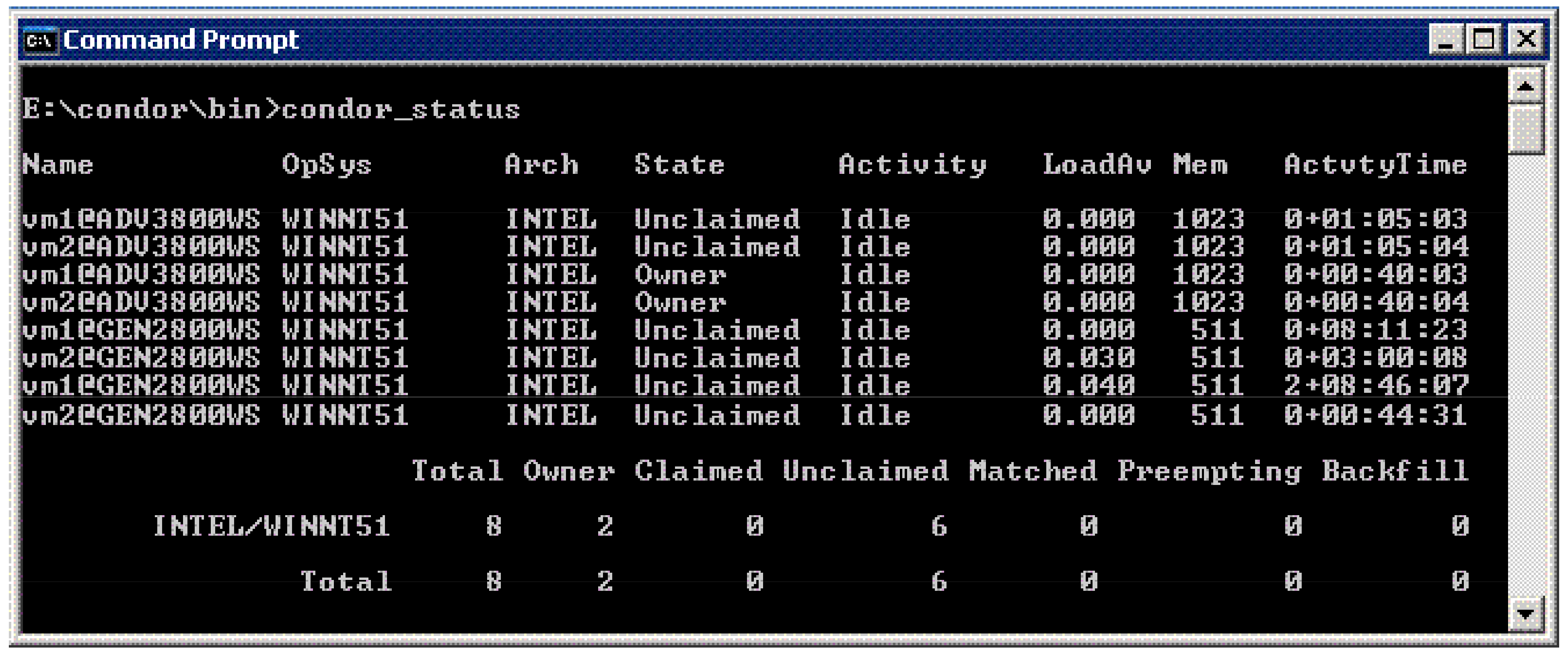
A Condor Job

```
Universe = java
Requirements = (Arch == "INTEL") && (OpSys == "WINNT51") && (Disk >= DiskUsage) && ((Memory * 1024) >= ImageSize) && (HasFileTransfer)
Executable = e:\condor\execute\FindPrimesHD.class
Arguments = FindPrimesHD
Output = e:\condor\execute\output.txt
Input = e:\condor\execute\input.txt
Error = e:\condor\execute\error.txt
Log = e:\condor\execute\log.txt
should_transfer_files = YES
transfer_input_files = e:\condor\execute\FindPrimesHD$1.class, e:\condor\execute\FindPrimesHD$2.class, e:\condor\execute\In.class
when_to_transfer_output = ON_EXIT
MaxJobRetirementTime = 10000
Queue
```

Still to be Done...

Now that Condor has been tested on both General Lab and Advanced Lab machines, it can be deployed to all of the machines in both labs. Then students at the School of Computing can begin to make use of the Condor cluster by writing code that can take advantage of the extremely powerful parallel universe. It will take some special accommodations to write code that can be run within Condor's parallel capabilities, but that is within the reasonable challenges of parallel software development.

The Condor Status Display



```
E:\condor\bin>condor_status

Name          OpSys       Arch   State     Activity   LoadAv Mem   ActvtyTime

vm1@ADV3800WS WINNT51     INTEL  Unclaimed Idle       0.000  1023  0+01:05:03
vm2@ADV3800WS WINNT51     INTEL  Unclaimed Idle       0.000  1023  0+01:05:04
vm1@ADV3800WS WINNT51     INTEL  Owner     Idle       0.000  1023  0+00:40:03
vm2@ADV3800WS WINNT51     INTEL  Owner     Idle       0.000  1023  0+00:40:04
vm1@GEN2800WS WINNT51     INTEL  Unclaimed Idle       0.000   511  0+08:11:23
vm2@GEN2800WS WINNT51     INTEL  Unclaimed Idle       0.030   511  0+03:00:08
vm1@GEN2800WS WINNT51     INTEL  Unclaimed Idle       0.040   511  2+08:46:07
vm2@GEN2800WS WINNT51     INTEL  Unclaimed Idle       0.000   511  0+00:44:31

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/WINNT51     8     2       0         6       0          0        0

                 Total     8     2       0         6       0          0        0
```

Condor Universes

	Standard	Vanilla	Java	Parallel
Description	Code that has been compiled using Condor's own special libraries.	Any arbitrary program that can be run natively on the workstations, compiled in any language.	Any Java class. For checkpointing support, implementation of a specific template is required.	Any arbitrary program; the same program will be run on as many machines as requested. MPI can be used for coordination.
Advantage	Allows for the use of checkpointing capabilities on non-parallel jobs.	Can run any non-parallel program on a cluster machine.	Allows Java programs to run in Condor.	Takes advantage of the power of parallel programming on a cluster.
Disadvantage	Requires special compilation, may not be possible or practical.	Does not use checkpointing, meaning that long jobs may be interrupted and restarted numerous times.	Does not allow for parallel execution: runs on one machine only.	Must use technologies such as MPI to coordinate communications between processors.
Experience	I did not have time to create and compile code with the Condor libraries.	The vanilla universe was quite easy to run jobs in, and handled test jobs with no trouble.	Jobs in the Java universe ran successfully except for trouble with the early termination of multi-threaded classes.	I do not currently have access to code written with MPI.