A Distributed Computing System with Condor and Cooperative Linux

Marc Noël

December 14, 2006

LSU’s large investment in High Performance Computing (HPC) keeps the campus at the forefront of modern technology and augments research in several fields. As a side effect of this great success, campus HPC resources run near full capacity, meaning that increased interest in HPC leaves each researcher with a smaller share of the campus’ computational power. By making use of idle desktops around campus, some researchers can shift their programs to a distributed pool of machines, freeing up resources on the traditional systems. To this end, a campus-wide distributed computing system is proposed, with a proof-of-concept implementation described and steps for moving forward outlined. This particular implementation makes use of two major off-the-shelf components: Condor, a ‘cycle scavenger’ and distributed scheduling tool, and coLinux, an open-source virtual machine for Linux on Windows.

The Need for a Campus Grid

Despite boasting over 8.5 Tflops across five production high-performance systems, LSU continuously runs against the limits of its computing power. Looking around campus HPC, an observer might note:

1. In 2006, LSU’s HPC department stated that, on average, SuperMike was 93.8% active, leaving only 6.2% of the machine for new jobs.

2. On 12 Nov 2006, when most researchers had left town for Supercomputing ’06, Santaka – a 32-processor machine with 128 GB shared memory – stood almost completely consumed by a single job, with two other jobs of similar requirements waiting in the wings. More than two weeks later, on 30 November, the last of these three jobs began executing.

---

1SuperMike, SuperHelix, Nemeaux, Santaka, and Pelican. MiniMike is dedicated to training and should not be considered a production resource. http://www.hpc.lsu.edu/systems/

2http://www.hpc.lsu.edu/systems/usage/reports/LSU_HPC_Usage_Reports_20050101-20060822.pdf

3See Listing 1

4See Listing 2
3. Also on 12 Nov, SuperMike showed 80% of resources consumed by 20 active jobs. Some jobs waiting in the wings requested as much as one half the entire system. Administrators explained that the 20% inactivity rarely occurs, and that a large job scheduled next was momentarily hoarding CPUs to accumulate enough for execution.

The campus HPC group tells of preparations for more than 17 TFlops worth of computer power in the near future, and the campus has also procured funds for an even more powerful 50 TFlops “centerpiece” ringed by 30 Tflops of additional power from LONI. This power won’t be available for some time, but while the LSU research community waits, some readily available resources may help alleviate the cycle crunch.

LSU’s Information Technology Services runs a number of campus computer labs and classrooms. Computers in these rooms stay powered on twenty-four hours a day, seven days a week, leaving them almost completely idle for about a third of their operational lifetimes. For instance, Middleton 241 closes between 11:30 PM and 7:30 AM on weekdays, leaving all 151 machines idle for eight hours a day. Even discounting the cycles thrown away by a machine only using a fraction of its CPU for routine desktop tasks, we can safely assert that we fail to use a third of the public labs’ desktop computing cycles. In the previously mentioned Middleton 241, a standard machine contains a Pentium 4 processor operating near 6 Gflops, and according to their website, ITS maintains 369 computers in public labs and 176 in classrooms. In this case, our cycle-strapped campus is neglecting a system rated at approximately 6 GFlops * (369 + 176 machines) * (1/3 day) = 1.09 TFlops. Finding a way to use these squandered cycles would boost available computing power by 13% and weigh in as the second largest HPC system currently on campus, trailing SuperMike at 6.267 Tflops and near SuperHelix’s 1.024 Tflops. However, ITS claims that they actually have closer to a thousand computers, and the Centers for Excellence and Learning and Teaching plans on expanding their 150 desktops to “300 to 400”. In this case, a potential Condor pool would have more CPUs than any currently available campus resource, and provide a to 27.9% boost of 2.8 Tflops

---

5See Listing 3
6http://hpc.lsu.edu/systems/; the sum of Tezpur and Pelican2
7http://app003.lsu.edu/itsweb/cioweb.nsf/Content/Presentations/file/execbriefing1006.ppt, page 19
8http://app003.lsu.edu/ocsweb/palsoftware.nsf/GetHours?openagent&Middleton%20241-%20Phone:%%20%20578-0044
9http://www.cs.utk.edu/~dongarra/Dongarra-ASCAC-101702.pdf, slide 16; also, consider that Supermike’s 6.267 Tflops are spread across 1024 processors, leading to an estimate of around 6.12 Gflops per processor, and that SuperMike and LSU desktops both hail from the same late-model Pentium 4 vintage
10http://app003.lsu.edu/ocsweb/palsoftware.nsf/(LabsByName)?OpenView; the sum of all workstations in all public labs, except Coates 261, which has been retired.
11http://app003.lsu.edu/ocsweb/palhome.nsf/$Content/Classrooms?OpenDocument; sum of all workstations in all classrooms
12Phone conversation with Sherri Thompson, IT Communications and Planning Officer, Office of the CIO, LSU ITS. 6 December 2006
to the campus HPC environment.

However, several circumstances prevent taking full advantage of this processor power. In contrast to cluster computers, which generally use high speed, low latency connections such as SuperMike’s Myrinet or Santaka’s NUMAlink, desktops dispersed around campus connect over 100 Mbps or 1 Gbps Ethernet. While Ethernet works well for desktop users moving files or streaming web content, its latencies often exceed those of high-performance connections by an order of magnitude,\(^{14}\) which severely slows message passing code and renders this pool of idle machines less attractive for MPI programs. Secondly, as desktop computers, the lab machines’ 32 bit processors and 1 GB RAM\(^{15}\) pale in comparison to dedicated cluster nodes which have at least twice as much memory and are moving to 64 bit architectures. Finally, and most dauntingly, campus desktops almost exclusively run Windows. Despite a failed pilot program with dual-booting Linux and Windows systems,\(^{16}\) and 30 new Macintoshes,\(^{17}\) Unix operating systems are understandably absent from general-purpose desktops.

In contrast, all of our current high-performance systems and software use some form of Unix, be it AIX, Linux, or Mac OS X. Even compiling the same code on different Unix systems is often difficult; for example, compiling the current version of some infrastructure tools took several days of trial and error for our administrative staff.\(^{18}\) Researchers hoping to use these spare cycles may not have to completely re-write their code, but would likely have to cope with even more idiosyncrasies than now, trading time spent analyzing data or devising new experiments for babysitting and tweaking their programs.

From this perspective, that potential 1.09 Tflops may remain unused despite any scheme to link campus desktops together. However, some research, such as UCoMS development, Dr. Eric Hillebrand’s financial modeling, or Dr. Zhengyuang “Hugh” Wang’s biological statistics, neither require low-latency message passing, nor 64 bit processor features, nor relatively large local memory. Two problems then remain: overcoming the impediment of different operating systems, and distributing researchers’ programs to the disparate available machines.

The Solution

When faced with the same problem, the staff at the University of Nebraska-Lincoln proposed an inventive answer: use the Cooperative Linux (‘coLinux’) to virtualize Linux on the scores of available Windows computers, and sched-
ule jobs using Condor, which is specifically designed as infrastructure for ‘embarassingly parallel’ distributed computing.\textsuperscript{19} Aside from the development time savings gained from using ‘off-the-shelf’ technologies, a Linux virtual machine provides binary compatibility for already-existing code, minimizing if not outright eliminating the time spent adapting existing programs or habits for a new system. A dual-boot environment could accomplish the same feat with better performance, but disrupts ITS’ typical Windows-only policies involving system installation, software updates (a computer booted into Linux cannot accept nightly Windows patches), or general support.\textsuperscript{20} In contrast, virtualized Linux appears as a simple extra service within the typical ITS Windows environment. Neither Linux nor Windows administrative personnel must give their counterparts passwords or permissions for the system, or consult each other as frequently as a dual-boot environment requires, leading ITS to publicly support this plan.\textsuperscript{21}

Virtualization has unfortunate downsides, however; most notably, the expense of virtualizing a separate system places special importance in choosing a low-overhead virtual machine. Though UNL cites\textsuperscript{22} an IBM-published assertion\textsuperscript{23} that coLinux performance approaches native execution speed, a CPU-bound, Fibonacci number-based benchmark rates the coLinux virtual machine at about 79.3\% of the system’s native speed.\textsuperscript{24} Considering the overhead of the coLinux, the expected power of our figurative 1.09 Tflops system falls to about .86 Tflops. Despite such a drop, this number still represents around a tenth of the current usable campus high performance cycles, and remains impressive as other available resources throughout ITS and the testing labs come to bear.\textsuperscript{25}

No obviously superior product challenges coLinux; even if other virtual machines display better performance in one area, they suffer slight setbacks in other areas. In comparison, a popular commercial solution, VMWare, performs at around 77\% of the native speed. Conversely, VMWare typically requires less RAM for additionally services and overhead. Since the virtual machine would likely run in the background at all times, a minimal overhead would create a more responsive environment for Windows users, lowering complaints and encouraging ITS to continue supporting cycle scavenging. In fairness to coLinux, however, memory can be swapped out to disk, but no backing store can accelerate CPU performance. To further differentiate the competitors, coLinux’s GPL allows wholesale redistribution of changes or modifications, which aids those hoping to duplicate or improve on our work; VMWare’s commercial position will not allow distribution as part of any software packages we might like to dis-

\textsuperscript{19}http://mindspawn.unl.edu/?page\_id=17
\textsuperscript{20}IT’s training policies for technicians only involve a token amount of Linux training; see http://status.lsu.edu/start/2006/01/ for an example
\textsuperscript{21}http://appl006.lsu.edu/itsweb/cioweb.nsf/$Content/Presentations/Presentations/$file/execbriefing1006.ppt, slide 4
\textsuperscript{22}http://mindspawn.unl.edu/CoC/CondorWeek2006.ppt
\textsuperscript{23}http://www.ibm.com/developerworks/linux/library/l-colinux/
\textsuperscript{24}See Figure 1, Relative CPU Performance
\textsuperscript{25}See Figure 4, Potential Yield
tribute in the future.\footnote{http://register.vmware.com/content/eula.html; specifically, 3.3 Restrictions. You may not (i) sell, lease, license, sublicense, distribute or otherwise transfer in whole or in part the Software or the Software License Key to another party...} Given the paucity of multi-core desktops,\footnote{Even the newest desktops are Dell Optiplex GX620’s. These cannot be configured with Intel’s Core Duo or Core 2 Duo processors; http://www.dell.com/content/products/productdetails.aspx/optix_gx620?c=us&cs=04&cl=en&s=bsd} CoLinux’s slightly higher CPU performance and licensing

Although the main focus of remains on implementation, the grid at LSU has some improvements over the UNL campus grid. Both the host PC and the virtual Linux node require distinct network addresses, a feat previously accomplished by use of the Generic Connection Broker (GCB). GCB had been a source of consternation, and its requirement of a desktop dedicated solely to routing incited a search for alternatives. After attempting solutions with Windows’ deprecated bridging tools, ITS created a 10.41.x.x private network parallel to the public 130.39.x.x network. Subsequently, the LSU grid is now able to use the Windows Packet Capture libraries, or WinPcap, to dispense with the dedicated GCB. The UNL system also relies on CD-ROM and a hand-guided installation procedure; however, thanks to ITS’ support, LSU’s Grid will be deployed and extended through the pre-existing mass installation procedures (such as Microsoft SMS or ITS’ PCUpdater-based system).

The Implementation

The following is a step-by-step procedure for reproducing the non-GCB proof of concept. It does not cover packaging and distribution, as different sites will have their own policies and procedures established by their IT departments. LSU’s public labs revolves around PCUpdater and in-house tools whose purpose is similar to \texttt{scp} and \texttt{ssh <host><command>}. colin\begin{verbatim}begin\end{verbatim} contains some hard-coded path strings, and so paths and file names are important. This process will not be necessary when deploying to host desktops; instead, it describes the process for creating a Condor-in-coLinux development environment.

1. Create a Condor master. Installing Condor, in general, surpasses the scope of this document; however, see \url{http://www.cs.wisc.edu/condor/manual/v6.8/3_2Installation.html}.

2. Create a private network parallel to the network the host machines will use. At LSU, subnets of the 130.39.0.0/16 network contains almost all computers connected via wired ethernet, including the target workstations. The campus networking group identified subnets containing host computers and decided that virtual machines would have an address in the 10.41.0.0/16 network. Physical interfaces have corresponding host ID numbers in both networks. In practice, this means host 130.39.208.120 will run the virtual machine with address 10.41.208.120, and that 10.41.208.120 can be reached from any 130.39.0.0/16 address. As a
bonus, LSU’s networking group also made the the 10.41.0.0/16 network available from other networks outside 130.39.0.0/16, such as the wireless network.

3. Download coLinux 0.6.4 from http://sourceforge.net/project/showfiles.php?group_id=98788&package_id=107317&release_id=429015. Earlier versions lack the necessary networking support to work without GCB, so be certain to use 0.6.4 or greater. Install coLinux in C:\colinux, creating the directory if it does not exist.

4. Download the base Fedora 5 disk image from http://sourceforge.net/project/downloading.php?group_id=98788&filename=FedoraCore5-2006.8-ext3-2gb.7z&90237686. Decompress the image in C:\colinux. If the Windows system does not understand .7z archives, a decompression tool is available from http://www.7-zip.org/. Name this image colinux.ext3.

5. Create a swap partition. First, estimate the amount of swap space desired (usually anywhere from one or two times the amount of physical RAM available). Next, run the following command in Linux:

```
dd if=/dev/zero of=swap.img bs=1024 count=swap size in MB
```

Compress this file (using zip, 7z, etc) and copy it to the Windows host. Place the uncompressed copy in C:\colinux, and check to make sure the name swap.img remains.


7. Create a temporary coLinux starter script for development purposes. For example, C:\colinux\start-colinux.bat (the name is unimportant) might contain:

```
colinux-daemon.exe ^
mem=768 ^
kernel=C:\colinux\vmlinux ^
cobd0=C:\colinux\colinux.ext3 ^
cobd1=C:\colinux\swap.img ^
cofs0=C:\colinux ^
eth0=pcap-bridge,"Local Area",04:AA:BB:CC:DD:EE ^
root=/dev/cobd0
```

The cofs0 directive allows the Linux virtual machine to mount the specified directory as a read/write device. Local Area is a substring of the name for the host’s network connection; it may need to be changed. The MAC address portion of the eth0 line should be an invalid MAC address (i.e., start with a byte other than 00) to avoid collisions with other ethernet devices on the same subnet.
8. Start coLinux by double-clicking on the development starter script. Within coLinux,

(a) log in with username root, password root

(b) Create a place to mount the Windows coLinux directory with the command

\[ \text{mkdir /mnt/windir} \]

(c) Add the following line to /etc/fstab, which will automatically mount the shared Windows directory on boot:

\[ \text{cofs /mnt/windir cofs defaults 0 0} \]

(d) Activate eth0 with an IP address on the host network (in the LSU example, the 130.39... network, as not the 10.41... network). Be sure not to use the same IP address as the Windows host.

(e) Download a custom coLinux starter script, which will repair the network connection using files generated by colinbegin, and launch coLinux:

\[ \text{wget -o /etc/init.d/condor_in_colinux \} http://cct.lsu.edu/~mnoel/condor_in_colinux} \]

(f) Set the coLinux starter script to run on boot:

\[ \text{ln -s /etc/init.d/condor_in_colinux \} /etc/rc3.d/S11condor_in_colinux} \]

(g) Download Condor 6.8.2 from http://www.cs.wisc.edu/condor/downloads/v6.8.license.html. Perform a typical install, with a base directory of /usr/local/ such that condor_* executables are in /usr/local/bin and /usr/local/sbin. (In addition to putting these commands in default $PATHs, scripts such as condor_in_colinux will expect this path.)

(h) Change /etc/hosts to make sure the machine’s hostname is not grouped with localhost. For example:

\[ \text{127.0.0.1 localhost localhost.local} \]
\[ \text{130.39.208.64 C263PC33.lsu.edu} \]

(i) Start condor_master. Ensure that condor_status lists the virtual machine, and that submitted jobs are accepted.

9. Download http://cct.lsu.edu/~mnoel/colinbegin.zip, which contains a Visual Studio project of the colinbegin starter program. Search for the string ‘‘130.39’’ and, if outside LSU, replace the network ID string. Build the project, and copy the compiled binary into C:\colinux. Double-clicking on colinbegin should now launch coLinux, generate relevant network configuration files for Linux, and leave the virtual machine ready to join the Condor pool.
The Future

Currently, the coLinux testing pool consists of a development laptop and the Distributed Systems Lab Condor test pool (mixed Windows, 32- and 64-bit Linux, and coLinux). PCs in CEBA have Condor in VMWare installed, and will soon be transitioned to coLinux. Labs in CEBA, Coates, and Middleton Library are targeted for the first installations, and already have the 10.41 parallel network in place. In keeping with other Linux administration practices around CCT, our Linux image targets the RPM repositories on is.cct.lsu.edu for updates, as well as a separate repository for new packages and changes specific to the desktop pool.

As a proof of concept, and for testing purposes, campus grid compute nodes currently start through the colinbegin process, which verifies a proper network configuration before generating configuration files and starting the virtual machine. Currently, colinbegin is transitioning to an on-boot service which perform the proper configuration and startup without user intervention or without starting the coLinux GUI console. Some minor enhancements, such as configuration files for important paths, will also take place. Thereafter, the current virtual machine and supporting programs will be packaged as a .msi installer or self-extracting PCUpdater package, to facilitate automated deployment. Additionally, the Linux distribution virtualized by coLinux needs slight modification before production. For convenience, certain research libraries may be added on all machines.

Several core research domains will provide the first applications for the campus grid. Single-processor fluid dynamics code with parameter sweeps will replace trivial test cases and provide a method for benchmarking the campus cluster’s growth. UCoMS researchers should become the first production users, followed by individuals who have expressed interest or who are known to work on embarrassingly parallel problems. For example, the Cactus group may also have a testbed for reviving their Cactus Task Farming Manager28, especially as Cactus users become aware of the new pool. As the desktop grid development moves to maintenance, individual undergraduate students may be given submission privileges to begin forays into grid computing – a mission similar to that of Gumbo Grid, but with a larger audience and a lower entrance barrier. In the future, nearby research groups (notably LIGO, which maintains their own Condor pool) may be able to use resources via Condor’s flocking, which provides submission rights to a distinct group of Condor uses on a separate network. Overall, despite newer HPC systems, operating this scavenging grid will give LSU experience in dealing with larger-scale grid environments than we have in the past, and demonstrate to others our competence in running diverse types of computational resources.

Figures and Listings

Figure 1: Relative Virtual Machine CPU Performance

<table>
<thead>
<tr>
<th>System</th>
<th>Trials</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows (native)</td>
<td>585</td>
<td>586</td>
</tr>
<tr>
<td></td>
<td>591</td>
<td>592</td>
</tr>
<tr>
<td></td>
<td>590</td>
<td>590</td>
</tr>
<tr>
<td></td>
<td>588.8</td>
<td></td>
</tr>
<tr>
<td>coLinux (Fedora 5)</td>
<td>740</td>
<td>739</td>
</tr>
<tr>
<td></td>
<td>740</td>
<td>739</td>
</tr>
<tr>
<td></td>
<td>755</td>
<td>755</td>
</tr>
<tr>
<td></td>
<td>742.6</td>
<td></td>
</tr>
<tr>
<td>VMWare (Fedora 5)</td>
<td>753</td>
<td>775</td>
</tr>
<tr>
<td></td>
<td>754</td>
<td>765</td>
</tr>
<tr>
<td></td>
<td>768</td>
<td>768</td>
</tr>
<tr>
<td></td>
<td>763</td>
<td></td>
</tr>
</tbody>
</table>

For Figure 1, all values are reported in seconds. The values reflect a Fibonacci number-based benchmark’s run time. Total overhead is calculated using the formula:

\[
\text{Overhead} = \sum(\text{VMWare processes}) - \text{output of free}
\]

Figure 2: VMWare Memory Overhead

<table>
<thead>
<tr>
<th>Process</th>
<th>Trials</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmmat</td>
<td>1.976</td>
<td>1.926</td>
</tr>
<tr>
<td></td>
<td>1.972</td>
<td>1.936</td>
</tr>
<tr>
<td></td>
<td>1.972</td>
<td>1.9564</td>
</tr>
<tr>
<td>vmnetdhcp</td>
<td>1.772</td>
<td>1.724</td>
</tr>
<tr>
<td></td>
<td>1.736</td>
<td>1.724</td>
</tr>
<tr>
<td></td>
<td>1.724</td>
<td>1.7392</td>
</tr>
<tr>
<td>vmount2</td>
<td>4.74</td>
<td>4.708</td>
</tr>
<tr>
<td></td>
<td>4.712</td>
<td>4.708</td>
</tr>
<tr>
<td></td>
<td>4.708</td>
<td>4.7152</td>
</tr>
<tr>
<td>vmserverdwin32</td>
<td>21.468</td>
<td>21.324</td>
</tr>
<tr>
<td></td>
<td>21.076</td>
<td>21.216</td>
</tr>
<tr>
<td></td>
<td>21.216</td>
<td>21.2232</td>
</tr>
<tr>
<td>vmware</td>
<td>23.876</td>
<td>23.26</td>
</tr>
<tr>
<td></td>
<td>23.876</td>
<td>23.264</td>
</tr>
<tr>
<td></td>
<td>23.264</td>
<td>23.6472</td>
</tr>
<tr>
<td>vmware-authd</td>
<td>5.864</td>
<td>5.904</td>
</tr>
<tr>
<td></td>
<td>5.892</td>
<td>5.904</td>
</tr>
<tr>
<td></td>
<td>5.904</td>
<td>5.8928</td>
</tr>
<tr>
<td>vmware-remotemks</td>
<td>8.504</td>
<td>8.492</td>
</tr>
<tr>
<td></td>
<td>8.484</td>
<td>8.44</td>
</tr>
<tr>
<td></td>
<td>8.44</td>
<td>8.484</td>
</tr>
<tr>
<td>vmware-vmx</td>
<td>193.968</td>
<td>211.488</td>
</tr>
<tr>
<td></td>
<td>218.896</td>
<td>200.8608</td>
</tr>
<tr>
<td>free command</td>
<td>185.796</td>
<td>185.74</td>
</tr>
<tr>
<td></td>
<td>185.74</td>
<td>185.74</td>
</tr>
<tr>
<td></td>
<td>185.74</td>
<td>185.74</td>
</tr>
<tr>
<td>Overhead</td>
<td>76.372</td>
<td>77.712</td>
</tr>
<tr>
<td></td>
<td>102.838</td>
<td>102.784</td>
</tr>
<tr>
<td></td>
<td>87.0052</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{Overhead} = \sum(\text{VMWare processes}) - \text{output of free}
\]

Figure 3: coLinux Memory Overhead

<table>
<thead>
<tr>
<th>Process</th>
<th>Trials</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>colinbegin</td>
<td>1.428</td>
<td>1.428</td>
</tr>
<tr>
<td></td>
<td>1.428</td>
<td>1.428</td>
</tr>
<tr>
<td></td>
<td>1.428</td>
<td>1.428</td>
</tr>
<tr>
<td>colinux-bridged-net-daemon</td>
<td>7.684</td>
<td>7.688</td>
</tr>
<tr>
<td></td>
<td>7.688</td>
<td>7.692</td>
</tr>
<tr>
<td></td>
<td>7.692</td>
<td>7.7</td>
</tr>
<tr>
<td></td>
<td>7.6904</td>
<td></td>
</tr>
<tr>
<td>colinux-console-fltk</td>
<td>4.712</td>
<td>4.748</td>
</tr>
<tr>
<td></td>
<td>4.692</td>
<td>4.696</td>
</tr>
<tr>
<td></td>
<td>4.696</td>
<td>4.704</td>
</tr>
<tr>
<td></td>
<td>4.704</td>
<td>4.7104</td>
</tr>
<tr>
<td>colinux-daemon</td>
<td>4.756</td>
<td>4.728</td>
</tr>
<tr>
<td></td>
<td>4.756</td>
<td>4.756</td>
</tr>
<tr>
<td></td>
<td>4.756</td>
<td>4.744</td>
</tr>
<tr>
<td>free RAM before VM startup</td>
<td>614.164</td>
<td>612.648</td>
</tr>
<tr>
<td></td>
<td>606.184</td>
<td>614.832</td>
</tr>
<tr>
<td></td>
<td>612.4456</td>
<td></td>
</tr>
<tr>
<td>free RAM after VM startup</td>
<td>476.244</td>
<td>487.5</td>
</tr>
<tr>
<td></td>
<td>474.552</td>
<td>461</td>
</tr>
<tr>
<td></td>
<td>461</td>
<td>474.132</td>
</tr>
<tr>
<td></td>
<td>474.132</td>
<td>474.6856</td>
</tr>
<tr>
<td>free command</td>
<td>55.416</td>
<td>55.8</td>
</tr>
<tr>
<td></td>
<td>55.352</td>
<td>55.16</td>
</tr>
<tr>
<td></td>
<td>55.16</td>
<td>55.5568</td>
</tr>
<tr>
<td>Overhead</td>
<td>101.084</td>
<td>102.324</td>
</tr>
<tr>
<td></td>
<td>87.94</td>
<td>108.404</td>
</tr>
<tr>
<td></td>
<td>104.128</td>
<td>104.128</td>
</tr>
<tr>
<td></td>
<td>100.776</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{Overhead} = \sum(\text{coLinux processes}) + (\text{Start RAM} - \text{Final RAM}) - \text{free}
\]

because Windows does not monitor coLinux RAM usage.
For Figures 2 and 3, all values in megabytes as reported by Windows XP taskman. The virtual machine contained a Fedora Core 5 installation booting to run level 3 (multiuser console with networking; no GUI). These values reflect the total memory utilization required to launch the virtual machine, less the actual memory utilization of the virtual machine (reported by the free command).

Figure 4: Potential Yield

<table>
<thead>
<tr>
<th>Class</th>
<th>Raw Class</th>
<th>Raw Total</th>
<th>Raw % HPC</th>
<th>Virtualized Class</th>
<th>Virtualized Total</th>
<th>Virtualized % HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classrooms</td>
<td>176</td>
<td>352</td>
<td>4.10%</td>
<td>281.6</td>
<td>3.28%</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td>369</td>
<td>738</td>
<td>8.59%</td>
<td>590.4</td>
<td>6.87%</td>
<td></td>
</tr>
<tr>
<td>Labs + Classrooms</td>
<td>545</td>
<td>1090</td>
<td>12.69%</td>
<td>872</td>
<td>10.15%</td>
<td></td>
</tr>
<tr>
<td>ITS campus-wide</td>
<td>1000</td>
<td>2000</td>
<td>23.28%</td>
<td>1600</td>
<td>18.63%</td>
<td></td>
</tr>
<tr>
<td>ITS + testing</td>
<td>1400</td>
<td>2800</td>
<td>32.60%</td>
<td>2240</td>
<td>26.08%</td>
<td></td>
</tr>
<tr>
<td>Earliest estimations</td>
<td>2000</td>
<td>4000</td>
<td>46.57%</td>
<td>3200</td>
<td>37.25%</td>
<td></td>
</tr>
</tbody>
</table>

The term % HPC indicates a percentage of the current campus HPC computing power, namely 8.59 Tflops.

Listing 1: Santaka, 13 Nov 2006

mnoel@santaka:~> mshow

active jobs------------------------
JOBID  USERNAME  STATE  PROC  REMAINING  STARTTIME
1 active job 30 of 32 processors in use by local jobs (93.75%)
1 of 1 nodes active (100.00%)

eligible jobs----------------------
JOBID  USERNAME  STATE  PROC  WCLIMIT  QUEUETIME

0 eligible jobs

blocked jobs-----------------------
JOBID  USERNAME  STATE  PROC  WCLIMIT  QUEUETIME
656    dorband   Idle  30  1:00:00:00  Sun Nov 12 19:48:47
657    dorband   Idle  30  1:00:00:00  Sun Nov 12 19:48:50
665    naclaren  BatchHold 1  00:20:00  Mon Nov 13 13:52:42
3 blocked jobs

Total jobs: 4

mnoel@santaka:~>> qstat -a

santaka1:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>S Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10
Listing 2: Santaka, 1 Dec 2006

mnoel@santaka1:\> mshow

active jobs------------------------
JOBID  USERNAME  STATE  PROC  REMAINING  STARTTIME
701   dorband   Running  30  3:51:21  Thu Nov 30 15:00:23
1 active job 30 of 32 processors in use by local jobs (93.75%)
1 of 1 nodes active (100.00%)

eligible jobs----------------------
JOBID  USERNAME  STATE  PROC  WCLIMIT  QUEUETIME
0 eligible jobs

blocked jobs-----------------------
JOBID  USERNAME  STATE  PROC  WCLIMIT  QUEUETIME
0 blocked jobs

Total job: 1

Listing 3: SuperMike

[mnoel@mike3 mnoel]\$ mshow

active jobs------------------------
JOBID  USERNAME  STATE  PROC  REMAINING  STARTTIME
87010  rnella1  Running  16  4:22:07:24  Mon Nov 13 13:32:43
87012  rnella1  Running  16  4:22:07:24  Mon Nov 13 13:32:43
87224  dorband  Running  64  4:16:53:18  Mon Nov 13 08:18:37
87227  dorband  Running  64  4:16:53:18  Mon Nov 13 08:18:37
87266  lacramio  Running  8  3:20:58:23  Mon Nov 13 06:40:22
87225  lacramio  Running  8  3:18:15:03  Mon Nov 13 06:40:22
87214  lacramio  Running  8  3:02:49:56  Sun Nov 12 14:15:15
87046  sroy   Running  64  3:23:20:43  Sat Nov 11 14:46:02
87201  ade Running  64  2:17:12:15  Mon Nov 13 08:37:34
87132  pdul Running  2  1:21:54:41  Fri Nov 10 13:20:00
87133  pdul Running  2  1:21:54:41  Fri Nov 10 13:20:00
87295  ryoji Running  96  1:21:50:47  Mon Nov 13 13:16:06
87248  ryoji Running  64  1:17:12:15  Mon Nov 13 08:37:34
87287  ou Running  16  1:07:09:31  Mon Nov 13 10:34:50
87116  lacramio Running  12  18:08:42  Fri Nov 10 05:34:01
19 active jobs 728 of 916 processors in use by local jobs (79.48%)
364 of 458 nodes active (79.48%)

eligible jobs----------------------
JOBID  USERNAME  STATE  PROC  WCLIMIT  QUEUETIME
86432  eschnett  Idle  512  2:00:00  Sun Oct 29 14:14:38
<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROC</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>87298</td>
<td>bourdin</td>
<td>Idle</td>
<td>192</td>
<td>2:00:00:00</td>
<td>Mon Nov 13 13:22:43</td>
</tr>
<tr>
<td>87228</td>
<td>ipark</td>
<td>Idle</td>
<td>64</td>
<td>2:00:00:00</td>
<td>Sun Nov 12 02:45:30</td>
</tr>
<tr>
<td>87149</td>
<td>ade</td>
<td>Idle</td>
<td>64</td>
<td>5:00:00:00</td>
<td>Fri Nov 10 11:03:57</td>
</tr>
<tr>
<td>87230</td>
<td>eliu</td>
<td>Idle</td>
<td>40</td>
<td>5:00:00:00</td>
<td>Sun Nov 12 08:22:43</td>
</tr>
<tr>
<td>87231</td>
<td>eliu</td>
<td>Idle</td>
<td>40</td>
<td>5:00:00:00</td>
<td>Sun Nov 12 08:23:54</td>
</tr>
<tr>
<td>87232</td>
<td>eliu</td>
<td>Idle</td>
<td>40</td>
<td>5:00:00:00</td>
<td>Sun Nov 12 08:25:01</td>
</tr>
<tr>
<td>87290</td>
<td>ade</td>
<td>Idle</td>
<td>64</td>
<td>5:00:00:00</td>
<td>Mon Nov 13 12:36:51</td>
</tr>
<tr>
<td>87297</td>
<td>ade</td>
<td>Idle</td>
<td>64</td>
<td>5:00:00:00</td>
<td>Mon Nov 13 13:15:15</td>
</tr>
<tr>
<td>87013</td>
<td>rnella1</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Wed Nov 8 14:33:15</td>
</tr>
<tr>
<td>87014</td>
<td>rnella1</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Wed Nov 8 14:34:58</td>
</tr>
<tr>
<td>87015</td>
<td>rnella1</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Wed Nov 8 14:36:37</td>
</tr>
<tr>
<td>87017</td>
<td>rnella1</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Wed Nov 8 14:39:43</td>
</tr>
<tr>
<td>87018</td>
<td>rnella1</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Wed Nov 8 14:40:54</td>
</tr>
<tr>
<td>87218</td>
<td>skneasler</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Sat Nov 11 13:20:19</td>
</tr>
<tr>
<td>87219</td>
<td>skneasler</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Sat Nov 11 13:20:19</td>
</tr>
<tr>
<td>87220</td>
<td>skneasler</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Sat Nov 11 13:25:40</td>
</tr>
<tr>
<td>87221</td>
<td>skneasler</td>
<td>Idle</td>
<td>16</td>
<td>5:00:00:00</td>
<td>Sat Nov 11 13:26:02</td>
</tr>
</tbody>
</table>

**Blocked jobs**

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USERNAME</th>
<th>STATE</th>
<th>PROC</th>
<th>WCLIMIT</th>
<th>QUEUETIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>87004</td>
<td>pbasuc1</td>
<td>BatchHold</td>
<td>376</td>
<td>1:00:00</td>
<td>Wed Nov 8 14:04:13</td>
</tr>
</tbody>
</table>

1 blocked job

Total jobs: 38