# Rapid Parallel Systems Deployment: Techniques for Overnight Clustering

*Donna Cumberland, Randy Herban, Rick Irvine, Michael Shuey, and Mathieu Luisier* – Purdue University

## ABSTRACT

Automated system deployment frameworks and configuration management systems have been in wide use for a number of years. However, due to increasing pressures to maintain high availability, coupled with the price effects of commodity servers, administrators may be required to deploy large numbers of systems in shorter time frames than is normally possible with available staff. In this paper, we describe a straightforward procedure using commonly-available infrastructure to enable rapid simultaneous deployment of hundreds of machines by temporary staff. As an example of the efficacy of this approach, we present a case study in rapid systems deployment at Purdue University. On May 5th, we deployed Purdue's "Steele" cluster, installing over 500 compute nodes in a single business day.

## Introduction

Changing system requirements, additional projects, and life-cycle system replacements all result in deployment of new machines in the data center. Given the current popularity of grid computing and software-as-a-service, and the proliferation of virtual servers within the enterprise, the number of new systems will likely continue to rise.

Modern services, particularly high performance computing centers, often deploy clusters of hundreds, if not thousands, of individual machines. While several custom tool sets exist to aid this sort of environment [1, 2], these tools are usually heavily adapted to their task and may not readily integrate into existing configuration management tools. Also, these tools are often built around a particular market segment, such as the high performance cluster community. As such, they may not be well suited to other uses (e.g., dedicated application servers, web farms, digital rendering clusters, etc.)

Several configuration management tools exist to allow few administrators to manage large collections of machines [3, 4, 5]. These can easily be used to encompass both large-scale system deployments (clusters, render farms, etc.), but do not encompass the actual operating system installation. Other tools must be employed to actually deploy the system, and load the configuration management software.

Most modern operating systems provide some means for repeatable, automatic software installation [6, 7]. These tools can, in some cases, be directed to prepare configuration management software on the new host. However, they assume some external method is used to identify the system being installed – either via a pre-configured service, like DHCP, or by manual network name and address assignment. These are generally quite labor intensive.

By combining the available component tools, we describe a server deployment approach more flexible than current purpose-built tools. With the addition of IPVS [8], an IP-level load balancing service, such an approach is readily able to handle the peak load of hundreds, if not thousands, of new servers being deployed simultaneously. This capability greatly reduces software deployment time, and (with the addition of temporary labor to assist with physical tasks) can enable the installation of large systems in surprisingly little time.

For the remainder of this paper, we present an insight into our motivation behind taking this deployment approach, followed by a more detailed discussion of the difficulties involved in a highly parallel deployment and our implementation to address these. As evidence of the success of this approach, we present a case study in rapid cluster deployment – the recent installation of Purdue's "Steele" cluster. We conclude with the lessons learned from this deployment exercise, and some indications of possible scaling limitations moving forward.

## Motivation

Purdue University is home to several active scientific research communities, many of which make use of high-performance compute clusters maintained in central campus data centers. As with many institutions, data center space is at a premium. New systems cannot be deployed without first removing older equipment. In most cases, the out-bound older equipment is still serving a research group the day it leaves. These users are effectively left without service until a new system can be deployed.

In the spring of 2008, Purdue faculty, staff, and administrators began the design and purchase of a 806-node compute cluster (more fully described below).

Data center facilities limitations forced the removal of three existing clusters, with a combined total of over 900 compute nodes. Equipment removal alone would take approximately one week, and required all available staff with familiarity with the facility. Only a small portion of the new cluster could be assembled before the existing equipment's removal.

To reduce the overall impact on Purdue's research community, we developed the techniques described to deploy the majority of the nodes in a single day. The success of this endeavor is covered in more detail in the case study section.

## Infrastructure Components

Rapid server deployment depends on automating the installation process as much as possible. Any amount of variation between servers requires manual intervention, and this must be minimized to maximize system administrators' productivity.

The server deployment process can be seen as a three-phase process: Installation of the base system software, identification of the machine (including assignment of network addresses), and merging the new machine into a full configuration management system.

### Base System Software

Modern gigabit networks can deliver data in excess of 100 MB/sec, rivaling hard disks and exceeding the speed of any other installation media. To improve deployment times and eliminate administrator time spent swapping media, using a network-based install method is essential.

As mentioned above, most modern operating systems have some sort of network-based installation system. With firmware-level network boot services (such as PXE [9]) on most new hardware, using this is fairly straightforward. Multiple machines can be simultaneously installed from a dedicated PXE configuration, to invoke systems like RedHat's KickStart or a Solaris JumpStart as appropriate.[1] For the remainder of this discussion we will focus on extensions to RedHat's KickStart system, which is in wide use in our environment.

The PXE environment does have one large drawback: in order to uniquely identify each machine being deployed, an administrator must either populate a DHCP server with the MAC address of each machine in question (along with each machine's proper IP address), or separate installation configuration files for each system. While scriptable, selecting the proper configuration for each host is often extremely time-consuming. For this reason, we have chosen to install all deployed systems with a common operating system load at first boot, using temporary network addresses. Network address assignment, and proper machine identification, can be done as a second pass through the mostly-installed systems (as described in the Machine Identification section).

[1]The PXE boot system can easily be extended to provide for more than a mass-deployment mechanism. In our environment, individual servers being installed are brought to a PXE menu where administrators can select from a number of installation and testing options including memtest, Knoppix [10], and dedicated entries for each operating system and version we have on site.

### KickStart Configuration

RedHat's KickStart system is a means to automate the installation and initial configuration of RedHat's Enterprise Linux operating system. Several types of media (e.g., CD-ROM, HTTP, NFS) are supported to deliver system software. In all cases, the install process follows directives listed in a file, `ks.cfg`. The `ks.cfg` file lists key system configuration values (such as the initial root password, timezone configuration, etc.), any network parameters needed, the software packages to be installed from the selected media, and (optionally) a post-install script to execute before rebooting from the OS installer.

In our environment, all server deployment takes place over the network. Our PXE boot environment will cause RedHat's installer kernel (and initial filesystem) to be loaded via TFTP. Linux command-line options, as well as configuration directives in the `ks.cfg` configuration file, are used to ensure that all systems use a random DHCP-provided address during the install process. This lets us use a single configuration to serve hundreds of clients with no customization. At the end of the OS install process, systems are moved to a fixed, production IP address (to correspond with their location in an equipment rack). This process is explained in detail in the machine identification section.

All system software is provided by a local HTTP server, and all RedHat packages to be installed on a server are listed in the `ks.cfg` file. Since HTTP is a unicast service, this does pose a bottleneck for bulk system deployments. However, as this data is only read (never written) during the installation process, it can easily be replicated. Multiple HTTP servers can be clustered to increase the overall bandwidth, provided they are aggregated behind a single network name (to simplify configuration and reduce the amount of customization in `ks.cfg`). Our preferred choice for this is an IPVS-based load balancing cluster (described further in the IPVS discussion, though in practice even a simple DNS record pointing to multiple addresses would suffice.

Once RedHat's KickStart system has loaded and configured the base operating system, the post-install script contained in the `ks.cfg` file is executed. We use this script to properly identify the machine and set its final network address, as described below, and install critical pieces of our cfengine configuration (as described in the configuration management section.

### Machine Identification

Without expending a great deal of effort to carefully catalog, place, and install each machine in a large group, a machine's final network address is often not apparent until it has been physically installed. A compute node's identity is often based on its location in an equipment rack – a property that is very difficult to discern from a boxed machine on the back of a truck.

Some systems (e.g., IBM's SP-2 [11]) are able to discern their identity by automatically detecting their location in a specially-wired management network. With additional scripting, this can be translated into node names, network addresses, and useful configuration data. However, this method is usually not an option when using today's low-priced commodity hardware.

Other systems (e.g., OSCAR [1]) use a mechanism to collect DHCP requests as machines are booted, and thus identify systems in order. These systems require machines to be booted serially, though, and prevent increasing the deployment parallelism to the level we would prefer. Instead, we have devised a simple mechanism to touch each machine in sequence, to have each system confirm its proper network address.

In our environment, we extended the kickstart configuration file to include an infinite sleep-wait-probe cycle in the postinstall section before configuring their production IP address. This allows us to start the installation process on many nodes simultaneously without regard to their final naming. The loop will only break if the machine sees the insertion of a USB thumbdrive, as shown in Listing 1, a kickstart script excerpt.

With this in place, an administrator can assign network addresses to a series of machines by simply making a pass through the entire group, inserting a USB thumbdrive into each machine. The thumbdrive itself is not important – we are using the act of inserting this USB device as a means of physically identifying one machine out of hundreds of identical systems. Other simple methods of quick, manual identification (e.g., a key press, CD insertion, etc.) would be equivalent.

Once a machine has been physically identified, it contacts the web server and downloads its proper network configuration. Obviously, the CGI program get_ip.rh.cgi must return a valid network configuration file, then increment the IP to the next production IP address in sequence.

### Configuration Management

Many configuration management systems are in widespread use. Assuming that all configuration information (beyond basic system parameters, such as partition layout and operating system version) has been expressed through a site's preferred configuration system, the remainder of a system's deployment should be a very simple exercise.

Purdue's high-performance computing environment uses cfengine [12] heavily. The final step in our customized kickstart configuration is to download a custom shell script to initialize cfengine, and run it at first boot. The script includes an in-line shar archive containing the necessary binaries (and key configuration files) for that architecture. This script is placed at the end of the boot sequence for the new host.

When the newly deployed system first boots, the script will run cfengine twice (first with special options to mitigate site-specific dependencies and ordering problems, then with default options). This configures the system, installs any missing site-specific software, ensures various cfengine-related processes are started at boot time, and applies any outstanding security errata not a part of the initial OS install. Finally, the script removes itself from the boot process and reboots the host. Once rebooted, the system is a production-ready server.

## Limitations/Acceleration

### IPVS

We felt the throughput capacity of our file serving infrastructure for both kickstart and cfengine initialization needed to be upgraded for massively parallel deployments. Our existing machine for these functions

```
%post
cd /tmp
/sbin/modprobe usb-storage
while (true); do
   if [ dmesg | tail -n 3 | grep 'USB Mass Storage device found' ]; then
      wget http://install-server/cgi-bin/get_ip.rh.cgi
      cp /tmp/get_ip.rh.cgi /etc/sysconfig/network-scripts/ifcfg-eth0
      #
      # configure any other network-related bits here
      #
      ifdown eth0
      ifup eth0
      break
   fi
   sleep 3
   echo "Still waiting for USB thumbdrive..."
done
```

**Listing 1**: Thumbdrive detection.

was an older dual-processor system, and we were concerned that disk I/O limitations of its aging RAID array would be exceeded during the installation.

By utilizing a load balancing cluster with a round robin distribution algorithm, we segmented the demand down to equal parts for each real server. By placing the cluster behind a single IP address, we avoided having to change any other part of our automated installation infrastructure. We chose to use Linux's IPVS kernel module [8] and a set of freely available tools [13] to create this load balanced cluster and to give it high availability as well.

Purdue's configuration required additional machines for this purpose, so we re-purposed a few existing LDAP authentication servers. As such, in addition to the kickstart and cfengine file sharing responsibilities, the IPVS cluster also now handles LDAP authentication. We ended up with two cluster front ends and four real servers, all newer dual-processor dual-core servers. This gave us theoretically 4 gigabits per second of file transfer rates for the cluster node installations and also gave each node in the cluster only a quarter of the load.

We could have simply used a single larger machine. However, by leveraging the existing LDAP authentication hardware (which was not being overly taxed) and using open source software, our solution required no monetary investment. Since all IPVS-related configuration changes were merged into cfengine while building out the initial server setup, the IPVS cluster itself was rapidly deployed. We simply treated it like any other cluster – PXE boot a new system, automatically install the OS and run cfengine, then manually add it to the IPVS cluster routing table on the front ends.

In addition, if we had determined during a mass deployment that the load was too high for the IPVS cluster, we had the option of utilizing several of the new cluster machines being deployed as additional kickstart/cfengine file transfer nodes. We could have simply installed them as generic servers, labeled them as IPVS real servers in cfengine, and set them to work. The initial cfengine on an IPVS real server does take a little while due to the large repository of install data that must be copied over, but it still was quick enough to have been an option during the install day exercise.

This solution to our increased file transfer needs was cheap, easy, and a very reliable means to scale out our deployment bandwidth.

### Squid Cache and RH Proxy

Our compute cluster nodes generally use Redhat's Enterprise Linux [22]. Purdue has a university-wide Redhat Network [23] proxy server that handles registering new OS installs and caches updated software downloads. The existing server was an older dual-processor system, with relatively slow disk array (as compared to current technology). We've seen this system become overloaded when any Purdue IT group would attempt to update more than a couple dozen machines at once.

For the installation day, we felt this system needed to be upgraded as well. We installed a new dual-processor, dual-core server with twice the memory of the old machine, from four to eight gigabytes, in order to handle the heavier I/O traffic. Linux makes aggressive use of unallocated memory to cache storage I/O, so we were hoping this system would yield significant performance improvements.

We briefly looked at putting the squid [14] web cache portion of the RHN proxy server behind the IPVS cluster, but it appeared that the squid server needed to be local to the registered RHN proxy server. In additional, the actual RHN proxy software was licensed for a single machine. As such could not legally be put on each real server, it had to run on a single machine. After testing, we determined that additional proxy servers, or further work integrating the RHN proxy server with our IPVS cluster, would not be necessary for the size of deployments we typically encounter. However, this may be a limiting factor for much larger installations (with thousands of machines).

The larger proxy machine was more than capable of handling the load generated by installing the sizable "Steele" cluster at Purdue. Had it not been able to, we would have simply not run so many initial operating system installs in parallel. We wanted to have every node installed and running by the end of the day, but we were realistic in expecting the possibility of some machines finishing their installs and cfengine configurations overnight. As it turns out, the proxy server never came close to heavy load. We estimate that the system could have easily handled two to three times more load.

### Subdivide Networks, Conquer DHCP

One shortcoming of this method is the serial nature of using the USB thumbdrive to number machines. It can quickly become the bottleneck and increasing the number of machines installed via this method only worsens the problem. One method to parallelize is to divide the machines up onto multiple network subnets and give them separate DHCP/PXE entries. Each entry can then use a separate source to obtain the IP address, and can have its addresses assigned independently. During previous large system deployments, we have spanned three subnets and have been able to identify three machines at once. We see no reason why this could not be trivially increased for larger installs, allowing for wider deployment parallelism.

### TFTP and PXE Performance

After the infrastructure improvements described above, TFTP remains a single point of contention. Mass machine deployment efforts may direct hundreds of clients to a single TFTP server as each machine loads its operating system installer. In practice, however, this service sees insignificant load – even during our large deployments.

The operating system installer we use, provided with RedHat Enterprise Linux, consists of 6.4 MB of software. On a gigabit ethernet network, that amount of data takes less than a second to transfer – less time than it takes to physically turn on a new machine. If several dozen systems were to start the PXE process at the exact same second, it is conceivable that we might have some contention at the TFTP server's network port. For our purposes, though, that was deemed an acceptable risk.

## Case Study: Purdue's "Steele" Cluster

Much of our work to streamline, parallelize, and accelerate system deployment came about as a result of the diminishing quantity of available data center capacity, coupled with the rising pressure to effectively use the remaining space as quickly as possible (as mentioned above). In the spring of 2008, we saw an opportunity to push the infrastructural improvements we describe to the limit of our facilities' capacity.

As mentioned above, Purdue's "Steele" cluster could not be supported in our facilities without the removal of a large number of existing production computing systems. To better accommodate our research community, we opted to deploy as many systems as possible ahead of time, leaving the remainder to be installed in one large batch on May 5th, 2008. To provide more labor during the install day, we recruited volunteers from positions throughout the university, none of whom were familiar with the deployment process. The details behind this installation day, and its successful reception by some of our researchers, are described in the following sections.

### Machine Description

The "Steele" cluster consists of 806 dual-processor, quad-core compute nodes, in four different memory and network configurations. Configurations break down as follows:

- 24 nodes, 32 GB memory and both gigabit ethernet and Infiniband network connections
- 41 nodes, 32 GB memory and only gigabit ethernet networking
- 180 nodes, 16 GB memory and both gigabit ethernet and Infiniband network connections
- 561 nodes, 16 GB memory and only gigabit ethernet networking

The three smaller configurations are housed in one room of Purdue's research datacenter, while the largest configuration (with 561 nodes total) is housed in a separate room.

The configuration is connected with a single large ethernet switch, with over 600 available ports of gigabit ethernet. Other configurations are connected to smaller gigabit ethernet switches, which then link to the larger switch via 10 gigabit ethernet connections.

### Pre-Deployment Steps

To lessen the service impact to University researchers, we opted to install as many compute systems as possible leading up to the May 5th install day. Unfortunately, due to facilities limitations, this amounted to about 12% of the system's total capacity. In addition, we were able to unbox, rack, and install a number of systems (roughly another 12% of the compute nodes) prior to the general installation – though these systems were not fully deployed, and in many cases had little or no power or network wiring.

In the week prior to the main install day, large sections of our facilities were cleared. Network components were installed and configured, equipment racks and electrical connections were set in place, and ethernet wiring was run from most switch gear to patch panels in each equipment rack.

### The Human Factor

Simply put, there were too many systems for our usual systems administration team to handle in a single business day. Without additional help, our lack of manpower alone would force us to prolong this deployment over at least a week, possibly two. However, with the deployment procedure fully automated, we were able to solicit assistance from the rest of the University.

In addition to the obvious need for aid in wiring systems and initiating PXE-based OS installs, we also sought volunteers for a number of easily overlooked, but practically indispensable tasks:

- Truck drivers – needed to drive trucks to and from our storage facility to deliver machines to the data center. These volunteers also helped unload machines from the trucks and move them to the unboxing area.
- Unboxers – responsible for unboxing machines and the rack mount equipment, and loading them on the carts to send into the data center.
- Machine movers – responsible for transporting equipment from the nearby delivery point into the data center.
- Recycling team/clean-up crew – responsible for sorting all materials into appropriate recycling bins and cleaning shipping areas and the data center of any debris.

As word of this initiative spread through campus, and the number of volunteers rose, we also added two other unlikely groups of volunteers:

- Check-in assistants – responsible for handing out nametags and directing volunteers to areas in need of assistance.
- Food area help – responsible for serving breakfast and lunch for volunteers, as well as clean-up.

In all, some 120 volunteers helped deploy the remainder of the cluster.

## Waste Removal

Every single compute node was provided by the vendor in an individual box. In addition to the server, and rack mounting equipment, each box also contained product manuals, a CD of firmware and diagnostic programs, a power cord, and two cable management trays. None of this extra material was used.[2] Given the number of systems involved in this cluster, managing the waste was a substantial task.

We recycled or reused nearly all of the extra materials we processed at the install site. Power cords and cable management trays were returned to the vendor for reuse. The shipping pallets were similarly collected, for reuse by Purdue's shipping and receiving group.

We arranged for recycling services to remove the large amounts of foam and cardboard generated as we unboxed these machines. This was all planned in advance and recycling services placed dumpsters at our location before the event started. Recycling services removed full dumpsters of foam and cardboard several times during our install. In total, some 660 lbs of packing foam, 6000 lbs of cardboard, as well as additional manuals and CDs, were recycled during the main installation.

[2]The vendor-supplied power cord was approximately 6 feet (2 meters) in length. Its use, or the use of the provided cable trays, would have caused airflow obstructions and would have presented a thermal hazard in our environment. However, it was more costly to have the vendor change the packing procedure for these systems than to just purchase shorter power cords ourselves.

## Infrastructure Performance

May 5th installation activities started at 8:00 am, Eastern Daylight Time (EDT). By 11:45 AM, when a general lunch recess was called, only 80 compute systems remained to be mounted in equipment racks. Over two hundred systems had been deployed, the effects of which can be seen in Figure 1. Following a one-hour break, work resumed. By 1:30 pm, all systems had been racked and wired. By 3:00 pm, all systems had at least begun the OS deployment process, requiring no further human intervention. At that time 750 systems (out of 806 total) were reporting as available, and 1400 user-submitted batch jobs had already begun running on the compute cluster.

By noon the following day, all but six machines had been made available to the University research community. By 4:00 pm on May 6th, all available systems had been connected to the Open Science Grid [15], and had begun processing additional international work.

By moving the bulk of the software installs into the kickstart configuration (which was served by the

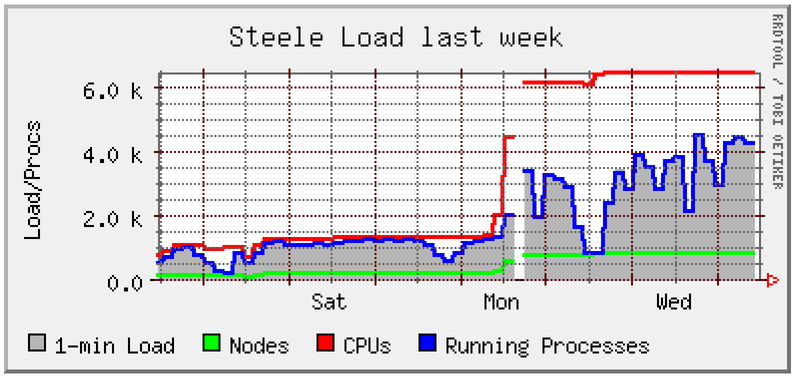
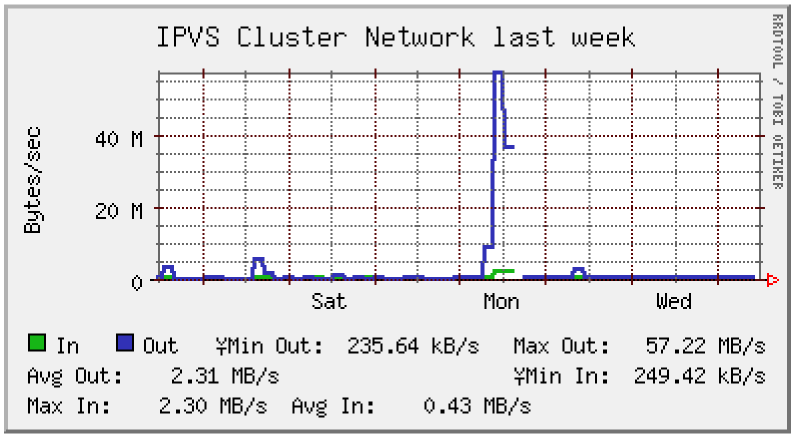


**Figure 1**: "Steele" cluster during deployment.



```
¥Min Out:  235.64 kB/s   Max Out:   57.22 MB/s
Avg Out:    2.31 MB/s                       ¥Min In:  249.42 kB/s
Max In:    2.30 MB/s   Avg In:     0.43 MB/s
```

**Figure 2**: IPVS cluster network use.

IPVS cluster), our RedHat proxy server wasn't overly strained – its load average ranged from 0.1-3.0 throughout the day. For larger deployments, however, we may consider either increasing the install time or, more likely, adding additional proxy servers.

As seen in Figure 2, our IPVS cluster was sustaining data rates of about 57 MB/sec during the installation. While we did see traffic bursts as high as 390 MB/sec, these were very short-lived and did not adversely impact the deployment. We feel this infrastructure could sustain simultaneous installation to two to four times the number of machines, though if it did become a serious bottleneck we could simply add additional IPVS backend servers to the cluster.

We also knew a fair number of machines might be dead on arrival. We merely skipped those IP addresses in the deployment procedure, by running the CGI program to increment to the next available address. By the end of the day 56 had been skipped, mostly due to minor wiring problems or having been passed over for the command to boot via PXE, and never beginning the install process. These were easily corrected the following morning.

We did encounter one unexpected failure: the cluster monitoring suite we use, Ganglia [16], was initially unable to cope with the amount of memory in the "Steele" cluster. This issue had since been corrected in a later software release, so we elected to upgrade immediately to the newest version. As a side effect, cluster monitoring was briefly unavailable, as evidenced by the brief discontinuities in Figures 1 and 2.

## Customer Acceptance and Benchmarking

One of the key groups behind the acquisition of the "Steele" cluster is Purdue's Network for Computational Nanotechnology (NCN), a community of researchers focused on simulation of nano-scale semiconductor devices. We began to study the performance of the fledgling cluster for this type of simulation the evening of May 5th, a matter of hours after the last of the compute nodes had been unboxed.

To evaluate the performances of the "Steele" cluster, a benchmark example was run on 16 to 6272 cores. For that purpose we used the quantum-mechanical nanoelectronic device simulator OMEN [17, 18]. This massively parallel software computes the current characteristics of nanotransistors as function of the input source, drain, and gate voltages. It has four levels of parallelism[3] and in its most inner loop two eigenvalue problems and a sparse linear system are solved. In a typical device simulation this happens more than hundred thousand times enabling the use of large computer clusters as "Steele."

Figure 3 shows the scaling properties of OMEN on the "Steele" cluster for the simulation of a silicon double-gate ultra-thin-body field-effect transistor designed according to the ITRS specifications for the 22nm technology node [19]. Only one bias point is computed with different parallelism schemes. The blue curve with crosses represents the case where the parallelization of OMEN is achieved with MPI only an no domain decomposition is applied to the transistor structure. Hence, the momentum and energy points are parallelized. To obtain the green curve with triangles we decompose the simulation domain on two cores with the help of a distributed memory sparse linear solver. Both these approaches require to launch as many MPI tasks as available cores.

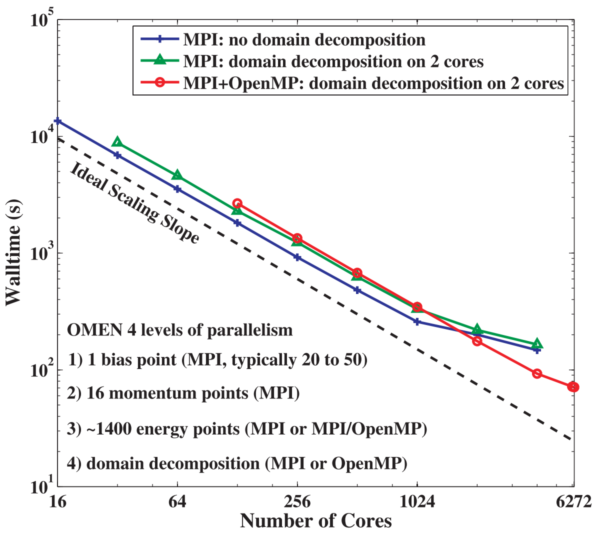


**Figure 3**: Application scaling on the "Steele" cluster.

An alternative is the implementation of a MPI-OpenMP hybrid where a single MPI task is started per node and eight threads are created within each node. The red curve with circles in Figure 3 illustrates this approach. Domain decomposition is realized on two cores using a shared memory sparse linear solver. For example, 784 MPI tasks are necessary to obtain the results on 6,272 cores. Note that the intra-node communication implied by the hybrid model helps reducing the overhead caused by MPI collective operations like MPI_Allreduce. Consequently, the red curve with circles exhibit almost no saturation of its scaling properties as compared to the two others.

Following these benchmarks, every research community involved in the acquisition of the "Steele" cluster resumed their normal usage patterns, after a total service outage of under nine days.[4] By May 8th, four days after 88% of the system was deployed, over four thousand batch jobs have been submitted to the new system. This has been both the largest single computer system in Purdue's history and the shortest delivery time for a high-performance scientific resource the University has seen.

[3]OMEN parallelizes across bias points, momentum points, energy points, and decomposition of the simulation domain.

[4]This counts both the time needed to remove older compute resources and to deploy all components of the new system.

## Further Work

While we originally developed this deployment process to deploy large cluster systems, we have since begun to use it for nearly all system installations within our environment. During normal operation, the PXE boot system will bring up a menu by default, containing a list of all our common machine configurations (each of which load a custom ks.cfg configuration file into the appropriate OS installer). Several versions of RedHat Linux are supported, as is Debian Linux (through their FAI [20], or Fully Automatic Installation, system).

Our network deployment infrastructure is used daily to deploy individual machines. For such small quantities, we slightly modify our PXE configuration to provide the correct network configuration for the machine. A slight modification to our ks.cfg post-install script bypasses the USB-driven address assignment procedures if DHCP is not in use during system installation, so machines will proceed directly to cfengine configuration. Once this process is initiated, new servers are fully customized and ready for production work in approximately fifteen minutes.

The availability of this rapid deployment framework has also changed our response to problem mitigation. Rather than painstakingly correcting an installation issue on hundreds of systems (for example, changing the partition table on production compute servers), we merely schedule a brief downtime and reinstall the entire group of affected machines. We have reinstalled over 160 machines in under an hour using this process, and have been able to successfully return these systems to full service with a minimum of user-visible downtime.

## Conclusions

The "Steele" installation was a huge success. We had planned for a full day for all of the physical labor and were expecting the OS installation and configuration pieces to run overnight. Considering that 200 nodes were online and processing jobs by lunch, with the rest of the functional systems available for cluster users by 3:00 pm, it's safe to say we met our target business-day turnaround time for this deployment.

Of particular concern was the handling of the large amounts of recycling materials. We had arranged for empty dumpsters for recycled goods to be delivered several times throughout the day. Thanks to the number of people that showed up to help, loading empty boxes and other debris was trivial. However, this just serves to underscore the need for proper site planning and coordination (in addition to the technical measures necessary) to prepare for a rapid deployment of this scale.

We believe mass server deployments, with hundreds of servers deployed in under a business day, are very achievable. Scaling the backend infrastructure to support these deployments can be done at low cost, using commonly-available (and largely open-source) technology, as we have demonstrated. With equivalent preparation, other institutions should be able to see similar rates of system installation. Eventually, we see this practice moving from the realm of IT "stunt" [21] to an accepted business process.

## Acknowledgments

The authors would like to thank a number of individuals for their role in helping bring about the "Steele" cluster. The following people aided in designing, purchasing, transporting, installing, deploying, and organizing portions of this project: Hans Aagard, Samarth Agarwal, Lon Ahlen, Syed Ali, Billy Anderson, Chris Andrus, Bill Ansley, Kim Arnold, Laura Arns, Marty Barnes, Andrew Bean, Jeremy Benner, Larry Biehl, Bart Bishop, Brian Black, Dave Black, Carolyn Bogan, Darryl Borgonah, Kyle Bowen, Allen Braunsdorf, Mitch Breneman, Dean Brusnighan, Birgitte Brydso, Michael Budzik, Tom Bunton, Clyde Burghardt, Wayne Byarlay, John Campbell, Jere Carroll, Felix Chang Lee, Phil Cheeseman, Min Chen, Tammy Clark, Kevin Colby, Mark Copeland, Benjamin Cotton, Douglas Couch, Robby Crain, Rob Cumberland, Donna Cumberland, Charles DeLa, Kent DeLa Croix, Noah Diffenbaugh, Steve Dong, Randy Drake, Jason Dufair, Ed Dunn, Rudy Eigenman, David Eisert, Alex Elman, Ed Evans, Vicki Farnsworth, Shane Farnsworth, Greg Farr, Alan Farrester, Pat Finnegan, Mike Fisher, Greg Flint, Barbara Fossum, Joe Francisco, Bruce Fuller, Hugh Fultz, Sue Gibson, Michael Gillaspy, James Gonzales-Meisler, Peter Goreham, Thomas Hacker, Ben Haley, Andrew Hall, Dave Hancock, Mike Hardebeck, Steve Hare, Matthew Harvazinski, Jessica Harvey, Patricia Harvey, Amy Haston, Brian Heaton, Nathan Heck, Laura Helms, Randy Herban, Michael Hicks, Scott Hicks, Elizabett Hillery, Rita Hillman, Matthew Hodson, Geoffrey Holden, Steve Holmes, Ben Holmes, Ryu Hoon, Mark Houser, Andy Howard, Nelson Howell, Fengping Hu, Shaomin Hu, Ben Huao, Kay Hunt, Rick Irvine, Wen Jiang, Bill Jones, Jennifer Kapp, Julie Kercher-Updike, Rene Ketterer, David King, Alex Kingman, Gerhard Klimeck, Scott Ksander, Sravan Kumar Dheeraj Kapilavai, Bruce Lambert, Ronald Lehman, Pam Lewellen, Markus Lill, Matt Link, Bruce Loftis, Matt Luisier, Doug Magers, Rishabh Mall, Mike Marsh, Ashlie Martini, Jennifer Maxson, Gerry McCartney, urtis McGrew, Dwight McKay, Diana McKinney, Sean McLane, Michael McLennan, Cory Meadows, Saumitra Mehrotra, Carol Middleton, Pia Mikeal, Manuriti Minhas, Eric Mullen, Mike Myers, Max Myers, Mary Nemacheck, Norbert Neumeister, Ryan O'Neill, Mark Padgett, Hyun Park, Terry Patterson, Abhijeet Paul, Jeff Pherson, Jim Polley, Rajini Prabhu, Nirmal Prajapati, Rajib Rahman, Jeffrey Rattray, Byron Reed, Neil Reed, Meiqi Ren, Matthew Riehle, River Chuan Xiao, Jim

Robinson, Timothy Rogers, Brian Rose, Michele Rund, Chad Sailors, Juan Santos, Richard Schick, Jim Schmitz, Krysten Schneider, Jeff Schwab, Louis Scott, Bernie Seabolt, Dave Seamen, Jie Shen, Jason Sheridan, Corey Shields, Mike Shuey, Richard Simmons, Anna Sledd, Steve Sloan, Nicholas Smith, Randolph Smith, Preston Smith, Julie Smith, Carol Song, Carmen Springer, John Steele, Jamie Stevens, Jeff Stewart, Jimmy Stine, Dave Stogner, Claire Strodtbeck, Matt Sutherlin, Andrew Sydelko, Steve Tally, Floyd Taylor, Helen Terrell, Andy Thomas, Richard Thompson, Jared Thompson, Jenett Tillotson, Todd Turner, Kerry Tyler, David Umberger, Kristen Van Laere, Phyllis Veach, Greg Veldman, Joe Wade, Judy Wagner, Bill Walker, Richard Westerman, Joe White, Bill Whitson, Drue Whitworth, Guneshi Wickramaarachchi, Kent Wiesner, Chad Wilhelm, Ramon Williamson, Jackie Wilson, Greg Wilson, Nancy Wilson Head, Dan Winger, Bryan Worthington, Jon Wright, Dongbin Xiu, Haiying Xu, Alex Younts, Joel Zarate, and Lan Zhao.

## Bibliography

[1] Naughton, Thomas, Stephen L. Scott, Brian Barrett, Jeff Squires, Andrew Lumsdaine, and Yung-Chin Fang, "The Penguin in the Pail – OSCAR Cluster Installation Tool," *Proceedings of SCI'02: Invited Session – Commodity, High Performance Cluster Computing Technologies and Application*, 2002.

[2] Papadopoulos, Philip M., Mason J. Katz, and Greg Bruno, "NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters," *Concurrency and Computation: Practice and Experience*, Vol. 15, Num. 7-8, pp. 707-725, December, 2001.

[3] Burgess, Mark, "Cfengine: A Site Configuration Engine," *USENIX Computing Systems*, Vol. 8, Num. 3, 1995.

[4] Desai, Narayan, Rick Bradshaw, Joey Hagedorn, and Cory Lueninghoener, "Directing Change Using Bcfg2," *20th Large Installation System Administration Conference (LISA '06)*, pp. 215-220, December, 2006.

[5] *Puppet*, http://reductivelabs.com/projects/puppet/ .

[6] *Kickstart*, http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/System_Administration_Guide/Kickstart_Installations.html .

[7] K. Amorin, *Solaris Jumpstart Automated Installation*, http://www.amorin.org/professional/jumpstart.php .

[8] *IPVS*, http://www.linuxvirtualserver.org/software/ipvs.html .

[9] *Preboot Execution Environment (PXE) Specification*, http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf .

[10] *Knoppix Live Linux Filesystem on CD*, http://www.knopper.net/knoppix/index-en.html .

[11] Agerwala, T., J. L. Martin, J. H. Mirza, D. C. Sadler, D. M. Dias, and M. Snir, "SP2 System Architecture," *IBM Systems Journal*, Vol. 34, Num. 2, 1995.

[12] *cfengine*, http://www.cfengine.org/ .

[13] *Linux HA (HighAvailability)*, http://www.linux-ha.org/ .

[14] *Squid web proxy/cache*, http://www.squid-cache.org/ .

[15] *Open Science Grid*, http://www.opensciencegrid.org/ .

[16] *Ganglia Monitoring System*, http://ganglia.info/ .

[17] Luisier, Mathieu and Gerhard Klimeck, "OMEN: An Atomistic and Full-Band Quantum Transport Simulator for Post-CMOS Nanodevices," *IEEE NANO 2008, 8th International Conference on Nanotechnology*, August, 2008.

[18] Luisier, Mathieu, A. Schenk, W. Fichtner, and Gerhard Klimeck, "Atomistic Simulation of Nanowire in the sp3d5s* Tight-Binding Formalism: From Boundary Conditions to Strain Calculations," *Physics Review B 74, 205323*, 2006.

[19] Luisier, Mathieu and Gerhard Klimeck, "Full-band and Atomistic Simulation of n- and p-doped Double-Gate MOSFETs for the 22nm Technology Node," *SISPAD 2008, 13th International Conference on Simulation of Semiconductor Processes and Devices*, September, 2008.

[20] *FAI – Fully Automatic Installation*, http://www.informatik.uni-koeln.de/fai/ .

[21] Hayes, Frank, "Frankly Speaking: Learning from IT Stunts," *Computerworld*, May, 2008.

[22] RedHat Enterprise Linux, http://www.redhat.com/rhel/

[23] RedHat Network, https://www.redhat.com/rhn/ .