Compaq's New Engineering Compute Farm Environment: Moving Forward with Windows NT

Don Brace, Andrew Gordon, and Scott Teel
Compaq Computer Corporation

Compaq’s New Engineering Compute Farm Environment: Moving Forward with Windows NT

Don Brace *(Don.Brace@compaq.com)*
Andrew Gordon *(Andrew.Gordon@compaq.com)*
Scott Teel *(Scott.Teel@compaq.com)*

Compaq Computer Corporation

Abstract

In the past, Compaq's design and verification distributed compute farm consisted of UNIX-based often times expensive proprietary hardware and operating systems. In addition, internal developers created site-specific load distribution software for the management of design verification compute intensive processing jobs. This environment offered Compaq design engineers (DEs) a finely tuned site-specific environment, where the benefits included on-site developers providing quick user-required modifications, in addition to a mature operating system environment. The downside included higher hardware purchasing and maintenance costs. However, with the emergence of Windows NT, Platform Computing's *Load Sharing Facility* (LSF) for Windows NT, the availability of Electronic Design Automation (EDA) applications for Windows NT, and high-end Intel-based workstations, a Compaq-based Windows NT LSF compute farm proved to be a viable solution. Compaq's internal systems engineers (SEs) and application engineers (AEs) began the process of developing a Windows NT-based compute farm environment that would provide a robust distributed compute farm to be used to facilitate the designing and verification of future Compaq products. This paper describes the issues encountered and the solutions required to bring the Compaq-based Windows NT LSF compute farm online and into production.

1. Compaq’s UNIX-based Compute Farm

Compaq's compute farm evolved over a period of time dynamically changing to meet the needs of its DEs. The path to the current compute farm configuration was built with hard work and innovative ideas that have helped to make it an operational success. Compaq DEs have come to rely on its stability and robustness to process their EDA designs.

1.1. History and Motivation

Components used in designing a computer system were simulated in software, starting an evolution in computer design. DEs were able to select components from a database and assemble them into simulated boards. These simulated boards could then be tested with other software, reducing the number of times that the designs needed to be sent to the board manufacturers. This in turn reduced design costs and shortened the design cycle.

Before compute farms, each EDA DE used powerful UNIX workstations to run the EDA applications. DEs soon outgrew their workstations requiring more memory and faster CPU speeds. It was not cost effective to replace each and every desktop workstation, so the next step was to purchase powerful servers. Each DE X-termed into one or more of their departmental servers to run their EDA applications. Frequently, these servers became overloaded, drastically reducing job throughput and slowing down the engineering design cycle. Conversely, there were times when these servers went unutilized by those assigned to them when others could have used them. The departmental server problem grew worse over time as EDA applications demanded more resources.

Attempts were made to do some rudimentary load balancing, meaning DEs called each other and reserved time slots on the servers. This was only a partial solution since problems often arose that caused them to overrun their time slots. The next step was to either purchase or design software, which would manage access to the departmental servers allowing for greater job throughput. Figure 1 illustrates a time-line that depicts the historical evolution of Compaq's EDA environment.

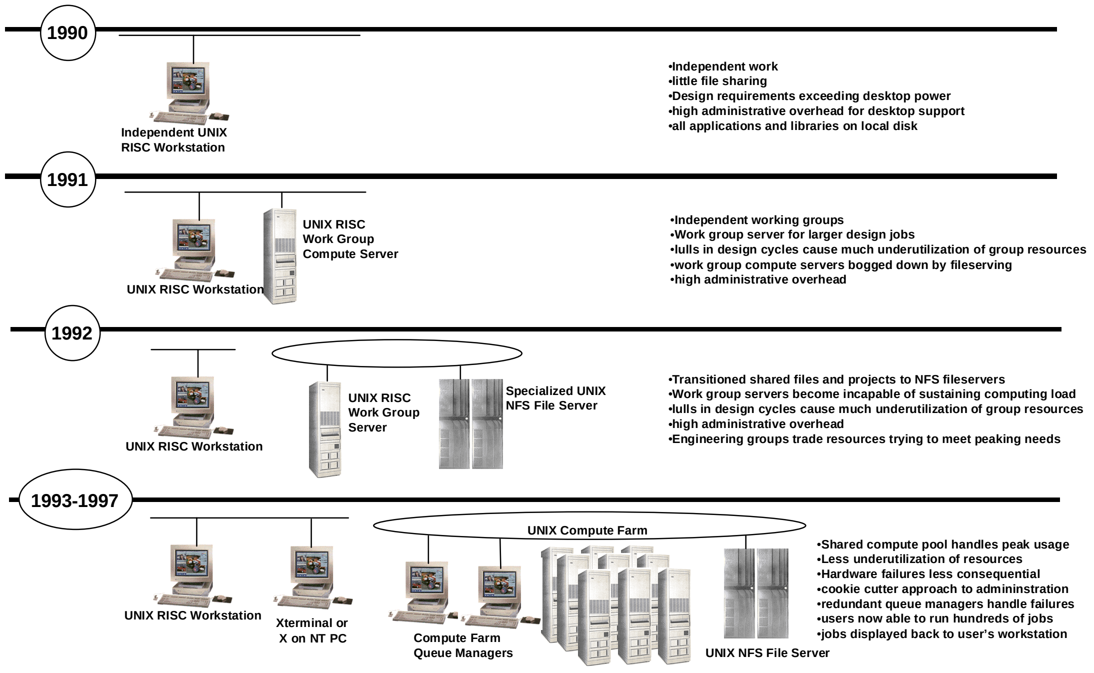
1990
Independent UNIX RISC Workstation
•Independent work
•little file sharing
•Design requirements exceeding desktop power
•high administrative overhead for desktop support
•all applications and libraries on local disk
1991
UNIX RISC Workstation
UNIX RISC Work Group Compute Server
•Independent working groups
•Work group server for larger design jobs
•lulls in design cycles cause much underutilization of group resources
•work group compute servers bogged down by fileserving
•high administrative overhead
1992
UNIX RISC Workstation
UNIX RISC Work Group Server
Specialized UNIX NFS File Server
•Transitioned shared files and projects to NFS fileservers
•Work group servers become incapable of sustaining computing load
•lulls in design cycles cause much underutilization of group resources
•high administrative overhead
•Engineering groups trade resources trying to meet peaking needs
1993-1997
UNIX Compute Farm
UNIX RISC Workstation
Xterminal or X on NT PC
Compute Farm Queue Managers
UNIX NFS File Server
•Shared compute pool handles peak usage
•Less underutilization of resources
•Hardware failures less consequential
•cookie cutter approach to admininstration
•redundant queue managers handle failures
•users now able to run hundreds of jobs
•jobs displayed back to user's workstation

Figure 1. Evolution of Compaq's UNIX-based Compute Farm

1.2. Why UNIX

EDA applications started and have continued on UNIX-based systems at Compaq for the following reasons:

- UNIX was the best-supported OS by EDA vendors.
- UNIX workstations provided high-resolution graphics.
- UNIX supported a large memory model.
- UNIX adopted the X-protocol for GUI support.
- UNIX adopted the Network File System protocol (NFS).

Early on UNIX-based workstations were configured with high-resolution graphics and a larger memory capacity than a normal DOS PC. Later on both the NFS protocol and the X-protocol added functionality in the areas of filesharing and distributed graphics.

Due to enhancements provided by UNIX vendors, often times it was easier to maintain a homogeneous set of hardware platforms and operating systems rather than a heterogeneous one. Once a specific hardware platform and operating system was chosen, it was much harder to adopt another.

Each UNIX vendor selected a different CPU architecture with which to run their UNIX operating system on. This meant that application vendors had to port their applications to each hardware and operating system type.

1.3. In-house Developed Batch Queuing System

At the time job management systems were researched, there were no viable third party solutions that met Compaq's needs. Thus an in-house solution was developed with the following requirements:

- Provide DEs with the ability to submit an unlimited number of both batch and interactive jobs.
- Provide secure access to the batch queuing system.
- Balance the compute farm load.
- Create a virtual environment for each job.
- Create a fault-tolerant batch scheduling system.
- Provide fair access to the compute farm.
- Provide an efficient administrative interface.

DEs typically ran regression tests that amounted to thousands of job submissions to the queuing environment. DEs automated their job submissions since the command line interface was developed to support this

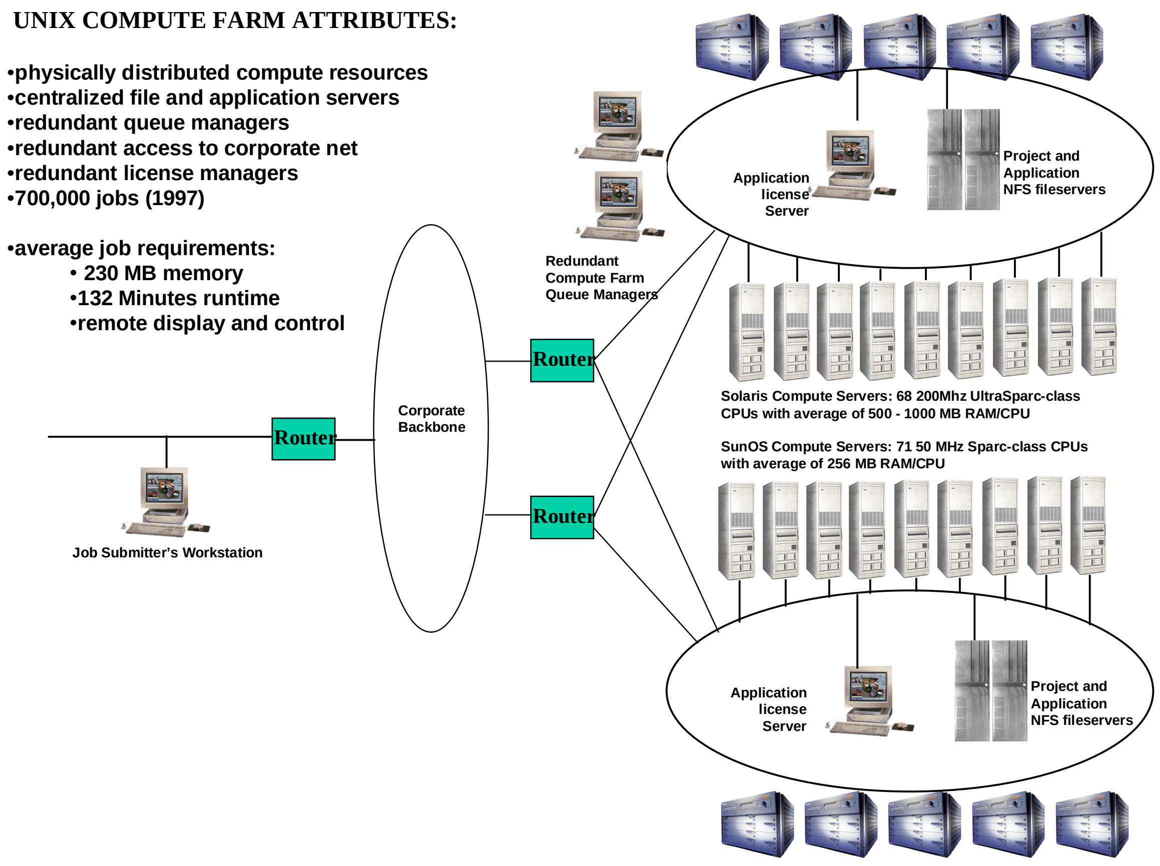
UNIX COMPUTE FARM ATTRIBUTES:
•physically distributed compute resources
•centralized file and application servers
•redundant queue managers
•redundant access to corporate net
•redundant license managers
•700,000 jobs (1997)
•average job requirements:
• 230 MB memory
•132 Minutes runtime
•remote display and control
Job Submitter's Workstation
Router
Corporate Backbone
Router
Router
Redundant Compute Farm Queue Managers
Application license Server
Project and Application NFS fileservers
Solaris Compute Servers: 68 200Mhz UltraSparc-class CPUs with average of 500 - 1000 MB RAM/CPU
SunOS Compute Servers: 71 50 MHz Sparc-class CPUs with average of 256 MB RAM/CPU
Application license Server
Project and Application NFS fileservers

Figure 2 Compaq's UNIX-based Compute Farm

functionality. Jobs that ran in the batch queuing system ran with the engineer's UNIX identity, thus protecting the project information.

Load balancing was an important feature of the batch queuing system. From benchmark testing of the actual applications, it was found that running one job per CPU maximized job throughout. Also, the scheduler tracked the amount of memory currently in use on each compute server, weighted by the projected amount of memory its running jobs would need over their lifetime.

The batch queuing system used two machines to provide instant fail-over. The scheduling mechanism was a rather complicated token fairness algorithm. DEs were assigned to various projects and the goal was to avoid giving one project priority over another. Ties were broken using round robin. The token costs were determined by benchmarking the compute servers and giving higher token costs to the faster machines.

The administrative interface was GUI based and allows modifications to be made to the running system. Users could be added and deleted from queues and projects, and machines and queues could be added or deleted from the system.

1.4. Compaq's Current UNIX-based Compute Farm

The UNIX compute farm has been in use for over five years and has been quite reliable. In 1997, over 700,000 jobs were run through the UNIX compute farm. The compute farm consists of a collection of Sun, Tatung and Solbourne compute servers and two major user queues. One queue is for batch oriented jobs and the other queue is for interactive jobs.

The compute server hardware consists of 3 Ultra Enterprise 4000 machines each with 8 GB of memory, 21 Tatung Ultra-2 machines each with 1.2 GB of memory, 15 Solbourne 900 machines with 1.2 GB of memory, and 2 SPARC 20 machines with 128 MB of memory. The Ultra class machines use Solaris 2.5 as the operating system and the Solbourne and SPARC 20s run SunOS 4.1.3_U1. In total, there are 142 job slots available for DEs to use. The entire batch queuing management system is managed from 2 Sun SPARC 20 machines.

2. Motivations for Establishing a Windows NT-based Compute Farm

As the UNIX-based compute farm matured, it became an integral part of Compaq's time-to-market engineering strategy. The memory size and complexity of jobs in the compute farm continued to grow. Older UNIX resources in the farm were becoming obsolete. Compaq SEs needed to continue the success of the compute farm while reducing costs and administrative overhead.

2.1. History & Motivation

When the time came to replace aging UNIX hardware and expand compute farm capacity, all available solutions were considered. While reports indicated that an Intel-based hardware platform would provide the best total cost of ownership, easiest maintenance, and highest price/performance ratio, this had yet to be proven. Selecting a hardware platform proved the easiest choice. Current Compaq hardware ran engineering jobs 2-3 times faster than the older SPARC-based hardware, and was comparable in performance to newer UNIX hardware choices for a lower purchase cost. Memory prices were similarly competitive. Rejecting RISC-based UNIX hardware for the less expensive Intel-based platform would allow Compaq's Design Environment to expand compute farm resources.

When considering operating systems (OSs), Solaris led the pack. Sun's Solaris X86 was available for Compaq hardware, was a mature product, and had the advantage of easing the task of porting administrative and engineering scripts and utilities. However, Microsoft's Windows NT boasted solid support from a growing number of engineering application software providers, which had no plans to port their tools to Solaris X86. Encouraging results from initial internal UNIX to Windows NT porting efforts established a confidence level that applications, scripts, and utilities could be ported. Hardware, operating system, and EDA applications indicated Windows NT as the OS of choice. It became clear that Intel-based hardware running Microsoft Windows NT would become the core of Compaq's future engineering computing farm.

2.2. Off-the-Shelf Load Sharing Product

Though Compaq's DEs had been enjoying the benefits of a locally developed load balancing, fault-tolerant job queuing system, the cost of those benefits were being weighed against the purchase of an off-the-shelf product. The development, testing, and support resources required to produce a system specifically tailored to Compaq's business, computing and engineering needs were growing, directly in contrast to the compute farm goals of shrinking administrative overhead.

Compaq required a commercially available, supported product capable of being configured to meet Compaq's very specific needs. A research project starting in early 1997 identified Platform Computing's Load Sharing Facility (LSF) as the only product on the Windows NT platform that would meet Compaq's needs for performance, reliability, flexibility, and support.

SEs looked at several products including an internally developed batch scheduler before selecting LSF's batch queuing software for Windows NT. Most of the products where mainly built for calendar event-driven scheduling, and our requirements could not be fulfilled by these types of products. However, the LSF software provided all of the basic requirements and most of the flexibility of the Compaq-developed compute farm system. LSF would leave SEs free to develop and tune other parts of the environment without having to port the internal batch scheduling system.

2.3. EDA Applications Migrating to Windows NT

As the Windows NT compute farm plan developed through the first half of 1997, EDA software vendors announced their support for the Windows NT platform. By August of 1997, a key application in ASIC chip design became available on Windows NT, increasing the desire for a Windows NT-based computing farm. Cadence Design Systems' Verilog-XL, the main EDA application in use in the UNIX-based compute farm, was now supported on Windows NT, and exhibited comparable performance to its UNIX-based cousin. Eventually, other EDA software tool vendors would announce support for the Windows NT platform.

Adding a Windows NT Compute Farm would expand Compaq's engineering computing capabilities without the purchase of additional non-Compaq hardware. It would make more job processing power available in the UNIX compute farm by allowing some work to shift into the Windows NT farm. Cost savings would be realized in initial purchase, hardware maintenance, support, and local development. Engineering application support was growing.

3. Setting up a Windows NT Compute Farm: Investigation, Testing, and Planning

After an initial internal proof-of-concept evaluation for a Windows NT compute farm using LSF, Compaq SEs recommended to continue on with an in-depth evaluation. SEs continued to evaluate the Windows NT compute farm concept extensively in addition to laying the foundation for a production-ready compute farm through investigation, testing, and planning. Due to the lack of experience with Windows NT in a batch-processing application environment, Compaq SEs worked through each of the following areas recording their results during each phase.

3.1. Defining the Compute Farm Environment Requirements

At the highest level, the project plan defined the major compute farm requirements:

- Compaq Hardware
- Windows NT-based Operating System
- Configurable Fault-tolerant Batch Scheduling Mechanism
- Real Time and Historical Usage Reporting System
- Distributed Graphics for Interactive Jobs
- Remote Capabilities for Dial-Up Users
- Remote Administration Capabilities

One of the most important goals of this project was to use Compaq-based hardware during the EDA design verification phase of Compaq's products. In a search for the right hardware, SEs evaluated all Compaq-based hardware for price/performance, and its processing granularity to reduce the compute farm's single point-of-failures. The Compaq Workstation and ProLiant products fulfilled these requirements.

Due to an existing user installed base of Windows NT on Compaq-based products and the availability (or planned availability) of EDA applications on the Windows NT platform, SEs agreed that Windows NT would be the operating system of choice.

Like the internally developed scheduler, the Windows NT compute farm scheduler would need to provide flexible configuration parameters to allow for the implementation of site-specific scheduling policies. Due to the criticality of internal schedules of EDA verification projects, the batch scheduling mechanism would need to provide the following functionality:

- Easy configuration capabilities.
- Fault tolerant across network or server failures.
- Scheduling based on resource needs such as memory.
- Load-balancing.
- Access control to batch queuing system.
- Access control to administration commands.

Once the compute farm reached production status, compute farm usage statistics, both real-time and historical, would provide critical resource planning information. EDA management defined the need for usage reports, both real-time and historical, for this compute environment.

During the initial stages of the ASIC verification process, engineers debug large verification jobs working out problems in both the design and the regression tests. The debugging process requires an interactive session with the regression software. As such, if an engineer wanted to debug a verification job using a compute farm resource, the engineer would need to interact with the remote job. Historically, some sort of distributed graphics mechanism provided this functionality. As a result, EDA engineers identified the need for interactive capabilities during large regression setup and debugging from within the compute farm.

Once ASIC verification engineers have worked through problems within their regression suite, they will submit a large set of regression jobs to the compute farm. This can be literally hundreds or even thousands of jobs per submission setting. Often times during time-critical regression, engineers will need to log in from off-site locations and check the status of a regression suite. As a result, engineers identified the need for remote access capabilities including job submission and job monitoring.

As with any distributed computing environment, there exists the need to administer machines in an efficient manner, especially when machines are physically located in separate locations. Simply, the time of traveling to the different computing facilities can be expensive. In addition, after the compute farm reached production status, hotline SEs who are normally tied to a phone line would need to remotely log onto the problem compute server and understand the cause of any problems. This was a definite requirement.

3.2. Selecting Initial Windows NT Compute Farm Applications

EDA AEs identified an initial set of design verification tools to be used within the Windows NT compute farm based on several factors:

- Availability on Windows NT
- Ability to run in batch mode
- Functionality during the verification process

Not only did the EDA applications need to be available on Windows NT, but certain internal verification libraries would need to be ported over to Windows NT. These modules are linked in at runtime. Furthermore, the verification tool would have to have the option to run in batch-mode. After identifying a small set of possible EDA applications, Verilog-XL would become the first application to run in the compute farm environment.

In order to run any EDA application on a Windows NT machine, the environment had to be setup correctly, and once EDA AEs establish a working environment, this environment would need to be transferred to the compute farm server at job submission time. During the initial application setup, AEs chose to install the EDA applications onto a centralized file-serving setup in which application binaries and libraries would reside in a single file-serving location. This would reduce the administration burden.

To help facilitate the job submission process, EDA AEs developed submission tools that submit jobs during the design verification process. A portion of these tools required the batch job to run from the submission (network drive) directory—this was added to the list of application requirements.

3.3. Researching the Major Issues

Having defined the requirements of the Windows NT compute farm environment, project members turned to discussing the issues that needed to be resolved. The following list enumerates the major issues that had been identified before and during the initiation of the compute farm project:

- Identifying the optimal compute farm hardware configuration (compute model) for our EDA applications: #CPU's/machine, Memory/CPU, network bandwidth.
- Identifying an initial optimal LSF batch-queuing configuration setup: #queues, scheduling parameters.
- Identifying an optimal UNIX-NT file sharing solution for our environment to access existing project information on UNIX-based fileservers.
- Identifying a flexible remote administration tool to allow remote access to compute servers used during administration or debugging.
- Identifying or developing a dynamic Windows NT drive-mounting mechanism needed by in-house developed EDA application tools.
- Identifying and reviewing the Windows NT file serving infrastructure configuration and setup.
- Identifying EDA application environment problems.
- Testing GLOBEtrotter's FLEXlm license mechanism from Windows NT using UNIX-based license servers.
- Identifying and installing UNIX-based development and scripting tools within the centralized Windows NT file-serving environment.
- Identifying administration tasks that could be automated.
- Identifying administration and support documents that needed to be developed.

3.4. Extensive Evaluation and Testing

To identify or resolve certain issues in addition to verifying previously made decisions, SEs spent a considerable amount of time testing, especially with regards to the LSF products, UNIX-NT file sharing products, and EDA applications. The following sections describe the test results.

3.4.1. LSF Product Evaluation and Testing

After an in initial proof-of-concept evaluation, the SEs identified the LSF batch-scheduling product as the only real possibility other than porting their internally written job scheduler. However, even though Platform provided a flexible batch queuing mechanism on Windows NT, SEs needed to extensively test the product, trying to understand its strengths and weaknesses especially on the Windows NT platform. As such the SEs prepared an extensive test plan to exercise the LSF product in the following areas:

- Administration, Manageability, & Usability
- Fault-tolerance
- Reliability
- Batch Scheduling Features and Capabilities
- Platform Computing's Support Capabilities

3.4.1.1. Administration, Manageability, and Usability Evaluation

Administration and manageability of a LSF cluster was fairly simple after completing the initial setup and configuration. Initially, the LSF 3.0 installation was not wrapped up inside of a simple Install Shield mechanism as most Windows-based applications, but Platform added this type of capabilities with the LSF 3.1 product. The administrative burden can be reduced when application binaries and configuration files are centralized. The LSF product incorporates this feature quite naturally having ported their product from a UNIX environment.

For administrators, the LSF product provides an administrative command suite that can stop, start, restart, and shutdown each respective service and/or daemons. The most difficult part of the Windows NT LSF administration is administrating the Windows NT itself. The limited number of debugging tools makes problem solving more difficult.

One administration example is adding a machine to the LSF cluster. Although there is no current Windows NT GUI that provides an editing interface, the process of adding a machine can be summarized in about five steps, each step being intuitive for a knowledgeable systems administrator.

From a user perspective, most LSF commands mimic the functionality of those used within the internally developed batch queuing system. In addition, the LSF product contains some functionality, such as the ability to view the history of a job. A job history provides historical job information: how long a job waited, the LSF server it executed on, the final exit status, along with other useful information.

3.4.1.2. Fault-tolerance Testing

SEs tested LSF's ability to recover from simulated network outages, simulated cluster failures, and simulated network file system failures. SEs simulated a network outage by unplugging the network connections to several LSF cluster machines. Likewise, members simulated a cluster failure by powering down the currently running master LSF cluster server. SEs simulated file system failures by shutting down the appropriate file share server.

Since there are two LSF layers—the LSF Base and Batch—one requiring the services of the other (Batch requires the services of the Base software), recovering from major network or cluster errors occurs in two phases. First, the *lim* (LSF Base software) daemons must determine if the master *lim* has gone away and if so, pick a new one. Secondly, the *sbatchd* (LSF Batch software) must determine if a master batch daemon has gone down, and if so, pick a new one. The *lim* daemons are considered functional once the *lshosts* commands reports a full list of LSF cluster servers, while the *sbatchd* daemons are considered functional after the bhosts command reports a full list of LSF batch cluster servers.

While either the *lim* or *sbatchd* layers are non-functional, both user commands and normal scheduling activities will cease; however, previously scheduled user jobs will continue to run. User commands will continue to try and contact the master scheduler until a specified time-out valued has been reached. If the time-out value is reached, the user command exits with a failure; otherwise, the user command will complete the task when it is able to finally contact either the master *lim* server or the *mbatchd* (depending on the type of user command). Tests indicate 30 to 45 seconds cluster reorganization. During reorganization, no user *lim* or *sbatchd* commands will complete successfully until each respective layer has regrouped itself.

3.4.1.3. Reliability Testing

With most out-of-the-box products, there exists certain stress conditions under which the application will stop functioning in a production quality manner. Knowing these conditions can be very beneficial to system administrators or managers. SEs essentially tried to find out how to break LSF. One such example was the MBD_SLEEP_INTERVAL parameter located in the lsb.params configuration file. Setting this interval too small causes the master batch server to spend too much time trying to schedule jobs, unable to respond to new requests from client machines.

3.4.1.4. Batch Scheduling Features Evaluation

SEs experimented with several scheduling parameters available in the LSF Batch software. In general, all parameters worked as expected. Jobs were scheduled on the correct resource based on the requested resource requirements. For Compaq's design verification computing environment, a batch scheduler's memory management capabilities are critical to efficient memory and CPU utilization. LSF provides this by means of a resource requirement selector with different selection possibilities. SEs implemented the memory resource requirement selector with a *linear decay duration value* specified at job submission time by either the user or a pre-determined default value. The *linear decay duration value* indicates how long the job will execute before it has consumed its total memory requirement. Initially, the default value will be set to 30 minutes, but SEs urged users to set this value using a command line option

SEs tested the D*ispatch Window* mechanism by running user jobs on certain team member's desktop machines at night. SEs configured these desktop machines to accept jobs from 7PM until 5AM the next morning. This feature worked, but SEs do not plan on implementing this feature in any initial release due to certain manageability issues such as remote administration and controlled downtime.

Fairshare testing pointed out the possibility of one group being temporarily starved of slots due to their past excessive use of the cluster. The window of accumulated CPU and wall clock time is a configurable option in the lsb.params file. Although the *Fairshare* mechanism does not exactly mimic the *Token Fairness* algorithm used within the Compaq’s internal-developed scheduler, it does provide a fairness policy based on user groups. The LSF scheduler implements *fairness* by setting up "fair-shares" which indicate how much usage units should be given to each user or user-group. The scheduler keeps track of past CPU and wall-clock job usage, and uses this information on future scheduling decisions.

Finally, SEs tested the scheduler's ability to scale by increasing the number of nodes in the cluster and repeating some of the earlier tests. Even with 10000 jobs queued up, the scheduler continued to respond to client requests in a constant predictable time.

3.4.1.5. Platform Computing's Support Capabilities

SEs tested Platform Computing's ability to provide quality support in a timely manner. Project staff concentrated on four specific types of problems often encountered within a production-computing environment: *high-priority installation problem*, *dead-in-the-water problem, enhancement request*, and *a possible bug problem*.

For all test cases, Platform Computing provided quick support response, usually resolving within 15 minutes. Due to the small size of the company, SEs could quickly access the actual LSF code to determine if a bug existed or if we had accidentally configured LSF incorrectly. And if online SEs could not resolve the problem, LSF support members would summon the Windows NT developers to the phone for help.

3.4.2. UNIX-NT File Share Testing

Since Compaq's design files resided on UNIX file-servers, it was important to identify a method for sharing those project files with Windows NT systems. Making copies to an Windows NT server and keeping them synchronized was not an option, nor was using ftp to transfer files as needed. A true filesharing solution enabling simultaneous access from both platforms was required.

Filesharing options are limited to either client-side NFS or server-side SMB/CIFS interpreters. The PC NFS client products allow an Windows NT PC to connect directly to remote NFS filesystems using NFS protocol through an add-on protocol stack. The server-side solutions are SMB interpreters, which cause the NFS fileserver to appear as an Windows NT server to Windows NT clients. None of the filesharing products have coordinated file locking between Windows NT and UNIX, so there is some danger that files may be corrupted or overwritten if accessed at the same time by both platforms. Since file-naming conventions are different between Windows NT and UNIX, there are also some things to look out for in this area. Some files created with perfectly legal filenames in UNIX become inaccessible from Windows NT. Case sensitivity is an issue that has been worked around in most products, but not solved. Filesharing users must be aware of the limitations of each platform's filenaming conventions. Windows NT and UNIX have different security ID and file permission models, so there were issues here too.

The filesharing product evaluation revealed that there was no product yet that provides a seamless file sharing solution. It would be up to engineers and administrators to use filesharing carefully, and avoid the known problems until better solutions become available.

3.4.3. EDA Application Testing

EDA AEs spent considerable amount of time testing Windows NT EDA applications both on the their Windows NT desktops and also in the Windows NT compute farm. It was not until just recently that certain EDA vendors had started porting their tools over to Windows NT and certain bugs were still being worked in parallel to bugs in our compute farm environment. EDA AEs first worked to provide a production desktop environment that would allow certain EDA applications to run from a single application server as done historically with the UNIX desktop and compute farm environment.

After EDA engineers worked through desktop support issues, they moved on to getting applications to correctly run in the compute farm. The number one issue centered on making sure the desktop environment was correctly mapped to an LSF server during job initialization. The LSF software appeared to have been written to transfer only certain environment variables based on their interpretation of the Windows NT environment. After working through the environment issues, SEs updated the already internally developed LSF wrapper script to work-around these items.

EDA AEs had developed a suite of submission and verification tools to augment the testing functionality that already existed in the vendor product such as Verilog-XL. As such, internal EDA AEs worked on porting their submission tools over to Windows NT and changing their submission modules to use the new LSF-based job submission mechanisms. Once AEs had worked through issues both from a desktop issue and a compute farm perspective, large scale job submissions commenced and application load testing began to identify any stress-related problems during intense processing by real EDA applications.

3.5. Windows NT Compute Farm Environment Version 1.0

After internal discussion about the compute farm requirements and impending issues, SEs constructed a project plan that defined reasonable project milestones. The initial release of the compute farm as a production system fulfilled only the following previously listed compute farm requirements:

- Compaq Hardware
- Windows NT-based Operating System
- Configurable Fault-tolerant Batch Scheduling Mechanism

SEs will complete the other compute farm requirements in subsequent phases.

4. Compaq's Initial Windows NT Compute Farm Environments

SEs installed and configured all hardware into a climate controlled computing facility. This section describes the actual compute farm setup and current solutions to the previously mentioned issues.

4.1. Compute Farm Hardware

As of the writing of this paper, SEs successfully configured a 60 CPU LSF cluster running Windows NT 4.0 with SP3 cluster consisting of the following list of Compaq Workstation and ProLiant machines (see figure 3):

- 20 Professional Workstation 8000 with 3 x 200 MHz Pentium Pro CPUs and 3 GB of total memory used as LSF Batch servers all using 100 Mbs FDDI connections.
- 3 ProLiant 5000 fileservers with 2 x 166 MHz CPUs each with 128 MB of total memory using 100 Mbs FDDI connections.
- 3 ProLiant 2500 fileservers with 100 Mbs FDDI connections.

At the start of this project, the Professional Workstation 8000s provided the largest expandable memory option of up to 3 GB of total configurable memory. As a result, the main core of the compute farm consists of 20 Workstation 8000s each configured with 3 GB of total memory using 100 Mbs FDDI connections.

Before the compute farm project, SEs designed a centralized fileserver model for the Windows NT desktop infrastructure. This allowed for EDA applications to be stored and executed from one central location—a similar UNIX environment already existed. The compute farm jobs access binaries, configuration files, and project data located on 2 ProLiant 5000 fileservers—a (primarily read-only) application server and a (read-write) project server.

During the evaluation phase, occasional network, application, and operating system errors caused problems for the master LSF batch scheduler. SEs recommended setting up two separate LSF server machines to be used only as schedulers and not as compute servers, preventing rogue jobs from causing a system failure on the master scheduler.

4.2. Batch Queuing Software

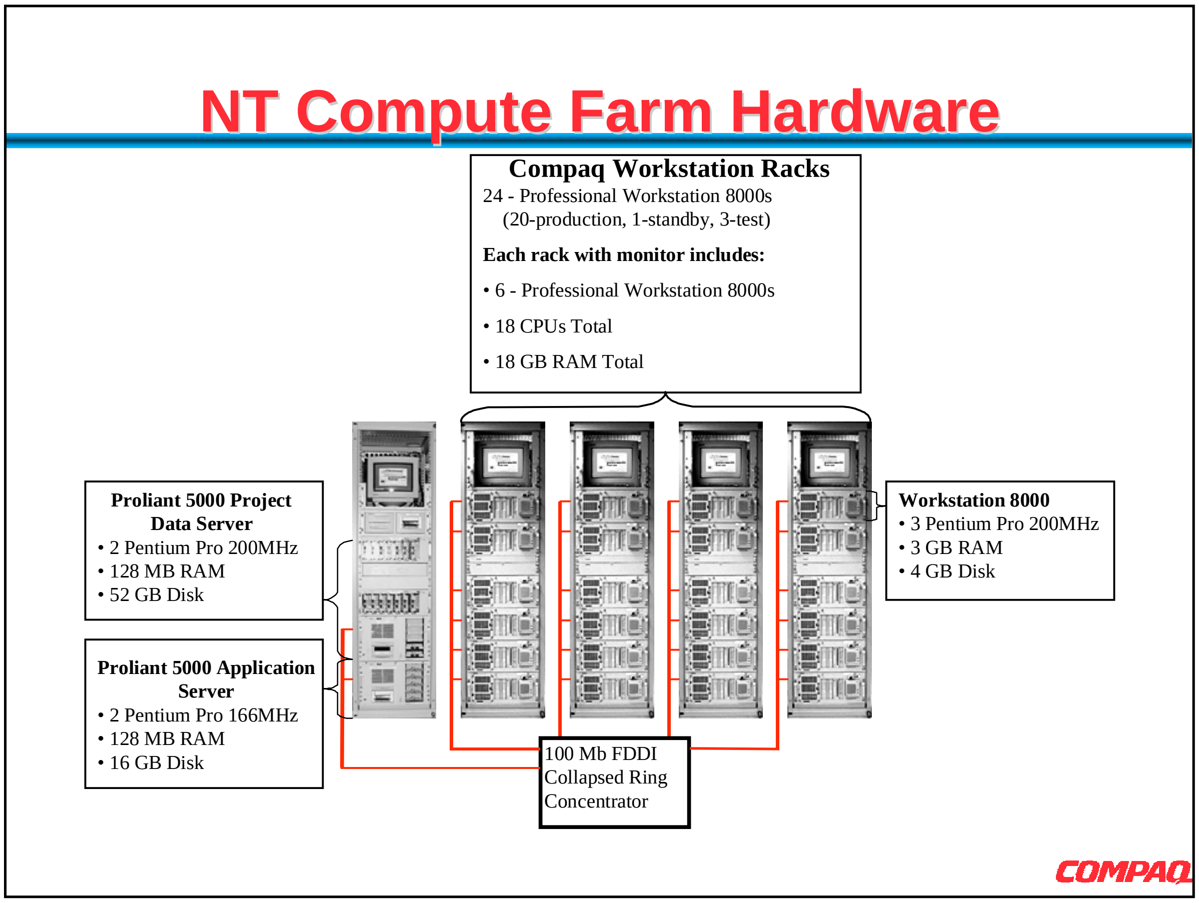
NT Compute Farm Hardware
Compaq Workstation Racks
24 - Professional Workstation 8000s
(20-production, 1-standby, 3-test)
Each rack with monitor includes:
• 6 - Professional Workstation 8000s
• 18 CPUs Total
• 18 GB RAM Total
Proliant 5000 Project Data Server
• 2 Pentium Pro 200MHz
• 128 MB RAM
• 52 GB Disk
Proliant 5000 Application Server
• 2 Pentium Pro 166MHz
• 128 MB RAM
• 16 GB Disk
Workstation 8000
• 3 Pentium Pro 200MHz
• 3 GB RAM
• 4 GB Disk
100 Mb FDDI Collapsed Ring Concentrator
COMPAQ

Figure 3. Compaq's Windows NT Compute Farm Configuration

SEs deployed LSF 3.0b version onto the compute farm resources. From an architectural view, the LSF software is cleanly divided into two layers:

- A Load Information Management layer (LIM) discretely manages the load indices of all machines in the cluster.
- A batch application layer uses a LIM API to determine the correct resources to dispatch jobs to.

Each layer provides a suite of commands for viewing or changing the cluster.

4.3. Remote Windows NT Administration

SEs had looked at several different products, including Microsoft's System Management Server (SMS), which was also being tested as an application installation tool. As such, the team tried using the SMS remote control tool, but the SMS remote control tool would often stop working very early into the remote connection, in addition to being slow in general. Next, SEs started testing PCI Ltd's PC-Duo product. The PC-Duo product had good response along with an access control mechanism to provide some security control. As a result, SEs decided to use PC-Duo as their remote administrative tool of choice.

4.4. Hardware Monitoring

To provide a hardware monitoring mechanism, SEs installed and configured the Compaq Information Management (CIM) agents on all of the compute server resources. The CIM agents provide hardware state information and errors to a CIM management application.

With regards to monitoring in general, SEs setup a separate monitoring machine to run the CIM management GUI application continuously. In addition, several other monitoring programs run on this designated machine.

4.5. Time Synchronicity

As with most distributed compute farm or cluster systems, synchronized clocks are critical to distributed algorithms. In addition, the application themselves can be time-dependent, relying on a common time-stamp for project files. Furthermore, licensing mechanisms such as FLEXlm rely on a consistent clock between server and requesting client.

In the current UNIX compute farm environment, SEs had already setup xntpd time servers used by XNTP clients updating their system clocks. Since the infrastructure already existed, SEs ported the xntp software code to Windows NT and deployed on all Windows NT machines accessing the compute farm environment. This proved to be a critical element as several problems were traced down to the xntpd client not running correctly on the problem-related compute farm server.

4.6. File System Infrastructure and Application Setup

SEs reduced administrative efforts by centralizing the LSF configuration files. This provided a large administrative gain by having only one location for updates. To simplify initial compute farm requirements, EDA AEs conceded to running jobs that would only access Windows NT files and not UNIX-based files via a UNIX-NT file sharing solution. However, due to the already large investment of UNIX-based file servers, some form of UNIX-NT solution will become a future requirement.

SEs had already established a common primarily read-only application server for EDA applications and supporting tools as done in the similar UNIX environment. In addition, SEs configured several data servers for project-based file storage.

4.7. Administration Scripts and Services

As part of the compute farm production release, SEs developed scripts and services to automate some administrative tasks:

- LSF job output file renamer—this utility changes the output file names to coincide with those generated by an in-house batch-scheduling UNIX-based system. This reduced certain EDA submission tool porting efforts.
- LSF output directory cleanup—this utility would manage the user's job output directory compressing and archiving job files at specific intervals.

4.8. LSF Wrapper Submission Script

Early in the evaluation, the SEs deveoped a *bsub* wrapper script for the following reasons:

- To reduce the LSF administrative burden of LSF by centralizing the file job scripts and output files.
- To implement control measures with regards to resource requirements and usage.
- To implement a dynamic Windows NT file-sharing solution on the LSF servers when needed.

The wrapper script takes the user's job and does the following:

- Checks the command line options for valid memory estimates.
- Checks for valid project names and project access rights.
- Inserts network drive mount and un-mount commands.
- Submits the newly created job via standard *bsub* command.

4.9. Automatic Mounting Utilities

The EDA AEs had created a suite of EDA application tools. When executed in the UNIX compute farm environment, these tools relied on the batch job running from the initial submission directory. As a result, Windows NT compute farm jobs would need to do the same. SEs placed hooks into the LSF wrapper script to automatically mount the network submission drive at job startup, move to the submission directory, and reverse the procedure during job cleanup. One caveat became the ability to cleanly kill a job without leaving drive mounts around—eventually running out of drive letters. Later on SEs looked to a special job killing mechanism to keep leftover drive letters from collecting on a compute resource.

4.10. Windows NT Configuration

SEs examined the Windows NT Workstation vs. Server decision and decided to build compute farm machines using the Windows NT Server product. SEs loaded Windows NT 4.0 server with Service Pack 3 on all machines to be used within the compute farm and left the default performance settings.

During system and application testing using UNC paths, SEs identified a Windows NT Redirector problem. After working with Microsoft SEs, Microsoft provided a patch that fixed the problem and is now applied during the build process for all LSF compute servers and other Windows NT machines (Q179983 and Q179873).

As per the LSF documentation, SEs configured LSF compute servers with static IP addresses. Although our Windows NT environment primary consists of DHCP clients, the persistency of the leases allowed our LSF clients to work successfully within the LSF cluster even though the LSF documentation recommended only static IP addresses.

4.11. License Management

Licensing issues were addressed and resolved before the application could correctly run in the Windows NT compute farm environment, including the batch queuing software itself. EDA applications and LSF's batch queuing software used the industry standard FLEXlm software for license management. This facilitated the initial setup for two reasons:

1. UNIX-based FLEXlm license servers could manage (issue) client license checkouts from Windows NT clients.

2. SEs had already established redundant UNIX-based FLEXlm servers for applications that ran in the UNIX-based desktop and compute farm environment.

As such, SEs and AEs setup the new Windows NT application license features and keys on the already established UNIX-based license servers, thus reducing the overall setup effort.

4.12. Web-based User and Support Documents

Like all products or projects, the final deliverables must include good documentation. For the Windows NT compute farm, AEs and SEs updated support procedures that helped both users and administrative staff to understand the new system. Due to its simplicity and usability, SEs chose to create web-based documentation and place on local (departmental) web-sites for easy accessibility and searching capabilities.

The final documentation included several deliverables including evaluation reports, administrative setup procedures, administrative trouble shooting guides, administrative test procedures, bug tracking lists, user guide, user FAQs, and future project plans.

4.13. Compute Farm Test Environments

Test environments are essential for improving the compute farm without risk to production job activity. As a completely separate environment, the test farm insulates the production farm from harmful events. New versions of utilities, scripts, computer hardware, and even the job queuing and load sharing facility itself may be tested without affecting production jobs. Coupled with a detailed test plan and bug tracking, the test farm environment ensures a stable production compute farm. Operating systems, utilities, applications, and hardware must be kept at the same revision levels as the production environment

Compaq SEs setup several test compute farm environments to allow SEs and AEs to test applications and compute farm tools independently without affecting each other or the production compute farm environment.

5. Future Windows NT Compute Farm Goals

As mentioned previously, SEs had only reached certain compute farm goals, leaving some for future compute farm releases. Work continues on the remaining initial requirements and some additional goals that will improve the robustness and functionality:

- Intelligent Job Terminator
- LSF Hardware and Software Monitoring System
- UNIX-NT File Sharing Solution
- Job Profiling Service
- Real Time and Historical Usage Reporting System
- Remote Capabilities for Dial-Up Users
- Remote User Control for Interactive Jobs

5.1. Intelligent LSF Job Terminator

The LSF wrapper script created by SEs is treated by the LSF system as a job component, it does not know that it should not kill this process whenever a user kills a running job. Thus the kill command kills the wrapper script as well, leaving behind mounted systems and used drive letters. A local kill command, called *cbkill* was developed as a workaround to this problem. The *cbkill* utility behaves similarly to LSF's kill utility, however it does not kill the wrapper script.

LSF has a rich set of APIs that enabled the *cbkill* utility to obtain the necessary information about a running job from the LSF server. The kill command is sent to the compute server though the use of a DCOM executable that starts up upon demand searches for the target processes and kills them.

5.2. Compute Farm Monitoring System

SEs are designing a compute farm monitoring solution to automate problem identification and resolution within the compute farm. Hopefully, this system will ease the administration burden, allowing administrators to accomplish productive tasks as opposed to firefighting compute farm problems.

Initially, the monitoring system will try to identify various system and LSF job-related problems:

- Jobs that are "stuck" or consuming considerably more resources than requested.
- Jobs with password problems. On Windows NT, the impersonation issue is resolved with the Windows NT application storing the user password encrypted in some hidden file and performing the *LogonAsUser* Win32 API call. The service control manager is a good example of a program storing an encrypted password to achieve a separate identification at startup time.
- Available drive letter problems. Currently, the setup wrapper script will mount drives for the job. These jobs can die abnormally and cause leftover mounts to accumulate. The Microsoft Windows Terminal product will help out here.
- LSF daemon problems such as an abnormal termination. Although LSF is fault-tolerant and will bypass a malfunctioning LSF server, SEs want to bring the compute server back into the cluster ASAP.

5.3. UNIX-NT File Sharing Solution

While Microsoft has announced plans for a PC NFS product in an add-on pack for enhancing UNIX/NT interoperability, current plans for the future compute farm call for using a server-side solution. Server-side solutions are easier to administer, debug, and license. Compaq plans to use Auspex Systems' NeTservices product on its Auspex NFS file servers to give engineers access to a common project directory from both Windows NT and UNIX computing platforms.

This strategy will assist engineering teams that are in transition from UNIX to Windows NT, preventing the need for a drastic, all-or-nothing transition from UNIX to Windows NT. Engineering teams that have already completed the transition to Windows NT will have less need of file sharing between platforms, and will have their project and home directories located on Compaq ProLiant fileservers running Windows NT Server.

5.4 Job Profiling Service

Engineers need performance information about their jobs to accurately predict the execution time and resources required for other jobs. A service program that runs on each compute server was created that periodically runs through the list of running processes and records statistical information which is appended to each job logging file. Applications Engineers have written utilities that take this information and store it in a database that is used in setting up job parameters for subsequent job submissions.

The information gathered is basically the same as what the task manager displays, however, only those processes that belong to the job stream are recorded in the logging file for the job. The code for this service was taken from the *tlist* utility found in the NTRESKIT.

5.5. Real Time and Historical Usage Reporting System

The compute farm is currently running real EDA application jobs, and there exists a need for both real-time trending and historical statistics that describe the general usage of the cluster. Historical and real-time statistical reports will be imperative for future capacity planning or current reallocation. LSF provides an LSF Analyzer product that may provide the necessary reports that SEs are requiring. Real-time statistics go beyond just providing a simple snapshot. These need to provide a trending ability to understand the overall load of the cluster over short dynamically configurable interval.

At this time, SEs are beginning to evaluate the LSF Analyzer and its ability to generate specified reports. SEs will begin to define the real-time statistical requirements and determine the need for an in-house developed solution.

5.6. Remote Capabilities for Dial-Up Users

Future plans include providing a dial-up solution for engineers wanting to access the compute farm from home or other remote sites. So far, the LSF software has been the biggest obstacle in that all LSF client and server machines within the cluster must be name-resolvable during configuration initiation for security reasons, which causes problems for dial-in users. Dial-in users do not generate an IP-name map until dial-up time via DHCP.

We are currently looking into several possible solutions:

- Using PC-Duo to allow engineers to access their desktop machines and the LSF compute cluster.

- Getting Platform Computing to make software modifications to allow this type of user to use the compute farm. At this time, LSF cannot handle this type of client machine.

5.7. Remote User Control for Interactive Jobs

DEs have requested the need for a remote interactive job control solution for jobs submitted to the Windows NT compute farm. Providing a solution has been difficult due to Windows NT's desktop only philosophy. Porting UNIX-type tty functionality over to Windows NT has been difficult due to a lack of process signals within Windows NT process model. Currently, SEs are looking at other remote terminal applications, such as Microsoft's Windows Terminal Server, to provide this functionality for interactive compute farm jobs. Microsoft's Windows Terminal Server may support a command line interface, which would allow its terminal sessions to be automatically started from a job stream and incorporated into the LSF software.

6. Summary

This paper describes the solutions required to bring the Compaq-based Windows NT LSF compute farm into production. Reasons for creating a Windows NT LSF compute farm include cost-effectiveness, migration of EDA tools to NT, and processor performance levels comparable to UNIX RISC machines.

Aging UNIX hardware and the need for expanded compute farm capabilities at a reduced cost triggered a series of hardware evaluations, seeking to upgrade existing platforms or adopt another. These evaluations resulted in the adoption of Compaq hardware running Windows NT. Lessons learned from UNIX-based compute farm were brought over to the Windows NT compute farm. Presently, the Windows NT compute farm consists of Compaq hardware, Platform Computing's LSF software, remote administration tools, and the main EDA tools that DEs use in the design process. Some issues remain and will be resolved in the future through projects that are in place to provide required solutions.