UNIVERSITY INFORMATION TECHNOLOGY

Tufts Linux Research Cluster

Introduction to Clusters and Parallel Computing

UIT: TRAINING AND DOCUMENTATION

Introduction to Clusters and Parallel Computing

Written by: Tina Riedel, Technical Writer
University Information Technology
Edited by: Andrew Fant, University Systems Group

Table of Contents

CHAPTER
1

Clusters and Parallel Computing – An Introduction

"The world can no longer afford to have all the expertise in computer science reside in elite places. We have problems to solve, and the world needs to contribute to solving them. With the combination of the Internet, Open Source operating systems and software, inexpensive "PC" hardware, and concepts like the Beowulf systems, any person from any country can now contribute to the development of these technologies. We hope to find and acknowledge the next "Albert Einstein" of computer science, whether he or she comes from the United States, from Brazil, from Korea, from China, or from any other country."

Jon "maddog" Hall
Executive Director
Linux International
January, 1999

SECTION
1

What is a Cluster?

AN OVERVIEW

Supercomputers are large systems that contain hundreds, sometimes thousands, of processors and gigabytes of memory which, in parallel, simultaneously run portions of large computer simulations, data analysis, and numerical modeling. They are used for scientific modeling, military simulations and planning, the rendering of film animation and special effects, and countless other data-crunching medical and research needs.

Although corporations such as Cray, IBM, and SGI continue to manufacturer supercomputer distributed systems, *cluster computing*, which is the alternative technology of grouping individual systems to accomplish the same goals, is making inroads in the supercomputer market and reducing the demand for these systems.

Cluster computing is the result of connecting many local computers (nodes) together via a high speed connection to provide a single shared resource. Its distributed processing system allows complex computations to run in parallel as the tasks are shared among the individual processors and memory. Applications that are capable of utilizing cluster systems break down the large computational tasks into smaller components that can run in serial or parallel across the cluster nodes, enabling a dramatic improvement in the time required to process large problems and complex tasks.

Cluster and grid computing[1] are alternative solutions that offer the advantages of parallel processing at a fraction of the cost and provide adequate performance for many applications. Although the Tufts Research Cluster is composed of new equipment, the cluster nodes can be composed of recycled and scavenged equipment that is no longer considered viable as user workstations.

The concept of cluster computing has been played with for decades and actually has roots in the large mainframe and supercomputers from the 60's and 70's. In those systems, many processors were bundled into one large system with tightly-coupled, shared memory and shared hardware resources. The Digital Equipment Corp. (*swallowed by Compaq Computer Corp., which was recently swallowed by Hewlett Packard*) was an early implementer of the clustering concept. VMS, DEC's operating system, had built-in cluster support. The reliable Digital VAXclusters went on the market in 1983.

But in the early 1990's, the convergence of several important factors made the replication of the supercomputer multiprocessor and VAXcluster models accessable with cheap, separate, individual computer systems possible. Commodity off-the-shelf (COTS) systems with relatively fast microprocessors and had become reasonably cheap and easy to come by. Furthermore, the rapid turnover of technology was providing most Universities and industries with piles of serviceable systems that were no longer adequate for daily work.

In 1994, Thomas Sterling and Don Becker, *who worked for CEDIS (Center of Excellence in Space Data and Information Sciences*), connected 16 systems with Intel 486 DX4 processors tied together with 10 Mbps channel bonded Ethernet (the processors were too fast for the standard 10 Mbps). That experiment grew into what is now called the Beowulf Project, a group of interested parties from the research and academic communities who work toward the development of related commodity off-the-shelf clusters. Beowulf class clusters are traditionally distinguished by low or no cost open source software for the operating system (*usually Linux*) and the message passing interface. But even that definition is changing, as commercial distributions for Beowulf class clusters are now available.

---

[1] *Grid computing is an alternate technology that interconnects many internet connected systems and/or clusters together. Its better suited to wide area connections and loosely coupled computing environments, such as groups of desktop workstations. Grid computing, in the context of large national and international projects, often refers to interconnected sites with large clusters and other high-performance computer systems that allow users at one site to utilize the resources of the other sites, and/or create and run software on the participating site resources.*

An Important Note – Cluster Naming Convention Used in This Manual

The Tufts Linux Research Cluster (cluster01.tccs.tufts.edu) is referred to throughout this manual as "*the research cluster*". This naming reference is accurate as of this writing, but it's important to note that cluster01 is the true identifying name and it's likely that other clusters will be part of the Tufts landscape in the future. Therefore, the information in this manual applies to the 32 nodes of cluster01 *only* and is not meant to define or describe future Tufts University clusters or cluster nodes.

SMPs, MPPs, and Parallel Processing

Parallel computing architecture encompasses a wide range of implementations, from very tightly coupled memory and processor usage to very loosely-coupled distributed computing systems. The term *coupling* (in this context) refers to the latency, or speed, with which the parallel tasks are performed. At the far end of the spectrum, vector computers[2] utilize fine-grained data parallelism and are on the very tightly coupled end of the scale. The original supercomputers, such as those produced by Cray, were part of this model and vector technology is still utilized by some high performance systems.

The term ***processor***, or ***microprocessor***, refers to the central processing unit (CPU). It's a single chip responsible for the execution of instructions given by a computer program, memory management and address translation, integer and floating point operations, and cache management. ***Uniprocessors*** have a single processor. Application instructions and hardware calls are executed in order, one at a time (*sequentially)*. The system appears to run concurrent processes, but the processor actually switches back and forth between instructions.

Symmetric multiprocessing (SMP) is the tightly-coupled process of program tasks being shared and executed, in true parallel mode, by multiple processors who all work on a program at the same time. Typically, these are large, single units with multiple processors that utilize shared memory, I/O resources, and, of course, a single operating system. The term SMP is so closely associated with shared memory that it is sometimes misinterpreted as standing for "shared memory parallel". By design, all processors in an SMP system have equal claim on the shared resources time. SMPs are closer to the tightly-coupled end of the spectrum. Because these systems share memory, they are vulnerable to processing bottlenecks.

Comparatively, massively parallel processing (MPP), in its most common form, is a loosely-coupled process of program tasks being shared and processed by multiple processors. The distinction is that each processor, with its own operating system and memory, works in a coordinated manner to process the different parts of the program. This is accomplished with the help of message passing libraries and

Vector computers are so named because the CPU is designed to be able to perform operations on a vector of numerical values simultaneously subroutines, such as those specified by MPI or PVM[3]. This variety of supercomputer avoids the memory bottleneck problems by having memory that is dedicated to each processor, but adds a layer of complication for programmers who write the parallel applications that utilize them.

Note: There are also large systems from traditional supercomputer vendors like Cray, SGI and IBM which have hundreds or thousands of processors in a single cabinet. These are also sometimes referred to as MMPs (massively parallel processors), but the key difference is that the term MPP, in this context, refers to **processor** (*noun*), as opposed to **processing** (*verb*). These systems usually use shared memory and are much closer to large SMP systems in our system of classification.

The distributed computing tasks of the hardware need to match the program, whose tasks must be broken up in a way that "messages" can be passed between the individual processors. MPPs are actually more multicomputer (with individual memory, processor and OS) than multiprocessor (multiple processors that share the memory and OS).

Related to massively parallel processing (*though not massive in the traditional sense*) is the distributed computing of grids and NOWs (*network of workstations*). In many cases, the participating "resources" are locally or globally scattered computer systems that have agreed to share their resources through commercial or non-profit entities that act as a coordinating host. These individual node workstations run the gamut of processor speeds, installed RAM, installed OS, and hardware manufacturers.

The classic example is the **SETI@Home** project (http://www.seti.org/seti/other_projects/seti_at_home.html), where an organization that is searching for extraterrestrial intelligence utilizes the unused processing cycles of Internet connected computers to process their project data while the thousands of node computers sits idle. But NOWs don't have to be global in scope – they can also be workstations spread across a local LAN. The distinguishing factor in this sort of distributed computing is that there is no guarantee of network speed or latency between systems. For this reason, they must be designed with a minimal amount of intersystem communications.

[3] Message Passing Interface and Parallel Virtual Machine.

Cluster computing, as used at Tufts, falls in the middle of the two extremes between the supercomputer MPPs and the globally or locally scattered NOWs. Cluster01 is closer in style to a compute farm or a Beowulf-style cluster.

SECTION

2

The Elements of a Cluster System

A quick search of the web for the term “cluster computing” will reveal that the concept and practice of clustering is causing a quiet revolution in the computing industry. Universities and research centers around the world are investing in the technology, setting clusters up, interconnecting as grids, teaching courses on clusters and parallel programming, and creating sections on the web for their support. It is clear that any educational institution that doesn’t want to be left in the dust needs to dedicate time and resources to this next wave of computing technology to stay ahead of the curve and maintain institutional credibility.

This said, one would expect to find (but doesn’t) volumes of documentation about cluster and parallel computing. Documentation is scarce because every cluster system has its own, unique identity based on the needs of its audience. Although the details that compose each individual cluster system vary, there are some overarching elements common to all:

- **Hardware and physical structure of the cluster**
- **Administration and management of the resources**
- **Parallel algorithms, libraries, and related tools**

Hardware and Physical Structure

The center of the hardware and physical structure of a cluster is the individual nodes that make up the system. As mentioned earlier, the nodes can be composed of "junk" PCs, commodity off the shelf (COTS) systems, or customized configurations that include multiple processors and reserved banks of memory. Some clusters utilize diskless workstations. But unless the cluster is a composed of a geographically dispersed network of workstations (NOW), the individual nodes have no need for audio, high-end video, printer, joystick, or many of the things a home or office user requires. For example, the Tufts cluster nodes have the capacity for attaching a mouse, keyboard, or monitor, but with remote administration of the nodes from the management node, even these basic peripherals aren't needed and aren't used. All management is done over the network or with specialized hardware attached to serial ports.

What does a compute node need? The common basic elements are:

Microprocessor –

The CPU is the home of the computing power of the node. It includes floating point and integer arithmetic logic units, pipeline and control logic, and high-speed on chip cache that holds buffer data and instruction sets from the memory. The processor interacts with the rest of the system through 2 external buses (memory and I/O).

Memory –

The memory stores the data and programs during their execution. Read/writes to memory are much slower than to cache but are still faster than disk access (virtual memory).

I/O capabilities (motherboard) –

The motherboard not only houses and integrates the other components in this list, but it also includes it's own, powerful chip set that co-ordinates the communication between the operating system and the hardware. In addition, the ROM chip contains the system BIOS and the motherboard carries the I/O ports for the various peripherals.

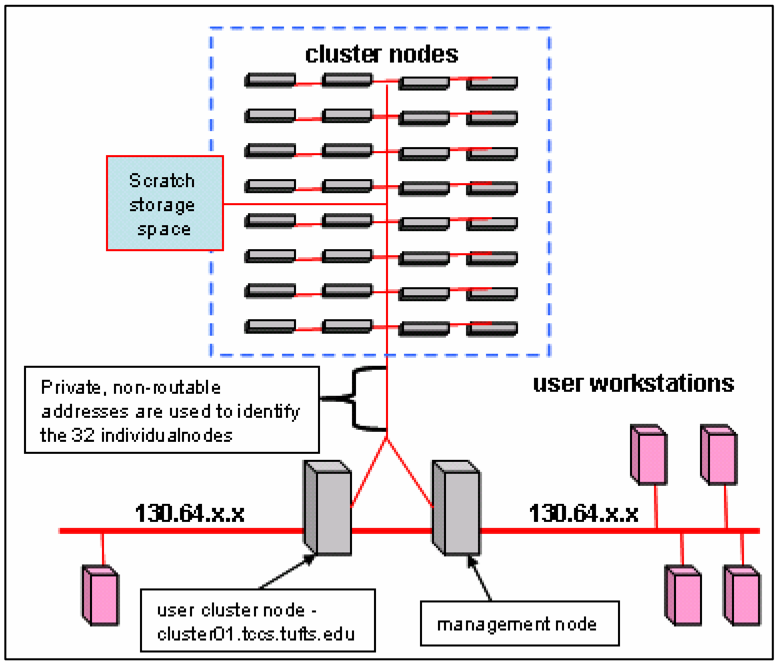
Secondary storage (hard drive) –

This entry is the only one that is optional. We mentioned earlier that diskless workstations can be utilized as nodes and actually run more efficiently with out an OS installed. The draw back here is that with out a hard drive, you can't use applications that require a node installation. You also can't take advantage of virtual memory usage or storage of your computational results. While access to hard drives is slower when compared to RAM access, it is usually much faster to read/write large files on a local hard drive than to access a network drive or file system.

NIC (NETWORK INTERFACE CARD) –

Without the NIC, there would be no communication between the individual nodes or the users. The user and management nodes commonly have 2 NICs – one to communicate with the compute nodes on the private network and one to communicate with the external network.

Tufts Cluster Nodes

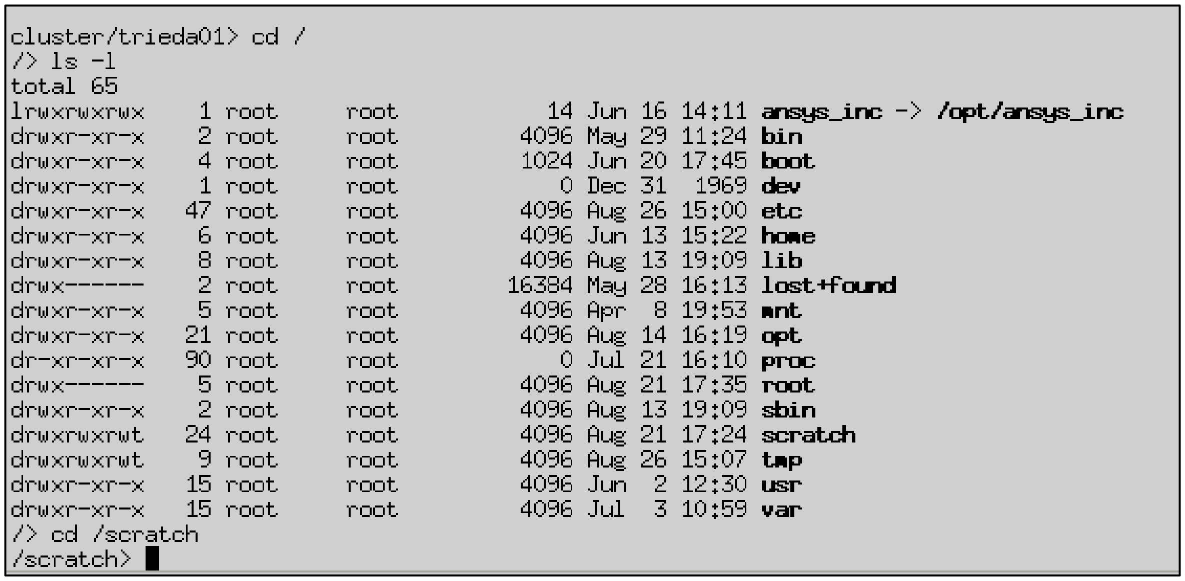

cluster nodes
Scratch storage space
Private, non-routable addresses are used to identify the 32 individual nodes
user workstations
130.64.x.x
130.64.x.x
user cluster node - cluster01.tccs.tufts.edu
management node

The Tufts Linux Research Cluster is currently comprised of 32 identical Linux systems (*compute nodes*) interconnected via a Gigabit Ethernet network. Each cluster node has a pair of 2.8 Ghz Intel Xeon CPUs and 4 gigabytes of memory. The Linux operating system on each node is configured identically across every machine.

The client workstations access the cluster via the 130.64.0.0 LAN. The user node, however, has an additional NIC that connects to the compute nodes using private non-routable IP addressing. This scheme allows the compute nodes to be a *virtualized* resource, abstracted away behind the user node. The cluster can scale up to any number of nodes without the users noticing anything other than the fact that their jobs are completing faster. This ability will be invaluable over time as usage demands increase.

Additionally, future cluster01 nodes are not restricted by current hardware and software configurations. The cluster can, for example, combine and connect nodes with different operating systems, nodes with large amounts of memory, nodes with more than 2 CPUs, and nodes that run on 64-bit architectures, such as Opteron, Itanium and Sparc.

The management node is a second server that is identical in hardware configuration to the user node. It's used for administrative and maintenance tasks and is invisible to the cluster users. Scratch storage space (*always located at the path of **/scratch/***) is provided for temporary storage of compiled data. Users are advised to treat the scratch directory as temporary storage because it is not backed up, not intended as a place for permanent storage, and will be periodically cleaned out. Important data should be saved in the user home directory, as it is on Amber.



```
cluster/trieda01> cd /
/> ls -l
total 65
lrwxrwxrwx    1 root     root           14 Jun 16 14:11 ansys_inc -> /opt/ansys_inc
drwxr-xr-x    2 root     root         4096 May 29 11:24 bin
drwxr-xr-x    4 root     root         1024 Jun 20 17:45 boot
drwxr-xr-x    1 root     root            0 Dec 31  1969 dev
drwxr-xr-x   47 root     root         4096 Aug 26 15:00 etc
drwxr-xr-x    6 root     root         4096 Jun 13 15:22 home
drwxr-xr-x    8 root     root         4096 Aug 13 19:09 lib
drwx------    2 root     root        16384 May 28 16:13 lost+found
drwxr-xr-x    5 root     root         4096 Apr  8 19:53 mnt
drwxr-xr-x   21 root     root         4096 Aug 14 16:19 opt
dr-xr-xr-x   90 root     root            0 Jul 21 16:10 proc
drwx------    5 root     root         4096 Aug 21 17:35 root
drwxr-xr-x    2 root     root         4096 Aug 13 19:09 sbin
drwxrwxrwt   24 root     root         4096 Aug 21 17:24 scratch
drwxrwxrwt    9 root     root         4096 Aug 26 15:07 tmp
drwxr-xr-x   15 root     root         4096 Jun  2 12:30 usr
drwxr-xr-x   15 root     root         4096 Jul  3 10:59 var
/> cd /scratch
/scratch> █
```

It should also be noted that **/scratch** on each system is unique. Putting a file in **/scratch/username** on the user node will not make it available to any of the compute nodes. Likewise, files put in **/scratch** on one compute node are not visible elsewhere. This topic will be covered in greater detail when we discuss logging in and using the cluster.

Additionally, applications that work on Amber will not run on the cluster, due, in part, to different binary formats. Programs compiled on Amber will need to be recompiled on the cluster to take advantage of linkage to the LSF libraries and helper files. If you have questions about application compatibility, contact the cluster administrator by sending an email to: cluster01-support@tufts.edu.

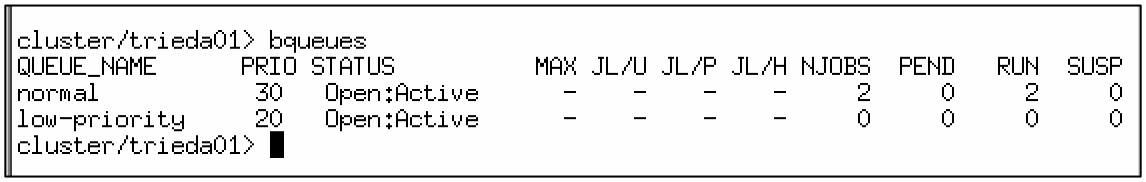
Administration and Management

We've discussed the hardware resources – now, it's time to consider what tasks of resource management and administration a cluster needs. One of the more attractive qualities of working on a cluster is that the user can choose to let the cluster nodes be as invisible as they like. They log in to the cluster, access their application and submit their job, unaware of the size, complexity, or resource requirements of the other jobs running on the cluster. Administrators can also choose to hide various processes or limit access to the individual nodes or functions of the cluster. All of this behind the scenes action needs a system to co-ordinate and orchestrate the administration and management of these resources. As is often the case, these tasks fall to one or a combination of several software applications. There are many applications options available, both open source and for pay, and some cover more resource tasks than others. But before we examine the software the Tufts cluster uses, let's detail what the general tasks required by the resource management system are.

JOB QUEUING

A job queuing system is required to buffer the jobs as they're submitted to the nodes. Think of the door person at a night club who lets you through the door when space becomes available.

Note: There are multiple job queues available on the cluster. User jobs will go to the default queue, unless specifically directed elsewhere.



```
cluster/trieda01> bqueues
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
normal           30   Open:Active      -    -    -    -     2     0     2     0
low-priority     20   Open:Active      -    -    -    -     0     0     0     0
cluster/trieda01> 
```

SCHEDULING

A scheduler has the complicated task of balancing job requirements and priorities as well as the system resources and policies set by the administrator. Does a large job need all the nodes at once? Can it run on one node during an off time? Does it require user input (interactive) or a higher priority? The scheduling system needs to take these needs into account and set the order of job execution.

RESOURCE CONTROL

The scheduler looks to the resource control system to know when the jobs can be placed on the nodes. It handles job movement to the nodes and the offload of results, along with job start, suspend, terminate tasks.

MONITORING

The tracking of resource availability, usage, and general health of the individual nodes must be monitored. While some of this data is followed by the administrators from the management node, this data is also important to resource control systems.

ACCOUNTING

A system is needed to track cluster and node usage by the applications and the individual users. By analyzing the accounting data, administrators can make educated decisions about policies and resource allocation. They can also use this data to anticipate the need to grow the cluster before it gets to a point of an overwhelmed and disrupted system.

Parallel Algorithms, Libraries, and Related Tools

Parallelism is defined as "the ability of many independent threads of control to make progress *simultaneously* toward the completion of a task [4]". If an algorithm is a procedure for solving a problem, parallel algorithms are procedures for solving problems that utilize parallelism. Clusters are built specifically to exploit parallelism and as a platform for parallel algorithms. As such, the applications that run on the cluster need a reasonable level of parallelism programmed into them. Serial applications can certainly be run on a cluster, but if an application can not execute tasks in parallel, it might as well run on an independent workstation.

There are many pre-existing applications that are programmed for parallel execution, such as those installed on the Tufts Cluster, and developers are beginning to build parallel functionality into their applications as a matter of course. Some users of the cluster may wish to develop their own code. To this end, the GNU compiler suite (C, C++, Fortran77, and the Gnu Java Compiler) has been installed. CMU's Common Lisp has been installed for those who prefer to use a functional programming paradigm. In addition, Tufts has licensed the Portland Group Compilers (C, C++, Fortran77, Fortran90, and HPF) for those who require Fortran90 or who already have a code base in the Portland Group dialects. Lastly, the Sun Java Development Kit is installed. Developers writing Java for use on the cluster, however, are encouraged to investigate the gcj java compiler, as this generates native binary executables for Linux, which are often much faster and less memory intensive than execution under the JRE.

The cluster also has Perl and Python installed and configured. These scripting languages are most useful for automating input and output processing where traditional languages are overkill. They are also appropriate for controlling complex workflows that include many dependencies and which require complex error checking. Users are advised, however, to not write their core applications in a scripting language, as they have higher overhead requirements and numerical processing tends to be much slower than C or Fortran.

[4]Page 145 in How to Build a Beowulf – A Guide to the Implementation and Application of PC Clusters , Thomas Sterling, John Salmon, Donald J. Becker, and Daniel F. Savarese, © MIT Press, 1999

Libraries

Specialized libraries also play a large role in the development of software for high-performance computing. To that end, Atlas, GSL, and GMP are available for numerical subroutines.

ATLAS -

Atlas is a tuned implementation of the BLAS reference standard with some additional LAPack routines.

GSL -

GSL is the Gnu Scientific Library, which is an extensive set of hand-coded routines for numerical and scientific computation oriented towards C developers.

GMP -

GMP is the Gnu Multiple Precision library, which is used to allow software to work exactly with very large and very small integers, rationals, and reals.

Many different parallel programming models have been put forward, but the most common, particularly on Linux clusters, is the message passing model. The two most popular message passing specifications are **PVM** (*Parallel Virtual Machine*) and **MPI** (*Message Passing Interface*). Both are specifications for libraries and utilities (or *APIs*) which can be used with programming languages, such as Fortran, C,and C++. MPI, which is newer than PVM, has several technical advantages and has been accepted as an open standard. There are 2 main implementations of MPI – LAM and Mpich, both of which are Open Source packages. The Tufts cluster uses LAM as its default MPI implementation. Mpich can be installed and used alongside of LAM, but there are some stability issues. PVM, if needed, can also be installed by a user, but its use is being deprecated. Contact a cluster administrators at cluster01-support@tufts.edu for more information.

The web is an excellent resource for information and tutorials that can get you started on the road to parallel programming. Parallel Programming - Basic Theory For The Unwary, at http://users.actcom.co.il/~choo/lupg/tutorials/ provides a nice introduction to the basics. Additional links to web resources are listed in Chapter 3 of this manual.

Platform LSF

Are you still wondering what application the Tufts cluster uses for resource management and administration? Tufts has chosen to use Platform Computing, Inc.'s Platform **LSF** (***Load Sharing Facility***) software, which is a distributed load sharing and batch queuing suite of applications that can dispatch user requests to compute nodes in accordance with a Tufts-defined policy. It manages, monitors, and analyzes resources and load in the cluster. Platform LSF is layered in a way that allows it to sit on top of and extend the operating system services, speaking to the competing needs of resource management on the cluster. As of May, 2001, it integrated with as many as 15 varieties of UNIX, Windows NT, and Windows 2000.

The LSF base product provides user access to dynamic load-balancing, load-sharing, and job queuing. Other layers of the suite include LSF JobScheduler, LSF Make, LSF Global Intelligence (which allows for complex analysis of system activity data), LSF MultiCluster, and Platform HPC. Platform HPC includes the Parallel Application Manager (PAM) and integration with a large set of numerical and scientific computing applications. We will detail the LSF commands that you use to run and monitor your jobs later in this manual.

With Platform LSF, users of the cluster can define what resources they need for a given problem (number of CPUs, special software licenses, CPU time, disk space, memory, etc.) and let resource management facilities of LSF determine where the job should run and when. If the cluster becomes overloaded, Platform LSF acts as a "traffic controller", ensuring that the work continues to flow without a system crashing. For programs written with MPI calls, Platform HPC provides parallel application management (PAM) and scripts that integrate with the MPI libraries and binaries. These manage the execution of the code and provide housekeeping services, such as assigning CPUs to the program when it starts and graceful termination in the event of an error.

SECTION 3

Compute Node Application Overview

All of this hardware and resource management software is interesting from a computer architecture standpoint, but without user applications, it is not much use to the research community. Initially, software for Mechanical Engineering, Computational Chemistry, and Electrical Engineering has been installed. R, gridMathematica and Matlab are also available. These are as much specialized programming languages for mathematical analysis as they are ordinary applications.

An in-depth discussion of the compute node applications available on the cluster is beyond the scope of this introductory manual. It is also anticipated that the list will grow and change to match the needs of cluster users. But to get you started, several of the cluster applications, accompanied by a brief overview, are listed below:

Cluster Applications

BioPerl (http://www.bioperl.org/)

Bioperl is a toolkit of perl modules useful in building bioinformatics solutions in perl. It is built in an object-oriented manner so that many modules depend on each other to achieve a task. The collection of modules in the bioperl-live repository consist of the core of the functionality of bioperl. Additional information and documentation is available at:http://www.bioperl.org.

R (http://www.r-project.org/)

R is a widely available object oriented statistical package. The current list of installed packages can be found in directory */usr/lib/R/library/*. This represents a base installation suitable for most routine tasks, however not all available packages as found on the website are installed. If some other package is needed, please make an email request to cluster01-support@tufts.edu. User documentation is also available on the website.

Maple (http://www.maplesoft.com/)

Maple is for mathematical problem-solving, exploration, data visualization, and technical authoring. It is similar to Mathematica and Matlab. Additional information may be found at http://www.maplesoft.com/.

Matlab (http://www.mathworks.com/)

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran. Over 12 additional Matlab toolboxes are available.

Materials Studio (http://www.accelrys.com/mstudio/index.html) –

Materials Studio® is a validated software environment that brings the world's most advanced materials simulation and informatics technology.
It dramatically enhances your ability to mine, analyze, present, and communicate data and information relating to chemicals and materials. Materials Studio's accurate prediction and representation of materials structure, properties, and inter-relationships provides valuable insight. The following Materials Studio products are available: CASTEP, DMol.

Fluent (http://www.fluent.com/) –

Fluent is a Computational Fluid Dynamics (CFD) software package commonly used in engineering education for research in fluid mechanics. The Fluent University Program provides universities with special, low-cost (90% discount for annual department or university site licenses) access to many of Fluents full-featured general use products, including FLUENT, FIDAP, POLYFLOW, Icepak, and Airpak. Each package includes a preprocessor, solver, and postprocessor. Training is also available, with support provided via email and web-based user services for licensed users.

Abaqus (http://www.hks.com) –

Abaqus is a suite of applications used by many in the engineering community for the analysis of multi-body dynamics problems that aid the medical, automotive, aerospace, defense, and manufacturing community.

DEFORM-3D (http://www.deform.com/)

DEFORM (**D**esign **E**nvironment for **FORM**ing) is an engineering software environment that enables designers to analyze metal forming processes. DEFORM-3D is a simulation system that is designed to analyze the three-dimensional flow of complex metal forming processes, allowing for a more complex analysis of shapes than 2D models can provide.

gridMathematica (http://www.wolfram.com/products/gridmathematica/) –

gridMathematica, advertised as a "one-stop shop" for technical work, "integrates a numeric and symbolic computational engine, graphics system, programming language, documentation system, and advanced connectivity to other applications". Not only does this application have parallel functionality built into it from the ground up, the www.wolfram.com web site has extensive documentation, including a terrific parallel computing toolkit with detailed tutorials - http://documents.wolfram.com/applications/parallel/.

Ansys –

Ansys is a suite of applications that provide real-world simulations of structural, thermal, electromagnetic and fluid-flow behavior of 3-D product. All Ansys products integrate with CAD environments.

Stata -

Stata is an integrated statistical package for Windows, Macintosh, and Unix platforms. More than just a statistical package, Stata is also a full data-management system with complete statistical and graphical capabilities. It features both X-window and text user interfaces.

Octave –

GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language. See http://www.octave.org for more details.

Gaussian –

Gaussian 03 is used by chemists, chemical engineers, biochemists, physicists and others for research in established and emerging areas of chemical interest. Starting from the basic laws of quantum mechanics, Gaussian predicts the energies, molecular structures, and vibrational frequencies of molecular systems, along with numerous molecular properties derived from these basic computation types. See http://www.gaussian.com for details.

Femlab Software –

The FEMLAB Chemical Engineering Module is the perfect tool for process-related modeling. It is specifically designed to easily couple transport phenomena-including computational fluid dynamics (CFD) as well as mass and energy transport-to chemical-reaction kinetics. More information is available at http://www.comsol.com/.

Modules licensed:
Chemical Engineering

Invocation on Cluster01:
```
> femlab
```

Documentation viewer:
```
> femlab -doc
```

NWchem –

NWChem is a computational chemistry package that is designed to run on high-performance parallel supercomputers as well as conventional workstation clusters. It aims to be scalable both in its ability to treat large problems efficiently, and in its usage of available parallel computing resources. See http://www.emsl.pnl.gov/docs/nwchem/nwchem.html
for details.

***NOTE:** Users of these following applications do not need to explicitly use **bsub** to start up interactive sessions. They can be started from the command line on the user node by typing the appropriate command at the prompt.*

The following compilers are also available:

Portland Group:

pgf77 - Fortran 77 compiler
pgf90 - Fortran 90 compiler
pgCC - C++ compiler
pgcc - Ansi and K&R C compiler

Documentation can be found at:
http://www.pgroup.com/
http://www.pgroup.com/resources/docs.htm

Manpages:

```
> man pgCC
```

Gnu Project compilers:

gcc GNU project C and C++ compiler
g77 GNU project Fortran 77 compiler
doumentation under manpages or via web:
Manpages:

```
> man gcc
> man g77
```

http://gcc.gnu.org/onlinedocs/

Intel:

ifort - Intel fortran compiler
icc - Intel C compiler

PDF and html Fortran documentation located at:
/opt/intel/compiler81/doc/

Fortran quick reference:

```
> ifort -help | more
```

Manpages:

```
> man ifort
```

Pdf and Html C documentation located at:
/opt/intel/compiler70/docs/c_ug/

CHAPTER 2

Using the Cluster

Getting an Account

Access to the cluster requires a Research Cluster account. Faculty, other staff, and students are required to fill out a UIT Research Computer Account Application form, available at the following locations:

Boston Campus:

- Learning Resource Center (HSL, Sackler Building, 5th floor)

Medford Campus:

- Contact the University IT Support Center at x7-3376 or by email at uitsc@tufts.edu for further assistance

Grafton Campus:

- Veterinary Library

> ***Note:*** Students must have their application form signed by a faculty member or advisor.

You can also download a copy of the account form at http://www.tufts.edu/tccs/r-cluster2.html.

ACCOUNT EXPIRATIONS

Undergraduate students must revalidate their accounts by completing a new application twice a year, in May and December.

Graduate students, faculty, and other staff must revalidate their accounts once each year, in May. To avoid interrupted access, it's a good idea to renew your access in advance of the expiration date.

Note: Your school may have other restrictions or requirements.

Account Use

The cluster is a new service that adheres to the standards of the scientific and research community at large and is intended for use in support of research activities requiring high-performance computation. It is not intended as a general-purpose shell account server. E-mail, printing, FTP, and web publishing are not available. Use of the cluster for personal backups or attempts to interfere with other user's processes, data, or to circumvent the system will be grounds for revocation of the account or other disciplinary action.

HOME DIRECTORY/ENVIRONMENT ISSUES (AMBER USERS ONLY)

Amber and the cluster use the same home directories. User defined "dotfiles" such as .cshrc, .login and .logout, should be set to ensure that Amber commands and variables aren't used on the Cluster, or Cluster commands and variables used on Amber. Failure to do so will result in application confusion. Users should NOT replace $PATH on the cluster, only add to it (*this also applies to non-Amber cluster users*). For more information, see man pages for **tcsh** (C shell) documentation by typing **man tcsh** <enter>.

SSH KEYS AND AGENTS

For MPI and LSF to coexist, a special ssh key is needed and is auto generated when the account is created. This key should not be used as a users Identity on any other system outside the cluster. Users who use ssh agents to simplify remote access should contact cluster01-support@tufts.edu for special information about integration.

Account Support

Research cluster account issues can be addressed by sending an e-mail to cluster01-support@tufts.edu. Some of the applications installed on the cluster require special procedures to take full advantage of the cluster environment. For more information, contact cluster01-support@tufts.edu. Requests to obtain and install new applications for use on the cluster should also be directed to cluster01-support@tufts.edu. These requests will be evaluated and a decision made based on cost and available staff resources.

Responsible Use of Your Account

By using this account (*and any other University system*), you agree to and are bound by the terms of the University's Responsible Use Policy. You are urged to review the terms of this policy, which apply to all members of the University community.

As stated in the policy, you are responsible for any and all activity regarding your account. Do not share your password with anyone - there is absolutely no reason for anyone else to have knowledge of your password or to use your account. Violation of the Responsible Use Policy may result in disciplinary action and the loss of account privileges.

For more information, contact cluster01-support@tufts.edu.

Additional Cluster Usage Considerations

- Faculty, staff, and students are constrained to approved scientific and research project use of the cluster
- Telnet, FTP, and rsh *will not work* for connecting to this system. Applications that *will* connect are SSH–based. Examples include SecureCRT, SecureFX, and Putty.
- Cluster compute nodes are the only targets under LSF control. Jobs can not be submitted to Amber or anywhere else.
- Scp and sftp are available for file transfer. Secure CRT users can also use vcp.exe *(see Appendix B for usage)*.
- If you are already using Amber, your home directory is the same as your existing Amber home directory.

- If you plan to use both Amber and the cluster, some form of housekeeping will help to keep your work separate; such as directories and file naming conventions.
- Selection of an individual node to exclusively run your program is forbidden without prior approval by the cluster administrator.
- Do not leave your session window open for long periods of time when you are away from your desk.
- Do not use SSH or other session tools to connect directly to a specific node. Repeated instances may result in the loss of account privileges. If you feel you have a valid need to do so, contact the cluster administrators at cluster01-support@tufts.edu.
- Chat features, e-mail, printing, TeX, and X-enabled versions of emacs are not available on the cluster.
- Researchers with extremely large disk storage needs are asked to contact cluster01-support@tufts.edu for suggestions about how your needs may be accommodated on the cluster. Several types of temporary storage are available, but the choice depends on the application(s) and workflow.

If you need to have files restored, contact the cluster administrators at cluster01-support@tufts.edu. Best effort will be made to restore your files or data. *There is no promise that files from deleted accounts can be restored*. Users of the cluster are strongly encouraged to take responsibility for the protection of their files and data.

SECTION
1

Getting Connected

SecureCRT Installation Instructions

You can obtain a copy of SecureCRT from the UIT website, at http://www.tufts.edu/tccs/s-securecrt.html. If you already have SecureCRT installed, you may wish to upgrade to the latest version (your icon indicates what version you are using).

Licensed copies of SecurtCRT are available for download from the 130.64.0.0 Tufts network. This copy of the application includes a pre-configured Tufts Research Cluster session (*cluster01.tccs.tufts.edu*). If you do not have access to the LAN or are off-campus, you can download a copy directly from www.vandyke.com and follow the instructions located at http://www.tufts.edu/tccs/OffcampusCRTstep1.html to obtain a license. Instructions for configuring your cluster session are located in Appendix C of this manual.

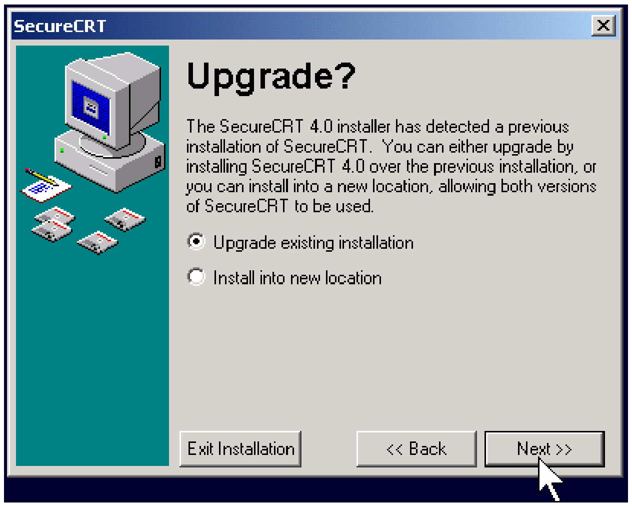
The following instructions detail an upgrade from version 3.5 to 4.0.8 on a Windows-based PC:

Download the file to a location on your hard drive that you can easily remember. Close all open applications, use Explorer to locate the file, and double-click on it. The application installation will begin.

1.) Click **I Agree** at the *SecureCRT License Agreement* screen.
Click **Next** at the *Welcome* screen.

2.) If you have a previous version installed, SecureCRT will detect it and give you the option to either upgrade your current copy or install the new version to a separate location. Selecting *Upgrade existing installation* will save any current session information. This is the recommended choice.

Accept the default upgrade option and click **Next**.


SecureCRT

Upgrade?

The SecureCRT 4.0 installer has detected a previous installation of SecureCRT. You can either upgrade by installing SecureCRT 4.0 over the previous installation, or you can install into a new location, allowing both versions of SecureCRT to be used.

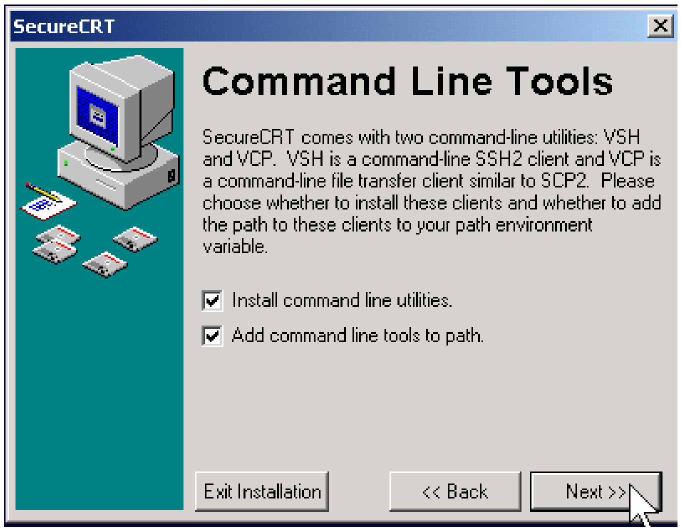
● Upgrade existing installation

○ Install into new location

3.) At the *Select Profile Options* screen, accept the default settings and click Next.

4.) At the *Choose Protocols* screen, accept the default settings (all protocols selected) and click **Next**.

5.) At the *Command Line Tools* screen, accept the default settings (both options selected) and click **Next**.


SecureCRT

Command Line Tools

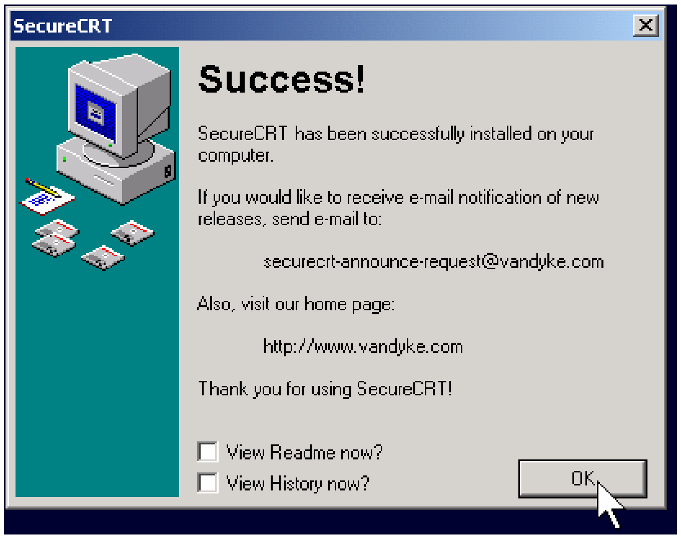
SecureCRT comes with two command-line utilities: VSH and VCP. VSH is a command-line SSH2 client and VCP is a command-line file transfer client similar to SCP2. Please choose whether to install these clients and whether to add the path to these clients to your path environment variable.

☑ Install command line utilities.

☑ Add command line tools to path.

6.) At the *Ready To Install!* screen, click **Finish**. The installation will begin.

At the *Success!* screen, deselect the options (*View Readme now?* and *View History now?*) and click **OK**.



SecureCRT

Success!

SecureCRT has been successfully installed on your computer.

If you would like to receive e-mail notification of new releases, send e-mail to:

securecrt-announce-request@vandyke.com

Also, visit our home page:

http://www.vandyke.com

Thank you for using SecureCRT!

☐ View Readme now?
☐ View History now?

SECTION 2

Commands

UNIX Class Concepts

***NOTE:* All LSF and UNIX commands are case sensitive.**

Unix is a *class of operating systems* that are based on the original UNIX code. There are many implementations and derivations. The Unix-like family includes Solaris, SCO UnixWare and OpenServer, MacOS X, and BSD/OS. Free implementations include FreeBSD, OpenBSD, and Linux. Unix commands are used for navigating your cluster home and directories. The jobs that you run on the cluster, however, are controlled and monitored with LSF commands.

The concept of Unix operation is simple. The core of the operating system is called the kernel, which is really just a program that loads when you turn the system on. The kernel can be customized and pared down to suit your needs. Linux , for instance, can be reduced to less than a megabyte and still function as an operating system.

There is a core group of commands that you run from outside of the kernel at the shell (walnutty, isn't it?). The shell is the command interpreter and is the prompt you see when connected (not necessarily by a network) with a Unix system via a monitor and keyboard. You type in commands that the shell program "interprets" and executes. It passes it to the kernel, which runs the process* and returns the results to the shell, which displays them for you.

** Programs are called a process when run by the kernel*.

If you've logged enough time in the computer industry to remember DOS 3.3 and 5 ½ low density floppy disks, the openness of UNIX logic and its attendant commands feel like an old friend. And LSF has a similar feel – direct, no nonsense, and a little intuitive. We'll share a few new-user tips, review some basic commands and flags, and then use what we've discussed to run your first cluster job.

Finding Your Way

If you're new to Unix flavor operating systems, you may find the command prompt a little daunting when you're trying to figure out where everything is. Here are a few tips to help you orient yourself:

Note: blue type *is the output from the preceding command.*

List your current directory with the **pwd** command:

```
cluster/trieda01> pwd    <enter>
/home/cluster/trieda01
```

View what is in the current directory with the **ls** command:

```
cluster/trieda01> ls    <enter>
parallel-example
```

Move up one directory level with the **cd ..** command:

```
cluster/trieda01> cd ..    <enter>
/home/cluster>
```

Alternate the **cd ..** and **ls** commands to continue exploring:

```
/home/cluster> ls   <enter>
across01  afant01  dpacker  jstile02  mryan01  scarpe01  tgangw01
afant     dag      durwood  klibby01  sault    skritz01  trieda01

/home/cluster> cd ..    <enter>
/home>

/home> ls   <enter>
cluster  httpd  management  quarry

/home> cd ..   <enter>
/>

/> ls    <enter>
ansys_inc  boot  etc  lib        mnt  proc  sbin    tmp  var
bin        dev   home  lost+found  opt  root  scratch  usr

/> cd /scratch   <enter>
/scratch>
/scratch> ls    <enter>
lost+found
```

And, finally, return home:

```
/scratch> cd /home/cluster/trieda01   <enter>
cluster/trieda01>
```

You can also go directly to the root by typing **cd /** *<enter>*.

Shell Tips

The following tips apply to both Linux/Unix and LSF:

- Pressing the spacebar will page through multiple screens worth of information, such as the output generated with the man command.

GETTING HELP

- **man <*options*><*command*>**

  example: **man ls** *<enter>*

  One of the most important and useful commands to know is the one that displays help information. **man**, used in conjunction with a command, will display the manual that details the commands usage and options. You'll get volumes of text with this command, so don't forget your spacebar tip.

  Type **q** *<enter>* to exit the manual information

You can also add a **–k** option to the **man** command to get a short synopsis:

example: **man –k passwd** *<enter>*

TO FIX MISTAKES:

- The Backspace key or Delete will remove the last character
- Hitting ctrl-w will remove the last word
- Hitting ctrl-u will remove the last line

NICE TO KNOW...

- ***./filename*** and ***./program*** mean *in the current directory*. **../** would point an action up 2 directories to find the file or program.

- A **./c** command will display everything in your current directory (the one you are in at that time) that starts with "**c**"

  Example:

  Inputting this command: *trieda01/parallel-example>* **./c** <enter>

  Displays this: calculatePi* cpi.c*
  trieda01/parallel-example>

- To change directories:
  cd .. = move up one directory
  cd . = move up to root

- Use the up arrow key to cycle through the commands that you've already typed to save yourself from retyping them.

- You can auto complete your commands by typing the first few letters and hitting the Tab key

  cluster/trieda01> **bjo** *<hit Tab>*
  auto completes to:
  cluster/trieda01> **bjobs**

UNIX Commands

Unix commands, like DOS commands, follow a syntax:

> **command (flags) argument1 argument2 …**
>
> **command** – the program you're calling, such as **ls**, **cd**, or **man**.
>
> **flag** – extra tasks you want performed; flags are sometimes optional and for that reason, are often referred to as "options". They are usually preceded with a dash (-). The **–l** flag after the **ls** (*list*) command tells the command interpreter that you want the **l**ong version of the list.
>
> **argument1** – the file, directory or program you want the command to act upon. If no argument is used, the command interpreter will act on the directory you're currently in.
>
> **argument2** – used if needed, as with the **cp** (copy) command:
> **cp weezil.txt weezil.bak**

Note that additional arguments can be added, such as when copying a large number of files to a single directory.

As it is with LSF, the art of Unix commands lies in the usage of flags and options. Use the **–k** flag and **man** command to explore usage options to get the most out of these commands:

Action Needed	**Command**	**Usage**
Display contents of file	**cat**	cat *filename*
Copy a file	**cp**	cp [-option] *source destination*
Change file protection	**chmod**	chmod *mode filename* chmod *mode directory_name*
Change working directory	**cd**	cd *pathname*
Display file (/w pauses)	**more**	more *filename*
Display first page of text	**less**	less *filename*
Display help	**man**	man *command or* *man –k topic*

Rename file	**mv**	mv [option] *filename1 filename2* *directory1 directory2* *filename directory*
Compare file	**diff**	diff *file1 file2*
Delete file	**rm**	rm [option] *filename*
Create a directory	**mkdir**	mkdir *directory_name*
Delete directory with files in it	**rmdir -r**	rm -r *directory_name*
Delete directory	**rmdir**	rmdir *directory_name*
Display a list of files	**ls**	ls [option] *directory_name*
Change the password	**passwd**	Passwd <enter>
Display a long list with details	**ls -l**	ls –l *directory_name*
Display current directory	**pwd**	pwd <enter>

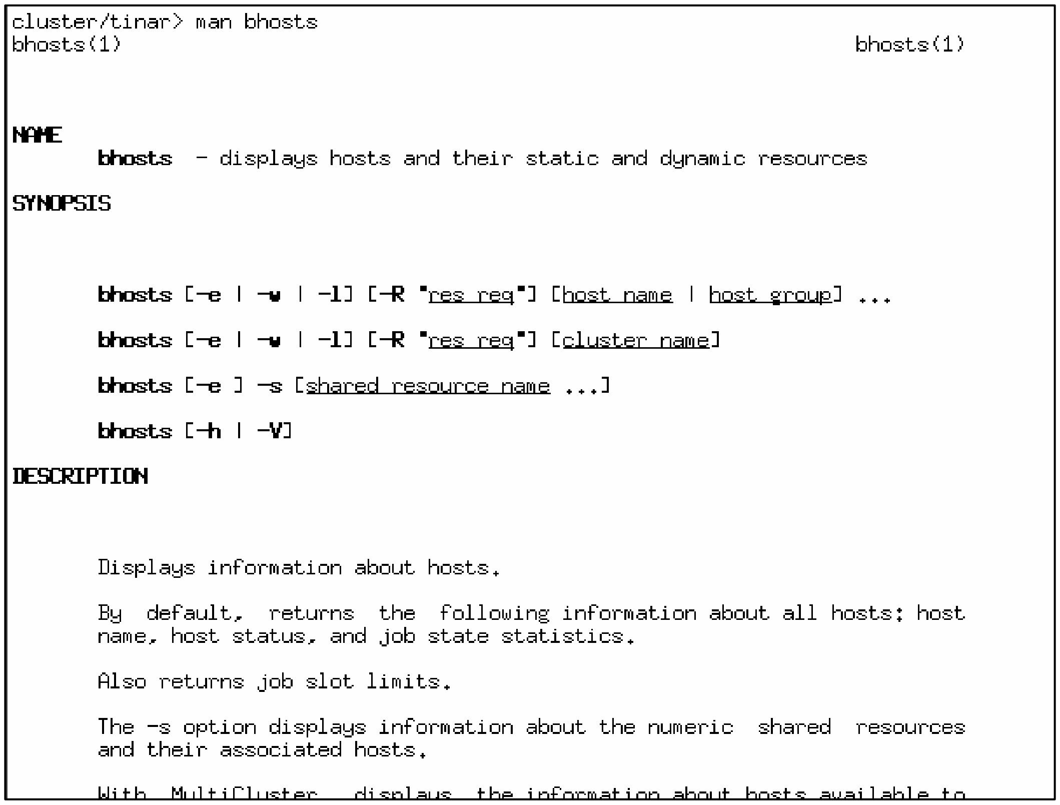
To read more about Unix commands and their usage, go to either http://www.isu.edu/departments/comcom/unix/workshop/unixindex.html or read the now classic Unix is a Four Letter Word at http://unix.t-a-y-l-o-r.com/Unix.html.

LSF (Load Sharing Facility) Commands

There are many LSF commands, but you'll tend to use the same dozen or so over and over. That's because the true nuance and art of using LSF lies in the use of switches. You can pull basic documentation up for most LSF commands by typing **man *command name***.

Example:

man bhosts *<enter>*



```
cluster/tinar> man bhosts
bhosts(1)                                                          bhosts(1)



NAME
       bhosts  - displays hosts and their static and dynamic resources

SYNOPSIS



       bhosts [-e | -w | -l] [-R "res_req"] [host_name | host_group] ...

       bhosts [-e | -w | -l] [-R "res_req"] [cluster_name]

       bhosts [-e ] -s [shared_resource_name ...]

       bhosts [-h | -V]

DESCRIPTION



       Displays information about hosts.

       By  default,  returns  the  following information about all hosts: host
       name, host status, and job state statistics.

       Also returns job slot limits.

       The -s option displays information about the numeric  shared  resources
       and their associated hosts.

       With  MultiCluster,  displays  the information about hosts available to
```

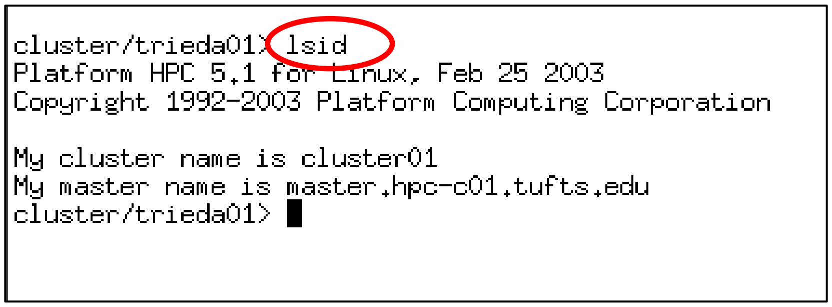
Press the spacebar to view each page. When you're done, type **q** <enter> to return to your prompt.

Checking In – Monitoring Commands

NOTE: *These commands do not need to be run each time that you log on to the cluster, but are valuable when troubleshooting problems.*

lsid

A good choice of LSF command to start with is the **lsid** command:



```
cluster/trieda01> lsid
Platform HPC 5.1 for Linux, Feb 25 2003
Copyright 1992-2003 Platform Computing Corporation

My cluster name is cluster01
My master name is master.hpc-c01.tufts.edu
cluster/trieda01>
```

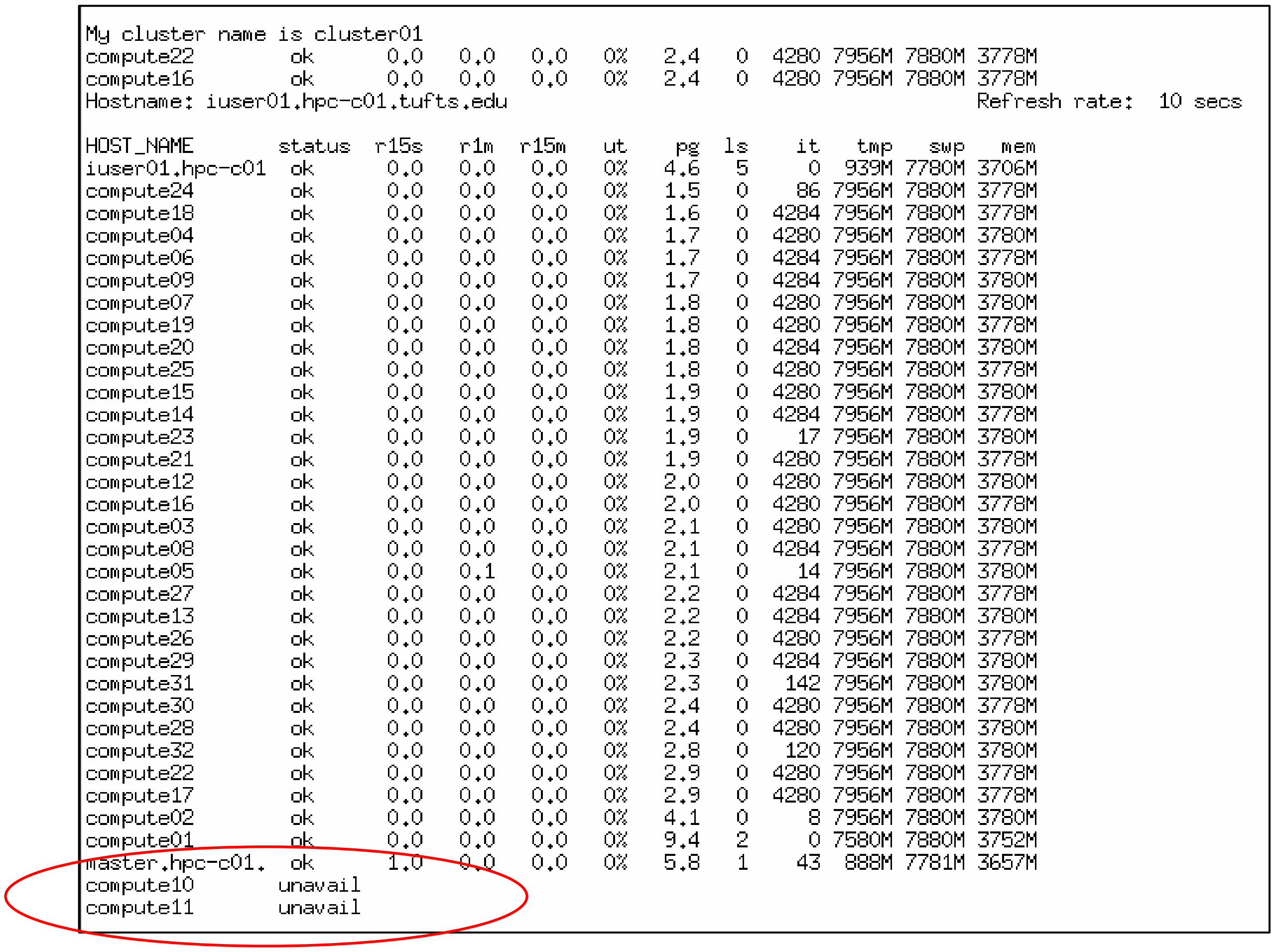
A successful response to the **lsid** command tells you that you are connected and communicating with the "master" and that LSF is working. If all is well, this command will display the version of LSF, the name of the Tufts cluster (*cluster01*), and the current master host (*master.hpc-co1.tufts.edu*).

The master machine, in a cluster, is the node that makes all of the scheduling decisions. At Tufts, the management node is also the master node. If configured to do so and the LSF master node goes down, an election is held and another node is promoted to the role of master. The new master then inherits all of the scheduling. With this Platform LSF feature, all save a few nodes can simultaneously crash and, although you would see a degraded response time, nothing will be lost and the jobs will continue to run. The Tufts cluster, however, is not configured this way and currently, the cluster can't failover and elect a new master from the compute nodes. Failover is only setup between the LSF Master and User node

lsmon

The **lsmon** command provides you with a graphic representation of the current status of the nodes. If you look at the graphic below, you will see that the first column lists the compute nodes by name. The second column, which is the most important, lists the status of each node. A status of "*unavailable*" does not necessarily mean that a node is down. *Unavailable* status can also mean that the

LSF software on that node has not been started yet. The other columns are less important and list workload on the node in increments of 15 seconds, 1 minute, and 15 minutes. The other listings include information about available temp space, swap space, and memory.



```
My cluster name is cluster01
compute22        ok       0.0   0.0   0.0   0%   2.4   0  4280 7956M 7880M 3778M
compute16        ok       0.0   0.0   0.0   0%   2.4   0  4280 7956M 7880M 3778M
Hostname: iuser01.hpc-c01.tufts.edu                                 Refresh rate:  10 secs

HOST_NAME       status   r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
iuser01.hpc-c01  ok       0.0   0.0   0.0   0%   4.6   5     0  939M 7780M 3706M
compute24        ok       0.0   0.0   0.0   0%   1.5   0    86 7956M 7880M 3778M
compute18        ok       0.0   0.0   0.0   0%   1.6   0  4284 7956M 7880M 3778M
compute04        ok       0.0   0.0   0.0   0%   1.7   0  4280 7956M 7880M 3780M
compute06        ok       0.0   0.0   0.0   0%   1.7   0  4284 7956M 7880M 3778M
compute09        ok       0.0   0.0   0.0   0%   1.7   0  4284 7956M 7880M 3780M
compute07        ok       0.0   0.0   0.0   0%   1.8   0  4280 7956M 7880M 3780M
compute19        ok       0.0   0.0   0.0   0%   1.8   0  4280 7956M 7880M 3778M
compute20        ok       0.0   0.0   0.0   0%   1.8   0  4284 7956M 7880M 3780M
compute25        ok       0.0   0.0   0.0   0%   1.8   0  4280 7956M 7880M 3778M
compute15        ok       0.0   0.0   0.0   0%   1.9   0  4280 7956M 7880M 3780M
compute14        ok       0.0   0.0   0.0   0%   1.9   0  4284 7956M 7880M 3778M
compute23        ok       0.0   0.0   0.0   0%   1.9   0    17 7956M 7880M 3780M
compute21        ok       0.0   0.0   0.0   0%   1.9   0  4280 7956M 7880M 3778M
compute12        ok       0.0   0.0   0.0   0%   2.0   0  4280 7956M 7880M 3780M
compute16        ok       0.0   0.0   0.0   0%   2.0   0  4280 7956M 7880M 3778M
compute03        ok       0.0   0.0   0.0   0%   2.1   0  4280 7956M 7880M 3780M
compute08        ok       0.0   0.0   0.0   0%   2.1   0  4284 7956M 7880M 3778M
compute05        ok       0.0   0.1   0.0   0%   2.1   0    14 7956M 7880M 3780M
compute27        ok       0.0   0.0   0.0   0%   2.2   0  4284 7956M 7880M 3778M
compute13        ok       0.0   0.0   0.0   0%   2.2   0  4284 7956M 7880M 3780M
compute26        ok       0.0   0.0   0.0   0%   2.2   0  4280 7956M 7880M 3778M
compute29        ok       0.0   0.0   0.0   0%   2.3   0  4284 7956M 7880M 3780M
compute31        ok       0.0   0.0   0.0   0%   2.3   0   142 7956M 7880M 3780M
compute30        ok       0.0   0.0   0.0   0%   2.4   0  4280 7956M 7880M 3778M
compute28        ok       0.0   0.0   0.0   0%   2.4   0  4280 7956M 7880M 3780M
compute32        ok       0.0   0.0   0.0   0%   2.8   0   120 7956M 7880M 3780M
compute22        ok       0.0   0.0   0.0   0%   2.9   0  4280 7956M 7880M 3778M
compute17        ok       0.0   0.0   0.0   0%   2.9   0  4280 7956M 7880M 3778M
compute02        ok       0.0   0.0   0.0   0%   4.1   0     8 7956M 7880M 3780M
compute01        ok       0.0   0.0   0.0   0%   9.4   2     0 7580M 7880M 3752M
master.hpc-c01.  ok       1.0   0.0   0.0   0%   5.8   1    43  888M 7781M 3657M
compute10       unavail
compute11       unavail
```

The options list for this command are:

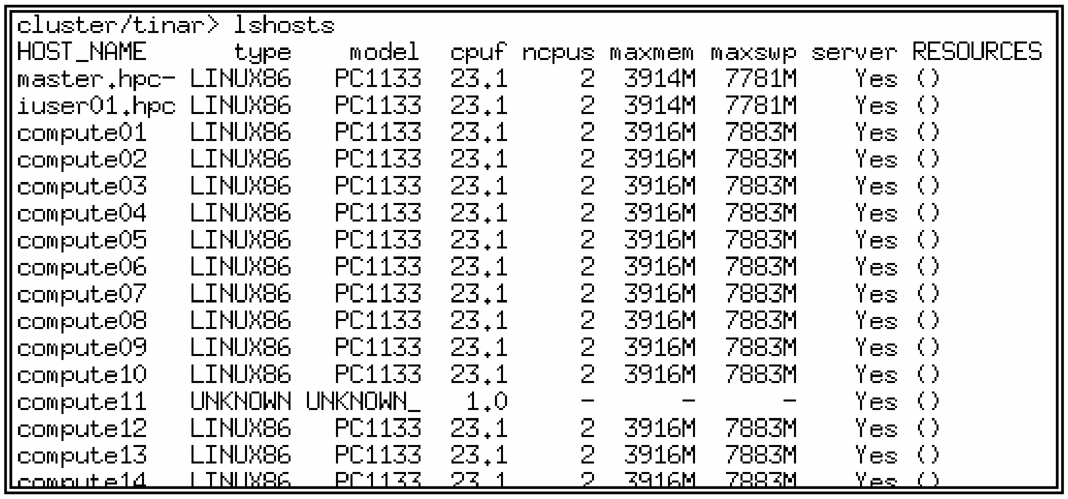
(**i**)nterval,(**n**)umber,(**N**)ormalize,(**E**)ffective,(**R**)esources,(**q**)uit,(**^L**)refresh

NOTE: The **lsload** command will provide similar display. It also, along with **lsmon**, shows what the load levels on the system are and tells you that LSF is running correctly.

lshosts and bhosts

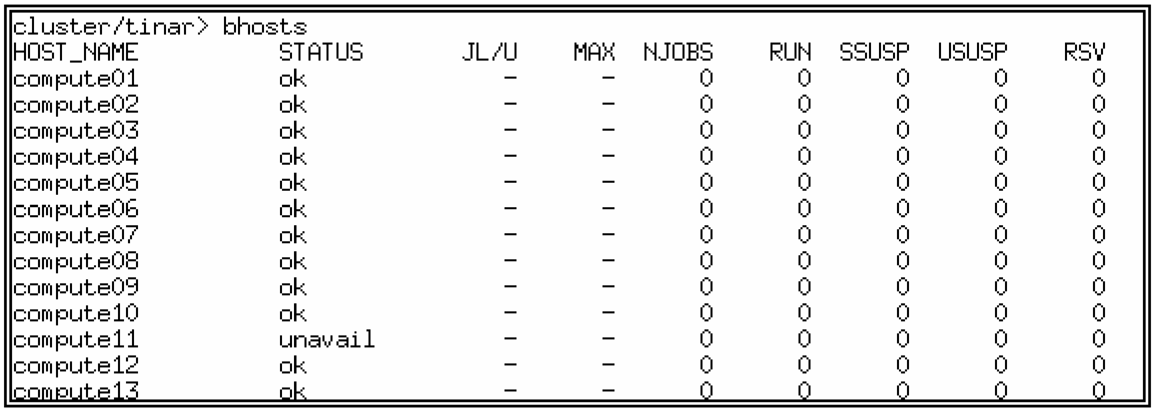
In theory, the **lshosts** and **bhosts** display different information. **lshosts** shows you all of the nodes that the LSF system is aware of and that are running what is referred to as the base LSF commands. **bhosts** displays all of the nodes that have been configured to do the batch processing of jobs and are running LSF Batch. *Batch*, in this context, is any system that the scheduler can make decisions for and run jobs on. The theory is that these 2 needs may require different configurations and be run on separate nodes. But in the real world, all nodes in the cluster are configured to both run the base LSF (commonly referred to as just LSF) and LSF Batch (commonly referred to as batch) jobs.

lshosts:



```
cluster/tinar> lshosts
HOST_NAME      type    model  cpuf ncpus maxmem maxswp server RESOURCES
master.hpc-  LINUX86   PC1133  23.1     2  3914M  7781M    Yes ()
iuser01.hpc  LINUX86   PC1133  23.1     2  3914M  7781M    Yes ()
compute01    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute02    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute03    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute04    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute05    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute06    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute07    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute08    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute09    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute10    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute11    UNKNOWN UNKNOWN_   1.0     -      -      -    Yes ()
compute12    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute13    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
compute14    LINUX86   PC1133  23.1     2  3916M  7883M    Yes ()
```

bhosts:



```
cluster/tinar> bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
compute01          ok              -      -      0      0      0      0      0
compute02          ok              -      -      0      0      0      0      0
compute03          ok              -      -      0      0      0      0      0
compute04          ok              -      -      0      0      0      0      0
compute05          ok              -      -      0      0      0      0      0
compute06          ok              -      -      0      0      0      0      0
compute07          ok              -      -      0      0      0      0      0
compute08          ok              -      -      0      0      0      0      0
compute09          ok              -      -      0      0      0      0      0
compute10          ok              -      -      0      0      0      0      0
compute11          unavail         -      -      0      0      0      0      0
compute12          ok              -      -      0      0      0      0      0
compute13          ok              -      -      0      0      0      0      0
```

Though formatted differently, the information displayed by these two commands is basically the same. The bottom line is that if a node is listed as unavailable, it will not be able to process jobs. Which command you use for monitoring the cluster is a matter of personal choice, but the most common monitoring command is **lsmon**.

Doing Work – Job Commands

LSF jobs have the following states:

- **PEND** - Waiting in a queue for scheduling and dispatch
- **RUN** - Dispatched to a host and running
- **DONE** - Finished normally with zero exit value
- **EXITED** - Finished with non-zero exit value (*there is an error*)
- **PSUSP** - Suspended while pending
- **USUSP** - Suspended by user
- **SSUSP** - Suspended by the LSF system
- **POST_DONE** - Post-processing completed without errors
- **POST_ERR** - Post-processing completed with errors
- **WAIT** - Members of a chunk job that are waiting to run

lsrun <job name>

There are 2 basic commands for running jobs on the Tufts cluster – **lsrun** and **bsub**. As you may be able to guess, jobs submitted with lsrun are run, interactively, on the best available host (based on load), queued and run by LSF. Jobs submitted with the bsub command are run interactively or in batch mode, based on the load, *and* handled by LSF batch. Needless to say, if you are running a simple, quick job, **bsub** is a waste of resources – use **lsrun**. The job will run very quickly, depending on current load and job complexity. The hostname job (*runs the Unix hostname application, which tells you the name of the host that hostname ran on*) is a perfect example – the response is almost instantaneous:

```
cluster/trieda01> lsrun hostname   <enter>
compute04
cluster/trieda01>
```

bsub <job name>

The bsub command, as we just stated, submits a job interactively (some applications require a shell or input) or in batch, using LSF batch scheduling and queue layer of the LSF suite. Our hostname job, submitted several times in a row with the bsub command, looks like this:

```
cluster/trieda01> bsub hostname   <enter>
Job <1023> is submitted to default queue <normal>.
cluster/trieda01> bsub hostname   <enter>
Job <1024> is submitted to default queue <normal>.
cluster/trieda01> bjobs   <enter>
```

```
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1023    trieda0 PEND  normal    iuser01.hpc             hostname   Jun 30 13:11
1024    trieda0 PEND  normal    iuser01.hpc             hostname   Jun 30 13:11
cluster/trieda01> bjobs
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1024    trieda0 PEND  normal    iuser01.hpc             hostname   Jun 30 13:11
cluster/trieda01> bjobs   <enter>
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1024    trieda0 PEND  normal    iuser01.hpc             hostname   Jun 30 13:11
cluster/trieda01> bjobs
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1024    trieda0 PEND  normal    iuser01.hpc             hostname   Jun 30 13:11
cluster/trieda01> bjobs
No unfinished job found
```

bjobs <JobID#>

bjobs displays information about a recently run job. You can use the **–l** option to view a more detailed accounting:

```
cluster/trieda01> bjobs -l 1024    <enter>

Job <1024>, User <trieda01>, Project <default>, Status <DONE>, Queue <normal>,
                     Command <hostname>
Mon Jun 30 13:11:17: Submitted from host <iuser01.hpc-c01.tufts.edu>, CWD <$HOM
                     E>;
Mon Jun 30 13:11:28: Started on <compute10>, Execution Home </home/cluster/trie
                     da01>, Execution CWD </home/cluster/trieda01>;
Mon Jun 30 13:11:28: Done successfully. The CPU time used is unknown.

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut     pg    io   ls    it    tmp    swp    mem
 loadSched   -    3.5   3.5    -      -     -    -     -     -      -      -
 loadStop    -     -     -     -      -     -    -     -     -      -      -
```

bqueues

The **bqueues** command displays information about the batch queues. Again, the **–l** option will display a more thorough description:

```
cluster/trieda01> bqueues   <enter>
QUEUE: normal
  -- Normal queue for all cluster users. Scheduling is 'fairshare' across all queues. Jobs in this queue are
limited to 7 days (10080 minutes) of CPU time. Jobs exceedin this limit will be killed.  This is the default
queue.

PARAMETERS/STATISTICS
```

```
PRIO NICE STATUS        MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 30    0  Open:Active      -    -    -    -     0     0     0     0     0    0

 CPULIMIT
 10080.0 min of master.hp

SCHEDULING PARAMETERS
          r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     3.5   3.5    -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

         gm_ports adapter_windows     poe
 loadSched      -              -       -
 loadStop       -              -       -

SCHEDULING POLICIES:  FAIRSHARE
USER_SHARES:  [default, 1]

SHARE_INFO_FOR: normal/
 USER/GROUP   SHARES  PRIORITY  STARTED  RESERVED  CPU_TIME  RUN_TIME
dag             1       0.333      0        0        0.0        0
kolum           1       0.333      0        0        0.0        0
scarpe01        1       0.333      0        0        7.0        0
trieda01        1       0.333      0        0       35.0        0
afant           1       0.332      0        0       46.0        0
dmarshal        1       0.329      0        0      180.0        0
jfmugg01        1       0.329      0        0      180.0        0
afant01         1       0.004      0        0   1159674.1       0

USERS:  all users
HOSTS:  computeNodes/

------------------------------------------------------------------------------

QUEUE: low-priority
  -- For lower priority and/or long running (1 week+)  jobs. 'fairshare' scheduling is enforced across all
queues. No penalizing system nice limits and no restrictions on jo length or CPU time.

PARAMETERS/STATISTICS
PRIO NICE STATUS        MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 20   10  Open:Active      -    -    -    -     0     0     0     0     0    0

 CPULIMIT
 40320.0 min of master.hp

SCHEDULING PARAMETERS
          r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     3.5   3.5    -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

         gm_ports adapter_windows     poe
 loadSched      -              -       -
 loadStop       -              -       -
```

SCHEDULING POLICIES: FAIRSHARE
USER_SHARES: [default, 1]

USERS: all users
HOSTS: computeNodes/

bkill <job # or –J job_name>

From time to time, things head south and you may need to kill an errant job. *You can only kill your own jobs* and if you're running a repetitive job, the current run will be killed and the rest of the repetitions will re-queue. If all else fails, you can use **bkill –r** to remove the job from the system without waiting. If your job is stuck in the dreaded loop, contact the LSF admin at cluster01-support@tufts.edu.

LSF Commands		
Action Needed	**Command**	**Usage**
System verification	**lsid**	lsid *<hit enter>*
Display load levels	**lsmon**	lsmon *<hit enter>*
Display *Base hosts	**lshosts**	lshosts *<hit enter>*
Display Base load levels	**lsload**	lsload *<hit enter>*
Run LSF Base job	**lsrun**	lsrun *filename*
Display *Batch hosts	**bhosts**	bhosts *<hit enter>*
View current jobs	**bjobs**	bjobs ***or*** bjobs *job ID #*
Run LSF Batch job	**bsub**	bsub [-op] *filename*
Kill a job	**bkill**	bkill *job id #*
Review/select queue	**bqueues**	bqueues *<hit enter>* ***or*** bqueues *queue_name*
View job history	**bhist**	bhist *job id #*
Changes job order (new or pending)	**btop**	btop *job ID* | *"job_ID"(index_list)"*
Suspend a job	**bstop**	bstop *job ID#*
Resume stopped job	**bresume**	bresume *job ID#*

ADDITIONAL TIPS:

If you have several jobs to run that will take several days worth of compute time, use the **–q low-priority** option with the **bsub** command. This will select the low priority queue for your jobs and allow them to run without delaying other, more time-sensitive jobs.

SECTION
3

Running a Sample Job on the Cluster

You're now ready to run your first job on the cluster. Although this exercise does not make use of the LSF HPC parallel manager, it's a good way to try out some of the Unix and LSF commands we've learned.

To begin, log in to the cluster. Once connected, you can check the status of the cluster with your opening commands: **lsid**, **lsmon / lsload**, and **lshosts / bhosts**.

```
cluster/trieda01> lsid    <enter>
Platform HPC 5.1 for Linux, Feb 25 2003
Copyright 1992-2003 Platform Computing Corporation

My cluster name is cluster01
My master name is master.hpc-c01.tufts.edu
cluster/trieda01>

cluster/trieda01> lsload   <enter>
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
iuser01.hpc-c01   ok   0.0   0.0   0.0   0%   8.0   5    0  939M 7780M 3358M
compute24        ok   0.0   0.0   0.0   1%   6.7   0 18320 7924M 7880M 3722M
compute03        ok   0.0   0.0   0.0   0%   6.7   0 18320 7924M 7880M 3722M
compute05        ok   0.0   0.0   0.0   0%   6.8   0 18320 7924M 7880M 3722M
compute04        ok   0.0   0.0   0.0   0%   6.8   0  2398 7924M 7880M 3722M
compute09        ok   0.0   0.0   0.0   0%   6.8   0  8440 7924M 7880M 3722M
compute29        ok   0.0   0.0   0.0   0%   6.9   0  2374 7924M 7880M 3722M
compute01        ok   0.0   0.0   0.0   0%   6.9   0  2372 7532M 7880M 3714M
compute14        ok   0.0   0.0   0.0   0%   7.0   0  2486 7924M 7880M 3728M
compute19        ok   0.0   0.0   0.0   0%   7.1   0  2482 7924M 7880M 3722M
compute08        ok   0.0   0.0   0.0   0%   7.1   0 18320 7924M 7880M 3722M
compute22        ok   0.0   0.0   0.0   0%   7.1   0 18320 7924M 7880M 3722M
compute28        ok   0.0   0.0   0.0   0%   7.1   0 18320 7924M 7880M 3722M
```

```
compute27        ok  0.0  0.0  0.0  0%  7.2  0 18320 7924M 7880M 3722M
compute23        ok  0.0  0.0  0.0  0%  7.2  0 18320 7924M 7880M 3722M
compute16        ok  0.0  0.0  0.0  0%  7.2  0 18320 7924M 7880M 3724M
compute31        ok  0.0  0.0  0.0  0%  7.2  0 18320 7924M 7880M 3722M
compute12        ok  0.0  0.0  0.0  0%  7.2  0 12648 7924M 7880M 3720M
compute11        ok  0.0  0.0  0.0  0%  7.2  0 18320 7924M 7880M 3722M
compute13        ok  0.0  0.0  0.0  0%  7.3  0  9864 7924M 7880M 3724M
compute32        ok  0.0  0.0  0.0  0%  7.3  0 18320 7920M 7880M 3722M
compute15        ok  0.0  0.0  0.0  0%  7.3  0  7096 7924M 7880M 3722M
compute02        ok  0.0  0.0  0.0  0%  7.3  0 11232 7924M 7880M 3718M
compute21        ok  0.0  0.0  0.0  0%  7.5  0 18320 7924M 7880M 3722M
compute07        ok  0.0  0.0  0.0  0%  7.5  0 18320 7924M 7880M 3722M
compute25        ok  0.0  0.0  0.0  0%  7.5  0  2348 7924M 7880M 3724M
compute20        ok  0.0  0.0  0.0  0%  7.5  0 18320 7924M 7880M 3722M
compute30        ok  0.0  0.0  0.0  0%  7.6  0  8592 7924M 7880M 3722M
compute06        ok  0.0  0.0  0.0  0%  7.6  0  7096 7924M 7880M 3722M
compute10        ok  0.0  0.0  0.0  0%  7.7  0  2872 7924M 7880M 3722M
compute17        ok  0.0  0.0  0.0  0%  7.7  0 18320 7924M 7880M 3722M
compute18        ok  0.0  0.0  0.0  0%  7.8  0  2550 7924M 7880M 3714M
compute26        ok  0.0  0.0  0.0  0%  8.2  0  2482 7924M 7880M 3724M
master.hpc-c01.    ok  0.0  0.0  0.0  0% 10.5  1    14  815M 7771M 3687M
cluster/trieda01>
```

Let's run a simple job, the Unix command *hostname*, on the cluster:

```
cluster/trieda01> lsrun hostname <enter>
compute17
cluster/trieda01>
```

Next, let's run it with the LSF batch command, **bsub**. We'll run the program (**hostname**) twice and then quickly hit the up arrow to access the bjobs command, to review the process (***tip*** *– you may want to run the* ***bjob*** *command first, to put it in the cache*):

```
cluster/trieda01> bjobs    <enter>
No unfinished job found
```

```
cluster/trieda01> bsub hostname     <enter>
Job <1081> is submitted to default queue <normal>.
```

Quickly, hit your up arrow to repeat the command on the screen and hit enter

```
cluster/trieda01> bsub hostname    <enter>
Job <1082> is submitted to default queue <normal>.
```

Hit your up arrow twice to bring up the bjobs command and hit enter- repeat the up-arrow and hit enter process until you see the "No unfinished jobs" message.

```
cluster/trieda01> bjobs     <enter>
JOBID  USER   STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
```

```
1081    trieda0 PEND  normal    iuser01.hpc          hostname   Jul  2 16:04
1082    trieda0 PEND  normal    iuser01.hpc          hostname   Jul  2 16:04

cluster/trieda01> bjobs    <enter>
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1082    trieda0 PEND  normal    iuser01.hpc          hostname   Jul  2 16:04

cluster/trieda01> bjobs    <enter>
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1082    trieda0 PEND  normal    iuser01.hpc          hostname   Jul  2 16:04

cluster/trieda01> bjobs    <enter>
No unfinished job found
cluster/trieda01>
```

You will receive email notification with a synopsis of the job process and the results (*more about email notification later*). But there are a few other ways to view the process and the results. One of them is with the **bhist** command:

```
cluster/trieda01> bhist 1082    <enter>
Summary of time in seconds spent in various states:
JOBID   USER    JOB_NAME  PEND    PSUSP   RUN     USUSP   SSUSP   UNKWN   TOTAL
1082    trieda0 hostname  11      0       0       0       0       0       11
```

If you find this data a little thin, you can run **bhist** in its long format with the **–l** option:

```
cluster/trieda01> bhist -l 1082    <enter>

Job <1082>, User <trieda01>, Project <default>, Command <hostname>
Wed Jul  2 16:04:09: Submitted from host <iuser01.hpc-c01.tufts.edu>, to Queue
                     <normal>, CWD <$HOME>;
Wed Jul  2 16:04:20: Dispatched to <compute03>;
Wed Jul  2 16:04:20: Starting (Pid 11552);
Wed Jul  2 16:04:20: Running with execution home </home/cluster/trieda01>, Exec
                     ution CWD </home/cluster/trieda01>, Execution Pid <11552>;

Wed Jul  2 16:04:20: Done successfully. The CPU time used is 0.0 seconds;
Wed Jul  2 16:04:20: Post job process done successfully;

Summary of time in seconds spent in various states by  Wed Jul  2 16:04:20
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  11       0        0        0        0        0        11
```

You can also use the output option (-o) to send the results to a text file:

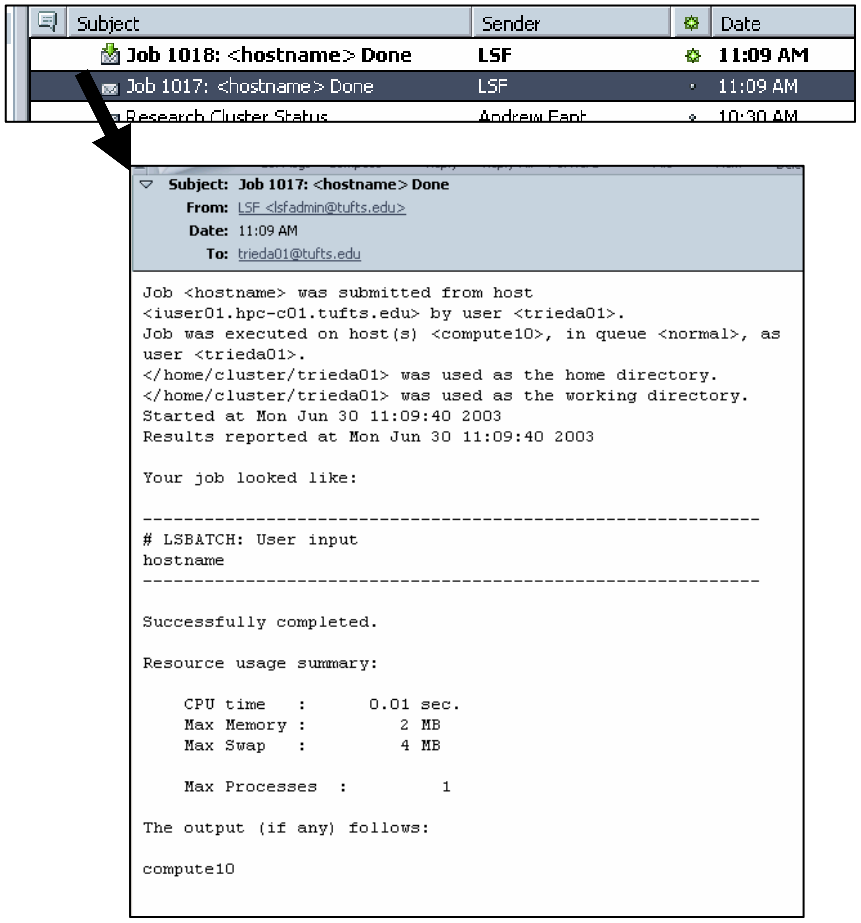
```
cluster/trieda01> bsub -o host1.txt hostname    <enter>
Job <1087> is submitted to default queue <normal>.

cluster/trieda01> ls    <enter>
host1.txt  parallel-example
```

```
cluster/trieda01> more host1.txt    <enter>
Sender: LSF System <lsfadmin@compute25>
Subject: Job 1086: <hostname> Done

Job <hostname> was submitted from host <iuser01.hpc-c01.tufts.edu> by user <trieda01>.
Job was executed on host(s) <compute25>, in queue <normal>, as user <trieda01>.
</home/cluster/trieda01> was used as the home directory.
</home/cluster/trieda01> was used as the working directory.
Started at Wed Jul  2 16:16:21 2003
Results reported at Wed Jul  2 16:16:21 2003

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
hostname
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.02 sec.
    Max Memory :         2 MB
    Max Swap   :         4 MB

    Max Processes  :         1

The output (if any) follows:

compute25
Sender: LSF System <lsfadmin@compute24>
Subject: Job 1087: <hostname> Done

Job <hostname> was submitted from host <iuser01.hpc-c01.tufts.edu> by user <trieda01>.
Job was executed on host(s) <compute24>, in queue <normal>, as user <trieda01>.
</home/cluster/trieda01> was used as the home directory.
</home/cluster/trieda01> was used as the working directory.
Started at Wed Jul  2 16:16:56 2003
Results reported at Wed Jul  2 16:16:56 2003

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
hostname
------------------------------------------------------------

Exited
The output (if any) follows:

compute24
cluster/trieda01>
```

Automatic Email Notification of Results

Whenever you run an LSF batch job (**bsub**), by default, you will receive email notification of your job results.

NOTE: You will not receive email notification of results with basic LSF commands (*ex:* **lsrun**).



Subject	Sender	Date
Job 1018: <hostname> Done	**LSF**	**11:09 AM**
Job 1017: <hostname> Done	LSF	11:09 AM
Research Cluster Status	Andrew Fant	10:30 AM

Subject: Job 1017: <hostname> Done
From: LSF <lsfadmin@tufts.edu>
Date: 11:09 AM
To: trieda01@tufts.edu

```
Job <hostname> was submitted from host
<iuser01.hpc-c01.tufts.edu> by user <trieda01>.
Job was executed on host(s) <compute10>, in queue <normal>, as
user <trieda01>.
</home/cluster/trieda01> was used as the home directory.
</home/cluster/trieda01> was used as the working directory.
Started at Mon Jun 30 11:09:40 2003
Results reported at Mon Jun 30 11:09:40 2003

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
hostname
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.01 sec.
    Max Memory :         2 MB
    Max Swap   :         4 MB

    Max Processes  :         1

The output (if any) follows:

compute10
```

You can opt out of this option by using the ***-o filename*** option, discussed earlier, which sends the output to whatever filename you choose. This becomes increasingly important if you are running multiple jobs.

Example:

Instead of the command:
bsub hostname

type:

```
bsub  –o hostjob1.txt hostname
```

You can also use the ***–e*** option to output results and errors to a text file. You will not receive email notification. To read the results of this application, you would type **more hostjob1.txt** <*enter*>.

The ***–I*** option will also subvert the email process by outputting the results to your screen. This can be invaluable if you're chasing a problem and need to know when in the process your program is heading south, but users of the cluster are *strongly discouraged* from using this option for as a matter of course.

SECTION

4

Running a Parallel Job on the Cluster

The ***hostname*** job we ran on the cluster in the previous section was actually run as a serial job – that is, all job tasks for each request ran completely on one node. We initiated several jobs in quick succession to watch them be distributed to different nodes, but they did not really take advantage of the clusters parallel capabilities. A different node ran each instance of the hostname process, but the tasks within the hostname program were not shared and running on multiple nodes. To do so requires a parallel application, one that has been written with MPI calls and compiled in a way that it can link to the LSF-supplied libraries on the cluster.

Traditionally, a programmer writes an application in an editor using their programming language of choice, such as C, C+, Fortran, or Java, for example. The lines of code they create are called *source statements*. The file is saved and run with a compiler that matches the language used to create the source code. The compiler turns the *source code* into *object code*, which is machine code that processors understand and can execute one instruction at a time. In other words, compiling the source statements turns them into a program.

If we created a file in a text editor, with C language statements, and saved the text file as **hello world.c**, the command to compile it with a c compiler would be:

```
cc hello.c
```

But a compiled program still won't run in parallel on the cluster. Here's why -

We mentioned earlier in this manual that, on the Tufts cluster, parallel applications use the message passing model and that Cluster01 uses the MPI programming specification open source implementation called LAM. LAM (Local Area Multicomputer) is an MPI programming environment. Platform LSF requires that its LAM/MPI compiler script be used to compile (or recompile) and link parallel applications. This allows the special libraries and options provided by Platform HPC to be utilized. Think of it as a helper script that gives you extra linkage to the LSF libraries. And your parallel application needs the LSF helper codes to run.

Compiling and Running a Parallel Application

Parallel programming and the use of parallel applications on a cluster is a deeply layered art form and something that can take a long time to perfect. The following exercise is meant as an example of parallel use of the cluster in its most basic form. If you're interested in parallel programming, the full text of **Designing and Building Parallel Programs**, by *Ian Foster(1995)* is available online, for free, from the Mathematics and Computer Science Division of the U.S. government's website (http://www-unix.mcs.anl.gov/dbpp/web-tours/).

Calculating π

Here is a sample of a simple source code text file named **cpi.c**, written and saved with an editor, such as Vi[5], that will calculate the number of π (Pi), written in c language. Windows users may prefer to use the DOS editor, available at the command prompt by typing **edit** *<enter> (see Appendix A for additional instructions)*.

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f( double );
double f( double a )
{
   return (4.0 / (1.0 + a*a));
}

int main( int argc, char *argv[])
{
   int done = 0, n, myid, numprocs, i;
   double PI25DT = 3.141592653589793238462643;
   double mypi, pi, h, sum, x;
   double startwtime = 0.0, endwtime;
   int  namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
```

[5] Unix is a Four Letter Word includes a nice tutorial on the usage of the Vi editor.

```
    MPI_Get_processor_name(processor_name,&namelen);

    fprintf(stderr,"Process %d on %s\n",
            myid, processor_name);

    n = 0;
    while (!done)
    {
        if (myid == 0)
        {
/*
            printf("Enter the number of intervals: (0 quits) ");
            scanf("%d",&n);
*/
            if (n==0) n=100; else n=0;

            startwtime = MPI_Wtime();
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            done = 1;
        else
        {
            h   = 1.0 / (double) n;
            sum = 0.0;
            for (i = myid + 1; i <= n; i += numprocs)
            {
                x = h * ((double)i - 0.5);
                sum += f(x);
            }
            mypi = h * sum;

            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

            if (myid == 0)
            {
                printf("pi is approximately %.16f, Error is %.16f\n",
                       pi, fabs(pi - PI25DT));
                endwtime = MPI_Wtime();
                printf("wall clock time = %f\n",
                       endwtime-startwtime);
            }
        }
    }
    MPI_Finalize();

    return 0;
}
```

When you save this file in an editor, as **cpi.c**, you've created the *source code* for a program. If the program is capable of parallel processing and makes MPI calls, you will need to compile it with the MPI compiler.

```
cluster/trieda01> mpicc -c cpi.c    <enter>
cluster/trieda01>
```

If we check the list of files created, we'll see a new addition, **cpi.o**, which is our cpi.c *source code*, compiled and transformed into *object code*:

```
cluster/trieda01> ls -l   <enter>
total 2
-rw-r--r--    1 trieda01 facstaff     1640 Jun 20 14:57 cpi.c
-rw-r--r--    1 trieda01 facstaff     2060 Aug 13 13:07 cpi.o
cluster/trieda01>
```

Next, we need to link our object code, with **–o**, to the Platform Parallel libraries and create an executable. Let's also rename our executable program to **calculate1**:

```
cluster/trieda01> mpicc -o calculate1 cpi.o    <enter>
cluster/trieda01>
```

Check to verify that your program was successfully created and renamed:

```
cluster/trieda01> ls -l   <enter>
total 380
-rwxr-xr-x    1 trieda01 facstaff   380326 Aug 13 13:28 calculate1
-rw-r--r--    1 trieda01 facstaff     1640 Jun 20 14:57 cpi.c
-rw-r--r--    1 trieda01 facstaff     2060 Aug 13 13:07 cpi.o
cluster/trieda01>
```

Now that you've compiled and linked the hard way, you can be told that the compile (**–c**) and link (**-o**) steps can be <u>combined and accomplished with one command line instead of 2.</u> This time, let's also rename our **cpi.c** source code, but change the name to **calcpi2**:

```
cluster/trieda01> mpicc cpi.c -o calcpi2    <enter>
cluster/trieda01>

cluster/trieda01>  ls –l    <enter>
total 752
-rwxr-xr-x    1 trieda01 facstaff   380326 Aug 13 13:54 calcpi2
-rwxr-xr-x    1 trieda01 facstaff   380326 Aug 13 13:28 calculate1
-rw-r--r--    1 trieda01 facstaff     1640 Jun 20 14:57 cpi.c
-rw-r--r--    1 trieda01 facstaff     2060 Aug 13 13:07 cpi.o
cluster/trieda01>
```

NOTE: *Using the MPI compilers to compile a program that does not make MPI calls or use parallel functions will work, but will make the program larger than it needs to be. Instead, use the gcc compiler for serial programs:*

Type **gcc hi.c -o hitufts**   *<enter>*

We're ready to test our parallel program with the LSF Parallel:

bsub -a lammpi -n 4 ./calcpi2

This command submits a parallel batch job (**bsub**) and tells the LSF HPC *Parallel Application Manager* (PAM is called by the **–a** flag) to run our *calcpi2* program, which is located in the current directory (**./calcpi2**), with the LAM implementation of MPI (**lammpi**) on 4 nodes (**-n 4**) in the cluster. Because we have not requested the output to go to a text file, the output will be emailed to us. Once we input the command and hit enter, the management resources of LSF will go to work, scheduling, queuing, and running the job:

```
cluster/trieda01> bsub -a lammpi -n 4 ./calcpi2    <enter>
Job <737> is submitted to queue <normal>.
cluster/trieda01>
```

The email output:

```
Job <pam -g 1 lammpirun_wrapper ./calcpi2> was submitted from host
<iuser01.hpc-c01.tufts.edu> by user <trieda01>.
Job was executed on host(s) <2*compute05>, in queue <normal>, as user
<trieda01>.
                            <2*compute31>
</home/cluster/trieda01> was used as the home directory.
</home/cluster/trieda01> was used as the working directory.
Started at Wed Aug 13 14:04:54 2003
Results reported at Wed Aug 13 14:04:57 2003

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
pam -g 1 lammpirun_wrapper ./calcpi2
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.20 sec.
    Max Memory :         2 MB
    Max Swap   :         4 MB
    Max Processes  :         1

The output (if any) follows:

21493 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
21495 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
21650 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
21652 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
Process 0 on compute05
Process 1 on compute05
Process 2 on compute31
Process 3 on compute31
pi is approximately 3.1416009869231245, Error is 0.0000083333333314
wall clock time = 0.000253

TID  HOST_NAME    COMMAND_LINE            STATUS                  TERMINATION_TIME
==== ========== ================  =======================
===================
0001 compute05                    Done                         08/13/2003
14:04:56
0002 compute05                    Done                         08/13/2003
14:04:56
0003 compute31                    Done                         08/13/2003
14:04:56
0004 compute31                    Done                         08/13/2003
14:04:56
```

Let's resubmit our job, but this time, send the output to a text file named calcpi2.txt. We will also view the status of our job submission by using the up-arrow trick to run the bjobs command in quick succession:

```
cluster/trieda01> bsub -o./calcpi2.txt -a lammpi -n 4 ./calcpi2    <enter>
Job <739> is submitted to queue <normal>.
cluster/trieda01>

cluster/trieda01> bjobs    <enter>
JOBID  USER   STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
739    trieda0 PEND  normal    iuser01.hpc          *./calcpi2 Aug 13 15:03
cluster/trieda01>

cluster/trieda01> bjobs    <enter>
JOBID  USER   STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
739    trieda0 PEND  normal    iuser01.hpc          *./calcpi2 Aug 13 15:03
cluster/trieda01>

cluster/trieda01> bjobs    <enter>
JOBID  USER   STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
739    trieda0 RUN  normal    iuser01.hpc compute02  *./calcpi2 Aug 13 15:03
                                          compute02
                                          compute15
                                          compute15
cluster/trieda01> bjobs    <enter>
No unfinished job found
cluster/trieda01>
```

We asked LSF to create a text file named ***calcpi2.txt***. Let's take a look at what detail was recorded there:

```
cluster/trieda01> more calcpi2.txt    <enter>
25749 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
```

```
25750 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
25444 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
25446 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
Process 0 on compute02
Process 1 on compute02
Process 2 on compute15
Process 3 on compute15
pi is approximately 3.1416009869231245, Error is 0.0000083333333314
wall clock time = 0.000233

TID  HOST_NAME    COMMAND_LINE          STATUS          TERMINATION_TIME
==== ========== ================  ======================  ===================
0001 compute02                Done                08/13/2003 15:03:17
0002 compute02                Done                08/13/2003 15:03:17
0003 compute15                Done                08/13/2003 15:03:17
0004 compute15                Done                08/13/2003 15:03:17


------------------------------------------------------------
Sender: LSF System <lsfadmin@compute02>
Subject: Job 739: <pam -g 1 lammpirun_wrapper ./calcpi2> Done

Job <pam -g 1 lammpirun_wrapper ./calcpi2> was submitted from host <iuser01.hpc-c01.tufts.edu> by
 user <trieda01>.
Job was executed on host(s) <2*compute02>, in queue <normal>, as user <trieda01>.
                            <2*compute15>
</home/cluster/trieda01> was used as the home directory.
</home/cluster/trieda01> was used as the working directory.
Started at Wed Aug 13 15:03:15 2003
Results reported at Wed Aug 13 15:03:17 2003

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
pam -g 1 lammpirun_wrapper ./calcpi2
------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.21 sec.
    Max Memory :         2 MB
    Max Swap   :         4 MB

    Max Processes  :         1

The output (if any) is above this job summary.
```

Common Errors

LIM is down; try later

```
cluster/trieda01> lsid
LSF 5.1, Dec 20 2002
Copyright 1992-2002 Platform Computing Corporation

lsid: ls_getclustername() failed: LIM is down; try later
cluster/trieda01>
```

This error says that LSF LIM (Load Information Manager) is not running on the cluster. It is not a hardware failure. Wait 10 minutes and try again. If you still receive the error after 10 minutes, send an email to cluster01-support@tufts.edu.

No such file or directory

```
cluster/trieda01> mpicc ./calcpi2 -o ./cpi.c     <enter>
gcc: ./calcpi2: No such file or directory
mpicc: No such file or directory
cluster/trieda01>
```

This error can be a little tricky. The error is the result of an incorrect command sequence. When compiling (or recompiling) *and* linking, the name of the source code file should appear first, not last, in the sequence. Confusion can arise because when you are linking previously compiled object code with the –o flag, the name for the executable you are creating is listed first:

Linking a previously compiled file:
mpicc -o calculate1 cpi.o

Compiling and linking all at once:
mpicc cpi.c -o calcpi2

Here is another instance of the "no such file or directory" error:

```
cluster/trieda01> mpicc cpi -o calcpi2     <enter>
gcc: ./cpi: No such file or directory
mpicc: No such file or directory
```

This time, the error is the result of an incomplete file name. The name of the source code file is **cpi.c**, *not* **cpi**. The correct command is:

```
mpicc cpi.c -o calcpi2
```

It's a good habit to get used to indicating the location of your files with **./**. Notice what happens to our calcpi2 program when we run it with the following command: NOTE: The –I (interactive) flag was used with this command to output the results to the screen to illustrate this error. ***This flag should only be used for debugging purposes.***

```
cluster/trieda01> bsub -I -a lammpi -n 4 calcpi2     <enter>
Job <743> is submitted to queue <normal>.
<<Waiting for dispatch ...>>
<<Starting on compute06.hpc-c01.tufts.edu>>
29579 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
29580 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n0 (o)
execvp error: No such file or directory
execvp error: No such file or directory
29066 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
29068 /opt/lsf/5.1/linux2.4-glibc2.1-x86/bin/TaskStarter running on n1
execvp error: No such file or directory
-----------------------------------------------------------------------------
It seems that [at least] one of the processes that was started with
mpirun did not invoke MPI_INIT before quitting (it is possible that
more than one process did not invoke MPI_INIT -- mpirun was only
notified of the first one, which was on node n0).

mpirun can *only* be used with MPI programs (i.e., programs that
invoke MPI_INIT and MPI_FINALIZE).  You can use the "lamexec" program
to run non-MPI programs over the lambooted nodes.
-----------------------------------------------------------------------------
execvp error: No such file or directory

TID  HOST_NAME   COMMAND_LINE          STATUS          TERMINATION_TIME
==== ========== ================  =======================
==================
0001 compute06               Exit (29)          08/13/2003 16:04:25
0002 compute06               Exit (29)          08/13/2003 16:04:25
0003 compute20               Exit (29)          08/13/2003 16:04:25
0004 compute20               Exit (29)          08/13/2003 16:04:25

cluster/trieda01>
```

CHAPTER 3

Getting Help

There are several places to turn when things aren't going right. The most obvious choice is to send an email to cluster01-support@tufts.edu. The cluster administrators will do their best to help you determine whether your problem is related to the particular application, LSF, or a command statement.

But, as in many other situations, knowledge is power. A little reading, research and programming practice can go a long way in making your use of the cluster efficient and effective. Listed below is a collection of tutorials and support links available on the web. You're not likely to find an "exact fit" – the wealth of diversity in cluster technology also means that there are 200 ways to skin a parallel program.

Tutorials

Parallel Programming - Basic Theory For The Unwary
http://users.actcom.co.il/~choo/lupg/tutorials/parallel-programming-theory/parallel-programming-theory.html

Programming Tutorials
http://users.actcom.co.il/~choo/lupg/tutorials/index.html

How to Build a Beowulf: a Tutorial (*Center for Advanced Computing Research*)
http://www.cacr.caltech.edu/beowulf/tutorial/tutorial.html

Parallel Computational Programs:
An Introduction to Parallel Processing and the Message Passing Interface (*Clemson University*)
    http://www.ces.clemson.edu/~rm/compute.html

Parallel Computing With Linux
    http://www.acm.org/crossroads/xrds6-1/parallel.html

Introduction to MPI
    http://webct.ncsa.uiuc.edu:8900/public/MPI/

MPI Tutorial (Notre Dame University)
    http://www.lam-mpi.org/tutorials/nd/

MPI Tutorials: Getting started with LAM/MPI
    http://www.lam-mpi.org/tutorials/lam/

Documentation

UNIX/Linux Commands
http://www.computerhope.com/unix.htm

High Performance Computing and Communications Glossary 2.1 – 1995, but still useful
    http://wotug.kent.ac.uk/parallel/acronyms/hpccgloss/

LAM/MPI Documentation
    http://www.lam-mpi.org/using/docs/

The UNIX Reference Desk
    http://www.geek-girl.com/unix.html

LinuxHPC.org
    http://www.linuxhpc.org/

How to Build a Beowulf
A Guide to the Implementation and Application of PC Clusters (MIT Press catalog)
    http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3898

Groups

Internet Parallel Computing Guide
http://wotug.ukc.ac.uk/parallel/index.html

Usenet
http://wotug.ukc.ac.uk/parallel/internet/usenet/index.html

Grid User Groups
http://enterthegrid.com/vmp/articles/contentsEnterTheGridResearch%20Repository.html#User%20groups

Acronyms and Terms

List of acronyms and terms reproduced and adapted from *Beowulf Cluster Computing with Linux* – edited by Thomas Sterling, © 2002, The MIT Press
Additional sources include documentation provided by Platform Computing Corporation

Beowulf-class system –
Commodity cluster employing personal computers or low-cost SMP servers to achieve excellent price performance initially developed by the Beowulf project at NASA Goddard Space Flight Center.

Cluster –
In the general sense, any interconnected ensemble of computers capable on independent operation but employed to service a common workload.

Commodity cluster –
A cluster of commercial computing nodes integrated with a commercial system area network.

COW –
Cluster of workstations; an early project at the University of Wisconsin

CPU –
Central Processing Unit

DVD –
Digital Versatile Disc

Fast Ethernet –
A cost effective local area network based on the original Ethernet protocol that has become very popular with low end Beowulf-class systems; providing 100 Mbps.

Gigabit Ethernet –
A LAN that is the successor of Fast Ethernet providing peak bandwidth of 1 Gbps. While employed for some clusters, it's use is limited by it's relatively high cost *(editors note: the "relatively high cost" statement is dated)*.

HPC –
High Performance Computing

I/O –
Input/Output

IP –
Internet Protocol

LAN –
Local Area Network; a network employed within a single administrative domain such as a laboratory or office complex, connecting PC's and workstations together to file servers, printers and other peripherals, and to the Internet. Low cost LAN technology has been adopted to provide Beowulf-class systems with inexpensive moderate bandwidth interconnect.

Linux –
The dominant Unix-like cross-platform operating system developed by a broad international community enabled by an open source code framework.

Mbps –
1 million bits per second data transfer or bandwidth.

Message passing –
An approach to parallelism based on communicating data and signals between processes running (usually) on separate computers.

MPI –
Message passing interface, a community derived standard for the transfer of messages between separate concurrent processes, on a single SMP system, between computer systems on a network, or a combination of the two.

MPP –
Massively Parallel Processors

NIC –
Network Interface Controller; usually the combination of hardware and software that matches the network transport layer to the computer node of the cluster.

NOW –
Network of Workstations, and early influential commodity cluster project at UC Berkeley.

PVFS –
Parallel Virtual File System

PVM –
Parallel Virtual Machine, a library of functions developed at the University of Tennessee-Knoxville and Oak Ridge National Laboratory supporting message-passing semantics between heterogeneous systems on a network.

SMP –
Symmetric Multiprocessor, tightly-coupled cache coherent multiprocessor with uniform memory access.

SSH –
Secure Shell

TCP –
Transmission Control Protocol

References

How to Build a Beowulf – A Guide to the Implementation and Application of PC Clusters - Thomas Sterling, John Salmon, Donald J. Becker, and Daniel F. Savarese, © 1999, The MIT Press

Beowulf Cluster Computing with Linux – edited by Thomas Sterling, © 2002, The MIT Press

LSF Primer – An Introduction to Load Share Facility – US Western District, Platform Computing, Inc., © 2001, Platform Computing, Inc.

Who Needs Supercomputers? - Darnell Little in Chicago and Ira Sager in New York June 3, 2003, BusinessWeek Online (www.businessweek.com)

Platform LSF 5.1 and Platform HPC for Linux 5.1 documentation, Platform Computing Corporation

Links

http://www.beowulf.org/ - The home of the Beowulf Project

http://www.top500.org – Updated twice a year, this site includes a list of the top 500 most powerful systems in the world

http://www.beowulf.org/beowulf/projects.html - Listing of current Beowulf projects

http://www.pgroup.com/products/serverindex.htm - Portland Group compiler

http://computer.org/parascope/ - IEEE Computer Society's listing of parallel computing sites

http://www.asu.edu/it/fyi/dst/helpdocs/lsf/ - LSF command documentation – old, but worth a look for http://www.cluster-rant.com/ - Cluster Rant – a cluster discussion site

Appendix A

Test Applications

Hello, Tufts!

Hello, World is a standard testing application that comes in a zillion varieties. You can get a copy of it in almost every language that exists (http://www.cuillin.demon.co.uk/nazz/trivia/hw/hello_world.html). The one listed below is for C:

………………………………………………………………………………………….

```
#include <stdio.h>
int main(void)
{
  printf("Hello Tufts!\n");
  exit(0);
}
```

………………………………………………………………………………………….

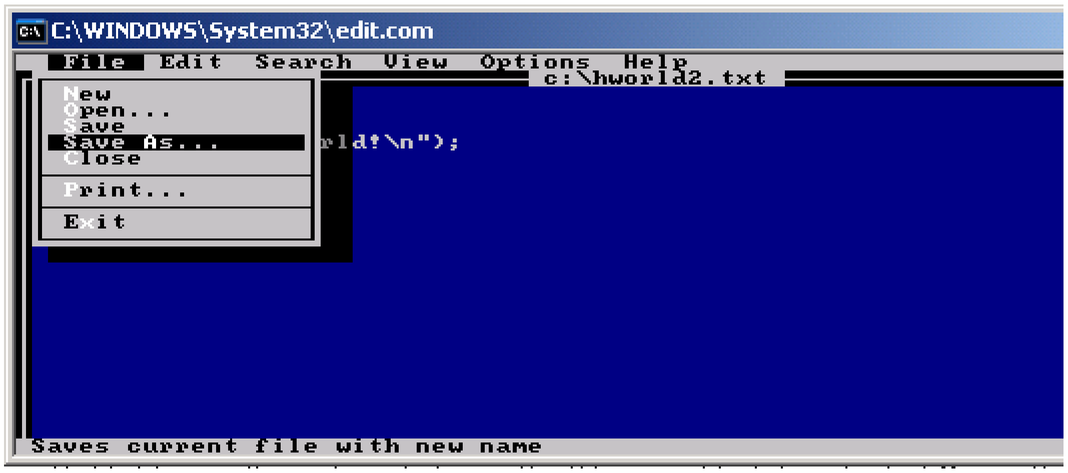
Highlight everything above between the 2 lines, right-click and select **Copy**. Then, click **Start**, **Run**, type *notepad* and click **OK**. When Notepad opens, right-click anywhere in the blank page space and select **Paste**. Save the file as a text file (ex: hi.txt) in a simple location, such as the root of C:. Next, open a command prompt (**Start, Run**, type *cmd*, click **OK**) and type:

edit c:\*location and name of text file* *<enter>*

Example:

edit c:\hi.txt *<enter>*

Press ***Alt-F*** *<enter>*, and hit ***a***



Press the back arrow to remove the *.txt* extension and name your file with a "c" extension
Example:

hi.txt saved as **hi.c**

When you've renamed the file, hit Enter. To exit the editor, press ***Alt-F*** *<enter>* and hit ***X.*** The editor will close.

TIP: You can also click **Start**, **Run**, type *edit c:\hi.txt* and click **OK.** The screen may be smaller, but you'll find that you can use your mouse to navigate.

This file can now be copied to a file in your cluster home directory and used to test or demonstrate command usage (*see Appendix B for more information*). Don't forget to compile and link your test program. Because this program will not make MPI calls and does not have parallel capability, you do not need to use the MPI compiler and, instead, can compile it with the gcc compiler, available on the cluster. The simplest way is to do it from within its directory.
Example:

hw/2> **gcc hi.c -o hey** --- This command compiles and links the file **hi.c** and names the resulting program ***hey*** .

CALCULATING PI

To create a simple program that will calculate pi, copy everything *between* the dotted lines below. Then, follow the same process that you did above for your *hi.txt* file. In the example below, the source code was saved as cpi.txt, and then, cpi.c in a text editor. After you've copied the cpi.c file to your cluster home, you will need to compile this code with the MPI compiler. Use the following command to compile and name your application:

mpicc *source-file.c* –o *new-app-name*

Example:

mpicc cpi.c -o calcpi

……………………………………………………………………………….

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f( double );
double f( double a )
{
   return (4.0 / (1.0 + a*a));
}

int main( int argc, char *argv[])
{
   int done = 0, n, myid, numprocs, i;
   double PI25DT = 3.141592653589793238462643;
   double mypi, pi, h, sum, x;
   double startwtime = 0.0, endwtime;
   int  namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
   MPI_Get_processor_name(processor_name,&namelen);

   fprintf(stderr,"Process %d on %s\n",
           myid, processor_name);

   n = 0;
   while (!done)
   {
       if (myid == 0)
       {
           if (n==0) n=100; else n=0;

           startwtime = MPI_Wtime();
       }
       MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
       if (n == 0)
           done = 1;
       else
       {
           h   = 1.0 / (double) n;
           sum = 0.0;
           for (i = myid + 1; i <= n; i += numprocs)
           {
               x = h * ((double)i - 0.5);
               sum += f(x);
           }
           mypi = h * sum;

           MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);

           if (myid == 0)
           {
               printf("pi is approximately %.16f, Error is %.16f\n",
                      pi, fabs(pi - PI25DT));
               endwtime = MPI_Wtime();
               printf("wall clock time = %f\n",
                      endwtime-startwtime);
           }
       }
   }
   MPI_Finalize();

   return 0;
}
```

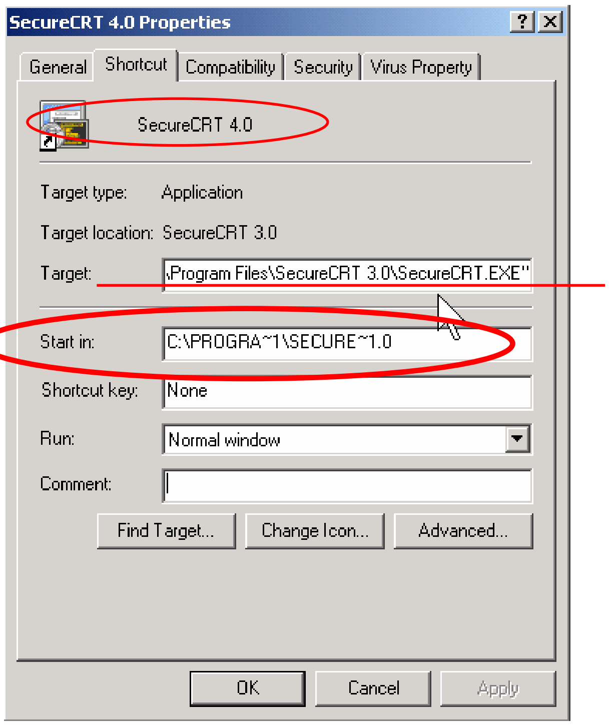
……………………………………………………………………………………

Appendix B

File Transfer with vcp.exe for Windows Clients with SecureCRT

If you're a veteran of the Unix world, you probably have an array of tools and utilities for file transfer to and from your hard drive to your Amber/Cluster01 or Emerald directories. If, however, you are a long time Windows user, you probably use SecureCRT to connect to these systems. And the good news is that SecureCRT comes with a command line utility that you can use to copy files to and from your hard drive to your Cluster01 directories.

The utility, named **vcp.exe**, is located within your SecureCRT directory. If you upgraded from SecureCRT 3.0 to SecureCRT 4.0, the directory may carry the original "3.0" name, even though the application level is 4.0. The simplest way to check the location of the files from your current version is to find the icon for SecureCRT, right-click on it, and select *Properties*.



SecureCRT 4.0

Target type: Application

Target location: SecureCRT 3.0

Target: \Program Files\SecureCRT 3.0\SecureCRT.EXE"

Start in: C:\PROGRA~1\SECURE~1.0

Shortcut key: None

Run: Normal window

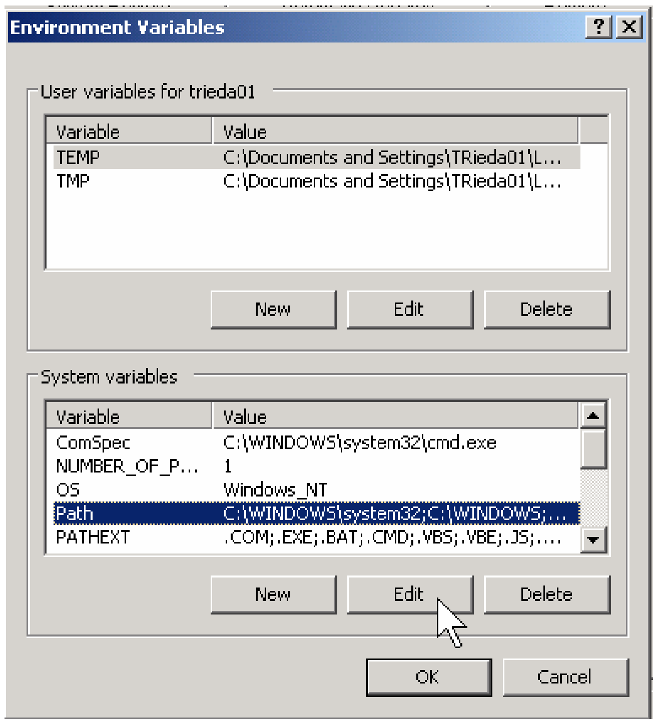
Comment:

You'll notice in the graphic to the right that *SecureCRT.EXE* on this system is actually located in the old *SecureCRT 3.0* folder, located at **c:\program files\securecrt 3.0\securecrt.exe.** Notice, also, that the *Start in:* location is **C:\Progra~1\Secure~1.0 .**

The *Start in:* location of SecureCRT must be added to your environment path statement to enable vcp to work from within any directory on your drive. Write the *Start in:* path down (*or highlight it, right-click and select **Copy***). The path is not case sensitive, but you must use the exact phrasing and spacing.

Next, locate you're *My Computer* icon, right-click it and select *Properties*. Select the *Advanced* tab and click on *Environment Variables.*

Highlight the *Path* statement in the system variables section and click **Edit**.


Environment Variables

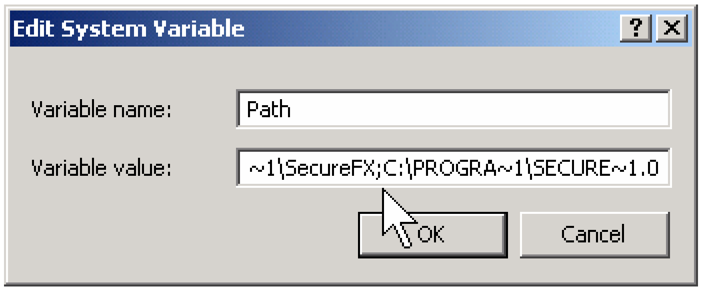
User variables for trieda01

Variable	Value
TEMP	C:\Documents and Settings\TRieda01\L...
TMP	C:\Documents and Settings\TRieda01\L...

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
NUMBER_OF_P...	1
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

Click on the *Variable value:* and arrow to the end of the statement.
<u>Add a semi-colon</u> to separate your new addition to the path statement from the last one and type in (*or right-click and select **Paste** if you copied it from the Properties box*) the path variable exactly as it appeared in the Properties box.


Edit System Variable

Variable name: Path

Variable value: ~1\SecureFX;C:\PROGRA~1\SECURE~1.0

When finished, click **OK**.

Click **OK** to close the *Environment Variables* box and click **Apply** and **OK** to close the *System Properties* box.

To test the application, create a new text file or use an existing one. Then open a command prompt (**Start**, **Run**, type ***cmd*** and click **OK**) and change to the directory that contains your test file. *In the example below, the test file (z-test.txt) is located in the root of C:*

TRANSFERRING FROM C: TO CLUSTER01

This command will copy the **z-test.txt** file to the users cluster home directory:

```
C:\>vcp c:\z-test.txt your-UTLN@cluster01.tccs.tufts.edu:    <enter>
```

You'll be prompted to enter your cluster01 password:

```
your-UTLN@cluster01.tccs.tufts.edu's password: <type cluster01 password and hit enter>
```

Note that you will not see your password displayed on the screen as you type it. Hit enter when you've finished typing your cluster01 password. A line similar to the one below will be displayed:

```
Copying: c:\z-test.txt

C:\>
```

Once you've logged into the cluster, you can use the **ls** command to verify that the file was copied:

```
cluster/trieda01> ls    <enter>
a.out   calctest  hw      jobtest2  myjob-1.c  myjob.txt  parallel-example
calc.c  host1.txt jobtest jobtest3  myjob-1.o  myjob2.c   z-test.txt
cluster/trieda01>
```

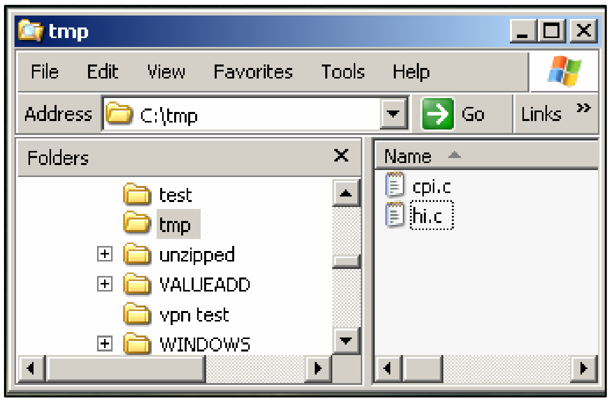
COPYING FROM CLUSTER01 TO C:

The following command line will copy the **hi.c** and **cpi.c** files from the cluster to the c:\tmp directory on the hard drive:

```
C:\>vcp your-UTLN@cluster01.tccs.tufts.edu:*.c c:\tmp    <enter>

your-UTLN@cluster01.tccs.tufts.edu's password: <type cluster01 password and hit enter>


Processing directory: .
ls: .\hi.c
ls: .\cpi.c
Copying: .\cpi.c
```

Copying: .\hi.c

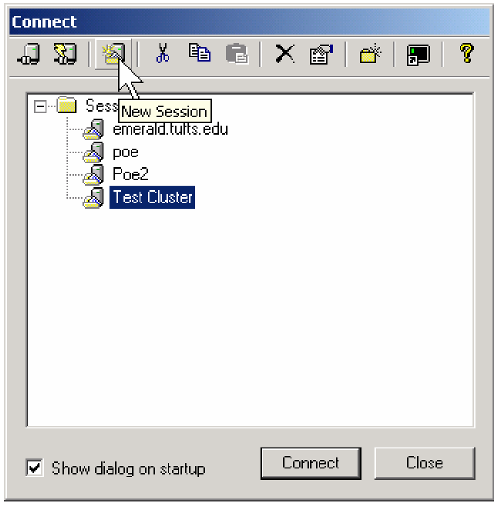
C:\>



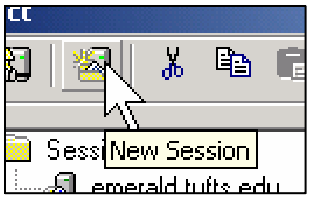
Appendix C

Configuring Your Cluster Session

1.) Locate the SecureCRT 4.0 icon and double-click to open the application. If you had existing session entries in your previous version, you may see something similar to the following screen:



Click the *New Sessions* icon to configure your cluster session.



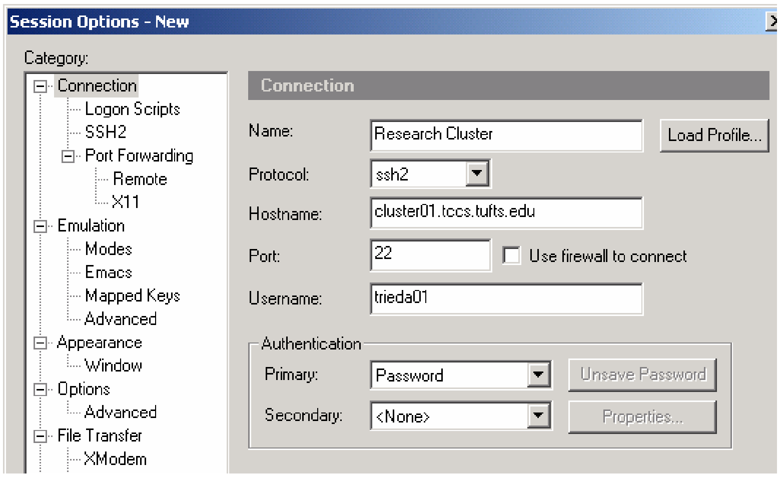
2.) Fill in the following 4 fields:

Name:
Enter any name you like.

Protocol:
Select ssh2

Hostname:
The hostname is
cluster01.tccs.tufts.edu

Username:
Enter your **UTLN** (*Tufts Universal Login Name*)

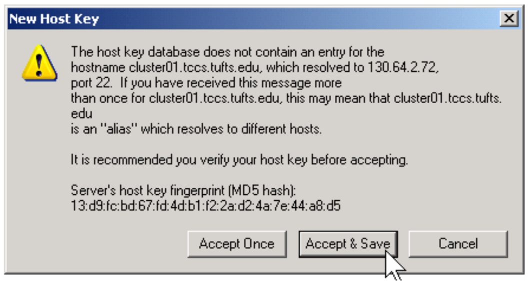


Name: Research Cluster
Protocol: ssh2
Hostname: cluster01.tccs.tufts.edu
Port: 22
Username: trieda01
Primary: Password
Secondary: <None>

Click **OK**.

3.) You should see your new entry highlighted in the *Connect* box. If not, click on it to select it and click **Connect**.

You *may* be prompted to accept and save the new host key. If so, click **Accept & Save**.



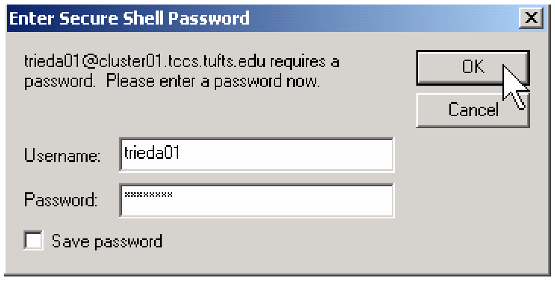
New Host Key

The host key database does not contain an entry for the hostname cluster01.tccs.tufts.edu, which resolved to 130.64.2.72, port 22. If you have received this message more than once for cluster01.tccs.tufts.edu, this may mean that cluster01.tccs.tufts.edu is an "alias" which resolves to different hosts.

It is recommended you verify your host key before accepting.

Server's host key fingerprint (MD5 hash):
13:d9:fc:bd:67:fd:4d:b1:f2:2a:d2:4a:7e:44:a8:d5

4.) When prompted, enter your password. Note that the *Remember my password* box is not checked. Letting your computer remember passwords for you weakens security and is *not recommended*.



Enter Secure Shell Password

trieda01@cluster01.tccs.tufts.edu requires a password. Please enter a password now.

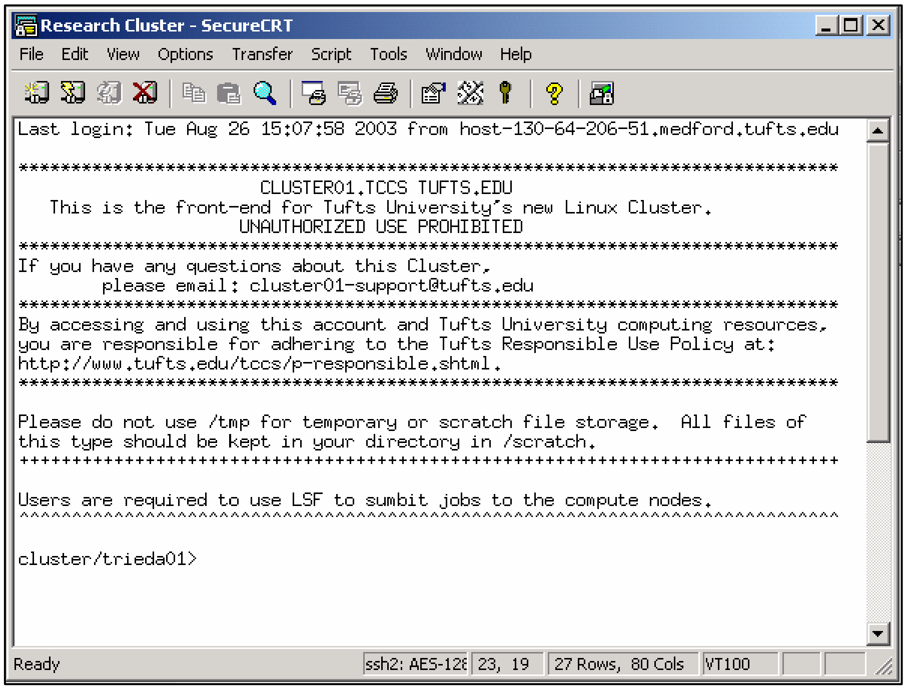
Username: trieda01

Password: ********

☐ Save password

5.) You should see a screen similar to this one:

Congratulations! You've successfully logged in to the Tufts Research Cluster. Next, we'll look around and explore some basic Unix and LSF commands.



Research Cluster - SecureCRT
File Edit View Options Transfer Script Tools Window Help

```
Last login: Tue Aug 26 15:07:58 2003 from host-130-64-206-51.medford.tufts.edu

*******************************************************************************
                        CLUSTER01.TCCS TUFTS.EDU
   This is the front-end for Tufts University's new Linux Cluster.
                      UNAUTHORIZED USE PROHIBITED
*******************************************************************************
If you have any questions about this Cluster,
        please email: cluster01-support@tufts.edu
*******************************************************************************
By accessing and using this account and Tufts University computing resources,
you are responsible for adhering to the Tufts Responsible Use Policy at:
http://www.tufts.edu/tccs/p-responsible.shtml.
*******************************************************************************

Please do not use /tmp for temporary or scratch file storage.  All files of
this type should be kept in your directory in /scratch.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Users are required to use LSF to sumbit jobs to the compute nodes.
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

cluster/trieda01>
```