

SNUPI: A grid accounting and performance system employing portal services and RDBMS back-end

Victor Hazlewood, Ray Bean, Kenneth Yoshimoto
San Diego Supercomputer Center
University of California, San Diego

Abstract

SNUPI, the System, Network, Usage and Performance Interface, provides an interface for resource utilization reporting for heterogeneous computer systems, including Linux clusters. SNUPI provides data collection tools, recommended RDBMS schema design, and Perl-DBI scripts suitable for portal services to deliver reports at the system, user, and job for heterogeneous systems across the enterprise, including Linux clusters. This paper will describe the background of process and project accounting systems for UNIX and Linux, process and batch accounting (JOBLOG) capabilities available for Linux, and describe the collection tools, the RDBMS schema and portal scripts that make up the Open Source SNUPI grid accounting and performance system as employed on the Linux cluster and other systems at NPAC/SDSC.

1 Introduction

Commodity hardware Linux clusters represent an exciting and potentially effective means of providing scalable high performance computing platforms from the desktop to teraflops. Linux standard base software technology for the desktop is well understood and reasonably mature. Additional software technologies to position Linux clusters for future use as an extremely high-performance, general-purpose, teraflop-scale production quality compute servers (what the authors call a supercomputer) must be developed and mature. Key cluster infrastructure components of production compute servers include, but are not limited to, cluster software installation and maintenance, integrated workload management (batch) and job scheduling, resource management and monitoring, process/job/user limit configuration and enforcement, and resource usage reporting. Many examples of Linux cluster software development exist for software installation and maintenance (NPAC/SDSC Rocks, UC Berkeley Millennium Project, Scyld Beowulf, and Caldera Volution), workload management (batch systems, such as, Condor, PBS and LSF), and resource management and monitoring (NPAC/SDSC Rocks, UC Berkeley Millennium Project, Scyld Beowulf, and Caldera Volution). However, not as much work has been done on Linux development in the areas of process/job/user limit configuration and enforcement or resource utilization and performance reporting.

Operating systems designed for desktops, including Linux, have never had robust accounting systems as high-priority items in their early development. But as Solaris, IRIX, AIX and Linux systems have scaled up from desktops to high-end server systems over the last decade, these operating systems are finding homes in environments where resource reporting, accounting and billing capabilities are a requirement. The use of Teraflop-scale compute servers costing millions of dollars to acquire and maintain must

be reported to funding agencies by organizations, such as, NPACI, NCSA, NASA, and LLNL. Computer system usage employed in the work of government and other contracts must be accounted for by such companies as Ford, Boeing, and Lockheed Martin. As Linux clusters find their way into these high-end production compute server environments, solutions for accounting and reporting on these systems must be developed. SNUPI, the System, Network, Usage and Performance Interface, provides a new approach and solution to this age-old requirement of resource utilization and performance reporting for production computing.

2 SNUPI Overview

SNUPI provides a new paradigm for resource utilization and performance reporting for any UNIX system, including Linux clusters. SNUPI can be employed by any single computer system, a collection of computer systems across the enterprise, or for a collection of computer systems across a computational grid. SNUPI provides a novel approach that includes grid data collection employing a relational database management system (RDBMS) and Perl-DBI scripts suitable for portal services to deliver reports at the system, user, job and soon to the process level.

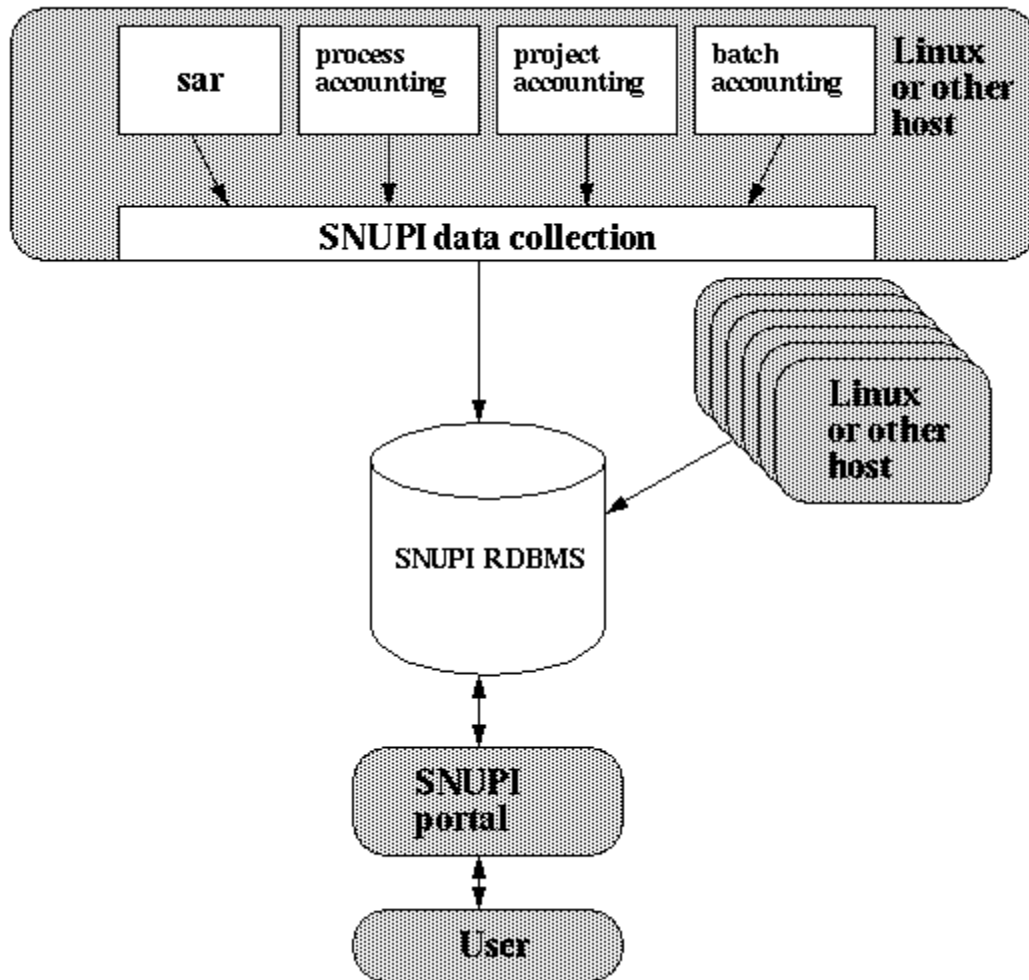
Traditional models of collecting, storing and reporting on accounting, performance and other resource utilization data available on UNIX systems employed the use of C programs, awk, sed, grep and other UNIX utilities to collect and process data available on a single host. If necessary, this data would be aggregated using UNIX utilities into a single data collection for subsequent viewing in text or Web-based reports. With hundreds, possibly thousands of systems, this model does not scale and requires a new paradigm. SNUPI introduces this new paradigm by providing a methodology to promote the collection of data and the storage of this data by a RDBMS where data aggregation and query is handled in a more effective and efficient manner.

The basic building blocks of SNUPI require the enabling of one or more of the following:

- batch system accounting,
- system activity reporting (sar)
- process accounting,
- and project accounting (if available).

The data collected by the enabled subsystems are then validated and stored into a SNUPI RDBMS schema. Any RDBMS can be used for this task. SDSC uses Oracle as the RDBMS. Once collected into the RDBMS, Web portal and other views of the data at the system, user, job and process level can be easily created. Figure 1 shows a pictorial view of this data and information exchange.

Figure 1: SNUPI Overview



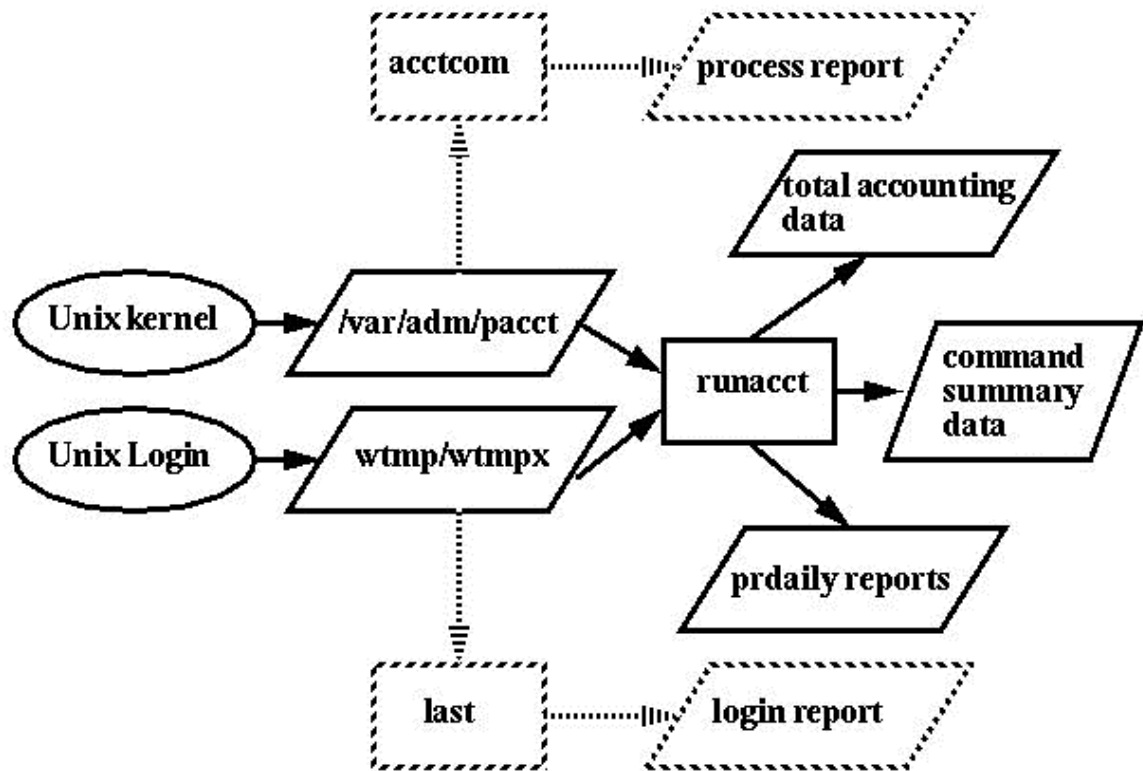
3 UNIX Accounting overview

The following sections will provide overview information regarding accounting capabilities available with UNIX in the following areas; process accounting, project accounting, system activity reporting (sar), and batch system accounting. Additionally, a summary of known accounting tools available for Linux will be described.

Process Accounting

Most variants of UNIX today provide some form of process accounting. Whether developed from System V or BSD, the methodology of process accounting on these systems was developed, in some cases, more than 20 years ago. The tools available to provide process accounting are primitive, yet useful. Process accounting, when enabled, provides a capability in the kernel to record a collection of information for each and every process completed by the kernel into a file usually called `/var/adm/pacct`. See `/usr/include/sys/acct.h` on your system for details of what is recorded. Additional information is provided by most UNIX systems in files such as `/var/adm/wtmp` or `/var/adm/wtmpx`. The `wtmp` files contain login, logout, system reboot and other information. The `/usr/lib/acct/runacct` script is usually run daily out of `adm`'s cron to collect, process and begin the reporting on process accounting data. Figure 2 shows an overview of the flow of data in a typical UNIX process accounting system. Figure 2 comes from and an overview of process accounting is available in Hazlewood[1].

Figure 2: UNIX Process Accounting Overview



Project Accounting

Project accounting is a capability that a small number of UNIX vendors have added to their process accounting subsystems. Having process accounting capabilities available on a version of UNIX does not guarantee the availability of project accounting. Project accounting is the ability to provide accounting at the process level and record a project identifier along with the process accounting data. The project identifier is separate and distinct from a user id and a group id. Examples of project accounting capable UNIX systems include: Solaris 8 (see project(4) and acctadm(1M)), Irix Comprehensive System Accounting (CSA(1M)) and Cray Unicos Cray System Accounting (CSA). Both versions of CSA have their origins on Cray Unicos (Recall that SGI at one time owned Cray Research).

System Activity Reporting

The system activity reporter, sar, is a collection of utilities that record activity counters available to the operating system on an ad hoc or periodic basis. Activity counters available include: CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging, to name a few. These counters can be recorded periodically by placing /usr/lib/sa/sadc into root's cron (See sar(1M)).

Batch System Accounting

Most large-scale UNIX systems or clusters in production computing environments employ some form of a batch system to manage the user job workload. Some examples of batch systems available for UNIX include; the University of Wisconsin-Madison's Condor, Cray, Inc's Network Queuing Environment (NQE), Platform Computing's Load Sharing Facility (LSF), IBM's Load Leveler, and Veridian System's Portable Batch System (PBS or OpenPBS). Each of these workload management systems has the capability to record information regarding the accounting of completed user jobs. This

information is (usually) collected into an accounting file for subsequent processing by the system administrator. Examples include: LSF's lsb.acct file and LoadLeveler's scheduler history file.

Linux Accounting

As mentioned earlier, Linux in its early stages having been primarily developed as a desktop solution has not had the implementation of an integrated accounting system as a high priority development item. However, accounting on Linux has not been completely ignored. The Linux kernel does have the ability to record BSD-style process accounting records. An rpm package for BSD style accounting can be found on <http://rpmfind.net/> if one searches for *psacct*. A bare minimum of process accounting information can be collected if the rpm *psacct-6.3.** is used.

Additionally, work is underway by SGI and Los Alamos National Lab to provide a Comprehensive System Accounting (SGI's CSA) package for Linux. The SGI CSA package for Linux is designed for Linux customers who require the ability to track system resource utilization by job or session and charge appropriately for the use of the resources. In the style of Cray System Accounting, SGI's CSA for Linux performs job level accounting, as opposed to providing the more familiar process level accounting. See <http://oss.sgi.com/projects/csa> for more details. To provide this job-based accounting, SGI proposes a generalized mechanism for providing process containers. SGI calls this mechanism Process Aggregates, or PAGGs. A more detailed description of PAGGs can be found at <http://oss.sgi.com/projects/pagg/>. These new mechanisms require small changes to the Linux kernel and the loading of an additional module. As of yet, they have not been incorporated into the Linux kernel (Linux-2.4.3 is current as of this writing).

The system activity reporting (sar) utility set is also available for Linux. This utility set is usually found with the package name *sysstat-**. The most recent versions of this utility set available for Red Hat 7.1 is *sysstat-3.3.5-2* and is usable with a 2.3.46 or above kernel. This package provides the *sar* and *iostat* commands, similar to the traditional UNIX counterparts. The package enables system monitoring of disk, network, and other IO activity.

Workload management systems are available for Linux systems and Linux clusters. A popular workload management system that follows the Open Source methodology and is available on Linux systems is OpenPBS (<http://www.openpbs.org/>). Commercially available workload management systems available for Linux include Veridian's PBS Pro and Platform Computing's LSF. Depending on your Linux system operating environment, the amount of money you have available to purchase a workload management system, and the availability of knowledgeable staff to maintain your workload management system, any of the aforementioned solutions provide an adequate workload management solution for a Linux system or cluster.

4 Data Collection

SNUPI data collection provides a key component for filtering, validation and insertion of data into the SNUPI RDBMS schema. SNUPI data collection is comprised of a collection of C and PERL programs. The following table describes the programs and their uses:

Table 1: SNUPI programs

Data type	Filter	Validation	Insertion
pacct	snupi_acctcom	data_loader.pl	data_loader.pl
sar	snupi_sar	data_loader.pl	data_loader.pl
Batch	sdscjoblog	data_loader.pl	data_loader.pl

The snupi_acctcom program is a modification of the standard UNIX acctcom utility which is written in C. The modifications generate the output in a form easily suitable for loading into a database. Modifications to acctcom include separating fields with commas and converting the date to a form easily loadable into a database (MM/DD/YY hh:mm:ss). Compare the original output from acctcom with the output from snupi_acctcom in the following.

```

victor@ultra% acctcom

ACCOUNTING RECORDS FROM: Sun Apr 29 16:50:00 2001
COMMAND          START END      REAL CPU MEAN
NAME             USER  TTYNAME   TIME  TIME  (SECS) (SECS) SIZE(K)
#accton         root  ?         16:50:00 16:50:00 0.02 0.01 640.00
turnacct        adm   ?         16:50:00 16:50:00 0.13 0.02 464.00
mv              adm   ?         16:50:00 16:50:00 0.05 0.01 352.00
utmp_upd        adm   ?         16:50:00 16:50:00 0.02 0.01 608.00

```

```

victor@ultra% snupi_acctcom
ultra,accton,1,root,?,04/29/01 16:50:00,04/29/01 16:50:00,0,0,640
ultra,turnacct,0,adm,?,04/29/01 16:50:00,04/29/01 16:50:00,0,0,464
ultra,mv,0,adm,?,04/29/01 16:50:00,04/29/01 16:50:00,0,0,352
ultra,utmp_upd,0,adm,?,04/29/01 16:50:00,04/29/01 16:50:00,0,0,608

```

The snupi_sar program contains a similar set of modifications to the original sar program written in C to separate the fields with commas and convert dates to a format easily inserted into the RDBMS.

The sdsccjoblog program is another C program which reads in a batch system accounting file and creates an output file in the SDSC JOBLOG format. Current batch system accounting file formats supported include LSF lsb.acct files, LoadLeveler history files, Cray CSA accounting files, and Maui scheduler history files. The JOBLOG format is comprised of a collection of 33 fields of data. These 33 fields represent a majority of the most useful pieces of data that the workload management systems (batch systems) usually collect. The following table describes the 33 fields used in the SDSC JOBLOG specification.

Table 2: JOBLOG field description

Field Name	Data Type	Description
NPACI ID	Number	Unique user id number for a user. May be different from SITE ID
SITE ID	Number	UNIX user id number for this user
LOGIN NAME	Text	User's login name from /etc/passwd
CPU	Text	Unique CPU code for each host in the grid
ACCOUNT	Text	User's account (charge code) for this job
QUEUE DATE	Number	The date the job was queued in epoch time format
START DATE	Number	The date the job started in epoch time format
END DATE	Number	The date the job completed in epoch time format
QUEUE	Text	Queue or class name
CPU TIME	Number	Cpu time used in seconds
WALLCLOCK	Number	The elapsed wallclock time in seconds
SU	Number	Total charge for this job
NODES	Number	Cumulative Sum of all nodes allocated to the job
MAXPAR	Number	Maximum node partition allocated to the

		job
NUMMPPJOBS	Number	Number of parallel applications run in the job
MAXMEMORY	Number	Memory high water mark
MEMORY	Number	Memory usage in kcore-hours
IO	Number	I/O Usage in megabytes transferred
DISK	Number	Disk charge in units local to the host from which collected
CONNECT TIME	Number	Connect time for interactive session
QWAIT	Number	Queue wait time for job
EXPF	Number	Expansion factor (Queue wait + wallclock / wallclock)
PRIORITY	Number	Priority weight value
APP_NAME	Text	Application name
JOB_ID	Number	Job id or session id of the originating host
QUEUE_ID	Text	Queueing system id code
JOB QUEUE DATE	Date	Job submission date in MM/DD/YY hh:mm:ss format
JOB START DATE	Date	Job start date in MM/DD/YY hh:mm:ss format
JOB END DATE	Date	Job completion date in MM/DD/YY hh:mm:ss format
REQUESTED TIME	Number	Amount of time requested by job
REQUESTED MEM	Number	Amount of memory requested by job
REQUESTED NODE	Number	Number of nodes requested
JOB COMP STATUS	Number	Completion status of the job

Once the sar, process accounting and JOBLOG data is collected and filtered with the above programs, the next step is to insert this data after validation into the SNUPI RDBMS schema. The data_loader.pl Perl program was written to perform both the data validation and the loading of the data into the RDBMS. The data_loader.pl utility or a data load tool available with the specific RDBMS instance can be employed for the record validation and database loading process. SQL*Loader is an example load utility available with Oracle. As part of the data load process available when using the data_loader.pl program and when JOBLOG records do not pass the validation tests, they are loaded into a JOBLOG_REJECTS table in the SNUPI RDBMS schema. The data_loader.pl program makes use of a configuration file that identifies the database instance, the database table name, authentication information, describes the structure of the input data file, and describes the validation rules.

5 RDBMS Schema

Table 3 describes the tables that make up the SNUPI RDBMS schema. Tables 4 – 8 describe the fields that are available in these tables. The JOBLOG_REJECTS table is essentially identical to the JOBLOG table with the exception of one additional field that contains the reason the job was rejected. Therefore, no additional table is listed for the JOBLOG_REJECTS table. These tables represent the location where sar, process accounting, batch accounting, user information and computer system information are stored. With host or cluster accounting data being stored into the SNUPI RDBMS, it is now easier and more efficient to enable data mining and reporting.

Table 3: SNUPI schema

Table	Number of fields	Description
SAR	7	Sar cpu utilization data
PACCT	11	Process accounting data
JOBLOG	36	Joblog (batch) accounting data

JOBLOG_REJECTS	36	Joblog (batch) reject data
PEOPLE	12	User information
CPU	15	Computer system information

Table 4: SAR

Field	Type	Description
ID	AutoNumber	Primary Key
CPU	Text	NPACI CPU Code
SAMPLE_DATE	Date/Time	Sample date/time stamp
PERCENT_USR	Number	%usr time from sar
PERCENT_SYS	Number	%system time from sar
PERCENT_IO_WAIT	Number	% i/o wait time from sar
PERCENT_IDLE	Number	%idel time from sar

Table 5: PACCT

ID	AutoNumber	Primary Key
CPU	Text	NPACI CPU Code
LOGIN_NAME	Text	User's login name
COMMAND	Text	Process name
TTY	Text	Terminal, if available
START_DATE	Date/Time	Start time of process
END_DATE	Date/Time	End time of process
ELAPSED_TIME	Number	Wallclock time of process
AVG_MEMORY	Number	Kcore memory usage
CPU_TIME	Number	Cpu time in seconds
PRIVILEGE	Number	1=root process 0=not root process

Table 6: JOBLOG

Field	Type	Description
ID	Number	Primary Key
NPACI_ID	Number	NPACI user id
SITE_ID	Number	UNIX user id
LOGIN_NAME	Text	User login name
CPU	Text	NPACI CPU Code
ACCOUNT	Text	NPACI Project Name
QUEUE_DATE	Number	Queue date in epoch time
START_DATE	Number	Start date in epoch time
END_DATE	Number	End date in epoch time
QUEUE	Text	Queue or class name
CPU_TIME	Number	Cpu time in seconds
WALLCLOCK	Number	Wallclock time in seconds
SU	Number	Billed time in seconds
NODES	Number	Number of nodes allocated
MAXPAR	Number	Max size of parallel partition allocated
NUMMPPJOBS	Number	Number of parallel programs in this job. Used on Cray T3E
MAXMEMORY	Number	High water memory mark

MEMORY	Number	Kcore memory used
I_O	Number	Number of bytes xferred
DISK	Number	Disk charge, if applicable
CONNECT_TIME	Number	Connect time in seconds
QWAIT	Number	Queue wait time
EXPF	Number	Expansion factor (Qwait+walleclock/walleclock)
PRIORITY	Number	Job priority
APP_NAME	Text	Application name if available
JOB_ID	Number	Job id
QUEUE_ID	Text	Queuing system identifier
JOB_QUEUE_DATE	Date/Time	Job queue date in (MM/DD/YY hh:mm:ss) format
JOB_START_DATE	Date/Time	Job start date in (MM/DD/YY hh:mm:ss) format
JOB_END_DATE	Date/Time	Job end date in (MM/DD/YY hh:mm:ss) format
REQUESTED_TIME	Number	Job's requested time
REQUESTED_MEM	Number	Job's requested memory
REQUESTED_PROCS	Number	Job's requestion nodes/processors
JOB_COMP_STATUS	Number	Job completion status
IS_VALID	Text	Is this entry valid? Y/N
ALTERED	Text	Has this entry been changed? Y/N

Table 7: PEOPLE

Field	Type	Description
PREFIX_NAME	Text	Mr., Mrs, Ms., Dr.
FIRST_NAME	Text	First Name
MIDDLE_INITIAL	Text	Middle Initial
LAST_NAME	Text	Last Name
EMAIL	Text	Email address
PEOPLE_ID	Number	People id
PID	Text	Another people id
SSN	Text	SSN
SOURCE_ID	Number	Reserved
SOURCE_INFO	Text	Reserved
NSF_USER_TYPE	Text	NSF User Code
LOGIN_NAME	Text	User's Login Name

Table 8: CPU

Field	Type	Description
CPU	Text	Cpu code
NSF_CPU	Text	NSF cpu code
CPU_NAME	Text	Cpu name
HOSTNAME	Text	Hostname
START_DATE	Date/Time	Start date for allocations
END_DATE	Date/Time	End date for allocations
IP_ADDR	Text	Ip address
PACI	Text	Paci system?
WEBNEWU_HOST	Text	Reserved
DOMAINNAME	Text	Domain name
SITE_ID	Number	Organization/owner

SITE_LOCATION_ID	Number	Location
VENDOR_ID	Number	Vendor
FUNC_ID	Number	Function
SUPPORT_ID	Number	Reserved

6 Portal Services

With the data being automatically collected from the required computer systems and this data being deposited into the SNUPI RDBMS, a variety of data mining and reporting capabilities are available to make information out of this data. The capabilities include linking foreign databases to Microsoft Access databases through open database connectivity (ODBC), command line reporting capabilities using Perl-DBI, and portal cgi-bin applications using Perl-DBI and/or JDBC, to name a few.

Portals, such as, the NPACI Hotpage (<http://hotpage.npaci.edu>) or the Grid Enabled Interactive Environment, GENIE, (<http://genie.npaci.edu>) provide information and other services for NPACI users. A Perl-DBI application has been developed and placed on the Hotpage and GENIE portal sites for viewing reports generated from SNUPI data. Current reports available include *node groupings by user* and *system node and expansion factor* reports. The *node groupings by user* reports allow a portal user to select the system and period of time, then, a report of all the batch jobs run on that system during the period of time selected is presented sorted by user by average node allocation size. See Figure 3 for an example *node grouping by user* report. The *system node and expansion factor* reports allow a portal user to also select the system and period of time, then, a report of all of the batch jobs run on that system during the period of time selected is presented grouped by node sizes. See Figure 4 for an example of a *system node and expansion factor* report.

As shown in Figure 3 and Figure 4, both the values in the *Userid* column and the *Number of jobs* columns are selectable and allow the SNUPI portal user the ability to drill down to the desired level of report detail. This, of course, is all back-ended by the database which contains the SNUPI schema and the NPACI schema. Current capabilities allow detail down to the user, project (account) and job level from the SNUPI portal report pages.

7 Conclusions and Future Work

With Linux clusters coming of age in the teraflop-scale supercomputer market, new capabilities must be developed and the introduction of a new paradigm must be employed to provide some production quality services to these clusters. Production services, such as, performance monitoring and resource utilization reporting at the system, user, job and process level for these clusters of possibly hundreds or thousands of nodes is required by the centers that spend tens of millions of dollars for these systems. SNUPI provides a framework for the paradigm shift from host based monitoring and reporting system to a grid-based system. By replacing existing data collection processes with a collection process that makes use of a relational database, more capabilities, efficiencies and effectiveness can be achieved for a single system, a collection of homogeneous systems, a collection of heterogeneous systems, one or more clusters, or a computational grid.

Most of the SNUPI work performed over the last year at SDSC has been done for the production compute servers which includes a Cray T3E, Cray T90, teraflop-scale IBM SP system, and Sun HPC cluster. With the potential price/performance value available with Linux clusters it is inevitable that a production Linux cluster will be put into use at SDSC sometime in the near future. SNUPI work is in progress on the Meteor cluster at SDSC (Papadopoulos [2]) and future work includes the possibility of providing a SNUPI rpm accounting package, Perl modules, development of a SNUPI web site for more detailed information regarding the project, continuing development of the portal interface to the SNUPI data for static (like Webalizer provides for Web site statistics) and dynamic

pages, and inclusion of the SNUPI system into the NPACI ROCKS Open Source Linux cluster toolkit.

Figure 3: Node Grouping By User Report

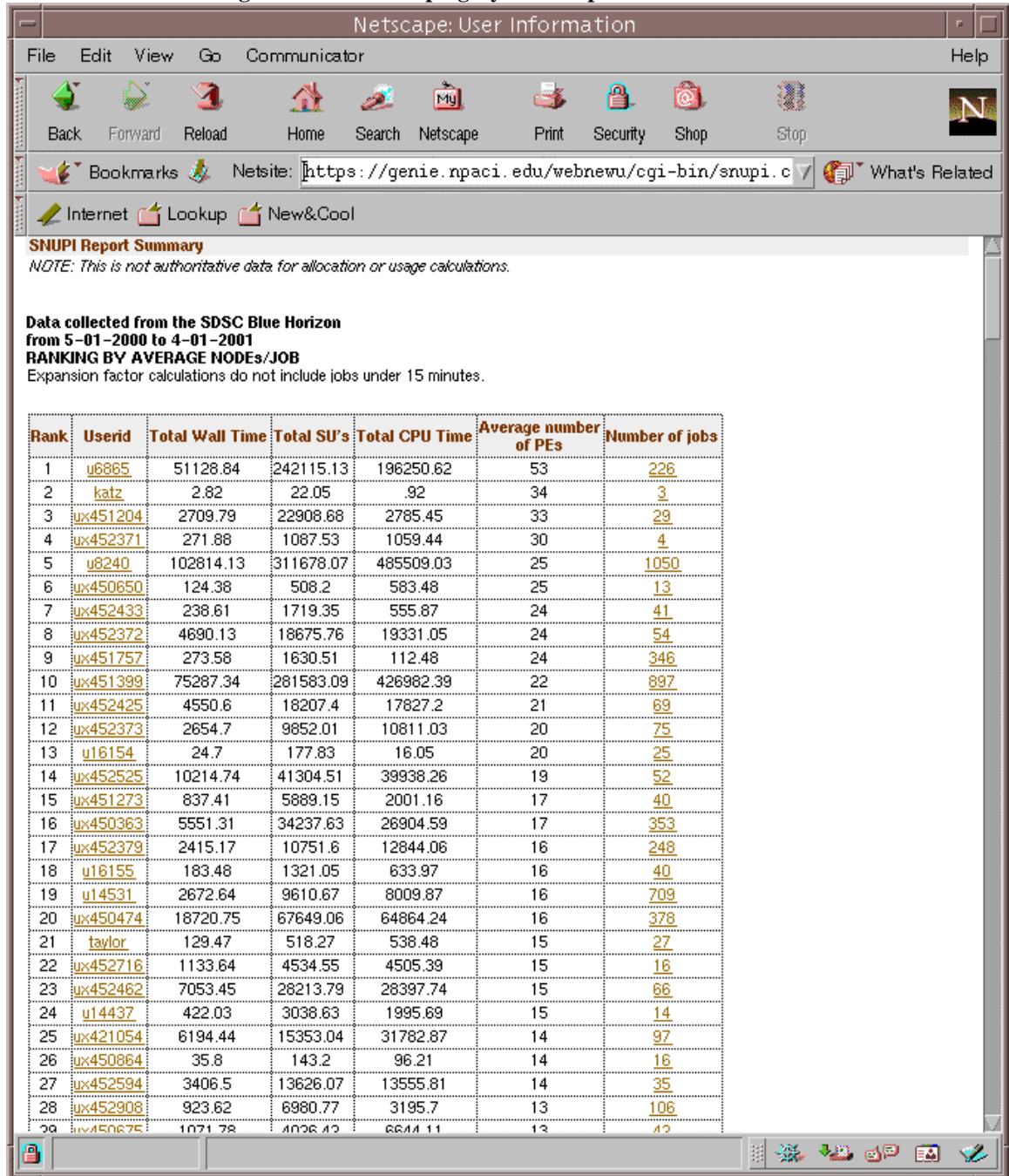
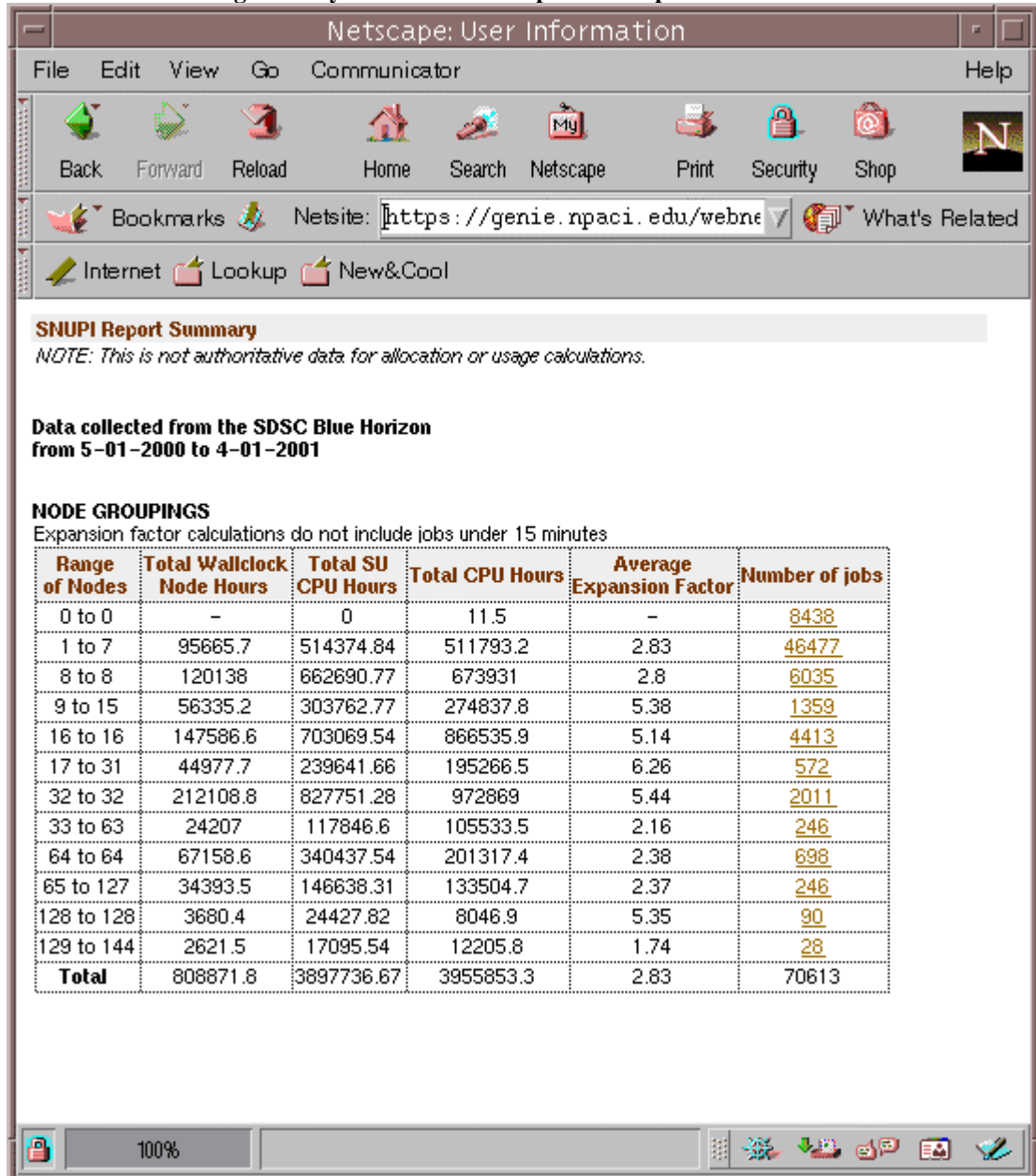


Figure 4: System Node and Expansion Report



References

- [1] Hazlewood, V.G., *Unix Accounting Magic*, SysAdmin, Miller Freeman, Inc., pp. 11–13, February 1998.
- [2] Papadopoulos P.M., Katz M.J., Bruno G. *NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters*, Cluster 2001, Newport Beach, October, 2001.
- [3] Basney J. and Livny M., *Deploying a High Throughput Computing Cluster*, in High Performance Cluster Computing, Rajkumar Buyya, Editor, Vol. 1, Chapter 5, Prentice Hall PTR, May 1999.
- [4] Ridge D., Becker D., Merkey P., Becker T.S., Merkey P., *Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs*, Proceedings, IEEE Aerospace, 1997