

Computer Performance Evaluation for UNIX

Robb Newman, Northwest Airlines

What has been standard production resource usage and capacity management practices for most mainframe shops is not in common practice in many UNIX based distributed processing shops. There are several reasons why the UNIX shops, while running on the hottest architectures and latest graphical user interfaces, have been the "backward cousins" in relation to the mainframe shops in this area. The foremost reason is because only recently have a significant number of UNIX systems moved into large production environments that necessitate performance and capacity management. The most common usage of UNIX systems to date has been by engineers and white collar professionals that have workstations that are minimally dependent on other systems in the network [1]. When there was contention for a shared resource, it usually involved a small number of users and a single server. In general terms, the environments while distributed, did not have complex dependencies for resources shared by a large number of users. In the "traditional" UNIX environment nothing needed to be done relating to resource usage and capacity planning until a system became so slow it could no longer be tolerated. Today, however, while running IS applications that are very important and have time critical processing requirements, it is unacceptable to blindly reach the point where production drops to a snail's pace before taking action.

When an IS management team realizes it has a UNIX environment in which it needs to implement standard resource usage and capacity management practices, it at first commonly finds itself in a "catch 22" type of scenario. With only a few businesses having implemented complex UNIX based distributed processing solutions, there are relatively few performance analysts with UNIX experience working in IS departments. Compounding the lack of experience problem has been until recently a general lack of "off the shelf" UNIX computer performance evaluation (CPE) tools.

An example of a complex and business critical UNIX based distributed processing environment is Northwest Airlines' Passenger Revenue Accounting (PRA) department. Today, the PRA department is processing over 300,000 ticket coupons a day. All ticket coupons are scanned and written to optical disk. During the night, sales and flight data is downline loaded from both IBM and Unisys systems into Sybase databases on the Sun datacenter servers. Using this data, AI applications determine which ticket coupons need exception processing. Exceptions would include everything from processing a refund request to debiting a travel agency that violated a ticket purchase rule, such as a 7 day advance purchase. The images of the ticket coupons with exceptions are pre-fetched from the FileNet optical jukeboxes and written to magnetic disk. The different work groups within the PRA department are assigned batches of ticket coupons to process during the day. To perform the exception processing Northwest Airlines' PRA department has almost 400

* Product of Sun Microsystems, Inc. Mountain View, CA.

** Product of SAS Institute Inc. Cary, NC.

diskless Sun SPARCstation SLC workstations in use by accounting clerks. Shared resources in the PRA network include over 30 Sun servers that are providing a variety of file, boot, application, and database services, plus several FileNet UNIX systems and a REI scanner. There are over 20 production Sybase databases, seven of which approach or exceed 1 GB in size. The largest database (imaging index database) is estimated to reach a Sybase as well as a UNIX record breaking 5.7 GB before year end.

The applications on the workstations concurrently access many of the databases as well as numerous data files. All database requests first go to a common database and then are directed to the server that has the required database. The data files are also distributed across most of the servers. It is typical for an accounting clerk to have 19 UNIX file systems NFS mounted on her/his workstation. The PRA environment processes a large volume of work using many shared resources, most of which are dependent on other shared resources. The potential for resource contention is near limitless. Should one of the several critical shared resources approach capacity, productivity for the entire environment degrades. Production UNIX based distributed processing environments with such a high requirement for resource usage and capacity management such as PRA's, were extremely rare not that long ago.

At Northwest Airlines, it did not take long for the people responsible for the PRA development project to realize the need for resource usage data in order to solve the several performance problems that arose during implementation. Due to the general lack of any tools, UNIX shell scripts were written to collect the data by executing a number of the standard UNIX system and network utilization commands such as *vmstat* and *nfsstat*. The reams of data collected were then whittled down and summarized

using the UNIX utilities *grep* and *awk*. While the process was crude and did not include any automated data analysis, it greatly improved the team's ability to find performance bottlenecks during implementation. If an IS department has had a resource monitoring function in place for any length of time, it most likely has developed a similar type of collection process. While this type of resource usage collection process works, and can even be improved with further custom development, it is very labor and resource intensive. Fortunately, there are a few "off the shelf" UNIX resource usage tool packages available today that have a much cleaner implementation and are therefore less labor and resource intensive.

One such package being marketed by Sun Microsystems is SunNet Manager. It greatly reduces the work required to collect data and enables active monitoring of the entire environment. SunNet Manager is based on the OSI Network management framework [2]. Agents run on the managed systems, collecting utilization data at predefined intervals and reporting back to the management console. The data is logged in ASCII files for later analysis. The console communicates with agents using the industry standard Simple Network Management Protocol (SNMP) [3]. This allows the product to communicate with agents running on other vendor's platforms as well as SNMP compliant network interface boxes (bridges, routers, concentrators, etc.). When Sun released its SunNet Manager product, Northwest Airlines stopped using its original process and switched to this product as the long term solution for system and network resource usage data collection, as well as for other system and network management activities. Switching from a non-standard, internally supported set of shell scripts to a standard, vendor supported, SNMP management product, like SunNet Manager, is a logical choice. It provides an easy to use graphical user interface, flexibility, and multiple vendor support.

While SunNet Manager contains some capabilities for summarizing and reporting utilization it is lacking when compared to other data analysis and reporting packages. For that reason the SAS software system on UNIX was chosen for data reduction, long term data management & retention, automated performance bottleneck and resource usage reporting, as well as statistical data analysis. While SAS CPE products exist for several operating systems, they do not yet have one for UNIX. The absence of a SAS CPE product required Northwest Airlines to write the SAS programs required to maintain, process, and report on the SunNet Manager data. It was, however, a straight forward task due to the capabilities of SAS.

Having the tools available to collect, maintain, process, and report on the resource usage data still, however, leaves problems unique to large distributed processing environments to be solved. A number of decisions have to be made concerning what to collect, and of that, what is really important. In Northwest Airlines' PRA environment encompassing over 400 UNIX computer systems and dozens of network interface boxes, the potential exists to collect several 100 MBs of performance data a day! If ever an environment provided the opportunity to prove the theorem that the process of measuring the item of interest perturbs the item being measured, environments of this size qualify. To avoid this potential problem, great effort must be exerted to limit what is being collected, and then to further reduce that to only the key indicator values.

In Northwest Airlines' PRA environment, four SunNet Manager agents are run on each of the servers and on a small (5%) random sample of workstations.

SunNet Mgr Agents Used

na.rpcnfs	collects rpc & nfs data (server or client variant depending on system type)
na.layers	collects udp data (udp variant of layers agent)
na.layers	collects tcp data (tcp variant of layers agent)
na.hostperf	collects system data

Each agent reports to the management console once every fifteen minutes. Totaled, the SunNet Manager logs accumulate approximately 10 MB of data a day. At the end of each day a combination of shell scripts and SAS programs are executed to reduce the logs by keeping only those data items currently deemed critical. The SAS data sets grow by approximately 1 MB per day. The unreduced logs are compressed and archived in case a data element not retained in a SAS data set is needed at a later date.

One megabyte of performance data is still to much data to look at daily. With this in mind another SAS program scans the day's data and generates an exception report containing only the performance "problems" observed. The process of exception reporting allows the 10 MB of raw data collected to be reduced to a couple of pages to be looked at by a performance analyst, (unless, of course, there are numerous problems).

Listed on the following page is the current set of data items deemed critical and the associated values used for the generation of the exception report. Only eleven of the thirty-three data items retained are used for exception reporting. Decisions concerning the data items and values were derived from a variety of sources, including Sun's Analyzing Network and File Server Performance paper [4,5]. It is anticipated that over time the list of key indicators will change as will some of the threshold values. Included

in the appendix is a very high level list of the justifications/reasons behind choosing these performance indicators and threshold values. If a more in-depth discussion is

required, it is recommended that at a minimum the reader obtain Sun's paper.

Key Performance & Resource Usage Indicators (using Sun's SunNet Mgr product)	Thresholds for Daily Exception Reporting ¹
Network utilization ²	35%
TCP (na.layers agent)	
packets sent	
ratio duplicate pkts sent/total pkts sent	10%
packets received	
ratio duplicate pkts received/total pkts received	10%
connections dropped by keepalive	
NFS (na.rpcnfs agent)	
RPC calls	
RPC timeouts	
ratio RPC retransmits/total RPC calls ³	10%
ratio RPC bad transaction ids/total RPC calls ³	10%
NFS calls	
ratio NFS bad calls/total NFS calls	10%
NFS reads	
NFS writes	
ratio NFS writes/total NFS calls	
UDP (na.layers agent)	
Socket overflows	5
UNIX (na.hostperf agent)	
CPU busy	
% CPU in system mode ²	
CPU avg. queue size	3
CPU interrupts	
total disk transactions	
% disk busy (for each disk on system) ²	60%
page ins	
page outs	
pages swapped in	
pages swapped out	
output packets	
ratio output packets errors/total output pkts	0.025%
ratio output collisions/total output pkts	5%

¹ Not all indicators are used for exception processing

² Agent not yet in production

³ Client data only

In addition to the daily processing, several SAS programs were written to perform a number of other CPE functions, including: resource usage reporting, high-level management reports, spotting of unbalanced resource usage, comparing current usage to past usage, trends, and using regression analysis to estimate future resource usage. To a large number of UNIX performance analysts who do not come from a mainframe background, SAS is a relatively foreign application. SAS, however, is a very powerful tool that can greatly improve a performance analyst's ability to detect problems and make correct decisions. Its use on UNIX should be seriously considered.

The ability to collect, maintain, process, and generate such plots without any human intervention is exactly what is being done in the more sophisticated mainframe shops and what can now be done in UNIX based distributed processing shops as well. Clearly resource usage and capacity management practices need to be incorporated into the management of complex UNIX based distributed processing environments, especially those responsible for critical business IS functions. While there are other methods by which to implement these practices, at Northwest Airlines the loosely coupling of SunNet Manager and SAS have been successfully implemented without investing an unreasonable amount of money.

Currently Northwest Airlines' plans are to continue expanding the SunNet Manager and SAS resource usage and capacity management environment to include the other UNIX based distributed processing systems. That includes a Convex supercomputer and an ever growing number of Sun SPARC systems and IBM RS/6000 systems.

References

- [1] E. Liederman, "An Interview with Andy Bechtolsheim," *SunWorld* 4,3 (March 1991), pp. 14-15.
- [2] Sun Microsystems, Inc., *SunNet Manager 1.1 Installation and User's Guide*. Mountain View, CA: Sun Microsystems., 1991. 277pp.
- [3] C.K. Law, "SunNet Manager," *SunWorld* 4,3 (March 1991), pp. 60-68.
- [4] Sun Microsystems, Inc. "Analyzing Network and File Server Performance," (Sun Microsystems, Inc. [July 1990]), 50pp.
- [5] G. Dennis, "PRA System Management Pilot," (Anderson Consulting, [Nov. 1990]), 32pp.
- [6] R. Berry and J. Hellerstein. "An Approach to Detecting Changes in the Factors Affecting the Performance of Computer Systems," *Proceedings of the 1991 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* 19,1 (May 1991), pp. 39-49.



Information Services
Distributed Systems Support

Robb Newman
Northwest Airlines
ms: MSPB3300
5101 Northwest Drive
St. Paul, MN 55111
(612) 726-0305

Appendix A: High Level Overview of Key Performance and Resource Usage Indicators

Network Utilization: How much of the Ethernet bandwidth is being used. General rule of thumb is that Ethernet network utilization should not exceed 30% to 35% for any length of time.

TCP Packets Sent & Received: For this site, TCP traffic is primarily database requests and replies. These elements give an indication as to the database workload characteristics.

Percent of Duplicate TCP Packets Sent & Received: Indicate that a performance problem exists. Either the servers and/or network are not keeping up.

Connections Dropped by Keepalive: Indicates that a NFS performance problem exists. The NFS server took too long to reply.

RPC Calls: For this site, RPC calls are primarily NFS calls. This element gives an indication of the NFS workload characteristics. Should the number of RPC calls significantly differ from the number NFS calls then that would indicate a change in this environment's workload characteristics.

RPC Timeouts: Indicates that a NFS performance problem exists. The NFS server took too long to reply.

Percent of RPC Retransmits: Indication of the server's and network's ability to keep up with the amount of RPC (NFS) calls.

Percent of RPC Bad Transaction IDs: Number of duplicate acknowledgments received for a single NFS request. Indication of a server and/or network problem. If the number of bad transaction ids is close to the number of retransmissions then it is an indicator that the server is having performance problems. If the number of bad transaction ids is much less than the number of retransmissions, then it is an indicator that the network is causing performance problems.

NFS Calls: This element gives an indication of the NFS workload characteristics.

Percent of Bad NFS Calls: Indication of potential hardware problems.

NFS Reads, Write, and Ratio of Writes to Total Calls: This element gives a breakdown of the NFS workload. The amount of writes is of particular interest because a number of vendors sell products which will "speed up" the writes by allowing them to complete asynchronously.

UDP Socket overflows: Indication of NFS performance problems. The server is unable to keep up with the NFS requests.

CPU Statistics: Usual CPU statistics collected on all systems. Queue size was deemed more important for exception processing, because it is a slightly better indicator of the level of service being delivered to the users[6]. With batch processing in the background, servers will run close to 100% utilized, but possibly with minimal impact on the users.

I/O Statistics: Usual I/O statistics collected on all systems. Provide an indication as to the I/O workload characteristics. Until recently SunNet Mgr was weak in this area. They now have distributed an unsupported agent that collects the same data as the *iostat* command.

Memory Statistics: Usual memory statistics collected on virtual memory systems. Provide an indication as to the memory usage characteristics of the workload.

Network Statistics: The standard *na.hostperf* agent also provides some network statistics which are not broken down. Retained are indicators of the amount of traffic generated by each system, potential hardware problems, and an additional indicator as to how busy is the network as experienced by this system.