SLAC-PUB-9202
July 2003

Overview of IEPM-BW Bandwidth Testing of Bulk Data Transfer

R. Les Cottrell and Connie Logg,
Stanford University / Stanford Linear Accelerator Center (SLAC),
2575 Sand Hill Road, Menlo Park, California 94025

***Abstract*—Grid Computing capabilities are needed for the High Energy and Nuclear Physics research of today and in the future. Groups such as the Particle Physics Data Grid are developing tools to meet these needs. An additional challenge is the evaluation and fine tuning of these applications, as well as support for long term monitoring, performance analysis, and troubleshooting. In September 2001, SLAC started the development of an infrastructure for measuring the available bandwidth and actual bandwidth utilization that is achievable by the network and various bulk data transfer applications. The purpose of these active and passive measurements is to understand what throughputs are achievable, the constraints, and how to optimize, and to make the data and predictions available for net-workers and application tuning. This paper discusses the measurement methodology and pathologies, analysis, results, and avenues for future development.**

Keywords— Network measurements, available vs achievable bandwidth, measurement infrastructure, high performance bulk throughput, international networks, quality of service, application steering, passive vs. active measurement.

I. INTRODUCTION

The strategies being adopted to analyze and store the unprecedented volumes of data being gathered by current and future High Energy and Nuclear Physics (HENP) experiments include the coordinated deployment of Grid technologies such as those being developed for the Particle Physics Data Grid (PPDG) [1] and the Grid Physics Network (GriPhyN) [2]. It is anticipated that these technologies will be deployed at hundreds of institutes that will be able to search out and analyze information from an interconnected worldwide grid of tens of thousands of computers and storage devices. This in turn will require the ability to sustain over long periods the transfer of large amounts of data between collaborating sites with relatively low latency.

The purpose of the Internet End-to-end Performance Monitoring – Bandwidth (IEPM-BW) project [3] is to develop a lightweight infrastructure, based on standard open technologies, to make active end-to-end application and network performance measurements and predictions. The measurements and predictions are targeted at high performance network links, such as those used worldwide by Grid applications and other academic and research (A&R) applications deployed over high performance networks such as ESnet, Internet2 and other A&R networks in the developed world. It may be regarded as complementary to the lighter-weight PingER [4] infrastructure in that it is not as extensive, it is more network-intrusive, and is aimed more at high performance links.

The monitoring infrastructure and results are expected to be valuable for:

- Providing planning information to applications, grid and network planners by:
  - Providing and understanding the achievable performance today in network and application (file copy & ftp) throughput.
  - Providing historical information on growth, and identifying incremental and sudden changes in performance.
- Providing trouble shooting information to networks and users by:
  - Indicating when there are incremental or sudden changes and the magnitude of the changes, and providing alerts.
  - Helping to pin-point whether a performance issue is at the network layer or application layer, or at some sub-component such as a disk.
- Providing network and applications developers, a better understanding of how networks and applications work together by providing:
  - Validation/correlation of how network performance relates to delays and loss performance (e.g. bandwidth estimators).
  - Assist users in selecting the optimum network (e.g. windows, streams, QoS) and application (e.g. compression) configuration options.
  - Identifying the critical bottlenecks such as disk, speed, file system, caching, operating system, network bandwidth, etc., for high throughput application performance.
  - Provide a public domain network performance data base, together with analyses, and provide public web accessible navigable reports and raw data. This data and information can be used for

· This work was supported in part by the Director, Office of Science. Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division uder U.S. Department of Energy. The SLAC work is under Contract No. DE-AC03-76SF00515.

Les Cottrell is with the Stanford Linear Accelerator Center (phone:650-5926-2523; fax: 650-926-3329; e-mail: cottrell@ slac.stanford.edu).

Connie Logg is with the Stanford Linear Accelerator Center (phone 650-926-2879; fax: 650-926-3329; e-mail: cal@slac.stanford.edu).

further research, for predictions and for application steering.

- Provide a base on which to test, compare and validate various bandwidth measurement techniques and tools, determine their robustness, regions of applicability, resource consumption, and accuracy, and make recommendations to developers and users.

There are several projects that are currently making continuous active (i.e. injecting probes) Internet End-to-end Performance Measurements. A fairly complete comparison made in July 1999, can be found in reference [5]. Of those projects that provide public (without subscription or some form of membership requirement) access to the data and reports: AMP [6], PingER, skitter/skping [7] make ping and traceroute measurements but no bandwidth estimation or throughput measurements; Surveyor [8] makes only one way delay and traceroute measurements. NIMI [9] is an infrastructure for making on demand measurements and does not have continuous measurements and reports. The Network Weather Service (NWS) [10] makes round trip measurements and bandwidth estimates (single stream only). The NWS also has a sophisticated prediction mechanism. Unlike the infrastructure being described here, the NWS is mainly aimed at full mesh type measurements and currently does not provide file copy/transfer application measurements. The Work Package 7 of the European Data Grid [11] have developed an infrastructure for making ping (using PingER), iperf TCP throughput and UDP measurements between seven European sites, currently they make no file copy/transfer measurements.

In the rest of this paper, we first describe the development of the measurement methodology. Next we describe the analysis and presentation of the data. We then describe results from the monitoring. We conclude with a summary of the most significant results so far, and finish up with a discussion of possible future directions.

II. METHODOLOGY

A. *First Version*

The first instantiation was for making TCP throughput measurements, using iperf [12], and secure file copy from memory-to-memory measurements, using bbcp [13], from a single measurements site, SLAC, to a set of 20-30 hosts at remote sites. The remote hosts were selected to be at PPDG sites or sites with strong collaborations with SLAC (usually HENP sites or Internet performance measurement sites). For each remote host we needed an account with secure shell ("ssh") [14] logon access. We successfully demonstrated the first instantiation for the SC2001 Bandwidth Challenge: Bandwidth to the World project [15] in November 2001.

One of the first steps was to contact people at the remote sites to request the accounts. It took about seven weeks to get accounts on suitable remote hosts at about 25 sites. The variety of forms and procedures required at the sites was a revelation in itself, ranging from just a phone call to multiple paper forms that had to be faxed, or web forms requiring considerable personal details.

We logged onto the first few remote hosts, set up the ssh keys, and copied over and installed the various initial applications (e.g. iperf, bbcp) by hand.

The diversity of the remote hosts: hardware, operating systems, network interfaces, directory structures (e.g. the userid, where to find various applications, the home directory) required a master configuration database to enable remote ssh access to execute commands. This was implemented as a Perl [16] "require" script. This database also enabled us to provide an alias for each remote host so some level of privacy could be maintained, customize how to call the measurement tools (sensors), and keep the email addresses of the contacts for each host.

B. *2nd Version*

Based on our experiences with SC2001, we rewrote the measurement infrastructure software with a major focus on improving the reliability and the ease of management, though still focused on making the measurements from a single monitoring host. To enable testing pathological cases we made it easy to call the production measurement script (`run-bw-tests`) from the command line with options to select the host and the measurement to perform. We also more formally defined the requirements for the remote host, and decided to support only Solaris 5.6 and above, and Linux 2.2 and above.

We used the Unix cron facility to schedule tasks at the measurement host. The ssh keys were saved in an AFS file on the measurement host, and its tokens timed out after 25 hours, so we had to use an AFS unattended token renew mechanism (`trscron`) to renew the tokens for cron jobs. We also added code to the measurements to verify that we had a token.

Early on we had many problems with the iperf server becoming non-functional on the remote host. For example, the remote host was rebooted and iperf was not restarted, and, in other cases the iperf process just disappeared or was still present but not responding (though in some cases it still had the TCP port attached). We were also concerned about leaving servers like iperf running all the time since it could assist in a denial of service attack. We therefore decided to start the remote server before each measurement and kill it when the measurement was complete. This helped to increase robustness, though it increased the complexity of the measurement process.

We also found it necessary to time out processes since some would hang up and run forever, or others

would run for elongated times. This also complicated the code since timing out required the code to fork processes, time them out and recover.

We added ping and traceroute (with only one measurement per hop to reduce execution time) to the measurement suite in order to have an ongoing record of round trip times (RTT) and routes. We also added other sensors to the suite for test and comparison purposes. These initially included bbcp disk to disk ("bbcpdisk"), bbftp [17] and pipechar [18]. We found we had to decrease the frequency for each round of measurements from 1 hour to 90 minutes in order to complete each round before the next round was scheduled. The start time of each round was randomized, by including a wait of up to 15 minutes with a flat random distribution. Most of the delays were caused by timeouts, so we also worked on optimizing the timeouts by carefully reviewing the reasons behind them. Since pipechar tended to run for long periods compared to the other sensors, we reduced the frequency of pipechar measurements for each remote host to one in four rounds of measurements.

To automate much of the remote host initialization and install updates we developed a tool (`remoteos.pl`). In addition we developed a tool (`getbwversions.pl`) to query and report on the configuration of the remote hosts (MHz, number of streams, TCP window sizes, Operating System (OS) etc.) as well as identifying what versions of the measurement sensors were installed.

We added about seven more remote hosts to the monitoring during this phase, and as a result documented and simplified the procedures for adding new remote hosts.

C. 3rd Version

To enable the monitoring code to run at another sites, we first ported it from Solaris to Linux. The major difficulty with doing this was caused by the different ways Solaris and Linux handled the threads we forked in the measurement script so we could timeout the various tasks. We also parameterized the locations of all the major file directories and placed the parameter values in a small configuration file. Since we planned to export the code to other sites we also added disclaimers to all the scripts at this time. We also took the opportunity to clean up the code and generalize it in many cases. Following porting from a Solaris to a Linux host we then ported the code manually to a second site, Manchester University, taking careful notes and documenting what was required. Using these notes we then automated many of the procedures required to port the monitoring code to a another site. At this stage we also added the UPPmon measurement tool to the infrastructure and began work on generalizing and simplifying adding new measurement tools.

D. Pathologies

We ran into problems getting ssh to work properly when the remote host was running SSH protocol version 2 and while the measurement host was defaulting to version 1. This was tracked down to an ssh mis-configuration error in the measurement host. We also ran into difficulties in capturing all the ssh output from commands, especially when running multiple processes. A third ssh challenge was making ssh work through a gateway machine which required cascading the ssh commands. When using an OpenSSH client with an SSH Communications, Inc. server we found we had to reformat the public key before saving it on the server host. There was also confusion about exactly where to save the public key on the server especially for protocol version 2 servers: the directory was sometimes called .ssh and other times .ssh2; sometimes the public key was appended to the file auhorized_keys, other times authorized_keys2; sometimes it was placed in a separate file with a pointer being placed in a file called authorization.

Usually we were able to copy the measurement executable that had been built at SLAC for the appropriate OS version, to the remote host. However, in some cases there were library incompatibilities. In about 40% of the iperf cases, and 20% of bbcp and bbftp cases we had to `make` the executables on the remote host. The information on whether an executable had to be made on the remote host was kept in the configuration database. Executables such as ping, traceroute and pipechar did not need anything to be installed on the remote host.

When measuring disk to disk throughput on fast links we had to be careful to understand the effects of caching. We used the Solaris Unix File System mount `forcedirectio` facility [19] to ensure that the source files were not cached (use the Solaris `man mount_ufs` command for more details) when we were reading them on the measurement host. Though this gave us a realistic estimation of disk read speed for large files, it also meant that for high speed links the gating factor in overall file transfer rates was often the speed of the disk reads. Further work is in progress to understand disk throughputs for various Operating Systems and file systems, with and without caching and with and without committing the writes, on about 25 different hosts. If possible we requested large amounts disk space at the remote host. Until we have sufficient disk space set aside we used space in /tmp. At the same we checked and recorded whether the /tmp space was using memory (e.g. swap space on Solaris).

Some hosts blocked a protocol, or rate limited throughput for the port. If this was permanent, for example a host did not respond to pings, then it was simple to add this to the configuration database. In other cases, ssh access or the applications server port would be blocked due to security concerns. To detect such failures we logged attempt information and developed a tool (`codeanal`) to analyze the logs to highlight

repeated failures, so we could send email to the host contact. For cases where we were unsure if the port was blocked, we tested by running the iperf server on the port at the remote host, and then running the iperf client at SLAC to see if the port was accessible. We checked what ports were required by a particular application by reading the man pages and also by tracing the packets by running tcpdump [20].

At any given time, we observed that about 20% of the hosts would be unreachable via ssh. Included among the reasons were: the host was changed or removed; difficulty in getting account/password or getting ssh to work on a changed host; problems with using Kerberos credentials to access the remote host in an unattended fashion; concerns at the remote site about charging for usage; difficulties in interworking between various versions of ssh; host was wrongly configured; link to host was down for a long period (e.g. several weeks in one case where a new link from Europe to Chicago was being brought up).

E. Current Deployment

Currently the participants to which the tests are made include: Argonne National Laboratory (ANL) in Chicago IL, Brookhaven National Laboratory (BNL) in Long Island NY, California Institute of Technology (Caltech) in Pasadena CA, Fermi National Accelerator Laboratory (FNAL) in Chicago IL, Thomas Jefferson National Laboratory (JLAB) in Newport News VA, Lawrence Berkeley National Laboratory (LBNL) in Berkeley CA, San Diego Supercomputing Center (SDSC) in San Diego CA, University of Wisconsin (UWisc) in Madison WI, National Energy Research Scientific Computing Center (NERSC) in Oakland CA, University of Florida (UFL) in Gainesville FL, Indiana University (IU), the University of Michigan (UMich) in Ann Arbor MI, CERN in Geneva Switzerland, KEK in Tokyo Japan, Rutherford Laboratory near Oxford England and Daresbury Laboratory near Liverpool England, Rice University in Houston TX, University of Delaware in Newark DE, Oak Ridge National Laboratory (ORNL) Oak Ridge TN, NASA/GSFC, IN2P3 in Lyon, France, Los Alamos national laboratory (LANL) in Los Alamos NM, Tri-Universities Meson Factory (TRIUMF) in Vancouver Canada, Internet2 Southern Exchange (SoX) in Atlanta GA, INFN/Rome and Milan, NIKHEF in Amsterdam, Netherlands, and of course the Stanford Linear Accelerator Center (SLAC) near San Francisco CA. There are currently (April '02) 33 active destination hosts at about 30 sites in 8 countries.

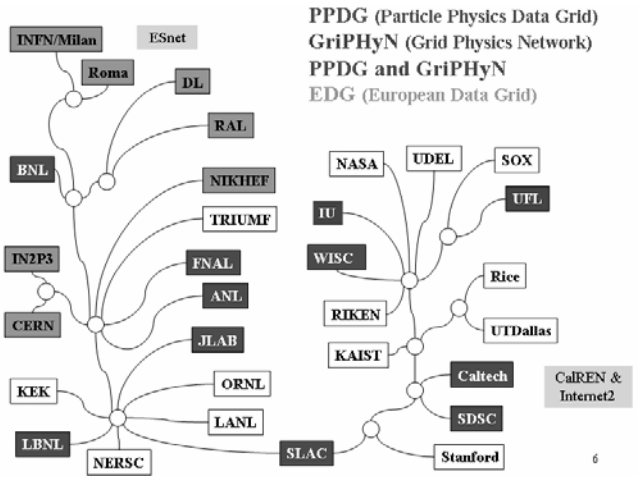

Figure 1: Topology of particpiating sites

Fig. 1 shows the logical routes between SLAC and many of the participants. Sites displayed to the left are routed across ESnet from SLAC, and sites to the right are routed across CalREN and Internet2. SLAC has OC12 (622Mbps) connections to ESnet and Internet2. Wide-area network connectivity between these sites is almost entirely managed by the Energy Sciences Network (ESnet) and the Internet2 networks. In this paper, Internet2 is considered to be the Abilene backbone network, and the regional connector networks such as the California Research and Education Network (CalREN). .

F. Performance Measurements

Current performance measurements ("sensors") include:

- Ping – If the ping fails, no further tests are done for the node
- Traceroute – if this fails the other tests are done anyway
- Iperf – if this fails the other tests are done anyway
- Bbcp memory to memory copy (bbcpmem) – if this fails, the bbcp disk to disk (bbcpdisk) test is not done
- Bbcp disk to disk (bbcpdisk)
- Bbftp
- UDPmon [21]
- Pipechar – done infrequently

In the future, others may be added and some of the above may be discontinued.

The sensor output from a test is captured, identified with a token, time-stamped and written to a "log" file. For each remote host there is one log file per day.

At the end of a run, two scripts (`extract-data` and `codeanal`) are called to process the log files for that day. `Extract-data` extracts the bandwidth results and other information related to the parameters of the test and saves it in flat data ASCII files for analysis, plotting and future reference.

`Codeanal` processes the logs for performance information on the running of the `run-bw-tests` code. The output of `codeanal` is a web page [22] that

node1.ctaalliance.org	00	01	02	03	04	05	06	07	08
PING		10	10		10	10		10	
TRACECMD		3	2		2	2		2	
SERVER		17	27		17	27		17	
IPERF		12	13		12	12		12	
BBCPMEM		20	NRH		20*	NRH		19	
BBCPDISK		21	-43		19*	-43		20	
PIPCHAR		NR	135		NR	NR		189	
BBFTP		16	16		15	25		19	
TOTAL		99	280		95	153		288	
Total Runtime	**915**								

Figure 2: Part of code performance report

presents, in tabular format, the amount of time it took to run a test (if it was successful) or a failure code if a test failed.

Fig. 2 shows a portion of code performance web page that is created by `codeanal`. "**NRH**" indicates that the remote host could not find a route back to the measurement host. The "*****" indicates that bbcp detected that "Sink I/O buffers > 25% of available memory; copy may be slow". "NR" indicates that pipechar was intentionally not run. The "**-43**" indicates that the sensor timed out and did not complete in the allotted time. This helped to establish the time limits for each test. In addition the page provides an overall picture of the reliability and performance of the test code itself. Patterns of failure are easily recognizable and can thus be dealt with in a timely manner.

III. ANALYSIS

A. *Time Series*

The first displays created were the time series displays [23]. That is, plot n days of data by date and time. Currently we use n=28. This displays all the measurements, while providing a reasonable density of points, and allowing immediate visualization of several weeks data. We also provide access to 56 (4 points/day) and 180 day (1 point/day) plots. An example of a 28 day time series is shown below in Fig. 3. This shows throughputs/bandwidths measured from SLAC to JLab for 28 days starting on April 6 '02. The measurements are made with iperf (top x), bbcp memory to memory (grey + symbols), bbcp disk to disk (black + symbols), bbftp (grey filled in boxes), average ping RTT (line) and min ping RTT (grey vertical bars). The time series plots are valuable for giving an overall view of the performance and relative throughputs, as well as identifying step changes and diurnal patterns in performance, long-term trends, and rough correlations between the sensors. For example in Fig. 3, one can see that when the average ping RTT increases the throughput is also reduced.

B. *Scatter Plots*

As can be seen by the header on the graph, other types of plots are also available. Clicking on "Scatterplots" brings up a panel of scatter-plots that indicate correlations. The panel plots each of the tests against the others. Fig. 4 shows part of a panel of scatter plots of the throughput/bandwidths measured by each test (e.g. iperf) vs. each other test. The line through the origin is a least squares fit to a straight line while constraining the line to go through the origin. The other line is a similar fit but without the constraint. The parameters of the fits are also reported.

In addition to the scatter plots between each measurement sensor for each remote host, we also provide superimposed scatter plots of each pair of sensors for all hosts. Each host is identified with a different symbol/color combination for its set of points. This facilitates looking for general trends such as whether a bandwidth estimator (e.g. pipechar) fails for some throughput domain, or whether there are common limits on throughput for disk related operations, etc. Together with the combined scatter plots, various statistical summaries of the data are reported including, the average and standard deviations for each coordinate, and the correlation coefficient *R*.

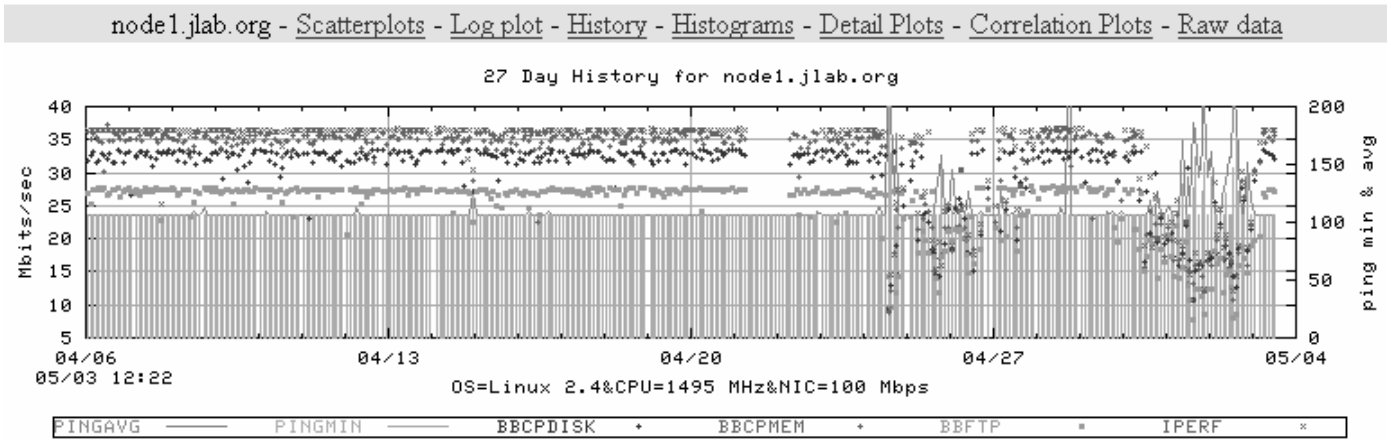

Figure 3: Time series display of performance from SLAC to JLab

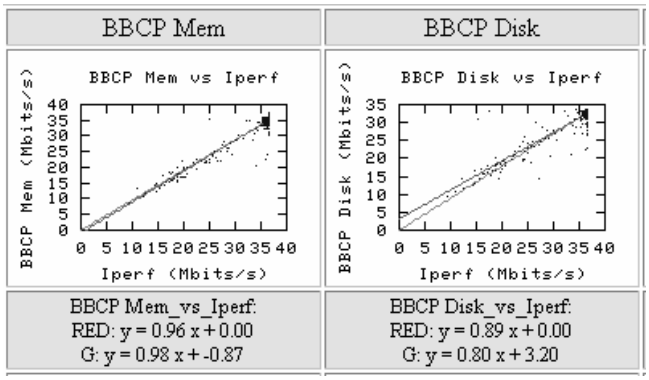

Figure 4: Scatter plot examples for JLab

C. *Histograms*

The results from the tests are also plotted in a frequency histogram panel. The histograms provide a clear picture of the distribution of the values. Fig. 5 is an example from one of the histogram panels and shows part of a frequency histogram panel for throughput measurements from SLAC to ANL for 21 days starting March 21 '02. The 2 histograms shown are for (left to right) iperf, and bbcpmem, and they are for CERN.

D. *Tables*

Tabular information with drill down URL links to more detailed information is also provided. The tables contain remote host configurations, the most recent measurement results, and more detailed statistical results such as averages, standard deviations, errors, correlation coefficients, etc. In many cases, the tables also provide access to the analyzed data in space or comma separated value (csv) format for further analysis.

IV. RESULTS

A. *Measurement Results*

To evaluate the effect of the duration of the individual measurements on the throughput measured we selected durations of 2, 5, 10, 20, 40, 80, 160, 250 and 320 seconds, and window sizes of 256, 512, 1024, 2048 and 4096 KBytes. For each of the above possible pairs we made a single stream measurement of the iperf TCP throughput 17 times from SLAC to the remote host. We used a single stream since multiple streams are in general more agile to adjusting to network conditions such as loss, and are thus expected to require less time to

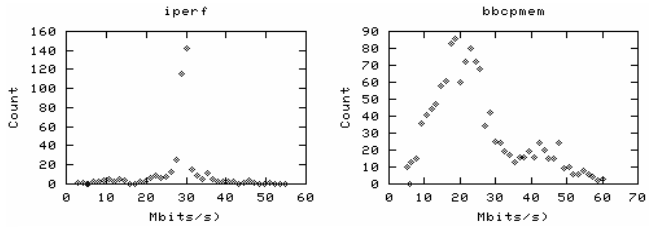

Figure 5: Examples of histogram reports

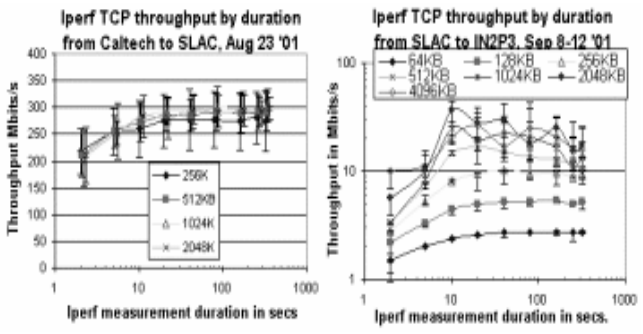

Figure 6: Iperf TCP throughput to Caltech and IN2P3.

reach a stable throughput rate. The results for SLAC to Caltech and SLAC to IN2P3 are shown in Fig. 6, which shows the iperf median TCP throughput measured from SLAC to Caltech (RTT 40 msec.) and to IN2P3 in Lyon, France (177 msec. RTT) for various window sizes. The points are the medians of each set of 17 measurements, and the error bars are determined from the Inter Quartile Ranges (IQRs).

It is seen that though the medians continue to rise for durations of over 10 seconds (by about 10% going from 10 to 20 seconds) to within the accuracy of the measurements this is a small effect. Since we are interested in the performance for long duration transfers, we took the miimum duration that was representative of a long duration transfer. So for most of our measurements we settled on a duration of 10 seconds.

It was pointed out to us by Matt Mathis [24], that the optimum duration may depend critically on the buffer sizes in the first router(s) in the path. Also as one moves to larger RTT times bandwidth products, slow start will take longer and thus more time will be needed for throughput to get close to capacity. As pointed out to us by Tom Dunigan [25], if you know when slow start is over, then you can more quickly get the bandwidth by just using Web100 [26] to look at how many bytes have been transferred over the last second once slow start is over.

B. *Impact on CPU Utilzation*

Fig. 7 shows the behavior of the ratio of measurement host MHz / iperf TCP throughput as a function of the speed (MHz) of the source. The utilization was obtained using the Unix "time" command and is the sum of the "system" and "user" times. The points are the medians for each complete set of measurements made with the various window sizes and streams. The error bars are the Inter Quartile Range for each complete set.

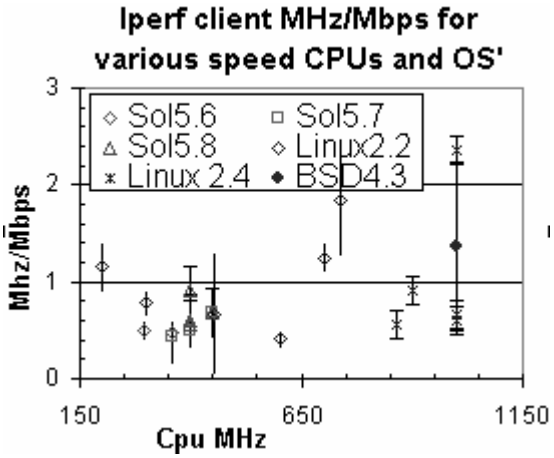

Figure 7: Ratio of measurement host MHz utilization to Mbits/s transferred

It is seen that there is a lot of variability in the observed values. More measurements would be needed to determine whether one OS is superior to another in terms of minimizing MHz/Mbps. The averages of the median values of MHZ/Mbps are: all 22 hosts 0.85+-0.5 (11 Linux hosts: 1.02+-0.6, 11 Solaris hosts: 0.68+-0.27).

C. Comparing TCP with application throughputs

We compared iperf TCP throughput versus bbcpmem throughput. An example of a scatter plot for iperf TCP vs. bbcpmem measurements, made for 28 days starting April 2 '02 between SLAC and about 30 remote hosts, is shown in Fig. 8.

The line shows a linear regression fit with the parameters *y=0.88x-5.5*. There was excellent agreement (correlation coefficient squared R^2 ~ *0.91*) when comparing the measurements for all remote hosts, with bbcpmem averaging about 88% of the iperf TCP throughput. It is reasonable to expect the bbcp throughput to be less than that of iperf since iperf simply measures TCP throughput while bbcp is a secure copy program built on top of TCP. Bbcp also synchronizes the streams so a slow down on one stream will cause others to slow down, whereas for iperf the streams are asynchronous. The points in a given cluster observed in Fig. 8 are usually associated with a given host. In fact, for many of the hosts, the correlation for that host is quite weak since the measurements all cluster around small ranges. It is only when we compare the measurements for all hosts that we get a strong correlation. The reason why some of the clusters are vertically or horizontally oriented will be the subject of further investigation.

When we compared iperf TCP throughput with bbcpdisk throughput, horizontal lines indicated a non-network throughput constraint. We believe this is due to disk I/O and are investigating this further to understand the effects of caching, file system, committing the data to disk, etc. We also observed vertical lines just under

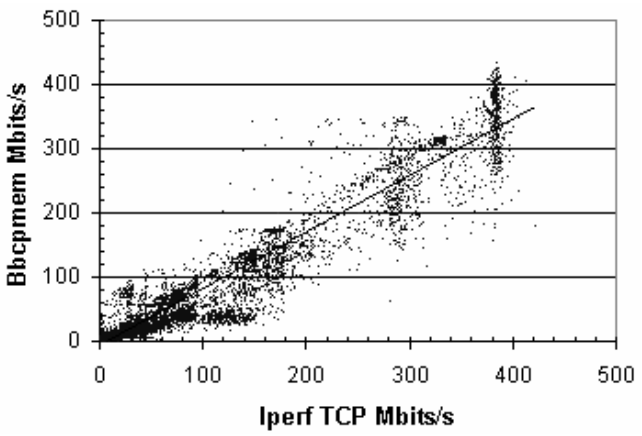

Figure 8: Bbcp memory to memory throughput vs, iperf TCP throughput

10Mbps, 100Mbps and 150Mbps where the constraint was probably network related (i.e. Ethernet, Fast Ethernet and OC3).

Fig. 9 shows a typical scatterplot of bbcpdisk vs. iperf TCP throughputs for about 30 remote hosts as seen from SLAC. The different symbols and shades indicate different remote hosts. The line is for a linear least squares fit of bbcpmem to iperf.

We also compared the iperf throughput with the minimum available predicted by pipechar. An example is shown in Fig. 10. The different shades and symbols represent different remote hosts. Since iperf is using TCP while pipechar uses packet trains, one might expect the agreement not to be excellent. In general the agreement is particularly poor for 6 hosts with throughputs above 100Mbits/s. About 50% of the hosts have reasonable agreement. Pipechar uses packet dispersion methods to estimate bandwidth. At high bandwidths, this requires increased accuracy (better than tens of microsecs) of the measurement clock. Packet dispersion techniques using host timings will probably also suffer badly if the network interface card (NIC)

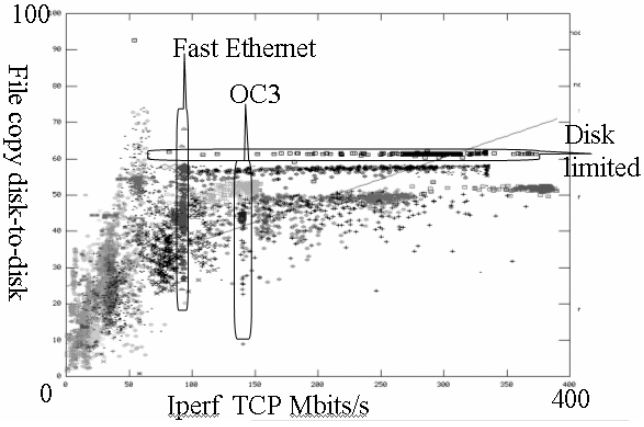

Figure 9: Bbcp disk to disk vs. iperf TCP throughput

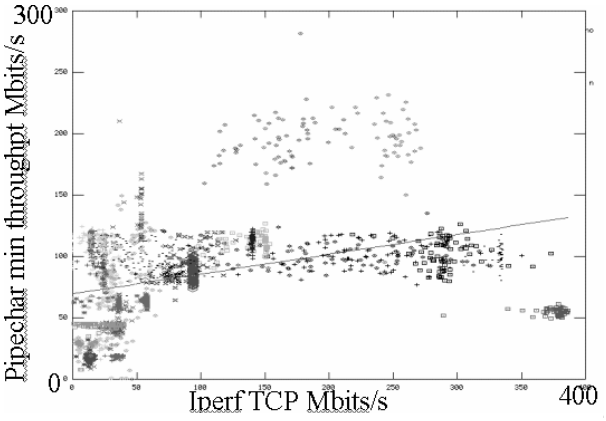

Figure 10: Pipechar estimates vs. iperf TCP throughput

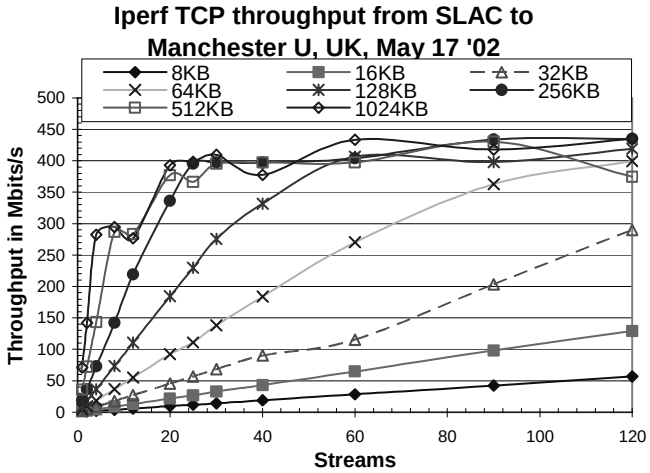

Figure 11: Ten second iperf TCP throughputs from SLAC to Manchester University

coalesces interrupts inbound or does buffering and fragmentation outbound.

D. Windows and Streams

To determine the optimum window size and number of parallel streams for each site we first configured the hosts to use the maximum buffer and window sizes recommended in [27]. Then we used iperf to send TCP bulk data for 10 seconds from SLAC to an iperf server at the remote host. For each site we used window sizes from 8Kbytes to 4Mbytes, and for each window size we used different numbers of parallel data streams from 1 up to 120 to comprise each transfer. The sequences of window sizes and number of parallel streams were deliberately chosen so they did not monotonically increase or decrease. Simultaneous with the data transfer, we also sent ten 100 byte pings separated by 1 second, each with a 20 second timeout. Following each transfer, we also sent 10 more pings with no iperf transfer. The idea of the two sets of pings was to evaluate the RTT with and without competing iperf TCP transfers.

We then plotted the throughput versus streams for each of the window sizes. See Fig. 11 for a typical example in this case from SLAC to IN2P3.

It is seen that, for small window sizes, the throughput grows linearly with number of streams. On unsaturated links, we can use this feature to generate TCP traffic with a known load. As the window size increases (in this case beyond 64Kbytes), the throughput begins to saturate as the number of streams increases. Since typical operating system default maximum window sizes vary from 8kBytes to 64kBytes, it is apparent, that in cases such as illustrated in Fig. 11, many streams may be required to achieve optimal throughput. We selected a windows streams combination that achieved about 80-90% of the maximum throughput measured, while minimizing the number of streams. We wished to minimize the number of streams since each stream consume resources (memory, a process, and extra CPU cycles).

E. Impact on Others

To investigate the impact of high bulk throughput on other users, we used iperf to send TCP traffic from a Sun Ultra 2 running Solaris 5.8 to a similar host in CERN. Iperf was set to have 1024KByte windows and 20 parallel streams. We ran iperf in this fashion for 35 minutes from 12:26 April 25 '02, simultaneously measuring the ping RTT and loss (we sent a 100 byte ping once a second with a timeout of 20 seconds). While doing this we also observed the link utilization. The aggregate measured throughput from SLAC to CERN was about 120Mbps, which was close to the bottleneck bandwidth at the time. The ping loss was about 0.15%, the minimum ping RTT was 166ms, the average was 295ms and the maximum was 408ms. We followed this up by measuring the ping RTT and loss for 24 minutes without generating any iperf traffic starting at 13:02. In this case there was no packet loss, and the minimum RTT was 166ms, the average was 167ms and the maximum was 377ms. The effect on the ping RTT distributions is seen in Fig. 12. The blue triangles indicate the RTT with no iperf load, and the red squares with an iperf load. The bottom axis is the ping RTT. The lines on the graph represent the Cumulative Distribution Functions (CDF) and their axis is labeled on the right.

It is seen that the unloaded RTT is sharply clustered between 166 and 170 msec, while the loaded RTT distribution is fairly flat for over 150msec above the minimum RTT. We are looking for ways to ameliorate this effect. Some possibilities include using the QBone Scavenger Service (QBSS) [28], self rate limiting the application (i.e. enable the application to restrict its throughput), providing a feedback loop for the

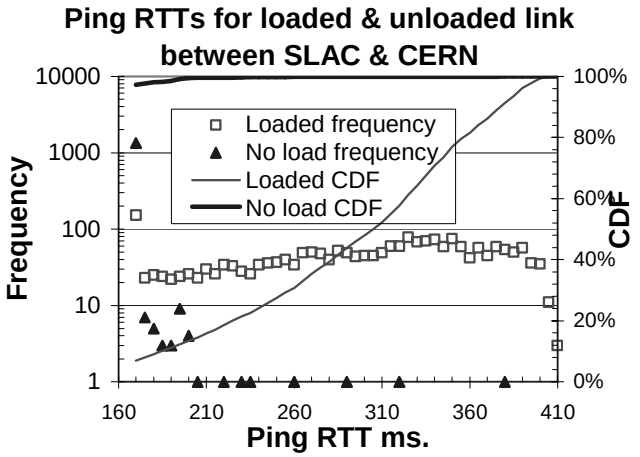

Figure 12: Ping RTTs with and without simultaneous iperf load

application by using Web100 to measure the RTT and/or retransmissions and using these values to adjust the application's offered throughput.

F. Forecasting

To enable use of the measurements for guiding applications, we looked at how to forecast the throughput from existing measurements. We developed a very simple prototype that, given a time, provides the average and standard deviation of the previous 5 measurements. An example comparing the actual vs. forecasted values for the SLAC to U. Wisconsin path is seen in Fig. 13. The plus signs with the error bars are equivalent to the standard deviation from the previous 5 measurements.

Besides being useful to assist applications, forecasting may also be useful to decide how often to make active measurements. For example, if the measurements are very consistent, then we may not need to make a measurement as frequently as otherwise. We also calculated the average error for the above type of measurements as:

error=average(abs(forecast-observed)/observed)

The average errors between the forecasted and observed values are shown in Table 1.

Table 1: average error between the forecasted and observed measurements.

33 hosts	iperf TCP	bbcp mem	bbcp disk	bbftp	pipechar
error	13%	23%	15%	14%	13%
Stdev	11%	18%	13%	12%	8%

It can be seen that, even with this simple forecasting method, reasonable agreement is achieved (better than 25% in most cases) for 90 minutes after the last measurement.

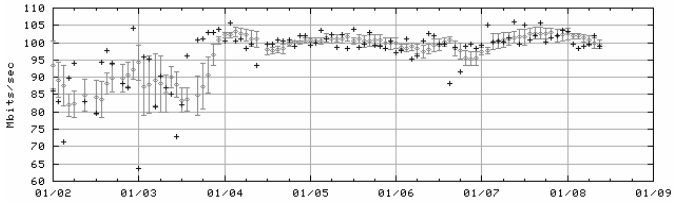

Figure 13: Forecasting iperf TCP throughput: observed (+) and forecasts (x with error bars) made for last 5 observations

G. Passive and Active Measurements

To validate whether the sensors were reporting the correct throughputs, we read the Netflow records [29] from a Cisco 6506 containing an MSFC module for routing. The Cisco 6506 is located at the SLAC network border and is connected to the outside world by 1 Gbit/s links, one to ESnet, the other to Stanford University and thence to CalREN 2. The methodology of collecting the Netflow records is described in [30]. A Netflow record includes the source and destination IP address and port 4-tuple (source IP, destination IP, source port, destination port), the protocol, the number of packets and bytes, the start and end times and active time for each flow/stream. The flows were sorted by source and destination IP address and start time and flows starting with the same source and destination address that start within 2 seconds of one another are assumed to belong to a given application process. Thus we could aggregate the throughput for the application instance as the sum of the bytes for all streams divided by the sum of the active times for all streams divided by the number of streams. Typically we see about 10-20K applications per day transferring greater than one MByte of data between 100 to 300 different pairs of hosts.

We then compare the passive Netflow throughputs, calculated as above, with the throughputs recorded by the associated active application (sensor) by means of time series, scatterplots, calculating *err = (passive-active)/passive* and the correlation coefficient *R.* An example of a time series is shown in Fig. 14. Fig. 14 shows the time series of active and passive throughput measurements for iperf from SLAC to Caltech for 28 days starting April 1 '02. For this case the *err = 2%* and

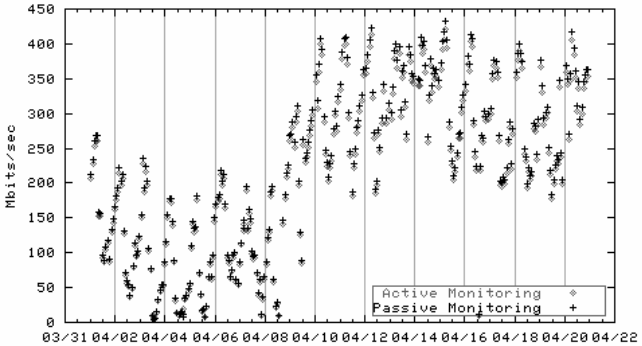

Figure 14: Example of time series of active and passive throughputs from SLAC to Caltech, Mar-Apr '02

R=0.99 and the agreement is seen to be excellent.

The overall agreements, for 28 days starting April 1 '02, are shown in Table 2 below. The ranges are the 25 percentile and 75 percentiles. On average the throughput for each sensor to each host was measured 279 times in that period. We excluded remote host-sensor combinations where there were fewer than 50 measurements. It is seen that in general the correlations are strong. The *err* ranges indicate that there is not an overall systematic difference between the active and passive measurements. For a given remote host-sensor the active measurements can be systematically greater (i.e. the *err* is negative) than the passive measurements and vice versa for another remote host-sensor. On average the bbftp active sensor reported throughputs 25% lower than observed by the passive measurement. This may be since bbftp starts the timer before or after it starts or ends the actual TCP session. In general the sign of the *err* would track for the bbcp and iperf measurements for a given host (i.e. if the iperf *err* was negative for a given host for then the bbcp *err* would also be negative). The strongest correlations are for iperf followed by bbcpdisk. The bbftp correlations are generally much weaker. Typically the agreement is poorer for remote hosts with lower throughputs and the disagreement for low throughputs is usually with a negative *err*. More work will be required to understand why bbftp has poorer agreement between active and passive measurements, and some of the details of the other disagreements. However, in general there is excellent correlation between the active and passive iperf and bbcp measurements, and the *errs* are < 5% for the majority of remote hosts.

This agreement is important since it encourages us to include passive measurements into the throughput measurement database. Thus we now have an important extra (roughly 100-300 pairs per day) source of throughput measurements for pairs of hosts matching real use patterns, but which do not add any extra load to the network.

Table 2: Errs and correlation coefficients (*R*) between active and passive measurements for throughput sensors for about 25 remote hosts seen from SLAC in April '02

Metric	iperf TCP	bbcp mem	bbcp disk	bbftp	Over all
err median	0%	-3.9%	-5.0%	25%	2.0%
err range	-4.5%, 2.0%	-7.5%, 5%	-14.5%, 2.5%	21.5%, 37.5%	-7%, 12%
R median	0.99	0.86	0.94	0.68	0.94
R range	0.98, 0.99	0.8, 0.98	0.82, 0.98	0.39, 0.89	0.73. 0.99
Remote hosts	27	24	23	23	

V. CONCLUSIONS

Preliminary results from IEPM-BW so far indicate:

- Reasonable estimates of throughput can be made in our case with 10 second iperf measurements. This is much shorter than it typically takes many bandwidth estimators, such as pipechar, to make an estimate.
- Roughly speaking one needs about 1 MHz of CPU cycles to provide 1 Mbit/s throughput on today's CPUs and OSs.
- Throughputs can vary by an order of magnitude with time of day or day of week etc.
- The bbcp file copy rates from memory to memory are typically (25 to 75 percentile) in the range of 58% to 96% of the iperf TCP throughputs.
- Disk to disk file copy rates are typically 90% of the memory to memory rates for rates below 60Mbits/s, Above 40-60Mbits/s performance can vary depending on disk/file system performance, caching etc. Uncached disk performance for the remote hosts we were measuring to appears to top out at between 4 and 8Mbytes/s in most cases.
- When running high throughput applications, the RTT for other users can be noticeably increased.
- We are able to predict performance 90 minutes into the future with better than 25% accuracy.
- Passive Netflow measurements agree to within 5% with active measurements for most remote hosts.
- Using standard operating systems (Linux and Solaris) for the monitoring and remote hosts, enabled us to easily take advantage of new sensors which in some case have not been ported to less popular operating systems.
- Using a hierarchical infrastructure, where each monitoring host selects the remote hosts to monitor (as opposed to a full mesh measurement infrastructure), lends itself very well to the requirements of HENP where there a few sites providing access to large amounts of data, and each site often collaborates with a different set of remote sites.

The next steps include: documenting the implementation, procedures and program logic, and porting the monitoring infrastructure to more sites. We plan to start the porting with Manchester University, FNAL and INFN/Milan. Initially, to preserve flexibility, each monitoring site will save its own data, and perform its own extraction/analysis and reporting. We are also working on validating other sensors such as pathrate [31], pathload [32], GridFTP [33], INCITE [34], UDPmon, etc., and hope to select a new recommended set of base measurement sensors. We have made the data available to the NWS project, and will look at more sophisticated methods to make the forecasts, as well as how to insert our data into their infrastructure. We hope the forecast study will also help to optimize the

frequency of measurements. In addition we will integrate Web100 into the measurements which, besides providing detailed information from TCP, may also help in optimizing the duration of measurements. The analysis of the active measurements vs. the passive measurements of users' applications is just beginning and further understanding of discrepancies is needed. Further work could involve looking at the effects and applicability of compression, application rate limiting, providing tools to assist in making applications such as bbcp network aware, and making the data available via more standard publish/subscribe methods.:

ACKNOWLEDGMENT

We would like to acknowledge the help of Manish Bhargava, Jerrod Williams of SLAC and Fabrizio Coccetti of INFN/Trieste in developing display and analysis code. Warren Matthews provided much assistance in installing Web100 and configuring the measurement hosts. We are indebted to Andrew Hanushevsky of SLAC for providing guidance and adding features to bbcp to improve its measurement capabilities. Jin Guojun of LBNL provided assistance in understanding the pipechar results and providing new versions to test. We also owe a large debt of gratitude to all the contacts at the remote sites who helped us to get accounts and put up with our questions. Finally we would like to acknowledge many useful discussions with Matt Mathis of PSC, Brian Tierney of LBNL, and Rich Wolski of UCSB.

REFERENCES

[1] Particle Physics data Grid: http/www.ppdg.org/.

[2] GriPhyN Project: http://www.griphyn.org/

[3] Internet End-to-end Performance Monitoring - Bandwidth to the World (IEPM-BW) project http://www-iepm.slac.stanford.edu/bw

[4] W. Matthews and R. L. Cottrell, "The PingER Project: Active Internet Performance Monitoring for the HENP Community", IEEE Communications Magazine Vol 38 No. 5 pp130-136, May 2000[i]

[5] R. L. Cottrell, " Comparison of some Internet Active End-to-end Performance Measurement projects", http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html

[6] A. J. McGregor and H. W. Braun, "Balancing cost and utility in active monitoring: The AMP example.," INET 2000, July 2000.

[7] Skitter/skping http://www.caida.org/tools/measurement/skitter/skping/index.xml

[8] "Introduction to the Surveyor Project", http://www.advanced.org/csg-ippm/

[9] V. Paxson, A. Adams, M. Mathis, "Experiences with NIMI", Passive and Active Measurements workshop 2000.

[10] R. Wolski, "Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service" in 6th High-Performance Distributed Computing, Aug 1997.

[11] "WP7 Networking", http://www.gridpp.ac.uk/wp7/index.html

[12] Iperf: http://dast.nlanr.net/projects/Iperf/

[13] A. Hanushevsky, A. Trunov, R. L. Cottrell, "Peer-to-peer Computing for Secure High Performance data Copying" Computing In High Energy Physics 2001, pp 444-447., Biejing 2001. Paper can be found at: http://www.slac.stanford.edu/~abh/CHEP2001/7-018.pdf

[14] D. J. Barrett and R. Silverman, "SSH, The Secure Shell: The Definitive Guide", O'Reilly & Associates, 2002.

[15] SC2001 Bandwidth Challenge proposal: Bandwidth to the World: http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2001/

[16] L. Wall, T. Christiansen and R. l. Schwartz, "Programming Perl", O'Reilly & Associates.

[17] Bbftp: http://doc.in2p3.fr/bbftp/

[18] Pipechar: http://www-didc.lbl.gov/pipechar/

[19] See the Solaris man mount ufs command for more details..

[20] Tcpdump: http://www.tcpdump.org/.

[21] UDPmon: www.hep.man.ac.uk/~rich/net

[22] Example of the code performance analysis: http://www.slac.stanford.edu/comp/net/bandwidth-tests/html/codereports/2002_04/2002_04_25.html

[23] Time series web page: http://www.slac.stanford.edu/comp/net/bandwidth-tests/html/slac_wan_bw_tests.html

[24] Matt Mathis, private communication.

[25] Tom Dunigan, private communication.

[26] "The Web100 Project, facilitating Effective and transparent network Use", http://www.web100.org/.

[27] "TCP Tuning Guide for Distributed Application on Wide Area Networks", http://www-didc.lbl.gov/tcp-wan.html

[28] Qbone Scavenger Service: http://qbone.internet2.edu/qbss/

[29] Cisco IOS Netflow, http://www.cisco.com/warp/public/732/Tech/netflow/

[30] C. Logg and R. L. Cottrell, "Passive Performance Monitoring and Traffic Characteristics on the SLAC Internet Border", Proceedings of Computing in High Energy Physics 2001 (CHEP01), Science Press, Beijing, New York.

[31] C. Dovrolis, P. Ramanathan, D. Moore, "What do Packet Dispersion Techniques measure?", Proceedings of the 2001 Infocom, Anchorage AK. April 2001.

[32] M. Jain and C. Dovrolis, "Pathload: a measurement tool for end-to-end available bandwidth", PAM 2002, Passive and Active Measurement Workshop,pp 14-25, Fort Collins Colorado March 2002.

[33] "GridFTP: Universal Data Transfer for the Grid", White paper. http://www.globus.org/datagrid/

[34] "INCITE: Edge-based Traffic Processing and Service Inference for High-Performance Networks", http://www-ece.rice.edu/INCITE/