Experiences and Results from a New High Performance Network and Application Monitoring Toolkit

R. Les. Cottrell, Connie Logg

Stanford University / Stanford Linear Accelerator Center (SLAC),
2575 Sand Hill Road, Menlo Park, California 94025

***Abstract*—Grid Computing capabilities are increasingly needed for scientific research. Groups such as Globus and 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 a toolkit for studying the available bandwidth as measured by various network sensing tools and comparing that with the bandwidth achievable by various bulk data transfer applications. This study has provided experience in the challenges of deploying and using the sensor tools and transfer applications, as well as information for fine tuning the applications and analyzing their performance. The results presented in this paper include the deployment challenges, techniques for optimizing the duration of measurements, the impacts of throughput on CPU utilization, optimizing windows and parallel streams, the impact on other users, comparisons of various throughput measurement techniques, patterns of throughput behaviors, forecasting, and comparisons of active and passive measurements. We finish up with possible avenues for future development.**

***Index Terms*—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. These institutes 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 high throughput.

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 passive and active end-to-end application and network performance measurements and predictions. The measurements and results are targeted at high performance network links, such as those used worldwide by Grid applications and other academic and research (A&R) applications. Typically these are 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 toolkit and results are expected to be valuable for:

- Providing planning information to applications, grid and network planners by:
  - Providing an understanding of the achievable performance in today's network and application (file copy & ftp) throughput.
  - Providing historical information on growth and changes in performance.
  - Providing predictions of throughput to applications so they can make decisions on how and where to send and receive data.
- Providing troubleshooting information to network administrators and users by:
  - Indicating when there are incremental or sudden changes, the magnitude of the changes, and providing alerts.
  - By comparing achievable throughput with known component performances and the performance of other paths with common links, thereby helping to pin-point whether a performance issue is in the host, network, firewall, application, or at some sub-component such as a disk.
- Providing network and applications developers with a better understanding of how networks and applications work together by:
  - Providing validation/correlation of how network performance relates to metrics such as delays and loss performance (e.g. bandwidth estimators).

Manuscript submitted November 30, 2002. This work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division under the U.S. Department of Energy. The SLAC work is under Contract No. DE-AC03-76SF00515..

Les Cottrell and Connie Logg are with the Stanford Linear Accelerator Center, 2575 Sand Hill Road, Menlo Park, CA 94025 (emails: cottrell@slac.stanford.edu, cal@slac.stanford.edu).

  - Assisting users in selecting the optimum network parameters (e.g. windows, streams), host and application (e.g. compression) configuration options.
  - Providing a public domain network performance database, together with analyses, and web-accessible reports and raw data. This data and information can be used for further research, for predictions and for application steering.
  - Providing information on the challenges of establishing and maintaining the secure transfer of large amounts of data over long periods of time
- Providing a base on which to test, compare and validate TCP stacks, configurations, 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]. Several projects provide public (without subscription or some form of membership requirement) access to the data and reports. The AMP [6], PingER, and skitter/skping [7] projects perform ping and traceroute measurements but no bandwidth estimation or throughput measurements. Surveyor [8] and RIPE [9] make one way delay, loss, inter-packet Delay Variability (IPDV), and traceroute measurements. RIPE also includes bandwidth and routing information but the results are only available by subscription. NIMI [10] is an infrastructure for making on demand measurements and does not have continuous measurements and reports. The European SCAMPI project [11] is developing a scaleable monitoring platform for the Internet, but there do not appear to be any Internet monitoring results published on a regular basis yet. The Network Weather Service (NWS) [12] makes round trip measurements and bandwidth estimates (single stream only). The NWS also has sophisticated prediction mechanisms. Unlike the infrastructure being described here, the NWS currently does not provide file copy/transfer application measurements. The Work Package 7 of the European Data Grid [13] have developed an infrastructure for making ping (using PingER), TCP throughput and UDP measurements between seven European sites; but currently they make no file copy/transfer measurements.

The current toolkit/infrastructure differs from most others since it makes measurements of applications as well as network performance. Other differences include:

- It does not require dedicated hosts for the target hosts and can therefore run close to the real applications of interest.
- It is cheap, simple and quick to extend.
- It makes the measurements in a hierarchical fashion, as opposed to full mesh measurements. Thus it mimics the organization of many collaborations.
- It runs under the same operating systems (Linux and Solaris) used by most Grid applications of interest.

The current work is an outgrowth of the exploratory work [14] reported at PAM 2002. We have redesigned the infrastructure to provide for the addition of new probes (network sensors and applications) for making measurements, and to allow for the use of the probes in other scheduling environments. The current SLAC scheduling is provided by the UNIX cron facility, and measurements are made on a regularly scheduled basis with the reports being generated after each run.

In the rest of this paper, we first describe the measurement methodology. We then describe results from the deployment. These include: deployment challenges, optimizing the measurement durations; the impact of high throughput on CPU utilization; comparisons of file transfer/copy application and a packet pair dispersion bandwidth estimation tool with iperf TCP throughputs; the impacts of maximum window size and number of parallel streams selection; the impacts of high throughput on other users; forecasting; and a comparison of active and passive measurements. We conclude with a summary of the most significant results so far, and finish up with a discussion of possible future directions.

II. Methodology & Deployment

A. *Methodology*

The methodology is described here in sufficient detail to enable an understanding of the results.

There are 2 types of hosts, monitoring and target hosts. The "monitoring" hosts run the measurement tools (probes), log the data from their runs, extract, analyze, and report on the information via the web. The "target" hosts receive the probes from the monitoring hosts and respond to them. The logs and data, although they are currently collected on each monitoring host, could be collected in a network file system. In that case the monitoring host function could be split into 2 or 3 separate hosts. One host would make the measurements, another would perform the analysis, and a third could be the web server.

Each monitoring site works with its collaborators to decide on the target hosts to probe. Typically multiple target hosts are monitored by a monitoring site. For each target host an account must be provided on it that is accessible, via the secure shell [15] ("ssh"), from the monitoring host. After installing the appropriate public key in the account on the remote host, the target host account is remotely configured and the target host toolkit is downloaded from the monitoring host. Information on the target hosts is kept in a target host configuration database which is accessible to the monitoring host.

The monitoring host schedules the measurement runs. Currently they are at regular intervals driven by a Unix `cron` table entry. The scheduling interval for each monitoring host is determined by the monitoring host administrator. The actual interval chosen depends on the load acceptable on the monitoring host's link, and the amount of time it takes to make a set of measurements to all target hosts. Typically at SLAC, the interval is about 90 minutes for 40 target hosts. At this time, for each set of measurements, the monitoring host selects each target host in turn and runs ping for 10 seconds, does a traceroute (with one probe per hop) followed by running the iperf [16] TCP transfer tool, secure file copy using the peer-to-

peer tool bbcp [17] with both memory to memory (bbcpmem reads from /dev/zero and writes to /dev/null) and disk to disk (bbcpdisk) copies, followed by the bbftp [18] file transfer program. Both bbcp and bbftp allow the selection of large window sizes and multiple parallel streams of data and provide measurements of the throughputs achieved. At one time pipechar [19] measurements were also performed, but they have been discontinued due to the inaccuracy of the results above 155Mbits/s and the time they take. The lists of probes done for any given target host can be specifically defined in the target host configuration file. The output from each probe is captured, identified with a token, time stamped and written to a "log" file. For each remote host there is one log file per probe type per day.

To provide robustness, servers are remotely started and killed for each measurement. Also each probe command (e.g. iperf) is started as a separate task, so it can be timed out and killed in case of problems. Some sanity checks are also done, e.g. if ping is expected to work (as defined in the remote host configuration database), but it fails, the other measurements are not attempted.

Following each measurement, the results are extracted and converted into space-separated tables that are made available via the web. The format of the extracted tables is documented in the first line of each file to enable others to access the data. The analysis is performed on the extracted data and it produces web accessible pages containing time series (short term for the last 28 days, and longer term aggregated) plots, histograms, scatter plots, statistical and analyzed tables (accessible over the web in formats suitable for loading into programs such as Excel), information on the success of the test, and narrative.

B. Current Deployment

There are currently 10 monitoring hosts running the IEPM-BW toolkit. They are at: APAN in Japan, the Stanford Linear Accelerator Center (SLAC) near San Francisco CA., FNAL near Chicago IL, Georgia Tech, INFN at Milan Italy, Internet 2 (in Michigan), Manchester University in England, NIKHEF in Amsterdam the Netherlands, University College London England, and the University of Michigan.

The results in this paper are from the SLAC monitoring host. The target host sites for the SLAC monitoring host were chosen from PPDG, HENP and major network monitoring collaborator sites. These sites 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), Thomas Jefferson National Laboratory (JLab) in Newport News VA, Los Alamos National Laboratory (LANL) in Los Alamos NM, Lawrence Berkeley National Laboratory (LBNL) in Berkeley CA, National Energy Research Scientific Computing Center (NERSC) in Oakland CA, Oak Ridge National Laboratory (ORNL) Oak Ridge TN, NASA/GSFC, San Diego Supercomputing Center (SDSC) in San Diego CA, Rice University in Houston TX, Stanford University in Palo Alto CA, Indiana University (IU), University of Florida (UFL) in Gainesville FL, University of Illinois at Urbana Champaign (UIUC), the University of Michigan (UMich) in Ann Arbor MI, University of Wisconsin (UWisc) in Madison WI, Starlight in Chicago, CERN in Geneva Switzerland, CESnet in Prague Czech republic, KEK in Tokyo Japan, Rutherford Laboratory near Oxford England and Daresbury Laboratory near Liverpool England, IN2P3 in Lyon, France, 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 SLAC. There are currently (January 2003) 40 active destination hosts at about 30 sites in 9 countries.

Fig. 1 shows the logical routes between SLAC and its remote site participants in December 2002. The boxes with bold outlines are monitoring sites in their own right. The labels in italics in the boxes indicate the host has a 100Mbit/s connection. Other hosts have Gbits/s connections. The box shading indicates the participant type. Diagonal lines are for PPDG/GriPhyN/HENP collaborators, hashed shading indicates the site is a network measurement collaborator, and the un-shaded boxes are for European Data Grid collaborators. The clouds are for Internet Service providers (ISPs). The grey lettering in the clouds indicates the "GigaPoP" (e.g. ATL means Atlanta). The numbers by the sites indicate the average measured throughput from August 24 through October 26, 2002. For the measurements reported SLAC had 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).

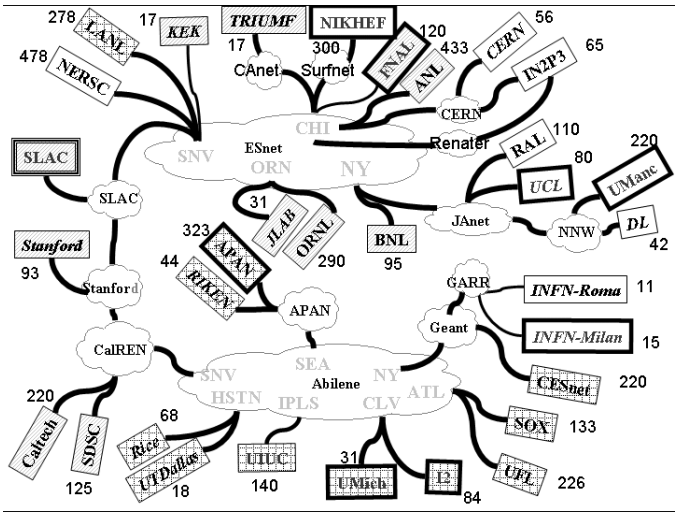

Figure 1: Routes and iperf TCP Mbits/s from SLAC to the remote sites.

III. RESULTS

A. Deployment Challenges

The deployment challenges can be classified in 3 categories: network, "security", and operational challenges

1. Networks are dynamic entities. Some applications require the reverse lookup of host names which can suddenly start to fail. Routes change over the course of time, sometimes failing and sometimes reconfiguring in a manner that provides for significantly lower throughput. The installation of rate limiting can result in greatly lowered throughput. These in turn can cause tests that run in a defined period of time (see operational challenges) to suddenly start timing out and/or failing.
2. Security mechanisms present another challenge. Ports suddenly become blocked. Target nodes are occasionally upgraded, rebuilt and/or reconfigured resulting in the loss of ssh keys, a change in the speed of the network interface used, and/or a change in the allowable buffer and window sizes. Note that the use of ssh does not scale well. Every target node must be individually set up to allow for communication from any monitoring host which is going to probe it.
3. Operational challenges result from instances of network and security challenges and under provisioning of the target hosts. Each and every probe must have a time out mechanism. Not all applications terminate gracefully when there is a network related problem. Sometimes the processes just hang around on the target and/or monitoring hosts, filling up process space, filling the network with data that is not being delivered, or chewing CPU. In cases where the ssh keys are no longer valid, the target host will sit for a long period of time before timing out on the password prompt. Tests can fail because the target host does not have enough disk space to store the data being transferred. Handling these challenges involves writing code that terminates the processes that are hanging around after a test, and removing any target test data files. Note that this code itself must be timedout out!

B. Measurement Duration

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 320s, and window sizes of 256, 512, 1024, 2048 and 4096kbytes. For each of the above possible pairs we made a single stream measurement of the iperf TCP throughput from SLAC to the target host. We repeated this multiple times (17-20) to estimate the magnitude of the variation. 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 reach a stable throughput rate. Fig. 2 shows the iperf median TCP throughput measured from SLAC to Caltech (40 ms. Round Trip Time (RTT)) for various window sizes. The points are the medians of each set of measurements, and the error bars are determined from the Inter Quartile Ranges (IQRs). It is seen that, in some cases, 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. Similar results are found for other paths such as SLAC to IN2P3 (RTT 177 ms and maximum throughputs of over 300 Mbits/s). Since we are interested in the performance for long duration transfers, we took the minimum duration that was representative of a long duration transfer. So for most of our measurements we settled on a duration of 10 seconds.

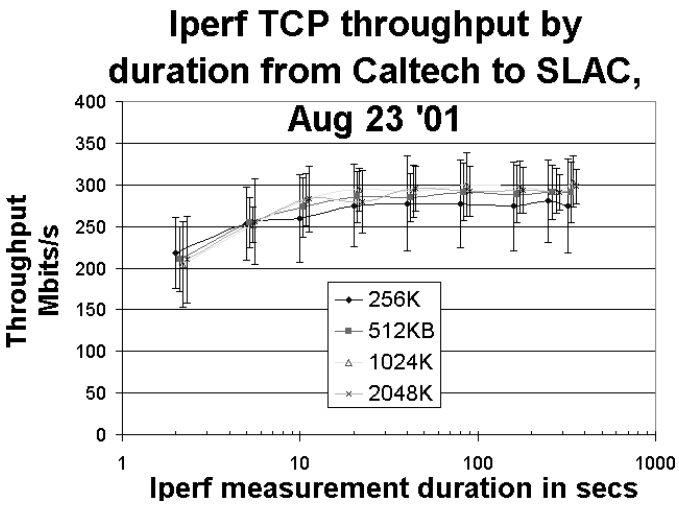

Figure 2: Iperf TCP throughput by measurement duration from SLAC to Caltech, Aug 23 2001

As one moves to larger RTT bandwidth products, slow start takes longer (e.g. for a single stream from about 1s for a 100ms RTT 100 Mbits/s link to about 5s for a 200ms RTT and 1Gbit/s link). Thus more time (to get 90% of the throughput outside slow start one needs about 10 * the slow start time) will be needed for the throughput to reach a stable value and for the data transferred during slow start to be a small fraction (say < 10%) of the total data transferred. Such a long probe of 50 or more seconds would not be net friendly. We are therefore investigating using Web100 [20] to look at how many bytes have been transferred over the last second, once the initial TCP slow start is over, and use this as an estimate of the stable throughput. We will report on the effectiveness of this in a future paper.

C. Impact on CPU Utilization

Fig. 3 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.

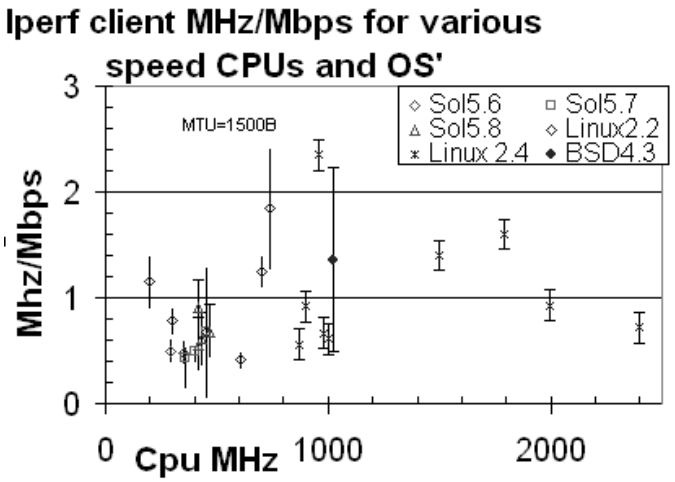

Figure 3: Ratio of measurement host MHz utilization to Mbits/s transferred

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 24 hosts = 0.89+-0.48 (13 Linux hosts = 1.05+-0..58, 11 Solaris hosts = 0.68+-0.27). The information on the MHz necessary to support high throughputs is important to enable selection of the monitoring host hardware.

D. Windows & 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 [21]. 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. 4 for a typical example in this case from SLAC to ANL.

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. 4, 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 consumes resources (memory, a process, and CPU cycles).

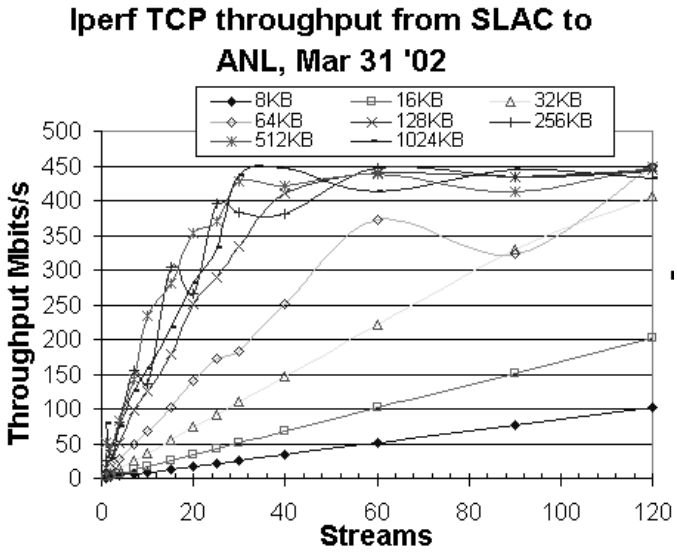

Figure 4: Ten second iperf TCP throughputs from SLAC to ANL

E. Impact on Others

To investigate the impact of high bulk throughput measurements 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 2002, 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 120Mbits/s, 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. 5. The triangles indicate the RTT with no iperf load, and the squares indicate the RTT 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.

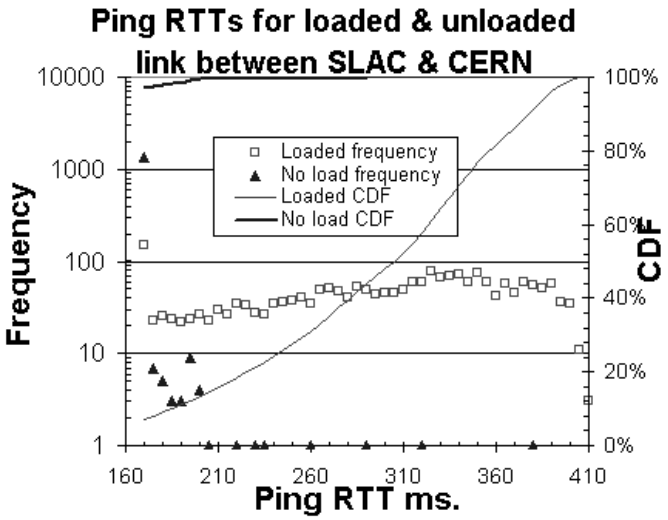

Figure 5: Ping RTTs with and without simultaneous iperf load. The triangles indicate the RTT with no iperf load, and the squares indicate the RTT with an iperf load. The bottom axis is the ping RTT. The lines represent the CDFs.

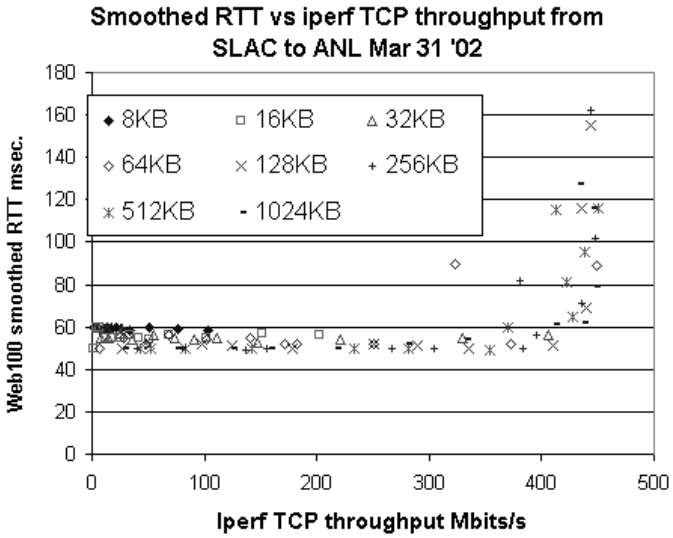

Figure 6: Web100 smoothed RTT vs. iperf TCP throughput from SLAC to ANL, Mar 31 2002.

It is seen that the unloaded RTT is sharply clustered between 166 and 170 ms (the CDF indicates that over 95% of the measured RTTs are in this range), while the loaded RTT distribution is fairly flat for over 150msec above the minimum RTT. We are looking for ways to alleviate this effect. Some possibilities include using the QBone Scavenger Service (QBSS) [22], self rate limiting the application (i.e. enable the application to restrict its throughput), providing a feedback loop for the application by using Web100 to measure the RTT and/or retransmissions and using these values to adjust the application's offered throughput.

Another way of looking at the impact is to look at the Web100 TCP information such as the smoothed RTT, re-transmissions or congestion events to understand the effect of the high throughputs. The points in Fig. 6 are the smoothed RTTs measured between SLAC and ANL for the iperf TCP throughputs shown in Fig. 4. It can be seen that there is little effect on the ping RTT until the throughput exceeds over 300 Mbits/s. Above 330Mbits/s (~73% of the maximum throughput observed) the smoothed RTT can increase dramatically by over 250%. Further work is in progress to understand how this information may be used to steer applications.

F. Comparing Throughputs from Measurement Probes

We compared the iperf throughput with the minimum available predicted by pipechar. An example is shown in Fig. 7. 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. Given these difficulties for high speed paths, and the time taken for pipechar to complete a measurement, we currently do not run pipechar as part of the standard suite of sensors.

At higher bandwidths (> 100Mbits/s), the packet dispersion method requires increased accuracy (better than tens of microseconds) of the measurement clock. Packet dispersion techniques using host timings will probably also suffer badly if the network interface card (NIC) coalesces interrupts inbound or does buffering and fragmentation outbound. We also looked at using other variable packet size techniques such as pathchar [23], pchar [24], and pathrate [25] but they all took too long (minutes to hours) to make an estimate. Investigations with early versions of pathload [26} also indicated that it gave poor agreement with iperf TCP for rates above 150 Mbits/s.

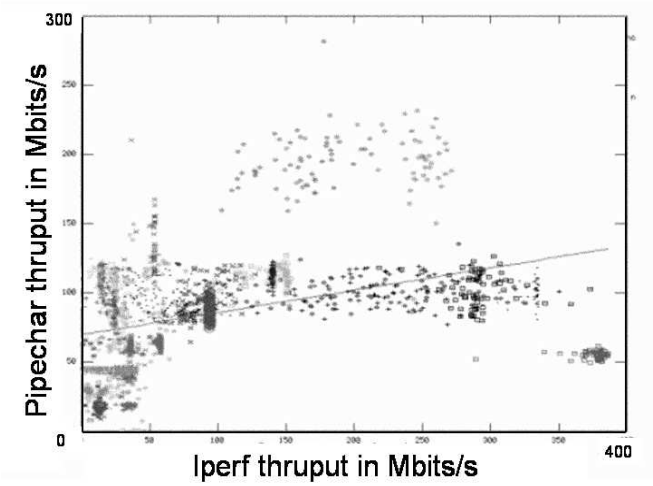

Figure 7: Pipechar estimates vs. iperf TCP throughput

To determine the relative performance of a file copy application without having to account for effects such as disk performance, file system, caching etc., 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 October 11, 2002, between SLAC and about 30 remote hosts, is shown in Fig. 8.

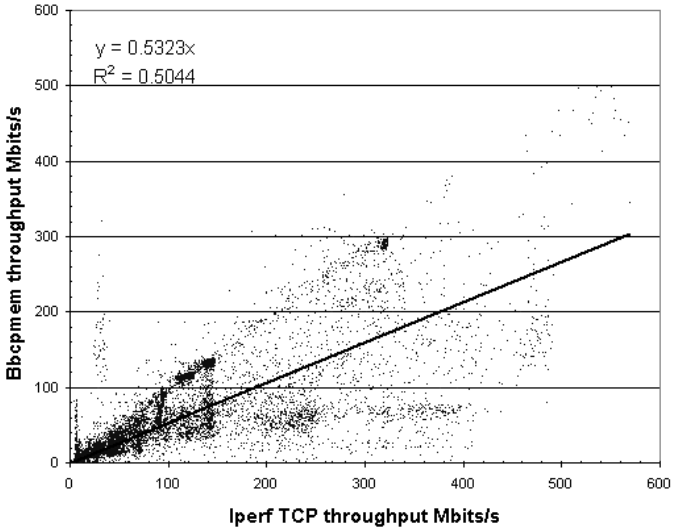

Figure 8: Bbcp memory to memory vs. iperf TCP throughput.

It is seen that the correlation is very variable (the square of the correlation coefficient is R^2 ~ 0.5). In some cases bbcpmem performs as well or even better than iperf, however in general it does not perform as well. The line shows a linear regression fit with the parameters *y=0.53x*. 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 (e.g. due to congestion or packet loss) will cause others to slow down, whereas for iperf the streams are asynchronous. If the cause of the losses is not due to congestion and thus does not affect all streams, then the bbcp synchronization strategy will be disadvantageous. 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. Looking at the iperf frequency histogram (see Fig. 9), we also observe vertical lines just under 45Mbps, 100Mbps and 150Mbps where the constraint is probably network capacity related (i.e. T3, Fast Ethernet and OC3).

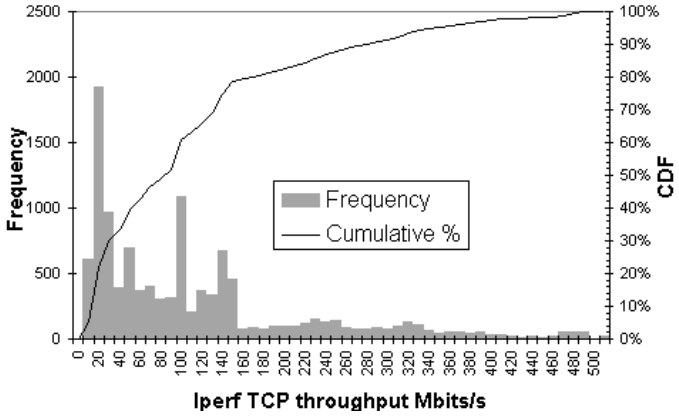

Figure 9: Iperf TCP throughput frequency histogram.

Disk to disk performance of bbftp and bbcpdisk is still under investigation [27]. The performance depends critically on caching, the file-system (e.g. local disk vs. NFS), when the file is committed, and the file size. Ideally we wish to measure the performance for a large file (Gbytes), since this is closer to the large data replication HENP applications we have in mind. However, transferring such files can take considerable time, can be very intrusive on the network, and disk space may not be available at the remote host to save the file. Reference [27] indicates that one can utilize a relatively small file (64Mbytes), committing the portion of the file remaining in the disk cache to the disk at the end of the copy, to obtain similar results for a much larger file (2Gbytes). We are therefore modifying the toolkit to use the "commit at end" strategy, and will report on the results at a later time.

G. 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 few measurements. Five was selected as a reasonable compromise between enabling a reasonable calculation of the variation, reasonable smoothing over the last few hours, and the need to reasonably closely track the most recent results. An example comparing the actual vs. forecasted values for the SLAC to Caltech path is seen in Fig. 10. 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 for measurements, averaged over the previous 5 observations, for 31 remote hosts for measured between June 23 and July 4, 2002.

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

33 hosts	iperf TCP	bbcp mem	bbcp disk	bbftp	pipechar
error	10%	17%	15%	16%	3%
Stdev	8%	15%	13%	12%	3%

It can be seen that, even with this simple forecasting method, reasonable agreement is achieved (better than 17% in most cases) for 90 minutes after the last measurement. We also tried using Exponentially Weighted Moving Averages (EWMA), i.e. the current average avg_i is given by:

$avg_i = (1 - w) * y_i + w * avg_{i-1}$

We found that for the data in Table 1, using $w = 0.7$, the average errors differed by less than 2%, or well within the standard deviations. Fig. 10 also shows the EWMA predictions.

H. Patterns of Throughput Behavior

The achievable iperf TCP throughputs (see the numbers in Fig. 1) varied by more than a factor of 10 from site to site. By design, hosts with 1000GE NICs had higher speed connections (typically 622Mbits/s) to the Internet and, as expected, higher performance was observed. By using large windows and multiple streams we were able to measure throughputs of several hundreds of Mbits/s across both transcontinental and transoceanic links.

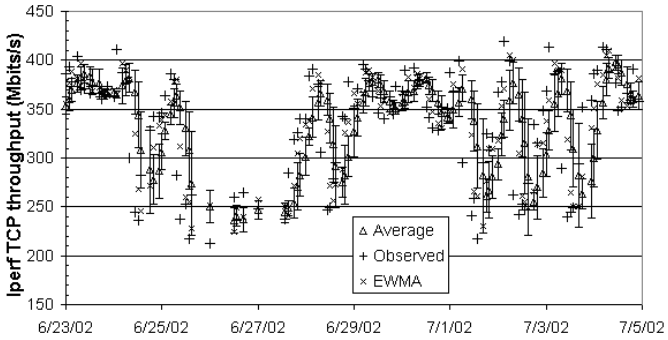

Figure 10: Forecasting iperf TCP throughputs: observed (+), moving average of last 5 observed point (triangles with error bars) and EWMA predictions (x).

Viewing our time series plots of the throughputs, we observe two major types of behavior that may overlap at times.

1. Sudden step changes in throughput, as can be seen in Fig. 11 around September 19. These are usually associated with a network change, e.g. a new route, or a link upgrade. They may also be associated with a remote host change, e.g. a new CPU or a change in the Network Interface Card (NIC) used.
2. Oscillations in the throughput on a daily basis, e.g. high throughput at night or weekends when there is lower utilization and congestion, and higher performance at other periods. We refer to the daily changes as diurnal variations.

If the time series are fairly flat (e.g. there are only small diurnal changes) then sudden changes in throughput show up as multimodal peaks in histograms of the throughput. They also show up in the moving averages with large relative standard deviations for the set of points close to the change.

About 25% of the probes to target hosts exhibit large diurnal variations (such variations can, for example, be observed in Fig. 12). For such hosts we use a simple fit to:

$f(x) = abs(a) * sin(x + b) + c$

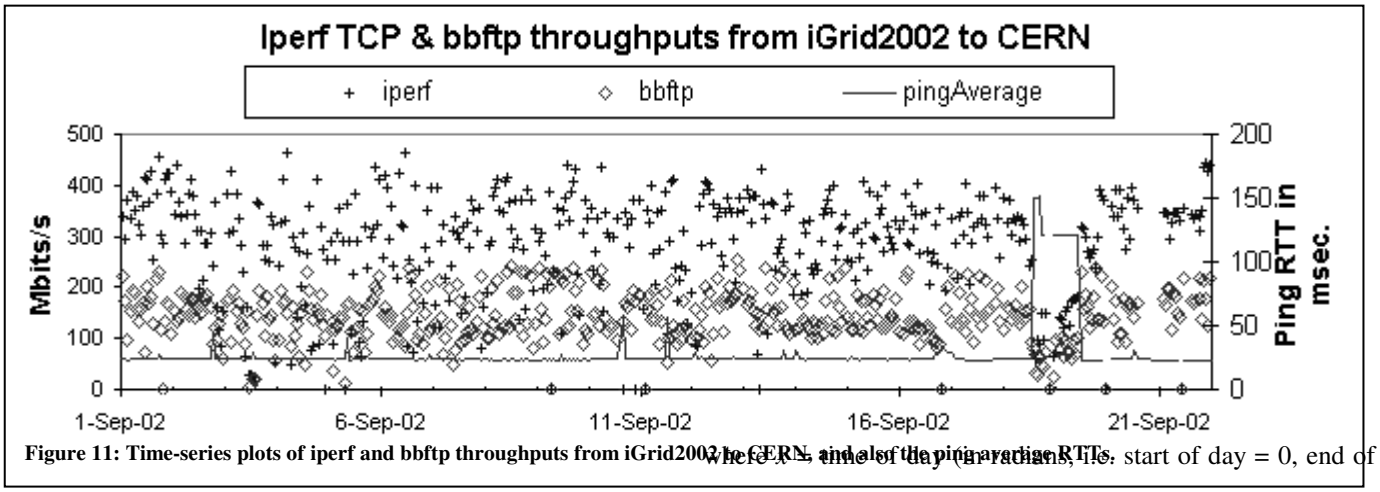

Figure 11: Time-series plots of iperf and bbftp throughputs from iGrid2002 to CERN, and also of the ping average RTTs.

where x = time of day (in radians, i.e. start of day = 0, end of

day = 2 * *pi*). We use as the least-squares fit starting values, *c* = average throughput, *a* = standard deviation of throughput, and *b* = *pi*/2. This fit enables an easy characterization of the diurnal variability. The fitting can be further simplified, by noting that *b* (the phase angle) should stay fairly constant for a given site (if there is a diurnal variation then the busy/congested periods are likely to be the same from weekday to weekday). Fig 12 shows a least squares fit to iperf TCP data measured from SLAC to Caltech from October 11 to November 8, 2002, where the x axis is the time of day of the measurement, and the weekday measurements have been separated from the weekend measurements. Also shown are the curves (dashed lines) from simply using the starting values for *a* and *c*, and leaving *b* at the value found in the fit. It can be seen that we can do almost as good with the simple fit, and not have to resort to least square fitting techniques. The difference in the fitted value and initial estimate can be expressed as *diff = abs((fit-initial)/fit)* and yields median values (for 39 remote hosts) of < 2% for *a* and 6% for *b*. Since *a* and c are simple to estimate from the data, and *b* should stay roughly constant for a given site, this proves to be a simple method for quantifying the diurnal nature of the data.

We have found that we can roughly quantify the "diurnalness" of the data for a given node by looking at the error on the fit parameter *b* (δb). In essence, δb determines how the diurnal nature of the data is clearly defined to allow *b* to be well determined. Values of δb of < 0.2 appear to indicate candidates for paths with large diurnal variations. We identify these large diurnal variations by eyeball by looking at a plot such as shown in Fig. 12. For Fig. 12 the values of δb are 0.04 (weekday) and 0.1 (weekend).

As expected, there is usually different and less diurnal variation for weekend data. We are looking at ways to fold the diurnal variations into the predictions, for example by predicting the value at some time, from the value at the same time a week ago. Though this may be less accurate than a prediction from more recent data, it may be of value if there is no recent data.

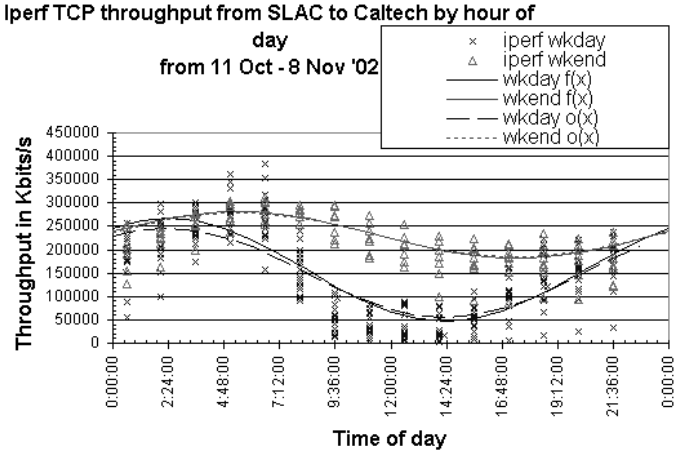

Figure 12: Iperf TCP diurnal variations

I. Passive and Active Measurements

To validate whether the sensors were reporting the correct throughputs, we read the Netflow records [28] 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 Gbits/s links, one to ESnet, the other to Stanford University and thence to CalREN. The methodology of collecting the Netflow records is described in [29]. 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. Flows with the same source and destination address that start within a few (currently we use 5 seconds, but are experimenting with better ways to associate the flows) seconds of one another are assumed to belong to a given application process. In some cases we could use the port number to further refine this selection. 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 (probe) by means of time series, scatter plots, calculating *err = (passive-active)/passive* and the correlation coefficient *R*. An example of a time series is shown in Fig. 13. Fig. 13 shows the time series of active and passive throughput measurements for iperf from SLAC to Caltech for 28 days starting April 1, 2002. For this case the *err* = 2% and *R*=0.99 and the agreement is seen to be excellent.

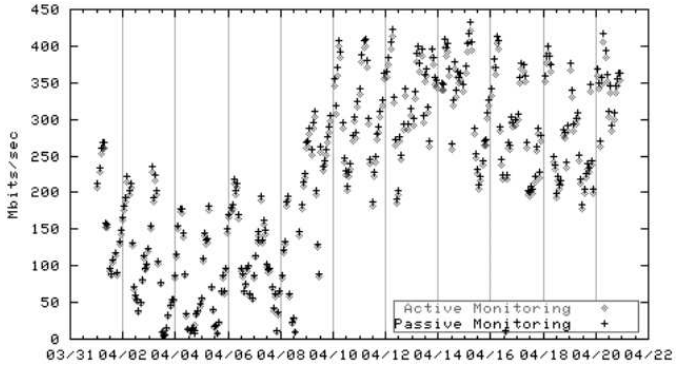

Figure 13: Example of time series of active and passive throughputs from SLAC to Caltech, Mar-Apr 2002

The overall agreements, for the 28 days starting April 1, 2002, are shown in Table 2 below. The ranges are the 25 percentile and 75 percentiles. On average the throughput for each probe 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 it starts 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 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 probes to target hosts with lower throughputs, and the disagreement for low throughput usually coincides with a negative *err*.

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 2002

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	

More work will be required to understand why bbftp has poorer agreement between its active and passive measurements, and bbcp, and some of the details of the other disagreements. Initial investigations point to implementation differences in the handling of parallel streams, calculating the throughput, and buffer allocation. 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.

IV. CONCLUSIONS

Preliminary results from IEPM-BW so far indicate:

- Using a hierarchical infrastructure, where each monitoring host selects the target hosts to probe (as opposed to a full mesh measurement infrastructure), lends itself very well to the requirements of HENP where there a few major sites providing access to large amounts of data, and each major site often collaborates with a different set of remote sites.
- Using standard operating systems (Linux and Solaris) for the monitoring and remote hosts enabled us to easily take advantage of new sensors and applications that in some case have not been ported to other operating systems.
- We have found the ssh infrastructure, that enables automatically installing software and dynamically start/kill servers at remote sites, to be valuable for making one time measurements, in particular for validating new measurement tools. However scaling it to a large number of nodes in a grid layout may not be practical.
- Not having a dedicated centrally managed standard monitoring host at each site has drawbacks in terms of having to support multiple configurations. This has required the development of remote installation tools and an extensive database to parameterize the remote host. The advantage, however, is that the procurement, installation, administration, control, security etc. of the remote host is left to the remote site. This in turn enables us to add a new remote host in a matter of hours from being given the account and password. Occasionally this leads to incompatibilities with IEPM-BW; however, in almost all cases this has not been a problem so far.
- 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. However, as the bandwidth RTT product continues to increase, either longer measurements will be needed or new methods need to be developed.
- Roughly speaking, about 1 MHz of CPU cycles provide 1 Mbits/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. Un-cached 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 less than 20% error.
- Passive Netflow measurements agree to within 5% with active measurements for most target hosts.
- The toolkit has also been effectively used for high throughput demonstrations [30]. Currently the aggregate (i.e. the throughput if all the measurements were made simultaneously) iperf throughput from SLAC to its remote hosts is about 4.5Gbits/s.

We plan to port the monitoring host toolkit to more sites. Initially, to preserve flexibility, each monitoring site is saving its own data, and performs its own extraction/analysis and reporting. We are working on making the data available via more standard publish/subscribe methods. As we increase the number of monitoring sites we will also need to pursue ways to provide probe timing control [31] to ensure the measurements do not collide with one another.

We are working on evaluating other probes (sensors and applications) including pathrate, pathload, GridFTP [32], INCITE [33], and UDPmon [34], and hope to select a new recommended set of base measurement sensors. As part of this we will simplify the way in which new probes are added and their data analyzed and added to the reports. We also intend to replicate the measurements from a second host at SLAC using various experimental TCP stacks [35], [36], [37]. This will enable us to compare the performance of the stacks on a wide variety of paths.

We 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 are integrating Web100 into the measurements that, 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, and providing tools to assist in making applications such as bbcp network aware.

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, and I-Heng Mei for help in the analysis of the correlations between the Web100, and passive and active measurements. 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 whom 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, Tom Dunigan of ORNL, 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[1]

[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] "RIPE NCC Test Traffic Measurements", http://www.ripe.net/ttm/

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

[11] "SCAMPI Overview", http://www.ist-scampi.org/overview.html

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

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

[14] "Passive and Active Monitoring on a High Performance Research Network". W. Matthews, R. L. Cottrell, D. Salomoni. SLAC-PUB-8776, Feb 2001. 6pp. Passive and Active Monitoring (PAM) 2001, Amsterdam, April 22 - 24.

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

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

[17] 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

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

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

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

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

[22] "Qbone Scavenger Service", http://qbone.internet2.edu/qbss/

[23] V. Jaconson, "Pathchar", ftp://ftp.ee.lbl.gov/pathchar/

[24] B. A. Mah, "Pchar", http://www.employees.org/~bmah/Software/pchar/

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

[26] 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.

[27] A. Tirumala, R. L. Cottrell. C. Logg, "Disk Throughputs", http://www-iepm.slac.stanford.edu/bw/disk_res.html

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

[29] 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.

[30] "SC2001 Bandwidth Challenge Proposal: Bandwidth to the World", http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2001/; "iGrid2002: Bandwidth from the Low-lands", http://www-iepm.slac.stanford.edu/monitoring/bulk/igrid2002/; "SC2002: Bandwidth to the World", http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2002/

[31] B. Gaidioz, R. Wolski and B. Tourancheau, "Synchronizing Network Probes to avoid Measurement Intrusivness in the Network Weather Service", http://www.cs.ucsb.edu/~rich/publications/nws-period.pdf

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

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

[34] Richard Hughes-Jones, "Some Tools used for Testing Network Behavior:" http://ww.hep.man.ac.uk/~rich/net

[35] S. Floyd, "HighSpeed TCP for Large Congestion Windows", Internet draft draft-floyd-tcp-highspeed-01.txt, work in progress, 2002. http://www.icir.org/floyd/hstcp.html

[36] S. Low, "Duality model of TCP/AQM + Stabilized Vegas", http://netlab.caltech.edu/FAST/meetings/2002july/fast020702.ppt

[37] “The Net100 Project”, http://www.net100.org/