Net100: Developing network-aware operating systems

Final Project Status Report
Funding: MICS (Thomas Ndousse), $2.6M over 3 years starting 9/2001
Research Team:
Lawrence Berkeley Lab: Brian Tierney (PI), Martin Stouffer, Jason Lee
Oak Ridge National Lab: Tom Dunigan (PI), Nagi Rao, Florence Fowler, Steven Carter
Pittsburgh Supercomputing Center: Matt Mathis (PI), Raghu Reddy, John Heffner, Janet Brown
National Center for Atmospheric Research: Peter O'Neil, Marla Meehl

Many high performance distributed applications require high network throughput but are able to achieve only a small fraction of the available bandwidth. The Net100 project is developing a network-aware operating system and a tuning daemon that uses path characteristics to transparently tune TCP flows over designated paths. Path characteristics are collected by a set of network probes and sensors located around the world. No modifications are required to the application, and the user does not need to understand network or TCP characteristics. On operating systems with Net100 tuning enabled, network applications have exhibited speedups of more than an order of magnitude.

1 Introduction

The Net100 project was motivated by complaints of computational scientists and researchers at DOE national labs that network applications were unable to transfer data across the country at any where near the available link bandwidths. Many high-performance distributed computing applications transfer large volumes of data over wide area networks and require data rates on the order of gigabits per second. Even though Internet backbone speeds have increased considerably in the last few years due to projects like Internet 2 and NGI, distributed applications rarely take full advantage of these new high-capacity networks. In fact, recent data for Internet 2 (April, 2002) show that 90% of the bulk TCP flows (defined as transfers of at least 10MB of data) use less than 5 Mbits/sec, and that 99% use less than 20 Mbits/sec out of the possible 622 Mbits/sec.

The objective of the Net100 projects is to improve the network performance of scientific applications without requiring the intervention of a "network wizard." The goal is to have the operating system dynamically tune network flows, so the application and the scientist would not have to be network-aware. At the time of the proposal, NSF's Web100 project seemed to offer a path toward developing a network-aware operating system.

The initial focus of the Net100 project has been on accelerating TCP over high speed, long delay networks. TCP is the primary protocol used on the Internet for reliable transfer, but may need some tweaking for some scientific applications. TCP adjusts its rate when it experiences packet loss. The time to recover from a packet loss is proportional the message size (usually 1500 bytes) divided by the square of the round-trip time (RTT). Floyd has noted that on a 10 Gbs path across the country, TCP can take over an hour to recover from a packet loss! Our project seeks to avoid packet loss if possible, and to speed recovery if packet loss does occur. We believe the Net100 project can have a significant impact on TCP performance on today's and tomorrow's networks.

2 Methodology

The method we used to construct a network-aware operating system consists of three major components.

- instrumented UNIX kernel
- TCP protocol accelerants
- network probes and monitors

The Linux kernel was modified to both monitor and tune TCP flows on an individual flow basis. Several TCP modifications were added to the Linux kernel based on our own protocol research and on work by Feng, Floyd, Kelly, FAST, and others. A collection of network probes and sensors using tools developed by the Net100 team and others collects information on network paths on a periodic basis. These path metrics are used by the Net100 tuning daemon to tune TCP flows.

Web100

The Web100 project is an NSF funded collaboration between the Pittsburgh Supercomputing Center (PSC), the National Center for Atmospheric Research (NCAR) and The National Center for Supercomputing Applications (NCSA). The Web100 vision is to enable users running ordinary applications on typical workstations to either saturate a workstation bottleneck or completely fill a network link. For the Net100 project, the vision was extended to make it easy for scientists to effectively use available bandwidth over high speed, long delay networks.

Web100 kernel extensions provide detailed TCP performance statistics through an enhanced standard "Management Information Base" (MIB) for TCP. This MIB uses TCP's ideal vantage point to provide statistics for diagnosing performance problems in both the network and the application. Our tuning daemon uses the Web100 TCP MIB to collect the necessary information from active flows to perform its "work-arounds."

Network Tool Analysis Framework (NTAF)

To provide tuning data for specific network paths, we have developed a framework for running network test tools and storing the results in a database, which we call the Network Tool Analysis Framework (NTAF). The NTAF manages and runs a collection of network probes and measurement tools with all results sent to an archive. This allows us to compare the results of various TCP modifications and various bandwidth estimation techniques over multiple paths, and see how the results vary over time. Recent results are cached and can be queried via a client API. Data collected by the network tools includes Web100 variables.

We initially deployed NTAF servers and tuning daemons on hosts at Oak Ridge National Lab (ORNL), Lawrence Berkeley National Lab (LBL), National Energy Research Scientific Computing center (NERSC), Pittsburgh Supercomputing Center (PSC), and National Center for Atmospheric Research (NCAR). In the second year of the project, other researchers have installed Net100 nodes at UT, CERN, Amsterdam, SLAC, and ANL. Part of our research includes evaluating network measurement tools. We are currently using NTAF to run the following tests: *ping*, *pipechar*, *iperf* (instrumented to collect Web100 information), *GridFTP*, *netest*, and a CPU and memory sensor based on *procinfo*. In our current implementation of the tuning daemon, the NTAF provides TCP socket buffer sizes for a path based on *pipechar* data (round trip time and bottleneck bandwidth).

Protocol research and TCP tuning

The Web100 kernel modifications have been extended as part of this research to permit tuning more than just TCP's send and receive buffers. At present, the Net100 extensions also allow us to tune TCP's slow-start and choose among several algorithms for tuning TCP's additive increase and multiplicative decrease (AIMD) to speed recovery from loss. We can also regulate packet bursts and reordering threshold and disable delayed ACKs.

We have developed a tuning daemon, or work-around daemon (WAD), to apply these new tuning options to designated TCP flows. The daemon has a configuration file that specifies what flows (source, source port, destination, destination port) are eligible for tuning. The configuration entry for a tunable flow also includes some static tuning parameters (mode, max ssthresh, AIMD parameters, reordering threshold, etc).

When a new connection event from the kernel awakens the daemon, the daemon checks the configuration file to see if it is a tunable flow. For tunable flows, the configuration file indicates if the static values in the table are to be used or if dynamic tuning from the NTAF is requested.[1] The tuning requires Net100 kernel modifications, but requires no changes to existing network applications.

In summary, the Net100 implementation provides several novel approaches to improving network performance. Detailed TCP performance on individual flows can be collected by the Net100 kernel and used with tuning data from distributed network probes and sensors to actively tune a flow. The tuning can improve legacy applications with no changes to the application itself. Tuning can be done on path characteristics and/or policy decisions. The result is an adaptable, high performance TCP protocol.

[1] The daemon periodically queries the NTAF for current path characteristics and saves the information for tuning flows.

3 Experimental Results

We have evaluated various TCP optimizations using both *ns* simulation and a Net100 TCP-over-UDP test harness as well as the Net100-modified Linux kernel. We have tested various tuning options using a NISTNet testbed as well as the Internet. Even though evaluating different TCP tuning options is problematic over real networks, one of the project goals was to show real performance improvements in TCP bulk transfers over real high delay, high bandwidth networks. As part of the project we have performed a large amount of testing of various TCP modifications over a variety of paths. The experiments described below were performed across the US and to Europe using hosts with GigE interfaces linked to DOE's ESnet (OC12/OC192) or Internet2 (OC48/OC192). For more detailed results refer the project web pages or papers.

WAD Buffer Tuning

A number of systems have emerged to address the issue of automatically tuning TCP buffers. Linux 2.4 automatically does sender side tuning, but this only works if the receive buffers have been set to a large value. Dynamic Right Sizing automatically does receiver size buffer tuning, but assumes large send buffers (or a Linux 2.4 autotuning sender).

WAD tuning daemons and NTAF clients are running on our test hosts around the country. When a TCP connection is requested for a tunable path, the Web100 notification mechanism will send a "new connection" event to the WAD. The WAD at each endpoint can then set the optimal buffer size based on static information in the configuration file for the path or using a buffer size provided by the NTAF based on recent network measurements.

Figure 1 (a) shows the throughput for a 10 second data transfer between ORNL and PSC (70 ms round trip time, GigE interfaces) over the ESnet and Internet2 network. The figure shows untuned throughput and WAD-tuned throughput. The network-unaware application runs at less than 10 Mbits/sec, but the same application runs at over 135 Mbits/sec when dynamically tuned by the WAD – more than an order of magnitude improvement in throughput. The tuning daemon at the source and destination uses dynamic data from the NTAF database to increase the TCP buffers to nearly 2 MB as the flow starts.

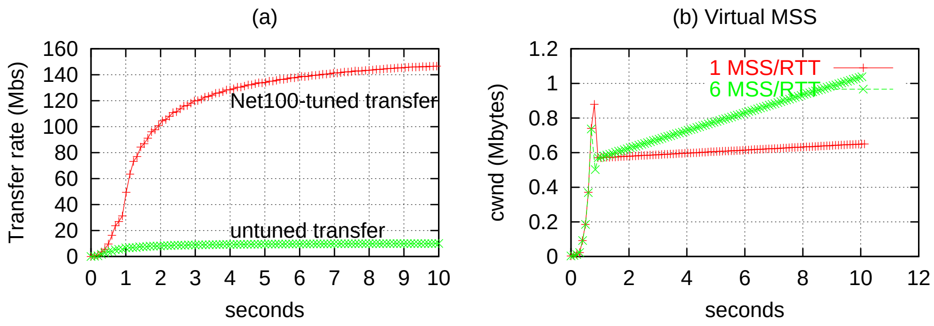

Figure 1: WAD-tuned buffers and virtual MSS

Virtual MSS and AIMD

The behavior of the aggregate congestion window of k parallel streams and the arguments for a larger MTU inspired us to use the WAD to create a "virtual MSS" for designated flows. The virtual MSS is implemented by adding k segments to the congestion window, *cwnd*, each RTT during congestion avoidance. The effect of the virtual MSS is best illustrated when there is packet loss. Figure 1 (b) illustrates two transfers from ORNL to NERSC with packet loss during slow-start. Both flows use the same TCP buffer sizes, but one flow is dynamically tuned by the WAD to use a virtual MSS of 6 segments (e.g., a virtual jumbo frame). The *cwnd* data was collected from the kernel with a Web100 tracer daemon. Note that while the use of a virtual MSS solves some of the AIMD issues, it has no impact on host interrupt issues, which are currently a big obstacle to obtaining very high-speed TCP in a host.

The virtual MSS modifies only the TCP additive increase, the multiplicative decrease can also be configured for a path and tuned by the WAD. Rather than decreasing the window by half on a congestion event, [2] we could designate

[2] A congestion event is one or more packet losses within a single round-trip time.

that for a particular path it only be decreased by 25%. One can choose the AIMD parameters statically based on path characteristics, or parallel-stream equivalence, or desired level of service. Floyd proposed a modified AIMD for high-delay, high-bandwidth flows that dynamically adjusts the AIMD parameters as a function of loss probability and current congestion window size. As the congestion window grows and shrinks, so do the AIMD values. We have experimented with Floyd's modified AIMD with *ns* simulation and with our TCP-over-UDP test harness and have added it to our Linux kernel so that the WAD can select it. We have also added Tom Kelly's scalable TCP to our Net100 kernel. Kelly's AIMD algorithm is independent of the RTT, so can more quickly recover from loss over a long-delay path. Figure 2 (a) compares standard TCP with Floyd's HS TCP and Kelly's TCP using the WAD. The plot shows three independent tests between LBL and CERN where we introduced congestion at about the 3 second mark with a burst of UDP packets. AIMD tuning usually can more than triple performance over a single, untuned flow and

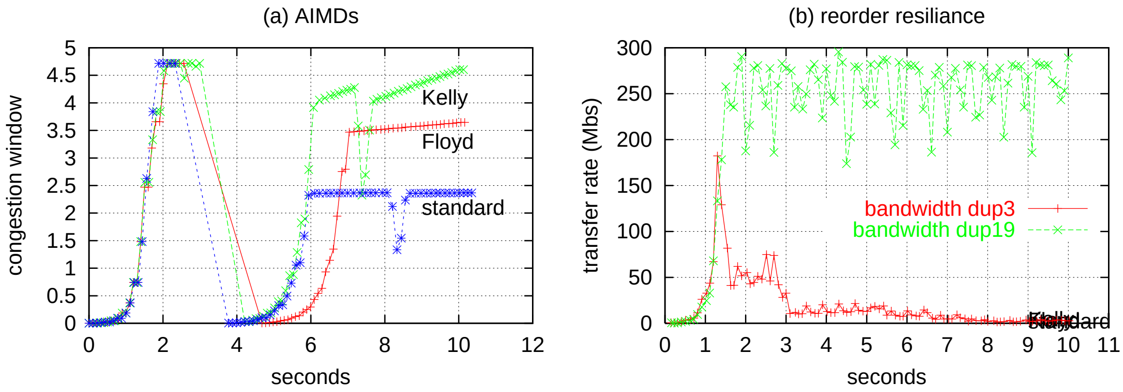

Figure 2: Tuned congestions avoidance and reorder threshold.

is competitive with parallel streams.

Part of our research has been to see if we could get a Net100-tuned TCP stream to perform as well parallel streams. A number of applications (GridFTP, bbftp) have been modified to use parallel TCP streams to work around buffer limitations and improve throughput. Parallel streams take advantage of the fact that TCP tries to share the bandwidth equally among all flows along a path. Performance is improved not only by getting a bigger share of the bandwidth, but k parallel streams will have k times larger aggregate buffer size. Slow-start will start k times faster. If a loss occurs in one stream, the multiplicative decrease will not be TCP's standard $1/2$, rather the congestion window will be reduced by only $1/(2k)$. The linear recovery can be k times faster (k segments per RTT rather than one), if all k streams are already in linear recovery.

To get a single TCP flow to perform as well as TCP streams, we could, for example, tune a single stream to have the TCP "characteristics" of 4 parallel streams by setting the additive increase to 4 and the multiplicative decrease to $1/8$. (We would also need to set the initial window to 4 to speed slow-start, but that is difficult for the WAD at this time.) As an example, Table 1 shows the throughput results of GridFTP transfers of a 200 MByte file from ORNL to LBNL (GigE/OC12, 80 ms RTT).

tuning	streams	Mbs	cong. events	rexmits
untuned	1	22.4	5	102
tuned	1	78.4	7	64
untuned	4	98.4	15	320

Table 1: GridFTP data rates from ORNL to LBNL.

During the second year of the project, we analyzed a number of "high-speed" protocols based on UDP (SABUL, RBUDP, FOBS, and Tsunami). These rate-based protocols often perform better than TCP, though they often are not net-friendly. Based on these studies, it would be interesting to investigate ways to reduce the burstiness of TCP.

Our kernel extensions also included an auto-tuning feature similar to Wu Feng's Dynamics Right Sizing. The Net100 autotuning can be enabled for all flows with a sysctl or can be tuned by the WAD on a per-flow basis. This auto-tuning ignores setsocket() options by the application and disassociates application network buffers from the kernel's

TCP buffer needs for a flow. The flow's TCP buffers will be controlled by the receiver's advertised window and will grow in proportion to the bandwidth-delay product estimated by the receiver.

In the final year of the project, we developed a light-weight WAD that did not require NTAF tuning data and could initiate path measurements between cooperating WAD-lite's if the user requested. The WAD-lite requires less network expertise, but it is still not a scientist-friendly interface. We think that just setting the kernel network parameters and the Net100 auto-tuning parameters at time of installation is probably the most user-friendly way of exploiting the Net100 tuning. The tuning daemon still remains a valuable tools for the network researcher.

Some of the paths over which we have been collecting NTAF data show significant reordering of packets. Standard TCP will treat out of order packets as a packet loss if the packets are more than three positions out of order. We have experimented with altering the re-ordering threshold for a path. Figure 2 (b) shows the instantaneous bandwidth for two tests from LBNL to ORNL (GigE/OC12, 80ms RTT, 2 MB buffers). One test uses the default threshold of 3 and reaches only 18 Mbs after 10 seconds. The flow experienced 9 congestion events and retransmitted 289 packets, but the receiver reported 289 duplicate packets received. So none of the packets were actually lost, they merely arrived out of order. We reran the test using a reordering threshold of 19 and reached 282 Mbs after 10 seconds with no packet loss. This tuning resulted in an order of magnitude improvement in throughput. We believe reordering will be an important issue in the future as Net100 and other projects look at multipath solutions.

3.1 Avoiding Losses

TCP detects network congestion when packets are lost, and our experiments with a virtual MSS and AIMD adjustments help TCP recover from losses faster. We have also tried to develop techniques to help TCP avoid packet loss. We have used the WAD to tune flows to reduce packet losses by adjusting TCP's slow-start and TCP's reordering threshold. We added Floyd's limited slow-start modifications to the Net100 kernel and saw performance improvements of a factor of two on some paths.

Another scheme to avoid packet loss is to use delay-based congestion avoidance. TCP Vegas and more recently, CalTech's FAST, used delay-based congestion avoidance. As part of our Net100 research in the project's final year, we have experimented with TCP Vegas using simulation (ns-2), and we implemented TCP Vegas in our TCP-over-UDP test harness. With encouraging results from these experiments, we ported an older Vegas implementation into the Net100 kernel and provided tuning options to the WAD. We are presently testing Vegas under various network conditions. Figure 3 illustrates how Vegas limits TCP to the path's bandwidth-delay product and avoids the packet loss that TCP experiences when it exceeds the link buffer capacity. The figure also illustrates our experiments with differing Vegas control parameters as well as the sensitivity to Vegas to resolution of the timestamps in the TCP packets.

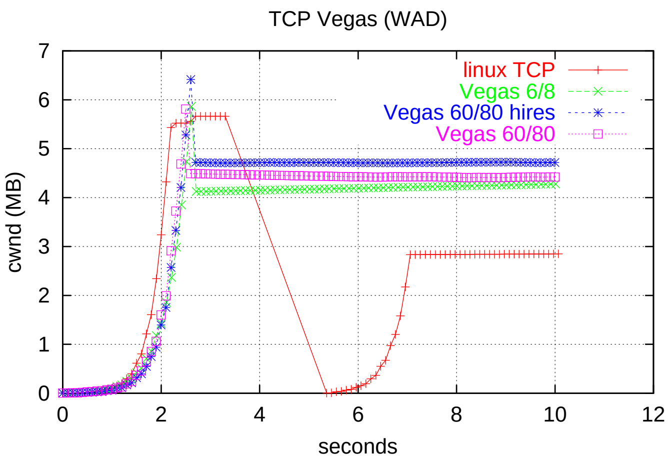

Figure 3: Standard TCP versus TCP Vegas.

Recent work by Feng and by Caltech's FAST project indicate that Vegas can be effective in avoiding packet loss by slowing the sender as the measured RTT starts to increase. Our results with Vegas were inconclusive, and we did not have time to integrate CalTech's FAST with Net100 to see if it would significant performance improvement.

In summary, the results of our Net100 project have improved network performance over high speed, long delay networks. We believe the Net100 project has met project milestones, achieved projects goas, and exceeded expectations.

4 Project Management

The preparation of the initial proposal, project reports, posters, and meetings are administered by PSC (Janet Brown). The collaborators have a phone conference every other week, access-grid meetings at least once a quarter, and meet at least twice a year face to face. PSC provides a project website (net100.org) that describes the project and recent accomplishments. The project also has a sourceforge site for the distribution of project software. Each institution provides a more detailed website describing recent results, talks, and papers (www.ornl.gov/ dunigan/net100 and www-didc.lbl.gov/net100/).

The project goals and milestones are derived from the original proposal. Goals and milestones are reviewed and amended semi-annually in accordance with DOE needs and advances in the network research community.

5 Outreach and Interactions

The project disseminates information to the DOE scientific community and the network research community through the Net100 web pages, seminars, technical reports, journal articles, and conference presentations. The Linux kernel modifications and Net100 tools are open-source and are being utilized at both national labs and university research institutions all around the world. The Net100 bandwidth testing tool that runs from any web browser is used by the ”common man” all over the world, getting over 200,000 visits since the project started. This tool was also the basis for the network diagnostic tool developed at Argonne National Lab.

We have described Net100 both to the operational network community (ESnet and Internet2 meetings), to the network research community, and to the scientific community. We have had more direct involvement with the SciDAC astrophysics and climate modeling applications group and with the high energy physics data grid projects. Net100 has been (and is currently) used to help profile, debug and tune various high bandwidth transatlantic links between the US and the CERN, Switzerland that are part of the EU DataTAG project. We have also collaborated with grid middleware research projects, in particular, we have focused on applying Net100 to GridFTP and worked closely with GridFTP developers on how to best tune TCP for use with parallel streams

We have had extensive interaction with the protocol research groups around the world. We have used our Net100 tools and testbeds to implement TCP extensions suggested by Sally Floyd, Tom Kelly, Wu Feng, and the CalTech FAST group. Our tuning daemon permits us to easily select a particular TCP variant for a specified path and compare the behavior of the TCP variants.

To compare our TCP tuning methods with competitive technologies, we have worked with the parallel streams researchers (Tom Hacker), and with the custom UDP protocol research teams (SABUL, FOBS, RBUDP, Tsunami). We also have interacted with the SCTP developers.

The Net100 team has worked with the network tools and measurement projects at Internet2 and SLAC. Net100 instrumentation can provide information beyond bits/second to the network measurement projects. Network measurements can be made smarter and less obtrusive by using the Net100 instrumentation (see PAM 2003 paper). Results from using NTAF to run *pathrate* and *pathload* (C. Dovrolis) and *netest* (G. Jin) over a variety of paths have provided valuable feedback to the developers of those tools, allowing them to greatly improve the accuracy of those tools. Figure 4 compares periodic estimations of available bandwidth on an OC12 (622 Mbs) path from LBNL to PSC using a variety of tools. The data from the NTAF archive is for a 24-hour interval. We have also utilized the tcpdump SCNM monitors to evaluate TCP and firewall bugs at ORNL.

We have worked with the national laboratories in trying to speed up wide-area HPSS/HSI transfers between the large computing centers. Much of this software is not based on Linux. We have had discussions with the IBM’s AIX network folks about incorporating the Net100 kernel mods. In the last year, working with Cray, we have ported the Net100 kernel to the network front end of the Cray X1 supercomputer, resulting in a factor of four improvement in wide area transfers! Recently, the Net100 patches were added to the SGI Altix supercomputer yielding similar improvements. Linux developers have recently invited the Web100/Net100 teams to contribute to the Linux 2.6 release. The Web100 MIB is being worked through the IETF standards process as well.

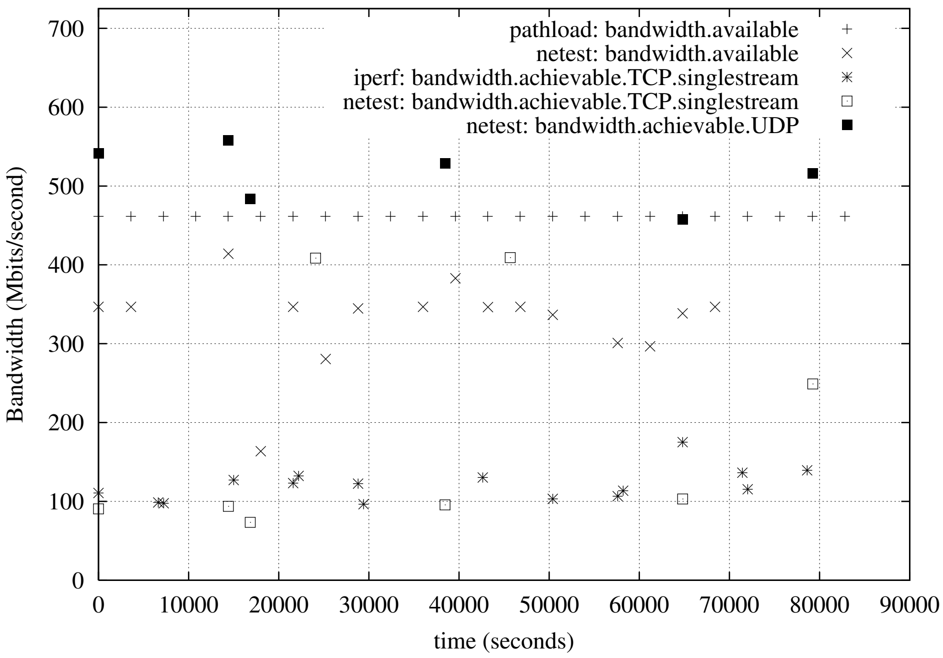

Figure 4: NTAF data comparing bandwidth tools.

6 Future research

If further funding were available, we would integrate the Caltech FAST Vegas-like kernel with our Net100 kernel and provide a tuning option to the tuning daemon. We had hoped that other researchers would add Web100/Net100 extensions to the FreeBSD kernel, but sadly that did not happen this past year. We continue to believe that components of our Net100 kernel will be integrated into production Linux distributions.

We believe Net100/Web100 has been beneficial to the scientific and network community both in understanding and in improving network performance on high speed networks.

7 Site highlights

The Net100 was a collaboration of many individuals besides the principal investigators. Each site helped with many parts of the projects, and the summary below just notes the lead institution for a particular phase of the research.

7.1 LBNL highlights

LBNL's has designed and deployed the data collection portion (NTAF) of the Net100 project. LBLNL has developed the database schema's and web-based retrieval software for interrogating and plotting the Web100 data collected from the Net100 nodes deployed across the world. The LBNL team designed the interface that allows the network tuning daemon to retrieve current path characteristics from the NTAF database. The LBNL also designed a Python version of the WAD that can easily incorporate equation-based tuning parameters. The NTAF research also included the deployment and evaluation of numerous network probing utilities. LBNL's Netlogger has been used by the project to collect additional NTAF data and to fully characterize application bottlenecks. The LBNL team has worked with the Globus/GridFTP developers to see how Net100 might be utilized by GridFTP. Brian Tierney served as co-chair for PFLDnet in 2004.

Besides Brian, Jason Lee and Martin Stouffer from LBNL participated in Net100 work. For details on past and ongoing Net100 work at LBNL, visit http://www-didc.lbl.gov/net100/

7.2 ORNL highlights

ORNL's Net100 research team has focused on TCP protocol accelerants. Over the past two and a half years, the ORNL team has added event notification to the Web100 Linux kernel extensions which permits the Net100 tuning daemon (WAD, also developed at ORNL) to dynamically tune network applications. The ORNL team has used *ns-2* simulations and a NISTNet emulation testbed in its TCP protocol evaluations. The ORNL team also developed a TCP-over-UDP protocol, that allows fast deployment and evaluation of TCP extensions, even on systems where kernel modifications were not allowed. ORNL has worked with the other collaborators in collecting and evaluating network data on real data transfers for climate and the HPSS storage systems at ORNL and NERSC. to the Linux kernel

In the past six months, ORNL has added TCP Vegas to the Net100 kernel as a tunable option. Numerous experiments were conducted with Vegas comparing it with CalTech's FAST. Test and evaluation of the Vegas modifications is on-going. ORNL has also ported the Net100 kernel to the Cray X1 and the SGI Altix, this has resulted in a 4x improvement in wide-area network throughput for these supercomputers. ORNL developed a lighter-weight tuning daemon (no NTAF attachments and default settings that should provide high throughput).

Besides Tom, other contributors to Net100 at ORNL included Nagi Rao, Bill Wing, Florence Fowler, and Steven Carter. Details and time lines of ORNL Net100 work is provided at www.csm.ornl.gov/ dunigan/net100/

7.3 PSC highlights

PSC's Net100 research team has worked on pushing the Net100/Web00 kernel into the standards track and has been working the with vendors to incorporate the kernel instrumentation to other operating systems. PSC has coordinated kernel extensions provided by the other collaborators and has added an automatic TCP tuning feature to the Net100/Web100 kernel. PSC has also developed several network diagnostic tools that utilize the Web100 instrumentation. PSC along with the NCAR team has provided project management, including maintaining the Net100 web site and download area. The PSC team worked with the Linux developers to include portions of Web100 in the next major Linux release. PSC has also been negotiating with CalTech to get FAST included as a Net100 tuning option.

Besides Matt, other PSC contributors included Janet Brown, Wendy Huntoon, John Heffner, Raghurama Reddy and at NCAR Basil Irwin, Marla Meehl, Peter O'Neil. See www.net100.org

Net100 Papers

Dunigan,Mathis,Tierney, *A TCP Tuning Daemon*, SuperComputing 2002

Tirumala, Cottrell, Dunigan, *Measuring end-to-end bandwidth with Iperf using Web100*, PAM 2003

Rao, Feng, *Performance Tradeoffs of TCP Adaptation Methods*, ICN'02

Mathis, Heffner, Reddy, *Web100: Extended TCP Instrumentation for Research, Education and Diagnosis*, ACM Computer Communications Review, Vol 33, Num 3, July 2003

Carlson, Dunigan, Hobby, Newman, Streck, Vouck, *Measuring End-to-End Internet Performance*, Network Magazine '03

Tierney, *Using NetLogger and Web100 for TCP Analysis, First International Workshop on Protocols for Fast Long-Distance Networks*, Switzerland, '02

Jin, Tierney, *Netest: A Tools to Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput*, ITRE '03

Rao, *On design of measurements for end-to-end delay minimization in wide-area networks*, ICACC '01

Rao, Feng, *Performance trade-offs of TCP adaptation methods*, Int. Conf on Networking '02

Rao, *Probabilistic guarantees on message delays*, ICCCN '02

Tierney, Lee, Gunter, Stoufer, Dunigan, *Improving Distributed Application Performance Using TCP Instrumentation*, LBLNL tech report

Dunigan, Carter, Fowler, *An Evaluation of UDP Transport Protocols*, ORNL Tech Report '04

Dunigan, Fowler, *A TCP-over-UDP Test Harness*, ORNL Tech Report '02

Net100 Presentations

Net100, SuperComputing 2001, BOF

Net100 Overview, ESnet ESCC, Feb '03 and April '02

A TCP Tuning Daemon, SuperComputing 2002

End-to-End Network Issues, DoE Science Computing Conference, '03

Net100, DOE network workshop, Oak Ridge, November, '01

Measuring end-to-end performance with iperf using Web100, PAM03

Net100, Intenet2 End-to-end workshop, Jan '02

Net100 Overview, Web100 workshops, July '01 and '02

Net100 Schedule and Milestones (condensed)

Year 1:
- deploy Web100 hosts, build NISTNet testbed
- evaluate network measurement tools, instrument with web100
- measure/understand DOE bulk transfer applications
- design NTAF and kernel extensions for tuning
- evaluate TCP protocol accelerants (DRS, AIMD)

Year 2:
- deploy/extend NTAF, develop/deploy NTAF database/visualization tools
- deploy/test tuning daemon, add TCP accelerants (Floyd, Kelly, autotuning)
- integrate NTAF with tuning daemon
- deploy/compare parallel streams (GridFTP) and UDP transports
- pursue ports to other OS's

Year 3:
- make user friendly, demonstrate with grid applications
- add multipath/multistream options, TCP accelerants (Vegas/FAST)
- pursue ports to other OS's
- develop tuning options for dedicated path (lambda)