Journal of Physics: Conference Series **219** (2010) 062062 doi:10.1088/1742-6596/219/6/062062

High Performance Data Transfer and Monitoring for RHIC and USATLAS

J Packard[1,5], D Katramatos[1], J Lauret[1], K Shroff[1]
J DeStephano[1], M Ernst[1], J Hover[1], T Ichihara[3], D Kim[2], S McKee[4],
M L Purschke[1], Y Watanabe[3], J Woo[2], I Yoo[2], D Yu[1]

[1] *Brookhaven National Laboratory, Upton, NY, USA*
[2] *Korea Institute of Science and Technology Information, Yuseong-gu, Daejeon and Dongdaemun-gu, Seoul, Korea*
[3] *RIKEN (PHENIX Computing Center in Japan), Wako City, Saitama, Japan*
[4] *University of Michigan, Ann Arbor, MI, USA*

E-mail: jpackard@bnl.gov

Abstract. Modern nuclear and high energy physics experiments yield large amounts of data and thus require efficient and high capacity storage and transfer. BNL, the hosting site for RHIC experiments and the US ATLAS Tier 1 Center, plays a pivotal role in transferring between other sites in the US and around the world for data distribution and processing. Each component in the infrastructure from data acquisition system to local analysis facility must be tuned, tested, monitored, and sometimes diagnosed to transfer such a massive volume of data, often over long distances. Maximal performance can be reached by performing a combination of hardware optimization, TCP tuning, transfer application tuning and network architecture optimization.

1. Introduction

Modern nuclear and high energy physics experiments yield large amounts of data and thus require efficient and high capacity storage and transfer. BNL, the hosting site for the RHIC experiments PHENIX [2] and STAR [3], and the Tier 1 Center for US ATLAS [1], plays a pivotal role in transferring between sites in the US and around the world for data distribution and processing. A team from the RACF group at BNL supports the experiments by ensuring the required data transfer rates are attained. The focus of this group is to identify the hosts involved in the experiment's data transfer infrastructure and then to maximize the bandwidth capabilities from source disks to destination disks. Typically, over long distance transfers, the majority of our work involves the network (comprising the WAN, LANs, and host network configuration) rather than disk or data transfer software since disk bottlenecks are compensated for by using multiple hosts or purchasing high performance disks, and since the the data transfer software is either straight forward and reliable in certain cases or managed by expert data storage personnel in other cases.

[5] Author to whom any correspondence should be addressed.

This paper describes a five step process used by this team to maximize throughput for each data transfer path that consists of the following steps: tune, test, monitor, troubleshoot and eventually harvest the fruits of hard work after passing the basic performance requirements (as monitored). This process is iterative as shown in figure 1, which schemes the flow of the different stages.

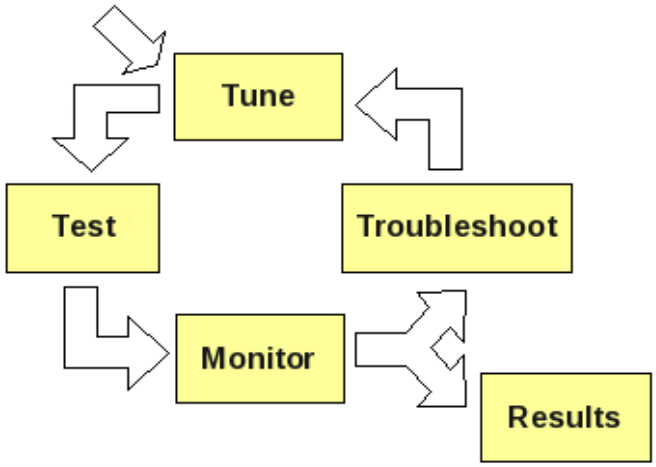

Figure 1: Maximizing throughput flowchart

These phases could be briefly described as follow:

1. Tune - apply optimal network tunings based on source to destination distance, bandwidth requirements, and previous test results.
2. Test – test at various layer (UDP, TCP, disk), test combinations of network parameters, test at various hops along the network path.
3. Monitor – deploy and use monitoring and graphing tools to gather bandwidth and error information and to study behaviour. If performance requirements are met, exit out of loop to the "Results" step. Otherwise go to the "Troubleshoot" step.
4. Troubleshoot – work with network engineers to identify problems, reconfigure, reroute, upgrade, etc... as needed.
5. Results – report results after passing basic performance requirements (as monitored).

In the next sections, we will present the various phases of the process. We will then present the improvements attained through this process for each experiment as case studies.

2. Tune

Tuning plays an important role in the process and is in fact fundamental in long distance network data transfer. The root of the need for tuning resides in the TCP protocol's behaviour and functional communication requirements in which packets sent from a source to a destination need reception acknowledgement (ACK) according to the protocol's flow chart. Over long distance, the round trip time (RTT) could be considerable and a drop of ACK packet will cause the transfer to be retried. Even a small percentage of packet drop will cause significant delays in the overall transfer rate.

Since the installation-default maximum window size parameters for many operating systems are typically small, one should increase them based on source to destination round trip time (RTT) and desired bandwidth. The desired bandwidth cannot be higher than what the end-to-end network supports of course. If a host will be transferring to a set of hosts, each located at a different distance,

the tuning needs to be done for the one that is furthest away. As a rule of thumb, the calculation of the TCP window size is made using the bandwidth delay product formula [13]:

*tcp_window_size = RTT * desired_bandwidth*

Typically, kernel based TCP parameters could be adjusted via specific system configurations. On Linux, these values are specified in `/etc/sysctl.conf` and the parameters are:

1. *net.core.wmem_max* - kernel write maximum window size
2. *net.core.rmem_max* - kernel read maximum window size
3. *net.ipv4.tcp_wmem* (third value) - TCP write maximum window
4. n*et.ipv4.tcp_rmem* (third value) - TCP read maximum window

[14] For example, if *tcp_window_size* = 4MB (4194304 bits), `/etc/sysctl.conf` should contain:

...
*net.core.wmem_max = **4194304***
*net.core.rmem_max = **4194304***
*net.ipv4.tcp_wmem = 4096 65536 **4194304***
*net.ipv4.tcp_rmem = 8192 131072 **4194304***
net.core.netdev_max_backlog = 10000
…

In high throughput scenarios, increasing *net.core.netdev_max_backlog* is also recommended for increasing the number of incoming packets that can be queued when the interface receives packets faster than the kernel can process them. To load the values in `/etc/sysctl.conf`, one may use:

```
% sysctl -p
```

Another tuning parameter is the maximum transmission unit (MTU), which is the largest size packet or frame for the network segment considered. One may want to consider using jumbo frames or TCP offloading to reduce packet processing overhead and thus improve performance. Finally, we tune other host and network parameters as knowledge dictates, letting previous tests results factor in. Examples of the dramatic change before and after tuning can be seen in figures 2.

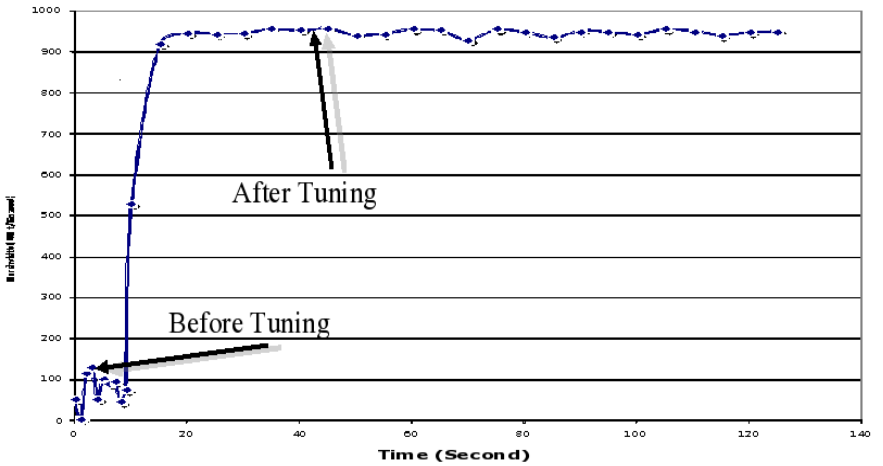

Figure 2: Network transfer speed as a function of time in Mb/s before and after tuning. Tuning increases the transfer rate by an order of magnitude.

3. Test

3.1. Types of Testing

To isolate performance problems, tests should be performed at multiple layers, using various parameters combinations, and at multiple hops along the way as described later. Since network data transfers involves independent components (from disk to network themselves for example) which affects the overall performance in a convoluted way, it is important to realized that the use of multiple tools and attempting to test different components separately is important. Testing (using tools such as Iperf, Gridftp, and PerfSONAR) should be done at least for the following space:

1. UDP to obtain packet loss statistics
2. Memory to memory (TCP) to determine network performance
3. Memory to disk to determine destination disk performance
4. Disk to memory to determine source disk performance
5. Disk to disk to determine likely production performance

Other tests are targeted to find optimal tuning parameter combinations (using tools such as BNL's MetaPerf) for the following parameters:

1. Number of streams
2. Kernel TCP parameters
3. NIC parameters

For a full comprehension of the effect of the network components, such as firewall, on performance, tests should include:

1. From and to host inside firewall
2. From and to host outside firewall
3. From and to intermediate WAN host(s) if bottleneck is discovered

3.2. Fixed TCP Window Size vs. Auto-Tuning

Most data transfer applications such as Iperf and GridFTP allow specifying a fixed TCP window size. But modern operating systems support auto-tuning, which in our experience, works well in most cases, and thus removes the need to specify a fixed TCP window size [16]. However, if performance (particularly ramp-up time) using auto-tuning is problematic, one should also try a fixed TCP window size.

3.3. Number of Socket Streams

Most network performance testing and data transfer applications, such as Iperf and GridFTP respectively, allow specifying the number of simultaneous socket streams to use. Using multiple streams, we have found, is an effective way to increase utilization of bandwidth and produce a more stable transfer. For, if a single packet is lost, which is quite probable over long distances, with multiple streams only one stream will scale back leaving the others unaffected. Also, firewalls have per-stream limits, so if crossing a firewall is unavoidable, the bandwidth needs to be divided up such that each stream can be processed by the firewall. One should be cautious about multiple streams, however. If using a fixed TCP buffer size, it will use memory equal to the number of streams multiplied by the TCP buffer size, which has the potential to be large and wasteful or possibly crash the system. One should also be cautious to not overload their site's firewall with too many streams coming from multiple hosts. Lastly, using multiple streams is in some sense violating fair-sharing of the network that is implicit in TCP’s design.

4. Monitor

Tuning without a way to see and trace the results is as good as a shot in the dark. Monitoring is hence at the very core of our ability to iterate on our tuning and performance improvements. There is a wide range and set of tools available on the market for monitoring, and typically a facility is equipped with from standard to home-made custom tools. We use the following graph based tools in monitoring and believe they provide a complete set of information serving our purpose:

1. Cacti [7] – Cacti is a complete network graphing solution providing a framework in which one can add graphs tailored to its need. It is designed to harness the power of RRDTool's data storage and graphing functionality hence providing historical and time dependent graphs as defined by the user. At BNL, we use Cacti to monitor the diverse segments of our data transfer network.
2. Ganglia [8] – Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. Ganglia provides a set of standard graphs such as the NIC data transfer of a source machine. Such measurement can be compared to what one gathers from the LAN using Cacti. Any discrepancies may indicate a problem. As Cacti may show aggregate information, we also leverage the information from Ganglia installed on all the sources to monitor discrepancies between the diverse data sending nodes.
3. Netflow [9] – This monitoring tool is used for viewing source and destination specific streams obtained from Cisco routers as redundant cross-correlation information.
4. PerfSONAR / PerfSONAR UI [5] – is an infrastructure for network performance monitoring, making it easier to solve end-to-end performance problems on paths crossing several networks. It is used in our tuning for viewing comparative and history plots of regularly scheduled tests.
5. MetaPerf – MetaPerf is a tool for conducting parametrical analysis of network performance. MetaPerf functions as a wrapper to existing network performance tools, such as iperf, and data transfer tools, such as GridFTP. MetaPerf provides features like automated parameter sweeps between multiple pairs of nodes and graphical output to display measured network performance in terms of bandwidth/throughput and number of dropped packets for various network protocols like TCP and UDP. This output allows for the network administrators to reach conclusions regarding the effects that each individual parameter as well as combinations of certain parameters have on network performance.

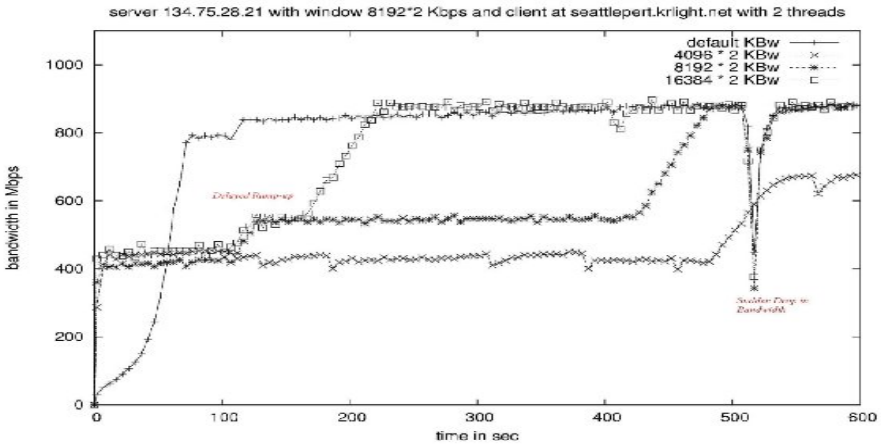

Figure 3: MetaPerf graph showing bandwidth in Mb/s over time for combinations of number of streams and TCP window sizes.

We also use the following command line based tools to monitor with text output:

1. Iperf [4] – Iperf performs memory to memory network tests and displays transfer speed measurments (use -i for defining the reporting interval).
2. Globus-url-copy (GridFTP protocol) [10] – Globus-url-copy is a GridFTP data transfer application that displays transfer speed measurements (use -vb option for showing the number of bytes transferred). It adds the GSI authentication overhead and the general grid

infrastructure on top of simple transfers. It is the underlying transfer mechanism used in our production transfer.

And we use a combination of Ethtool [11] to display or change ethernet card settings and Ifconfig [12] to set up network interfaces and allow the user to view information about the configured network interfaces including such errors as packet drop.

5. Troubleshoot

If residual performance exists, collaborative work with network engineers at both ends is essential to troubleshoot performance issues such as packet drop or inefficient or non-symmetric routing. One may need to understand network topology in depth including bandwidth pipe sizes, MTU, site firewall rules, routing rules, and DSCP. Other important steps are also to understand hosts in depth including network parameters, host-based firewall configuration, disk IO, and NIC make and model. As one such example, an important discovery of ours to note is that iptables needs to allow ACKS in and SYN-ACKS out for optimal performance; if not, explicit rules to allow this should be created. As necessary and upon investigation and findings, one may need to upgrade, reconfigure, or modify the routing implemented by network devices and may also need to bypass firewall if there is severe performance degradation.

After following these guidelines and check list, it is necessary to proceed to the "Tune" step again and iterate until the targeted performance goal is reached.

6. Results

Following are the results of following the process described thus far to each of the experiments.

6.1. US ATLAS

BNL is the Tier 1 Center for US ATLAS. After receiving and storing data from CERN (Tier 0 site), the data (and derived data) is transferred to and from multiple Tier 2 sites for processing and analysis. These Tier 2 sites are located across the US and the transfer traverses wide area networks, ESnet and Internet2.

A total throughput of 3.7 Gb/s disk-to-disk was attained from BNL to each of the Tier 2's simultaneously during a multi-day test [figure 4]. Individual tests were also performed [figure 5]. These individual tests assist in isolating specific paths as well as helping to determine degradation due to disk and/or grid software that lies on top of the network layer. We are planning to reach a milestone of 1GB/sec by the end of the second quarter of 2009.

The outcomes of this work are being integrated into the ATLAS production and analysis framework to allow US ATLAS regional centers to more effectively perform data processing.

6.2. PHENIX

BNL transfers data to RIKEN CCJ in Japan for PHENIX traversing wide area networks, ESNet and SINET.

A total throughput of approximately .96 Gb/s average and 1.5 Gb/s peak disk-to-disk was attained from BNL to RIKEN CCJ during a multi-day test at the beginning of 2008 [figure 6]. Network tests, which remove disk IO limitations, reached a peak of 2.7 Gb/s [figure 7]. These results are some of the fastest ever achieved between the US and Asia according to ESnet engineer feedback.

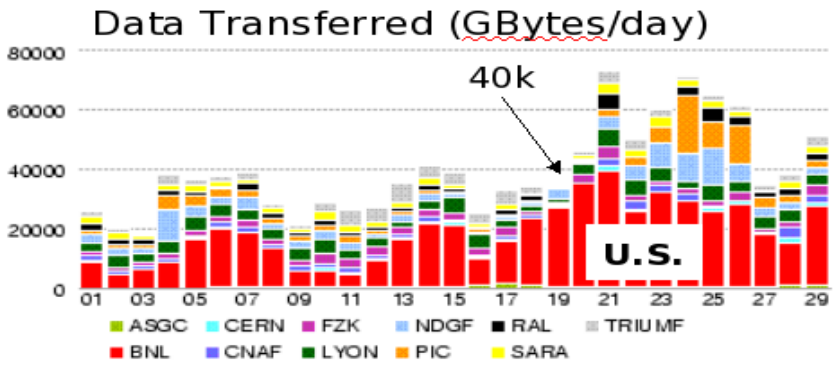

Figure 4: Total disk-to-disk throughput from BNL to USATLAS Tier 2 sites (40000GB/day * 8bit/1Byte * 1day/24*60*60sec = 3.7Gb/s)

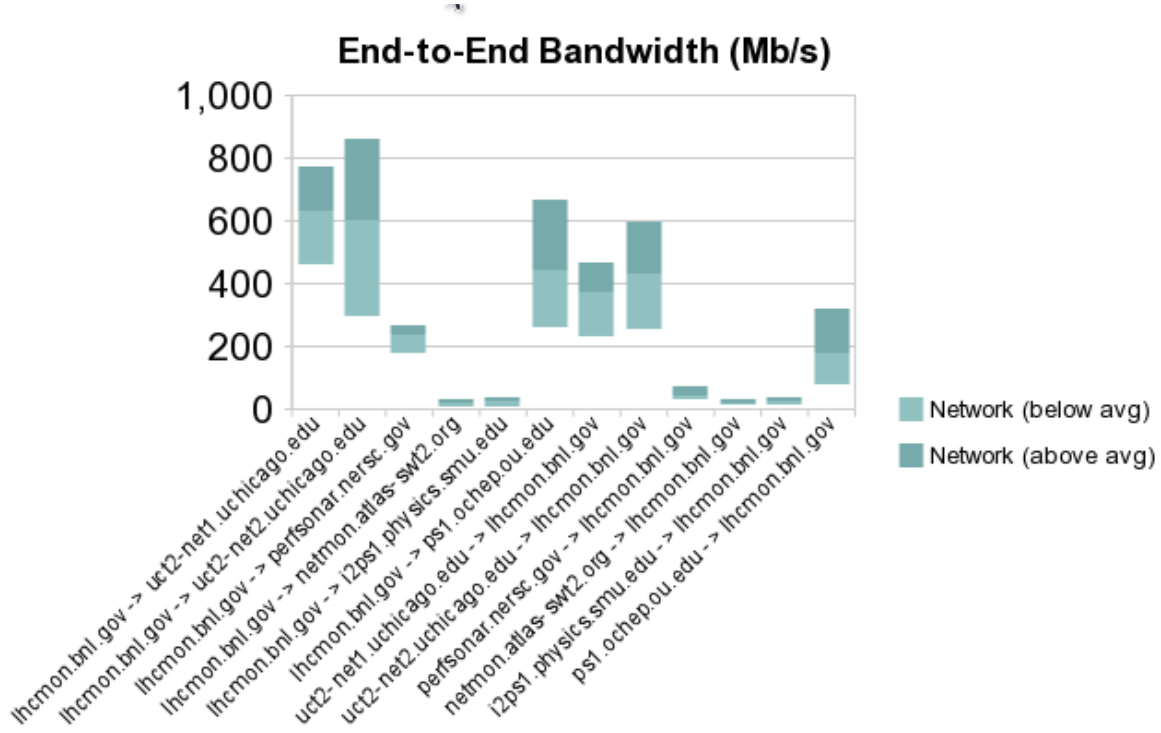

Figure 5: Comparative network tests for USATLAS from BNL to its Tier 2 sites (generated from PerfSONAR UI beta)

The outcomes of this work are being integrated into the global data taking and reconstruction framework for RHIC experiments to leverage the computing resources of RHIC international collaborators.

6.3. STAR

BNL transfers a portion of STAR's data to KISTI in Korea traversing wide area networks ESNet, PNW Gigapop, and GLORIAD.

A total of 1 Gb/s disk-to-disk was attained from BNL to KISTI during a multi-day test [figure 8] at the same time that 4 Gb/s was attained to BNL's HPSS (High Performance Storage System) [figure 9]. As

previously stated [section 4], it is important to have multiple measurement of the network data transfer to make sure the network data transfers are as expected and consistent end-to-end.

This provides the potential of establishing RHIC Tier 1 centers and data redistribution hubs in Asia (STAR already has 27% of its institutional workforce located in Asia).

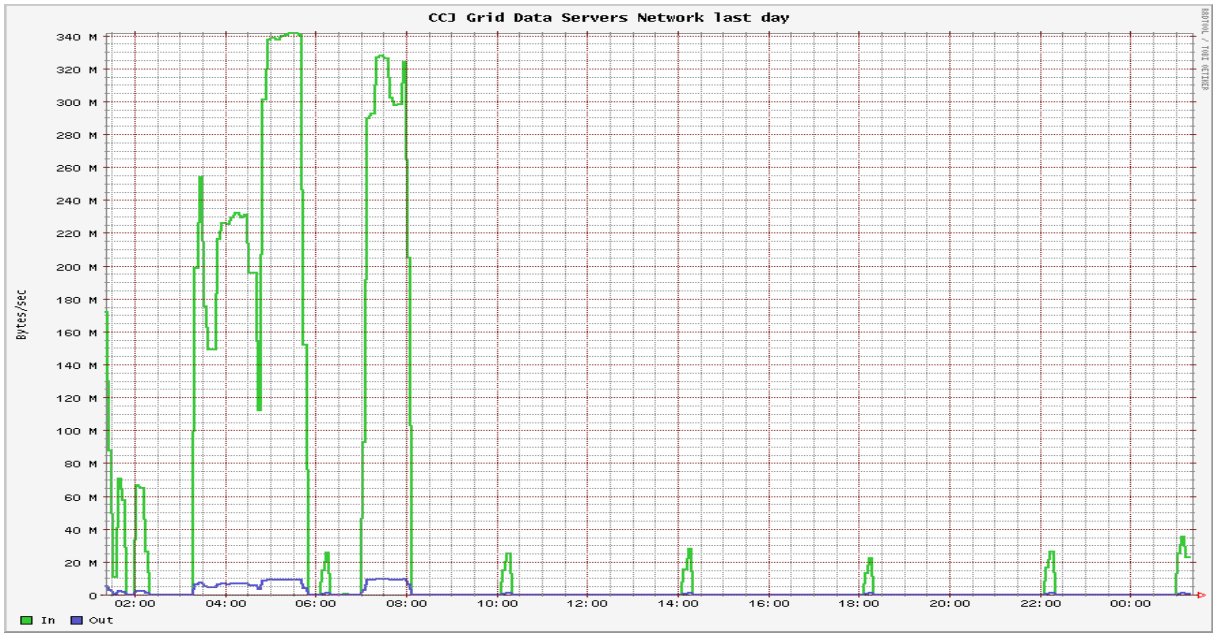

Figure 6: Total network throughput showing 340 MB/s (2.7 Gb/s) peak (see "In" data) over for PHENIX from BNL to CCJ (generated from Ganglia)

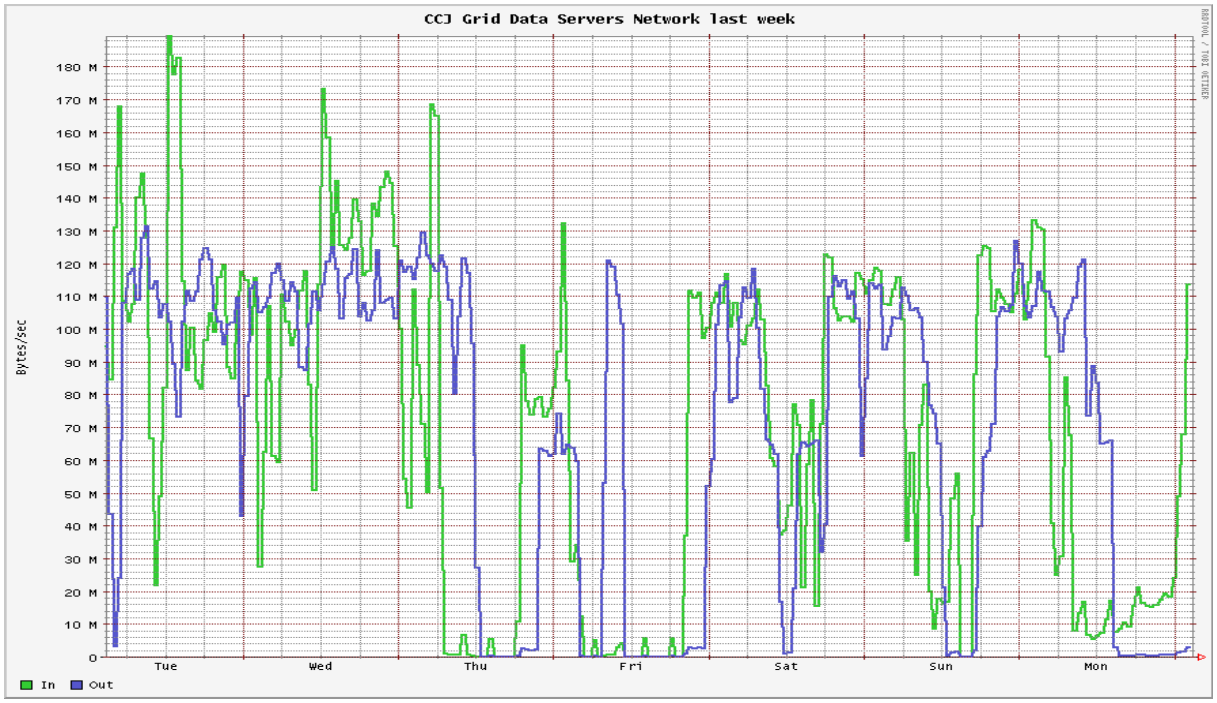

Figure 7: Total disk-to-disk throughput showing approximately 120MB/s (.96 Gb/s) average and 190MB/s (1.5 Gb/s) peak (see "In" data) for PHENIX from BNL to CCJ (generated from Ganglia)

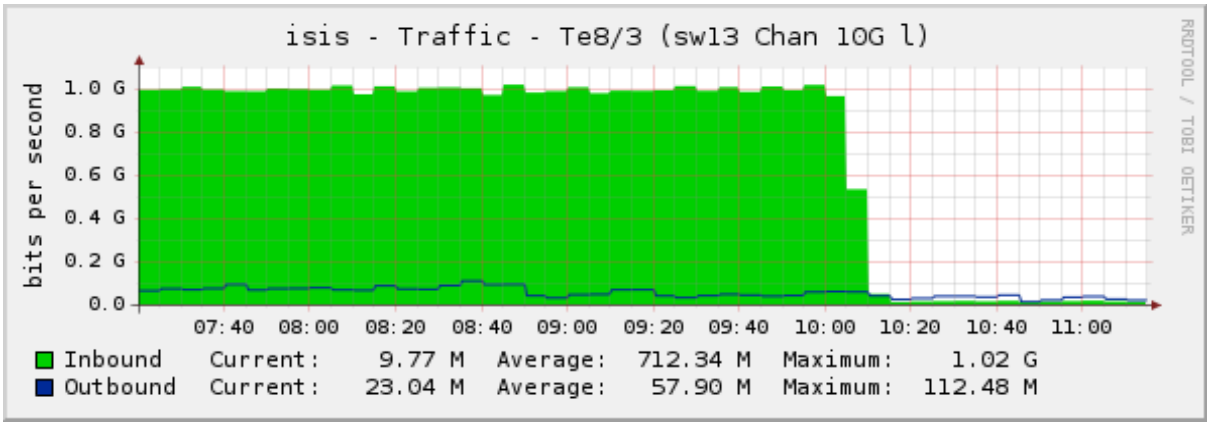

Figure 8: Total network throughput showing 1 Gb/s disk-to-disk (see "Inbound" data) for STAR from BNL to KISTI (generated from Cacti)

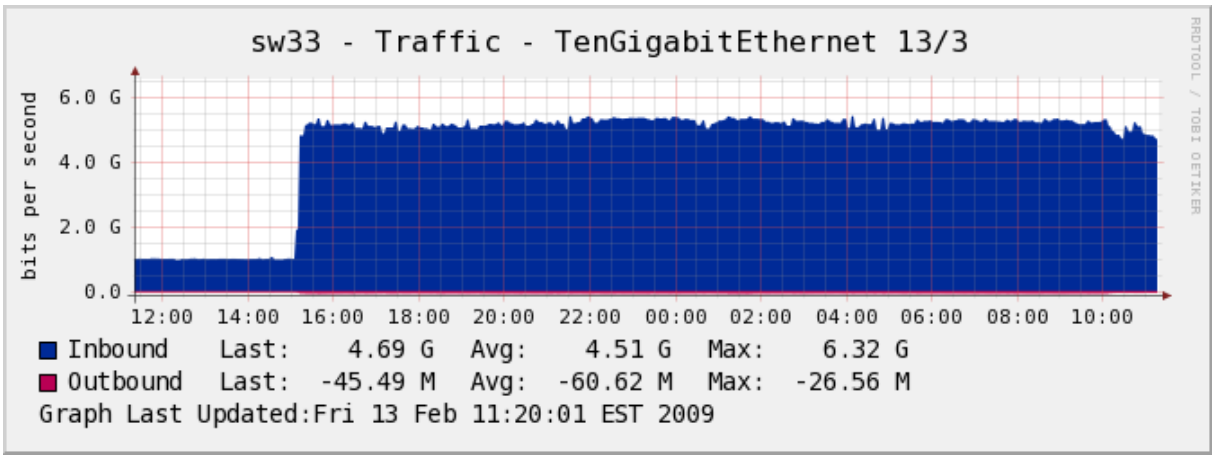

Figure 9: Total network throughput showing 4 Gb/s (see "Inbound" data) from the STAR counting house at BNL to HPSS at BNL (generated from Cacti). 1 Gb/s is from KISTI transfer.

7. Conclusions

This paper presented an overall process and specific techniques for improving data transfer performance for experiments that require efficient and high capacity storage and transfer. Performance tuning is an iterative process involving not only tuning but monitoring, troubleshooting and often requires assistance from network providers for optimal performance. For the experiment, such tuning is well invested time spent as it could lead to an order of magnitude difference in data transfer achieved end-to-end over long distances. In our paper, we have showed the effect of our procedure and techniques applied to the experiments hosted by BNL, hence supporting their data distribution plans and scientific milestone.

8. Acknowledgements

We would like to than out colleagues from ESNet, Internet2 Kreonet2, GLORIAD, SINET, without whom optimal performance tuning would not have been achieved. This work was supported in part by the HENP Divisions of the Office of Science of the U.S. DOE and the U.S. NSF.

References

[1] *US ATLAS*, retrieved May 12, 2009 from http://www.usatlas.bnl.gov/
[2] *PHENIX* front page, retrieved May 12, 2009 from http://www.phenix.bnl.gov/
[3] *STAR*: The STAR Collaboration, retrieved May 12, 2009 from http://www.star.bnl.gov/
[4] *Iperf*, retrieved May 12, 2009 from http://sourceforge.net/projects/iperf
[5] *perfSONAR*, retrieved May 12, 2009 from http://www.perfsonar.net/
[6] *MonALISA*, retrieved May 12, 2009 from http://monalisa.caltech.edu/monalisa.htm
[7] *Cacti, retrieved* May 12, 2009 from http://www.cacti.net/
[8] *Ganglia*, retrieved May 12, 2009 from http://ganglia.info/
[9] *Netflow,* retrieved May 12, 2009 from http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html
[10] *globus-url-copy*, retrieved May 12, 2009 from http://www.globus.org/toolkit/docs/4.0/data/gridftp/rn01re01.html
[11] *ethtool*, retrieved May 12, 2009 from http://gd.tuwien.ac.at/linuxcommand.org/man_pages/ethtool8.html
[12] *ifconfig*, retrieved May 12, 2009 from http://linux.die.net/man/8/ifconfig
[13] Pittsburgh Super Computing Center TCP Tuning Guide, retrieved May 12 2009, from http://www-didc.lbl.gov/TCP-tuning/
[14] Ipsysctl tutorial 1.0.4, retrieved May 12, 2009 from http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html
[15] Yu D 2008 BNL RHIC WAN Data Transfer and Grid Activity *RACF/BNL Networking Group at ITD, Upton, NY* slides 22-26
[16] Weigle E and Feng W 2002 A Comparison of TCP Automatic Tuning Techniques for Distributed Computing *IEEE Computer Society* 265
[17] MetaPerf, retrieved May 12, 2009 from https://www.racf.bnl.gov/metaperf