Demonstration of a Prototype System-Aware Data Transfer Mechanism for Computational Grids

Phillip Dickens, Ondrej Hrebicek, Vinod Kannan, Arun Mathew, Jesse Yurkovich, Theron Voran
Illinois Institute of Technology

William Gropp
Argonne National Laboratory

Abstract[1]

We are developing a high-performance data transfer engine that collects and incorporates *system-level knowledge* into the congestion/flow control mechanism for increased performance and better utilization of end-to-end system resources. As part of this research, we are developing a visualization system that displays the current state of major system components, the current state of the congestion/control flow mechanism, and the current *packet-loss profile*, which shows the distribution of lost packets within the given transmission window. The demonstration will focus on how the congestion/flow control mechanism responds intelligently to changes in the state of the end-to-end system and how changes in various system components are reflected in changes to the packet-loss profile. It will be shown that (quite often) there are discernable patterns in the packet-loss profile that appear to depend on the root cause of the observed data loss. This phenomenon leads us to speculate that it may be possible to distinguish packet loss caused by congestion in the network and packet loss caused by other system components. Such a distinction, in turn, could lead to even more intelligent and focused responses to problems that develop during a long-running large-scale data transfer.

1. Introduction

The national computational landscape is undergoing radical changes as a result of the introduction of cutting-edge networking technology and the availability of powerful, low-cost computational engines. This combination of technologies has led to an explosion of advanced high performance distributed applications that, because of the limited bandwidth and best-effort nature of the Internet1 environment, were heretofore infeasible. Concurrently, research efforts have focused on the development of *Grid computing*, a fundamentally new set of technologies that create large-scale distributed computing systems by interconnecting geographically distributed computational resources via very high-performance networks. The advanced applications being developed to execute in Grid environments include distributed collaboration across the Access Grid, remote visualization of terabyte (and larger) scientific data sets, large-scale scientific simulations, Internet telephony, and multimedia applications. The emerging Grid technologies are becoming increasingly important to the national computational infrastructure, and research projects aimed at supporting these new technologies are both critical and timely.

Arguably, Grid computing will reach its vast potential if, *and only if*, the underlying networking infrastructure (both hardware and software) is able to transfer vast quantities of data across (perhaps) quite long distances in a very efficient manner. Experience has shown, however, that advanced distributed applications executing in existing large-scale computational Grids are often able to obtain only a very small fraction of the available underlying bandwidth [3, 8][2]. The reason for such poor performance is that the Transmission Control Protocol (TCP), the communication mechanism of choice for most distributed applications, was not designed and is not well suited for a high-bandwidth, high-delay network environment [2, 8]. This issue has led to research aimed at improving the performance of the TCP protocol itself in this network environment [2], as well as developing application-level techniques that can circumvent the performance problems inherent within the protocol [5, 9, 10].

Despite all of the research activity undertaken in this area, significant problems remain in terms of obtaining available bandwidth while being in some sense fair to competing flows. While TCP is able to detect and respond to network congestion, its very aggressive congestion-control mechanisms result in poor

[1] Please note that the final paper will be in the proper LNCS style.
[2] Please note that due to space limitations the reference list is representative rather than comprehensive.

bandwidth utilization even when the network is lightly loaded. User-level protocols such as GridFTP [1], RUDP[9], and previous versions of FOBS [4, 5, 7] are able to obtain a very large percentage of the available bandwidth but rely on the characteristics of the network to provide congestion control (that is, they generally assume there is no contention in the network). Other approaches such as SABUL [10] attempt to bridge this gap by using UDP for fast delivery with a back-off mechanism that is triggered when the loss rate climbs above a certain threshold.

Each of these approaches shares the view that the issue of large-scale data transfers is largely (or exclusively) a network-level problem. When viewed from this perspective, *all data loss* is interpreted and treated as a network congestion event. However, *any* component in the end-to-end transmission path can cause data to be delayed or lost, and thus the problem is more appropriately viewed from a *system-level perspective*. Our preliminary research strongly suggests that collecting system-level information and *integrating such information* into the congestion/flow control mechanism offers the potential for significant gains in terms of both increased performance and better utilization of system-level resources**.** As discussed in [6], this new perspective opens the door to a set of very powerful capabilities unique to this approach. Such capabilities include the following:

- The ability to distinguish data loss caused by network conditions from data loss caused by other factors. Knowledge of the root causes of observed behavior leads to corrective actions that are more focused, more effective, and, in the final analysis, less expensive than current approaches allow.
- The ability to perform proactive rather than reactive congestion/flow control. If the transfer mechanism can accurately predict the state of the system and the state of the transfer itself, it can, in many instances, predict in advance those times during which the data transfer needs to modify its behavior. Without such knowledge, the control mechanism has to *infer* when it needs to modify its behavior based on the increasing/decreasing packet loss.
- The ability to collect a wealth of information, over long time periods, regarding the state of the system from the perspective of the application. This historical information may help to predict the future behavior of the end-to-end system (or a particular component therein), which may, in turn, be used to further improve the performance of the congestion/flow control algorithms.

2 Scientific Foundations

The prototype system-aware data-transfer mechanism is based on our previous work with FOBS: a simple, user-level communication mechanism designed for large-scale data transfers in the high-bandwidth, high-delay network environment typical of computational Grids. FOBS is UDP based and provides reliability through an application-level acknowledgment and retransmission mechanism. Experimental results have shown that FOBS performs extremely well in a computational Grid environment, consistently obtaining on the order of 90% of the available bandwidth across both short- and long-haul network connections[5, 7]. Thus FOBS addresses quite well the issue of obtaining a large percentage of the available bandwidth in a Grid environment. However, FOBS is a very aggressive transport mechanism that does not adapt to changes in the state of the end-to-end system. We are thus developing a congestion/flow control mechanism that can collect and leverage system-level knowledge to determine as precisely as possible those times when FOBS needs to step up or step down its level of aggression.

2.1 System-Aware FOBS

System-aware FOBS adds a *transfer-control agent* at the communication endpoints, where the agents exchange information pertaining to the state of the end-to-end system to cooperatively manage the behavior of the FOBS sender and receiver. The congestion/flow control mechanism is implemented at the sender side and functions as a simple state machine where the behavior of the FOBS sender is determined by the current state. Such behavior includes the rate at which data is placed onto the network, the maximum amount by which this rate can be changed, when to initiate a disk read, and whether to perform the disk read in the foreground or in the background. The state of the congestion/flow control mechanism is determined by the *aggregate state* of the components in the end-to-end system. Transitions between states are triggered by changes to the state of the end-to-end system (when such information is available). The lack of global information will also trigger changes in the state of the controller. The functionality of the

agent at the receiving end is primarily related to collecting and disseminating information about the current state of the data transfer and the state of the receiving host. This includes changes in CPU load, changes in the primary activity of the receiver (e.g., switching from pulling packets off of the network to performing a disk write), and the I/O bandwidth available to the application. Currently, four major system components are tracked and leveraged in the congestion/flow control algorithm:

- *The state of the CPUs at the communication endpoints*. As the load on the CPUs fluctuates, so to does the ability to pump messages into the network or to take messages off of the network. This is not an issue in a dedicated set of machines, but it becomes very important when the communication is taking place between shared computational clusters.
- *The state of the I/O subsystem.* The disk bandwidth available to the sender and receiver determines the amount of time required for a disk read or write to complete, which will be reflected in the availability of the sender/receiver to process additional data. Using background threads to perform disk I/O will allow disk and network I/O to be overlapped, but the progress of the network thread will be impeded because of reduced CPU cycles.
- *The state of the data transfer itself.* Operating at the application layer exposes the state of the *data transfer* to the control mechanisms, and this information can be used to optimize system-level resources. For example, the data receiver can directly communicate to the data sender when it begins/ends a disk I/O operation (during which time the sender should not be placing more data onto the network). If the disk I/O is operating in the background, the sender may need to reduce the sending rate until the operation is completed.
- *The current packet-loss profile.* The packet-loss profile is a bitmap that depicts those packets that have been successfully received and those packets still outstanding. Given the importance of packet-loss profiles to our approach, we next discuss them in more detail.

2.2 Packet-Loss Profiles

FOBS employs a simple application-level reliability mechanism. The data to be transferred is divided into a set of *chunks* (on the order of 100 MB and chosen based on extensive experimentation), the chunks are subdivided into *segments*, and the segments are divided into *packets*. Packets are 1,470 bytes (within the MTU of most transmission mediums), and a segment consists of 10,000 packets. The receiver maintains a bitmap for each segment depicting the received/not-received status of each packet in the segment. The 10,000-packet segment size requires a bitmap of 1,250 bytes, which was also chosen in consideration of the MTU size. These bitmaps are sent from the data receiver to the data sender and trigger (at a time determined by the congestion/control flow algorithm) a retransmission of the lost packets. These bitmaps are also sent to the visualization system and displayed on the screen.

What these bitmaps represent can be interpreted as a *packet-loss profile*. Such profiles not only provide a visual image of the degree of packet-loss but also present a view of the *distribution* of lost packets. Experimentation with data transfers across varying networks and communication hosts strongly suggests that packet-loss profiles change *as a function of the cause of the packet loss*. For example, the packet-loss profiles associated with a congested network appear to be significantly different from the profiles associated with an increased load on the CPU of the data receiver. We are conducting statistical analyses to verify these observations, and are deriving ways in which such information can be integrated into the congestion/flow control mechanism. A major focus of the demonstration will be the visualization of the packet-loss profiles.

3. Demonstration of System-Aware FOBS

The demonstration will visualize the following four important aspects of our prototype data transfer mechanism.

- Packet-loss signatures and how they vary depending on system-level conditions.
- End-to-end system state and how this corresponds to packet-loss signatures.
- Changes in sending rate as a function of system-level information.
- Progress of the transfer, including the retransmission of lost data.

The minimal requirement for this demonstration is a LCD projector. The visualization system is de-coupled from the data transfer system and receives its data over the Internet. Thus, the demonstration can be performed in real time using the computational resources available to the authors (which are currently limited to the continental U.S.), and the visualization information could be sent via the Internet to the conference location. The data can be buffered in such a way as to minimize jitter; and once the pipeline becomes full, the quality of the display should be quite good. Alternatively, the output from experiments conducted before the conference could be captured and played back at the conference, or the conference organizers could provide the authors with a (more) local account on which the data transfers could be performed in real time. It is also quite possible that the authors can establish accounts on more local computational platforms through Argonne National Laboratory. These issues can be worked out with the conference committee well in advance of the conference.

This research is most closely related to Topic 14 (*Routing and Communication in Interconnection Networks*) which specifically mentions lightweight communication protocols in the Topic description. It is also closely related to Topic 6 (*Grid Computing and Middleware Systems*) since FOBS is designed to operate in a Grid environment. It is also closely related to Topic 2 (*Performance Evaluation and Prediction*), which calls for papers related to performance data analysis and visualization. This will be the first demonstration of System-Aware FOBS.

References

[1] Allcock, W., Bester, J., Breshahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S.
Secure, Efficient Data Transport and Replica Management for High-Performance Data_Intensive Computing. In *Proceedings of IEEE Mass Storage Conference*, 2001.

[2] Automatic. *Automatic TCP Window Tuning and Applications. National Laboratory for Advanced Networking Research Web Page*
http://dast.nlanr.net/Projects/Autobuf_v1.0/autotcp.html

[3] Boyd, E.L., Brett, G., Hobby, R., Jun, J., Shih, C., Vedantham, R., and Zekauska, M. *E2E piPEline: End-to-End Performance Initiative Performance Environment System Architecture.* July, 2002.
http://e2epi.internet2.edu/e2epipe11.shtml

[4] Dickens, P.
A High Performance File Transfer Mechanism for Grid Computing. In *Proceedings of The 2002 Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA).* Las Vegas, Nevada, 2002.

[5] Dickens, P., and Gropp, B.
An Evaluation of Object-Based Data Transfers Across High Performance High Delay Networks. In *Proceedings of the 11th Conference on High Performance Distributed Computing*, Edinburgh, Scotland, 2002.

[6] Dickens, P., and Gropp, B.
Towards a System-Aware Data Transfer Mechanism. *In Preparation.*

[7] Dickens, P., Gropp, B., and Woodward, P.
High Performance Wide Area Data Transfers Over High Performance Networks. In *Proceedings of The 2002 International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems.*, 2002.

[8] Feng, W., and Tinnakornsrisuphap, P.
The Failure of TCP in High-Performance Computational Grids. In *Proceedings of Proceedings of Super Computing 2000 (SC2000).*

[9] He, E., Leigh, J., Yu, O., and DeFanti, T. Reliable Blast UDP : Predictable High Performance Bulk Data Transfer. In *Proceedings of Proceedings of the IEEE Cluster Computing*, Chicago, Illinois, September 2002.

[10] Sivakumar, H., Mazzucco, M., Zhang, Q., and Grossman, R.
Simple Available Bandwidth Utilization Library for High Speed Wide Area Networks. *Submitted to Journal of SuperComputing*.