

Java Parallel Secure Stream for Grid Computing ¹

Jie Chen, Walt Akers, Ying Chen and William Watson III
High Performance Computing Group
Thomas Jefferson National Accelerator Facility
12000, Jefferson Ave.
Newport News, Virginia 23606, U.S.A
Email: chen@jlab.org

Abstract

The emergence of high speed wide area networks makes grid computing a reality. However grid applications that need reliable data transfer still have difficulties to achieve optimal TCP performance due to network tuning of TCP window size to improve the bandwidth and to reduce latency on a high speed wide area network. This paper presents a pure Java package called **JPARSS** (Java Parallel Secure Stream) that divides data into partitions that are sent over several parallel Java streams simultaneously and allows Java or Web applications to achieve optimal TCP performance in a grid environment without the necessity of tuning the TCP window size. Several experimental results are provided to show that using parallel stream is more effective than tuning TCP window size. In addition X.509 certificate based single sign-on mechanism and SSL based connection establishment are integrated into this package. Finally a few applications using this package will be discussed.

keywords: Java Parallel Stream Security SSL X.509 Grid

1 Introduction

With the rapid growth of high performance networking technologies, grid computing has become reality among national laboratories and universities. Although grid computing applications usually demand high TCP[1] bandwidth in wide area networks, achieving the optimal bandwidth in practice can be difficult due to lack of automatic network tuning[2] that is usually required to achieve better network performance. To maximize TCP performance, the sender and the receiver of applications should adjust send/receive buffer(window) sizes no less than the capacity of the TCP pipe. This capacity usually is the product of transmission rate and round trip time (bandwidth-delay product). TCP performance problems arise when the bandwidth-delay product is large, where high performance wide area networks are deployed in grid computing environment, due to default smaller send/receive buffer sizes on most operating systems.

Until recently grid applications have had to adjust TCP window sizes at both ends of communication peers in order to achieve optimal TCP performance. The adjustment usually involves changing parameters for the TCP stack in the kernel by the system administration or tuning buffer sizes in the applications by setting socket options. This paper proposes an alternative Java solution that lets applications achieve optimal TCP bandwidth without tuning TCP window sizes.

Grid applications not only require efficient transfer of a large amount of data in wide area networks but also demand security and authentication services that enhance data integrity and enable data access control. In the past, user authentication required users to enter a password or pass-phrase to identify themselves to the server. This is inconvenient to users who initiate many short duration connections to the server. This paper describes a security structure, that is similar to security architecture used in globus[3] toolkit, allowing applications to transfer data either in encrypted form or in plain text form. In addition the security structure lets users authenticate only once to a server (e.g. when starting a grid application) and initiate additional connections to the server during a interval of time (single sign-on) without being further prompted for identification.

¹Work Supported by the Department of Energy, Contract DE-AC05-84ER40150.

In recent years, Web technologies, driven by the huge and rapidly growing electronic commerce industry, provided valuable components to construct Web portals[4] allowing users to access grid servers or services through Web browsers. Since Java servlet technology is becoming the overwhelming choice for Web server programming and Java applets are the standard way to deploy sophisticated user interfaces through Web browsers, non Java solutions[5][6] will be difficult to use in Java/Web applications.

There have been a few stand alone applications or libraries[5][6] that meets some, but not all, of the above requirements. This paper focuses on a Java networking toolkit called JPARSS (**J**ava **P**arallel **S**ecure **S**tream) that enables grid Java/Web applications to handle encrypted or plain data transfer with optimal bandwidth and to access data with proper authentication in wide area networks.

2 JPARSS

2.1 TCP Bandwidth Performance

Applications using JPARSS can achieve near optimal utilization of network bandwidth without the necessity of tuning TCP window sizes. The idea is to divide data into partitions that are sent over the network through several Java socket streams in parallel. This method sometimes is called network striping. Since the default socket buffer size is usually much smaller than the bandwidth-delay product in a grid environment, applications using a single socket will have sub-par performance in TCP bandwidth without tuning the buffer sizes. However, applications using multiple streams/sockets can overcome the limitations impacting single socket implementation.

JPARSS opens multiple socket streams between the sender and the receiver for applications. The application data are then partitioned into segments. The number of segments is equal to the number of streams. The segments are sent through streams in parallel by different threads and are then reassembled by the receiver. The final data are presented to applications as if they were transmitted through a single socket. Figure 1 illustrates why sending partitioned data through multiple socket streams can achieve near optimal bandwidth. In the figure the shaded areas represent socket buffer size and the large empty rectangles represent TCP pipe capacity (bandwidth*delay). Applications using a single socket stream will not fill the TCP pipe capacity, meanwhile applications using either a single socket stream with an optimal buffer size or multiple socket streams can fill the TCP pipe. Therefore applications using JPARSS can obtain near optimal TCP performance without adjusting TCP window size.

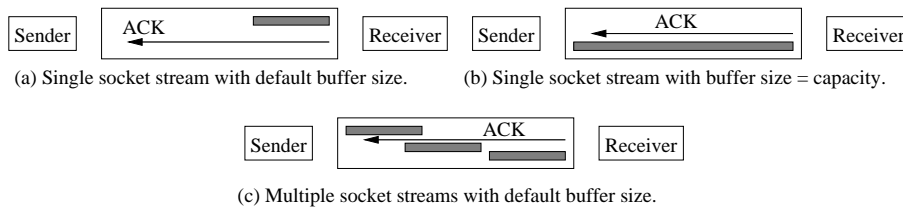


Figure 1: Single or multiple TCP socket streams with different buffer sizes.

Several experiments have been conducted between Jefferson Lab in Newport News (Virginia) and Florida International University in Miami (Florida). Both sites are connected to the Internet via OC12 connection. The link is not only limited by the 100 Mb/s Fast-Ethernet connection on host machines (PIII 700MHz), but also by numerous intermediate hops. Experimental host machines are running RedHat Linux 6.2 with kernel 2.2.16 and using the Java Virtual machine version 1.2. Figure 2 summarizes the TCP bandwidth performance results for various configurations along with the TCP bandwidth results from Iperf[7] which is a C utility using one socket stream. Specifically figure 2(A) shows that TCP bandwidth increases as the number of socket streams increases from 1 to 16 with fixed

TCP window size and fixed send buffer size Figure 2(B) highlights the improvements of TCP bandwidth using JPARSS over a traditional single socket stream C implementation with varying TCP window sizes and fixed TCP window sizes. Figure 2(C) illustrates the tremendous effect on bandwidth using multiple socket streams with varied send buffer sizes and fixed TCP window sizes.

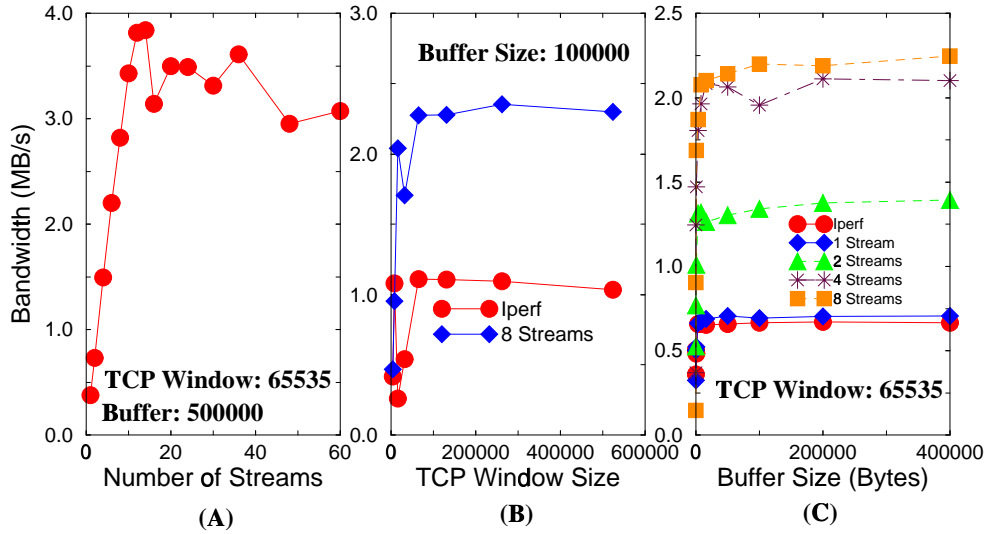


Figure 2: TCP bandwidth results for various JPARSS configurations.

2.2 Security Architecture

The JPARSS package contains a run-time option of security features that enables a subject (a user or a process) to be authenticated by a JPARSS server only once during a time interval using a temporary X.509 certificate signed by the subject whose permanent X.509 may be issued by any trusted certificate authority. This security feature allows the subject to connect to a JPARSS server securely via SSL[8] to transport data either in encrypted form or in plain text form, and maps the subject to a local user on the server host so that accounting or privileged operations can be carried out.

A JPARSS application first checks whether a temporary X.509 exists at an well known location on a host when the application starts. If the certificate exists and is valid, it will be used to authenticate the application to a potential server. Otherwise a new X.509 certificate, that will be valid for a short period of time (usually 24 hours) and has a non-encrypted private key, will be generated, signed by the user's permanent certificate and stored in the well known location. Once a JPARSS server receives the X.509 certificate, it will verify it and map the certificate onto a local user on the host. This scheme allows single sign-on, protects user credentials because of the short lived certificate, enables interoperability with local security solutions due to mapping from a certificate to a local user, and provides interoperability with secure Web servers because of the standard of X.509.

2.3 Java Implementation

Applications using JPARSS need only deal with three socket classes: *PClientSocket*, *PSocket* and *PServerSocket*. The first two are derived from *Java.net.Socket* and the other one is derived from *Java.net.ServerSocket*. The internal parallel streams can be obtained from *getInputStream* or *getOutputStream* of the derived socket classes. All I/O operations are handled asynchronously via threads to improve I/O efficiency.

Client connections are established without callbacks from servers to clients to avoid problems introduced when clients are behind firewalls.

2.4 Applications

Several applications using JPARSS are implemented and deployed in the Jefferson Lab/MIT Lattice QCD grid environment. One application is a simple command line file transfer utility that imitates *scp* syntax but without requiring the user to enter a password each time. Another application is an applet allowing the user to transfer data from one site to another through simple mouse clicks.

3 Conclusions

This paper presents a pure Java package called JPARSS. It helps grid applications to achieve near optimal TCP bandwidth without tuning TCP window sizes, provides simple Java classes as building blocks for rapid development using applets or servlets, and offers a single sign-on mechanism and secure connection between communication peers. Finally the source code and classes can be obtained through <http://www.jlab.org/hpc/software/jparss.tar.gz>

References

- [1] J. Postel, "Transmission Control Protocol", Request for Comments 0793, Sep. 1981.
- [2] Jeffery Semke, Jamshid Mahdavi and Matthew Mathis. Automatic TCP Buffer Tuning. *ACM SIGCOMM'98/Computer Communication Review*, Vol. 28, No.4, Oct. 1998.
- [3] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. In *Proceedings 5th ACM Conference on Computer and Communications Security Conference*. pg. 83-92, 1998.
- [4] G. von Laszewski, I. Foster. In *Proceedings of the Workshop Distributed Computing on the Web*. 1999.
- [5] Ann Chervenak, Bill Alcock, Joe Bester, John Bresnahan, Ian Foster, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, Steven Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In *Proceedings of 18th IEEE Symposium on Mass Storage Systems*. San Diego, California. April, 2001.
- [6] H. Sivakumar, S. Bailey and R. L. Grossman. P.Sockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In *Proceedings of SuperComputing 2000*. Dallas, Texas. Nov. 2000.
- [7] Ajay Tirumala, Jim Ferguson. <http://dast.nlanr.net/Projects/Iperf/index.html>, May 2001.
- [8] K. Hickman and T. Elgamal. The SSL protocol. Internet draft, Netscape Communication Corp., June 1995. Version 3.0.