MAXIMISING SECURE DATA THROUGHPUT WITH VERNE GLOBAL

VERNE GLOBAL

VERNEGLOBAL.COM

EXECUTIVE SUMMARY:

Iceland is well connected with existing submarine fibre-optic cables that are routed via optimal paths resulting in very reasonable latency characteristics. The combination of this measured latency with careful manipulation of computer system parameters and selection of data transfer protocol will allow for extremely high throughput over the distance between your on-premise network and the Verne Global campus.

BACKGROUND

Given that servers have historically been either on-premises or collocated nearby, enterprise has rarely been exposed to the throughput-limiting factor of the *bandwidth-delay product,* which is experienced when servers are connected by high capacity connections that are rather long and therefore latent.

In simplest form, connection-oriented protocols such as TCP utilise a sequence number to identify each segment of data transmitted. Upon successful receipt of the sequence, the receiver sends an acknowledgement. Assuming the maximum segment size is 1460 bytes, and a latency of 10ms, the theoretical maximum throughput is 1460 bytes every 20ms.

To solve this problem, several features are implemented into the TCP protocol; namely *Slow Start, Selective ACK,* and *Window Scaling.* These additions to the TCP protocol allow senders to send more than one segment before expecting acknowledgement, while preserving a method for increasing or decreasing confidence in the underlying transport, and thus maintaining a reasonable level of throughput under all conditions.

Applications may further constrain throughput, however. SFTP and SCP, for example, are generally subject to a static 64KB window size imposed by the underlying SSH protocol implementation. Therefore, even with an adequately tuned TCP implementation, a pair of hosts transferring data via SFTP or SCP will be limited to a throughput equal to the product of this 64KB window size and the round-trip latency between the hosts.

Fortunately, alternative SSH implementations that remove this window size constraint are available, and where necessary, alternatives to SSH are available for high-volume and secure data transfer.

SSH BASED FILE TRANSFER LIMITATIONS AND SOLUTIONS

SSH implements a layer of flow-control above and beyond that of TCP, and unfortunately, the buffer sizes are statically defined. Realising this, the Advanced Networking division of the Pittsburgh Supercomputing Center has developed patches to augment SSH's internal flow-control to scale with that of the underlying host's TCP implementation and parameters.[i]

DATA TRANSFER THROUGHPUT

Transfer Method	Efficiency[ii]
OpenSSH OpenSSH[iii]	11.5%
iperf[iv]	63.5%
OpenSSH + HPN-SSH[v]	60.2%
OpenSSH + HPN-SSH[vi] (32 MB window)	76.6%

Table 1 - Average Data Transfer Throughput[vii]

As shown in Table 1, average throughput can be increased significantly when the application layer makes use of TCP windowing, and careful adjustment of interface MTU and window size may provide even greater efficiency. Moreover, encryption of data payload can be achieved without impact to throughput and without significant hardware expense.[vii]

CONCLUSION

Enterprises need not be concerned about latency or throughput potential when considering the Verne Global campus as a home for their high performance computing assets. Verne Global stands ready to assist in procurement of test connectivity and optimisation of server hardware and operating systems to demonstrate Iceland's compatibility with your needs and requirements, applications, and bottom line.

APPENDIX

Software Availability

Verne Global has made available a repository of the OpenSSH source code with the most recent HPN patches applied. Also provided are instructions for building the software on Ubuntu- or RedHat-based Linux distributions. The repository can be found at https://github.com/VerneGlobal/openssh-portable and instructions can be found at https://github.com/VerneGlobal/openssh-portable/wiki.

Test Transfer File Creation

Files created for use during testing were created with dd and output from the Linux urandom device. An example, as would be performed to create a 1000-megabyte file:

```
% dd if=/dev/urandom of=test-file bs=1048576 count=1000
```

iperf Test Method

iperf was built from source and invoked on both server and client as follows:

Server

```
% iperf –s –w 16M
```

Client

```
% iperf -c server -t 90 -i 1 -w 16M
```

OpenSSH + HPN-SSH Test Method

OpenSSH was extracted from source, patched with HPN-SSH patches, compiled, and invoked on both server and client as follows:

Server

```
% sudo sbin/sshd -d -p 2022 -o TCPRcvBufPoll=yes \
 –o HPNBufferSize=16777216          (or 33554432 in 32 MB Window Test)
```

Client

```
% bin/sftp -o TCPRcvBufPoll=yes -P 2022 root@server
root@server's password:
Connected to west.
sftp> lcd /tmp
sftp> cd /tmp
sftp> put <file>
```

Verification of file consistency

To verify that both source and destination files were identical, their MD5 checksum was obtained on both source machine prior to transfer and on destination machine post transfer as follows:

```
% md5sum <file>
62f6a283859a16221797ea5dee4963f8  <file>
```

Test Host Kernel TCP Options

The following options were set via `/etc/sysctl.conf` on the CentOS 5.6 Linux hosts which were used for testing throughput:

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

For certain tests, the window size was set to 32MB as follows:

```
net.core.rmem_max = 33554432
net.core.wmem_max = 33554432
net.ipv4.tcp_rmem = 4096 87380 33554432
net.ipv4.tcp_wmem = 4096 65536 33554432
```

Further information on TCP performance tuning is given in an archived treatment by the Pittsburgh Supercomputing Center.

[i] http://www.psc.edu/networking/projects/hpn-ssh/

[ii] Efficiency is calculated assuming an IP MTU of 1500 bytes, TCP Maximum Segment Size of 1460 bytes, and Ethernet frame size of 1538 bytes inclusive of interframe gap. For a Gigabit Ethernet, the frame rate for the largest supported TCP payload is 81,274 frames / second, rendering a theoretical maximum TCP throughput of ~ 949.2 Mbps. See http://www.cisco.com/c/en/us/about/security-center/network-performance-metrics.html for a discussion of Ethernet throughput rates.

[iii] This implementation of OpenSSH is as delivered with CentOS 5.6; specifically OpenSSH_4.3p2, OpenSSL 0.9.8e-fips-rhel5 01 Jul 2008. The measurement was obtained through transfer of a 1000 MB file; details of file creation found in the Appendix.

[iv] iperf is a tool developed by the United States National Laboratory for Applied Network Research, and can be downloaded from http://sourceforge.net/projects/iperf/. This specific test was performed for 90 seconds to determine average throughput.

[v] This test was performed with vanilla OpenSSH 5.9p1 as retrieved from openbsd.org and augmented by the applicable HPN-SSH patch set as retrieved from http://www.psc.edu/networking/projects/hpn-ssh/. The measurement was obtained through transfer of a 3000 MB file.

[vi] Ibid. TCP send and receive windows were adjusted to 32 MB on client and server and application invocations updated accordingly.

[vii] In this test, client and server were each connected with Gigabit Ethernet with 1500 Byte MTU and with approximately 45ms one-way latency between them.

[viii] HPN-SSH achieves approximate parity with unencrypted data transfer throughput as shown in the iperf test by utilising a multithreaded encryption algorithm.