SLAC-TN-15-002

An Overview of high-performance parallel Big Data transfers over multiple network channels with Transport Layer Security (TLS) and TLS plus Perfect Forward Secrecy (PFS)

04/25/2015

Chin Fang

R. "Les" A. Cottrell

Abstract -- This Technical Note provides an overview of high-performance parallel Big Data transfers with and without encryption for data in-transit over multiple network channels. It shows that with the parallel approach, it is feasible to carry out high-performance parallel "encrypted" Big Data transfers without serious impact to throughput. But other impacts, e.g. the energy-consumption part should be investigated. It also explains our rationales of using a statistics-based approach for gaining understanding from test results and for improving the system.

The presentation is of high-level nature. Nevertheless, at the end we will pose some questions and identify potentially fruitful directions for future work.

SLAC National Accelerator Laboratory, Stanford University, Stanford, CA 94309

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515 and Citrix Systems, Inc.

An Overview of high-performance parallel Big Data transfers over multiple network channels with Transport Layer Security (TLS) and TLS plus Perfect Forward Secrecy (PFS)

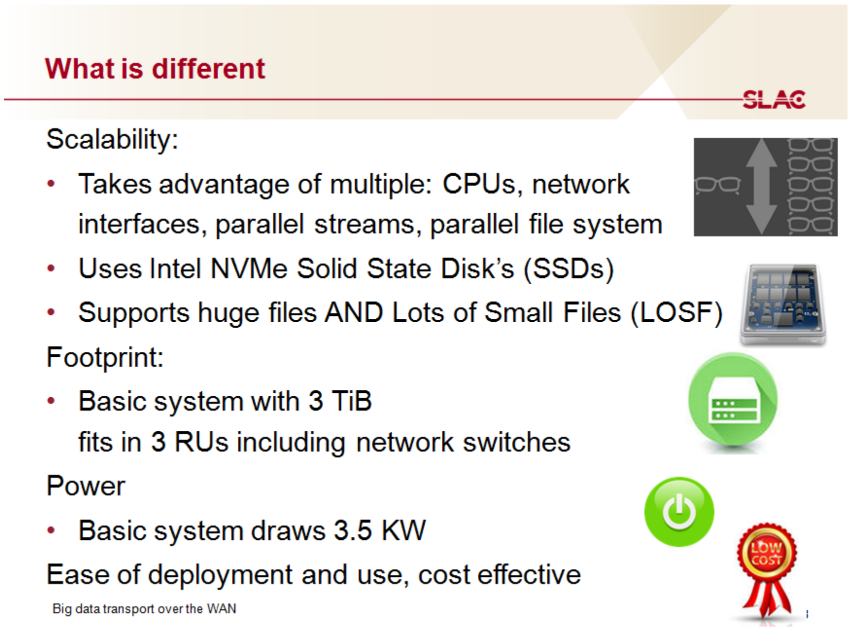
Chin Fang, Ph.D. Founder, Zettar Inc. <fangchin@zettar.com>
R. "Les" A. Cottrell, Ph.D. SLAC National Accelerator Laboratory <cottrell@slac.stanford.edu>

Apr 25, 2015

Table of Contents

Introduction

This is the 2nd installment in a series of SLAC Technical Notes (TN) focusing on high-performance parallel Big Data transfers over multiple network channels. In the spirit of SLAC TNs, this series intends to share our experience in coming up with practical, simple, effective but rarely seen engineering solutions to the various challenges that have been and will be encountered in our quest of creating an ultra-high performance parallel Big Data transfer system. It is meant to be easy to deploy, to use, space-and-energy-efficient, cost-effective, and applicable to Big Data transfers over both the LAN and WAN environment. The following figure summarizes how the system differs from others available today:



What is different

Scalability:

- Takes advantage of multiple: CPUs, network interfaces, parallel streams, parallel file system
- Uses Intel NVMe Solid State Disk's (SSDs)
- Supports huge files AND Lots of Small Files (LOSF)

Footprint:

- Basic system with 3 TiB fits in 3 RUs including network switches

Power

- Basic system draws 3.5 KW

Ease of deployment and use, cost effective

SLAC
LOW COST
Big data transport over the WAN

FIG. 1 The main differentiations of the data transfer system (hardware + software) that SLAC and Zettar are working on.

The major goals of this Technical Note

- To provide an overview of high-performance parallel Big Data transfers with and without encryption for data in-transit over multiple network channels, where the term
  - "High-performance" means a data transfer rate of 10Gbps or higher
  - "Parallel" is used in the sense of HPC parallel computing utilizing a cluster of servers
  - "Big Data" means a data set sized anywhere from a few hundred GiB[1]s to well-beyond PiBs.
- To show that with the parallel approach, it is feasible to carry out high-performance parallel "encrypted" Big Data transfers without serious impact to throughput. But other impacts, e.g. the energy-consumption part should be investigated
- To explain our rationales of using a statistics-based approach for gaining understanding from test results and for improving the system

As befitting the title of this TN, the presentation is of high-level nature. Nevertheless, at the end we will pose some questions and identify potentially fruitful directions for future work.

Zettar data transfer system design and software overview

Zettar creates a US patent-pending, scale-out, ultra-fast, Big Data *parallel* data transfer software. The software is deployed on a cluster of commodity servers on both the sending and receiving sides. It leverages all the available aggregated **1**) storage IOPS, **2**) compute power, and **3**) network bandwidth without resorting to low-level link aggregation[2]**,** in the OS or with

[1] In the networking industry, it is typical to measure bandwidth using power of ten, for example, 1Kbps = 10^3 bps, 1Mbps = 10^6 bps. The same practice prevails in the storage industry too, thus 1GB = 10^9 bytes for example. Nevertheless, elsewhere in computing, International Electrotechnical Commision (IEC) convention is used, so does this report.

[2] Link Aggregation from Wikipedia, see http://en.wikipedia.org/wiki/Link_aggregation

hardware, for low OPEX, manageability, and high scalability. The attainable Big Data transfer throughput scales according to the three types of aforementioned computing resources *listed in order of descending importance*[3]. In practice, Zettar software marshals such resources and provides highly scalable Big Data transfer throughput, with the potential of addressing the challenge of transferring ever-growing Big Data.

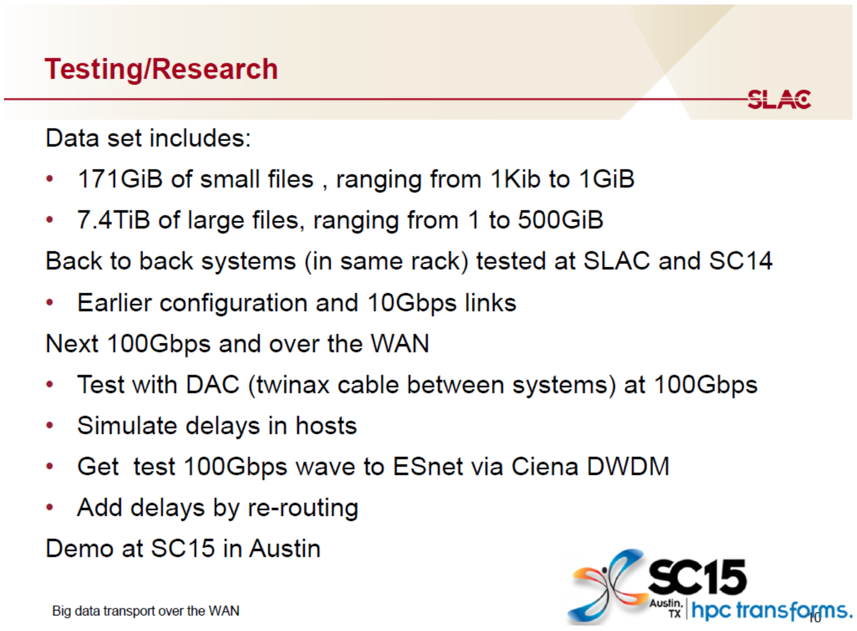
In parallel with the collaboration with the SLAC National Accelerator Laboratory, Zettar also works with a tier-1 media and entertainment (M&E) studio, helping it meet two emerging challenges:

1. Inability to move Big Data sets timely on campus and between data centers.
2. Data breaches and lack of privacy for data in transit.

The rapid and continuous growth of Big Data will put even more stresses on the M&E studios. Today's data transfer solutions have not been designed to handle this level of data volume and size. Nevertheless, Zettar's `zx` software has already achieved the following over 3 *independent* (i.e. *not* link-aggregated) 10Gbps network links (see **FIG. 3**) in Q1, 2015, with more performance boosts to come:

1. 19+ Gbps transfer rate using a single 200GiB large file test data set.
2. 17+ Gbps transfer rate using a 391.6GiB Lots of Small Files (LOSF) test data set, consisting of 200,000 files, with sizes ranging from 8KiB to 4MiB.
3. Strong data in-transit security with hard to predict dynamic/random data transfer and encryption patterns. This is because Zettar software is architected to use multiple fast data links (if available and possibly independent) simultaneously instead of a single link and transfers a data sets in slices sent in random order.

The following figure summarizes the testing and research that have been done and planned. The next few sections will describe the test data and the data transfer system in more detail.



Testing/Research

SLAC

Data set includes:
- 171GiB of small files , ranging from 1Kib to 1GiB
- 7.4TiB of large files, ranging from 1 to 500GiB

Back to back systems (in same rack) tested at SLAC and SC14
- Earlier configuration and 10Gbps links

Next 100Gbps and over the WAN
- Test with DAC (twinax cable between systems) at 100Gbps
- Simulate delays in hosts
- Get test 100Gbps wave to ESnet via Ciena DWDM
- Add delays by re-routing

Demo at SC15 in Austin

Big data transport over the WAN

SC15 Austin, TX hpc transforms.

10

FIG. 2 Testing and research that SLAC and Zettar are working on throughout 2015

3 Please see **SLAC-TN-15-001** for more background

The data transfer cluster overview

The physical test setup is shown in **FIG. 3** below. Zettar and SLAC have been using the setup since Supercomputer 2014[4] until now. For the genesis of the design, please see **SLAC-TN-15-001**. In this note, the setup is called the "basic system" and is to be upgraded in May – June timeframe 2015 with newer and more powerful hardware. All data sets are initially generated programmatically according to the specifications from said M&E studio and stored on the bastion host. As the two clusters have only ~3*TiB* storage space/cluster, each test data set is ingested from the bastion host into the top cluster in turn for testing. All files consist of fully random content and thus are not compressible – this is typical for media files.

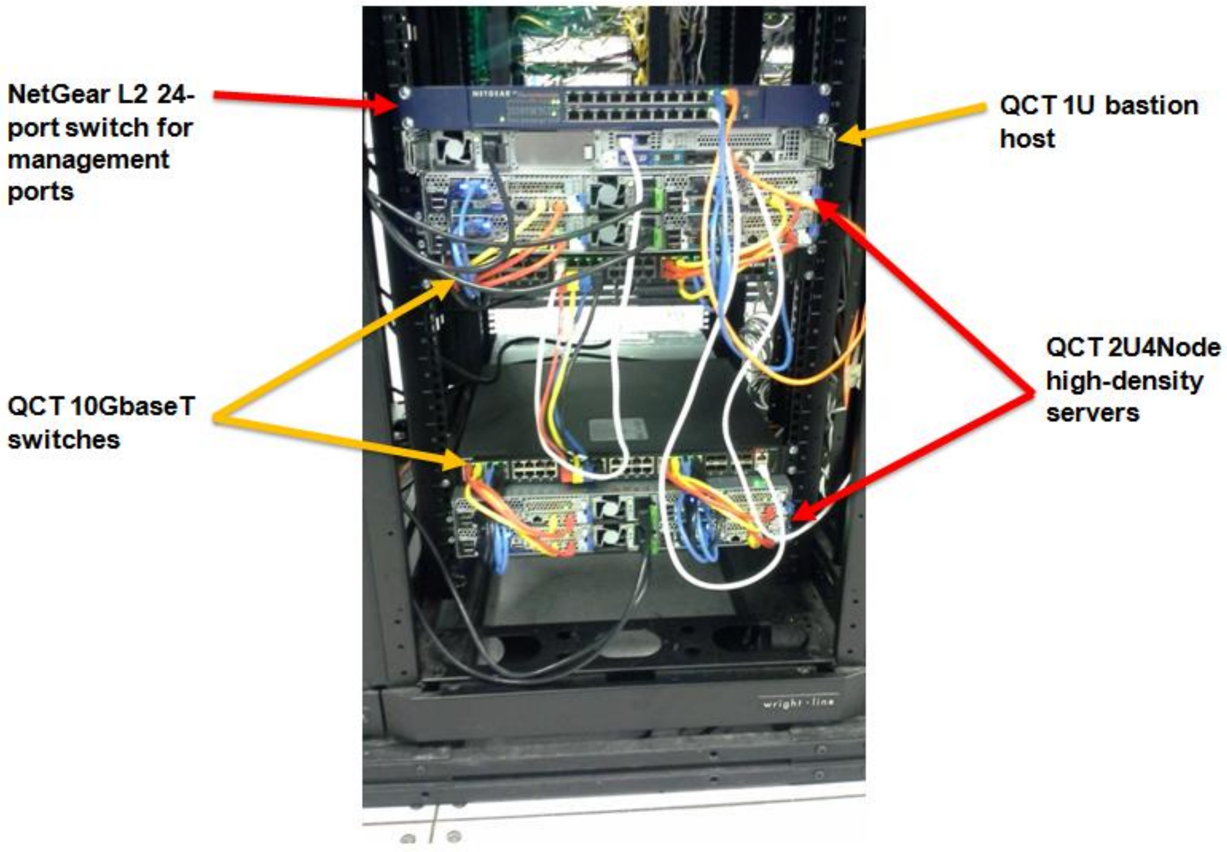

NetGear L2 24-port switch for management ports
QCT 1U bastion host
QCT 10GbaseT switches
QCT 2U4Node high-density servers

FIG. 3 The test setup consists of two QCT (Quanta Cloud Technology LLC) S810-X52L high-density 2U-4 Node servers (*thus each S810-X52L is a 4-node cluster*), two QCT T3040-LY3 10GbaseT switches. A QCT S210-X12RS 1U server acts as the bastion host and cold test data storage. The NetGear 24 port L2 switch connects the management ports of selected devices. Each node in the two QCT S810-X52L has 4 10GBaseT ports and an 800GB Intel P3700 NVMe Gen3 PCIe SSD card[5]. The top cluster and the bottom cluster are connected using three CAT6 patch cables (**yellow**, **blue**, and **black**) - 10Gbps each or 30Gbps collectively - for Big Data transfers.

The Zettar software overview

The following figure illustrates the software's architecture and how it is deployed on two clusters - one sending, one

[4] SuperComputing 2014, see http://sc14.supercomputing.org/

[5] Intel SSD DC P3700 Series Specifications, see http://www.intel.com/content/www/us/en/solid-state-drives/ssd-dc-p3700-spec.html

receiving. In addition, it also points out the need of using a tiered-storage concept[6] to keep the Big Data transfer performance high, but at the same time to load/store data from an enterprise storage pool effectively.

The Zettar software, `zx`, is implemented in the C++11 programming language, and comes as a single 64bit executable binary. It targets modern Linux operating system, e.g. RHEL[7] 7 and its free rebuilds such as CentOS[8] 7 and Scientific Linux[9] 7 and Ubuntu[10] 14.04 LTS. By default the software is provided as a package, in RPM for RHEL and its free-rebuilds and DEB for Debian and compatibles. Zettar `zx` has a multithreaded design with extensive use of asynchronous processing so as to leverage the available computing resources as efficiently as possible. It employs elements from the peer-to-peer technology to achieve high-availability and self-organizing distributed processing of large data sets collaboratively using a cluster. The peers in a cluster work together to slice a data set, regardless of data types (numerous small files, mix-sized files, or large files) into an appropriate number of logical "pieces", to be transferred to a destination, under the guidance of a "data transfer controller" peer. The deployment of the software can be made simple via the employment of a configuration management system[11] such as ansible[12].

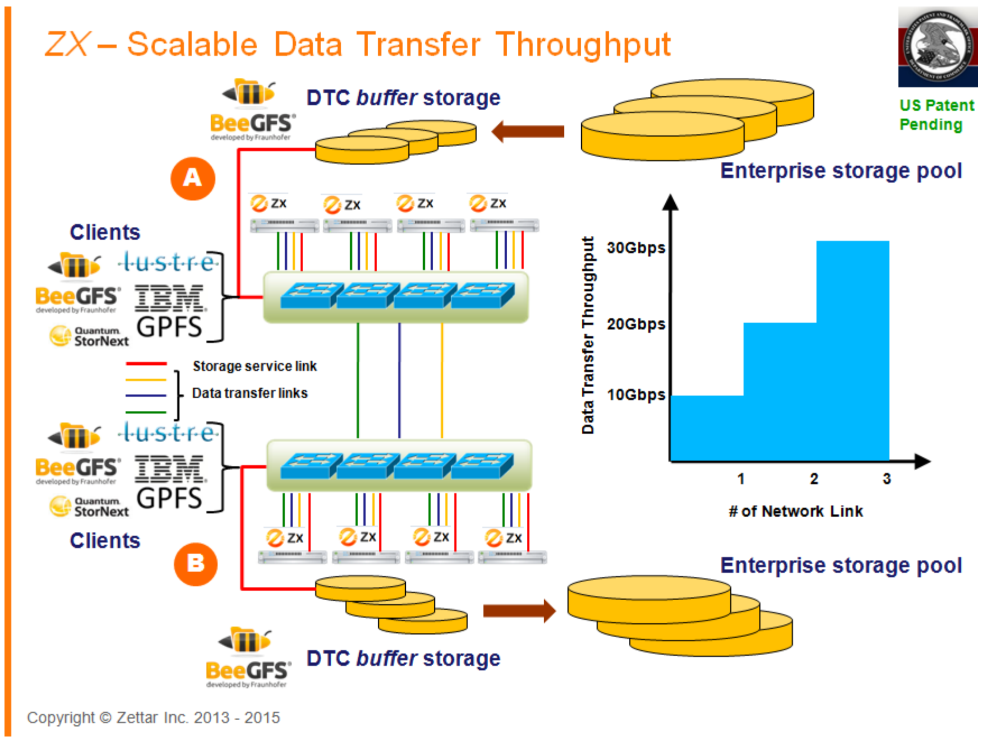


FIG. 4 As illustrated, Zettar `zx` software is deployed on a four node cluster on both the sending and receiving sides. Each cluster is termed a "data transfer cluster" (DTC). Each node of a data transfer cluster also runs the BeeGFS[13] parallel file system, which aggregates the capacity and IOPS of one or more NVMe SSD devices installed in each node to form a "DTC buffer storage". Said storage is not meant to be a custodial storage

[6] Tiered Storage, see http://searchstorage.techtarget.com/definition/tiered-storage
[7] RHEL, see https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/
[8] CentOS, see https://www.centos.org
[9] Scientific Linux, see https://www.scientificlinux.org
[10] Ubuntu, see http://www.ubuntu.com
[11] Configuration management system, see http://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software
[12] Ansible, see http://www.ansible.com/home
[13] BeeGFS, see http://www.beegfs.com/content/

for enterprise data. Thus, high availability is not essential, although if needed, can be obtained via BeeGFS' data replication feature. Its foremost purpose is to provide the extremely high IOPS that is required for saturating multiple high-speed network links. See SLAC-TN-15-001, section "**A key to high-speed data transfers – high-performance storage**, and **FIG. 2 – 4** therein.

At the beginning of this section, the need for using a tiered-storage concept is introduced. Said concept actually is the application of the decades-old memory hierarchy as often discussed in computer architecture, but applied to storage. Please see the following figure:

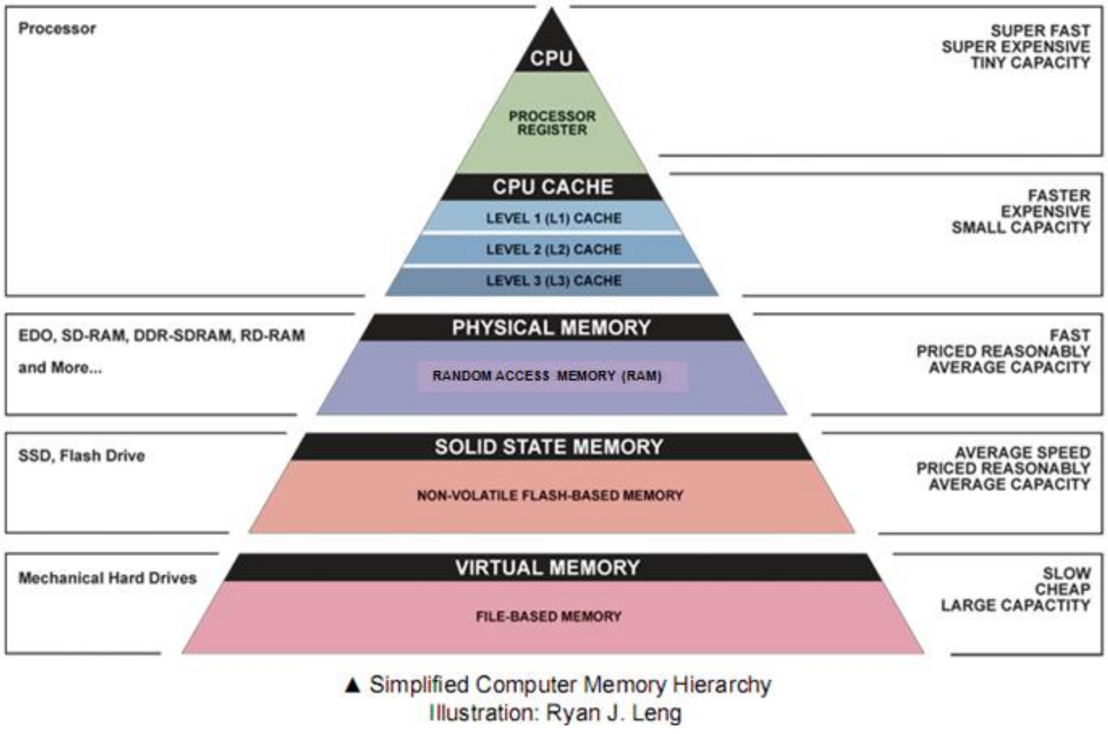


FIG. 5 A typical memory hierarchy diagram often seen in computer architecture literature, e.g. Computer Architecture, Fifth Edition: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design) **– September 30, 2011 by John L. Hennessy and David A. Patterson**[14]

In short, in a data processing chain, if there is a set of processing components (e.g. CPUs) that are much faster than the slowest components (e.g. conventional HDDs) in said chain, then it's often made more efficient and higher-performing by the employment of components (e.g. SATA SSDs) with gradually increasing processing speed but with less and less capacity into the chain, forming a pyramid or hierarchy, more or less as the one shown in **FIG. 5** above. Furthermore, it should be noted that for memory and storage, a layer can be optimized for either capacity or performance, but rarely both. Also, as discussed in **SLAC-TN-15-001**, high-speed Big Data transfers demands *very high* IOPS which are often unavailable from most enterprise storage pools. Therefore, a tiered approach is becoming essential as higher and higher transfer throughput is required. In the short term, a two-tiered approach may still be applicable as illustrated in **FIG. 4**: each DTC node is also a client of a common parallel file system, e.g. IBM GPFS[15], Intel Lustre[16], BeeGFS, and Quantum StorNext[17]. Then, data in the

[14] Computer Architecture, Fifth Edition: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design) – September 30, 2011 by John L. Hennessy and David A. Patterson, see http://store.elsevier.com/product.jsp?isbn=9780123838728

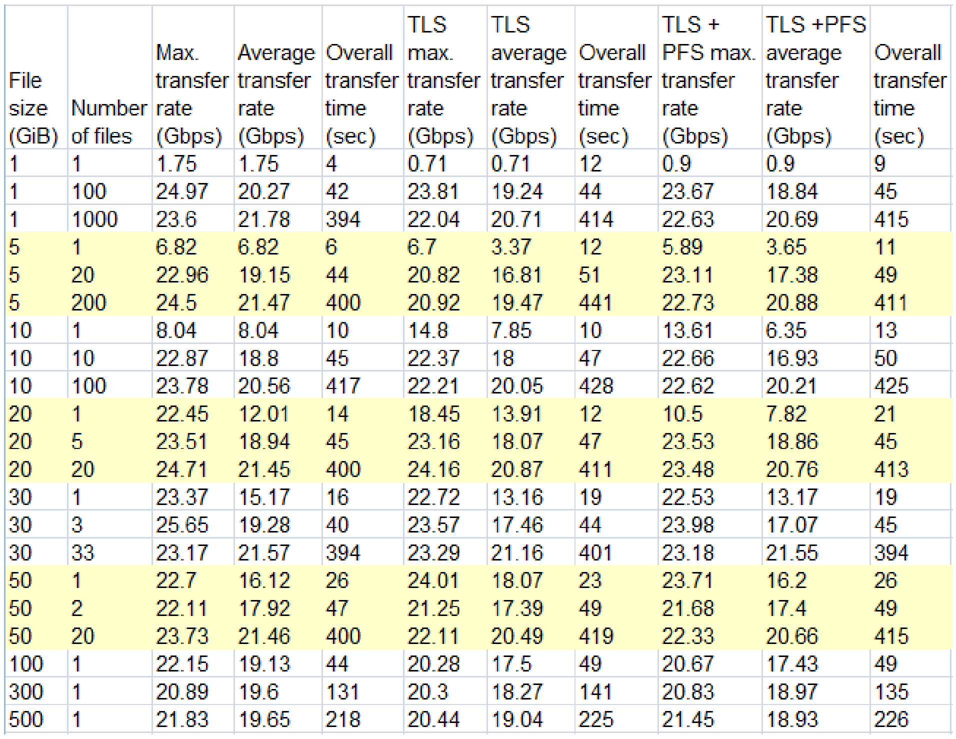
[15] IBM General parallel File System, see http://www-03.ibm.com/software/products/en/software

[16] Intel Solutions for Lustre Software, see http://www.intel.com/content/www/us/en/software/intel-enterprise-edition-for-lustre-software.html

pool is ingested via a lazy-evaluation[18] alike approach collaboratively carried out by all DTC nodes. The cost benefit of such an approach is obvious as can be seen by the simple observation that if we wanted to use the Zettar system to transfer a PiB and all the data was to be saved in the DTC it would take about 667 Rack Units and be very expensive. But a tiered approach can still provide excellent performance at much lower cost – more space and energy efficiency too.

Test data specifications

The aforementioned tier-1 M&E studio provided Zettar with test data specifications for both large files and lots of small files. Nevertheless, with the goals of this TN, only large files are employed. The specifications, together with test results, are given in the following table:



File size (GiB)	Number of files	Max. transfer rate (Gbps)	Average transfer rate (Gbps)	Overall transfer time (sec)	TLS max. transfer rate (Gbps)	TLS average transfer rate (Gbps)	Overall transfer time (sec)	TLS + PFS max. transfer rate (Gbps)	TLS +PFS average transfer rate (Gbps)	Overall transfer time (sec)
1	1	1.75	1.75	4	0.71	0.71	12	0.9	0.9	9
1	100	24.97	20.27	42	23.81	19.24	44	23.67	18.84	45
1	1000	23.6	21.78	394	22.04	20.71	414	22.63	20.69	415
5	1	6.82	6.82	6	6.7	3.37	12	5.89	3.65	11
5	20	22.96	19.15	44	20.82	16.81	51	23.11	17.38	49
5	200	24.5	21.47	400	20.92	19.47	441	22.73	20.88	411
10	1	8.04	8.04	10	14.8	7.85	10	13.61	6.35	13
10	10	22.87	18.8	45	22.37	18	47	22.66	16.93	50
10	100	23.78	20.56	417	22.21	20.05	428	22.62	20.21	425
20	1	22.45	12.01	14	18.45	13.91	12	10.5	7.82	21
20	5	23.51	18.94	45	23.16	18.07	47	23.53	18.86	45
20	20	24.71	21.45	400	24.16	20.87	411	23.48	20.76	413
30	1	23.37	15.17	16	22.72	13.16	19	22.53	13.17	19
30	3	25.65	19.28	40	23.57	17.46	44	23.98	17.07	45
30	33	23.17	21.57	394	23.29	21.16	401	23.18	21.55	394
50	1	22.7	16.12	26	24.01	18.07	23	23.71	16.2	26
50	2	22.11	17.92	47	21.25	17.39	49	21.68	17.4	49
50	20	23.73	21.46	400	22.11	20.49	419	22.33	20.66	415
100	1	22.15	19.13	44	20.28	17.5	49	20.67	17.43	49
300	1	20.89	19.6	131	20.3	18.27	141	20.83	18.97	135
500	1	21.83	19.65	218	20.44	19.04	225	21.45	18.93	226

FIG. 6 Test data specification (large files only) and test results

The above tabular data are summarized graphically in the **Graphs** section below.

Test objectives, methodology, and test results

As far as the authors are aware of, all published high-speed data transfer tests and demonstrations have been done with specially constructed test data sets. In addition, none of the reviewed tests were carried out using encryption, for example, Results From NASA High End Computing (HEC) WAN File Transfer Experiments/Demonstrations Super Computing 2013 (SC13), ESnet, Caltech and NERSC Team Up to Send Data Screaming at 90 Gbps. Furthermore, the respective setup employed by these trials is quite complex, bulky and costly.

[17] Quantum Scale-Out-Storage, see http://www.quantum.com/products/scale-out-storage/index.aspx
[18] Lazy Evaluation, see http://en.wikipedia.org/wiki/Lazy_evaluation

Objectives

Thus the objective of the set of tests is to use not only realistic data as used in actual daily operation of a tier-1 M&E studio, but also with encryption and a simple and compact setup. To help visualize the results, several plots are provided. *Please note that these results do not represent absolute Zettar software performance ceilings* - they are what are currently attainable with the computing resources available from the test set up shown in **FIG. 3**. Furthermore, all test results are obtained using identical settings for Zettar `zx`, and without any operating system, file system, and networking stack tuning.

Methodology

Zettar `zx` software has an embedded Web server and RESTful APIs for management. Thus, all tests conducted for preparing this TN have been carried out via a test harness (implemented in the Python programming language) that calls `zx`'s RESTful APIs. The test harness iterates through all test data sets, carries out test transfers from the top cluster to the bottom cluster as shown in **FIG. 3**, and performs data transfer runs first without encryption, then with Transport Layer Security (TLS), and then with TLS and Perfect Forward Secrecy (PFS). It also logs all test results, which are analyzed further and graphed whenever appropriate.

Before proceeding, it might be appropriate to briefly introduce TLS and PFS:

TLS refers to Transport Layer Security[19], a cryptographic protocol designed to provide communications security over a computer network. It uses X.509 certificates and hence asymmetric cryptography to authenticate the counterparty with whom it is communicating, and to negotiate a symmetric key. This session key is then used to encrypt data flowing between the parties.

PFS refers to Perfect Forward Security[20], a property of key-agreement protocols ensuring that a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future.

In addition it should be noted:

- We didn't use CPU-assisted hardware encryption acceleration
- The file systems (local: XFS, distributed: BeeGFS) are not highly tuned
- We intentionally limited the number of cores utilized by the Zettar `zx` software (see below)
- The networking stack is not tuned (we didn't even turn on the TCP fast start for example; window sizes are left to auto-tune, not manually pre-set)

So, there should be some more reserves for better performance. Also, there are many topics to look into and investigate.

Referring to **FIG. 3** again, the four nodes in each system each have 24 hyperthread[21]s (i.e. logical processors, two per physical core). Only the 12 hyperthreads are designated to engage in the task. The three figures below are for different encryption requests. They show images of the CPU utilization by each of the 24 logical cores for each of the 4 nodes in the cluster. Referring to **FIG. 6 – 8** below, it is evident that for each node, cores 1 through 6 and cores 13 through 18 are the ones mainly in use with others essentially idle.

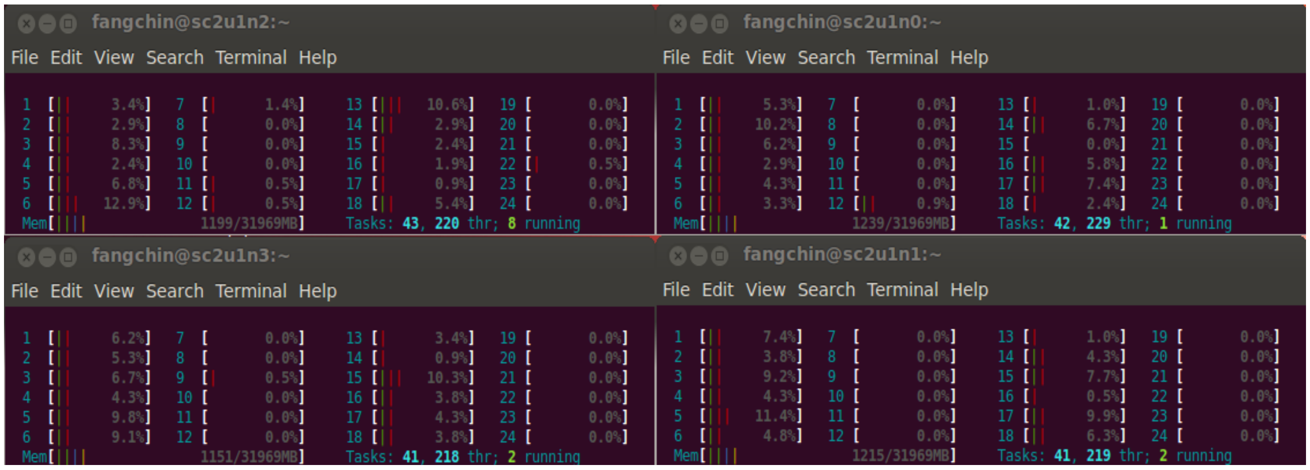
Results

Presented in the following are a few screenshots, a min-max summary, some graphs and notes.

[19] Transport Layer Security from Wikipedia, see http://en.wikipedia.org/wiki/Transport_Layer_Security
[20] Perfect forward secrecy, see http://en.wikipedia.org/wiki/Forward_secrecy
[21] Hyperthread, see http://en.wikipedia.org/wiki/Hyper-threading

Screenshots

The following screen shots are from using htop[22] to measure utilization. The 4 nodes are represented two in the top row, two in the row below, mirroring their placement in a high-density server as shown in **FIG. 3**. Each node actually provides overall 24 hyperthreads. We decided to use only the 12 (i.e. half) of them – the number of actual physical cores/node. This is accomplished by specifying `thread=12` in each node's `/etc/zettar/zx/zx.conf`, the configuration file of the `zx` software running on the node. Each node runs CentOS 7.0 x86_64 as OS.


fangchin@sc2u1n2:~
File Edit View Search Terminal Help
fangchin@sc2u1n0:~
File Edit View Search Terminal Help
fangchin@sc2u1n3:~
File Edit View Search Terminal Help
fangchin@sc2u1n1:~
File Edit View Search Terminal Help

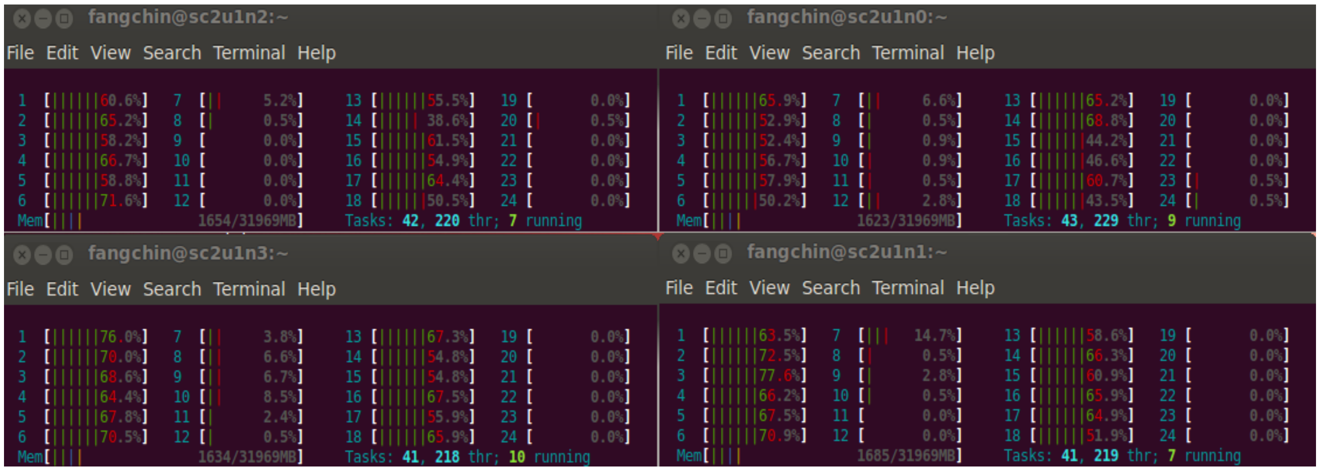
```
  1  [||      3.4%]   7  [|       1.4%]    13 [|||   10.6%]   19 [        0.0%]
  2  [||      2.9%]   8  [        0.0%]    14 [||     2.9%]   20 [        0.0%]
  3  [||      8.3%]   9  [        0.0%]    15 [|      2.4%]   21 [        0.0%]
  4  [||      2.4%]   10 [        0.0%]    16 [|      1.9%]   22 [|       0.5%]
  5  [||      6.8%]   11 [|       0.5%]    17 [|      0.9%]   23 [        0.0%]
  6  [|||    12.9%]   12 [|       0.5%]    18 [||     5.4%]   24 [        0.0%]
  Mem[||||             1199/31969MB]    Tasks: 43, 220 thr; 8 running
```

```
  1  [||      5.3%]   7  [        0.0%]    13 [|      1.0%]   19 [        0.0%]
  2  [||     10.2%]   8  [        0.0%]    14 [||     6.7%]   20 [        0.0%]
  3  [||      6.2%]   9  [        0.0%]    15 [       0.0%]   21 [        0.0%]
  4  [||      2.9%]   10 [        0.0%]    16 [||     5.8%]   22 [        0.0%]
  5  [||      4.3%]   11 [        0.0%]    17 [||     7.4%]   23 [        0.0%]
  6  [||      3.3%]   12 [||      0.9%]    18 [|      2.4%]   24 [        0.0%]
  Mem[||||             1239/31969MB]    Tasks: 42, 229 thr; 1 running
```

```
  1  [||      6.2%]   7  [        0.0%]    13 [|      3.4%]   19 [        0.0%]
  2  [||      5.3%]   8  [        0.0%]    14 [|      0.9%]   20 [        0.0%]
  3  [||      6.7%]   9  [|       0.5%]    15 [|||   10.3%]   21 [        0.0%]
  4  [||      4.3%]   10 [        0.0%]    16 [||     3.8%]   22 [        0.0%]
  5  [||      9.8%]   11 [        0.0%]    17 [||     4.3%]   23 [        0.0%]
  6  [||      9.1%]   12 [        0.0%]    18 [||     3.8%]   24 [        0.0%]
  Mem[||||             1151/31969MB]    Tasks: 41, 218 thr; 2 running
```

```
  1  [||      7.4%]   7  [        0.0%]    13 [|      1.0%]   19 [        0.0%]
  2  [||      3.8%]   8  [        0.0%]    14 [||     4.3%]   20 [        0.0%]
  3  [||      9.2%]   9  [        0.0%]    15 [||     7.7%]   21 [        0.0%]
  4  [||      4.3%]   10 [        0.0%]    16 [|      0.5%]   22 [        0.0%]
  5  [|||    11.4%]   11 [        0.0%]    17 [||     9.9%]   23 [        0.0%]
  6  [||      4.8%]   12 [        0.0%]    18 [||     6.3%]   24 [        0.0%]
  Mem[||||             1215/31969MB]    Tasks: 41, 219 thr; 2 running
```

FIG. 6 Four nodes of the bottom cluster working without encryption during transferring 50 20GiB files


fangchin@sc2u1n2:~
File Edit View Search Terminal Help
fangchin@sc2u1n0:~
File Edit View Search Terminal Help
fangchin@sc2u1n3:~
File Edit View Search Terminal Help
fangchin@sc2u1n1:~
File Edit View Search Terminal Help

```
  1  [||||||60.6%]   7  [||      5.2%]    13 [||||||55.5%]   19 [        0.0%]
  2  [||||||65.2%]   8  [|       0.5%]    14 [||||| 38.6%]   20 [|       0.5%]
  3  [||||||58.2%]   9  [        0.0%]    15 [||||||61.5%]   21 [        0.0%]
  4  [||||||66.7%]   10 [        0.0%]    16 [||||||54.9%]   22 [        0.0%]
  5  [||||||58.8%]   11 [        0.0%]    17 [||||||64.4%]   23 [        0.0%]
  6  [||||||71.6%]   12 [        0.0%]    18 [||||||50.5%]   24 [        0.0%]
  Mem[||||              1654/31969MB]    Tasks: 42, 220 thr; 7 running
```

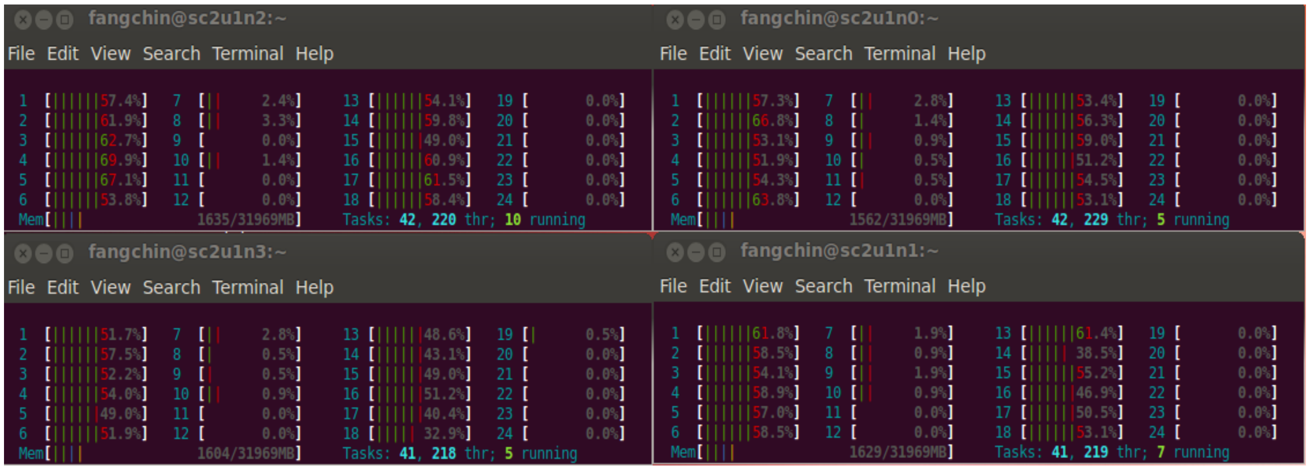
```
  1  [||||||65.9%]   7  [||      6.6%]    13 [||||||65.2%]   19 [        0.0%]
  2  [||||||52.9%]   8  [|       0.5%]    14 [||||||68.8%]   20 [        0.0%]
  3  [||||||52.4%]   9  [|       0.9%]    15 [||||||44.2%]   21 [        0.0%]
  4  [||||||56.7%]   10 [|       0.9%]    16 [||||||46.6%]   22 [        0.0%]
  5  [||||||57.9%]   11 [|       0.5%]    17 [||||||60.7%]   23 [|       0.5%]
  6  [||||||50.2%]   12 [||      2.8%]    18 [||||||43.5%]   24 [|       0.5%]
  Mem[||||              1623/31969MB]    Tasks: 43, 229 thr; 9 running
```

```
  1  [||||||76.0%]   7  [||      3.8%]    13 [||||||67.3%]   19 [        0.0%]
  2  [||||||70.0%]   8  [||      6.6%]    14 [||||||54.8%]   20 [        0.0%]
  3  [||||||68.6%]   9  [||      6.7%]    15 [||||||54.8%]   21 [        0.0%]
  4  [||||||64.4%]   10 [||      8.5%]    16 [||||||67.5%]   22 [        0.0%]
  5  [||||||67.8%]   11 [|       2.4%]    17 [||||||55.9%]   23 [        0.0%]
  6  [||||||70.5%]   12 [|       0.5%]    18 [||||||65.9%]   24 [        0.0%]
  Mem[||||              1634/31969MB]    Tasks: 41, 218 thr; 10 running
```

```
  1  [||||||63.5%]   7  [|||    14.7%]    13 [||||||58.6%]   19 [        0.0%]
  2  [||||||72.5%]   8  [|       0.5%]    14 [||||||66.3%]   20 [        0.0%]
  3  [||||||77.6%]   9  [|       2.8%]    15 [||||||60.9%]   21 [        0.0%]
  4  [||||||66.2%]   10 [|       0.5%]    16 [||||||65.9%]   22 [        0.0%]
  5  [||||||67.5%]   11 [        0.0%]    17 [||||||64.9%]   23 [        0.0%]
  6  [||||||70.9%]   12 [        0.0%]    18 [||||||51.9%]   24 [        0.0%]
  Mem[||||              1685/31969MB]    Tasks: 41, 219 thr; 7 running
```

FIG. 7 Four nodes of the bottom cluster working with TLS during transferring 50 20GiB files

[22] Htop an interactive process viewer for Linux, see http://hisham.hm/htop/


fangchin@sc2u1n2:~
File Edit View Search Terminal Help
fangchin@sc2u1n0:~
File Edit View Search Terminal Help
fangchin@sc2u1n3:~
File Edit View Search Terminal Help
fangchin@sc2u1n1:~
File Edit View Search Terminal Help

```
 1  [||||||57.4%]   7  [||      2.4%]    13 [||||||54.1%]   19 [         0.0%]
 2  [||||||61.9%]   8  [||      3.3%]    14 [||||||59.8%]   20 [         0.0%]
 3  [||||||62.7%]   9  [        0.0%]    15 [||||||49.0%]   21 [         0.0%]
 4  [||||||69.9%]   10 [||      1.4%]    16 [||||||60.9%]   22 [         0.0%]
 5  [||||||67.1%]   11 [        0.0%]    17 [||||||61.5%]   23 [         0.0%]
 6  [||||||53.8%]   12 [        0.0%]    18 [||||||58.4%]   24 [         0.0%]
 Mem[||||          1635/31969MB]    Tasks: 42, 220 thr; 10 running
```

```
 1  [||||||57.3%]   7  [||      2.8%]    13 [||||||53.4%]   19 [         0.0%]
 2  [||||||66.8%]   8  [|       1.4%]    14 [||||||56.3%]   20 [         0.0%]
 3  [||||||53.1%]   9  [||      0.9%]    15 [||||||59.0%]   21 [         0.0%]
 4  [||||||51.9%]   10 [|       0.5%]    16 [||||||51.2%]   22 [         0.0%]
 5  [||||||54.3%]   11 [|       0.5%]    17 [||||||54.5%]   23 [         0.0%]
 6  [||||||63.8%]   12 [        0.0%]    18 [||||||53.1%]   24 [         0.0%]
 Mem[||||          1562/31969MB]    Tasks: 42, 229 thr; 5 running
```

```
 1  [||||||51.7%]   7  [||      2.8%]    13 [||||||48.6%]   19 [|        0.5%]
 2  [||||||57.5%]   8  [|       0.5%]    14 [||||||43.1%]   20 [         0.0%]
 3  [||||||52.2%]   9  [|       0.5%]    15 [||||||49.0%]   21 [         0.0%]
 4  [||||||54.0%]   10 [||      0.9%]    16 [||||||51.2%]   22 [         0.0%]
 5  [||||||49.0%]   11 [        0.0%]    17 [||||||40.4%]   23 [         0.0%]
 6  [||||||51.9%]   12 [        0.0%]    18 [|||||  32.9%]   24 [         0.0%]
 Mem[||||          1604/31969MB]    Tasks: 41, 218 thr; 5 running
```

```
 1  [||||||61.8%]   7  [||      1.9%]    13 [||||||61.4%]   19 [         0.0%]
 2  [||||||58.5%]   8  [||      0.9%]    14 [|||||  38.5%]   20 [         0.0%]
 3  [||||||54.1%]   9  [||      1.9%]    15 [||||||55.2%]   21 [         0.0%]
 4  [||||||58.9%]   10 [||      0.9%]    16 [||||||46.9%]   22 [         0.0%]
 5  [||||||57.0%]   11 [        0.0%]    17 [||||||50.5%]   23 [         0.0%]
 6  [||||||58.5%]   12 [        0.0%]    18 [||||||53.1%]   24 [         0.0%]
 Mem[||||          1629/31969MB]    Tasks: 41, 219 thr; 7 running
```

FIG. 8 Four nodes of the bottom cluster working with TLS + PFS during transferring 50 20GiB files

Min-max summary

A min-max summary of the utilization of the 12 designated cores is seen below. Obviously encryption is very CPU intensive, however in general ≤ 50% of the available hyperthreads have been used and even these are not saturated.

	No encryption	TLS	TLS + PFS
Min	1.00%	28.00%	53.00%
Max	13.00%	71.00%	73.00%

Table 2 Min-Max summary of CPU/core utilization

Graphs

The throughputs and duration of transfer as a function of file size for a single file are shown below:

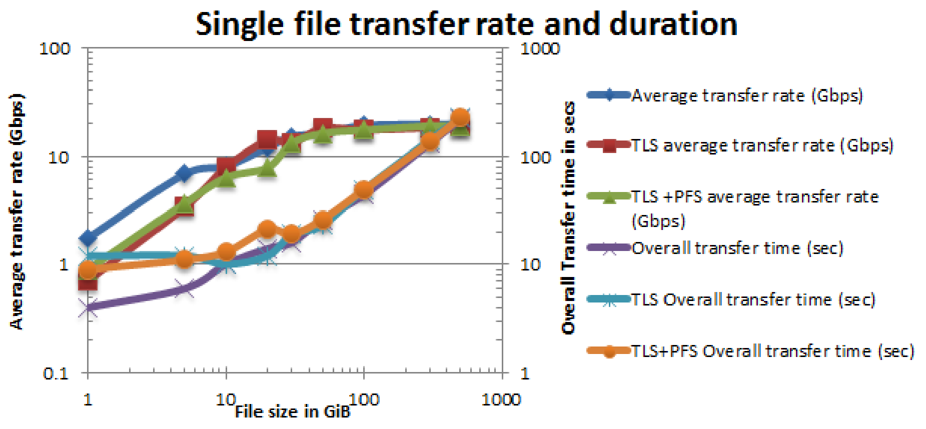


FIG. 9 Single file transfer rate and duration vs. file size in GiB; x-axis in log scale

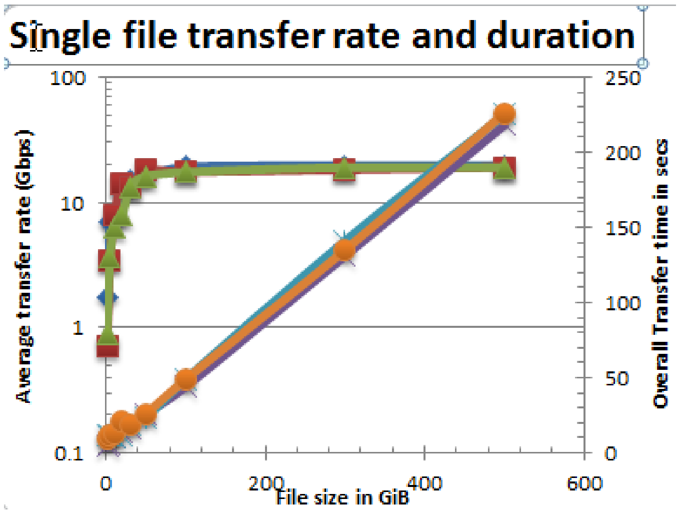


FIG. 10 Single file transfer rate and duration vs file size in GiB; x-axis in linear scale

It is seen that the transfer time is (as expected) roughly a linear function of file size. The transfer rate reaches a transfer rate plateau of 19 - 20Gbps/sec between 50 and 100GiB file size.

For multiple file test data sets the transfer rates are shown below:

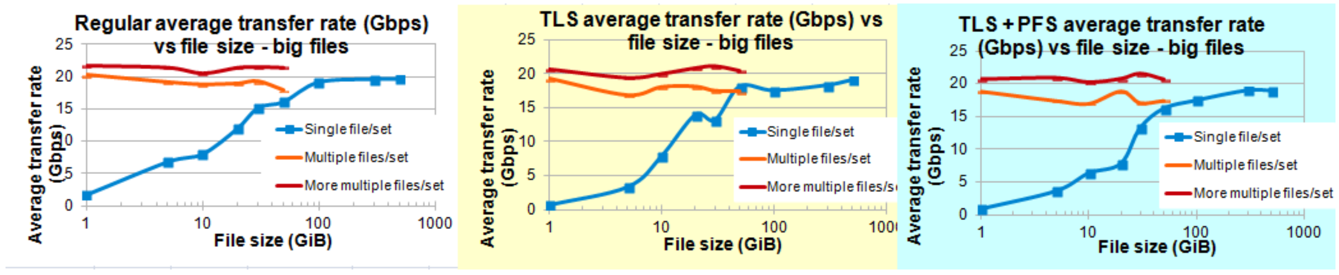


FIG. 11 Transfer rates vs file size in GiB; x-axis in log scale

For the reader's convenience, the numbers of files represented by single, multiple and more multiple files/set are:

File size (GiB)	Nb. of single	Total (GiB)	Nb. of multiple	Total (GiB)	Nb. of more multiple	Total (GiB)
1	1	1	100	100	1000	1000
5	1	5	20	200	200	1000
10	1	10	10	100	100	1000
20	1	20	5	20	20	400
30	1	30	3	90	33	990
50	1	50	2	100	20	1000

CPU/core percentage utilization related statistics

Popular system monitoring and process viewing tools such as the aforementioned `htop` always shows CPU/core utilization in terms of "percentage", but the term itself is not well-defined. Nevertheless, with a review of `htop`'s source, `ProcessList.c`

```
fangchin@zettar1:~/Desktop/htop-1.0.3$ cat -n ProcessList.c|less
[...]
   947        // Guest time is already accounted in usertime
   948        usertime = usertime - guest;
   949        nicetime = nicetime - guestnice;
   950        // Fields existing on kernels >= 2.6
   951        // (and RHEL's patched kernel 2.4...)
   952        idlealltime = idletime + ioWait;
   953        systemalltime = systemtime + irq + softIrq;
   954        virtalltime = guest + guestnice;
   955        totaltime = usertime + nicetime + systemalltime + idlealltime + steal + virtalltime;
[...]
```

the following relations are revealed:

```
Prev_Idle=prev_idle+prev_iowait
Idle=idle+iowait
Prev_Non_Idle=prev_user+prev_nice+prev_system+prev_irq+prev_softirq+prev_steal
Non_Idle=user+nice+system+irq+softirq+steal
Prev_Total=Prev_Idle+Prev_Non_Idle
Total=Idle+Non_Idle
```

Thus:

`CPU_Percentage= ((Total-Prev_Total)-(Idle-Prev_Idle))/(Total-Prev_Total)`

With the definition, a Python tool is being implemented that reads the CPU/core utilization time histories collected by collectd[23], and saved into graphite whisper[24] databases. Then, other than the snapshots as provided by **FIGs. 6 – 8**, during an encryption run, various statistics can be computed for selected cores, e.g, median, percentiles (starting with Inter Quartile Range). Means and standard deviations are OK for nice normal distributions, but are sensitive to outliers. For more detail, plots of probability distributions and cumulative distributions can be generated. Box plots too. We will supplement this overview TN as soon as we have such results available. Furthermore, for CPU performance monitoring, especially for recent generation of Intel Xeon processors, we intend to explore the use of Intel's own tool, the Intel® Performance Counter Monitor - A better way to measure CPU utilization[25]. Regardless, we anticipate that a statistics-based evaluation of CPU utilization should be fruitful.

Additional notes

The following table shows an example of a set of test runs using the three test data sets consisting of 30GiB files

[23] Collectd, see http://collectd.org
[24] Graphite, see http://graphite.readthedocs.org/en/latest/overview.html
[25] Intel® Performance Counter Monitor - A better way to measure CPU utilization, see https://software.intel.com/en-us/articles/intel-performance-counter-monitor

TLS	PFS	File size	Number of files	Max. transfer rate	Average transfer rate	Cipher suite
N	N	30	1	23.05	19.48	N/A
N	N	30	3	22.85	19.77	N/A
N	N	30	33	27.81	18.68	N/A
Y	N	30	1	19.69	17.41	TLS_RSA_WITH_AES_256_GCM_SHA384
Y	N	30	3	19.91	16.99	TLS_RSA_WITH_AES_256_GCM_SHA384
Y	N	30	33	19.98	17.51	TLS_RSA_WITH_AES_256_GCM_SHA384
Y	Y	30	1	19.71	16.32	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
Y	Y	30	3	19.96	17.61	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
Y	Y	30	33	19.6	17.45	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

Table 3 data transfer rates for test data sets consisting of 30GiB test files, without encryption, with TLS and the cipher suite TLS_RSA_WITH_AES_256_GCM_SHA384, and TLS + PFS with the cipher suite TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256. The reduction in throughput caused by TLS is ~10%. There is little significant difference between TPS and TLS + PFS.

Discussions and further work

Before concluding this overview, we would like to discuss a few aspects and point out directions for further work. The hope is that readers will also have a better outlook of our quest. We welcome any feedback and suggestion.

Test Automation

Evaluation of a cluster system that is formed using new software and hardware is a challenging endeavor. A statistics-based approach is often practical and fruitful. Nevertheless, such an approach demands the formulation of a comprehensive test matrix, numerous test runs, careful analysis of results, and skillful data reduction and graphing to gain insights and convey understanding. Both the software and hardware should be sufficiently instrumented so as to be able to provide enough meaningful statistics. At the same time, such instrumentation must be light-weight so as not to impact the observed system performance. To automate testing, a Python based test framework has been developed as the basis for a high-degree of test automation, in conjunction with an extensive use of ansible. For gathering system run-time statistics, we have been using collectd. For monitoring and visualization, we use graphite. Recently, we are introducing Grafana[26] with graphite as its data source to enhance the visualization aspect. Zettar `zx` is capable of providing many run-time statistics which can be gathered using RESTful APIs. To observe its run-time statistics, both its built-in Web UI and/or a stand-alone CLI tool can be used. For casual statistics work, we use spreadsheet tools such as MS Excel and/or OpenOffice Calc. For more demanding calculations, we use R[27].

Encryption approach, cipher suite choice, and system energy-consumption

Referring to **FIG. 6 – 8**, the fact that encryption increases CPU utilization should be evident. Zettar `zx` uses the Botan cryto library[28] implemented in C++11, and as a result has the flexibility of selecting both a desired encryption approach (TLS only or TLS + PFS) and cipher suite. A few obvious questions are: which encryption should be the default choice? Other than Botan's default cipher suite for each, what is the impact to data transfer performance of other supported cipher suites? For

[26] Grafana, see http://grafana.org
[27] R: The R Project for Statistics Computing, see http://www.r-project.org
[28] Botan, see http://botan.randombit.net

each combination of encryption approach and cipher suite, what is the impact to CPU utilization and thus energy-consumption of the data transfer system? What's a good way to measure, record, and analyze such consumption rates? To the authors, there seems to be a dearth of literature for such highly practical engineering issues. We intend to address this shortage of useful references as much as we can in 2015 and beyond.

Hardware-assisted encryption and hash computation

The basic system will be upgrade to use the new Haswell microarchitecture featured in the 4th generation Intel® Core™ Processor family that implements several new instructions designed to improve cryptographic processing performance. We need to work with Intel to integrate this into our research.

Impact of compilers

Zettar `zx` is implemented in the C++11 programming language, so we need a compiler to support the aforementioned Haswell instruction set. It should be of strong interest to compare the executables generated using different compilers, e.g. GCC[29] (system default), Clang[30], and Intel C++ compiler[31].

Impact of WAN latency

The basic system so far have both clusters mounted on the same rack and connected using short patch cables, therefore it doesn't simulate data transfers over long distance. Since Zettar `zx` uses TCP for data transfers, thus the impact of the WAN (delays etc) on performance must be evaluated. For example, the software uses multiple distributed and randomized TCP streams to minimize the impact of TCP's two basic default behavior: slow start and collision avoidance, as shown in **FIG. 12**. Nevertheless, the effectiveness of the approach must be evaluated over high-latency WAN environment – a major effort of 2015.

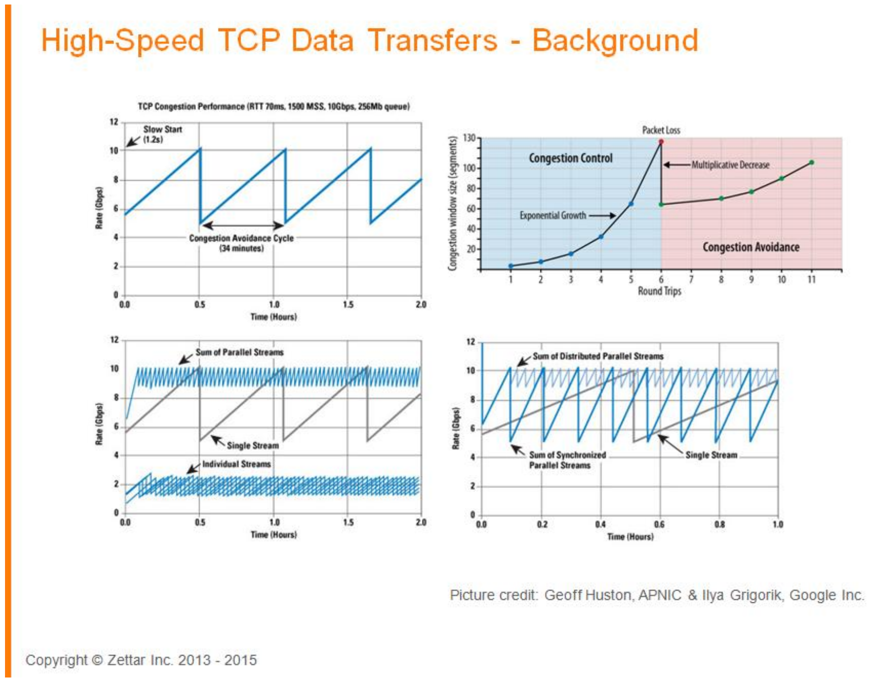


FIG. 12 Background for using TCP for high-speed data transfers. See also Gigabit TCP[32] and High Performance Browser Networking[33]

[29] GCC, see https://gcc.gnu.org
[30] Clang, see http://clang.llvm.org
[31] Intel C++ compiler, see https://software.intel.com/en-us/c-compilers

Areas for further performance improvments

As shown in **FIG. 9 – 10**, with the current system and application settings, the data transfer throughput tops out around 19Gbps - 20Gbps. This represents ~63% bandwidth utilization by default *without any tuning*. At the same time, the max. transfer rate results tabulated in **Table 1** suggests that it should be possible to boost the average default performance. The question is then which area is the most fruitful to tackle first? We have observed that even with the basic system, there is bandwidth to spare, so obviously there is no shortage in that regard. With the CPU utilization as shown in **FIG. 7 – 8**, even without resorting to hardware-assisted encryption acceleration, there is plenty CPU cycles to tap into. That leads to tuning file systems as the most likely candidate. In this regard, again there is a dearth of published literature for references, especially when the employed basic storage components are the latest Intel DC P3700 NVMe SSDs, which has its own unique requirements for burning-in and for file system construction[34]. It should be noted that SSDs in general demand more understanding to use them well[35]. BeeGFS, despite having extensive tuning guides[36] published on its Web site, there is scant information for tuning it when running on top of NVMe devices in conjunction with XFS. XFS tuning for NVMe devices for high-speed Big Data transfers is also a so far under-explored area. Finally, we also need to find a tool that can accurately measures and records the I/O patterns of Zettar zx. Such topics will be covered in a future SLAC Technical Note.

Conclusions

Based on the evaluation of the basic system since mid-December 2014 up to now, The Zettar system embodies a very promising approach for high-speed Big Data Transfers. The biggest question so far is how well it performs over WAN with high-latency (50ms – 200ms). SLAC has the experience, expertise, and facilities to push the limits. With the participation of several vendor partners, Arista Networks Inc., Intel NSG, Quanta Cloud Technology LLC (QCT), and ThinkParQ GmBH, a spin-off of Fraunhofer ITWM, the team endeavors to find out the answer to the aforementioned question. We together plan to create a solution that solves the Big Data mobility challenge.

Acknowledgements

The authors wish to thank Antonio Ceseracciu and Yee Ting Li, for the many fruitful brain-storming sessions. Furthermore, they wish to thank James Williams, SLAC CIO, and the three associate CIOs, Chris Kennedy, Theresa Bamrick, and Adeyemi Adesanya, for their encouragement and support. To Mike Yang, VP and GM of Quanta Computers, parent of Quanta Cloud Technology LLC (QCT), for his unequivocal endorsement of the 2015 joint-project. Raymond Huang, Sales and Marketing Manager, and his QCT colleagues, Konrad Cichocki, Solution Architect, Phong Vu, Field Application Engineer, for their help in upgrading the current basic system to the latest QCT hardware and their always cheerful answers to questions. To James E. Myers, Intel NSG Manager, for his continued support of the 2015 endeavors. To Andrey O. Kudryavtsev, Solution Architect at Intel NSG, for the many insights and tips about maximizing the performance potential of Intel NVMe devices. His expert advices and always pleasant manner are both highly appreciated. To Ranjit Kurup, Long-Range Business Strategy, DCG/Storage Group, Intel, for his warm support and advocacy of the 2015 joint-project inside of the Intel Corp. To Robert Thompson, Named Account Manager, Chris Summers, Technical Sales, Manuel Mendez, System Engineer, all at Arista Networks, Inc. for coming through with two 100G capable switches. The two units greatly simplify the 2015 joint-project

[32] Gigabit TCP, The Internet Protocol Journal – Volume 9, Number 2, June 2006 by Geoff Huston, APNIC

[33] High Performance Browser Nteworking: what every web developer should know about networking and performance Sep. 30, 2013 by Iiya Grigorik, O'Reilly Media

[34] Intel® Linux* NVMe* Driver | Reference Guide for Developers, see http://downloadmirror.intel.com/23929/eng/Intel_Linux_NVMe_Driver_Reference_Guide_330602-002.pdf

[35] SSD Performance – A Primer | August 2013 *An Introduction to Solid State Drive Performance, Evaluation and Test*, see http://www.snia.org/sites/default/files/SNIASSSI.SSDPerformance-APrimer2013.pdf

[36] Extensive tuning guides, see http://www.beegfs.com/wiki/TuningAdvancedConfiguration

setup. To Matt Petach, Sr. Principal Packet Pusher at Yahoo!, for his help in introducing us to Arista folks, thus speeding up the preparation. To Jan Heichler at ThinkParQ GmBH, for always being a pleasure to work with. Chin wishes to thank Dr. Michael Harries, Chief Technologist at Citrix Startup Accelerator, for his support, encouragements, and guidance. He also wishes to give his compliments to his two outstanding engineering colleagues, Igor Solovyov and Alex Nazarenko, for their consistent and excellent contributions, and to Michael Alam, Zettar Enterprise Sales, for bringing Zettar to the attention to the top-tiered M&E studios – a feat in itself.   Lastly, Chin is grateful to his family's unwavering support and love - both are the true foundation of Zettar's accomplishments.