UCI DTN/HPC Filesystem and Network Performance

**FIONA**

CPU

*less /proc/cpuinfo*

* Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz (12 cores, 15M cache)

Memory

RAM:

*sudo dmidecode --type 17*

*sudo lshw -short -C memory*

* 128GB System Memory (8 x 16GB DIMM Synchronous)
  + Size: 16384 MB
  + Speed: 2133 MHz

Storage and Filesystems:

*sudo lshw -class disk*

*dmesg | less*

*sudo lshw | grep SAS* (See disks listed under)

*sudo lshw | grep SATA* (See disks listed under)

* 16 x 240GB SSDS (Serial Attached SCSI controller, LSI Logic / Symbios Logic, SAS-3)
  + ZFS partition
  + SAS-3 has 12Gbps speed

*sudo zpool status*

There are two raidz1 virtual devices and one pool:

* + - pool: bigdata
      * raidz1-0
      * raidz1-1

*sudo df –h*

* + - bigdata 2.9T 700G 2.2T 25% /bigdata

*sudo zpool iostat*

Shows more accurate information on the pool allocations.

capacity operations bandwidth

pool alloc free read write read write

---------- ----- ----- ----- ----- ----- -----

bigdata 831G 2.66T 0 17 18.8K 11.4M

* 2 x 240GB SSDS (SATA controller, C610/X99 series chipset 6-Port SATA Controller)
  + XFS
  + Linux raid

*cat /proc/mdstat*

* + - *md126 : active raid1*
    - *md127 : active raid1*

*df –h*

* + - */dev/md127 223G 86G 137G 39% /*
    - */dev/md126 1015M 224M 792M 23% /boot*

Since the partition is a raid1, the two 240GB drives are mirrored.

*sudo dmidecode -t 2*

Manufacturer: Supermicro

Product Name: X10SRL-F

Googling the manufacturer and product name:

<http://www.supermicro.com/products/motherboard/xeon/c600/x10srl-f.cfm>

* **10x SATA3 (6Gbps)** via C612

*sudo lshw -C disk*

product: INTEL SSDSC2BW24

Googling it up we see these benchmark tests:

<http://ssd.userbenchmark.com/SpeedTest/3104/INTEL-SSDSC2BW240A4>

**Avg. Sequential Write Speed 225MB/s**

***So it seems that even though the SATA3 controller maxes out at 6Gbps, the SSD Hard Drives’s average sequential write speed is 225MB/s.***

Is the maximum achievable write speed set by the individual disks or the SATA3 controller (if aggregate)? How does the number of users affect these speeds?

Tests

DD Fun

*time sh -c "dd if=/dev/zero of=/home/jtatar/10GB.test bs=1M count=10240 && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 4.65732 s, 2.3 GB/s

real 0m28.916s

user 0m0.006s

sys 0m6.034s

10737418240/28.916s/10^9 = **370MB/s**

*time sh -c "dd if=/dev/zero of=/home/jtatar/10GB.test bs=1M count=10240 conv=fdatasync"*

gives the same result as && sync above

* Note: sync – forces the process to write the entire file to disk before completing. Note, that dd will return before completing but the ***time*** command will not, therefore the ***time***output will include the sync to disk.

*time sh -c "dd if=/dev/zero of=/bigdata/transfers/10GB.test bs=1M count=10240 && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 2.24651 s, 4.8 GB/s

real 0m8.365s

user 0m0.007s

sys 0m1.827s

**1.3GB/s**

*time sh -c "dd if=/dev/zero of=/bigdata/transfers/10GB.test bs=1M count=10240 conv=fdatasync"*

gives the same result as && sync above

Let’s sync to disk every 1M – this should be the slowest mode (lower threshold), as the write cache is basically unused:

*time sh -c "dd if=/dev/zero of=/home/jtatar/10GB.test bs=1M count=10240 oflag=dsync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 152.497 s, 70.4 MB/s

real 2m33.895s

user 0m0.013s

sys 0m10.307s

**70MB/s**

*time sh -c "dd if=/dev/zero of=/bigdata/transfers/10GB.test bs=1M count=10240 oflag=dsync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 87.6996 s, 122 MB/s

real 1m27.728s

user 0m0.011s

sys 0m4.805s

**122MB/s**

***FIONA DD Fun Summary***

Writing to /home/\* (XFS mirror) is slower than /bigdata (ZFS) because of the filesystem

SATA controller upper bound: **6Gbps**

Avg. SSD upper bound *per channel*: **225MB/s**

DD && sync /bigdata: limits = (**122MB/s, 1.3GB/s**)

DD && sync /home: limits = (**70MB/s, 350MB/s**)

**For fastest transfers, /bigdata should be used.**

**HPC Interactive Node**

*lshw -short -C memory*

256GiB System Memory

8GiB DIMM DDR3 Synchronous 1600 MHz (0.6 ns)

*lshw -class disk*

size: 2794GiB (3TB)

*lshw | grep SATA*

SB7x0/SB8x0/SB9x0 SATA Controller [IDE mode]

*dmesg | grep SATA*

SATA link up 3.0 Gbps (SStatus 123 SControl 300)

*df -H*

Filesystem Size Used Avail Use% Mounted on

/dev/sda2 2.2T 12G 2.1T 1% /

tmpfs 136G 0 136G 0% /dev/shm

nas-7-7.local:/data 16T 9.0T 7.1T 56% /data

beegfs\_dfs1 685T 515T 170T 76% /dfs1

beegfs\_dfs2 443T 409T 34T 93% /dfs2

beegfs\_fast-scratch 14T 1.6T 12T 12% /fast-scratch

beegfs\_sbak 597T 352T 245T 59% /sbak

compute-4-43.ib:/adl 111T 48T 64T 44% /share/adl

nas-7-2.ib:/pub 61T 53T 7.4T 88% /share/pub

nas-1-1.ib:/zfs//samdata

119T 91T 29T 77% /share/samdata

compute-2-13.ib:/jje 12T 9.5T 2.6T 79% /share/jje

So to test the local filesystem we should write to /scratch

/data – nas-7-7 (mounted over Ethernet)

/pub – nas-7-2 (over IB)

/ssd-scratch – nas-7-1

/fast-scratch – nas-7-1

/flash-scratch – nas-7-1 (not mounted on compute-1-13)

So nas-7-1 seems to be the fast moose in the forest.

*lshw -short -C memory*

512GiB System Memory

16GiB DIMM DDR3 Synchronous 1600 MHz

*lshw | grep SAS*

product: SAS2008 PCI-Express Fusion-MPT SAS-2 [Falcon]

product: MegaRAID SAS 2208 [Thunderbolt]

product: SAS2X36

product: SAS2X36

These are 6Gbps controllers.

*lshw | grep SATA*

description: SATA controller

product: SB7x0/SB8x0/SB9x0 SATA Controller [IDE mode]

**DFS1 and DFS2**

*jtatar@compute-1-13/dfs1/jtatar$ time sh -c "dd if=/dev/zero of=/dfs1/jtatar/10GB.test bs=1M count=10240 && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 21.2095 s, 506 MB/s

real 0m23.144s

user 0m0.014s

sys 0m11.822s

*jtatar@compute-1-13/dfs2/jtatar$ time sh -c "dd if=/dev/zero of=/dfs2/jtatar/10GB.test bs=1M count=10240 && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 12.9465 s, 829 MB/s

real 0m14.911s

user 0m0.011s

sys 0m6.416s

*jtatar@compute-1-13/dfs1/jtatar$ time sh -c "dd if=/dev/zero of=/dfs1/jtatar/10GB.test bs=1M count=10240 conv=fdatasync && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 22.0205 s, 488 MB/s

real 0m24.603s

user 0m0.014s

sys 0m13.174s

*jtatar@compute-1-13/dfs2/jtatar$ time sh -c "dd if=/dev/zero of=/dfs2/jtatar/10GB.test bs=1M count=10240 conv=fdatasync && sync"*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 23.2052 s, 463 MB/s

real 0m25.391s

user 0m0.014s

sys 0m7.178s

Where are the dfs1/dfs2 file servers:

*less /etc/hosts*

# BeeGFS Filesystems - Infiniband

10.2.255.160 nas-7-1.ib

10.2.255.153 dfm-1-1.ib

10.2.255.139 dfm-2-1.ib

10.2.254.164 dfs-3-1.ib

10.2.254.163 dfs-3-2.ib

Transferring the arianna SIM dataset:

*globus-url-copy -vb -cd -r -sync -p 8 -s "/C=US/O=Globus Consortium/OU=Globus Connect Service/CN=bf20a3d8-880f-11e7-a95a-22000a92523b" -cd -r /pub/arianna/SIM/ gsiftp://jtatar@fiona.oit.uci.edu/bigdata/data/arianna/*

While transfer is taking place, let’s take a look at what dfm-2-1 is doing:

*ssh dfm-2-1*

*iftop –i ib0*

compute-1-13 is 10.2.255.157

dfm-2-1.ib => 10.2.255.157 445Kb 579Kb 646Kb

<= 427Kb 555Kb 620Kb

Transferred 370GB from HPC to FIONA. The peak rate was 1.36Gbps. Transfer started 10:28am PST finished at 1:12pm PST.

Clocked the transfer speed at 11:30am and got ~100MB/s avg

Clocked the transfer speed at the end: 370000MB/(164\*60)s = 40MB/s avg

After the first hour, the transfer rate dropped significantly. Could this be because the file sizes decreased? Yes, in part.

Let’s test setup end to end as a researcher would have to if their follow current instructions. For these tests, let’s use 380GB of the ARIANNA experiment SIM data.

Since /dfs1 and /dfs2 are now the largest capacity file systems, I am guessing users won’t have to clean data they put on them:

Filesystem Size Used Avail Use% Mounted on

/dev/sda2 2.0T 11G 1.9T 1% /

tmpfs 127G 100K 127G 1% /dev/shm

nas-7-7.local:/data 15T 8.2T 6.4T 57% /data

beegfs\_dfs1 623T 470T 153T 76% /dfs1

beegfs\_dfs2 403T 377T 26T 94% /dfs2

beegfs\_fast-scratch 13T 532G 12T 5% /fast-scratch

beegfs\_sbak 543T 348T 196T 65% /sbak

nas-7-2.ib:/pub 55T 50T 5.5T 91% /share/pub

Otherwise, they would have to move their data from /pub to /dfs#. Let’s see how long that takes:

*jtatar@compute-1-13/dfs2/jtatar$ time cp -r /pub/arianna/SIM .*

real 129m25.413s

user 0m3.156s

sys 9m30.080s

**Wow. It took 2h to move 380GB of data from /pub to /dfs2**

**When we take the above result and consider the result of the data transfer of the same dataset from HPC to FIONA it becomes clear that the /pub is the bottle neck. What else have we learned? /fast-scratch is not that fast.**

Let’s confirm these suspicions and see how long it takes to transfer the data set from HPC /dfs2 to FIONA’s /bigdata. I expect about 400MB/s.

──────────────────────────────────────────────────────────────────────────────────────

TX: cum: 11.8MB peak: 2.09Mb rates: 1.78Mb 1.90Mb 1.77Mb

RX: 14.3GB 2.60Gb 1.86Gb 2.26Gb 2.15Gb

TOTAL: 14.3GB 2.60Gb 1.86Gb 2.26Gb 2.16Gb

──────────────────────────────────────────────────────────────────────────────────────

TX: cum: 18.7GB peak: 6.43Gb rates: 1.43Mb 1.78Mb 3.24Gb

RX: 55.3GB 2.56Gb 2.56Gb 2.19Gb 2.17Gb

TOTAL: 74.0GB 7.64Gb 2.56Gb 2.20Gb 5.41Gb

**250GB in 20 minutes or at about 210MB/s**

For large files peak speed was about 350MB/s

**300GB were transferred in 25 min**

**The last 80GB are small files around 300KBs.**

──────────────────────────────────────────────────────────────────────────────────────

TX: cum: 28.2GB peak: 1.60Mb rates: 1.60Mb 1.16Mb 1.23Mb

RX: 148GB 171Mb 171Mb 123Mb 130Mb

TOTAL: 177GB 172Mb 172Mb 124Mb 131Mb

──────────────────────────────────────────────────────────────────────────────────────

TX: cum: 28.2GB peak: 1.56Mb rates: 1.17Mb 1.21Mb 1.22Mb

RX: 149GB 166Mb 124Mb 128Mb 128Mb

TOTAL: 177GB 168Mb 125Mb 129Mb 130Mb

**Clocking at 8GB in 8min for around 17MB/s.**

**Final results: 370GB in 102min = 60MB/s**

real 102m32.829s

user 2m44.064s

sys 13m23.580s

The 370GB ARIANNA test dataset we have been using consists of a variety of file sizes that range from 1.1GB on down to several KB. The fast majority of the files ~300KB:

*du -ah /bigdata/data/boo/DTN\_Tests | grep -v "/$" | sort -rh*

1.1G ./CoREAS\_May2015/229

1.1G ./CoREAS\_May2015/200/DAT000001

1.1G ./CoREAS\_May2015/200

1.1G ./CoREAS\_July2016/2195/DAT000001

1.1G ./CoREAS\_July2016/2195

1.1G ./CoREAS\_Feb2016/544

1021M ./CoREAS\_June2015/310

1000M ./CoREAS\_Feb2016/544/DAT000001

992M ./CoREAS\_Feb2016/542

983M ./CoREAS\_May2015/229/DAT000001

976M ./CoREAS\_June2015/310/DAT000001

967M ./CoREAS\_May2015/182

960M ./CoREAS\_June2015/348

947M ./CoREAS\_Feb2016/542/DAT000001

946M ./CoREAS\_June2015/382

932M ./CoREAS\_May2015/209

922M ./CoREAS\_May2015/182/DAT000001

918M ./CoREAS\_June2015/483

915M ./CoREAS\_June2015/348/DAT000001

902M ./CoREAS\_June2015/382/DAT000001

887M ./CoREAS\_May2015/209/DAT000001

883M ./CoREAS\_May2015/218

883M ./CoREAS\_June2015/339

873M ./CoREAS\_June2015/483/DAT000001

866M ./CoREAS\_Feb2016/549

859M ./CoREAS\_June2015/394

850M ./CoREAS\_Feb2016/545

838M ./CoREAS\_May2015/218/DAT000001

838M ./CoREAS\_June2015/339/DAT000001

837M ./CoREAS\_May2015/136

836M ./CoREAS\_June2015/323

827M ./CoREAS\_May2015/125

821M ./CoREAS\_Feb2016/549/DAT000001

820M ./CoREAS\_May2015/193

818M ./CoREAS\_May2015/156

814M ./CoREAS\_June2015/394/DAT000001

813M ./CoREAS\_June2015/340

810M ./CoREAS\_July2016/1919

808M ./CoREAS\_July2016/2199

808M ./CoREAS\_July2016/2196

805M ./CoREAS\_Feb2016/545/DAT000001

803M ./CoREAS\_June2015/381

792M ./CoREAS\_May2015/136/DAT000001

…etc..

7.0K ./CoREAS\_August2017\_HCR/2502/steering/SIM000001.reas

7.0K ./CoREAS\_August2017\_HCR/2502/steering/RUN000001.inp

7.0K ./CoREAS\_August2017\_HCR/2501/steering/SIM000001.reas

7.0K ./CoREAS\_August2017\_HCR/2501/steering/RUN000001.inp

7.0K ./CoREAS\_August2017\_HCR/2500/steering/SIM000001.reas

7.0K ./CoREAS\_August2017\_HCR/2500/steering/RUN000001.inp

1.5K ./CoREAS\_July2016/2226

1.5K ./CoREAS\_July2016/2223

1.5K ./CoREAS\_July2016/2222

1.5K ./CoREAS\_July2016/2221

1.5K ./CoREAS\_July2016/2194

1.0K ./CoREAS\_July2016/2229

1.0K ./CoREAS\_July2016/2228

1.0K ./CoREAS\_July2016/2227

1.0K ./CoREAS\_July2016/2226/steering

1.0K ./CoREAS\_July2016/2223/steering

1.0K ./CoREAS\_July2016/2222/steering

1.0K ./CoREAS\_July2016/2221/steering

1.0K ./CoREAS\_July2016/2220

1.0K ./CoREAS\_July2016/2219

1.0K ./CoREAS\_July2016/2218

1.0K ./CoREAS\_July2016/2217

1.0K ./CoREAS\_July2016/2205

1.0K ./CoREAS\_July2016/2198

1.0K ./CoREAS\_July2016/2194/steering

1.0K ./CoREAS\_July2016/2193

512 ./SimFromJames/outdata/day\_99

512 ./SimFromJames/outdata/day\_97

512 ./SimFromJames/outdata/day\_96

512 ./SimFromJames/outdata/day\_93

512 ./SimFromJames/outdata/day\_92

512 ./SimFromJames/outdata/day\_81

512 ./SimFromJames/outdata/day\_8

Let’s transfer this data to NERSC. Executing on the FIONA:

*myproxy-logon -b -T -l jtatar -s nerscca.nersc.gov*

*time globus-url-copy -vb -cd -r -p 8* [*file:///bigdata/data/boo/*](file:///bigdata/data/boo/) *gsiftp://dtn01.nersc.gov/global/u2/j/jtatar/DTN\_Tests/*

Reached the 50GB quota on the NERSC DTN. Let’s find another filesystem to write to…

Let’s try writing to the cori scratch filesystem:

*time globus-url-copy -vb -cd -r -p 8 file:///bigdata/data/boo/ gsiftp://corigrid.nersc.gov/global/cscratch1/sd/jtatar/DTN\_Tests/*

real 249m38.422s

user 4m17.832s

sys 5m20.170s

jtatar@cori17:/global/cscratch1/sd/jtatar> du -hs

381G .

[jtatar@fiona ~]$ du -hs /bigdata/data/boo/

370G /bigdata/data/boo/

Weird, does ZFS compress the data? Is that why it is 370G on FIONA but 381GB on Cori?

Well, that’s pitiful: 25MB/s upload from FIONA to Cori at NERSC.

Now pulling data by initiating the transfer from the FIONA by downloading the data we stored at NERSC. On the FIONA is is 370GB/282min = 22MB/s

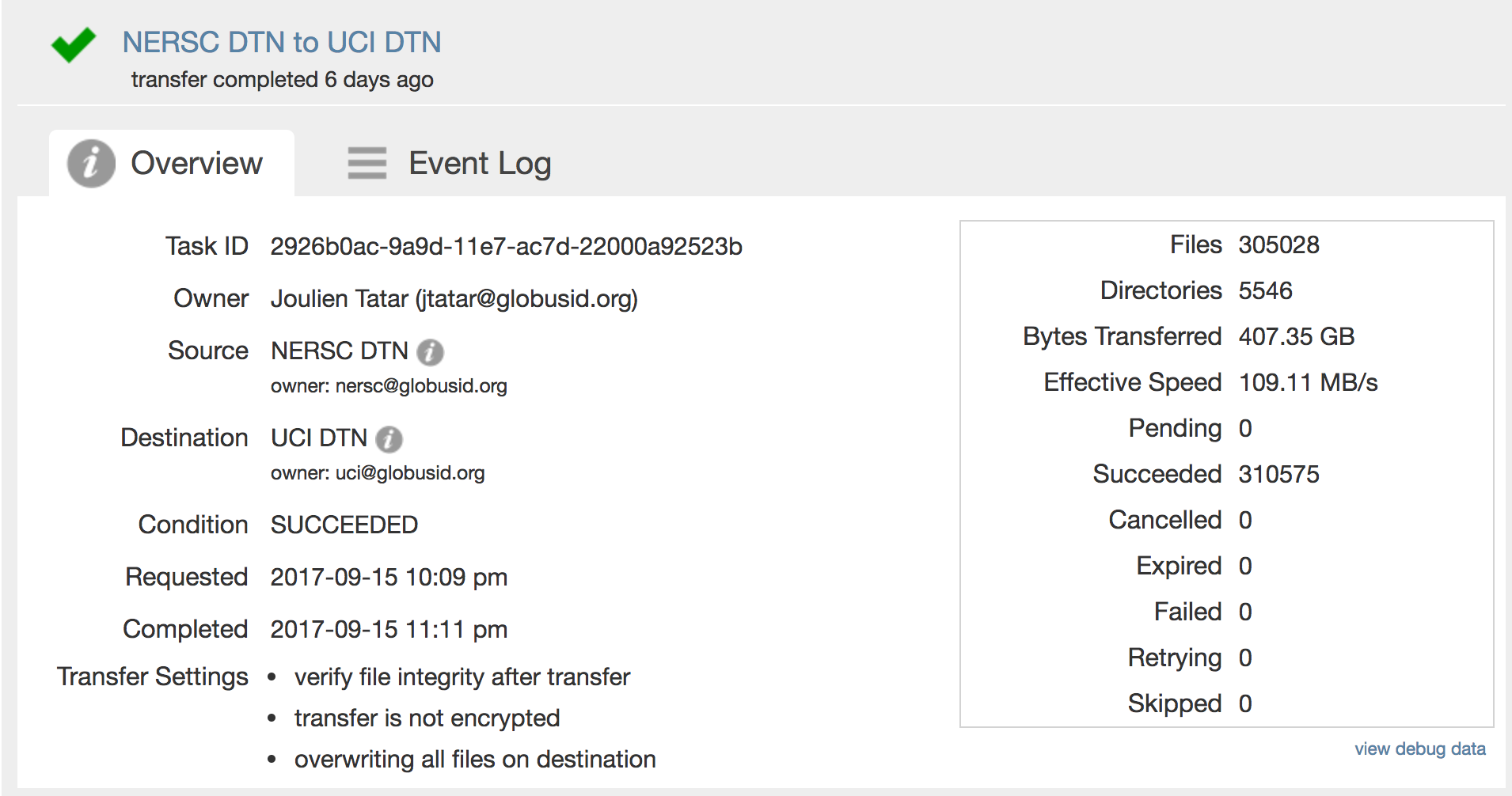
real 282m52.008s

user 5m25.621s

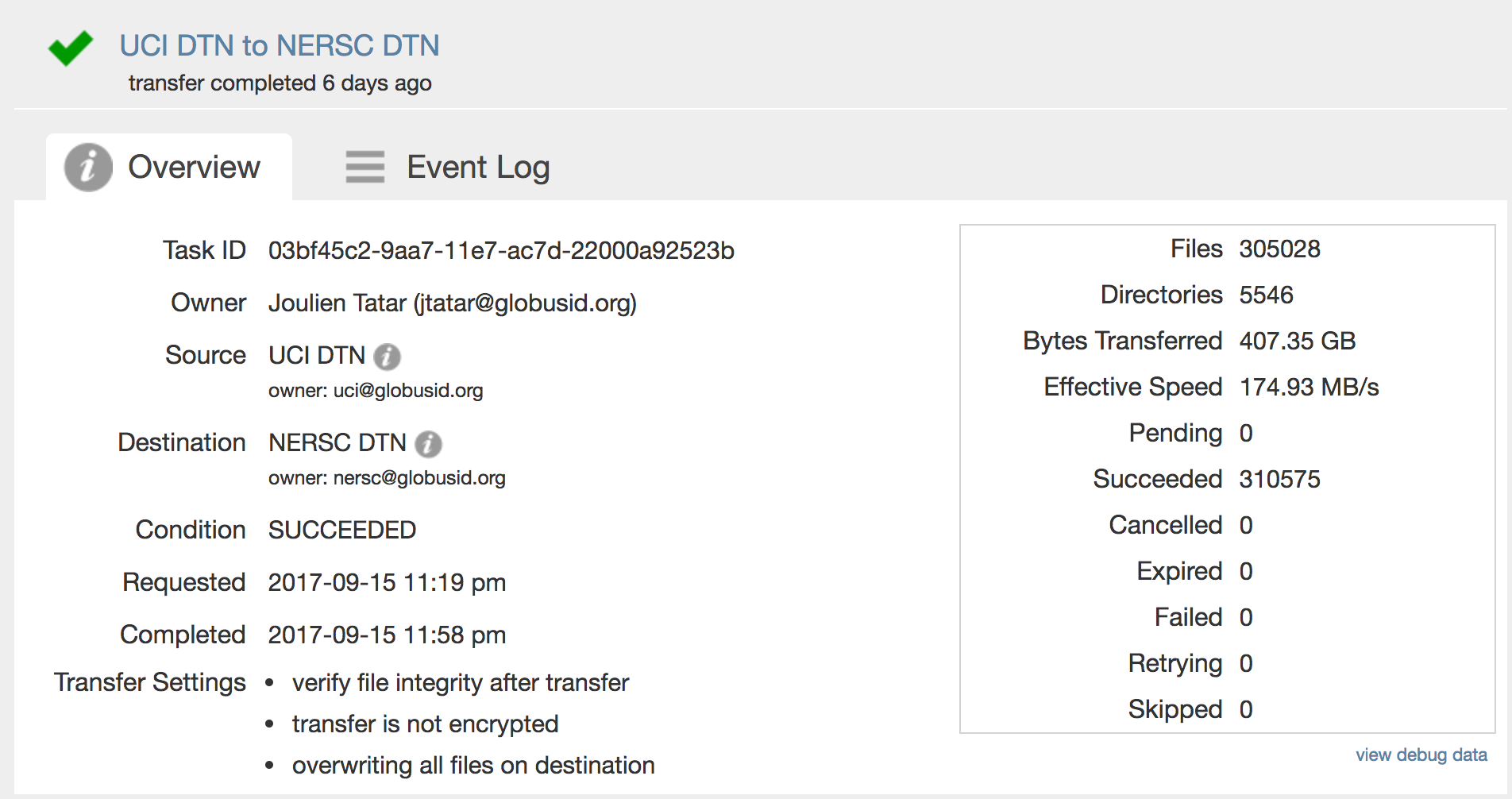
sys 5m50.605s

We should try this with Globus…

**NERSC to FIONA: 110MB/s**



**FIONA to NERSC: 175MB/s**



Let’s double check the file speed of large files by transferring 4 files equaling ~380GB total. Creating 3 different 100GB files and a 80GB file:

*time sh -c "dd if=/dev/zero of=/bigdata/data/boo/BigFiles\_test/fl1\_100GB.test bs=1M count=102400 && sync"*

File 1:

102400+0 records in

102400+0 records out

107374182400 bytes (107 GB) copied, 70.1362 s, 1.5 GB/s

real 1m25.613s

user 0m0.057s

sys 0m18.980s

File 2:

102400+0 records in

102400+0 records out

107374182400 bytes (107 GB) copied, 70.1698 s, 1.5 GB/s

real 1m22.648s

user 0m0.053s

sys 0m20.383s

File 3:

102400+0 records in

102400+0 records out

107374182400 bytes (107 GB) copied, 70.051 s, 1.5 GB/s

real 1m25.611s

user 0m0.060s

sys 0m19.215s

*time sh -c "dd if=/dev/zero of=/bigdata/data/boo/BigFiles\_test/80GB.test bs=1M count=80240 && sync"*

File 4:

80240+0 records in

80240+0 records out

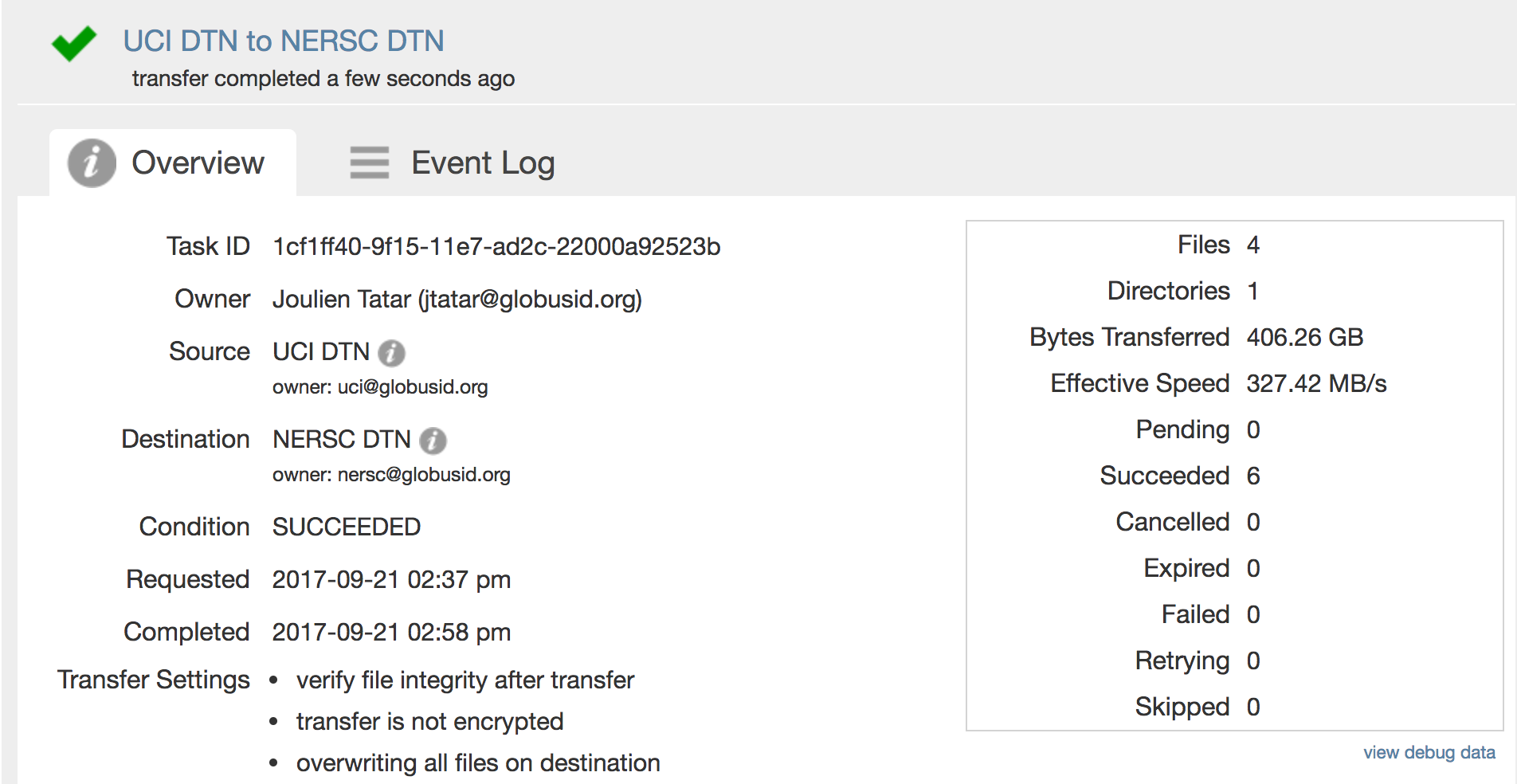
84137738240 bytes (84 GB) copied, 53.2722 s, 1.6 GB/s

real 1m7.832s

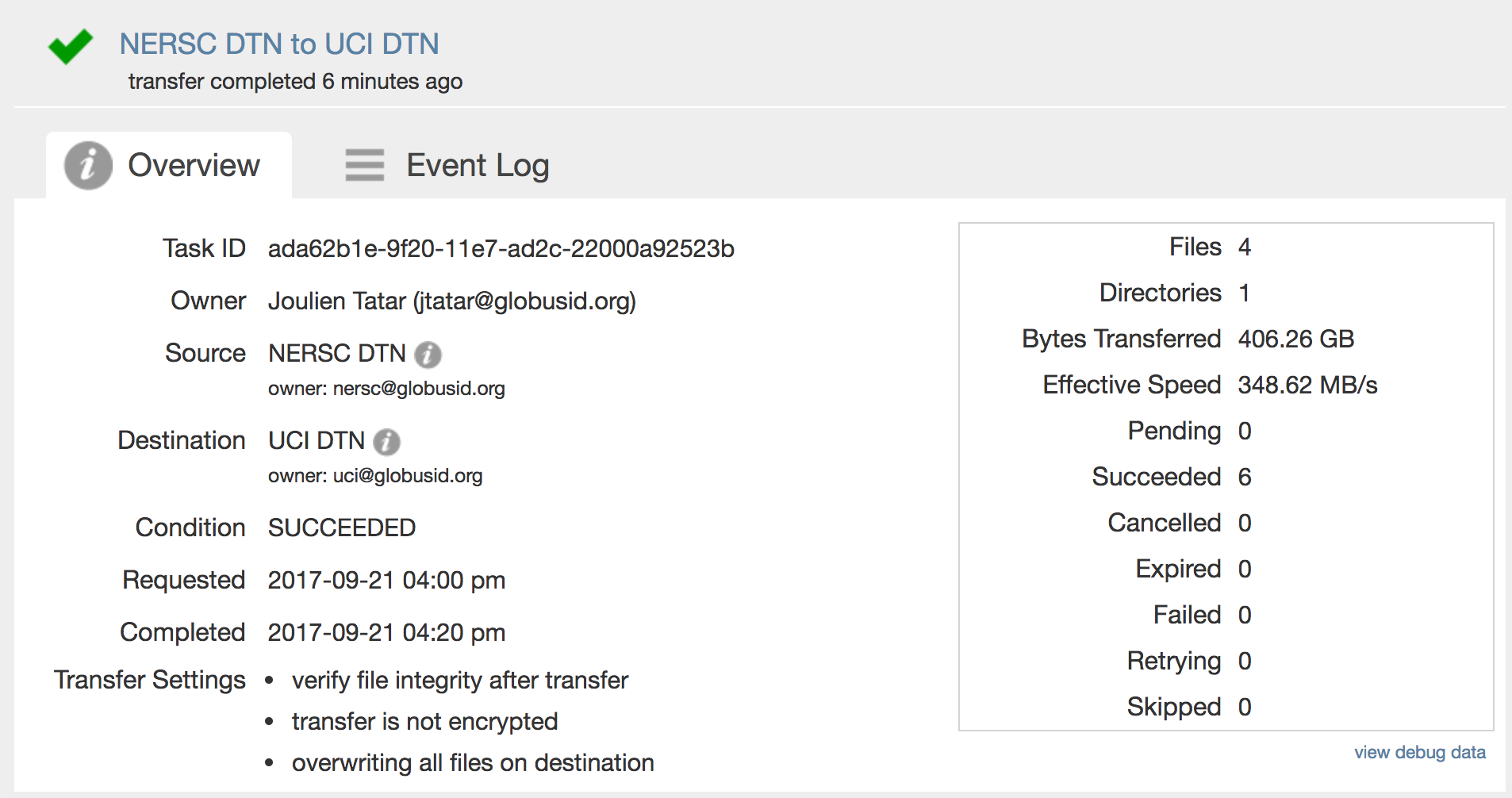
user 0m0.050s

sys 0m15.339s

**FIONA to NERSC = 330MB/s**



**NERSC to FIONA = 350MB/s**



Silly to do transfers to FIONA just to be able to initiate a GridFTP session from outside UCI. Let’s mount /dfs1 and /dfs2 on FIONA by installing the BeeGFS client.

*sudo wget -O /etc/yum.repos.d/beegfs-rhel7.repo* [*https://www.beegfs.io/release/latest-stable/dists/beegfs-rhel7.repo*](https://www.beegfs.io/release/latest-stable/dists/beegfs-rhel7.repo)

*sudo yum install beegfs-client beegfs-helperd beegfs-utils*

*sudo vim /etc/beegfs/beegfs-client-autobuild.conf*

Let’s try it with default kernel IB modules:

*buildArgs=-j8 BEEGFS\_OPENTK\_IBVERBS=1*

“**Note:** In the buildArgs example above, OFED include files are installed to /usr/src/openib. If you are using the default kernel IB modules (i.e. not the separate OFED modules), then the option OFED\_INCLUDE\_PATH does not need to be set at all.”

*sudo /etc/init.d/beegfs-client rebuild*

Quick test of the read speed:

**FIONA /bigdata:**

*[jtatar@fiona BigFiles\_test]$ dd if=80GB.test of=/dev/null bs=1M count=80240*

80240+0 records in

80240+0 records out

84137738240 bytes (84 GB) copied, 46.1888 s, 1.8 GB/s

*[jtatar@fiona BigFiles\_test]$ dd if=80GB.test of=/dev/null bs=1M count=10240*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 7.94453 s, 1.4 GB/s

**HPC DFS:**

*jtatar@compute-1-13/dfs2/jtatar$ dd if=10GB.test of=/dev/null bs=1M count=10240*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 20.5664 s, 522 MB/s

*jtatar@compute-1-13/dfs1/jtatar$ dd if=10GB.test of=/dev/null bs=1M count=10240*

10240+0 records in

10240+0 records out

10737418240 bytes (11 GB) copied, 15.2723 s, 703 MB/s