DRAGNET Cluster Benchmark Numbers

Parts of the cluster and interconnects have been stress tested to optimize configuration and to find upper performance bounds that can be useful for application optimization and rough capacity estimates.

The tests described tend to cover *maximum* achievable performance results on a synthetic ideal workload. It is very likely that your (real) application will never reach these numbers, as the workload is non-ideal, and reaching peak performance can take a lot of effort. But these numbers can serve as a top reference.

Cluster Specifications

CPU computing and memory

Computing and memory numbers N/A

GPU computing, memory, PCIe

Computing numbers N/A

Memory and PCIe Bandwidth

For PCIe bandwidth, there is a substantial difference between the 2 local GPUs and the 2 GPUs local to the other CPU in the same node. NVIDIA's `bandwidthTest` is not meant for performance (because of GPU Boost) (as it says), but here are the numbers anyway:

```
[amesfoort@drg23 bandwidthTest]$ ./bandwidthTest --device=0
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: GeForce GTX TITAN X
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         6325.2

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
```

```
   33554432         4279.2

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         250201.6

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results
may vary when GPU Boost is enabled.
```

```
[amesfoort@drg23 bandwidthTest]$ ./bandwidthTest --device=1
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 1: GeForce GTX TITAN X
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         6178.8

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         4412.2

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         249804.8

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results
may vary when GPU Boost is enabled.
```

```
[amesfoort@drg23 bandwidthTest]$ ./bandwidthTest --device=2
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 2: GeForce GTX TITAN X
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         10547.1
```

```
 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         10892.2

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         249503.5

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results
may vary when GPU Boost is enabled.
```

```
[amesfoort@drg23 bandwidthTest]$ ./bandwidthTest --device=3
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 3: GeForce GTX TITAN X
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         10626.5

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         10903.4

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)    Bandwidth(MB/s)
   33554432         249342.6

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results
may vary when GPU Boost is enabled.
```

Networking

We have benchmarked the infiniband and 10G networks. A good guide is available at the http://fasterdata.es.net/ under `Host Tuning` (and to a lesser extend under `Network Tuning`). But the indicated Linux kernel sysctl knobs did not help; CentOS 7 already has decent settings, and our transfers are all on a low latency LAN (as opposed to wide-area).

Infiniband

Each drgXX node has an FDR (54.545 Gbit/s) HCA (Host Channel Adapter) local to the second CPU (i.e. GPU 1). The 36 port cluster switch is connected to the COBALT switch with 5 aggregated lines (272.727 Gbit/s). See below what can be achieved under ideal circumstances.

IPoIB: TCP and UDP

An application that uses the Infiniband (ib) network normally uses IPoIB (IP-over-Infiniband) to transfer data via TCP or UDP. DRAGNET IPoIB settings have been optimized for TCP (at the cost of UDP performance). We (mostly) use TCP and will not receive UDP data from LOFAR stations directly.

We used the iperf3 benchmark and got the following bandwidth numbers between two drgXX nodes:

```
Out-of-the-box TCP bandwidth: 26-28 Gbit/s. We can ''get iperf3'' TCP bw to
45.4 Gbit/s:

# Set CPU scaling gov to 'performance' (default is 'powersave') (from 39.3
to 45.4 Gbit/s (TCP, 2 streams))
$ for i in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do echo
performance | sudo tee $i; done

[amesfoort@drg23 ~]$ sudo iperf3 -A 10 -B drg23-ib -s
[amesfoort@drg21 ~]$ sudo iperf3 -N -4 -A 10 -B drg21-ib -i 1 -l 1M -P 2 -t
20 -c drg23-ib

-A 10: sets CPU affinity to hw thread 10. mlx_4 is on the 2nd CPU
(8-15(,24-31)) and the INT handler happens to be on hw thr 9
-A 10 (or another suitable nr): from 26.2 to 45.4 Gbit/s (tcp, 2 streams)

-P 2: 2 parallel streams instead of 1: from 36.4 to 45.4 Gbit/s (tcp, -A 10)
```

I have extensively tried the sysctl knobs of the Linux kernel networking settings (net.core.*, net.ipv4.*, txqueuelen), but I did not get an improvement. Likely, Linux kernel autotuning + CentOS 7 settings are already ok for local area networking up to at least 45 Gbit/s.

A real application (not a synthetic benchmark) likely does something else except for data transfer and may have trouble reaching these numbers, because CPU load is a limiting factor and the clock frequency boost of 1 core on drgXX node CPUs is higher (up to 3.2 GHz) than for all cores at once (up to 2.6 GHz). Standard drgXX node CPU clock frequency is 2.4 GHz.

Numbers (before tuning iperf3 tests) obtained using qperf can be seen below under the RDMA sub-section.

RDMA

RDMA (Remote Direct Memory Access) allows an application to directly access memory on another node. Although some initial administration is set up via the OS kernel, the actual transfer commands and completion handling does not go via the kernel. This also saves data copying on sender and receiver and CPU usage.

Typical applications that may use RDMA are applications that use MPI (Message Passing Interface) (such as COBALT), or (hopefully) the LUSTRE client. NFS can also be set up to use RDMA. You can program directly into the `verbs` and `rdma-cm` C APIs and link to those libraries, but be aware that extending some code to do this is not a 1 hr task... (Undoubtedly, there is also a Python module that either only wraps or even makes makes life easier.)

We used the `qperf` benchmark and got the following bandwidth and latency numbers between two `drgXX` nodes (TCP/UDP/SCTP over IP also included, but not as fast as mentioned above):

```
[amesfoort@drg22 ~]$ qperf drg23-ib sctp_bw sctp_lat tcp_bw tcp_lat udp_bw
udp_lat
sctp_bw:
    bw  =  355 MB/sec
sctp_lat:
    latency  =  8.94 us
tcp_bw:
    bw  =  3.65 GB/sec
tcp_lat:
    latency  =  6.33 us
udp_bw:
    send_bw  =  6.12 GB/sec
    recv_bw  =  3.71 GB/sec
udp_lat:
    latency  =  6.26 us
```

```
[amesfoort@drg22 ~]$ sudo qperf drg23-ib rc_bi_bw rc_bw rc_lat uc_bi_bw
uc_bw uc_lat ud_bi_bw ud_bw ud_lat
rc_bi_bw:
    bw  =  11.9 GB/sec
rc_bw:
    bw  =  6.38 GB/sec
rc_lat:
    latency  =  5.73 us
uc_bi_bw:
    send_bw  =    12 GB/sec
    recv_bw  =  11.9 GB/sec
uc_bw:
    send_bw  =  6.24 GB/sec
    recv_bw  =  6.21 GB/sec
uc_lat:
    latency  =  4.03 us
ud_bi_bw:
    send_bw  =  10.1 GB/sec
    recv_bw  =  10.1 GB/sec
ud_bw:
    send_bw  =  5.93 GB/sec
```

```
    recv_bw  =  5.93 GB/sec
ud_lat:
    latency  =  3.94 us
```

```
[amesfoort@drg22 ~]$ sudo qperf drg23-ib rc_rdma_read_bw rc_rdma_read_lat
rc_rdma_write_bw rc_rdma_write_lat rc_rdma_write_poll_lat uc_rdma_write_bw
uc_rdma_write_lat uc_rdma_write_poll_lat
rc_rdma_read_bw:
    bw  =  5.6 GB/sec
rc_rdma_read_lat:
    latency  =  4.99 us
rc_rdma_write_bw:
    bw  =  6.38 GB/sec
rc_rdma_write_lat:
    latency  =  5.35 us
rc_rdma_write_poll_lat:
    latency  =  925 ns
uc_rdma_write_bw:
    send_bw  =  6.26 GB/sec
    recv_bw  =  6.23 GB/sec
uc_rdma_write_lat:
    latency  =  3.58 us
uc_rdma_write_poll_lat:
    latency  =  922 ns
```

```
[amesfoort@drg22 ~]$ sudo qperf drg23-ib rc_compare_swap_mr rc_fetch_add_mr
ver_rc_compare_swap ver_rc_fetch_add
rc_compare_swap_mr:
    msg_rate  =  2.08 M/sec
rc_fetch_add_mr:
    msg_rate  =  2.14 M/sec
ver_rc_compare_swap:
    msg_rate  =  2.1 M/sec
ver_rc_fetch_add:
    msg_rate  =  2.4 M/sec
```

10 Gbit/s Ethernet

The drgXX and dragproc nodes have a 10 Gbit/s ethernet adapter local to the first CPU (i.e. GPU 0). A 48 port Ethernet switch is connected to a LOFAR core switch with 6 aggregated lines (60 Gbit/s). See below what can be achieved under ideal circumstances.

One should be able to achieve (near) 10 Gbit/s using a TCP socket stream. The iperf benchmark achieves 9.91 Gbit/s.

The 6x 10G trunk is apparently not able to utilize all 6 links simultaneously with up to 10 streams (some info on the why is available, but off-topic here). The following is what can be max expected for applications between DRAGNET and COBALT across 10G:

```
DRAGNET -> COBALT, 8x iperf TCP: 38.8 Gbit/s. Second run: 39.7 Gbit/s. (Aug
2015)
```

And a week later we achieved 10G higher throughput using 10 COBALT and 10 DRAGNET nodes. Commands: `iperf -N -B 10.168.130.1 -s` and `iperf -N -B 10.168.96.1 -c 10.168.130.1 -i 1`

```
DRAGNET -> COBALT (1 iperf TCP stream per node) (6x10G) (otherwise idle
clusters and network) (Sep 2015)
#streams        bandwidth (Gbit/s)
1                        9.91 (1 full bw)
2                       19.82 (2 full bw)
3                       29.73 (3 full bw)
4                       29.76 (2 full bw, 2 ~half bw)
5                       29.68 (1 full bw, 4 ~half bw)
6                       39.54 (2 full bw, 4 ~half bw)
7                       49.49 (3 full bw, 4 ~half bw) (best result, 47-49.5)
8                       46.19 (1 full bw, 7(?) ~half bw) (best result after
6 runs, half of the runs didn't do 8 streams)
9                       49.28 (1 full bw, 8 ~half bw)
10                      49.13;50.52 (0 full bw, 10 ~half bw)

COBALT -> DRAGNET (1 iperf TCP stream per node) (6x10G) (otherwise idle
clusters and network) (Sep 2015)
#streams        bandwidth (Gbit/s)
1                9.91 (1 full bw)
2               19.82 (2 full bw)
3               19.82;9.92 (2 full bw, 2 ~half; or 3x 1/3rd)
4               29.72 (2 full bw, 2 ~half)
5               39.64 (3 full bw, 2 ~half) (better than dragnet->cobalt
#streams=5)
8               49.52 (4 full bw, 3 2-3G, 1 near <1G)
10              48.87 (3 full bw, the rest 2-3 G)
```

From 16 CEP2 nodes equally spread over the 4 CEP2 switches to 16 DRAGNET nodes (Sep 2015), iperf got us in 3 test runs:

```
Total iperf (synthetic/benchmark) bandwidth for each test after initial
ramp-up (~8 s):
48.02 Gbit/s
41.97 Gbit/s
44.84 Gbit/s
If you'd transfer large, equal sized files, some files in a set of 16 would
be transferred way earlier than others, since some indiv streams reached
only 1 Gbit/s, while others reached 5 Gbit/s.
```

Doing this with 14 LOTAAS .raw files data using `scp` from CEP2 → DRAGNET is not going to be blazingly fast:

```
time \
```

```
scp locus001:/data/L370522/L370522_SAP000_B000_S0_P000_bf.raw
drg01-10g:/data1/ & \
scp locus004:/data/L370522/L370522_SAP000_B002_S0_P000_bf.raw
drg02-10g:/data1/ & \
scp locus026:/data/L370522/L370522_SAP000_B021_S0_P000_bf.raw
drg03-10g:/data1/ & \
scp locus027:/data/L370522/L370522_SAP000_B022_S0_P000_bf.raw
drg04-10g:/data1/ & \
scp locus028:/data/L370522/L370522_SAP000_B023_S0_P000_bf.raw
drg05-10g:/data1/ & \
scp locus029:/data/L370522/L370522_SAP000_B024_S0_P000_bf.raw
drg06-10g:/data1/ & \
scp locus051:/data/L370522/L370522_SAP000_B045_S0_P000_bf.raw
drg07-10g:/data1/ & \
scp locus052:/data/L370522/L370522_SAP000_B046_S0_P000_bf.raw
drg08-10g:/data1/ & \
scp locus053:/data/L370522/L370522_SAP000_B047_S0_P000_bf.raw
drg09-10g:/data1/ & \
scp locus054:/data/L370522/L370522_SAP000_B048_S0_P000_bf.raw
drg10-10g:/data1/ & \
scp locus076:/data/L370522/L370522_SAP000_B068_S0_P000_bf.raw
drg11-10g:/data1/ & \
scp locus077:/data/L370522/L370522_SAP000_B069_S0_P000_bf.raw
drg12-10g:/data1/ & \
scp locus078:/data/L370522/L370522_SAP000_B070_S0_P000_bf.raw
drg13-10g:/data1/ & \
scp locus079:/data/L370522/L370522_SAP000_B071_S0_P000_bf.raw
drg14-10g:/data1/
real    4m34.894s  ( 274.894 s )
user    0m0.000s
sys     0m0.000s

Total size: 14x 18982895616 bytes = 247.5087890625 Gbyte
=> 7.2 Gbit/s (14 scp streams (idle sys), no dynamic load-balancing)
```

Storage

Storage numbers N/A