Lustre File System on ARM

Architecture Evaluation v1.1

September 2017  Carlos Thomaz
Motivations

▶ ARM momentum
  • 64bit evolution
  • Recent debuts on HPC
  • Traction in new areas such as Machine Learning and AI

▶ Another option in the market
  • Intel established as *de-facto* standard
  • Market needs competitors; cost reduction

▶ Technical reasons
  • Potential high bandwidth, high throughput processor
  • Low power consumption option
  • The Technical challenge
The Cavium ThunderX Architecture

- **SoC architecture**
  - ISA: ARMV8

```bash
dir@sl67:/proc# lsCPU
Architecture: aarch64
Byte Order: Little Endian
CPU(s): 96
On-line CPU(s) list: 0–95
Core(s) per core: 1
Core(s) per socket: 48
Socket(s): 2
NUMA node(s): 2
L1d cache: 32K
L1i cache: 78KL2
cache: 16384K
NUMA node0 CPU(s): 0–47
NUMA node1 CPU(s): 48–95
```
ARM ThunderX and Intel Xeon
ARM Ecosystem

ARM HPC ecosystem roadmap

Hardware
- AppliedMicro X-Gene 1 & 2
- AMD Seattle
- Cavium ThunderX
- AppliedMicro X-Gene 3
- Phytium Mars
- Cavium ThunderX2
- Fujitsu – Post K (SVE)

Open-Source software
- ARM Optimized Routines
- Altair PBS Pro
- GCC (gcc/g++/gfortran)
- LLVM - clang
- OpenHPC 1.2
- ARM Optimized Routines – vector versions
- LLVM – Flang

ARM HPC tools
- ARM Performance Libraries
- ARM Code Advisor (Beta)
- ARM Instruction Emulator
- ARM C/C++ Compiler – ahead of LLVM trunk
- ARM Fortran Compiler
- ARM Code Advisor (Full release)

ISV software
- Allinea DDT and MAP
- NAG Library & Compiler
- Rogue Wave TotalView
- PathScale ENZO
- ISV software

© ARM 2016

Courtesy of ARM – http://arm.com/hpc
DDN Goals evaluating ARM

- Understand if it is a viable option for mid/long term future products
- Understand what’s the effort necessary to make Lustre running optimally on ARM (client and server-side)
- Understand how Lustre and general I/O behaves on ARM SoC architecture
- Contribute to the community
Test Environment used for the study

Diagram showing connections between various components:
- SFA7700KX Block Flash Array
- IB-SRP FDR (56Gbps)
- IB FDR (56Gbps)
- IB FDR 56 Gbps
- IB FDR 56 Gbps
- IB FDR (56Gbps)
- IB Switch 36p
- ARM
- Es7k-vm1
- Es7k-vm1
- IB FDR
- IB FDR
- IB FDR
- IB FDR
- s164
- s165
- s166
- s167
- s161
- s162
- s163
Test environment

- **4 x Gigabyte, Cavium ThunderX2 ARM servers**
  - 128GB RAM, 3 x 40GbE | 4 x 10GbE, 1 x IB FDR 56Gbps
- **1 x SFA7700-IB (ib-srp)**
  - Full flash array, 8 x RAID6 LUNs (200GB SSDs)
- **1 x ES7KE-IB (Intel based, DDN appliance)**
  - Embedded Lustre appliance, 2 controllers, 8 RAID6 pools (OSTs), 2 SSD RAID1 pools for MDT
- **3 x DELL R620 servers**
  - 2 sockets, 12 cores total, 64GB RAM
Lustre File System configuration

▶ ARM Servers and Clients
  • OS: Ubuntu 16.04.03 LTS – Xenial Xerus
  • Kernel: Linux s166 4.4.0-31-generic #50-Ubuntu SMP Wed Jul 13 00:06:30 UTC 2016 aarch64 aarch64 aarch64 GNU/Linux
  • Lustre: 2.10.0.0 + patches
    ◦ LU-9950, LU-9951, LU-9564 (backported for Ubuntu/debian)

▶ X86 clients
  • OS: CentOS Linux release 7.2.1511 (Core)
  • Kernel: Linux s162 3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
  • Lustre: 2.10.0.0

▶ ES7K Embedded Lustre Server
  • OS: CentOS Linux release 7.3.1611 (Core)
  • Kernel: Linux vm01-es7k01 3.10.0-514.21.2.e17.x86_64.lustre #1 SMP Wed Jun 21 03:34:21 PDT 2017 x86_64 x86_64 x86_64 GNU/Linux
  • Lustre: DDN Lustre 2.7.x + Patches
Stand alone ARM servers

Baseline Performance
Single ARM Server – first glimpse
Memory Bandwidth (stream)

Memory bandwidth - Absolute results

Memory bandwidth - Normalized (per core) results

ARM# gcc -O3 -march=ARMv8.1-a -fopenmp -mcmodel=large \ -DSTREAM_ARRAY_SIZE=2600000000 -Wall stream.c -o stream_h
DELL# gcc -Ofast -fopenmp stream.c -Wall -m64 -mcmodel=medium -DSTREAM_ARRAY_SIZE=1100000000 -o stream_h
IB RDMA Network test with IB_SEND_RWD

Sanity tests

```
root@s167:~# ib_send_bw -a -b -c UC -x 192.168.0.185
root@s165:~# ib_send_bw -a -b -c UC -z
***********************************************************************************
*.Waiting for client to connect... *
***********************************************************************************
```

<table>
<thead>
<tr>
<th>#bytes</th>
<th>#iterations</th>
<th>BW peak [MB/sec]</th>
<th>BW average [MB/sec]</th>
<th>MsgRate [Mpps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1000</td>
<td>16.95</td>
<td>16.42</td>
<td>4.303677</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>33.17</td>
<td>32.75</td>
<td>4.292886</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>66.34</td>
<td>65.39</td>
<td>4.285470</td>
</tr>
<tr>
<td>32</td>
<td>1000</td>
<td>132.68</td>
<td>130.73</td>
<td>4.283882</td>
</tr>
<tr>
<td>64</td>
<td>1000</td>
<td>265.36</td>
<td>262.16</td>
<td>4.295241</td>
</tr>
</tbody>
</table>

<SNIP>

```
131072  1000  8250.48  8244.47  0.065956
262144  1000  8263.43  8256.41  0.033026
524288  1000  8252.15  8246.37  0.016493
1048576 1000  8256.10  8248.31  0.008248
2097152 1000  8254.87  8251.13  0.004126
4194304 1000  8256.12  8251.63  0.002063
8388608 1000  8138.98  8138.19  0.001017
```
ARM Server – point to point IB BW
MPI OSU BW and BIBW

### Unidirectional BW

<table>
<thead>
<tr>
<th>Size</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.22</td>
</tr>
<tr>
<td>2</td>
<td>0.84</td>
</tr>
<tr>
<td>4</td>
<td>2.14</td>
</tr>
<tr>
<td>8</td>
<td>4.28</td>
</tr>
<tr>
<td>16</td>
<td>8.56</td>
</tr>
<tr>
<td>32</td>
<td>17.05</td>
</tr>
<tr>
<td>64</td>
<td>33.76</td>
</tr>
<tr>
<td>128</td>
<td>65.66</td>
</tr>
<tr>
<td>256</td>
<td>115.18</td>
</tr>
<tr>
<td>512</td>
<td>245.05</td>
</tr>
<tr>
<td>1024</td>
<td>399.22</td>
</tr>
<tr>
<td>2048</td>
<td>735.48</td>
</tr>
<tr>
<td>4096</td>
<td>1090.75</td>
</tr>
<tr>
<td>8192</td>
<td>1235.69</td>
</tr>
<tr>
<td>16384</td>
<td>2838.55</td>
</tr>
<tr>
<td>32768</td>
<td>4864.02</td>
</tr>
<tr>
<td>65536</td>
<td>4688.12</td>
</tr>
<tr>
<td>131072</td>
<td>5270.83</td>
</tr>
<tr>
<td>262144</td>
<td>5333.59</td>
</tr>
<tr>
<td>524288</td>
<td>5297.62</td>
</tr>
<tr>
<td>1048576</td>
<td>5387.66</td>
</tr>
<tr>
<td>2097152</td>
<td>5400.73</td>
</tr>
<tr>
<td>4194304</td>
<td>5415.79</td>
</tr>
</tbody>
</table>

### Bidirectional BW

<table>
<thead>
<tr>
<th>Size</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.28</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
</tr>
<tr>
<td>4</td>
<td>2.51</td>
</tr>
<tr>
<td>8</td>
<td>4.97</td>
</tr>
<tr>
<td>16</td>
<td>4.02</td>
</tr>
<tr>
<td>32</td>
<td>19.13</td>
</tr>
<tr>
<td>64</td>
<td>38.35</td>
</tr>
<tr>
<td>128</td>
<td>72.64</td>
</tr>
<tr>
<td>256</td>
<td>142.76</td>
</tr>
<tr>
<td>512</td>
<td>269.86</td>
</tr>
<tr>
<td>1024</td>
<td>519.32</td>
</tr>
<tr>
<td>2048</td>
<td>872.59</td>
</tr>
<tr>
<td>4096</td>
<td>1261.87</td>
</tr>
<tr>
<td>8192</td>
<td>1449.39</td>
</tr>
<tr>
<td>16384</td>
<td>2608.61</td>
</tr>
<tr>
<td>32768</td>
<td>4686.06</td>
</tr>
<tr>
<td>65536</td>
<td>7477.37</td>
</tr>
<tr>
<td>131072</td>
<td>8310.94</td>
</tr>
<tr>
<td>262144</td>
<td>8436.23</td>
</tr>
<tr>
<td>524288</td>
<td>8539.24</td>
</tr>
<tr>
<td>1048576</td>
<td>8597.96</td>
</tr>
<tr>
<td>2097152</td>
<td>8624.46</td>
</tr>
<tr>
<td>4194304</td>
<td>8472.55</td>
</tr>
</tbody>
</table>
Single ARM server – storage backend

Simple test to evaluate storage backend – FIO

- 1 x ARM server connected to SFA7700X via IB-SRP (FDR)

```bash
root@sl65:/sys/block# fio --name=foo --rw=read --bs=1m --runtime=30 --time_based --ioengine=libaio --iodepth=64 --direct=1 --numjobs=8 --group_reporting --filename=/dev/sdb --filename=/dev/sdc
foo: (q=0): rw=read, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-1024KiB, (T) 1024KiB-1024KiB, ioengine=libaio, iodepth=64

fio 3.0
Starting 8 processes
Jobs: 8 (f=16): [R(8)][100.0%][r=5260MiB/s,w=0KiB/s][r=5260,w=0 IOPS][eta 00m:00s]
foo: (groupid=0, jobs=8): err= 0: pid=37191: Tue Sep 26 18:51:52 2017
read: IOPS=5426, BW=5427MiB/s (5690MB/s) (159GiB/30074msec)
  slat (usec): min=191, max=101843, avg=1202.10, stdev=5434.79
  clat (usec): min=342, max=562238, avg=92961.48, stdev=54905.53
  lat (usec): min=807, max=562616, avg=94164.77, stdev=55509.54
clat percentiles (msec):
  | 1.00th=[  7], 5.00th=[ 17], 10.00th=[ 31], 20.00th=[ 48],
  | 30.00th=[ 63], 40.00th=[ 75], 50.00th=[ 86], 60.00th=[ 96],
  | 70.00th=[112], 80.00th=[136], 90.00th=[165], 95.00th=[190],
  | 99.00th=[251], 99.50th=[284], 99.90th=[510], 99.95th=[527],
  | 99.99th=[550]

bw (KiB/s): min=153600, max=983040, per=12.50%, avg=694524.80, stdev=105260.97, samples=480
iops : min= 150, max= 960, avg=678.12, stdev=102.78, samples=480
lat (usec) : 500=0.01%, 750=0.01%, 1000=0.01%
lat (msec) : 2=0.06%, 4=0.31%, 10=1.87%, 20=3.92%, 50=15.35%
lat (msec) : 100=41.89%, 250=35.57%, 500=0.90%, 750=0.11%
cpu    : usr=0.34%, sys=31.73%, cxt=19114, majf=0, minf=21757

<SNIP>

Run status group 0 (all jobs):
  READ: bw=5427MiB/s (5690MB/s), 5427MiB/s=5427MiB/s (5690MB/s=5690MB/s), io=159GiB (171GB), run=30074-30074msec

Disk stats (read/write):
  sdb: ios=59827/0, merge=55472/0, ticks=2579464/0, in_queue=2582500, util=94.10%
  sdc: ios=53850/0, merge=57636/0, ticks=2780888/0, in_queue=2793204, util=95.68%
```
Part 1 – ARM Server
IOR Single Client Performance – Multiple Threads

IOR - Single Client Performance - 4MB RPCs
/mnt/arm/bin/ior.arm.mvapich -a POSIX -b 1g -r -w -F -B -t 4m -o
/mnt/arm/file.out

- Writes 4M - Real IO
- Reads 4M - Real IO
IOR Single Client Performance – Multiple Threads

IOR Single Client Performance - 4MB RPCs - REGULAR vs FAKE IO
/mnt/arm/bin/ior.arm.mvapich -a POSIX -b 1g -r -w -F -B -t 4m -o /mnt/arm/file.out

- Writes 4M - Fake IO
- Reads 4M - Fake IO
- Writes 4M - Real IO
- Reads 4M - Real IO
IOR Results – end to end multiple clients (Real I/O)

Multiple Client Performance - 2 to 128 threads, 16MB RPC
Command line used: /mnt/arm/bin/ior.arm.mvapich -a POSIX -b 1g -r -w -B -F -t 16m -o /mnt/arm/file-out -vv
X86 clients against Lustre ARM server
IOR Sequential Performance

Multiple Client IOR Performance - x86 Clients against ARM Server
/mnt/arm/bin/ior.x86.mvapich -a POSIX -b 1g -r -w -F -B -t 16m -o
/mnt/arm/file.out

x86 Writes (16MB) vs x86 Reads (16MB)
ARM and x86 Clients comparison
IOR, multiple clients - Sequential

ARM and x86 Clients - IOR Sequential Reads / Writes (ARM Server)
Sniplet from brw_stats during a set of runs (2 to 128 threads)

| Ltest-OST0000<br> read | write |<br> disk I/O size | ios | % cum | | ios | % cum | |
|---|---|---|---|---|---|---|---|
| 4K: | 127 | 0 | 0 | | 1 | 0 | 0 |
| 8K: | 146 | 0 | 1 | | 0 | 0 | 0 |
| 16K: | 403 | 2 | 4 | | 0 | 0 | 0 |
| 32K: | 681 | 4 | 8 | | 0 | 0 | 0 |
| 64K: | 1590 | 10 | 19 | | 0 | 0 | 0 |
| 128K: | 1565 | 10 | 29 | | 0 | 0 | 0 |
| 256K: | 631 | 4 | 33 | | 0 | 0 | 0 |
| 512K: | 89 | 0 | 34 | | 0 | 0 | 0 |
| 1M: | 9905 | 65 | 100 | | 169184 | 99 | 100 |

Very much the same for all other OSTs Ltest-OST000[0-6]
Part II – ARM Clients
Single Client Performance comparison

Single Client Performance (ARM x x86) - ES7K
/mnt/arm/bin/ior.x86.mvapich -a POSIX -b 1g -r -w -F -B -t 16m -o
/mnt/es7k/file.out
Multiple Client performance comparison

Multiple Client Performance (ARM x x86) - ES7K
/mnt/arm/bin/ior.x86.mvapich -a POSIX -b 1g -r -w -F -B -t 16m -o /mnt/es7k/file.ou

- x86 Writes (16MB)
- ARM Writes (16MB)
- x86 Reads (16MB)
- ARM Reads (16MB)
Preliminary Conclusions
ARM Server - RAW vs Lustre

- RAW performance indicates the ARM systems could potentially sustain high bandwidth
  - We achieved about 7GB/sec reading/writing into and from a Flash based storage that is capable of doing 10GB/sec I/O.
  - The bottleneck is the IB-FDR used with IB-SRP as connection
  - Concurrent Infiniband traffic also performs well. Tests executed demonstrated about 6GB/sec unidirectional BW and about 9GB/sec bi-directional on both IB_RDMA calls (ib_send_bw) and also on MPI layer.
  - Memory bandwidth per core is much lower than other x86 architecture that probably will affect Lustre IO too.
"Noise" - Unpredictability on the Server side

- We observed noise and unpredictable server behavior when scaling up the IO workload thus increasing the number of OSS service threads.
  - Such behavior is related to the highly scalable number of cores on two NUMA domains.
  - Changing LNET partitions plays a little but yet visible effect on server performance.
  - Lustre PIO _should_ help since the effects we are seeing on ARM servers are similar to KNL nodes (high core count, low frequency) – Avoiding serialization should help.

- The best numbers are observed when using 24 to 32 cores
  - More than 32 cores causes noisy and the results become unpredictable. This effect is known, specially on high count core SoC architecture.
  - No L3 cache line and all coherent helps to minimize the effect
  - 4 LNET partitions seems to be optimal for the tested CPU
Server Performance

▶ Reads seems reasonable, writes needs improvement

- Lustre back-end **write** performance is limited to 3-3.5GB/sec
  - It’s about 50% of RAW I/O performance
  - Client concurrency slow down to 2-2.5GB/sec
    - Increasing the default number of OSS service threads didn’t take much effect (default 360).
- Lustre back-end **read** performance seems to be max out to 5-5.5GB/sec
  - Compared to other Lustre back-end, Read performance seems good.
  - Ext4 can provide maximum of 6-6.5GB/sec (for this test environment).
Minimizing NUMA effects

Change LNET partition table
- Initially set to 8 partitions, brought the inflexion point lower
- 4 partitions was the setting that provides better and more reliable performance

```
root@s165:~# cat /proc/sys/lnet/cpu_partition_table
0    :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
1    : 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
2    : 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
3    : 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95

root@s165:~# cat /etc/modprobe.d/lustre.conf
options lnet networks=o2ib(ib0)options
libcfs cpu_npartitions=4options
libcfs cpu_pattern=""
```
ARM Lustre Clients

- Overall Performance equivalent to OLD Xeons, but likely to be half of the current ones.
  - 24 core ARM matches the 12 core Haswells (reads and writes)
  - Ability to write faster on an optimized DDN ES7K also helps to blame ARM server for lower numbers

- Similar type of NUMA issues found on client, but harder to understand and tune.
  - Benefits of LNET partitions and other NUMA tuning still not clear
  - Applications can probably have better behavior using `numactl`
Lustre

► Build procedure required three patches
  • LU-9950 and LU-9951
    o Build process, not really a Lustre code change
    o Patches on JIRA
  • LU-9564 backported (in order to build server on Ubuntu/debian)
  • Not very complicated, but require some cleanup in the process (built on Ubuntu - caused some library incompatibilities)

► The process overall is easy and straightforward
What next?

- Study still in very preliminary stage
- More research on the server side
  - We are interested in alternatives for the current offerings
  - Evaluate SoC features for better utilization (crypto, RAID engine, virtualization)
  - Profile IO and general workloads
- Need to test 40GbE
  - Native chips and embedded Switch on SoC is supposedly to deliver better I/O balance (opposed to utilization of single IB card)
- Experiment in larger scale
  - Looking for large environments willing to cooperate
- Lustre side
  - P0: Run tests with PIO and compare results
  - Profile writes
Thank you

Carlos Thomaz
Thanks for the help from Frank Leers, Gu Zheng and rest of the team.
Extra slides
Building Lustre

► Submitted patches in JIRA
   • https://review.whamcloud.com/#/c/27323/
   • https://jira.hpdd.intel.com/browse/LU-9950
   • https://jira.hpdd.intel.com/browse/LU-9951

► Prepare kernel source

root@s164:~ git clone http://kernel.ubuntu.com/git-repos/ubuntu/ubuntu-xenial.git/ ubuntu-kernel
root@s164:~/ubuntu-kernel# uname -r 4.4.0-93-generic
root@s164:~/ubuntu-kernel# git tag | grep 4.4.0-93 Ubuntu-4.4.0-93.116
root@s164:~/ubuntu-kernel# git checkout Ubuntu-4.4.0-93.116

► Configure Kernel source

root@s164:~/ubuntu-kernel# touch .scmversion
root@s164:~/ubuntu-kernel# cp /boot/config-`uname -r` .config
root@s164:~/ubuntu-kernel# cp /usr/src/linux-headers-`uname -r`/Module.symvers
Building Lustre

Patch Makefile

```
root@s164:~/ubuntu-kernel# git diff
diff --git a/Makefile b/Makefile
index f1fee0c..5f235dc 100644
--- a/Makefile
+++ b/Makefile
@@ -1,7 +1,8 @@
 VERSION = 4
 PATCHLEVEL = 4
 -SUBLEVEL = 79
 -EXTRAVERSION =
 +SUBLEVEL = 0
 +EXTRAVERSION = -93-generic
 +
  NAME = Blurry Fish Butt

# *DOCUMENTATION*
root@s164:~/ubuntu-kernel# make modules_prepare
```

Patch Lustre

- LU-9950, LU-9951, review.whamcloud.com/#/c/27323/

Build Lustre

```
bash autogen.sh && ./configure --enable-server --enable-ldiskfs --with-zfs=no --with-o2ib=/usr/src/ofa_kernel/default/ \
--with-linux=/root/ubuntu-kernel/ --enable-module && make debs
```
Installing e2fsprogs

Build and replace e2fsprogs

git clone git://git.hpdd.intel.com/tools/e2fsprogs.git
cd e2fsprogs git checkout v1.42.13.wc6 -b v1.42.13.wc6
wget -P . / http://archive.ubuntu.com/ubuntu/pool/main/e/e2fsprogs/e2fsprogs_1.42.13-1ubuntu1.debian.tar.xz
tar --exclude "debian/changelog" -xf .. / e2fsprogs_1.42.13-1ubuntu1.debian.tar.xz
sed -i 's/ext2_types-wraper.h$/g' lib/ext2fs/Makefile.in
dpkg-buildpackage -b -us -uc
dpkg -i libcomerr2_1.42.13-1_arm64.deb libss2_1.42.13-1_arm64.deb e2fsck-static_1.42.13-1_arm64.deb e2fslibs_1.42.13-1_arm64.deb e2fsprogs_1.42.13-1_arm64.deb