

Hero benchmarks on the Alpine filesystem

Dustin Leverman and Bill Hartner

20180919

ORNL is managed by UT-Battelle
for the US Department of Energy



Overview

- Spider3 file system specifications
- Spider3 innovations/improvements
- Sequential IOR to 2.5TB/s
- Random IOR to 2.2TB/s
- 1.05m 32KiB creates

Spider3 – Alpine

- 2.5TB/s sequential read/write
- 2.2TB/s random read/write
 - IO randomized by GPFS+GMR
 - 16M block size
- 2.6M IOPs at 32KiB file size
 - NVMe coalescing
- 1/4 EB usable capacity
- 77+1x IBM ESS GL4s
 - GMR de-clustered RAID with dual parity
 - Will serve as burst-buffer servers
- 32,494 10TB NL-SAS drives
 - Need de-clustered parity for rebuilds



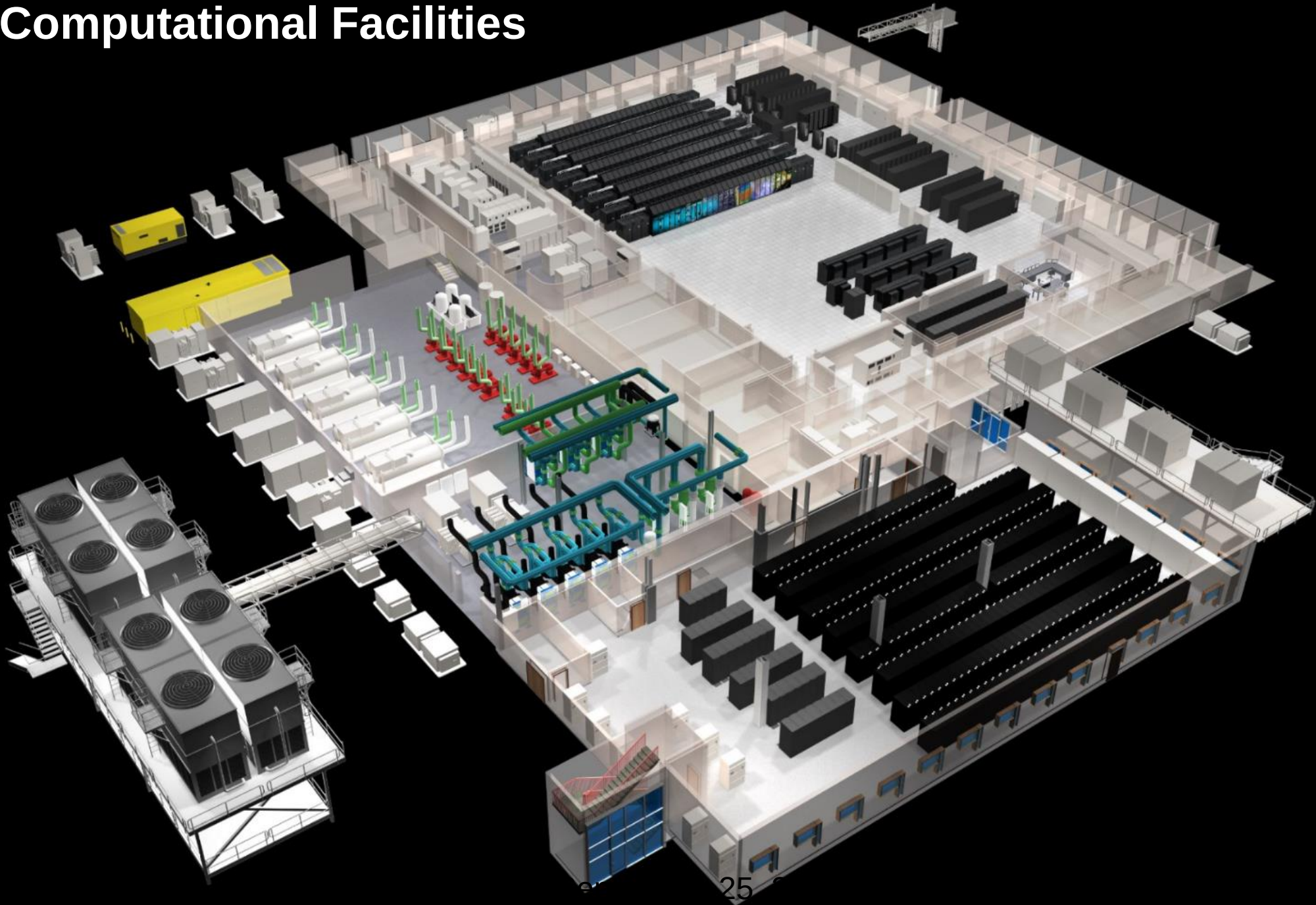
- GPFS-5.x based
- EDR InfiniBand attached (100Gb/s)

Alpine Layout/Facilities

- Each ESS GL4 is a SSU
 - 77x SSUs in system
- 8x SSUs make an SSC (quad)
 - 10x SSCs in system
 - Smallest unit that can operate in isolation
 - SSCs share a pair of TOR IBSWs
 - SSCs have a single Ethernet sw



OLCF Computational Facilities



Spider3 Innovations/Improvements

- GPFS improvements:
 - Largest GPFS file system install (single namespace of 250PB)
 - GPFS tool improvements (fsck, etc...)
 - 10M files in a single directory
 - Max file size equivalent to Summit aggregate system memory
 - Metadata creates of 50k/sec in a single directory
 - RPC and GPFS daemon traffic performance improvements
 - Mutex and semaphore removal and optimization
 - Monitoring/metrics changes (big data store)

Sequential IOR at 2.5TB/s

- This was the first test we attempted to optimize
 - Should be the most difficult to achieve
- Requirements:
 - 2.5TB/s sustained read/write
 - 3 iterations
 - 20 minute runs
 - IO size set to file system block size

Sequential IOR at 2.5TB/s: Monitoring

We monitor many aspects of the system using Nagios and snmp exec:

- BMC sensors: CPU health, memory, power supply, etc...
- Recovery group active on primary or failover server
- IB HCA health: IB cable link speed health, link to PCI bus health, error counter thresholds
- Host SAS health: SAS cable link speed health, link to PCI bus health, error counter threshold
- SAS cable topology check (gssfindmissing disks)
- Mmhealth
- Others: file systems, crond, puppet, load, ntpd, sshd, ping, postfix, etc...

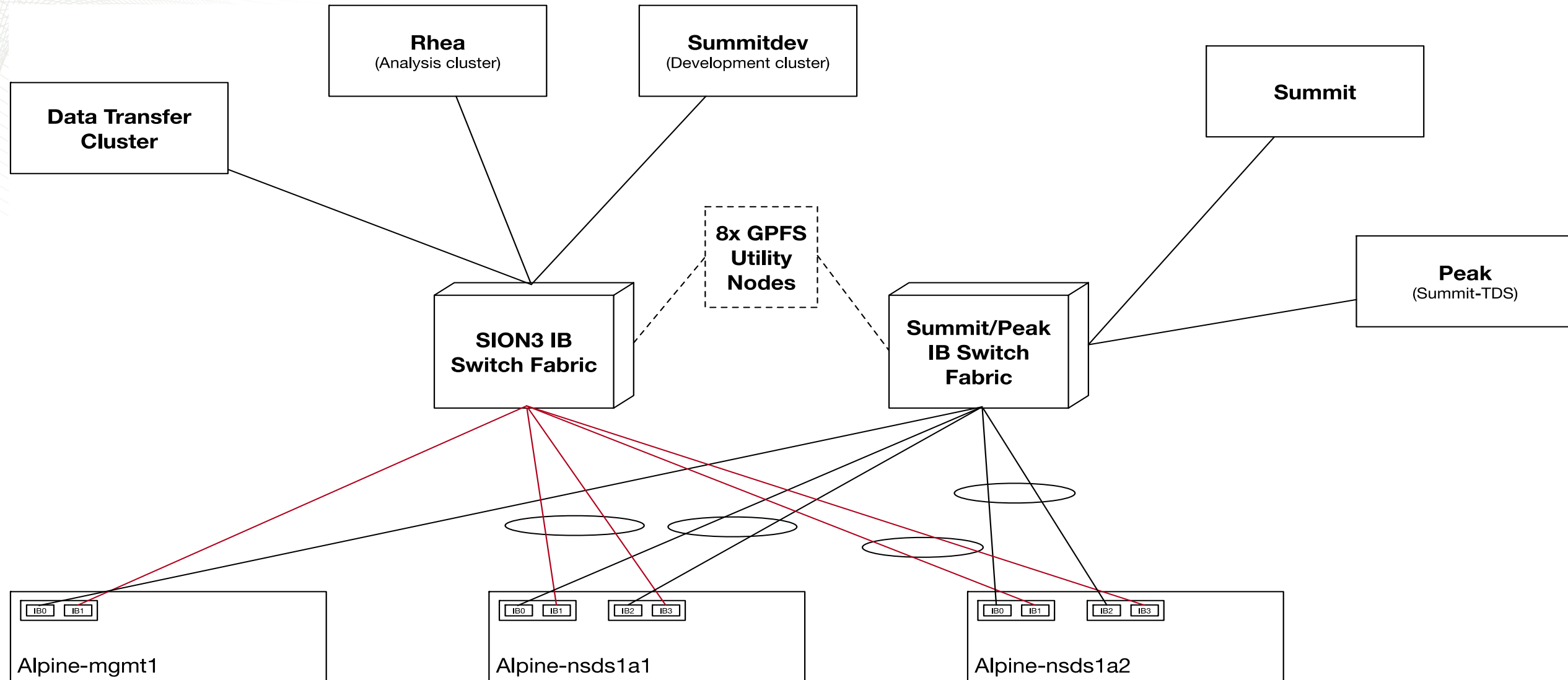
Sequential IOR at 2.5TB/s: Monitoring

We observed that filesystem performance was dropping at regular intervals from ~2.5TB/s -> ~1.8TB/s

- nsdRAIDDiskDiscoveryInterval
 - Queries SAS network every 180 seconds (3 minutes)
 - Very disruptive to performance (~25% overall runtime hit, ~50% instantaneous)
 - Changed interval to 3600 seconds
- Mmhealth
 - Default was HIGH, which is not good for a large cluster
 - Changed to LOW and significantly less performance disruptions
- Host SAS health:
 - No performance impact
- SAS cable topology check (gssfindmissingdisks)
 - No performance impact

InfiniBand design

Dual-homed network design for global file system – (not fully supported – need routing)



** Red links denote where the daemon traffic interface is bound

Acceptance Issues(s): InfiniBand Network

- Lots of slow routes
 - Enabled Adaptive Routing (AR)
 - verbsPortsOutOfOrder option added
 - Increased performance by roughly 30%?
 - Performance variability if too many nodes per TOR switch
 - Original testing was with 18 computes per cabinet (max)
 - We found that 6-7 nodes per rack was optimal
 - Performance degradation of >50%?
 - Aligning benchmarks on leaf modules (balance)
 - LSF host list not supported on over 500 nodes
 - Regained ~1-3% performance
- GPFS doesn't recover well from InfiniBand disruption
 - Reroutes caused by switch->switch cables being unplugged
 - Much better with SHIELD enabled: Mellanox feature that caches routes so if heavy sweep isn't required, you don't have to completely reroute (3 minute disruption)
 - Alpine still doesn't ride gracefully through UFM reboots
 - Investigation still ongoing

Acceptance Issues(s): InfiniBand Network

- VerbsRdmasPerConnectionOverride 4
 - Default is 16
 - Decreases the number of RDMA's to a peer on the IB network; lowering this decreasing congestion.
 - 10% - 20%?
- verbsPorts mlx5_1/1/1/0 mlx5_3/1/1/0 mlx5_0/1/100/2 mlx5_0/1/102/2
mlx5_0/1/104/2 mlx5_0/1/106/2 mlx5_0/1/108/2 mlx5_2/1/101/2
mlx5_2/1/103/2 mlx5_2/1/105/2 mlx5_2/1/107/2 mlx5_2/1/109/2
 - We have 4x IB links on the server and 2x IB links on the client. Maps the buffers more cleanly between clients and servers.
 - Small performance impact, but addressed issues with hangs
- Change payload size:
 - /usr/bin/mlxconfig -y -d 0000:01:00.0 s INT_LOG_MAX_PAYLOAD_SIZE=12
 - What does this do?
 - ~1GB/s performance improvement per server/HCA

Acceptance Issues: IPoIB

- Though we have RDMA enabled, GPFS still uses IPoIB for initial connection setup, control traffic, and some small IO
- '/sbin/ip link set %k txqueuelen 10000'
 - Number of frames allowed per kernel transmission queue; can queue up more traffic
- Ring buffer tuning
 - Increasing ring buffer tunables to avoid discards during bursts of IPoIB traffic
 - '/sbin/ethtool -G %k rx 8192'
 - '/sbin/ethtool -G %k tx 8192'
 - Eats memory ☹
 - MOFED bug that is being addressed
 - 4k vs. 64k pages
 - Don't know the %memory impact; at least 32GB...

Sysctl tunables (mostly network)

- kernel.sched_migration_cost_ns: {value: '5000000'}
- kernel.sched_min_granularity_ns: {value: '10000000'}
- kernel.shmmax: {value: '137438953472'}
- kernel.sysrq: {value: '1'}
- net.core.netdev_max_backlog: {value: '250000'}
- net.core.optmem_max: {value: '4194304'}
- net.core.rmem_default: {value: '4194304'}
- net.core.rmem_max: {value: '4194304'}
- net.core.somaxconn: {value: '10000'}
- net.core.wmem_default: {value: '4194304'}
- net.core.wmem_max: {value: '4194304'}
- net.ipv4.conf.all.arp_filter: {value: '1'}
- net.ipv4.conf.all.arp_ignore: {value: '1'}
- net.ipv4.conf.all.rp_filter: {value: '0'}
- net.ipv4.conf.bond0.rp_filter: {value: '0'}
- net.ipv4.conf.bond1.rp_filter: {value: '0'}
- net.ipv4.conf.ib0.rp_filter: {value: '0'}
- net.ipv4.conf.ib1.rp_filter: {value: '0'}
- net.ipv4.conf.ib2.rp_filter: {value: '0'}
- net.ipv4.conf.ib3.rp_filter: {value: '0'}
- net.ipv4.ip_local_port_range: {value: '2000 65535'}
- net.ipv4.neigh.bond0.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.bond0.mcast_solicit: {value: '9'}
- net.ipv4.neigh.bond0.ucast_solicit: {value: '9'}
- net.ipv4.neigh.bond1.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.bond1.mcast_solicit: {value: '9'}
- net.ipv4.neigh.bond1.ucast_solicit: {value: '9'}
- net.ipv4.neigh.default.gc_interval: {value: '2000000'}
- net.ipv4.neigh.default.gc_thresh1: {value: '30000'}
- net.ipv4.neigh.default.gc_thresh2: {value: '32000'}
- net.ipv4.neigh.default.gc_thresh3: {value: '32768'}
- net.ipv4.neigh.ib0.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.ib0.mcast_solicit: {value: '9'}
- net.ipv4.neigh.ib0.ucast_solicit: {value: '9'}
- net.ipv4.neigh.ib1.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.ib1.mcast_solicit: {value: '9'}
- net.ipv4.neigh.ib1.ucast_solicit: {value: '9'}
- net.ipv4.neigh.ib2.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.ib2.mcast_solicit: {value: '9'}
- net.ipv4.neigh.ib2.ucast_solicit: {value: '9'}
- net.ipv4.neigh.ib3.gc_stale_time: {value: '2000000'}
- net.ipv4.neigh.ib3.mcast_solicit: {value: '9'}
- net.ipv4.neigh.ib3.ucast_solicit: {value: '9'}
- net.ipv4.tcp_adv_win_scale: {value: '1'}
- net.ipv4.tcp_congestion_control: {value: 'htcp'}
- net.ipv4.tcp_fin_timeout: {value: '10'}
- net.ipv4.tcp_low_latency: {value: '1'}
- net.ipv4.tcp_max_syn_backlog: {value: '4096'}
- net.ipv4.tcp_max_tw_buckets: {value: '1440000'}
- net.ipv4.tcp_mem: {value: '16777216 16777216 16777216'}
- net.ipv4.tcp_mtu_probing: {value: '1'}
- net.ipv4.tcp_recovery: {value: '0'}
- net.ipv4.tcp_reordering: {value: '3'}
- net.ipv4.tcp_rfc1337: {value: '1'}
- net.ipv4.tcp_rmem: {value: '4096 87380 4194304'}
- net.ipv4.tcp_sack: {value: '0'}
- net.ipv4.tcp_timestamps: {value: '0'}
- net.ipv4.tcp_window_scaling: {value: '1'}
- net.ipv4.tcp_wmem: {value: '4096 65536 4194304'}
- net.ipv6.conf.all.disable_ipv6: {value: '1'}
- net.ipv6.conf.default.disable_ipv6: {value: '1'}
- vm.dirty_background_ratio: {value: '3'}
- vm.dirty_ratio: {value: '10'}
- vm.min_free_kbytes: {value: '512000'}
- vm.swappiness: {value: '0'}
- vm.max_map_count: {value: '262144'}

GPFS specific configs

• autoloading no	• [ems,gss_ppc64,util]	• nsdRAIDTracks 1048576	• [ems]	• pagepool 32G
• dmapiFileHandleSize 32	• nsdClientChecksumTypeLocal ck64	• nsdMultiQueue 256	• verbsPortsOutOfOrder mlx5_0	• [common]
• clusterName alpine.ccs.ornl.gov	• nsdClientChecksumTypeRemote ck64	• [common]	• [common]	• preferDesignatedMnode yes
• clusterId 3045639913303279948	• verbsRdma enable	• idleSocketTimeout 0	• verbsRecvBufferCount 110	• mnodeRemoteAccessThreshold 0
• verbsRdmaUseGidIndexZero yes	• verbsRdmaSend yes	• [util]	• [gss_ppc64,util]	• [gss_ppc64]
• [gss_ppc64]	• [common]	• prefetchThreads 384	• verbsPorts mlx5_1/1/1/0 mlx5_3/1/1/0 mlx5_0/1/100/2 mlx5_0/1/102/2 mlx5_0/1/104/2 mlx5_0/1/106/2 mlx5_0/1/108/2 mlx5_2/1/101/2 mlx5_2/1/103/2 mlx5_2/1/105/2 mlx5_2/1/107/2 mlx5_2/1/109/2	• nsdRAIDDiskDiagThreads 3
• maxBufferDescs 2m	• cipherList AUTHONLY	• [gss_ppc64]	• [ems]	• [common]
• nsdpQueues 64	• nsdRAIDEventLogToConsole all	• nsdRAIDBufferPoolSizePct 90	• verbsPorts mlx5_1/1/1/0 mlx5_0/1/100/2 mlx5_0/1/101/2 mlx5_0/1/102/2 mlx5_0/1/103/2 mlx5_0/1/104/2 mlx5_0/1/105/2 mlx5_0/1/106/2 mlx5_0/1/107/2 mlx5_0/1/108/2 mlx5_0/1/109/2	• avoidDirFragments yes
• [common]	• maxGeneralThreads 8192	• nsdRAIDFastWriteFSMetadataLimit 1048576	• [common]	• ccrEnabled yes
• maxblocksize 16m	• [gss_ppc64]	• nsdRAIDMediumWriteLimitPct 0	• minReleaseLevel 5.0.1.0	• maxTokenServers 162
• tscCmdPortRange 60000-61000	• nsdRAIDReconstructAggressiveness 0	• nsdRAIDMinMetadataFlushThreads 32	• [gss_ppc64]	• mnodeLocalAccessTimeout 10800
• [ems,gss_ppc64,util]	• [ems,util]	• nsdpThreadsPerQueue 4	• numactlOptionC 24-63 88-127	• [gss_ppc64]
• numaMemoryInterleave yes	• verbsSendBufferMemoryMB 1024	• verbsRdmaSendEnableCallback yes	• [common]	• nsdRAIDDefaultIOTimeout 30000
• maxFilesToCache 128k	• [gss_ppc64]	• [util]	• systemLogLevel notice	• [common]
• [gss_ppc64]	• nsdRAIDBlockDeviceNrRequests 0	• maxReceiverThreads 64	• [gss_ppc64]	• nsdRAIDDiskDiscoveryInterval 3600
• maxMBps 30000	• nsdRAIDBlockDeviceQueueDepth 0	• [common]	• verbsRdmassPerConnectionOverride 4	• tscWorkerPool 2048
• [ems,util]	• nsdRAIDBlockDeviceMaxSectorsKB 0	• remoteCommandFanout 256	• pagepool 384G	• numTscWorkerPools 128
• maxMBps 10000	• nsdRAIDBlockDeviceScheduler off	• nsdRAIDSSDPerformanceMinLimitPct 10	• [util]	• logRecoveryThreads 256
• [ems,gss_ppc64,util]	• nsdMaxWorkerThreads 3842	• nsdRAIDDiskPerformanceMinLimitPct 50	• pagepool 128g	• trace all 4 tm 2 thread 1 mutex 1 vnode 2 ksvfs 3 klockl 2 io 3 pgallo 1 mb 1 lock 2 fsck 3
• workerThreads 1024	• nsdMinWorkerThreads 3842	• subnets 10.39.0.0/alpine.ccs.ornl.gov;dtm.ccs.ornl.gov 10.41.0.0/summit.olcf.ornl.gov;peak.olcf.ornl.gov	• [ems]	• traceRecycle global
• [ems,util]	• nsdRAIDThreadsPerQueue 15	• [gss_ppc64,util]		• tracedevWriteMode overwrite
• ignorePrefetchLUNCount yes	• nsdRAIDSmallThreadRatio 1	• verbsPortsOutOfOrder mlx5_0 mlx5_2		• adminMode central

Core isolation

- To meet compute jitter requirements we had to put OS processes in “*core isolation*” using cgroups
 - Take 1 core from each P9 CPU (2 cores per node) and put all system processes on those
 - Leaves 42 cores for compute jobs (works well with 6 GPUs per node...)
- This impacts GPFS performance at low thread counts
 - SMT1 with core isolation on (2 writer threads): 5.34GB/s
 - SMT1 with core isolation off (2 writer threads): 10.74 GB/s
 - SMT4 with core isolation on (2 writer threads): 8.47 GB/s
- Performance not significantly impacted at large thread counts
- Tunable at job runtime
 - #BSUB -alloc_flags "smt4 isolategpfs"

Sequential IOR at 2.5TB/s

- Final IOR run script

```
#BSUB -nnodes 504                # Number of nodes to use in the job

#BSUB -W 240                     # wallclock -W [hour:]minute[/host_name | /host_model]

#BSUB -alloc_flags "smt4 isolategpfs"  # Isolate GPFS processes and configure for SMT4


BDIR="/gpfs/$MOUNT/stf002/scratch/leverman/alpine_acceptance"

TDIR="$BDIR/ior_testdir"

ITERS=3

BSIZE="7168g"


jsrun -n 504 -c ALL_CPUS -a 1 ${BDIR}/build/ior_timed_patched/ior-master-bh/src/ior -L -g -d 360 -o ${TDIR}/POSIX_fpp_ior -F -i 3 -b
${BSIZE} -t 16m -w -r -a POSIX -e -v -v

date

exit 0
```

Random IOR at 2.2TB/s

- This test just worked from all the tuning done to seq IOR
 - The only deviation is that we need to prefetch more heavily:
 - `tschfs alpine --prefetch-buffers 447`
 - This gets set once at file system creation and there is no `mmchconfig` setting ☹
- Requirements:
 - 2.2TB/s sustained read/write
 - 3 iterations
 - 20 minute runs
 - IO size set to file system block size

Random IOR at 2.2TB/s

- Final IOR run script

```
#BSUB -nnodes 504                # Number of nodes to use in the job

##BSUB -n 160                    # min_proc[,max_proc]

#BSUB -alloc_flags "smt4 isolategpfs"  # Isolate GPFS processes and configure for SMT4


BDIR="/gpfs/$MOUNT/stf002/scratch/leverman/alpine_acceptance"

TDIR="$BDIR/ior_testdir"

ITERS=3

BSIZE="48g"


jsrun -n 504 -c ALL_CPUS -a 168 ${BDIR}/build/ior_timed_patched/ior-master-bh/src/ior -L -g -d 360 -o ${TDIR}/POSIX_fpp_ior -z -F -
i ${ITERS} -b ${BSIZE} -t 16m -w -r -a POSIX -e -v -v

date

exit 0
```

32KiB IO test

- This test was designed to stress the system with a non-ideal IO size (not a nice hero run with 16MiB IOs)
- Requirements:
 - 1.05m IO/s sustained with 32KiB files
 - 3 iterations
 - 20 minute runs

32KiB IO test

- Performance issues exposed with traditional RAID config (separate RAID6 sets for metadata)
 - Moved to DataAndMetadata pools on the declustered RAID set
- Single malfunctioning disk can cause a huge performance degradation
 - (1m IO/s -> 600k IO/s) or (2.5TB/s -> ~1.6TB/s for IOR tests)
 - Need to run mdtest once to identify slow nodes, and then run again excluding those
 - Slow disks on ESS being used for metadata cause client log performance issues
- Disk hospital needs time to collect stats on disks before marking them as slow
 - We had to find and fail disks because we didn't have time

32KiB IO test

- Final mdtest run script

```
#BSUB -nnodes 630                # Number of nodes to use in the job
```

```
#BSUB -alloc_flags "smt4 isolategpfs"    # Isolate GPFS processes and configure for SMT4
```

```
MOUNT=$(pwd | awk -F/ '{print $3}')
```

```
BINDIR="/gpfs/$MOUNT/stf002/scratch/leverman/alpine_acceptance"
```

```
OUTDIR="$BINDIR/${LSB_JOBID}_md_test"
```

```
jsrun -n 630 -c ALL_CPUS -a 20 -X 1 $BINDIR/build/mdtest_build/mdtest -n 30000 -w 32768 -F -u -C -r -i 3 -p 360 -y -u $OUTDIR
```

#-C: only create files/dirs

#-F: perform test on files only (no directories)

#-i: number of iterations the test will run

#-n: every task will create/stat/remove # files/dirs per tree

#-p: pre-iteration delay (in seconds)

#-r: only remove files/dirs

#-u: unique working directory for each task

#-w: number of bytes to write to each file

#-y: sync file after write completion

Misc issues/tunables

- Network tunable changes to help with fsck expels.
 - No fsck checkpointing, run from command line?, online fsck not feature rich
- Diskless configuration makes rolling out changes (efixs/PTFs) trivial
- 2 metadata replicants
 - Half of the system in different failure group
 - Not default ESS config, but needed for performance
 - Haven't tested performance impact

Misc issues/tunables

- Creating a 30B inode file system was problematic
 - Needed to increase preallocated inodes in groups of 2B (otherwise performance impact)
 - `mmchfs alpine --inode-limit 4000m:4000m`
 - `mmchfs alpine --inode-limit 6000m:6000m`
 - `mmchfs alpine --inode-limit 8000m:8000m`
 - `mmchfs alpine --inode-limit 10000m:10000m`
 - `mmchfs alpine --inode-limit 12000m:12000m`
 - `mmchfs alpine --inode-limit 14000m:14000m`
 - `mmchfs alpine --inode-limit 16000m:16000m`
 - `mmchfs alpine --inode-limit 18000m:18000m`
 - `mmchfs alpine --inode-limit 20000m:20000m`
 - `mmchfs alpine --inode-limit 30000m`
- CPU tuning:
 - Disable CPU sleep states 1 and 2
 - Set minimum frequency to the max of the CPU
 - Set SMT to 4 (default)

Questions?

50k creates per second

- Need the ability to quickly create files in a shared directory
- Requirements:
 - 50,000 creates per second
 - Empty files (0B)
 - Shared directory
 - 3 iterations
 - 20 second run time

50k creates per second

- The performance issues are when you need to split the directory
- New options in GPFS to more aggressively split to avoid the overhead (better to split more aggressively up front than to split several times)
 - Didn't yield significant performance gains
- Currently seeing ~36k creates/s

50k creates per second

- Final mdtest run script

```
#BSUB -nnodes 24                # Number of nodes to use in the job

#BSUB -alloc_flags "smt4 isolategpfs"  # Isolate GPFS processes and configure for SMT4

MOUNT=$(pwd | awk -F/ '{print $3}')

BINDIR="/gpfs/$MOUNT/stf002/scratch/leverman/alpine_acceptance"

module load gcc

jsrun -n 24 -c ALL_CPUS -a 1 -X 1 $BINDIR/build/mdtest_build/mdtest -n 50000 -w 0 -F -T -C -r -i 1 -v $OUTDIR

#-C: only create files/dirs                #-p: pre-iteration delay (in seconds)
#-F: perform test on files only (no directories)
#-i: number of iterations the test will run
#-n: every task will create/stat/remove # files/dirs per tree
#-r: only remove files/dirs
#-w: number of bytes to write to each file
#-T: only stat files/dirs
```

Glossary of acronyms:

- ORNL: Oak Ridge National Laboratory
- DDN: Data Direct Networks (storage vendor)
- ESS: Elastic Storage Server – an IBM storage appliance
- OSS: Object Storage Server (a component of a Lustre file system that serves object data to clients).
- SATA: Serial ATA – a hard disk interconnect technology
- SAS: SCSI Attached Storage
- NL-SAS: Near-Line SAS – disk technology that uses commodity SATA disk parts, but has a SAS disk controller.
- IB: InfiniBand – High speed interconnect technology
- DDR: Double Data Rate InfiniBand – 4 lanes of 2.5GB/s IB traffic
- FDR: Fourteen Data Rate InfiniBand – 4 lanes of 14GB/s IB traffic
- EDR: Enhanced Data Rate InfiniBand – 4 lanes of 25GB/s IB traffic
- RDHX: Rear-Door Heat Exchanger – a device that mounts to the back of a compute cabinet that has chilled water running through it so when heated air gets cooled as it passes through the device.
- SSU: Scalable Storage Unit – A single storage unit (like a DDN controller pair, or an ESS controller pair)
- SSC: Scalable Storage Cluster: The smallest unit of hardware that can operate independently (example: 8x ESSs, 2x 36-port TOR (Top Of Rack) switches, and 1x Ethernet switch).
- xCAT: system provisioning software. Supports diskfull and diskless provisioning (state-lite and stateless).
- NVMf: short for NVMe over Fabrics, which is a solid state storage device that plugs into the PCI bus on your system.
- VM: Virtual Machine – a virtual OS instance that sits on top of a hypervisor.
- SR-IOV: Single-Root I/O Virtualization: allows a PCI device (like a network card) to serve access to that device to virtual OS instances (kinda).