BeeGFS

Sizing and Tuning

BeeGFS.io

2022

Agenda

- Sizing
  - Characteristics and requirements of each BeeGFS service
  - General design guidelines
  - Performance evaluation

- Tuning
  - General tuning options
  - Service-specific tuning options

- Benchmark Results

Definitions

- Sizing
  - Design a system to meet a list of requirements
    - Performance
    - Scalability
    - Availability
    - Cost

- Tuning
  - Get the best possible performance, scalability, availability from given hardware for a defined workload

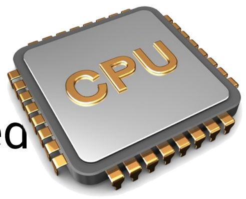
Service Requirements and Characteristics

- Management service
  - No special requirements
  - Typically runs on cluster master node or login node
  - Can also run on one of the storage servers

- Management target
  - Capacity
    - a little more than 1 MB
    - A floppy disk would be enough
  - Typically directory on the OS partition is used

Service Requirements and Characteristics

- Metadata service
  - CPU
    - Single or dual socket
    - Large number of cores for low latency prefered over fewer cores with higher clockspeed
    - Large number of cores for systems with lots of clients



  - RAM
    - RAM is used for caching – the more, the better
    - Server RAM needed for connections:
      - Per client: connRDMABufSize x connRDMABufNum x 2 x connMaxInternodeNum = ~12 MB

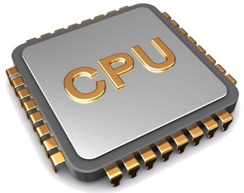
  - Network
    - Type often defined by existing infrastructure
    - Low latency important for metadata access

Service Requirements and Characteristics

- Metadata target
  - RAID protection
    - Avoid RAID-5 or RAID-6 write overhead with parity update
    - RAID 1 or 10 are common choices
    - RAID chunk size ~64 KB – depending on disk type
    - Hardware or software RAID
      - Hardware RAID has advantage of battery-backed write cache
  - Disks
    - SSDs and NVMes are widely used today
    - ext4 as local file system
  - Capacity
    - 0.3% to 0.5% of the total storage space as best practice

Service Requirements and Characteristics

- Storage Service
  - CPU
    - Single or dual socket – depending on hardware/software RAID and network
    - With hardware RAID and InfiniBand no high CPU load observed
      - Mid-range CPUs are sufficient



  - RAM
    - RAM is used for caching – the more, the better
    - Server RAM needed for connections:
      - Per client: connRDMABufSize x connRDMABufNum x 2 x connMaxInternodeNum = ~12 MB

  - Network
    - Type often defined by existing infrastructure
    - Low latency important for metadata access

Service Requirements and Characteristics

- Storage Targets
  - RAID protection
    - RAID-6 is the most common choice
    - For HDD: Around 6 disks for workloads characterized by small writes
    - For HDD: Around 12 disks for workloads characterized by large writes
    - NVME and ZFS: Performance loss if more than 6 disks in a raidz2
    - RAID chunk size between 128 KB and 512 KB – depending on workload and controller
    - Software RAID: RAID6 Performance with ZFS is better than with Linux MD RAID
    - Software or hardware RAID
      - Hardware RAID has advantage of battery-backed write cache

Service Requirements and Characteristics

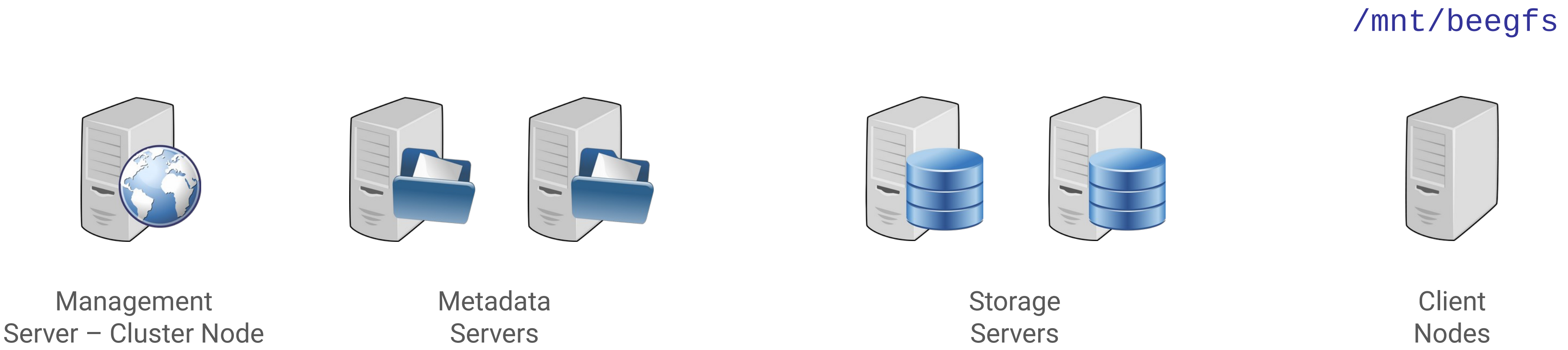
- Storage Targets
  - Disks
    - SSD or NVMe more often than HDD
    - (Nearline) SAS performs better than SATA and has better error detection features and error handling
    - HDD or SSD: with hardware RAID XFS as local file system
    - NVMe: ZFS as local file system, because of RAID capabilities

RAID with NVMe disks

- Metadata and Storage Targets
  - Metadata
    - Linux Software RAID in RAID1 or RAID10
    - Or ZFS mirror
    - Software RAID1 with xfs has better performance than ZFS mirror
  - Storage
    - RAID6 with Linux Software RAID has very bad performance
    - Better with ZFS raidz2
    - No more than 6 disks in a raidz2 because of performance loss

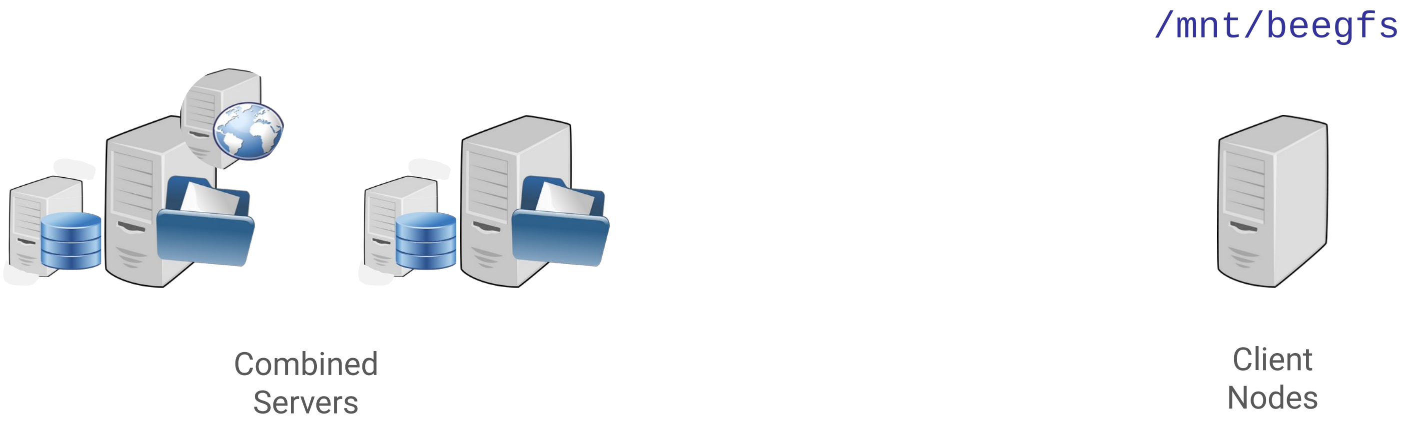
General Guidelines

- Dedicated servers
  - Performance-critical systems
  - More expensive
  - Prevent storage workload from impacting metadata service
    - Eviction of cached metadata from memory
    - Increase of metadata latency
  - Prevent client applications from impacting server services



General Guidelines

- Combined servers
  - Non performance-critical systems
  - Low cost
  - Most common design
  - Negative impact may be minimal
  - Prevent client applications from impacting server services



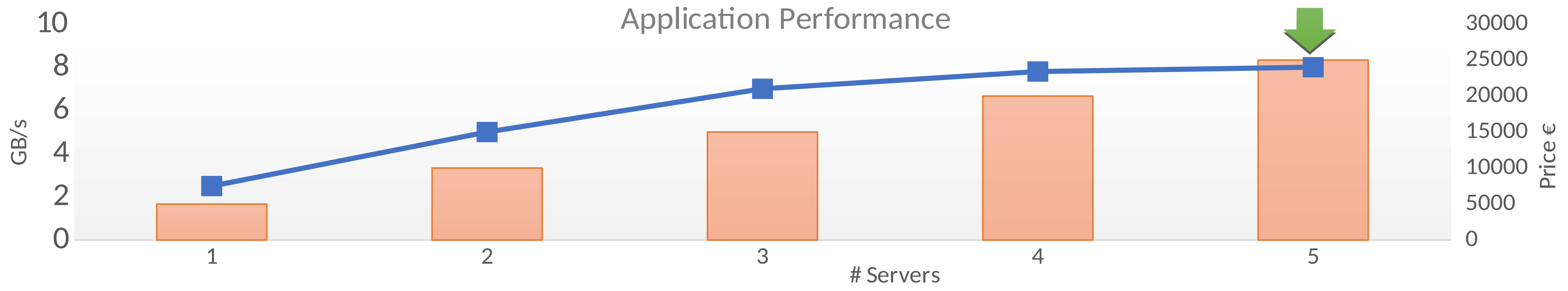
General Guidelines

- Hyper Converged solution
  - Non performance-critical systems
  - Client applications may impact the servers
  - Low cost
  - Client services don’t use network to communicate with local server services
  - Servers and client services scale at the same proportion

Hyper-converged
Nodes

General Guidelines

- Define building blocks and scale them out until price/performance breaks down



Servers

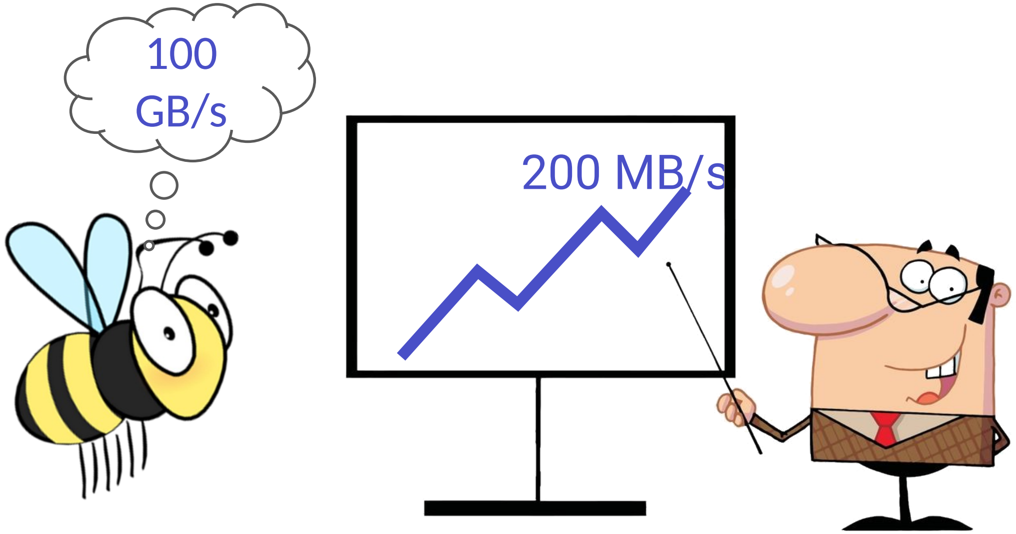
General Guidelines

- Use hot-spare drives when possible, keep at least cold spares handy
- Redundant power supply should be used
- Network bandwidth higher than disk bandwidth

General Guidelines

- When estimating system throughput
  - Take all system components into consideration
  - Use conservative points of reference

Component	Throughput
Old HDD	75 MB/s
New HDD	100 MB/s
Old SSD	300 MB/s
New SSD	400 MB/s
NVMe	1.5 GB/s
Disk Controller	2.5 GB/s
10 GibE	1 GB/s
Mellanox DDR	1.5 GB/s
Mellanox QDR	3 GB/s
Mellanox FDR	6 GB/s
Mellanox EDR	11 GB/s



General Guidelines

- Example
  - User requirements
    - Minimal data streaming throughput: 6 GB/s
    - System optimized for small writes, and non-sequential IO
    - Metadata performance less important but should not be impacted by the storage service

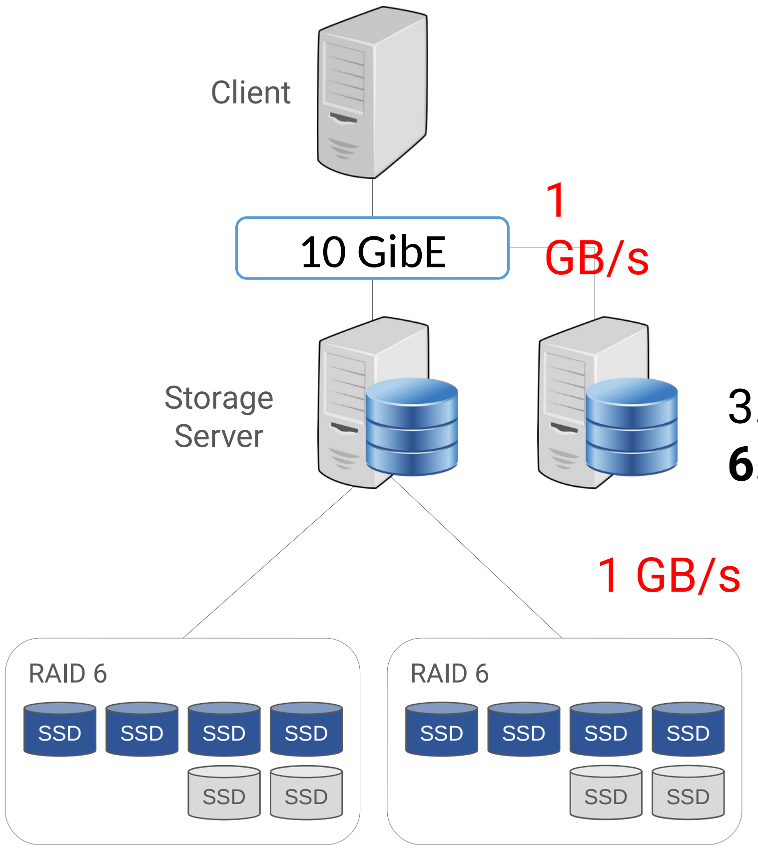
General Guidelines

- Example
  - Dedicated servers
  - Storage server
    - CPU: Intel Xeon 2.40 GHz
    - RAM: 128 GB DDR4
    - 2 storage targets
      - 2 X 6 X 1 TB SSD, RAID 6, XFS
      - SSD throughput: 400 MB/s
  - 2 storage servers
  - Bottleneck: Network: 10 GibE

6 X 500 MB/s=3.0 GB/s

(6 – 2) X 400 MB/s = **1.6 GB/s**

6 X 400 MB/s = 2.4 GB/s ?



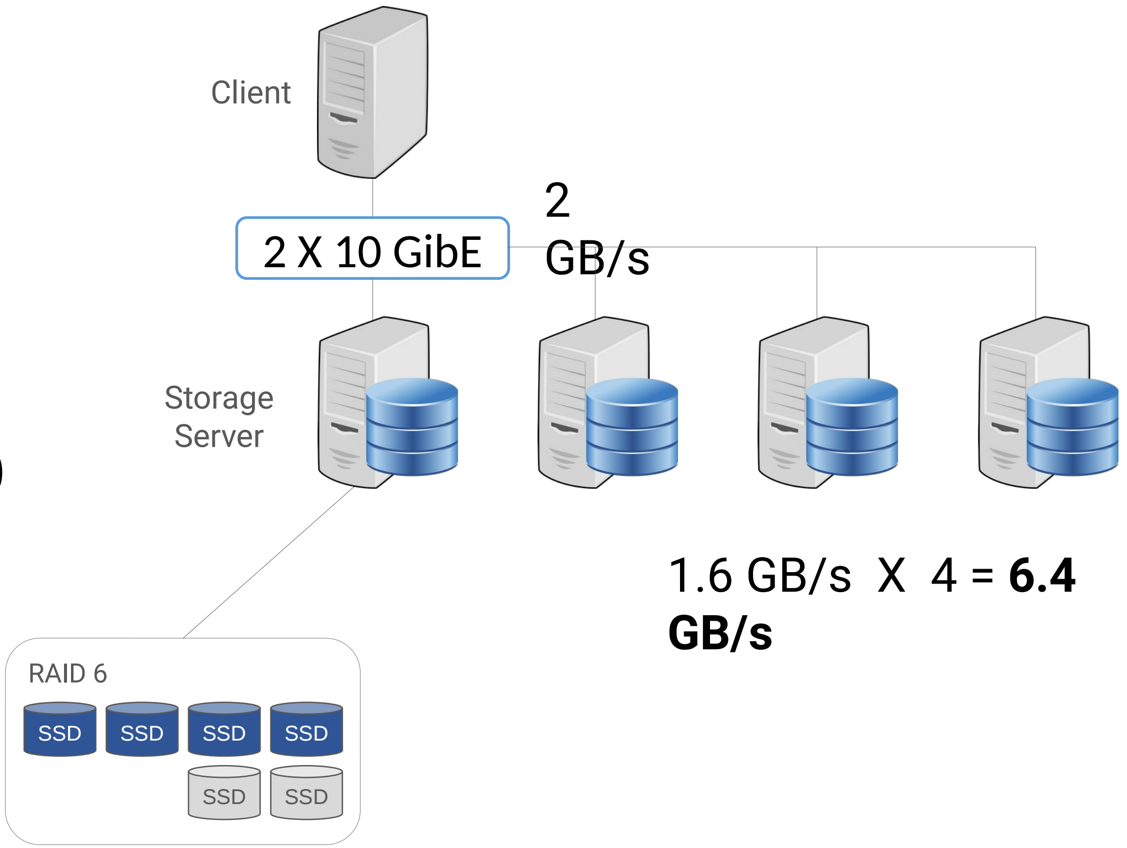
3.2 GB/s X 2 = **6.4 GB/s**

1 GB/s X 2 = 2.0 GB/s

1.6 GB/s X 2 = **3.2 GB/s**

General Guidelines

- Example
  - Dedicated servers
  - Storage server
    - CPU: Intel Xeon 2.40 GHz
    - RAM: 128 GB DDR4
    - 1 storage target
      - 1 X 6 X 1 TB SSD, RAID 6, XFS
      - SSD throughput: 400 MB/s
  - Network: 2 X 10 GibE (Network interface bonding)
  - 4 storage servers

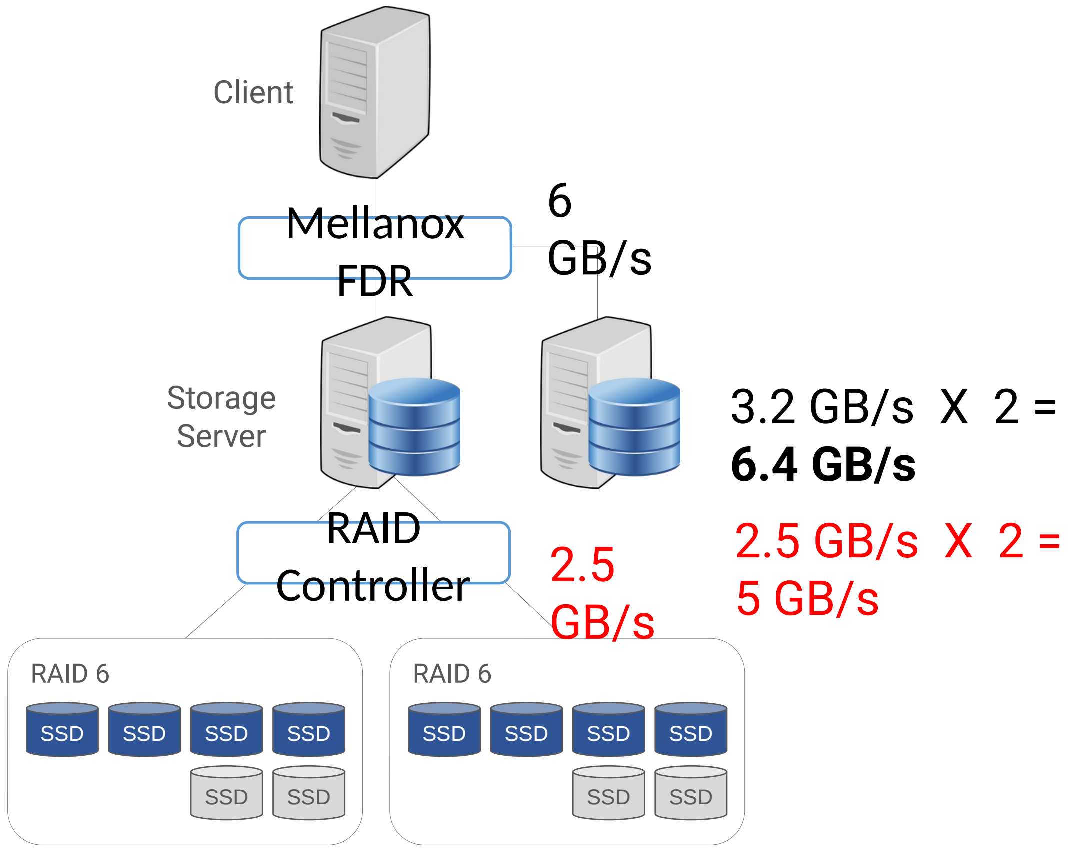


1.6 GB/s X 4 = **6.4 GB/s**

(6 – 2) X 400 MB/s X 1 = **1.6 GB/s**

General Guidelines

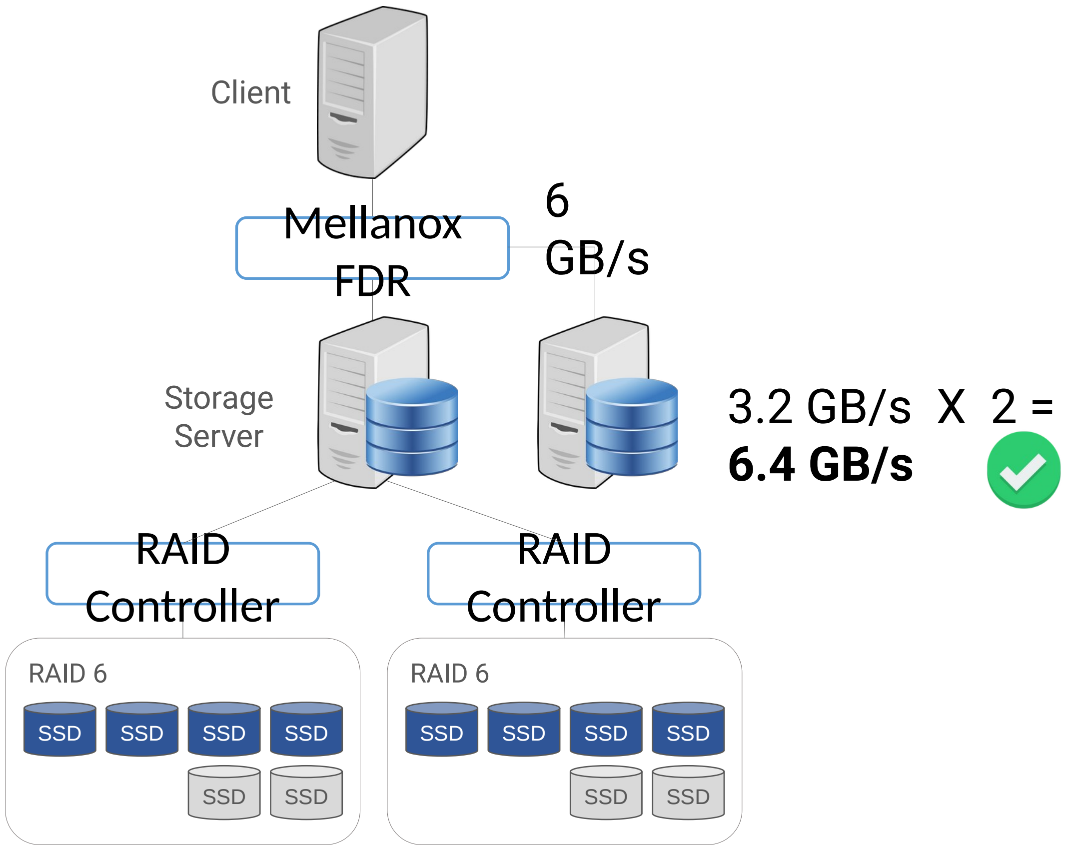
- Example
  - Dedicated servers
  - Network: Mellanox FDR
  - Storage server
    - CPU: Intel Xeon 2.40 GHz
    - RAM: 128 GB DDR4
    - 2 storage targets
      - 2 X 6 X 1 TB SSD, RAID 6, XFS
      - SSD throughput: 400 MB/s
  - 2 storage servers
  - New bottleneck: RAID controller



(6 – 2) X 400 MB/s X 2 = **3.2 GB/s**

General Guidelines

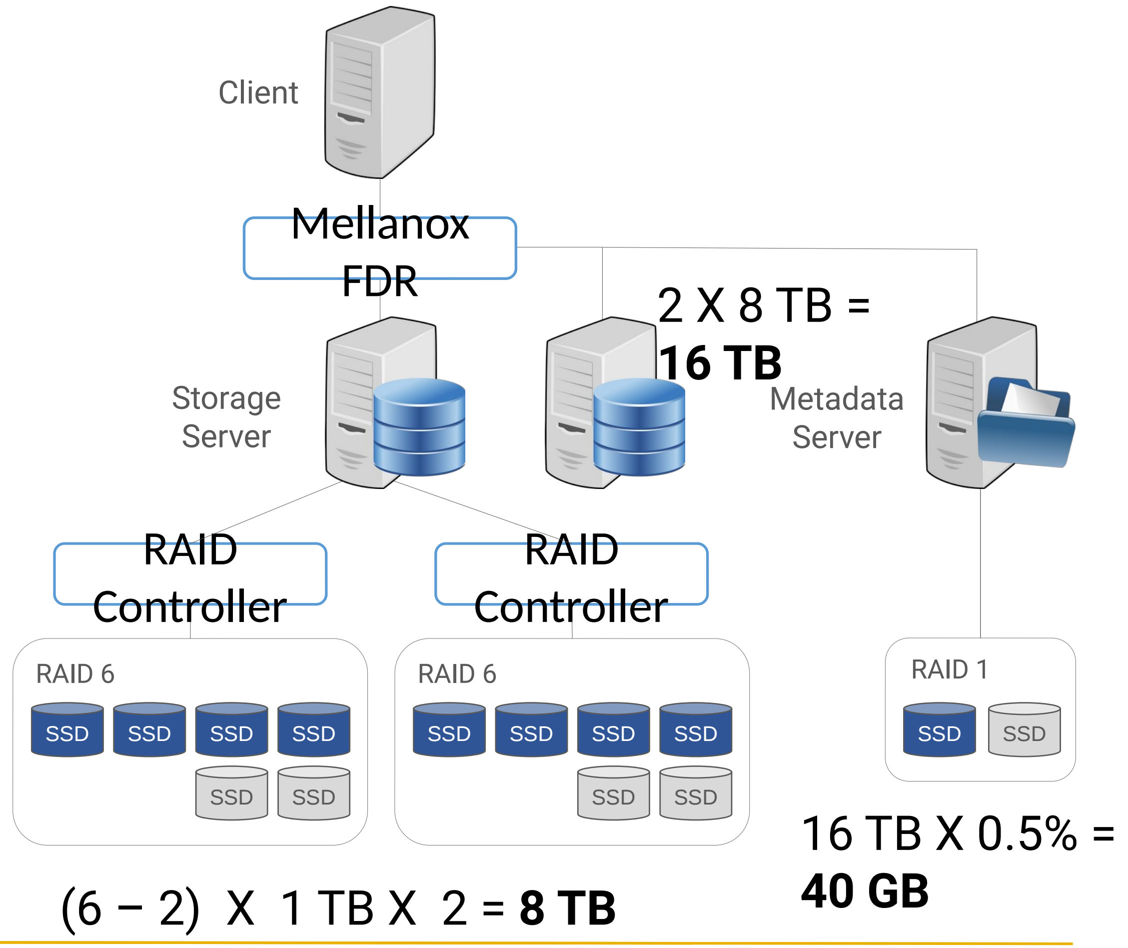
- Example
  - Dedicated servers
  - Network: Mellanox FDR
  - Storage server
    - CPU: Intel Xeon 2.40 GHz
    - RAM: 128 GB DDR4
    - 2 storage targets
      - 2 X 8 X 1 TB SSD, RAID 6, XFS
      - SSD throughput: 400 MB/s
    - 2 RAID controllers
  - 2 storage servers



(6 – 2) X 400 MB/s X 2 = **3.2 GB/s**

General Guidelines

- Example
  - Dedicated servers
  - Network: Mellanox FDR
  - Metadata server
    - CPU: Intel Xeon 3.6 GHz
    - RAM: 128 GB DDR4
    - 1 metadata target
      - 2 X 100 GB SSD, RAID 1, ext4



Performance Evaluation

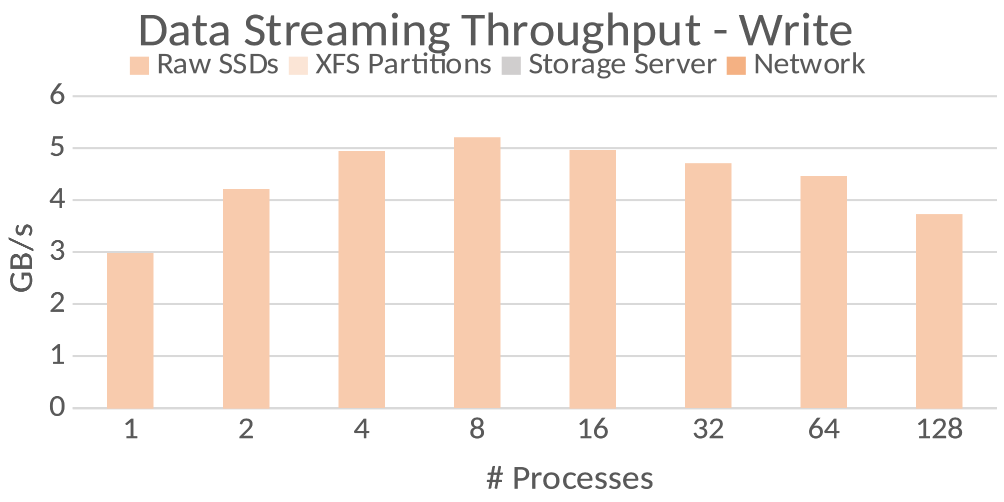
Performance Evaluation

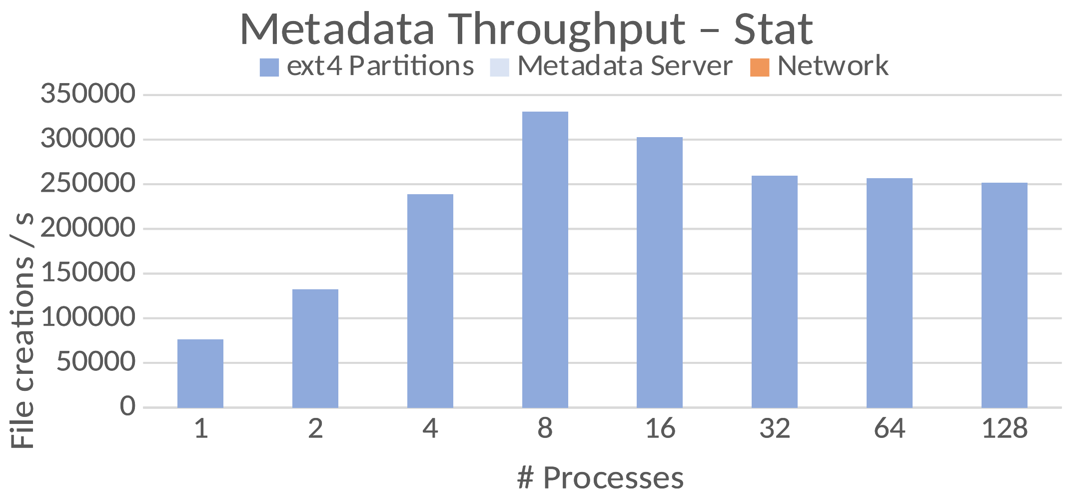
IOZone

IOR

mdtest

io500





StorageBench

NetBench

fio

qperf

Built-in Benchmark Tools

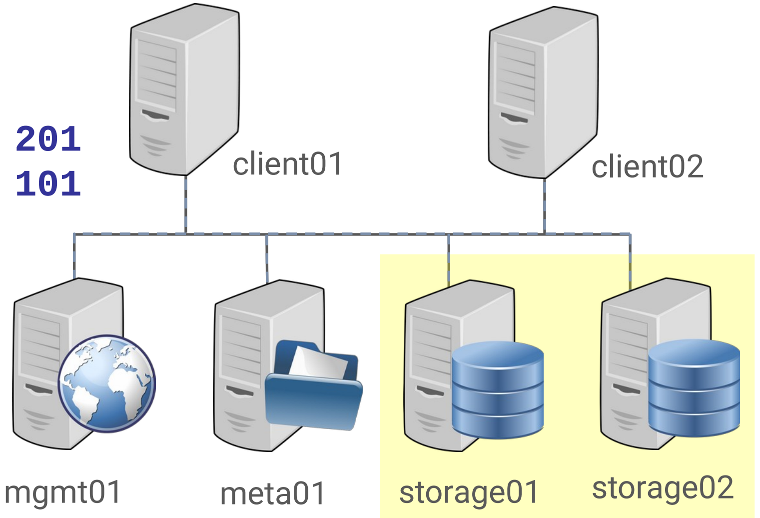
- StorageBench
  - Data streaming throughput
  - Focus on the storage servers, no use of network

No caching effect: 2.5 X RAM

Testing caching: 50% X RAM

```
client01:~ # beegfs-ctl --storagebench --write --blocksize=512K --size=640G --threads=4 --alltargets
client01:~ # beegfs-ctl --storagebench --read  --blocksize=512K --size=640G --threads=4 --alltargets
```

```
client01:~ # beegfs-ctl --storagebench --status --alltargets

Server benchmark status: Running:    4

Read benchmark results:
Max throughput:       1762 MiB/s nodeID: storage02 [ID:2], targetID: 201
Min throughput:       1507 MiB/s nodeID: storage01 [ID:1], targetID: 101
Avg throughput:       1683 MiB/s
Aggregate throughput: 6890 MiB/s
```



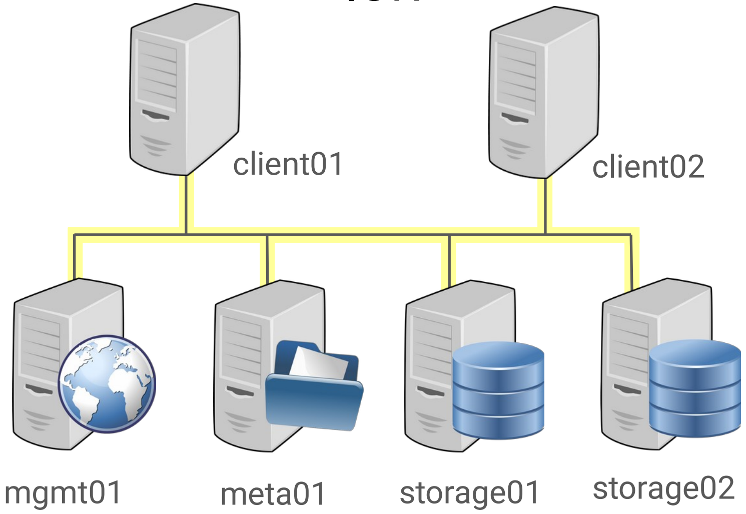
  - Used initally to confirm that all targets are working normally (e.g. no bad disks)

Built-in Benchmark Tools

- NetBench mode
  - Network streaming throughput
  - Roughly equivalent to writing to a RAM drive
  - You need to run IOR or another tool to get results

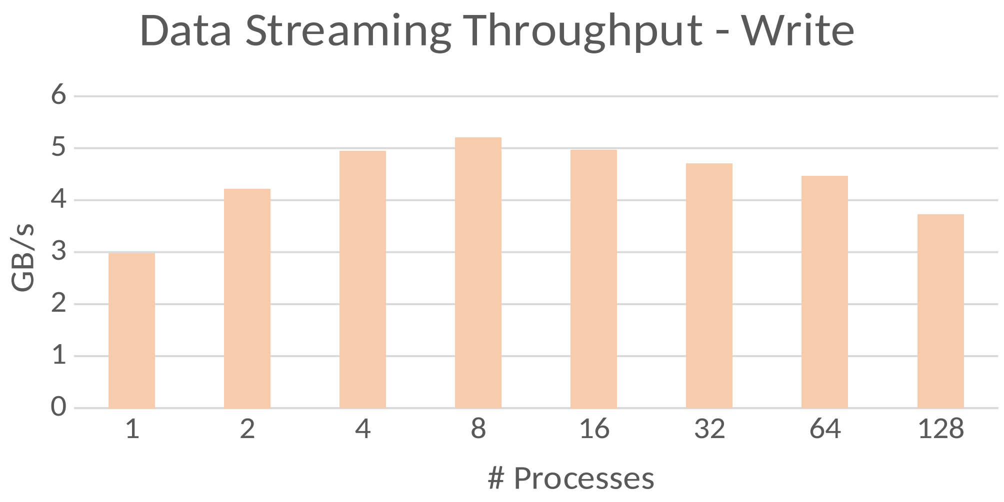
```
client01:~ # $ echo 1 > /proc/fs/beegfs/<clientID>/netbench_mode
...
client01:~ # $ echo 0 > /proc/fs/beegfs/<clientID>/netbench_mode
```

fio
IOZone
IOR



External Benchmark Tools

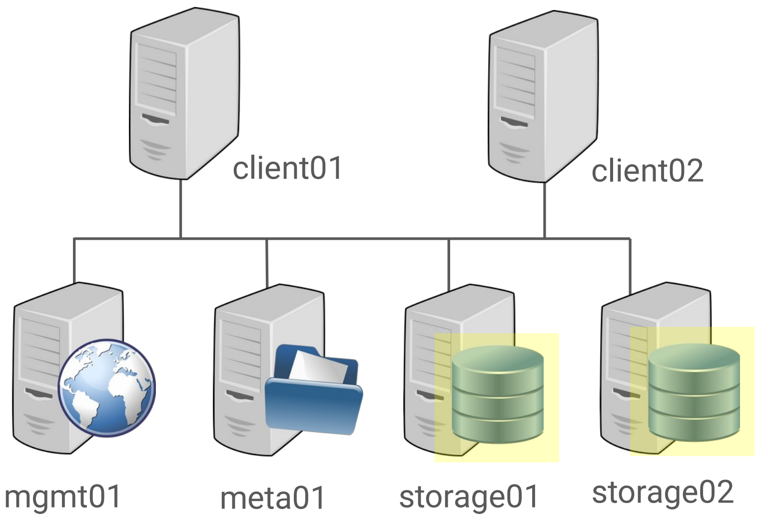
- IOZone
- MDtest
- IOR
- Fio
- ElBencho
- qperf



Performance Evaluation

- IOZone - www.iozone.org
  - Local file system performance
  - Reveal controller configuration issues

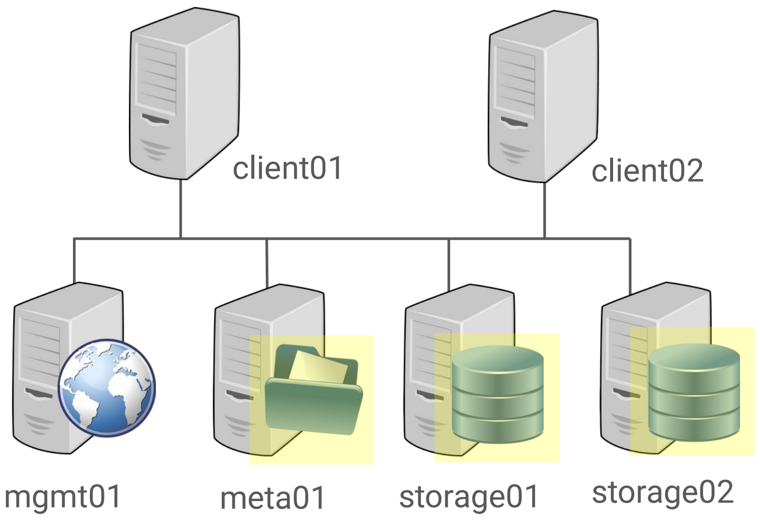
```
# iozone -i 0 -i 1 -c -e -w -r 1m -s 40g -t 64 -+n
  Throughput test with 32 processes
  Each process writes a 83886080 kByte file in 1024 kByte records

  Children see throughput for 32 writers  = 7310362.51 kB/s
  Parent sees throughput for 32 writers   = 7157739.27 kB/s
  Min throughput per process              =  205000.02 kB/s
  Max throughput per process              = 251255.08 kB/s
  Avg throughput per process              =  243446.33 kB/s
  Min xfer                                = 77416132.01 kB

  Children see throughput for 32 readers  = 5242248.32 kB/s
  Parent sees throughput for 32 readers   = 5333933.23 kB/s
  Min throughput per process              =  108385.88 kB/s
  Max throughput per process              =  298969.99 kB/s
  Avg throughput per process              =  180185.26 kB/s
  Min xfer                                = 32857495.01 kB
```



Performance Evaluation

- Flexible I/O Tester (fio) Benchmark
  - Raw block device performance
  - File System performance
  - **fio --rw=readwrite --name=test --size=1G --direct=1 --bs=128k**

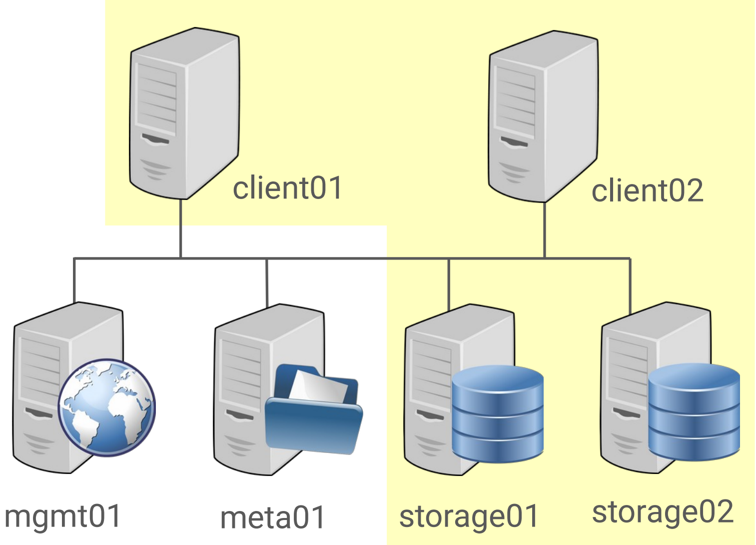
```
test: (g=0): rw=rw, bs=(R) 128KiB-128KiB, (W) 128KiB-128KiB, (T) 128KiB-128KiB, ioengine=psync, iodepth=1
fio-3.7
Starting 1 process
test: Laying out IO file (1 file / 1024MiB)
Jobs: 1 (f=1): [M(1)][100.0%][r=13.9MiB/s,w=12.5MiB/s][r=111,w=100 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=3358: Tue Jun 16 10:13:14 2020
   read: IOPS=102, BW=12.8MiB/s (13.4MB/s)(506MiB/39476msec)
    clat (usec): min=2091, max=65069, avg=3957.43, stdev=3460.56
     lat (usec): min=2091, max=65069, avg=3957.88, stdev=3460.52
    clat percentiles (usec):
     |  1.00th=[ 2212],  5.00th=[ 2311], 10.00th=[ 2343], 20.00th=[ 2409],
     | 30.00th=[ 2474], 40.00th=[ 2573], 50.00th=[ 2999], 60.00th=[ 3523],
     | 70.00th=[ 4359], 80.00th=[ 5145], 90.00th=[ 6063], 95.00th=[ 7111],
     | 99.00th=[12649], 99.50th=[15008], 99.90th=[53216], 99.95th=[53216],
     | 99.99th=[65274]
   bw (  KiB/s): min= 7168, max=17408, per=99.77%, avg=13102.08, stdev=2138.47, samples=78
   iops        : min=   56, max=  136, avg=102.28, stdev=16.66, samples=78
.
.
Run status group 0 (all jobs):
   READ: bw=12.8MiB/s (13.4MB/s), 12.8MiB/s-12.8MiB/s (13.4MB/s-13.4MB/s), io=506MiB (531MB), run=39476-39476msec
  WRITE: bw=13.1MiB/s (13.8MB/s), 13.1MiB/s-13.1MiB/s (13.8MB/s-13.8MB/s), io=518MiB (543MB),
run=39476-39476msec

Disk stats (read/write):
  sde: ios=4043/4136, merge=0/3, ticks=14969/23020, in_queue=37990, util=96.17%
```



Performance Evaluation

- IOR - github.com/LLNL/ior
  - System overall performance – Data streaming

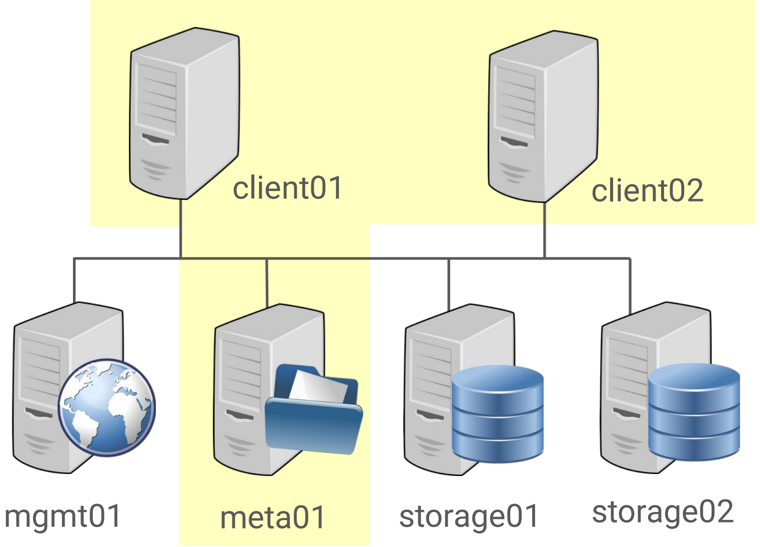
```
# /usr/local/bin/IOR -N 2 -i 5 –g -e -F -C -b 320g -t 1g -w -r -o /mnt/beegfs/testfile

Summary:
    api                  = POSIX
    test filename        = /mnt/beegfs/testfile
    access               = file-per-process
    clients              = 64 (32 per node)
    repetitions          = 5
    blocksize            = 320 GiB
    aggregate filesize = 640 GiB

Operation  Max (MiB)  Min (MiB)  Mean (MiB)  Std Dev ...
---------  ---------  ---------  ----------  ------- ...
write        6088.56    4977.01     6034.04    43.03 ...
read         6190.62    4957.82     6099.51    78.67 ...
Max Write: 6088.56 MiB/sec (6190.01 MB/sec)
Max Read:  6190.62 MiB/sec (6297.03 MB/sec)
```



Performance Evaluation

- mdtest - sourceforge.net/projects/mdtest
  - Metadata throughput
  - More than 1,000,000,000 files in ~33 Minutes

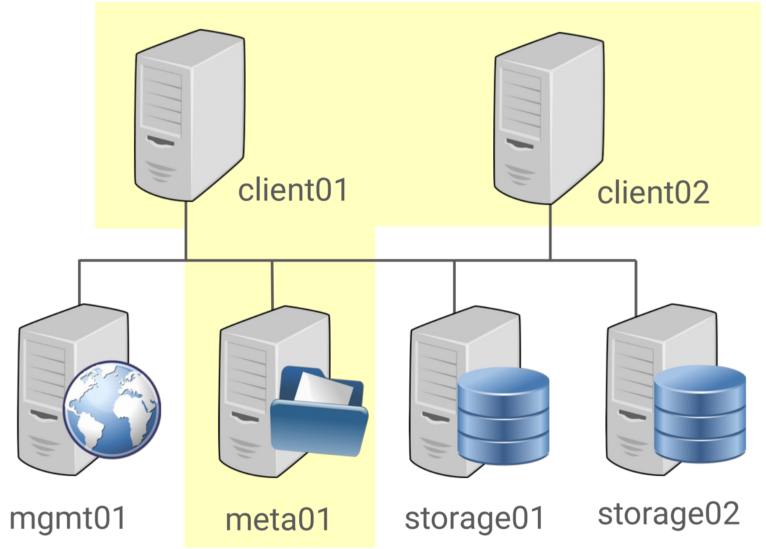
```
# /opt/mdtest-1.9.3/mdtest -C -T -d /mnt/beegfs/mdtest -i 5 -I 122 -z 2 -b 8 -L -u -F
...
Path: /mnt/beegfs
FS: 218.2 TiB   Used FS: 0.6%   Inodes: 0.0 Mi   Used Inodes: nan%

128 tasks, 1139968 files

SUMMARY: (of 5 iterations)
   Operation                      Max          Min         Mean     Std Dev
   ---------                      ---          ---         ----     -------
   File creation     :      53987.685    51374.199    52808.687     838.825
   File stat         :     180607.210   172831.261   175919.093    2615.747
   File read         :          0.000        0.000        0.000       0.000
   File removal      :          0.000        0.000        0.000       0.000
   Tree creation     :        120.841       87.121      109.578      12.101
   Tree removal      :          0.000        0.000        0.000       0.000
```



Performance Evaluation

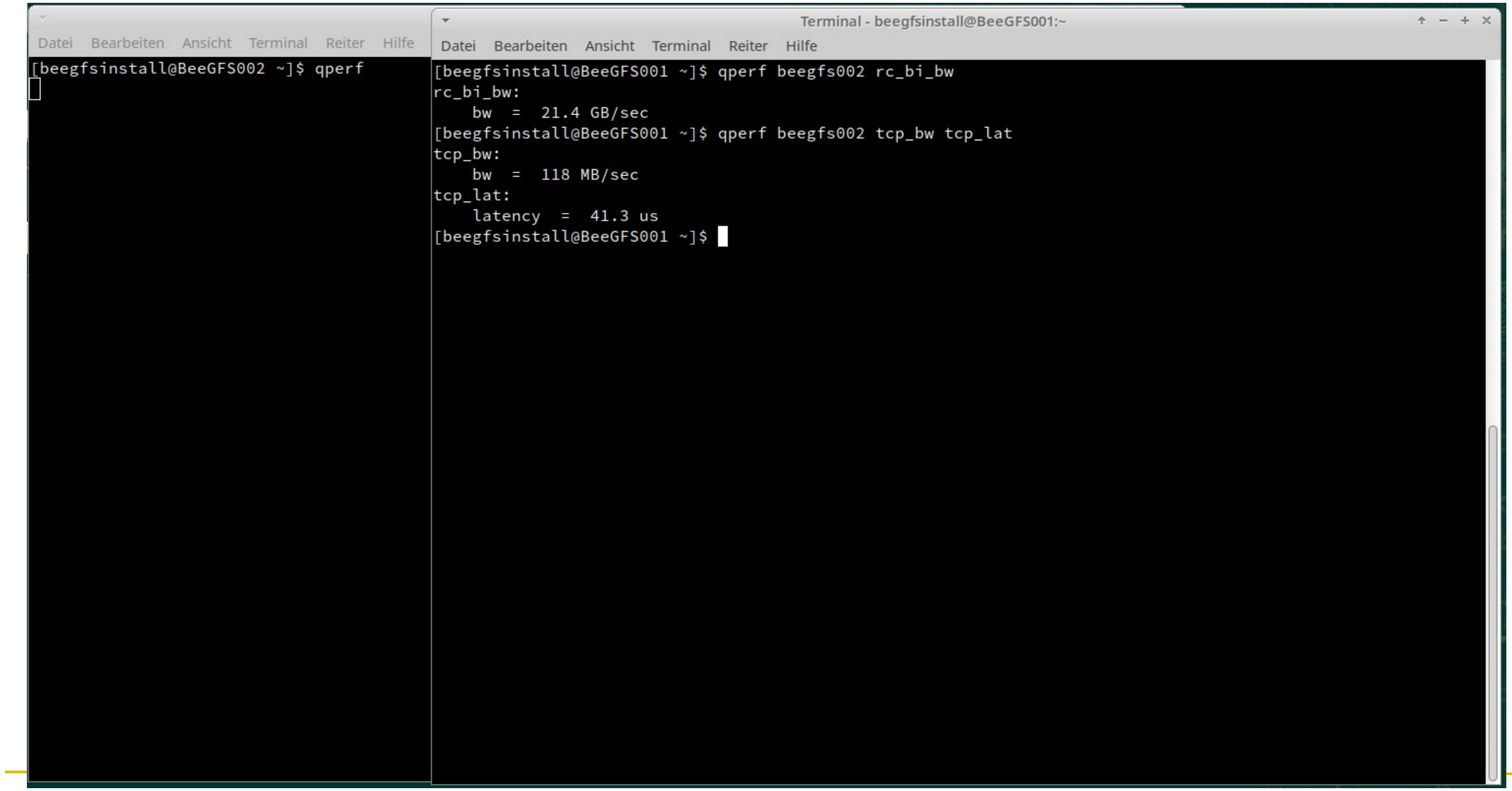
- elbencho - https://github.com/breuner/elbencho#elbencho
    - Metadata throughput
    - Storage throughput

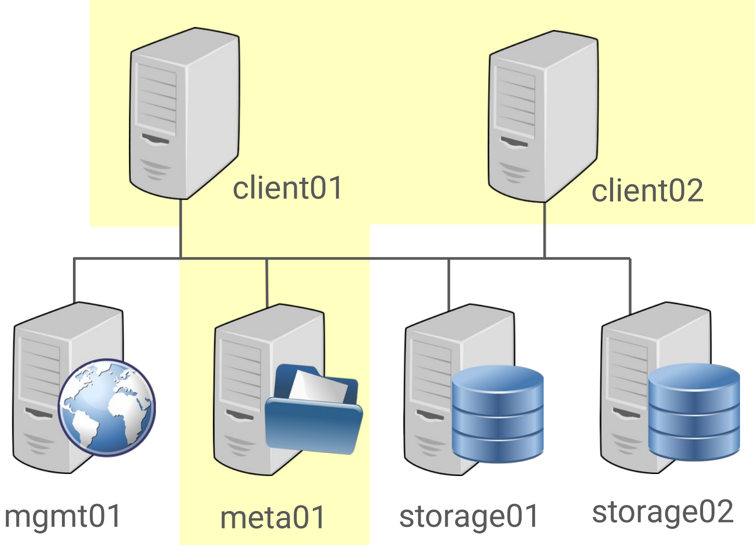
```
# /root/bin/elbencho -t 2 -d -n 16 -w -N 10000 -s 512k -b 512k /mnt/beegfs/testdir
…
OPERATION RESULT TYPE        FIRST DONE  LAST DONE
========= ================   ==========  =========
MKDIRS    Elapsed ms       :          2          6
          Dirs/s           :       7076       4732
          Total dirs       :         16         32
---
WRITE     Elapsed ms       :     135292     135528
          Files/s          :       2361       2361
          Throughput MiB/s :       1180       1180
          Total files      :     319521     320000
          Total MiB        :     159761     160000
```



Performance Evaluation

- qperf
  - RDMA throughput
  - TCP throughput





What if you don‘t make it?

- Tune it!

- Benchmarking and tuning before system goes to production

- Our recommendations:
  - https://www.beegfs.io/wiki/TuningAdvancedConfiguration
    - System BIOS
    - Formatting, mounting
    - IO scheduler, controller
    - Concurrency, caching
    - Network settings
    - Data striping patterns

Tuning Local File System (XFS)

- Underlying file system
  - Align your partitions properly when defining their sizes
  - Formating: mkfs.xfs -d su=128k,sw=8 -isize=512 /dev/sdb
  - mount options
    - noatime (atime not used by BeeGFS)
    - nodiratime (atime not used by BeeGFS)
    - logbufs, logbsize (handle and queue pending file and directory operations more efficiently.)
    - allocsize (increase to reduce the risk of fragmentation for large files)

Tuning Local File System (ext4)

- Underlying file system
  - Align your partitions properly when defining their sizes
  - Formating: mkfs.ext4 -i 2048 -I 512 -J size=400 -Odir_index,filetype /dev/sdb
  - Careful with the number of Inodes.
  - mount options
    - noatime (atime not used by BeeGFS)
    - nodiratime (atime not used by BeeGFS)
    - Nobarrier
  - Please do not use in a buddymirror environment!

Tuning Local File System (ZFS)

- ZFS is not part of RedHat or SUSE repositories
- Download at https://zfsonlinux.org (source code)
- Compile and build RPM packages for your environment
- Underlying file system
  - Don't use ZSF raidz2 with more than 6 disks
  - Align your partitions properly when defining their sizes
  - Formating: zpool create vault raidz2 /dev/sd[b-g]
  - ZFS mounts automatically at /vault
  - ZFS tuning options
    - #>zfs set compression=lz4 vault
    - #>zfs set dedup=off vault
    - #>zfs set atime=off vault
    - #>zfs set xattr=sa dnodesize=auto vault
    - #>zfs set recordsize=<?> „Set according to your environment → benchmark"

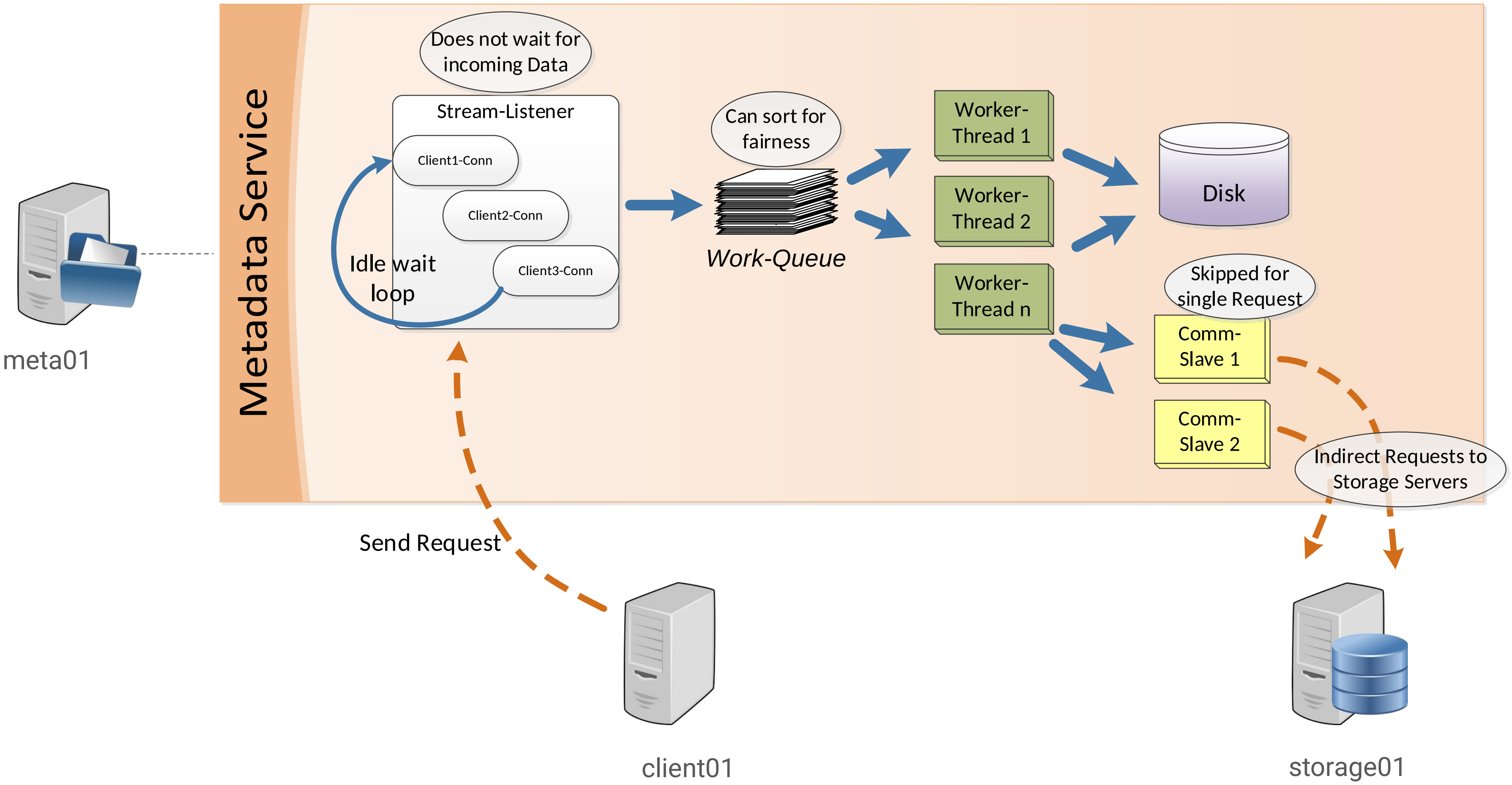
Tuning Local Disk

- Block devices
  - Pick good IO scheduler (noop, [deadline], cfq)
  - For NVME: noop has good results
  - Adjust the number of schedulable requests „nr_requests“ (128, 4096)
  - Set the read-ahead properly (4 … 64 MB)
  - Set „max_sectors_kb“ (maximum IO size) to fit your hardware

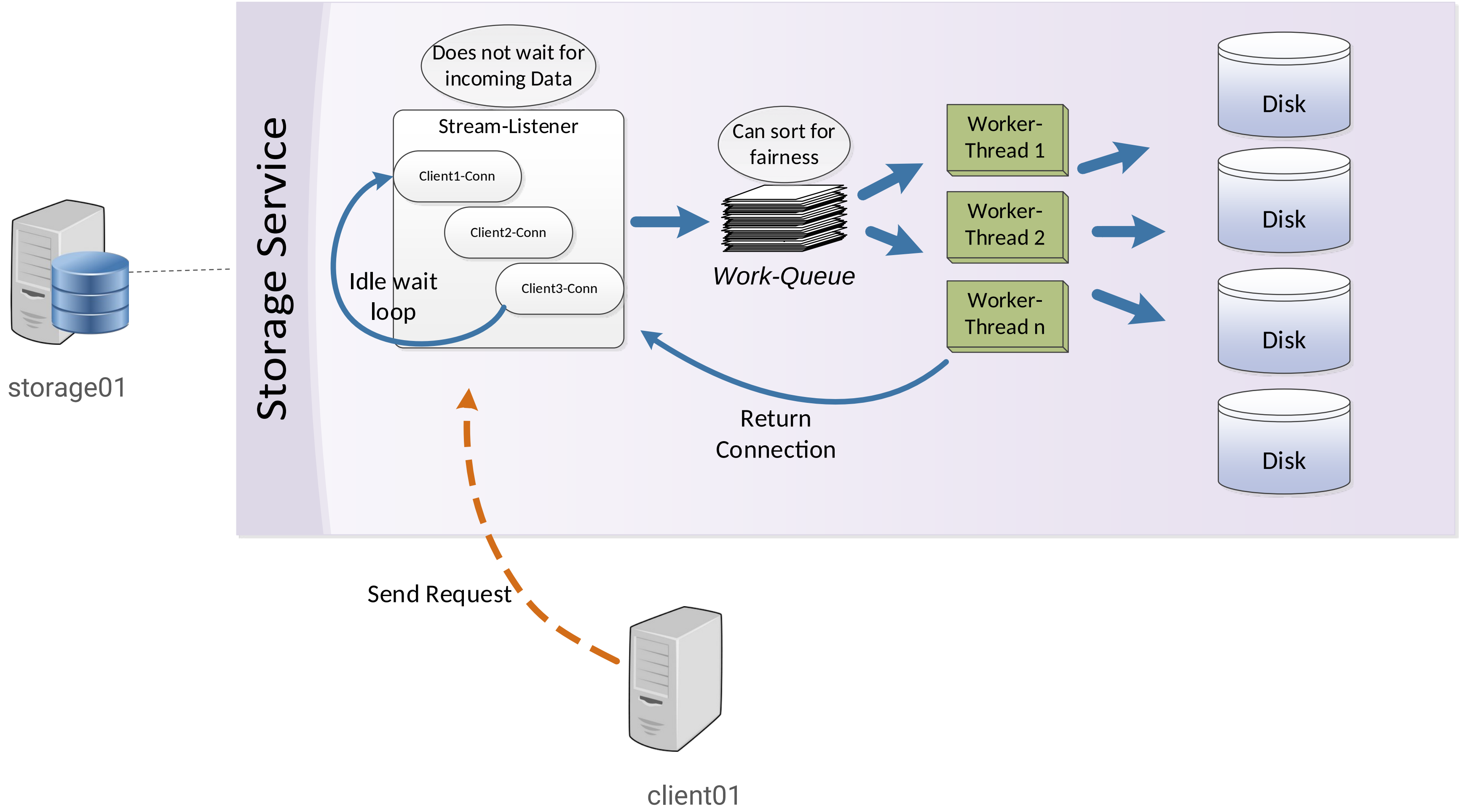
Tuning Memory Management

- File system cache
  - /proc/sys/vm/dirty_background_ratio [1…5]
    - % of memory which when dirty then system can start writing data to the disk.
  - /proc/sys/vm/dirty_ratio [10…75]
    - % of memory which when dirty, the process doing writes block and write out dirty pages to disk.
  - /proc/sys/vm/vfs_cache_pressure [50]
    - the rate at which VFS caches are reclaimed.
- Kernel memory
  - /proc/sys/vm/min_free_kbytes [262144]
    - amount of memory that is kept free for use by special reserves including “atomic” allocations
  - echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled

Tuning Daemons

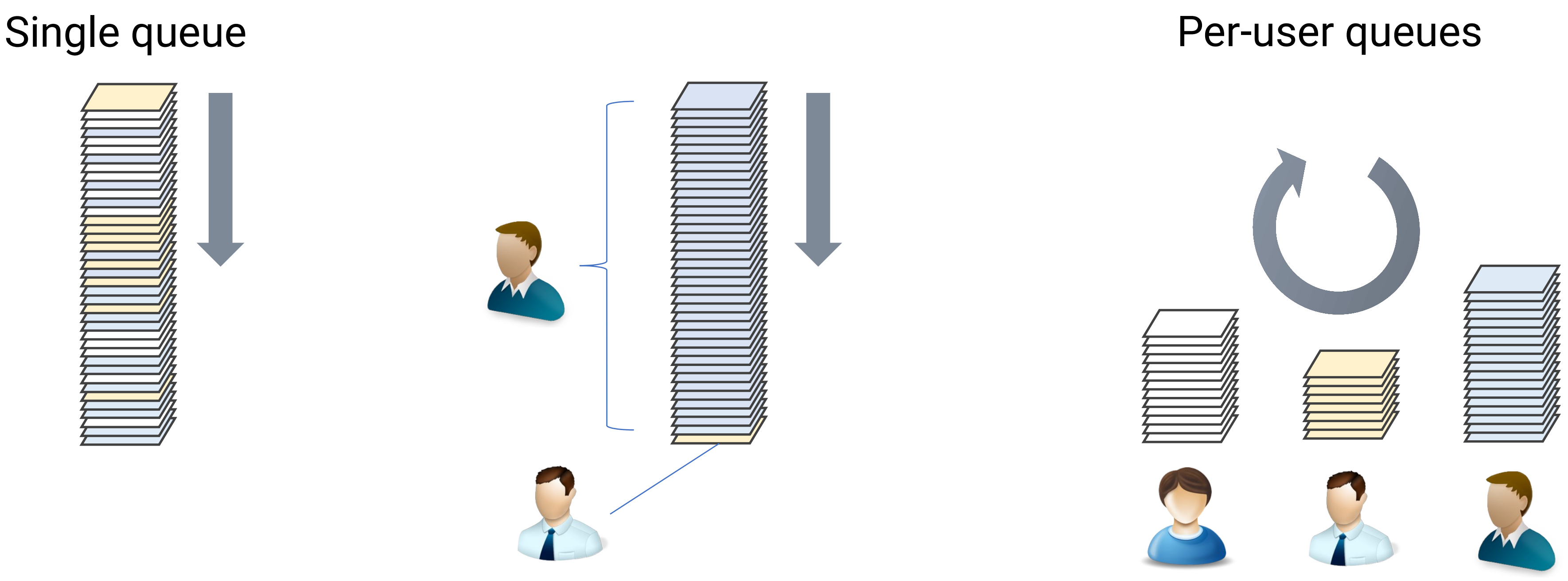


Tuning Daemons



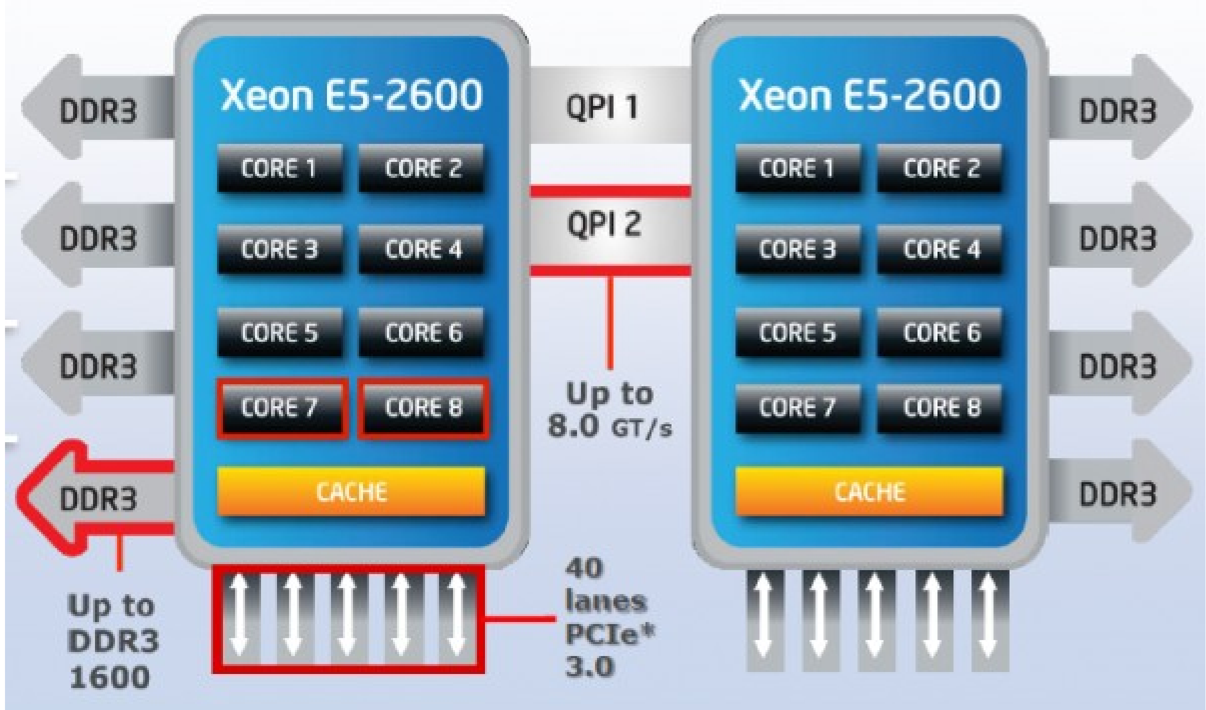
Tuning Daemons

- beegfs-{meta,storage}.conf
  - connUseRDMA (true)
  - tuneUsePerUserMsgQueues (false)



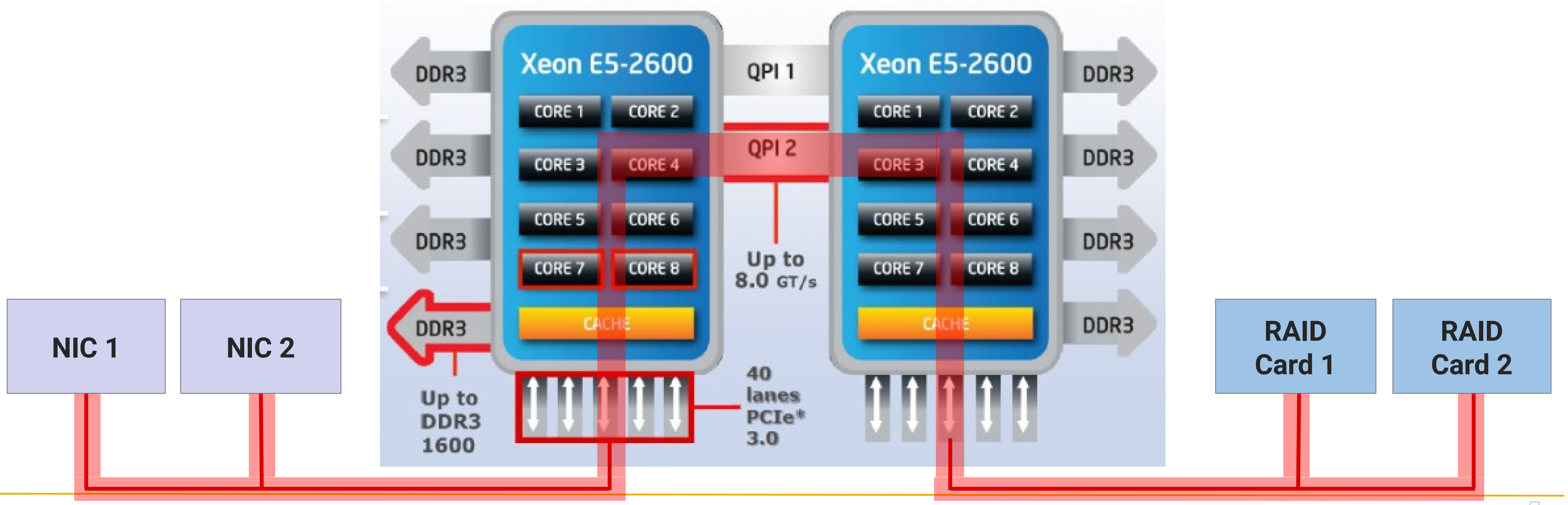
Tuning Daemons

- Devices connected to specific CPUs
  - ls –al /sys/devices/system/node/node{0,1}
  - lstopo command shows which devices are attached to which zone
  - For Centos 7 the command is hwloc-ls or hwloc-info



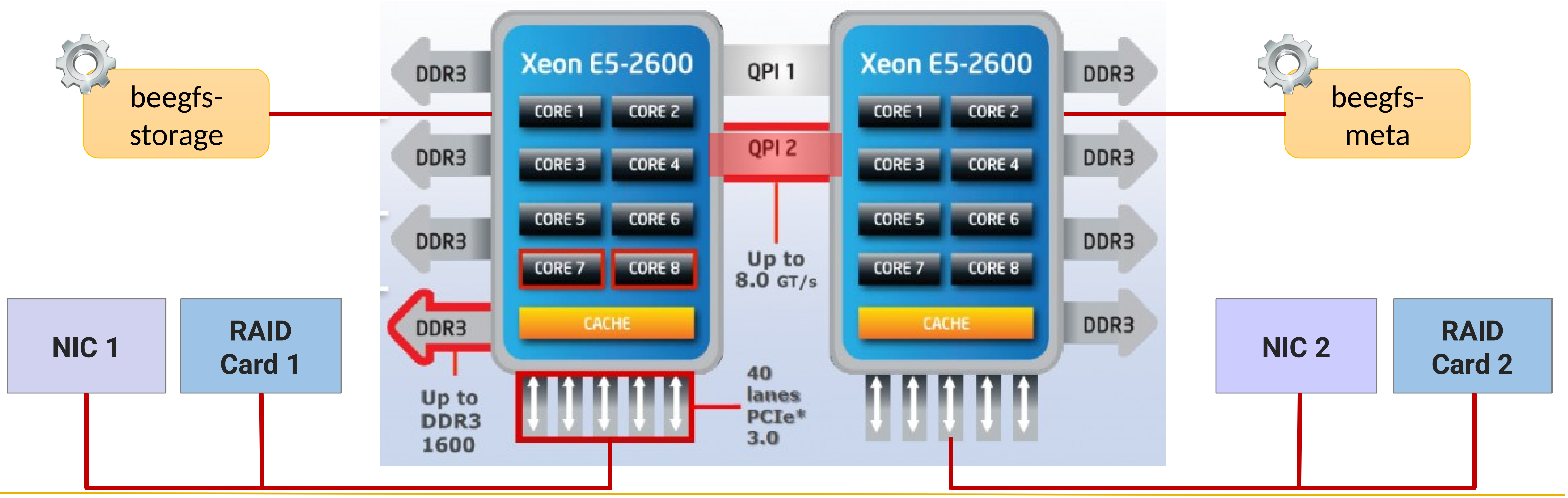
Tuning Daemons

- Bad configuration
  - Data path always involves both sockets (NIC ↔ RAID-Card)
  - Important for Omni-Path and Mellanox EDR (12 GB/s)



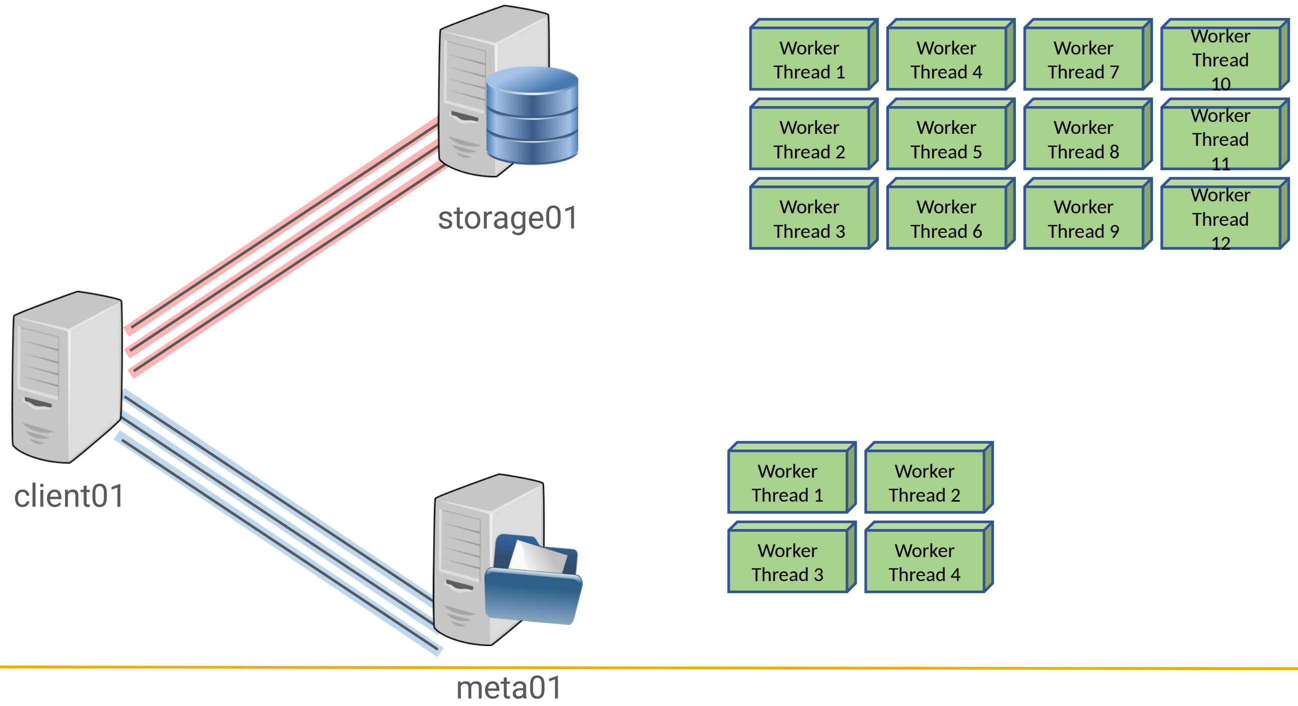
Tuning Daemons

- Better configuration
  - Each CPU connected to pair of NIC and Raid Card
  - Use tuneBindToNumaZone to pin each BeeGFS process to a specific CPU



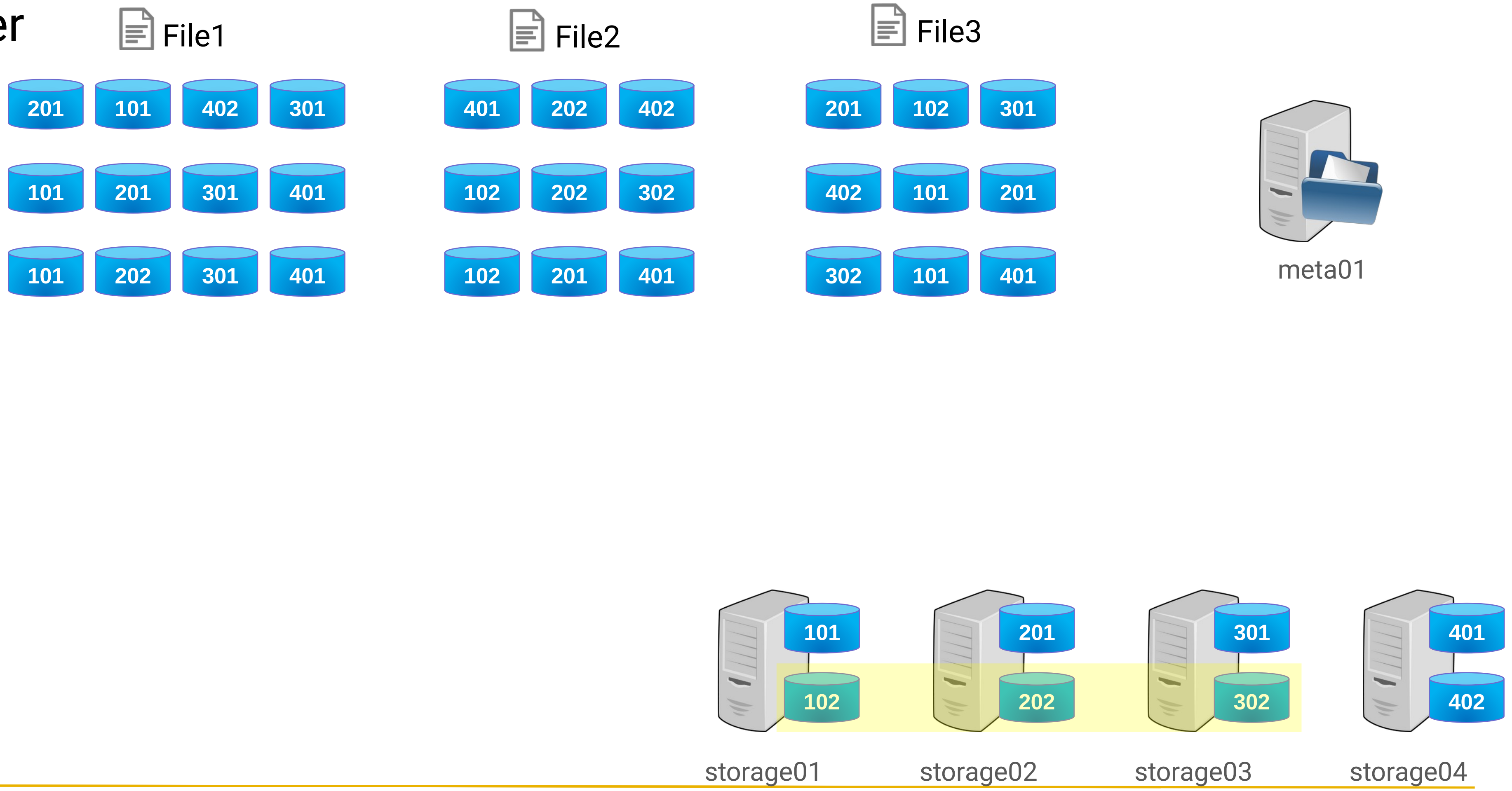
Tuning Daemons

- beegfs-meta.conf
  - connMaxInternodeNum -> default 32
  - tuneNumWorkers -> default 0 (twice the number of CPU cores)
- beegfs-storage.conf
  - connMaxInternodeNum -> default 24
  - tuneNumWorkers -> default 12 (per target)



Tuning Daemons

- beegfs-meta.conf
  - tuneTargetChooser
    - randomized
    - roundrobin
    - randomrobin



Tuning Daemons

- beegfs-client.conf (expert options)
  - TuneFileCacheType
    - Default: buffered
  - TuneDirSubentryCacheValidityMS
    - Default: 1000
  - TuneFileSubentryCacheValidityMS
    - Default: 1000

Tuning Example with NVMe Storage I

- BeeGFS Storage Service Tuning
  - Formatting Options

NVMe file system	XFS
Partion alignment	yes

  - XFS Mount Options

Last file and directory access	noatime,nodiratime
Log buffer tuning	logbufs=8, logbsize=256k
Streaming performance optimization	Largeio, inode64,swalloc
Streaming write throughput	allocsize=131072k

Tuning Example with NVMe Storage II

- BeeGFS Storage Service Tuning
  - IO Scheduler

Scheduler	deadline
Number of schedulable requests (nr_requests)	1023 (max)
Read-ahead data (read_ahead_kb)	4096
Max kilobytes per filesystem request (max_sectors_kb)	512

  - BeeGFS Storage Service Options

Worker threads (tuneNumWorkers)	64
tuneBindToNumaZone	0

Tuning Example with NVMe Metadata I

- BeeGFS Metadata Service Tuning
  - Formatting Options

NVMe file system	ext4
Partition alignment	yes
Large Inodes	-I 512
Number of Inodes	-i 2048
Large journal	-J size=400
Extended attributes	user_xattr

  - ext4 Mount Options

Last file and directory access	noatime,nodiratime
Write barriers	nobarrier

Tuning Example with NVMe Metadata II

- BeeGFS Metadata Service Tuning
  - IO Scheduler

Scheduler	deadline
Number of schedulable requests (nr_requests)	1023 (max)
Read-ahead data (read_ahead_kb)	4096
Max kilobytes per filesystem request (max_sectors_kb)	512

  - BeeGFS Metadata Service Options

Worker threads (tuneNumWorkers)	120
Requests in flight to same server (connMaxInternodeNum)	32
tuneBindToNumaZone	-

Tuning Example with NVMe Client and striping

- BeeGFS Client Service Tuning
  - Client

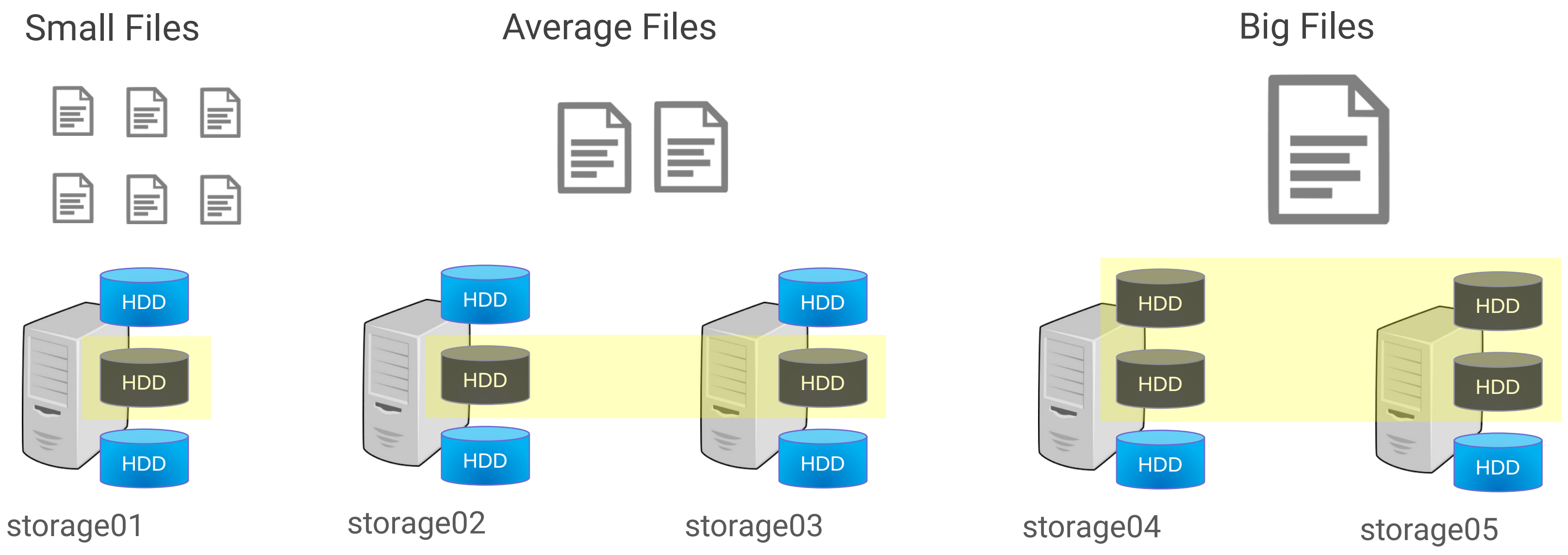
Requests in flight to the same server	18
Number of available RDMA buffers (connRDMABufNum)	70
Maximum size of RDMA buffer (connRDMABufSize)	8192
Remote fsync (tuneRemoteFSync)	true

  - striping

Chunksize	512K
Storage targets per file (beegfs-ctl --numtargets)	1

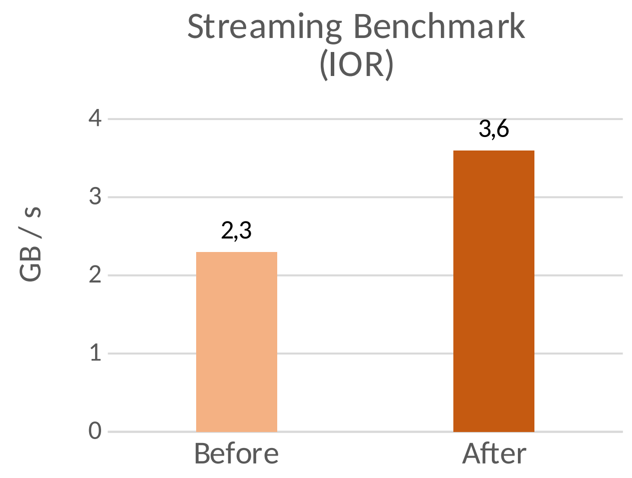
Tuning Data Striping

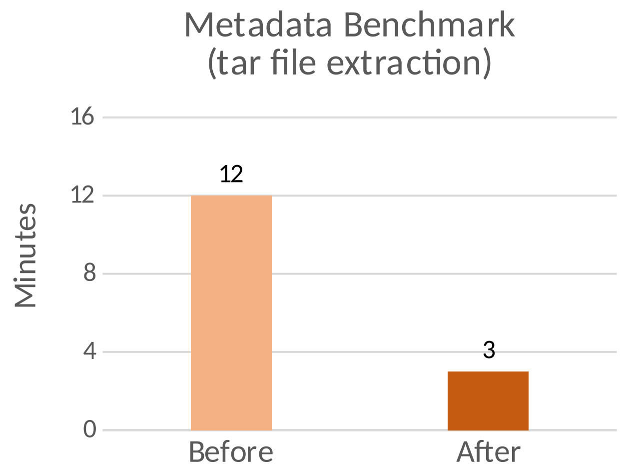
- beegfs-ctl --setpattern
  - --chunksize -> default 512KB
  - --numtargets -> default 4
  - --storagepool
  - --buddygroup



Benchmark Results

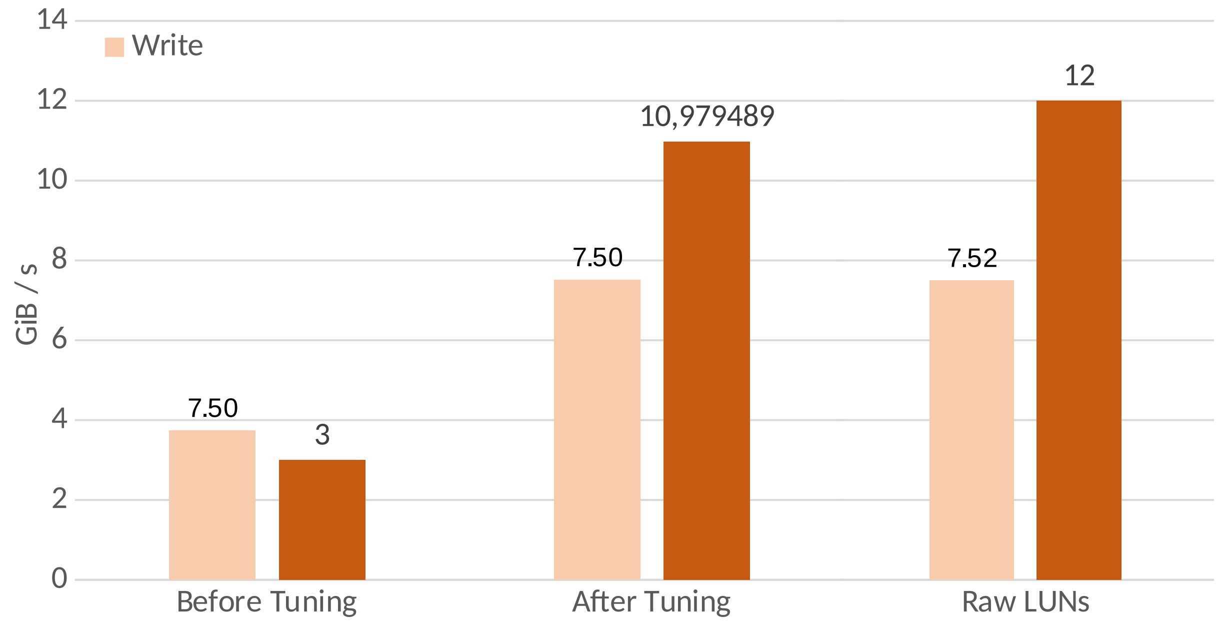
- Before: System set up with…
  - sub-optimal choice for metadata target
  - sub-optimal RAID-array settings
  - no block device and OS tuning
- Streaming benchmark
  - Before: 2.3 GB/s
  - After: 3.6 GB/s
- Metadata benchmark (tar file extraction)
  - Before: ~12 minutes
  - After: 3 minutes





Benchmark Results

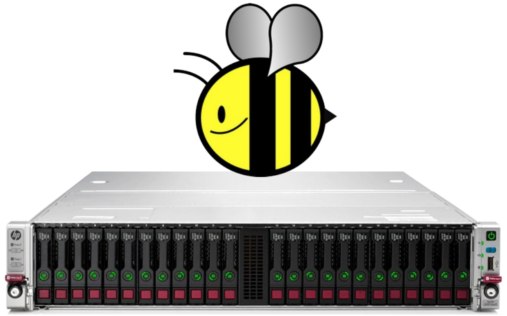
- Shared Storage HA System throughput before and after tuning:
  - 4 Servers, each with
    - CPU: 2 x 8 cores, 2.6 GHz
    - RAM: 128 GB
  - External storage
    - 4 x 368 GB SSD
    - 32 x 14.5 TB LUN – 6 x HDD (4+2 RAID 6)



Benchmark Results

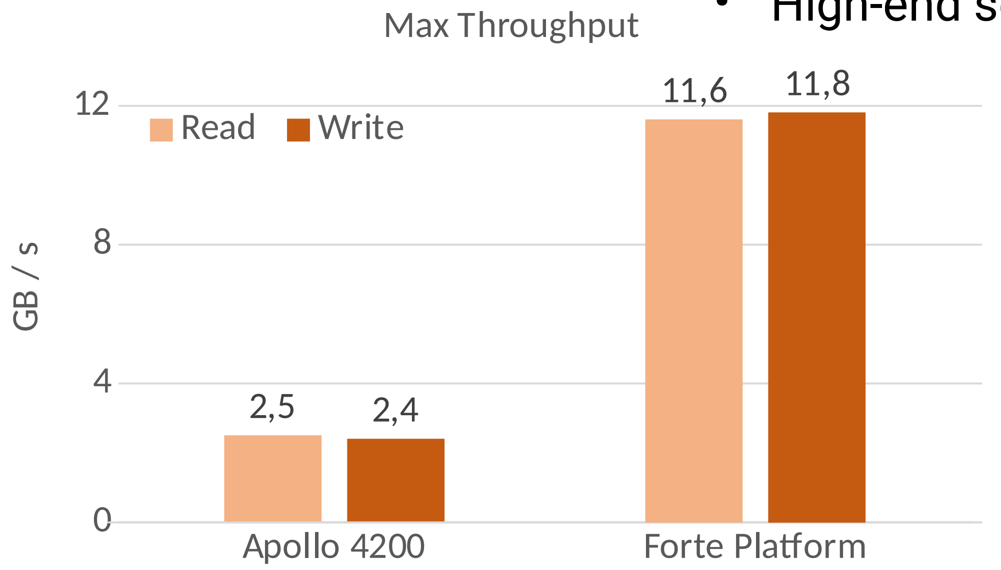
HP Apollo 4200 Gen9 Server

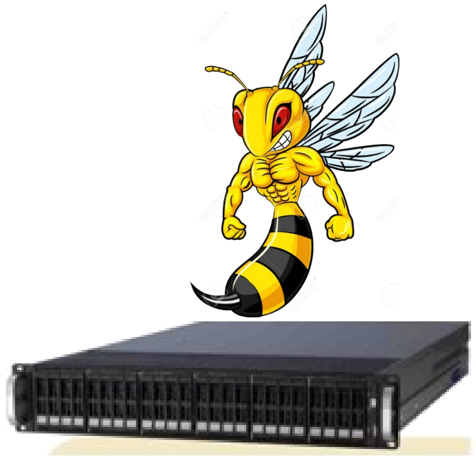
- 2 x Intel Xeon E5-2620 v3 2.40 GHz
- 2 x 400 GB SSD, 2x LFF disk array 12 x 4 TB HDDs
- 128 GB RAM



Scalable Informatics Forte Platform

- Up to 36 x Intel “Haswell” CPU cores
- 24 x NVMe drives
- 1 TB DDR4 RAM
- High-end service





What's Next?

- Typical Administration Task ✓
- Sizing & Tuning ✓
- BeeGFS Resiliency Tools
- BeeOND: BeeGFS on Demand
- BeeGFS Hive Index
- Conclusion

Thank You

Follow BeeGFS: