xyratex

Advancing Digital Storage Innovation

Bridging the peta- to exa-scale I/O gap

Peter Braam

Dwarfs and offspring under the roofs

xyratex

Forward Looking Statement

The following information contains, or may be deemed to contain, "forward-looking statements" (as defined in the U.S. Private Securities Litigation Reform Act of 1995) which reflect our current views with respect to future events and financial performance. We use words such as "anticipates," "believes," "plans," "expects," "future,"' "intends," "may," "will," "should," "estimates," "predicts," "potential," "continue" and similar expressions to identify these forward-looking statements. All forward-looking statements address matters that involve risks and uncertainties. Accordingly, you should not rely on forward-looking statements, as there are or will be important factors that could cause our actual results, as well as those of the markets we serve, levels of activity, performance, achievements and prospects to differ materially from the results predicted or implied by these forward-looking statements. These risks, uncertainties and other factors include, among others, those identified in "Risk Factors," "Management's Discussion and Analysis of Financial Condition and Results of Operations" and elsewhere in the company's 20-F filed with the SEC. Xyratex Ltd. undertakes no obligation to publicly update or review any forward-looking statements, whether as a result of new information, future developments or otherwise.

Goal of this talk

- Who is Xyratex?
- Exa-scale systems
- A sample use case
- Characterizing load
- Reasoning about performance
- Examples

Who is Xyratex?

Xyratex - Unique and Deep Understanding of Storage

NETWORKED STORAGE SOLUTIONS

High-Capacity Storage Systems

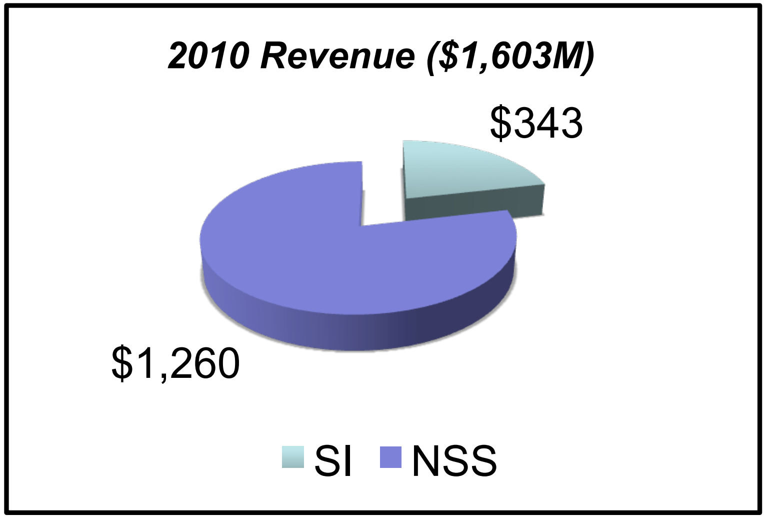
STORAGE INFRASTRUCTURE

High-Density Disk Drive Testing

xyratex

Leading OEM Provider of Digital Storage Technology

- *SI: Largest independent supplier of Disk Drive Capital Equipment*
  - ***~ 50% of w/w disk drives are produced utilizing Xyratex Technology***
  - ***~ 75% of w/w 3.5" LFF disk drives***
- *NSS: Largest OEM Disk Storage System Supplier*
  - ***33% WW OEM Market Share in 2009, 5 Tier-1 OEM's***
  - ***16% of worldwide external storage capacity shipped in 2009 (IDC)***
  - ***> 3.0 Exabyte's of storage shipped in 2010***
  - ***~ 139,000 storage enclosures shipped in 2010***



Xyratex – Storage Hardware & Software

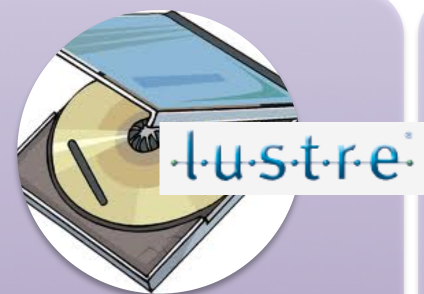
Firmware

Designs, Develops & Firmware for enclosures & controllers

OS

Linux based Storage Appliance

File Systems



World-Class Clustered File System Development & Support Expertise

Management

Storage Management Framework

Unique Ability to Deliver and Support Storage Solutions

Lustre is doing well: Top 500

- Nov 2010:
  - 9 of top 10 systems run Lustre
  - >70 of the top 100 systems run Lustre
- Dozens of research efforts modify it
- Dozens of OEMs have shipped it
- IDC indicates its future is *very* bright

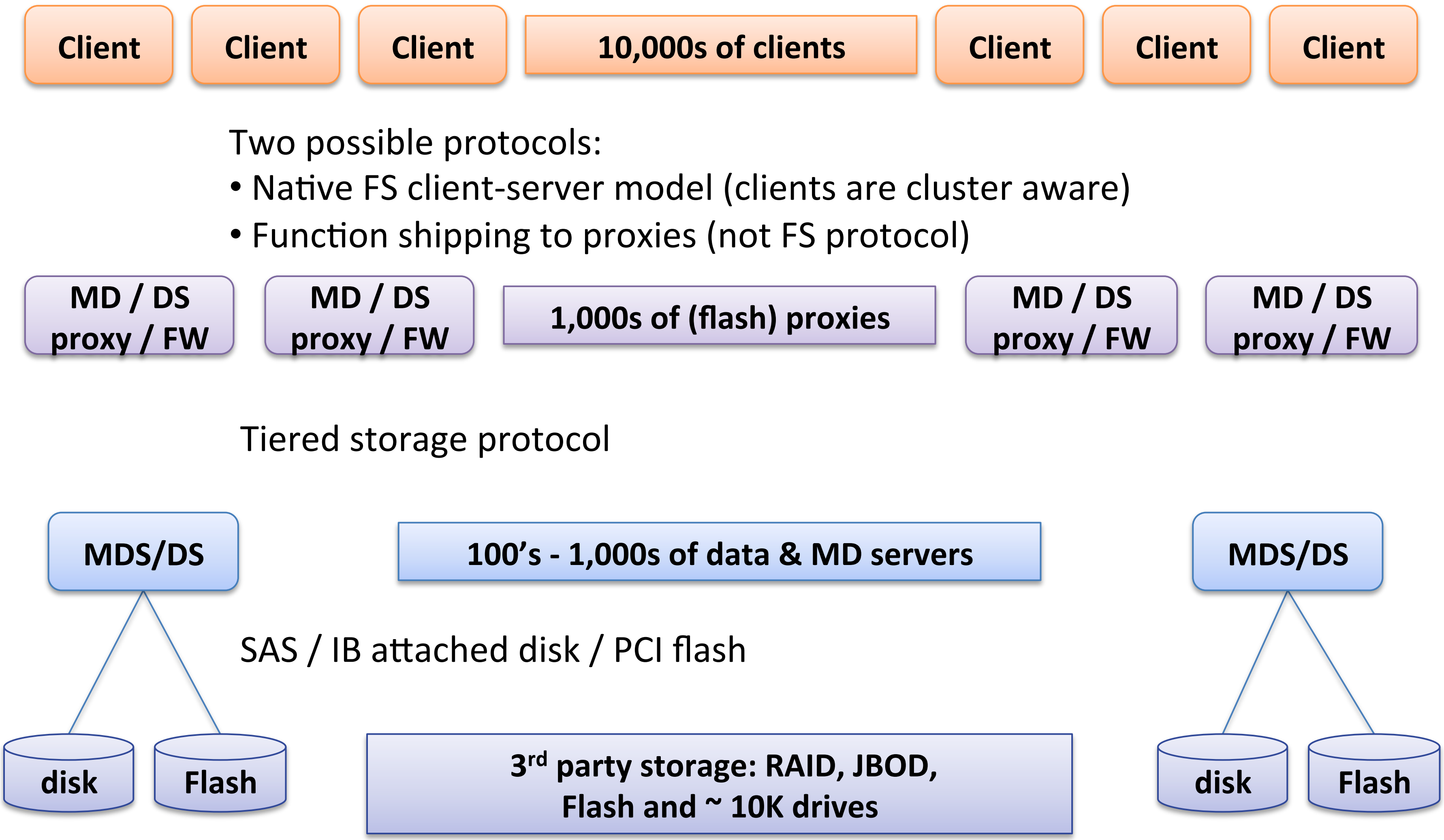
Peta & Exa-scale systems

Exa scale clusters

- Exa scale systems
  - 10^8 cores – each ~10GF/sec, each ~1G RAM
  - 5,000 cores / node, 5 TB RAM / node (50 TF / node)
  - 20K cluster nodes, 100 PB RAM / cluster
  - I/O: 300 TB / sec, one node 15 GB / sec
  - File system > 1 EB
- Technology revolutions
  - File system clients will have ~10,000 cores
  - Architectures will be heterogeneous
  - Flash and/or PCM storage leads to tiered storage
  - Anti revolution – disks will only be a bit faster than today

Sample use case

Example deployment styles



Lessons from benchmarking

- 1 TB FATSAS drives (Seagate Barracuda)
  - 120 MB/sec bandwidth with cache off
  - 4MB allocation unit is “optimal”
- PCI flash and NFSv4.1 RPC system
  - IB connection
  - Embedded database backend
  - 100K transactions / sec aggregate, sustained
    - Update 2 tables and using transaction log
    - One server

100 PF Solution

- 500 servers, each acting as MDS and DS
- Disk capacity 500 x 8TB x 40 dr = 160 PB raw
  - BW ~ 20,000 x 120 MB/sec = 2.4 TB /sec
- Network 4x EDR IB – effective BW 25 GB/sec
- PCI flash
  - capacity 500 x 6 TB = 3 PB
  - BW/node: 25 GB/sec, aggregate: 12.5 TB/sec
- MD throughput aggregate: 50M trans / sec
  - 1 copy of MD remains in flash
  - 10^12 inodes x 150 B = 150 TB, or 5% of flash

HDF5 file I/O – use case

- HDF5 is a file format containing directories and data
- Servers detect ongoing small I/O on part of a file
- It chooses to migrate a section of the file and the file allocation data into flash

- During migration, small I/O stops briefly
- Now 100K iops are available to flash
- When file is quiescent, data migrates back
- In summary: treat disk as HSM when needed

- Promising!

But…

- Flash
  - price and performance aren’t scaling as we were hoping
- Current systems have shown low disk BW utilization
  - On ‘optimal benchmarks’ ~ 50% (then try dbench 100)
  - This picture may not help that
- Bridging the last 10x from 100 PF to 1EF gap looks hard
  - Remember the disk drives
- The exa-scale community is open to revolution

Describe an approach to performance modeling & analysis

- Simple enough that it can easily be done
  - Contrast with simulation, which appears to be hard
- Semi-quantitative
  - Ideal numbers and boundaries are easily visible
- Systematic
- Applies to all kinds of devices and to clusters

Acknowledgement

- P. Colella, “Defining Software Requirements for Scientific Computing,” presentation, 2004.
- The Landscape of Parallel Computing Research: A View from Berkeley. Many authors, Report, Berkeley, 2006.
- Roofline: An insightful Visual Performance model for multicore Architectures (Williams, Waterman, Patterson, IEEE Computer 2009)

What about the remainder?

- There are good modeling frameworks for availability
  - Markov models and state machines
- They are not widely used, but provide crystal clear guidance on availability models for a product
- This talk isn't focusing on that.

Seven I/O Dwarfs

Mimic Berkeley – seven I/O Dwarfs

- There are far too many I/O benchmarks
- Identify the typical I/O kernels
- These kernels are called dwarfs

- Requirements on set of dwarfs
  - Small enough to be manageable
  - Broad enough to cover essential points in architecture
    - Typically some dwarfs may require special architecture

List of the dwarfs

1. Download
   - Summary: All clients read the same file
   - Key problem: server bottlenecks
2. SSF Write
   - Summary: All clients / threads write to one file
   - Key problem: Many partial stripe writes are inefficient
3. Tree read
   - Summary: Many clients do small I/O with seeks on large file
   - Key problem: Seeks make I/O inefficient
4. FPP Write
   - Summary: All processes write their own file
   - Key problem: Storm of file creates

List of the dwarfs - ctd

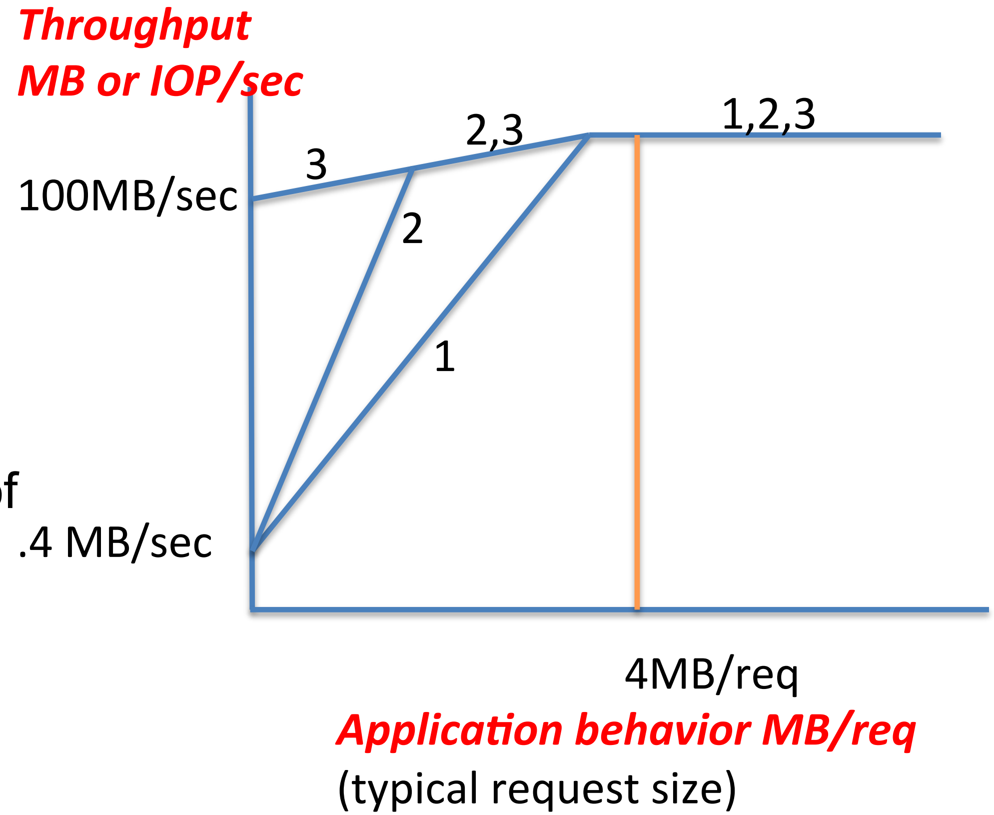
5. Metadata and Small I/O
   - Summary: find, ls –l, rsync, rm –r, tar {cx}f (on a large tree)
   - Key problem: Performance, locality
6. Highly multithreaded I/O
   - Summary: Thousands of threads do FS operations on one node
   - Key problems: Fragmentation, fairness
7. Cache integration
   - Summary: A cache with many objects migrates to slower tier
   - Key problem: Iteration

- Some dwarfs are undoubtedly missing
  - One is obliged to start with 7

Rooflines

Roofline

- Rooflines indicate ***maximum possible performance*** given typical request size
- Multiple roof lines
  - Associated with presence of optimizations
- E.g.
  - Sample graph for disk
  - 3 no rotational delay, no seek
  - 2 rotational delay, no seek
  - 1 rotational delay & seek (random)



Sample rooflines for hard drive

Rooflines – applicability

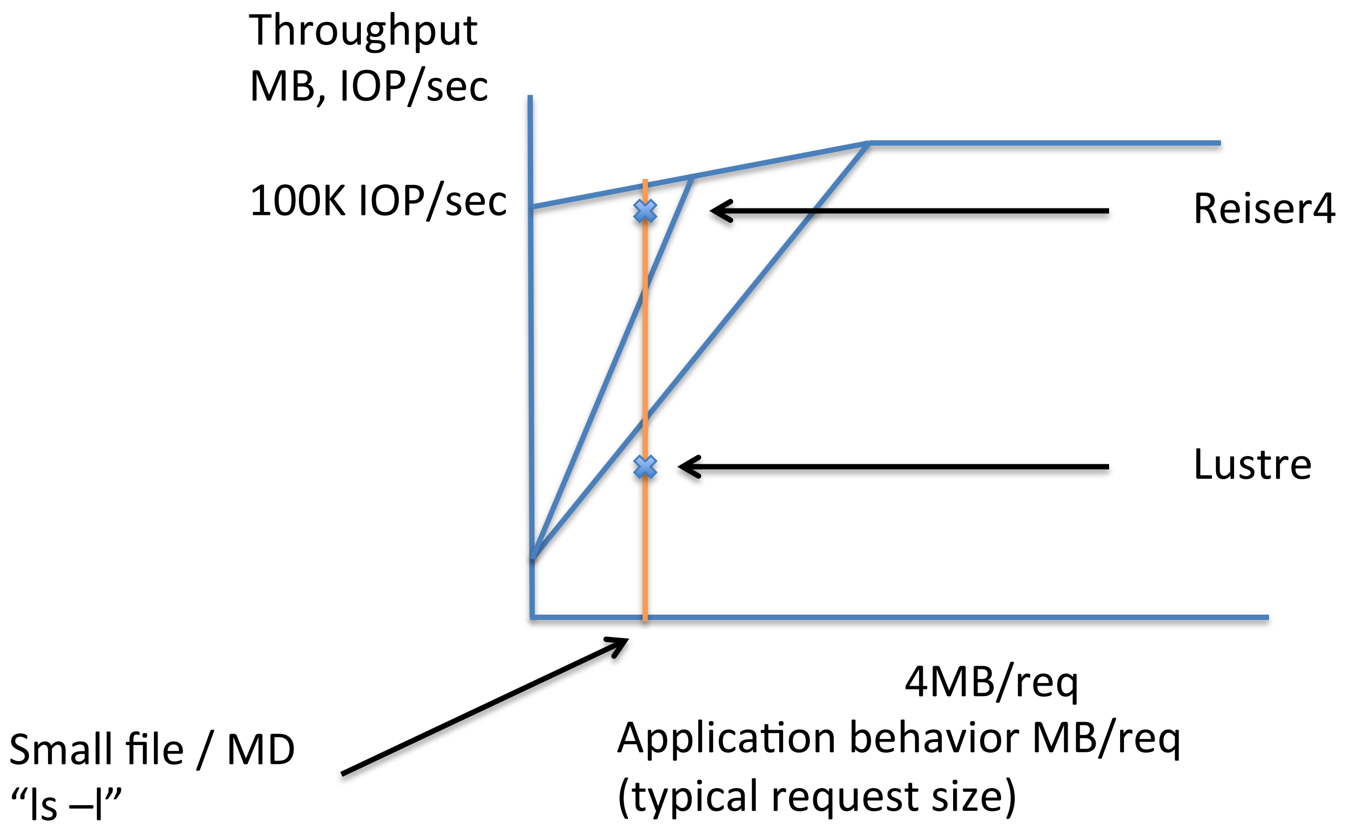
- Applicable to any storage related system
  - Clients
  - Enclosures
  - Servers
  - Drives, Flash
- Semi-quantitative
- Different parameters define regions
  - For enclosure the SAS HBA and expander may be important
  - For clients memory, network, CPU

Dwarf Applications

- Dwarf application has typical I/O size
  - Hence determines a point on the horizontal axis
  - If you change the application, the point may move
    - This can be an optimization, e.g. do larger I/O

- The dwarf's performance is the y-coordinate
  - By optimizing the storage system, this can go up

Sample, hypothetical application & roofline

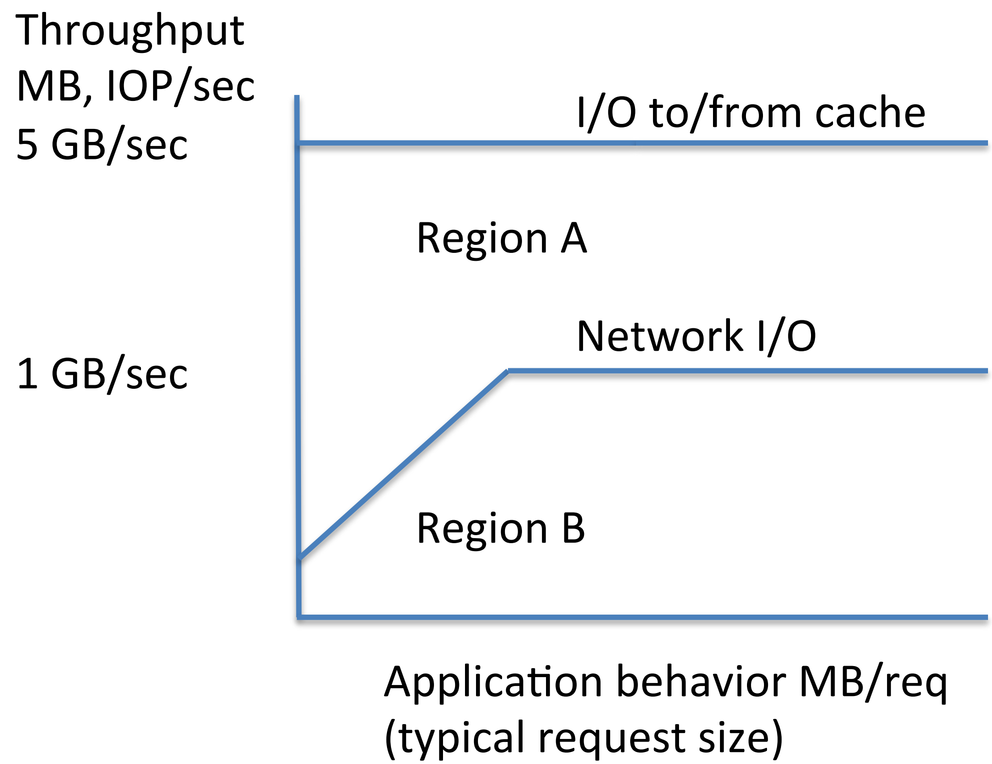


Optimizations

- A seemingly finite set of regions indicate what optimization might be most fruitful, e.g.
  - Larger I/O
  - Aligned I/O – don't write half stripes
  - Eliminate rotational delays or seeks
  - Caching for aggregation
  - Introducing a changelog to avoid scanning
  - Read ahead
  - Collective operations
  - RAM or flash caches
  - Re-ordering (elevators, network request schedulers)
  - Avoiding lock revocations in protocols

File system client rooflines

- If a dwarf is in region A
  - Eliminate remaining network I/O
  - Optimize memory access & threading
- If a dwarf is in region B
  - Increase I/O sizes (e.g. read-ahead)
  - Start leveraging caches
- Note: not necessarily one “best approach”



Dwarfs have offspring

- Striping
  - One I/O load on a client become a set of loads on servers
- Client server model
  - Many loads on clients combine to one load on servers
- Thread to node
  - Many threads combine to a load on a node

Exa-scale I/O

High end HPC storage systems

- 10 years ago, Fortran ruled
- Now new methods are embraced
  - Global address spaces (PGAS), languages (e.g. X10), ..
- File systems cause HPC I/O bottlenecks: remedies
  1. **Surrender control to an I/O library used with application**
  2. **Embrace local storage – much higher aggregate BW**
- But
  - POSIX operations will remain important
  - Data re-organization is a central part of HPC I/O
- Begin to develop a new I/O library
  - Not layered on file systems, using deeper API's
  - Obviously this needs to be an open effort

Some examples what you cannot do with file systems

- Overlapping stripes from different clients
  - Very costly to write
  - An I/O library can detect this
- I/O models, free of locking with barriers
  - Very similar to what HPC applications do anyway
  - Tuned to HPC like Hadoop was to map-reduce
- Compiler supported speculative & aligned read-ahead
- Allow applications to simply map data structures
- PGAS like location / layout descriptors
- Integrate HDF5 / NetCDF formats into file system
  - Do not use file I/O do to metadata (current practice)

Conclusions

Summary

- 100PF is “easy”
- There is a better design methodology
- A new style of doing I/O holds promise

Thank you