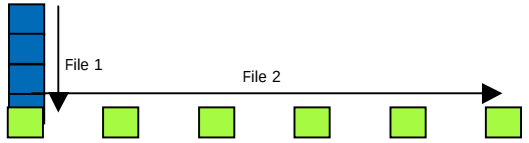
PERCS File System and Storage

Roger Haskin
Senior Manager, File Systems
IBM Almaden Research Center

Outline

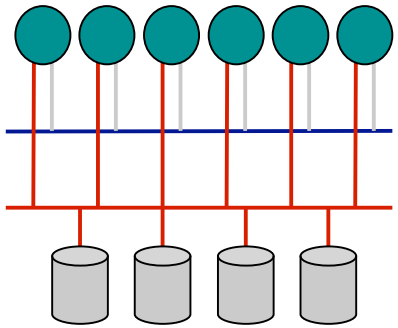
- Background: GPFS, HPCS, NSF Track 1, PERCS architecture, Blue Waters system
- Scaling GPFS to Blue Waters
- Storage Management for Blue Waters

GPFS Concepts



- **Shared Disks**
  - All data and metadata on globally accessible block storage
- **Wide Striping**
  - All data and metadata striped across all disks
  - Files striped block by block across all disks
  - … for throughput and load balancing
- **Distributed Metadata**
  - No metadata node – file system nodes manipulate metadata directly
  - Distributed locking coordinates disk access from multiple nodes
  - Metadata updates journaled to shared disk

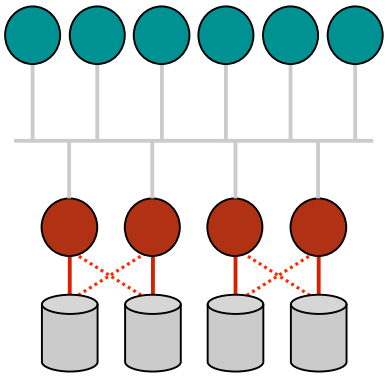
Principle: scalability through parallelism and autonomy



GPFS file system nodes

Control IP network

Disk FC network

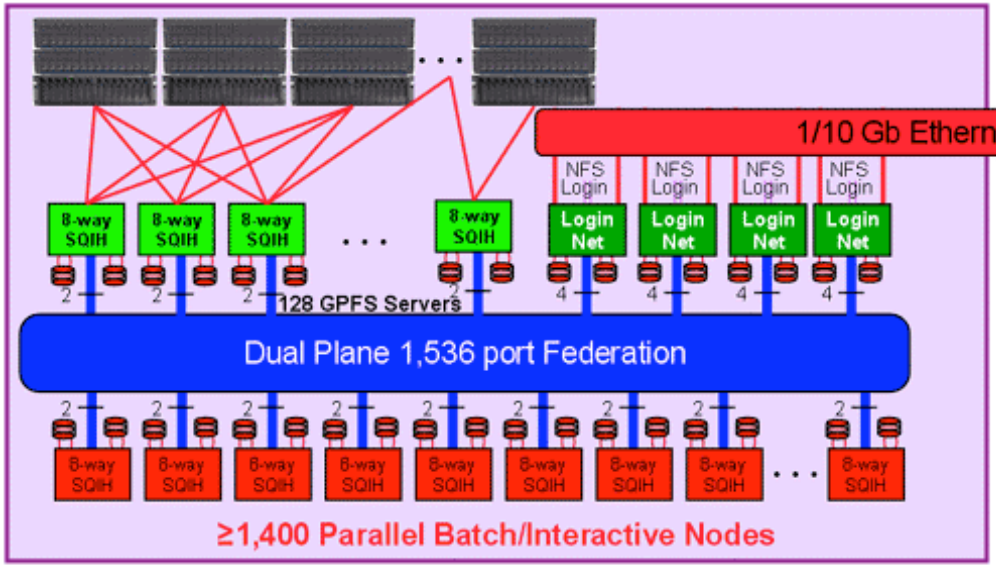


GPFS file system nodes

Data / control IP network

GPFS disk server nodes:
VSD on AIX, NSD on Linux – RPC interface to raw block devices

GPFS on ASC Purple/C Supercomputer



Purple System
- At least 1,400 parallel batch/interactive nodes
- 4 Login/network nodes from 2 SQH
- Clustered I/O with 128 SQIH for global I/O
- Dual plane 1,536 port Federation switch
- External networking
  - Login/network nodes for login/NFS/PFTP
  - All external networking is 1-10Gb/s Ethernet

Programming/Usage Model
- Application launch over all compute nodes
- 1 MPI task/CPU and Shared Memory, full 64b support
- Scalable MPI (MPI_allreduce, buffer space) to 8,192 tasks
- Likely usage
  - multiple MPI tasks/node with 2-4 OpenMP/MPI task
- Single STDIO interface
- Parallel I/O to single file, multiple serial I/O (1 file/MPI task)

- 1536-node, 100 TF pSeries cluster at Lawrence Livermore National Laboratory
- 2 PB GPFS file system (one mount point)
- 500 RAID conroller pairs, 11000 disk drives
- 126 GB/s parallel I/O measured to a single file (134GB/s to multiple files)

Peta-scale systems: DARPA HPCS, NSF Track 1

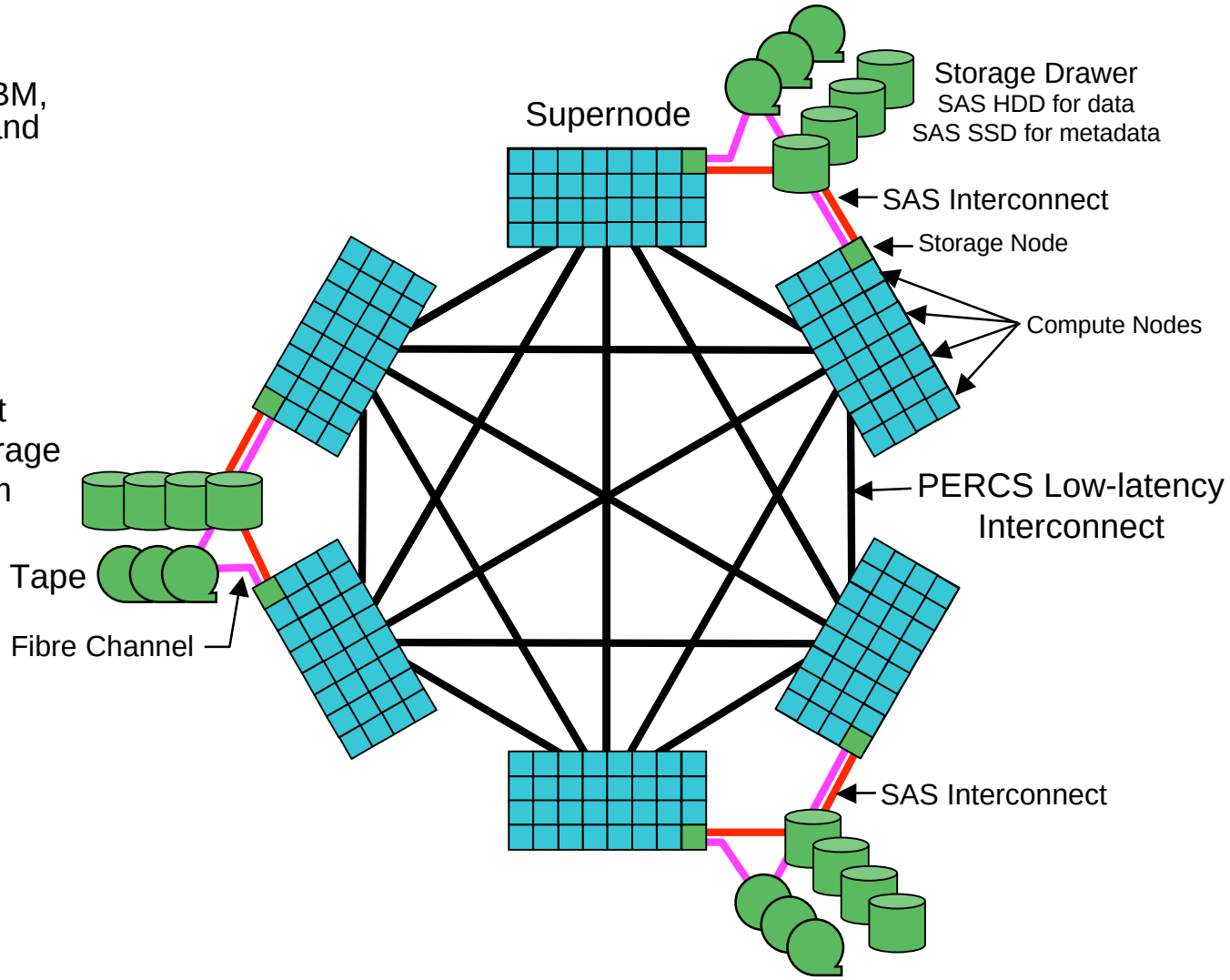
- U.S. Government investing heavily in HPC as a strategic technology
  - DARPA HPCS goal: “Double value every 18 months”
  - NSF Track 1 goal: at least a *sustained* petaflop for actual science applications
- Technical challenges for storage systems
  - Technology curves (flops/core, bytes/drive, bytes/second/drive) are flattening
  - *How to make reliable a system with 10-100x today’s number of moving parts?*
  - *How to do the above at an acceptable cost?*

40x 40x 20x

System	Year	TF	GB/s	Nodes	Cores	Storage	Disks
Blue Pacific	1998	3	3	1464	5856	43 TB	5040
White	2000	12	9	512	8192	147 TB	8064
Purple/C	2005	100	122	1536	12288	2000 TB	11000
HPCS (rough est.)	2011	6000	6000	65536	512K	120000+ TB	200000+

Blue Waters at NCSA

- Blue Waters System
  - NSF Track 1 program
  - Collaboration between IBM, NCSA, State of Illinois, and partners
  - Sustained petaflop
  - 200K processor cores
  - 10 petabytes file storage
- PERCS architecture
  - Power7 processor
  - Low-latency interconnect
  - Shared memory and storage
  - GPFS parallel file system



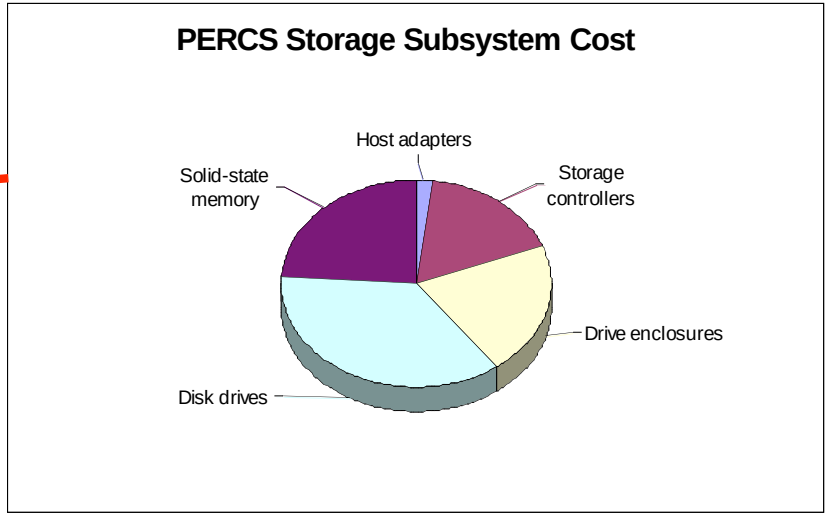
GPFS and PERCS

- HPCS file system requirements (a subset)
  - "Balanced" capacity and performance
    - (100 PB file system, 6 TB/s file I/O)
  - Reliability in the presence of localized failures
  - Support for full-up PERCS system (~64K nodes)
  - One trillion files to a single file system
  - 32K file creates per second
  - Streaming I/O at 30GB/s full duplex (for data capture)

5 years of iTunes music in 32 min!

1PB of metadata!

- Storage Requirements
  - Reasonable cost - 10-20% of system cost
    - Large number of disk drives makes this difficult to achieve
    - Metadata performance requires substantial amount of expensive NVRAM or SSD
  - Reliability - system must continue to be available in spite of component failures
    - One or more drives continually in rebuild
    - Hard error rate between 10^{-14} and 10^{-16}
    - "Silent" data corruption
  - Productivity - non-disruptive repair and rebuild
    - Goal: rebuild overhead in the 2-3% range
    - Standard RAID rebuild can affect performance 30%
    - Parallel file system with wide striping: x% hit on one LUN causes same x% hit to entire file system

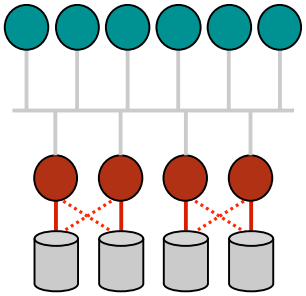


MTTDL 2 mo for RAID-5, 56 yr for RAID-6

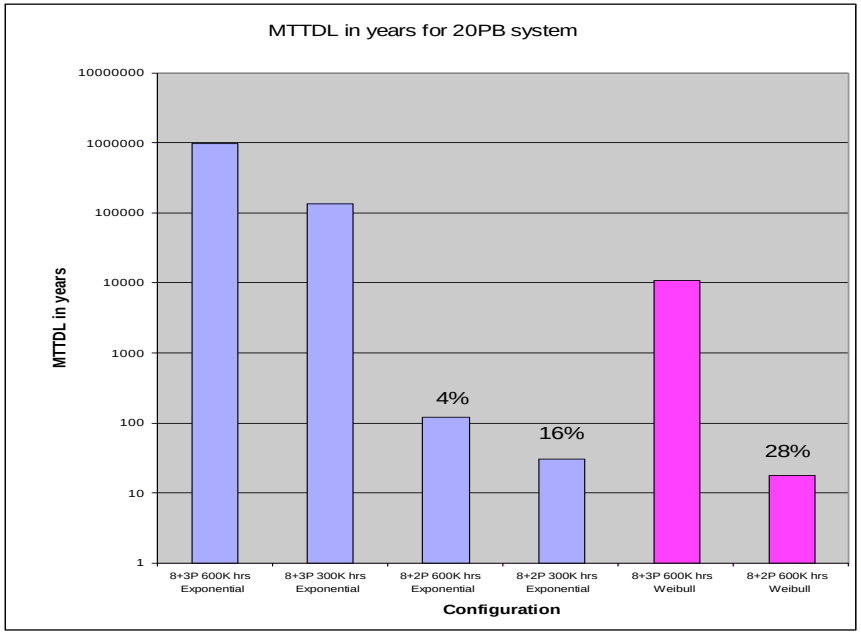
Scaling GPFS metadata operations

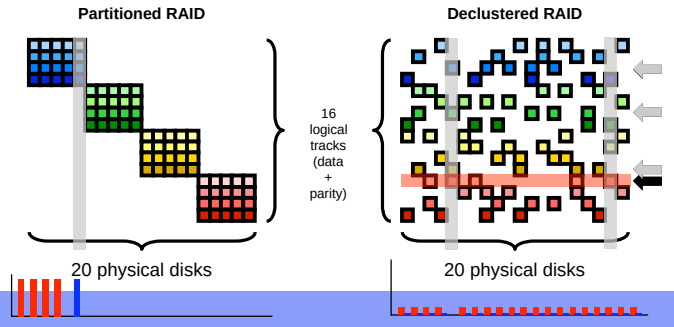
- Metadata: the on-disk data structures that represent hierarchical directories, storage allocation maps, file attributes and data pointers, recovery journals, etc.
- Why is it a problem? Structural integrity of the file system requires metadata read/writes to be properly synchronized with each other and with data I/O. Performance is sensitive to the latency of these (small) I/O's.
- Techniques for scaling metadata performance
  - Scaling synchronization (distributing the lock manager)
  - Segregating metadata from data to reduce queuing delays
    - Separate disks
    - Separate fabric ports
  - Different RAID levels for metadata to reduce latency, or solid-state memory
  - Adaptive metadata management (centralized vs. distributed)
  - GPFS currently provides for all these to some degree; work always ongoing
  - Fine-grained directory locking (multiple nodes creating in same directory)
  - Solid-state metadata storage

PERCS Storage Subsystem (Perseus)

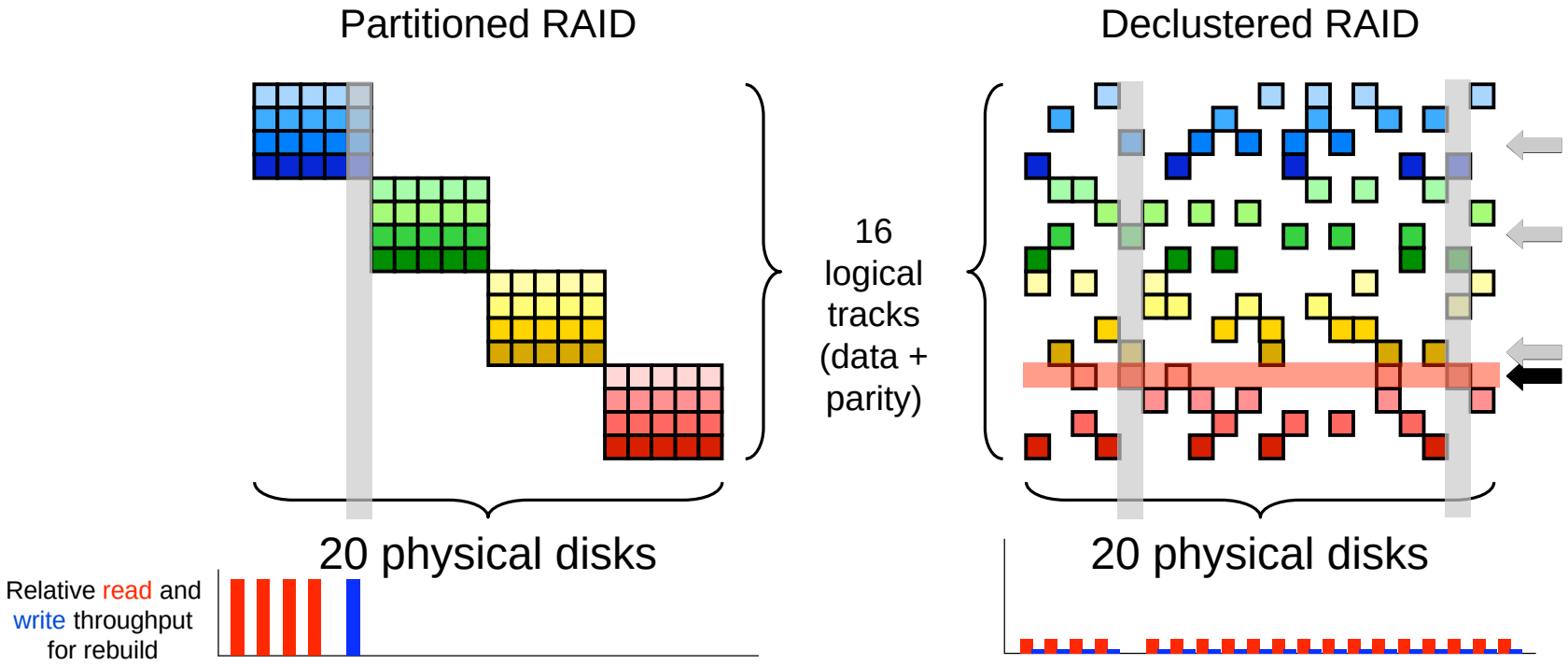


- Use PERCS nodes, direct-attached JBOD, and software RAID vs. conventional RAID controllers
  - Parallel file system already use I/O nodes to connect to storage
    - Forwards data between RAID controller and the cluster fabric
    - Typically a powerful node with almost nothing to do
    - Example: GPFS NSD server, Lustre OSS
  - Use the I/O nodes to repalce the RAID controllers!
    - Conventional SAS JBOD Host Bus Adapters
    - Special JBOD storage drawer for very dense drive packing
    - Solid-state drives for metadata storage (1-2% of total)
    - Software RAID
    - Save 20-30% of the storage subsystem cost
    - Freedom to pick more appropriate algorithms for petascale system
- Software RAID – strong codes for better MTTDL
  - Reed Solomon erasure code, “8+3P”
  - ~10^5 year MTDDL for 100PB file system
- End-to-end, disk-to-GPFS-client data checksums
  - Generated and checked by compute node, stored on disk
  - Necessary to overcome silent data corruption
- Declustered RAID for non-disruptive repair and rebuild
  - spread 8+3 parity groups *and spare space* randomly across large numbers of drives
  - Allows drive rebuild with only ~2% performance degradation



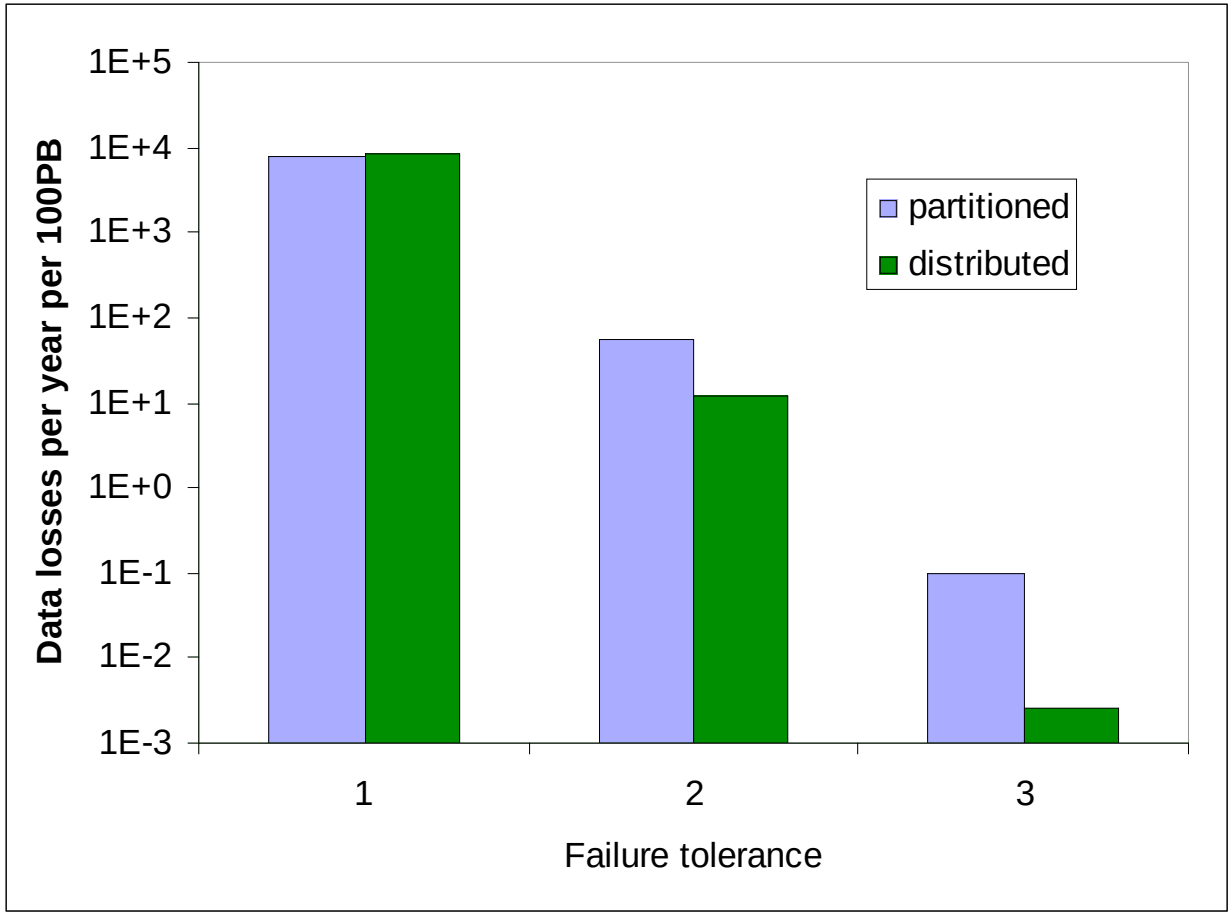


Better Data Distribution through Declustered RAID



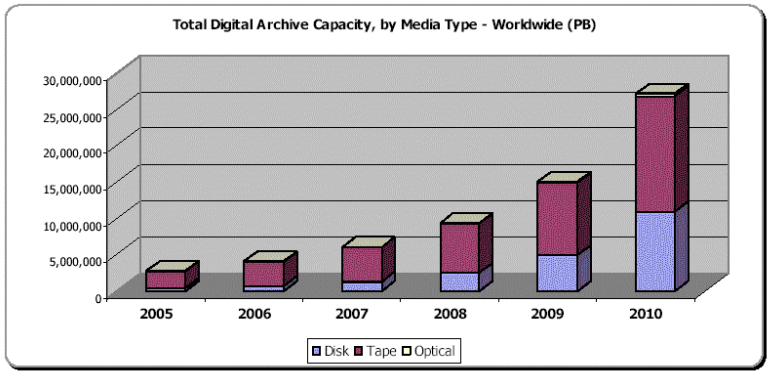
- Conventional *Partitioned* RAID partitions drives into arrays, then creates LUNs on top of the arrays
  - Drives must be added in partition quanta
  - Spare space on unused physical drives
  - Work to rebuild concentrated on the remaining drives in the array
- *Declustered* RAID distributes data and parity strips of logical tracks evenly across all drives
  - Arbitrary number of drives in the array
  - Individual drives can be added/removed
  - Spare space is also distributed evenly
  - Rebuild spread evenly

Declustered vs. Partitioned RAID Reliability



Simulation results

Hierarchical Storage Management

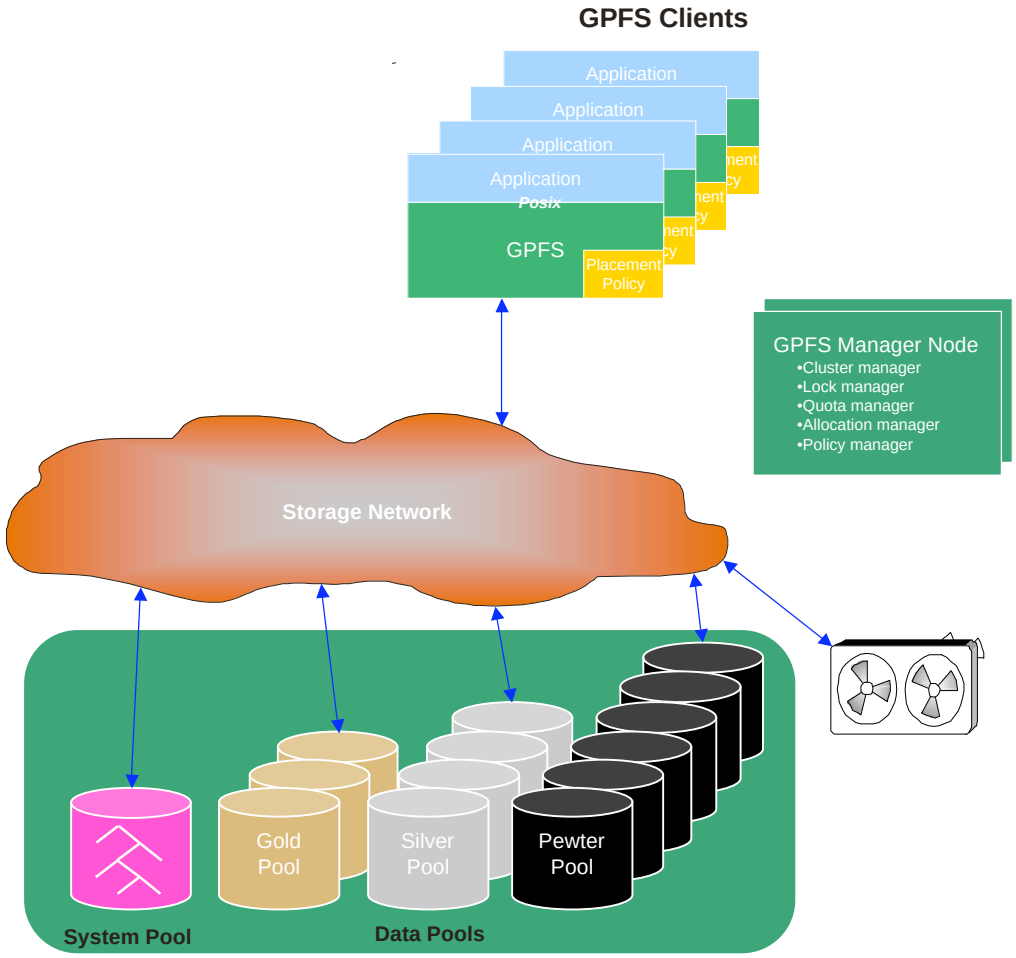


Source: Enterprise Storage Group, January 2006

- Increased differentiation in storage hardware
  - Ten years ago….
    - "disks is disks"
    - Tape will soon be dead
  - Now, solid-state, "enterprise" disk, SATA, tape
  - Performance differences (IOPS, throughput, power)
  - … all with different cost curves!
- Compute cycles are getting cheaper relative to storage
  - Efficient use of storage is becoming more important
- As a result, hierarchical storage management is more important than ever
- Unfortunately, HSM is hard to scale
  - How to make it fast?
  - How to optimize the use of various storage technologies?
  - How to make it reliable?

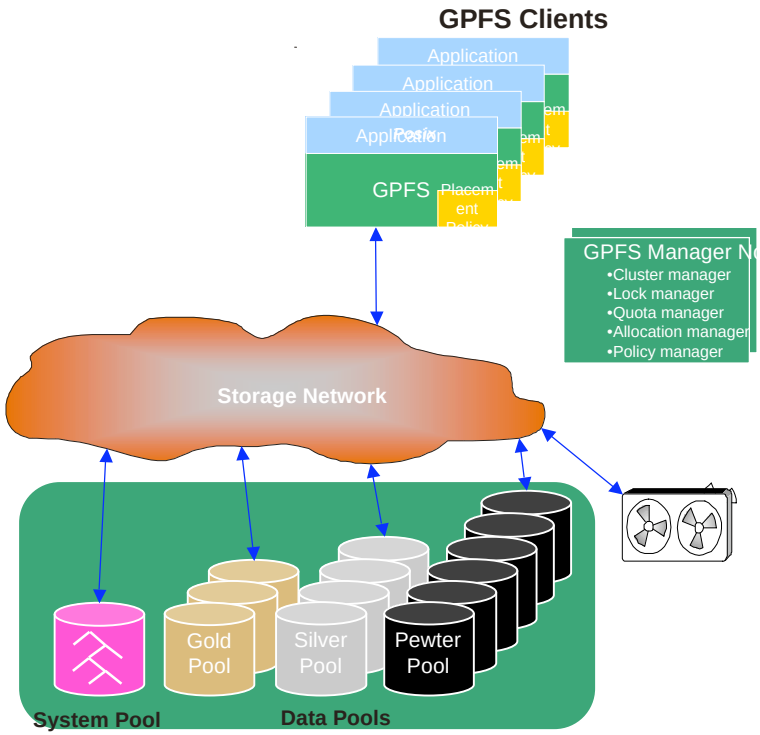
GPFS Information Lifecycle Management (ILM)

- GPFS ILM abstractions:
  - Storage pool – a group of storage volumes (disk or tape)
  - Policy – rules for placing files into storage pools
- GPFS policy rules much richer than conventional HSM “how big is the file and when was it last touched”
  - Tiered storage – create files on fast, reliable storage (e.g. solid state), move files as they age to slower storage, then to tape (a.k.a. HSM)
  - Differentiated storage - place media files on storage with high throughput, database on storage with high IO’s per second
  - Grouping - keep related files together, e.g. for failure containment or project storage



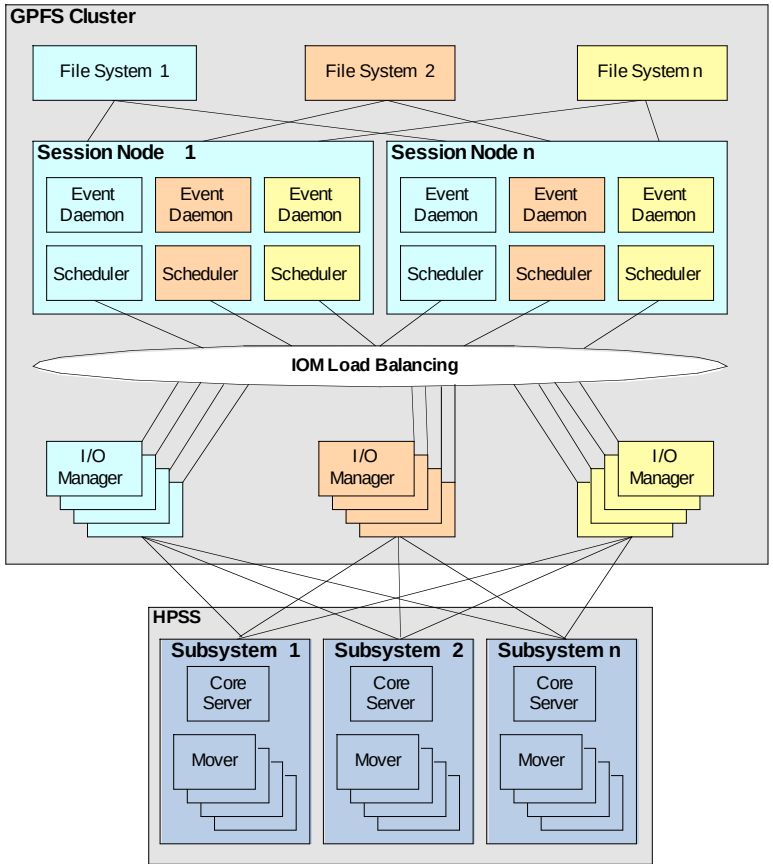
Policies

- Policy rules control the placement, migration, and retention of files
  - Declarative SQL-like language
  - Rule types:
    - Disk pools: placement, migration, deletion
    - External (tape) pools: premigrate, migrate, recall, purge, list
- Policy rule evaluation
  - Rules evaluated according to schedule, event (e.g. low space in a pool), or explicit command
  - Policy rules applied in turn to each file's metadata
    - Scan file names, order by inode number
    - Join result with inode file records
    - Apply policy rule predicate to each (inode #, name, metadata) tuple
    - If file matches, apply the policy rule's action (e.g. migration)



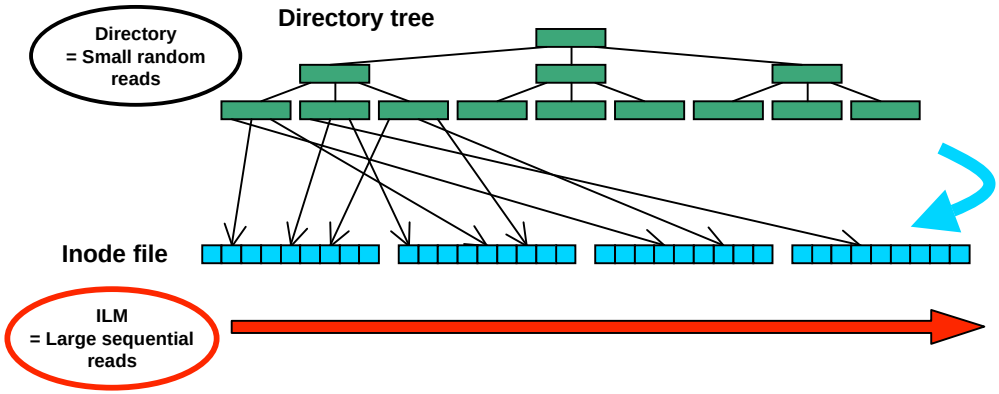
GPFS + HPSS

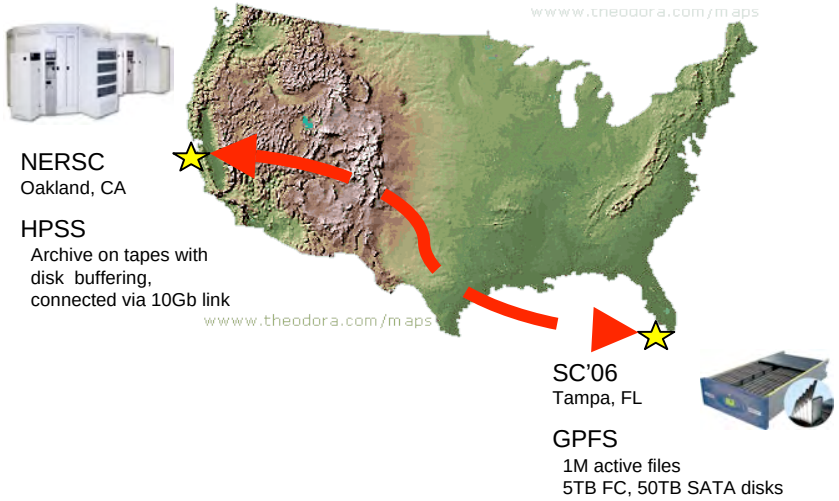
- HPSS provides a highly-scalable tape archive integrated with GPFS ILM
  - 10s of petabyte to an exabyte of data.
  - 10s of millions to 100s of billions of files
- External HPSS pools provide:
  - HSM services for GPFS.
  - Backup/restore services for GPFS.
  - Both services are tightly coupled for better tape management.
- Files move between GPFS and HPSS using:
  - Scalable HPSS I/O Manager architecture inside the GPFS cluster.
  - Intelligent load balancing logic.
  - Scalable HPSS Mover architecture.
- High performance data movement via:
  - Tape striping for parallel movement of huge files to tape.
  - File aggregation to stream small files to tape.



Integrated GPFS/HPSS for Scalable HSM

- Prior to integrating HPSS with GPFS policy scans
  - HSM performance limited the size to which the file system could be scaled
  - The problem: HPSS determined HSM candidates by “tree walk” of the file system - recursive readdir() and stat()
  - Sequential stat() calls kill performance (small, random, sequential reads from disk)
  - For large (>1B file) file systems, tree walk can take days
- New Integrated GPFS/HPSS uses GPFS ILM policy scans
  - Readdir/stat replaced by parallel sort/merge
    - Conceptually similar to map-reduce
  - 2B files in 2 hours measured in the lab
  - SC06 demo: 275M files per hour *over WAN*





Questions?