Storage Challenges for Petascale Systems

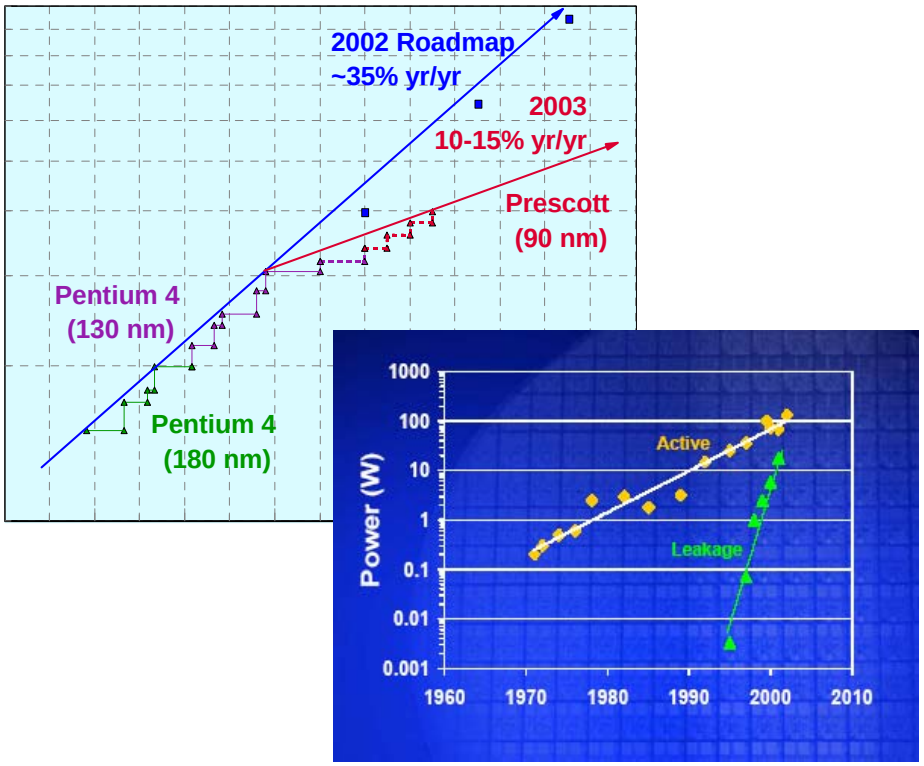
Dilip D. Kandlur
Director, Storage Systems Research
IBM Research Division

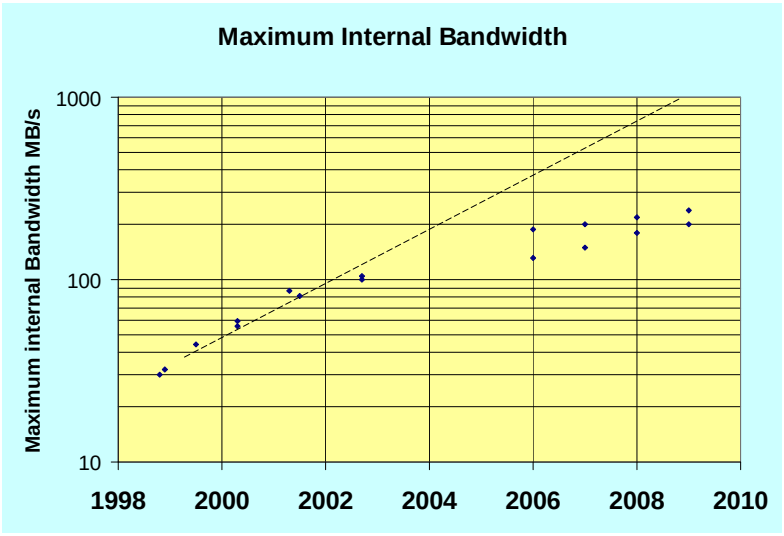
Outline

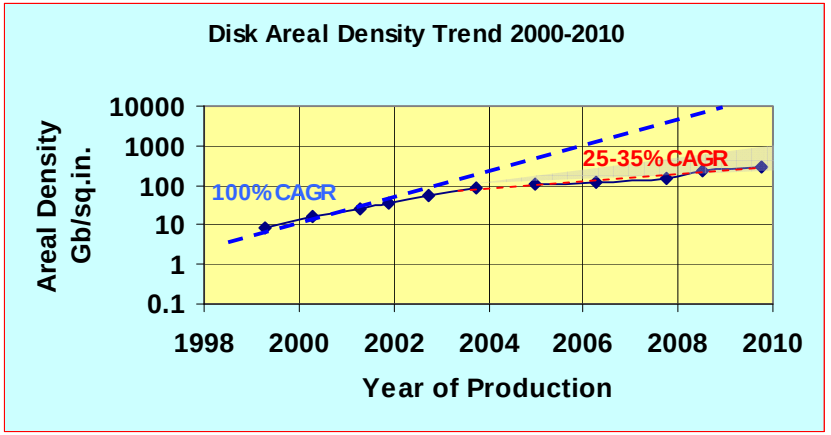
- Storage Technology Trends
- Implications for high performance computing
- Achieving petascale storage performance
- Manageability of petascale systems
- Organizing and finding Information

Extreme Scaling

- **There have been recent inflection points in CAGR of processing and storage – in the wrong direction!**
- **Programs like HPCS are aimed at maintaining throughput at or above the CAGR of Moore's Law in spite of these technology trends**





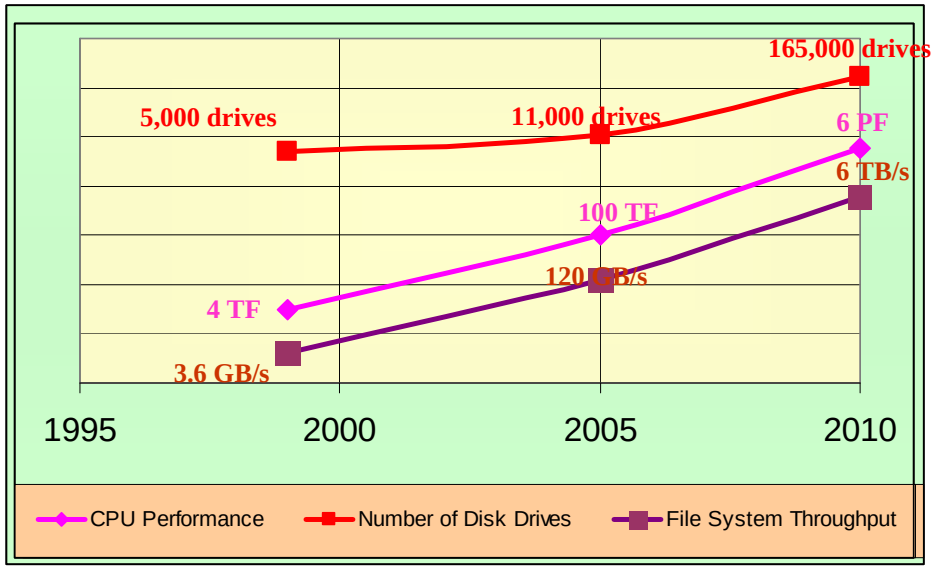


Peta-scale systems: DARPA HPCS, NSF Track 1

- HPCS goal: “Double value every 18 months” in the face of flattening technology curves
- NSF track 1 goal: at least a *sustained* petaflop for actual science applications
- New technologies like multi-core will keep processing power on the rise but will make storage relatively more expensive
- Maintaining “balanced system” scaling constants for storage will be expensive
  - Storage bandwidth: .001byte/second/flop capacty: 20 bytes/flop
  - Cost *per drive* will be same order of magnitude, so proportionally the same amount of storage will be a higher fraction of total system cost
- *How to make reliable a system with 10x today’s number of moving parts?*

System	Year	TF	GB/s	Nodes	Cores	Storage	Disks
Blue P	**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**
NSF Track 1 (possible)	**2011**	**2000**	**2000**	**10000**	**300000**	**40000 TB**	**50000**

HPCS Storage



Fast

5 TB/sec sequential bandwidth
30,000 file creates/sec on one node
Capable of running fsck on 1 trillion files

300,000 processors
150,000 disk drives

Robust

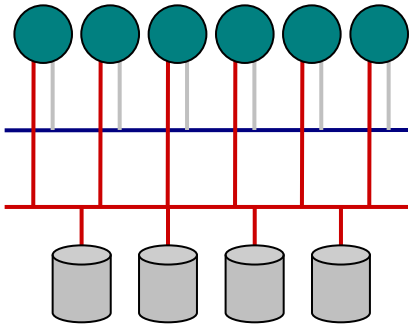
Fix 3 or more concurrent errors
Detect “undetected” errors
Only minor slowing during disk rebuild
Detect and manage slow disks

Managable

Unified manager for files, storage
End-end discovery, metrics, events
Managing system changes, problem fixes
GUI scaled to large clusters

GPFS Parallel File System

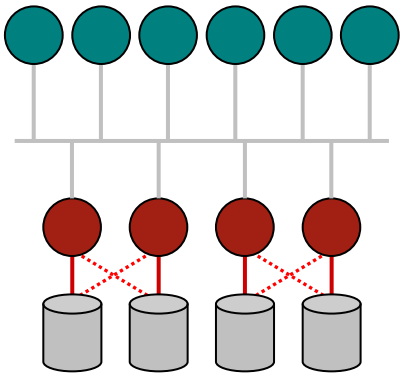
- *Cluster*: thousands of nodes, fast reliable communication, common admin domain.
- *Shared disk*: all data and metadata on disk accessible from any node, coordinated by distributed lock service.
- *Parallel*: data *and metadata* flow to/from all nodes from/to all disks in parallel; files striped across all disks.



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 disks

Scaling GPFS

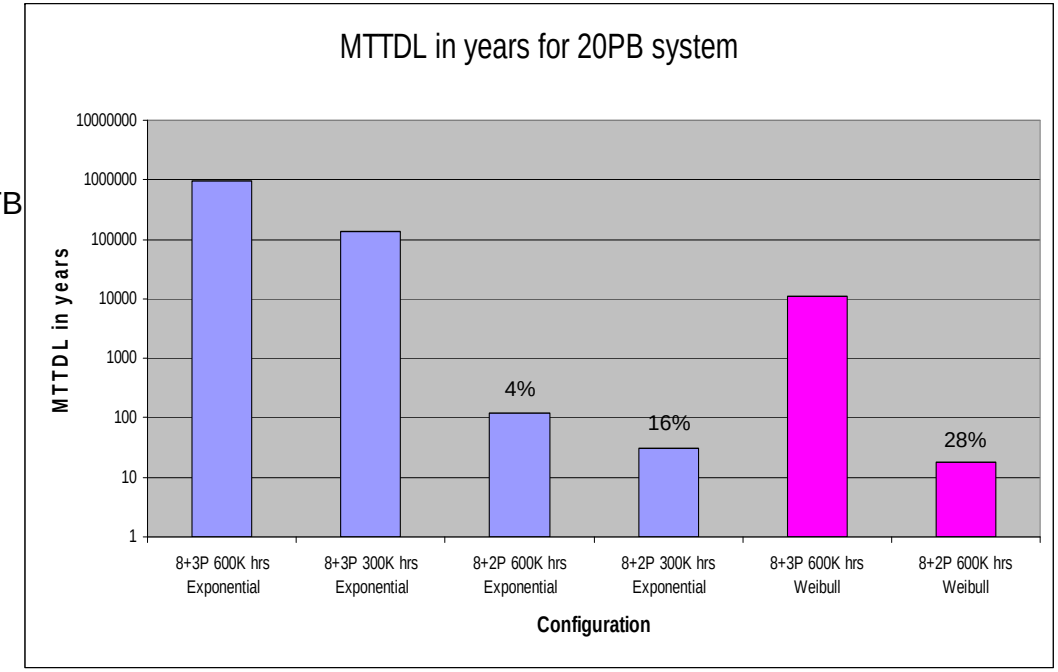
- HPCS file system performance and scaling targets
  - "Balanced system" DOE metrics (.001B/s/F, 20 B/F)
    - This means 2-6 TB/s throughput, 40-120 PB storage!!
  - Other performance goals
    - 30 GB/s single node to single file for data ingest
    - 30K file opens per second on a single node
    - 1 trillion files in a single file system
    - Scaling to 32K nodes (OS images)

Extreme Scaling: Metadata

- **Metadata: the on-disk data structures that represent hierarchical directories, storage allocation maps, …**
- **Why is it a problem? Structural integrity requires proper synchronization. Performance is sensitive to the latency of these (small) I/O's.**
- **Techniques for scaling metadata**
  - 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 provides for all these to some degree; work always ongoing
  - Sensible application design can make a big difference!

Data loss in Petascale Systems

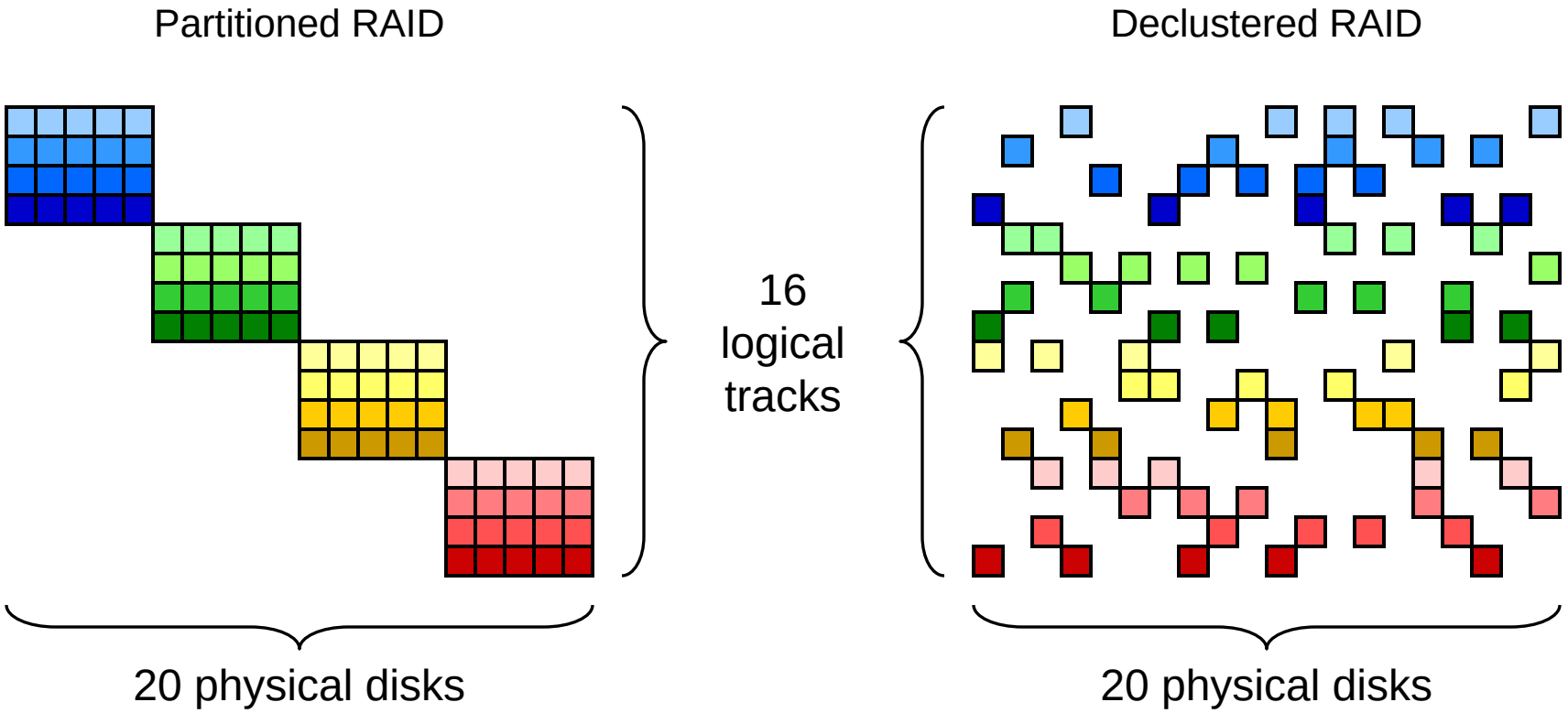
- **Petaflop systems require tens to hundreds of petabytes of storage**
- **Evidence exists that manufacturer MTBF specs may be optimistic (Schroeder & Gibson)**
- **Evidence exists that failure statistics may not be as favorable as simple exponential distribution**
- **Hard error rate of 1 in 10^{15} means one rebuild in 30 will get an error**
  - Rebuild of 8+P array of 500GB drives reads 4TB or 3.2×10^{13} bits
- **RAID-5 is dead at petascale; even RAID-6 may not be sufficient to prevent data loss**
  - Simulations of file system size, drive MTBF, failure probability distribution show 4%-28% chance of data loss over five-year lifetime for 8+2P code
  - Stronger RAID (8+3P) increase MTTDL by 3-4 orders of magnitude for extra 10% overhead.
  - Stronger RAID is sufficiently reliable even for unreliable (commodity) disk drives



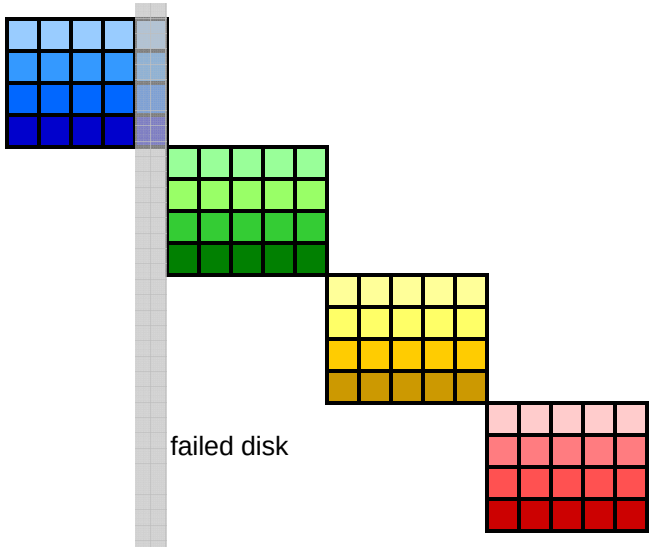
GPFS Software RAID

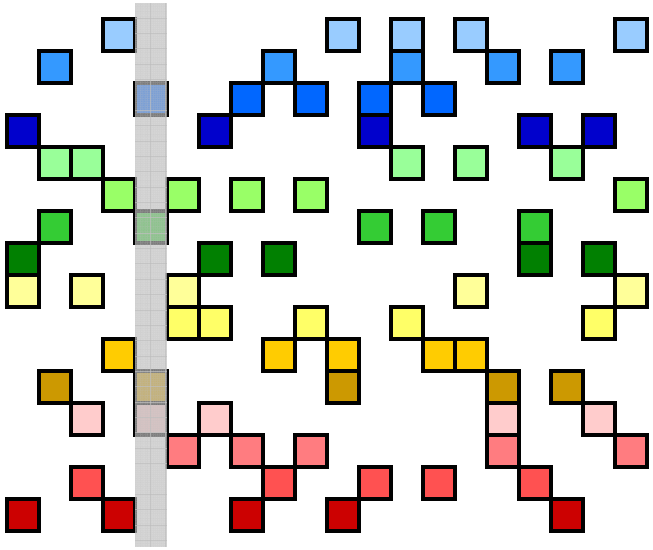
- Implement software RAID in the GPFS NSD server
- Motivations
  - Better fault-tolerance
  - Reduce the performance impact of rebuilds and slow disks
  - Eliminate costly external RAID controllers and storage fabric
  - Use the processing cycles now being wasted in the storage node
  - Improve performance by file-system-aware caching
- Approach
  - Storage node (NSD server) manages disks as JBOD
  - Use stronger RAID codes as appropriate (e.g. triple parity for data and multi-way mirroring for metadata)
  - Always check parity on read
    - Increases reliability and prevents performance degradation from slow drives
  - Checksum everything!
  - Declustered RAID for better load balancing and non-disruptive rebuild

Declustered RAID

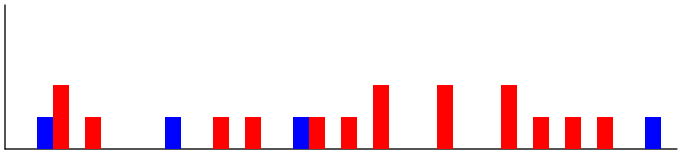


Rebuild Work Distribution

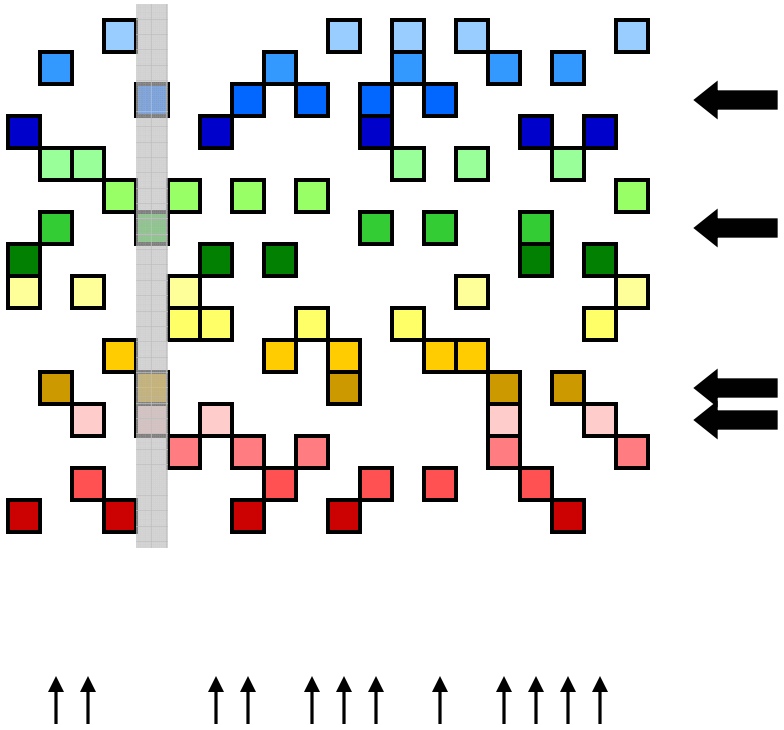




Relative read and write throughput for rebuild



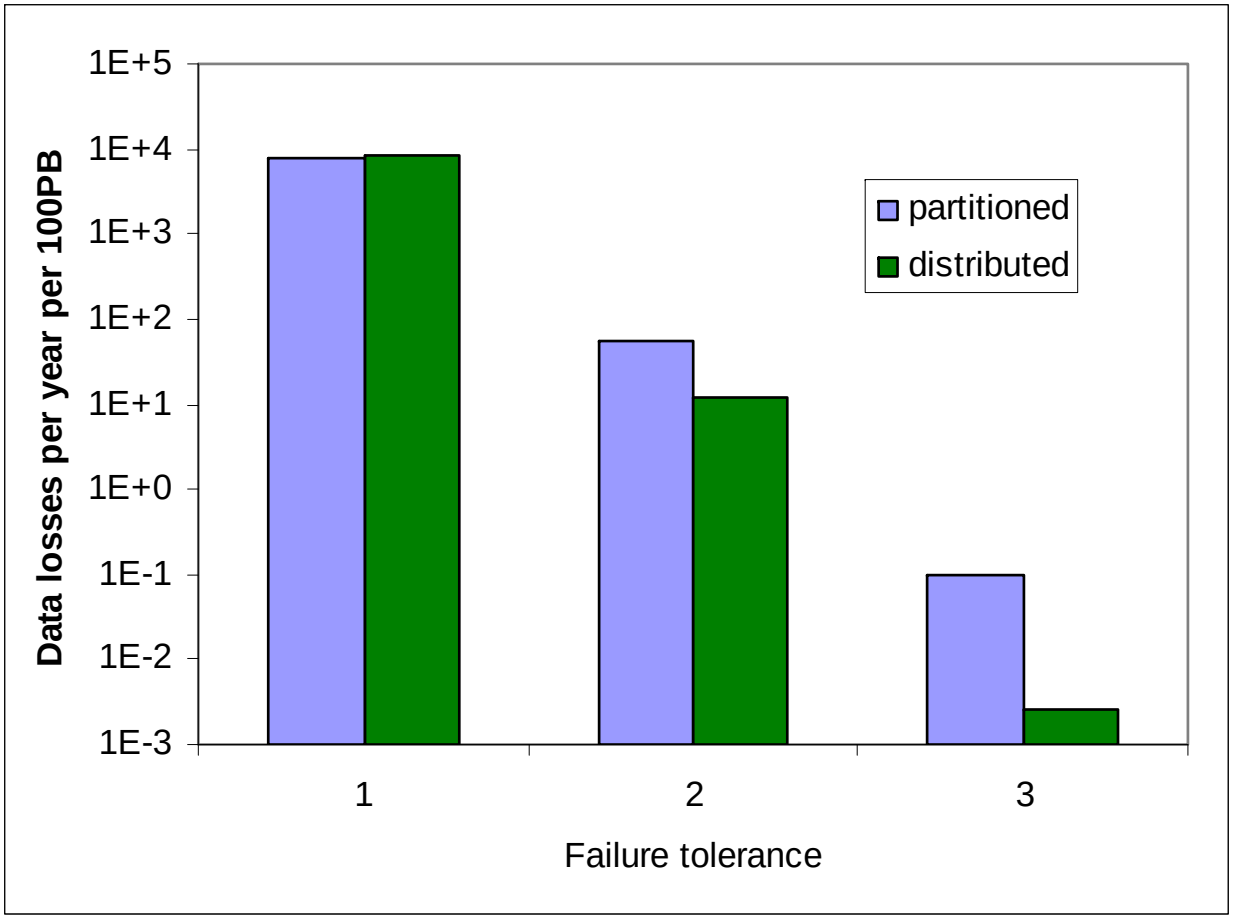
Rebuild (2)



Upon the first failure, begin rebuilding the tracks that are affected by the failure (large arrows).

Many disks involved in performing rebuild, so work is balanced, avoiding hot spots.

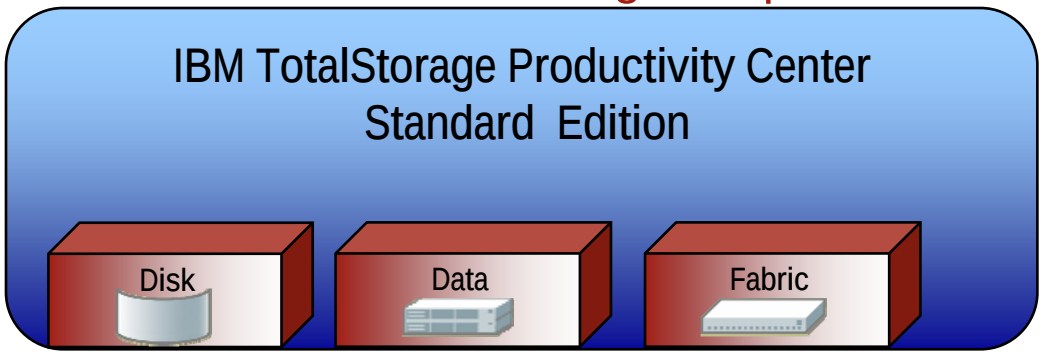
Declustered vs. Partitioned RAID



Simulation results

Autonomic Storage Management

Making Complex Tasks Simple



A Single Application

with modular components

Business Resiliency

Integrated Repication Manager
- Metro Disaster Recovery
- Global Disaster Recovery
- Cascaded Disaster Recovery

Application Disaster Recovery

Console Enhancements

End-to-End DataPath Explorer
- Integrated Storage Planner
- Configuration Change Rover
- Configuration Checker
- Personalization
- TSM Integration

Ease of Use

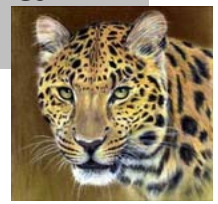
Streamlined Installation and Packaging
Single User Interface
Single Database
Single Set of services for consistent administration and operations

Policy Based Storage Management

SAN Best Practices
- SAN Configuration Validation
- Storage Subsystem Planning
- Fabric Security Planning
- Host Planning (Multi-path)

Integrated Management

Seamlessly integrate systems management across servers, storage and network & provide end-to-end problem determination and analytics capabilities



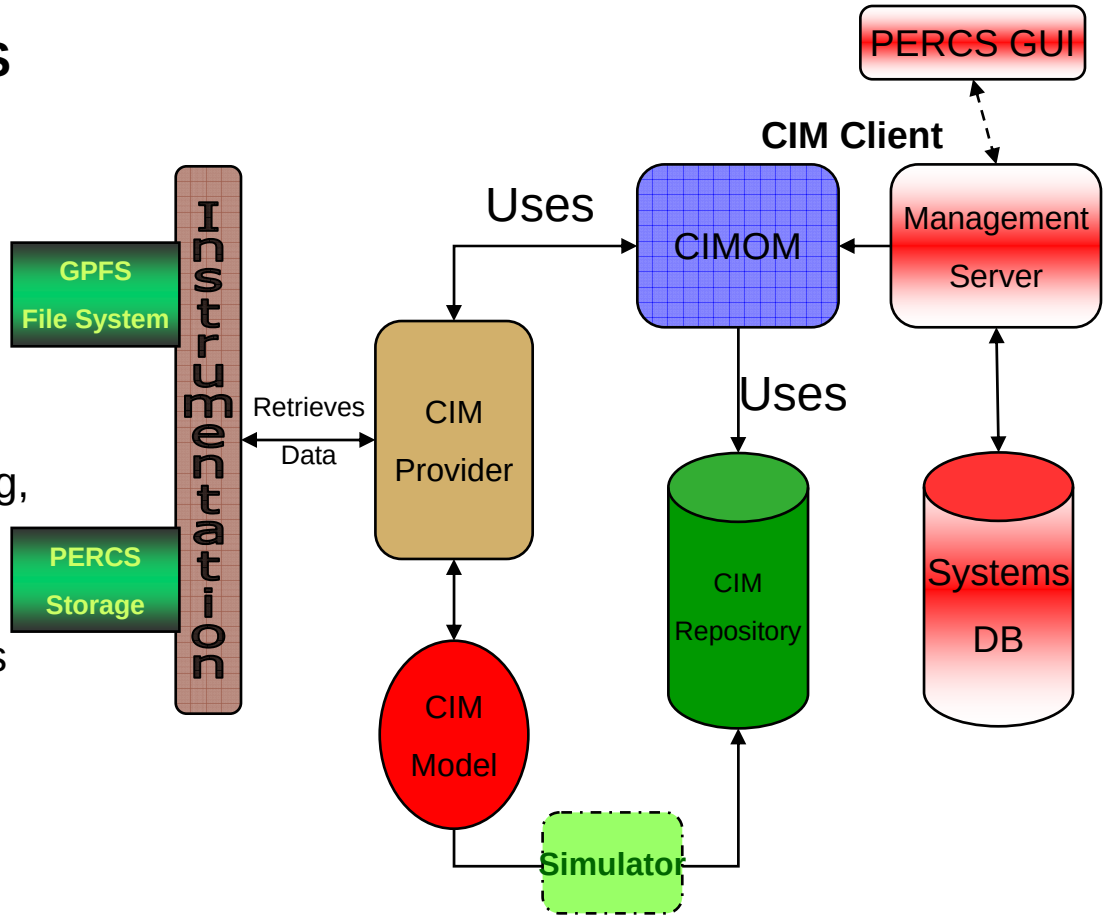
PERCS Management

- **A unified and standards based management for GPFS and PERCS Storage**

- **A GUI that is designed for large-scale clusters**
  - Supporting PERCS scale
  - GPFS

- **The PERCS UI will support:**
  - Information collection: asset tracking, end-end discovery, metrics, events
  - Management: system changes, problem fixes, configuration changes

- **Rich visualizations to help them maintain situational awareness of system status**
  - Essential for large systems
  - Also enable GPFS to satisfy commercial customers requiring ease-of-use



Analytics

- Problem Determination and Impact Analysis
  - Root cause analysis: discover the finest-grain events that indicate the root cause of the problem
  - Symptom suppression: correlate alarms/symptoms caused by a common cause across the integrated infrastructure
- Bottleneck Analysis
  - Post-mortem, live and predictive analysis
- Workload and Virtualization Management
  - Automatically monitor multi-tiered, distributed, heterogeneous or homogeneous workloads
  - Migrate virtual machines to satisfy performance goals
- Integrated Server, Storage and Network Allocation and Migration
  - Integrated allocation accounting for connectivity, affinity, flows, ports based on performance workloads
- Disaster Management
  - Provides integrated server/storage disaster recovery support

Visualization

- Integrated Management is centered around Topology Viewer capabilities based on Web 2.0 technologies
  - Data Path Viewer for Applications, Servers, Networks and Storage
  - Progressive Information Disclosure
  - Semantic Zooming
  - Information Overlays
  - Mixed Graphical and Tabular Views
  - Integrated Historical and Real Time Reporting

The Changing Nature of Archive

Current Archive: Data landfill

- Store and forget
- Not easily accessible, typically offline and offsite with access time measured in days
- Not organized for usage, retained just in case needed

Emerging Archive: Leverage information for business advantage

- Readily accessible, access time measured in seconds
- Indexed for effective discovery
- Mined for business value

Building Storage Systems Targeted at Archive

Scalability

- **Scale to huge capacity**
  - Exploit tiered storage with disk and tape
  - Leverage commodity disk storage
- **Handle extremely large number of objects**
  - Support high ingest rates
  - Effect data management actions in a scalable fashion

Emerging Archive: Leverage info for business advantage

Current Archive: Data landfill

Functionality

- **Consistently handle multiple kinds of objects**
- **Manage and retrieve based on data semantics**
  - E.g. Logical groupings of objects
- **Support effective search and discovery**
- **Provide for compliance with regulations**

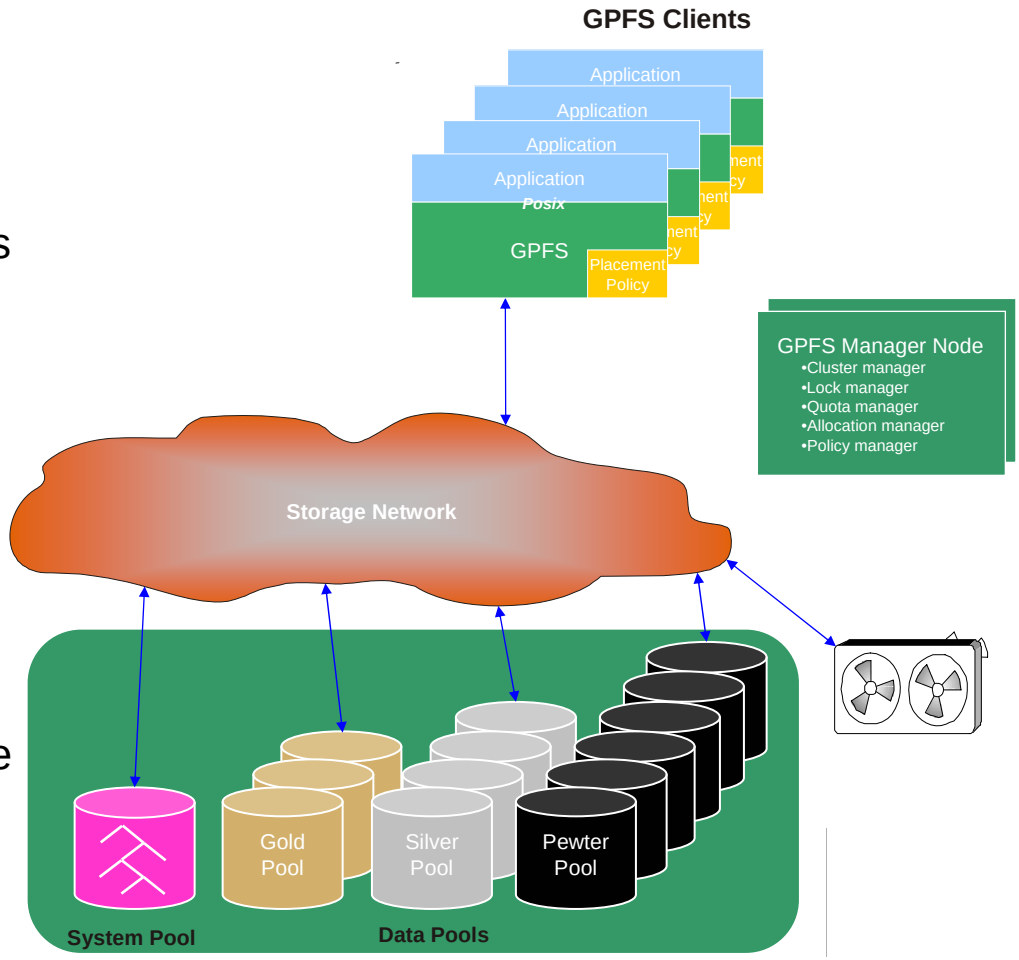
Reliability

- **Ensure data integrity and protection**
- **Provide media management and rejuvenation**
- **Support long-term retention**

GPFS Information Lifecycle Management (ILM)

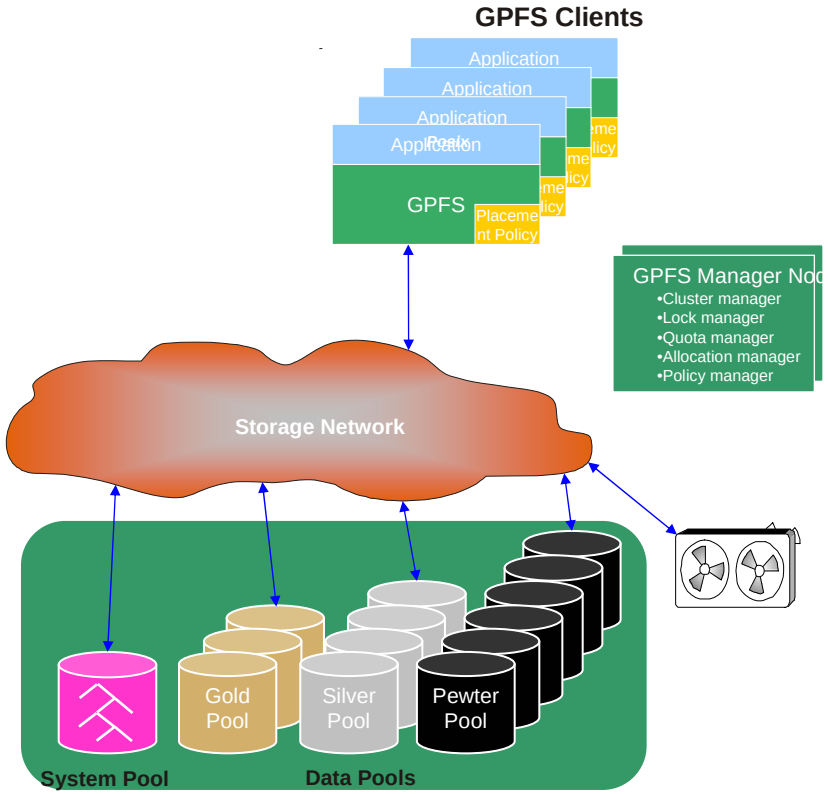
- **GPFS ILM abstractions**
  - Storage pool – group of LUNs
  - Fileset – subtree of a file system namespace
  - Policy – rule for file placement, retention, or movement among pools

- **ILM Scenarios**
  - Tiered storage – fast storage for frequently used files, slower for infrequently used files
  - Project storage – separate pools for each project, each with separate policies, quotas, etc.
  - Differentiated storage – e.g. place media files on media-friendly storage (QoS)



GPFS 3.1 ILM Policies

- Placement policies, evaluated at file creation, example
- Migration policies, evaluated periodically
- Deletion policies, evaluated periodically



GPFS Policy Engine

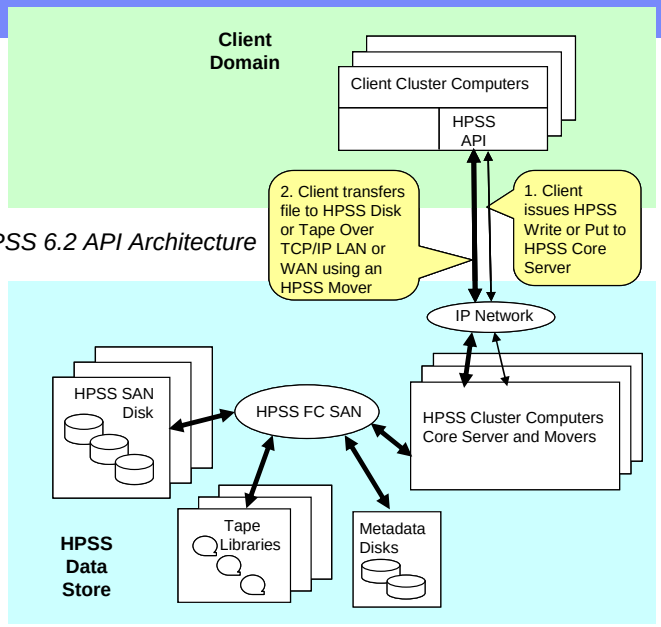
- **Migrate and delete rules scan the file system to identify candidate files**
  - Conventional backup and HSM systems also do this
    - Usually implemented with readdir() and stat()
    - This is slow – random small record reads, distributed locking
    - Can take hours or days for a large file system
- **GPFS Policy Engine uses efficient sort-merge rather than slow readdir()/stat()**
  - Directory walk builds list of path names (readdir() but *no stat()*!)
  - List sorted by inode number, merged with inode file, then evaluated
  - Both list building and policy evaluation done in parallel on all nodes
  - … > 10^5 files/sec *per node*!

Storage Hierarchies – the old way

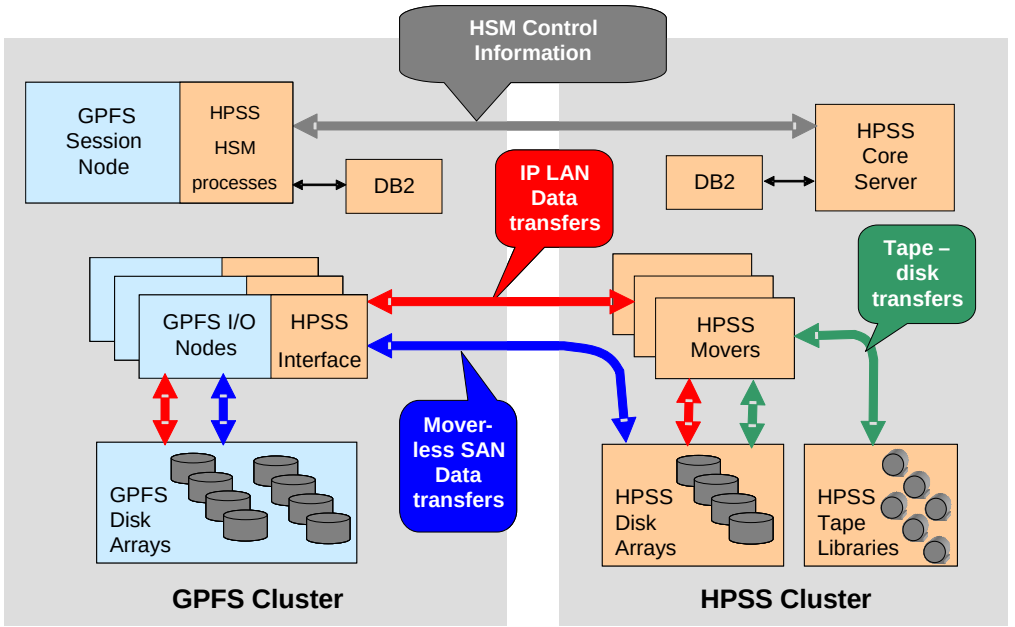
Normally implemented one of two ways:

- Explicit control
  - archive command (IBM TSM, Unitree)
  - copy into special “archive” file system (IBM HPSS)
  - copy to archive server (HPSS, Unitree)
  - … all of which are troublesome and error-prone for the user

- Implicit control through an interface like DMAPI
  - File system sends “events” to HSM system (create/delete, low space)
  - Archive system moves data and punches “holes” in files to manage space
  - Access miss generates event; HSM system transparently brings file back



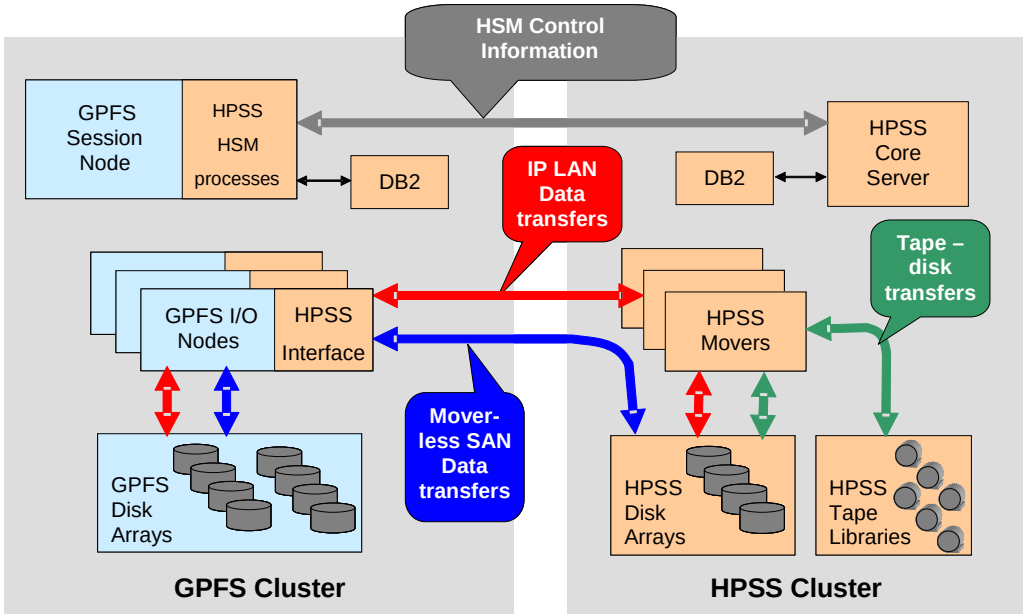
HPSS 6.2 API Architecture



GPFS 3.1 and HPSS 6.2 DMAPI Architecture

DMAPI Problems

- Namespace events (create, delete, rename)
  - Synchronous and recoverable
  - Each is multiple database transactions
  - Slow down the file system
- Directory scans
  - DMAPI low-space events trigger directory scans to determine what to archive
    - can take hours or days on large FS
  - Scans have little information upon which to make archiving decisions (what you get from "ls –l")
    - As a result, data movement policies are usually hard-coded and primitive
- Read/write managed region
  - Blocks the user program while data brought back from HSM system
  - Parallel data movement isn't in the spec, but everyone implements it anyway
  - Data movement is actually the one thing about DMAPI worth saving



GPFS 3.1 and HPSS 6.2 DMAPI Architecture

GPFS Approach: “External Pools”

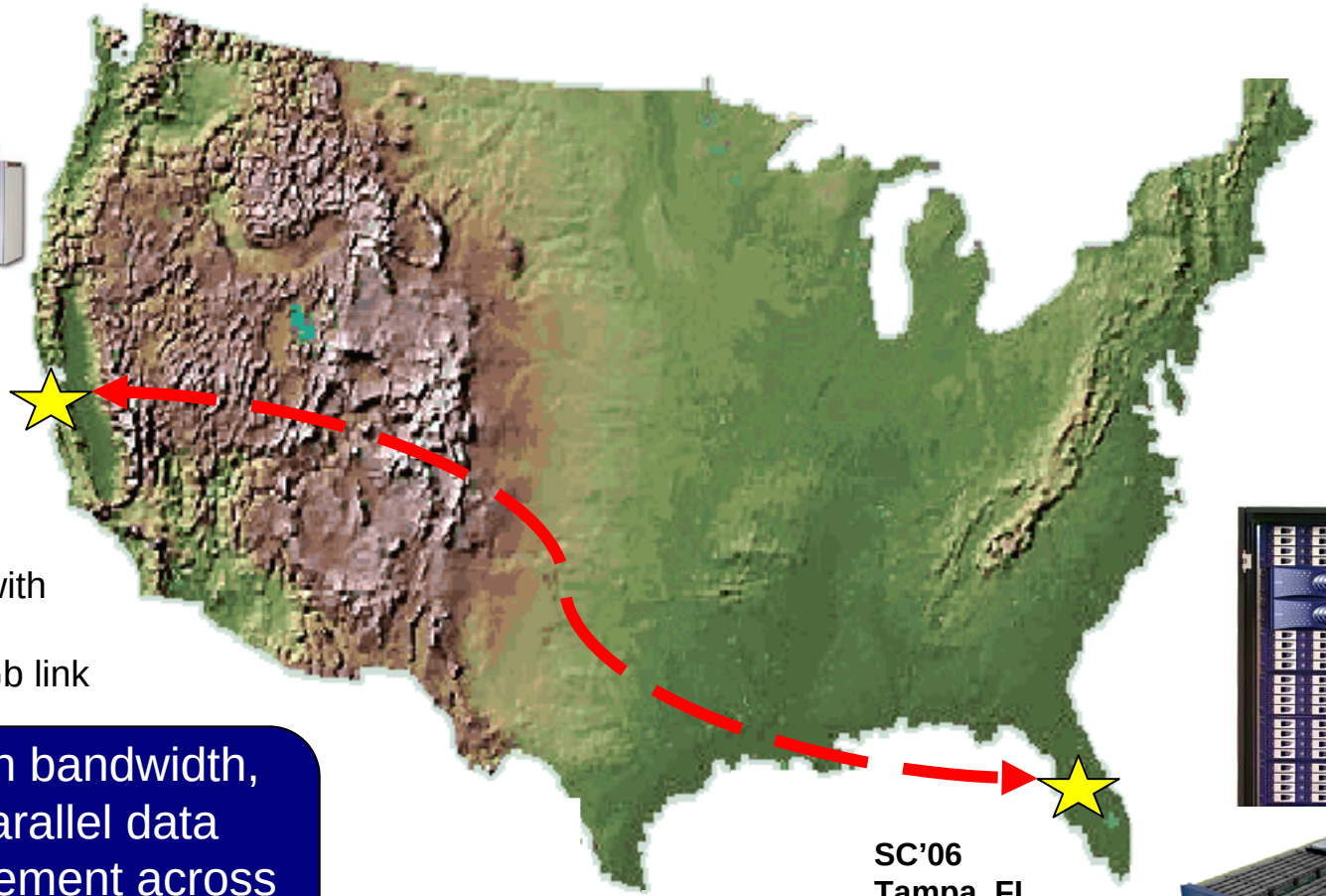
- **External pools are really interfaces to external storage managers, e.g. HPSS or TSM**
  - External pool “rule” defines script to call to migrate/recall/etc. files

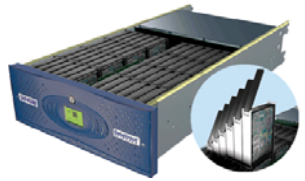
    RULE EXTERNAL POOL ‘PoolName’ EXEC ‘InterfaceScript’ [OPTS ’options’]
- **GPFS policy engine builds candidate lists and passes them to external pool scripts**
- **External storage manager actually moves the data**
  - Using DMAPI managed regions (read/write invisible, punch hole)
  - Or using conventional Posix API’s

GPFS ILM Demonstration

NERSC
Oakland, CA

HPSS
Archive on tapes with disk buffering, connected via 10Gb link



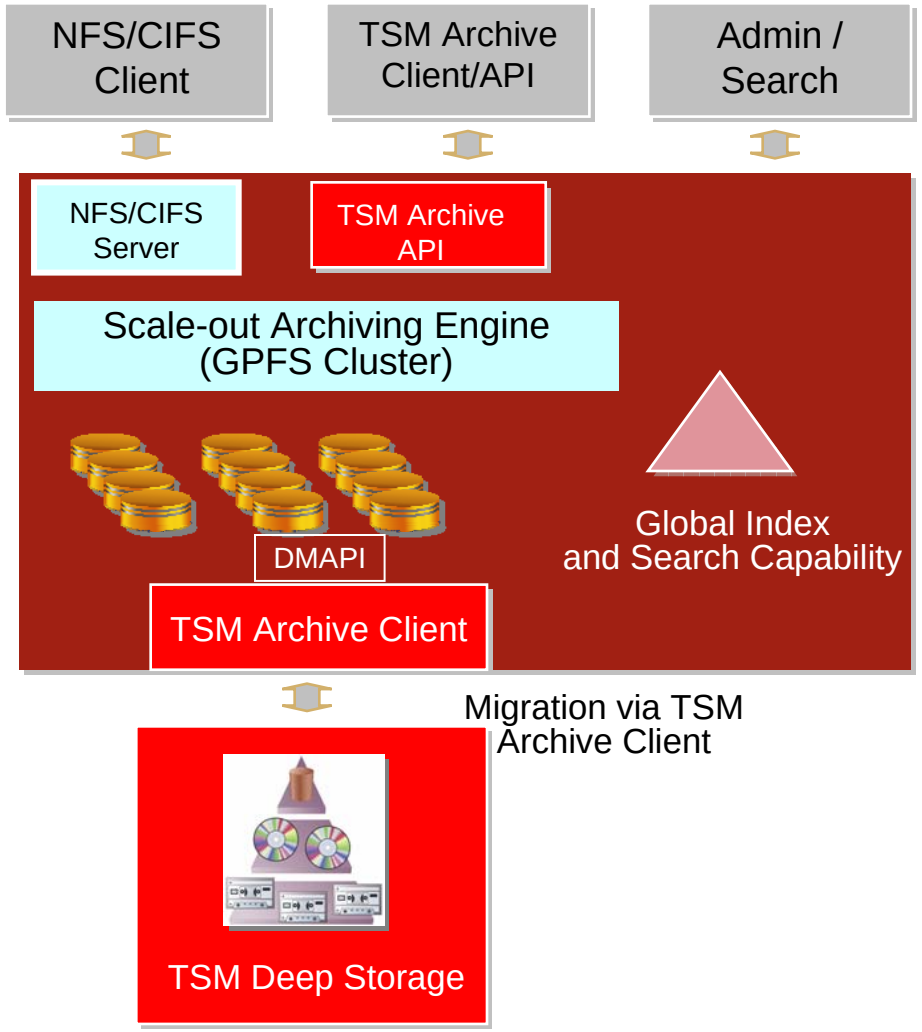


SC'06
Tampa, FL

GPFS
1M active files
FC, SATA disks

High bandwidth, parallel data movement across all devices and networks

Nearline Information – conceptual view



- **Provides capability to handle extended meta-data**
- **Meta-data may be derived from data content**
- **Extended attributes (integrity code, retention period, retention hold status, and any application meta-data)**
- **Global index on content and EA meta-data**
  - **Allow for application-specific parsers (e.g., DICOM)**

Summary

- Storage environments moving from petabytes to exabytes
  - Traditional HPC
  - New archive environments
- Significant challenges for reliability, resiliency, and manageability
- Meta-data becomes key for information organization and discovery