Storage for Petascale Computing

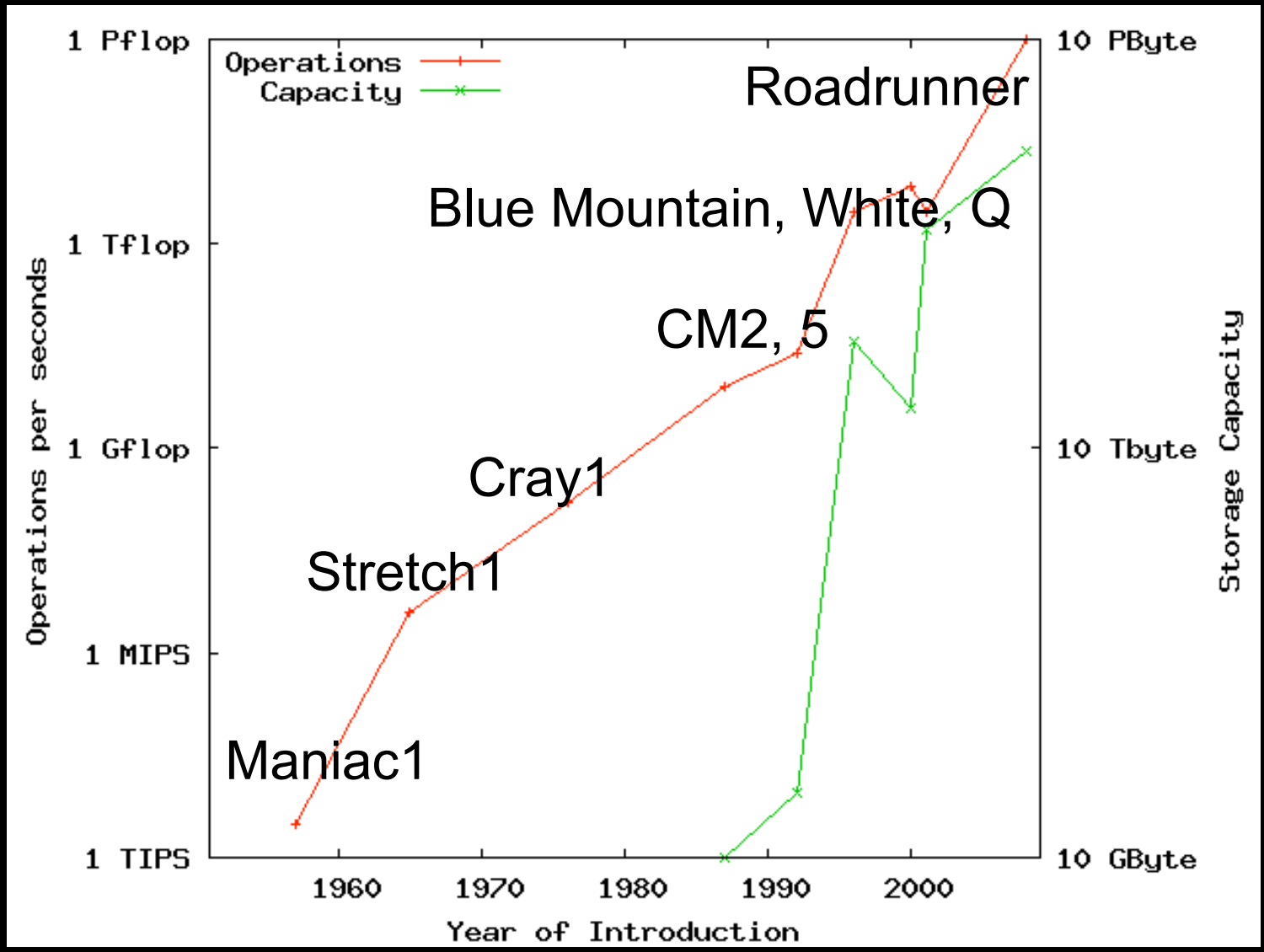
John Bent, LANL
(Gary Grider, James Nunez)

LA-UR-07-6213, LA-UR-07-0828, LA-UR-07-5075

Why Petascale?

- Scientific computing demands it
  - Bio-informatics, astro-informatics, climate modeling, materials design, high energy physics, intelligence, cyber security
- Modeling lends itself to parallelism
  - Detailed 3D modeling needs petascale

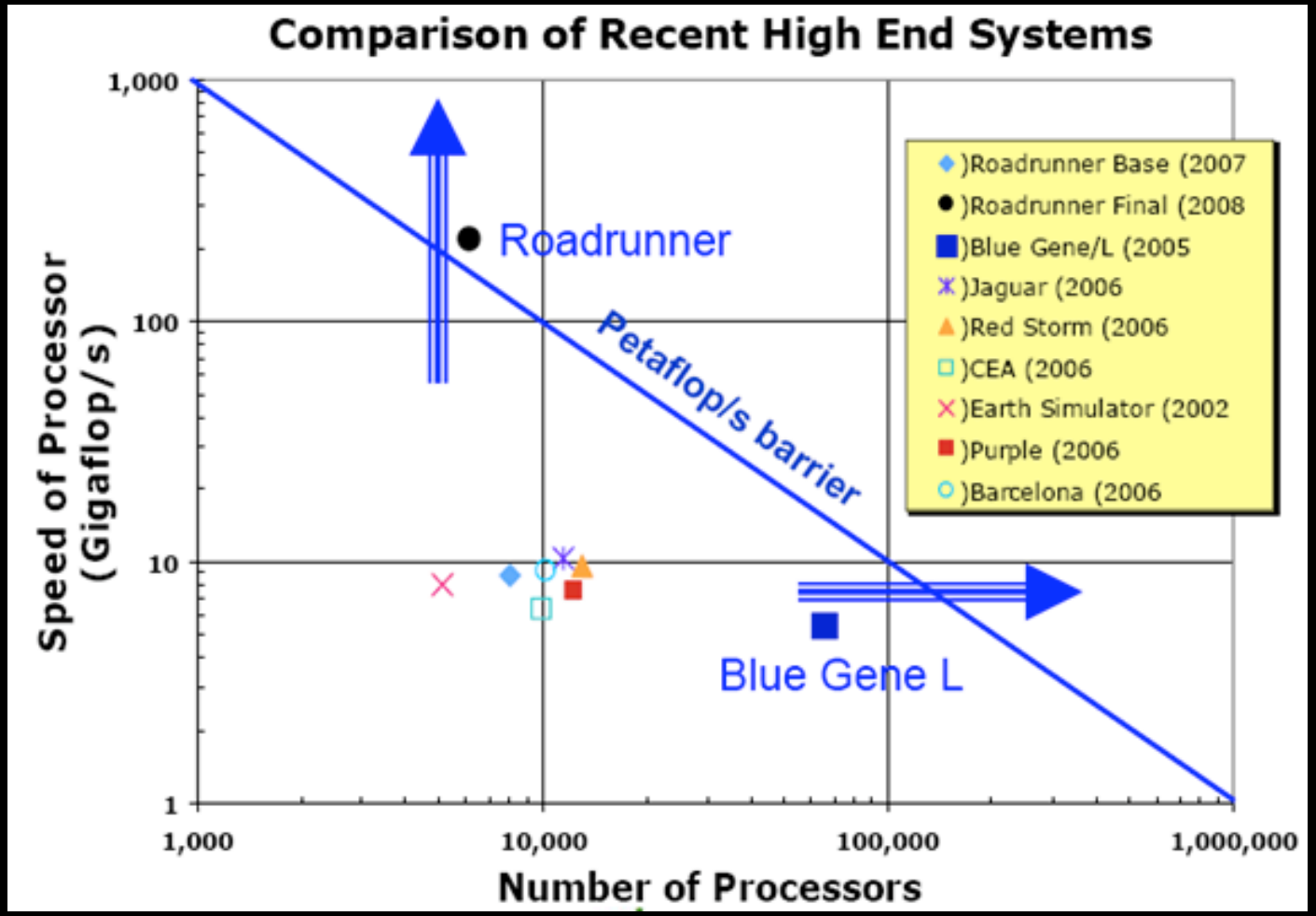
The Road to Petascale



The First Petaflop* Machine!!

* Not really
  - MDGrape-3 (June 2006)
  - folding@home (September 2007)

Roadrunner breaks the mold

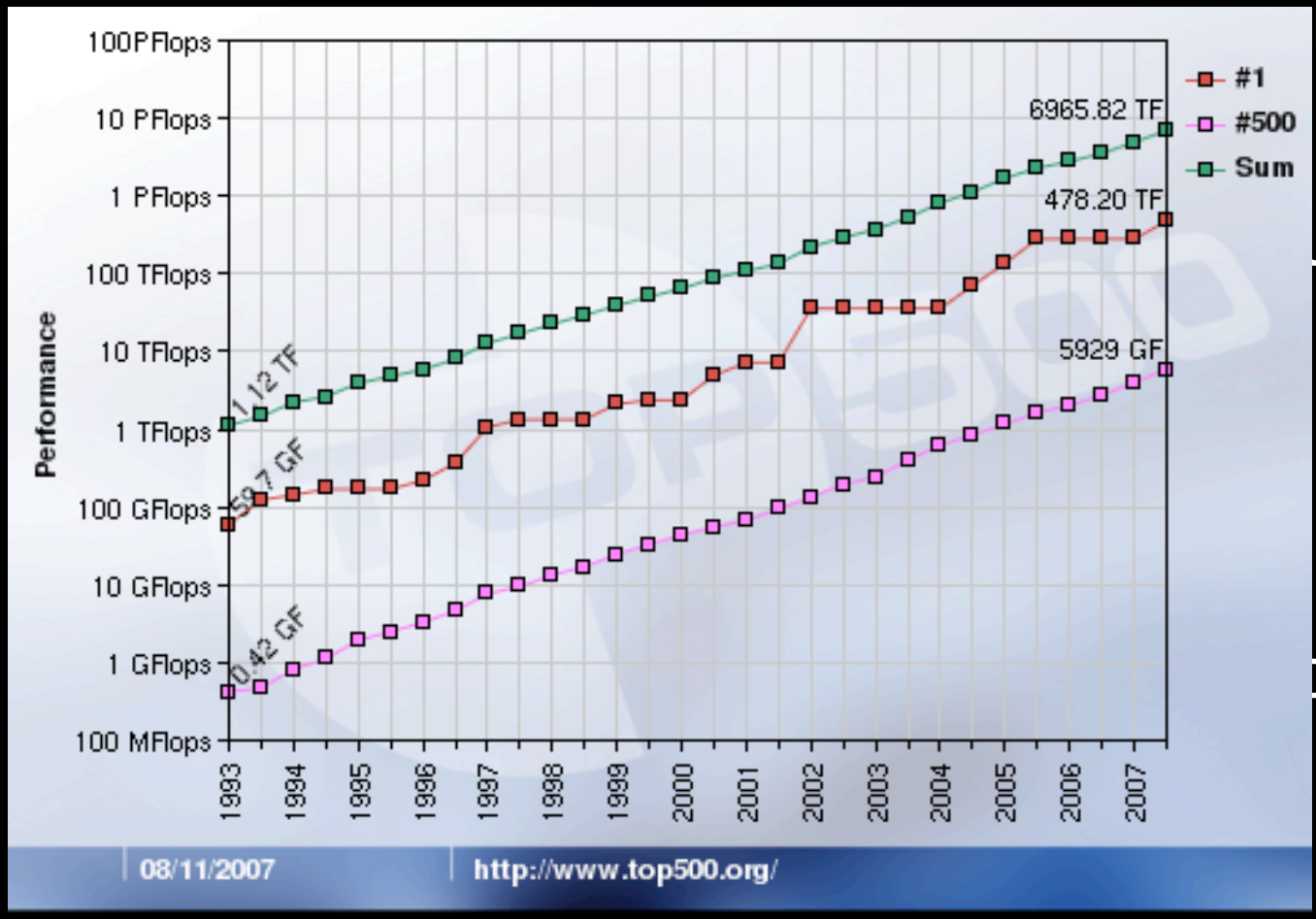


Only 6000 processors?

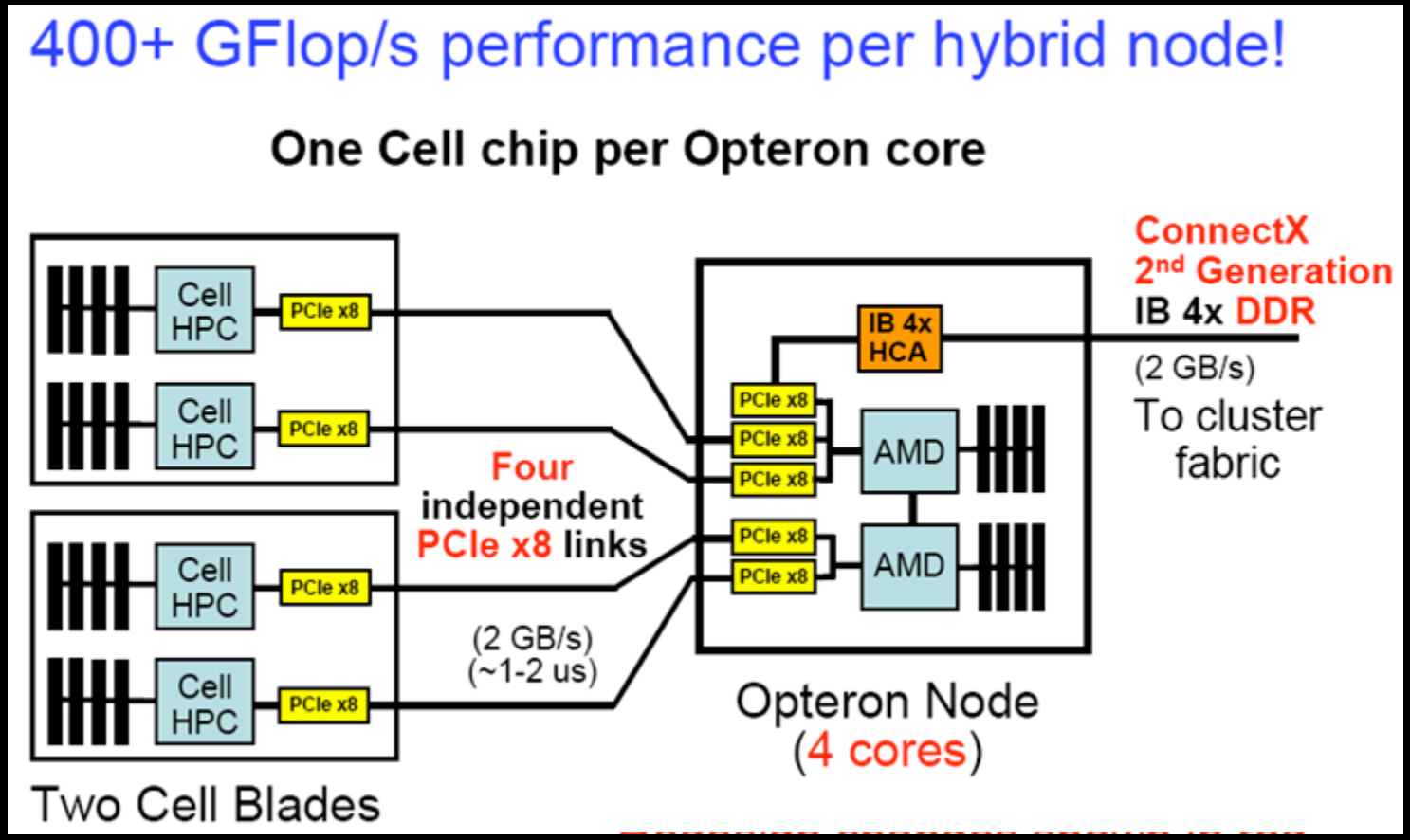
- 6,120 general purpose processors
  - AMD dual-core Opterons
  - ~ 50 Teraflops
- 12,240 cell chips
  - Each has 1 PPU controller and 8 SPU's
  - ~ 1.4 Petaflops
- Total
  - Only 6,120 processors
  - But 33,660 processing units

Hybrid Design

- u'd
- ovation



Four Cell Chips per Node (Triblade)

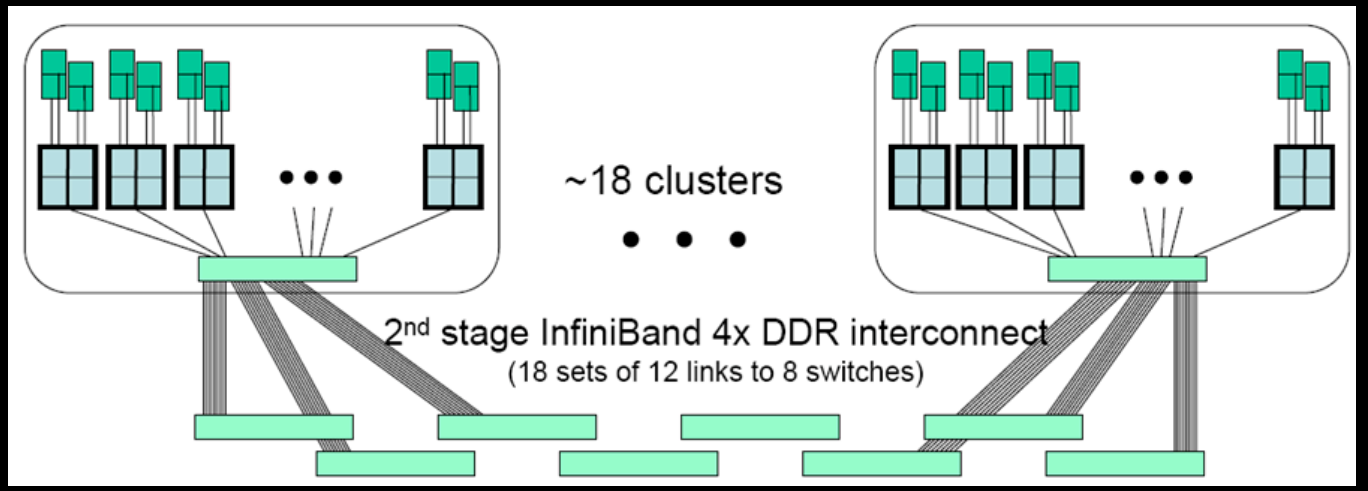


Hybrid or Hy-bad?

- Advantages
  - Less power
  - Less cooling
  - Less floorspace
  - Less cabling
  - Fast interconnect between SPU's
- and disadvantages
  - Complicated programming model

"Ankle bone's connected to the shin bone"

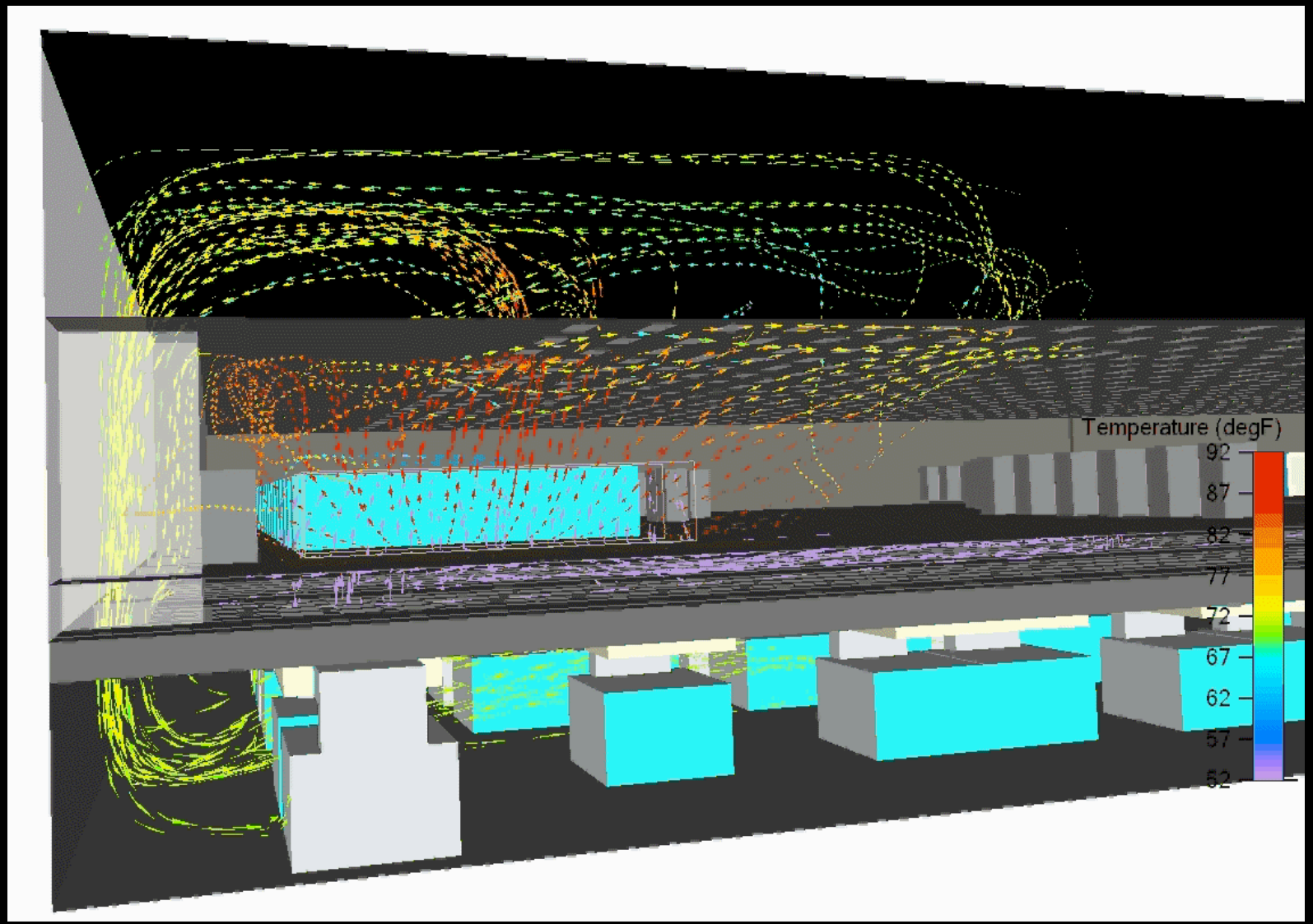
- Three triblades in a blade center
- Four blade centers in a rack
- 15 racks in a cluster (CU)
- 17 CU's in total
  - 17 * 15 * 4 * 3 = 3,060 triblades
  - 2 general purpose CPU's per triblade = 6,120
- CU's internally connected via Voltaire infiniband switch
- Externally connected by eight more



Not exactly plug and play

- Thousands of cables (tons and miles)
- Optimal IB routing config is NP-complete
- Entire software stack must be scaled
- Power and cooling must also be planned

Roadrunner not adapted for desert



- Rick Rivera, LANL

Wasn't this supposed to be a storage talk?

- Yes, but failure first
  - With 10's of thousands of processors, power supplies, fans, and 100's of thousands of memory dims, MTTI is at most a few hours.
- Failure must be dealt with
  - Apps must be prepared (checkpointing)
- Recent failure data and analysis
  - Google paper
    - http://labs.google.com/papers/disk_failures.pdf
  - CMU paper
    - http://www.cs.cmu.edu/~bianca/fast07.pdf
  - Netapp paper
    - http://opera.cs.uiuc.edu/~chu7/publication/fast08-jiang.pdf
  - LANL data
    - http://institute.lanl.gov/data/lanldata.shtml

Checkpointing

- Store all state needed to restart onto disk
- Application specific
  - Store all data structures
  - Restart mode loads from stored structures
- Application independent
  - Do complete memory dump (and registers)
  - Reload old state and resume execution
  - Can be done with virtual machines or scheduler
  - Difficult for parallel jobs (needs synchrony)
  - Doesn’t allow N-M checkpoint/restart
- LANL apps do own specific checkpointing
  - Because they are parallel and also . . .
  - Because they want N-M checkpoint/restart
  - Dumps also used for visualizing partial results

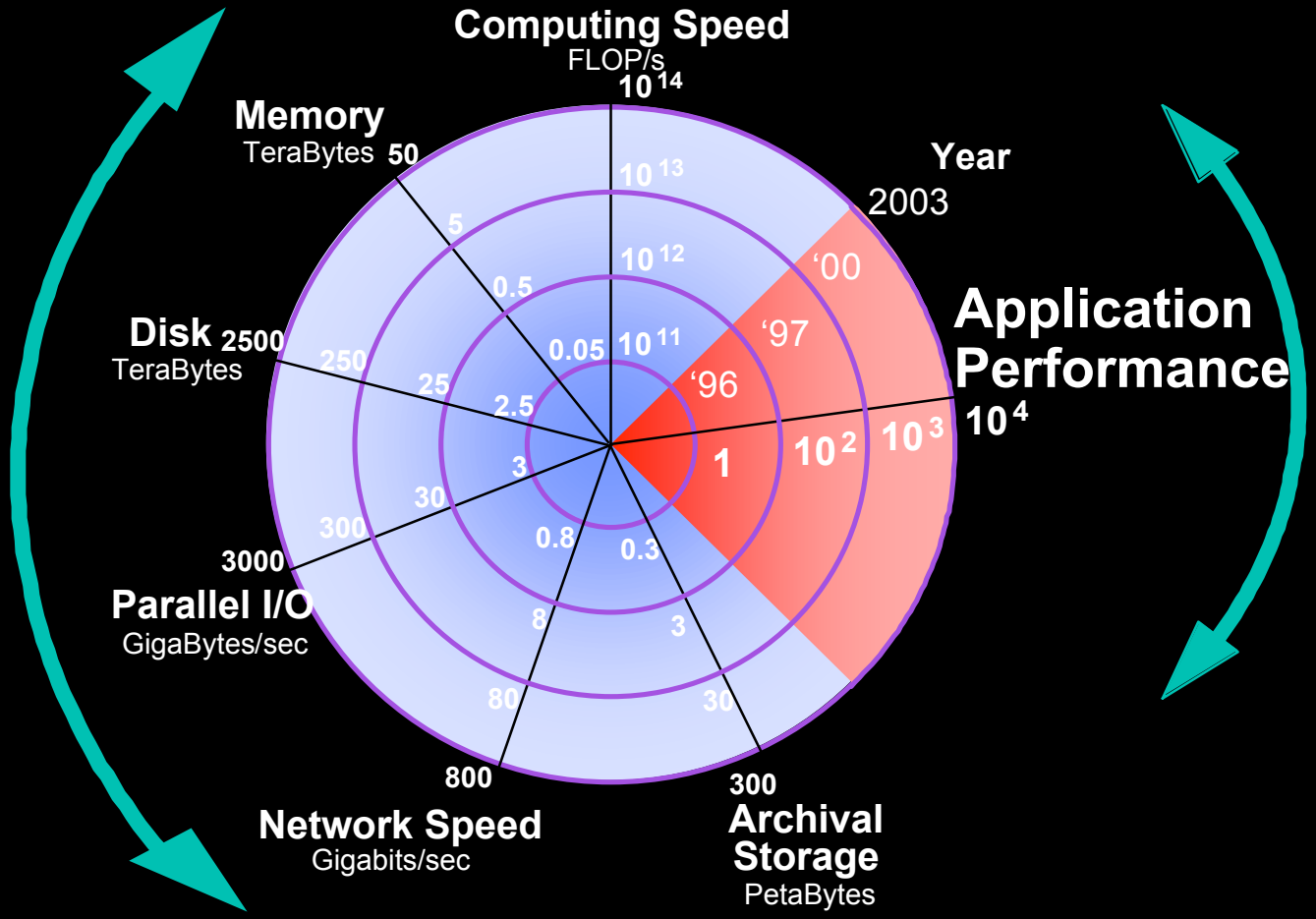
Finally storage

- How much storage do supercomputers need?
  - A little for final results
  - A lot for checkpoint dumps
- How much bandwidth is needed?
  - A little for final results
  - A lot for checkpoint dumps

Roadrunner storage

- High bandwidth
  - About one PB of scratch space
- Lower bandwidth
  - Couple hundred TB of project space
  - Several PB of archive

How much storage?



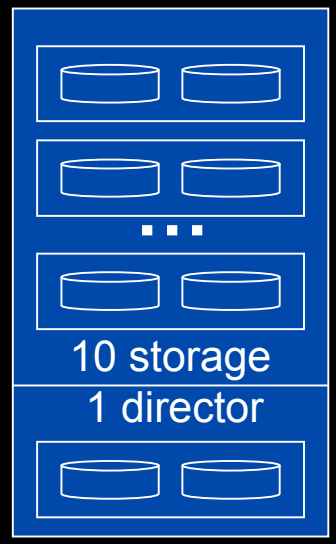
Old rule of thumb.

New storage rule of thumb

- Checkpoint in less than 30 minutes
  - At least 90% of machine time should be useful
- Assume entire memory is used
  - 17 CU's * 15 racks * 12 triblades * 32 GB
  - ~ 100 TB's of memory
- ~ 58 GB/s of parallel I/O is needed

Roadrunner Parallel File System

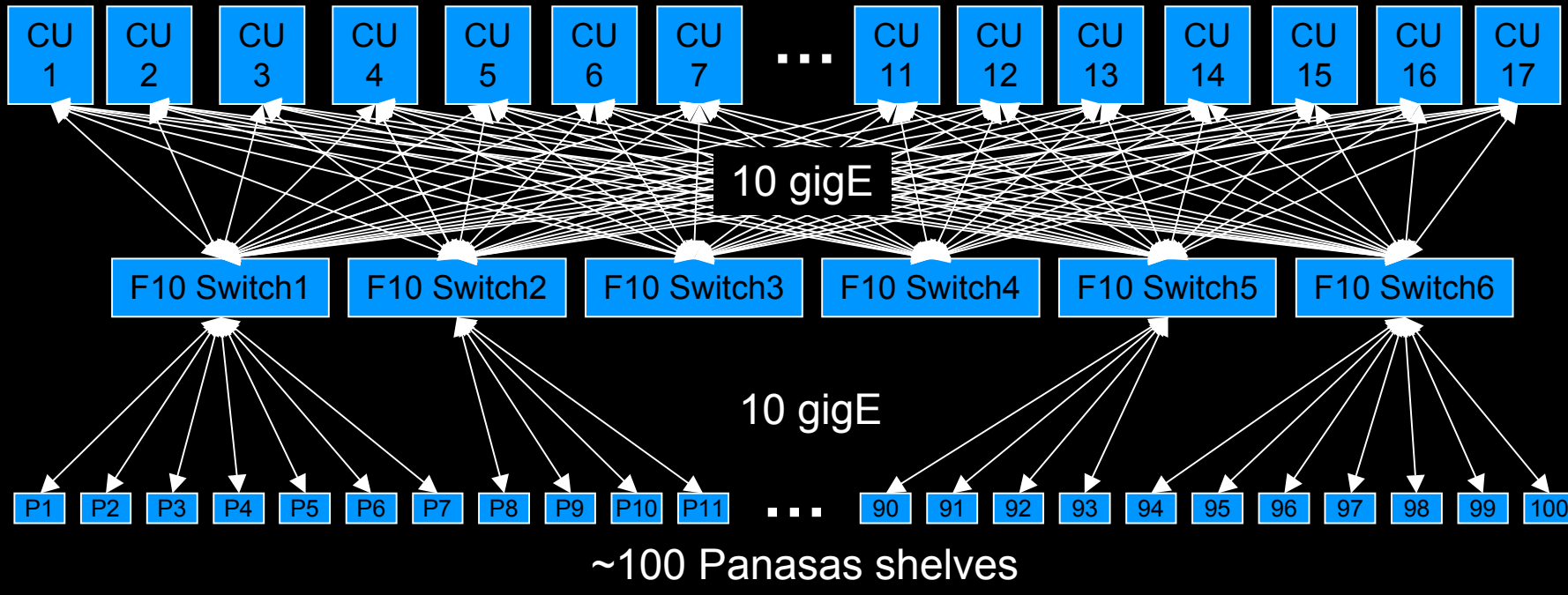
- Comprised of Panasas blades (2 disks)
- Physically organized into shelves
  - 1 director blade, 10 storage blades
- Logically organized into volumes
  - 1 director blade, ALL storage blades
- Object storage system
  - Director blades are metadata servers
- RAID 5 or RAID 10 on a per-object basis
  - Client computes parity
- Extremely fast rebuild
  - Important to us for availability not recoverability



Panasas Shelf

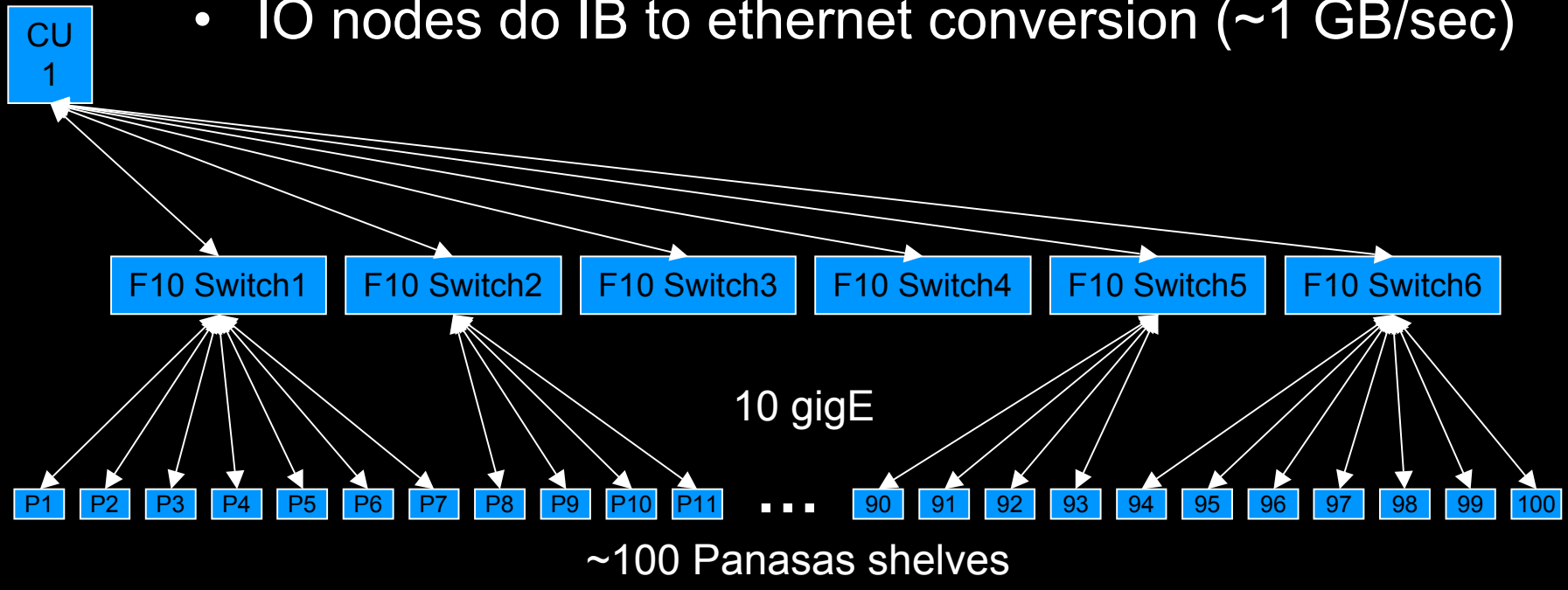
“Ankle bone, shin bone…”

- 58 GB/s requires
  - ~100 shelves (around 500 MB/s per shelf)
  - Fully connected fat tree ($$$) or six F10’s ($$)

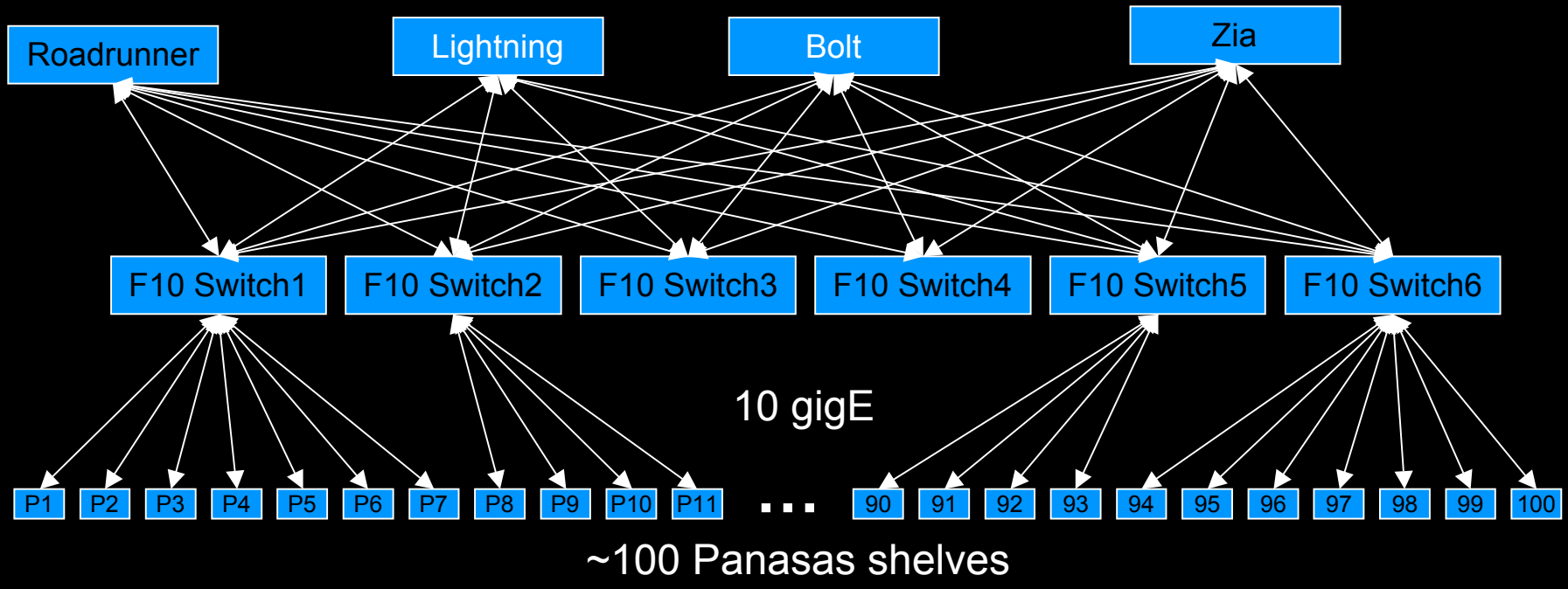


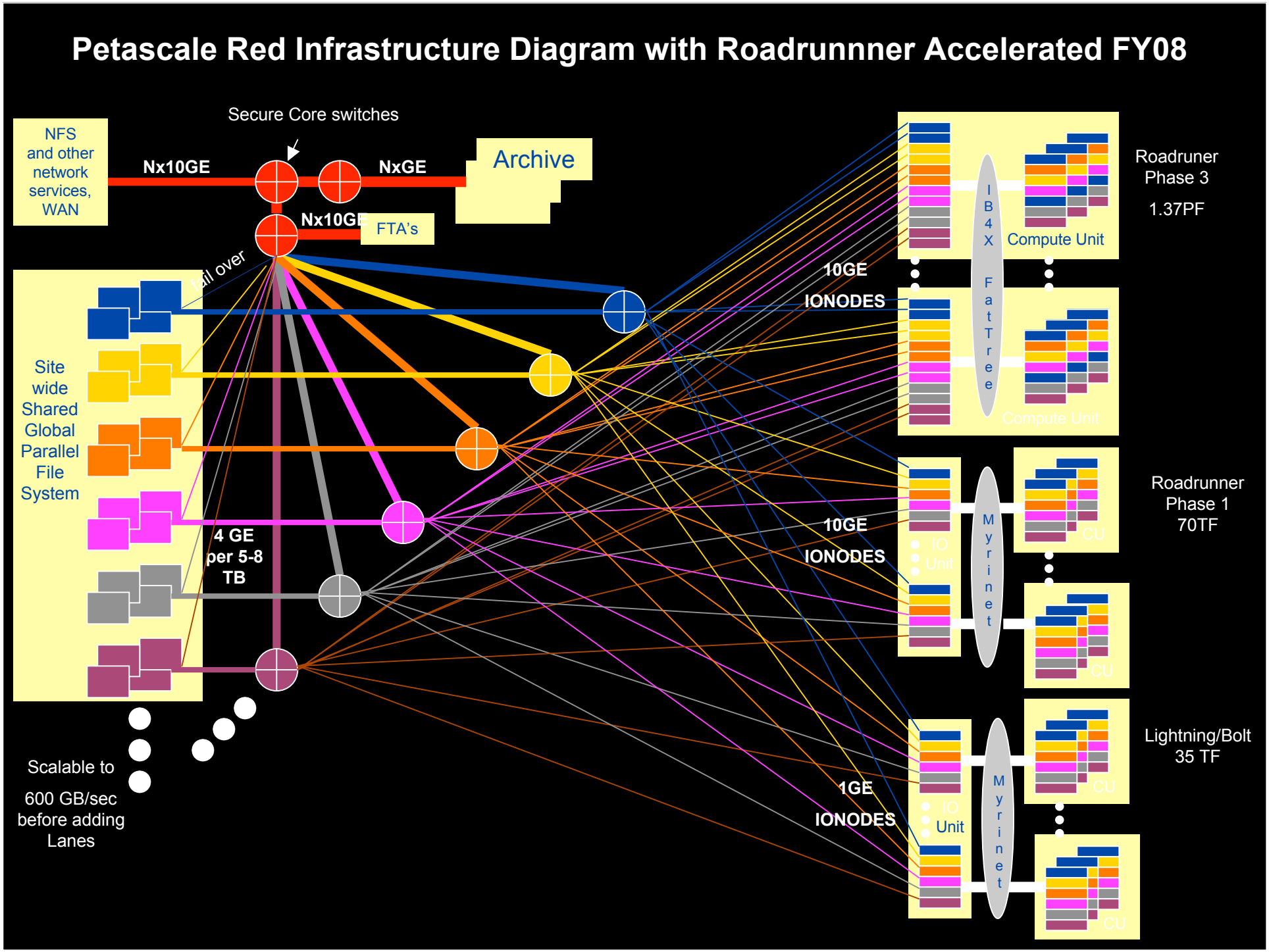
LANL PaScalBB

- Each CU has 190 nodes, 180 compute, 12 IO
- Network divided into six subnets
  - Each IO node has two 10 gige connections to one subnet
  - Storage split over subnets
- IO nodes do IB to ethernet conversion (~1 GB/sec)



Shared storage infrastructure

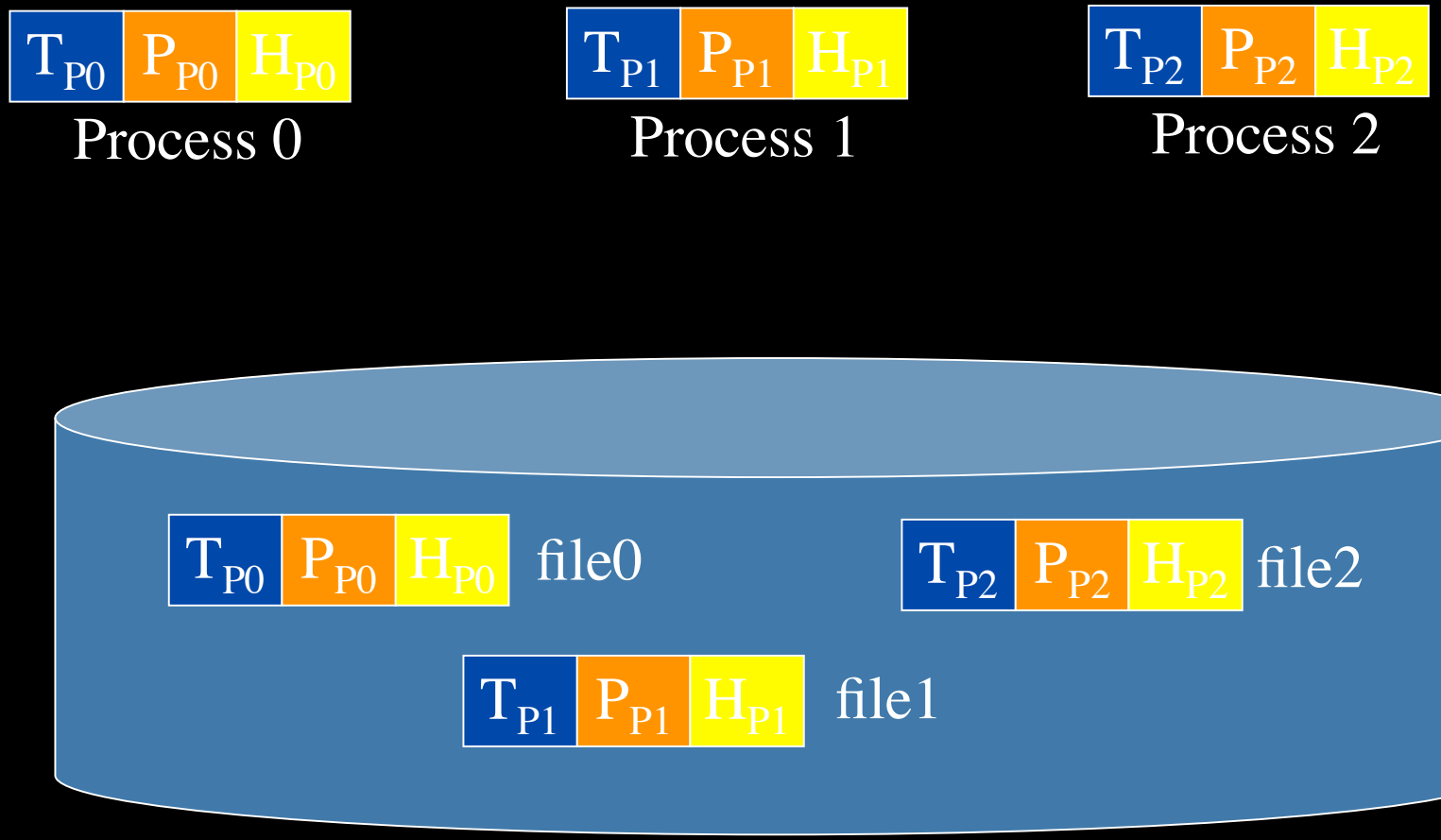



Petascale Red Infrastructure Diagram with Roadrunnner Accelerated FY08
NFS and other network services, WAN
Secure Core switches
Nx10GE
NxGE
Archive
Nx10GE
FTA's
fail over
Site wide Shared Global Parallel File System
4 GE per 5-8 TB
Scalable to 600 GB/sec before adding Lanes
10GE IONODES
10GE IONODES
1GE IONODES
IB 4X FatTree
Compute Unit
Compute Unit
IO Unit
Myrinet
CU
CU
IO Unit
Myrinet
CU
CU
Roadruner Phase 3 1.37PF
Roadrunner Phase 1 70TF
Lightning/Bolt 35 TF

Challenges: Short-term

- IO patterns
  - N-N: no problem
  - N-1 non-strided: no problem
  - N-1 strided: problematic

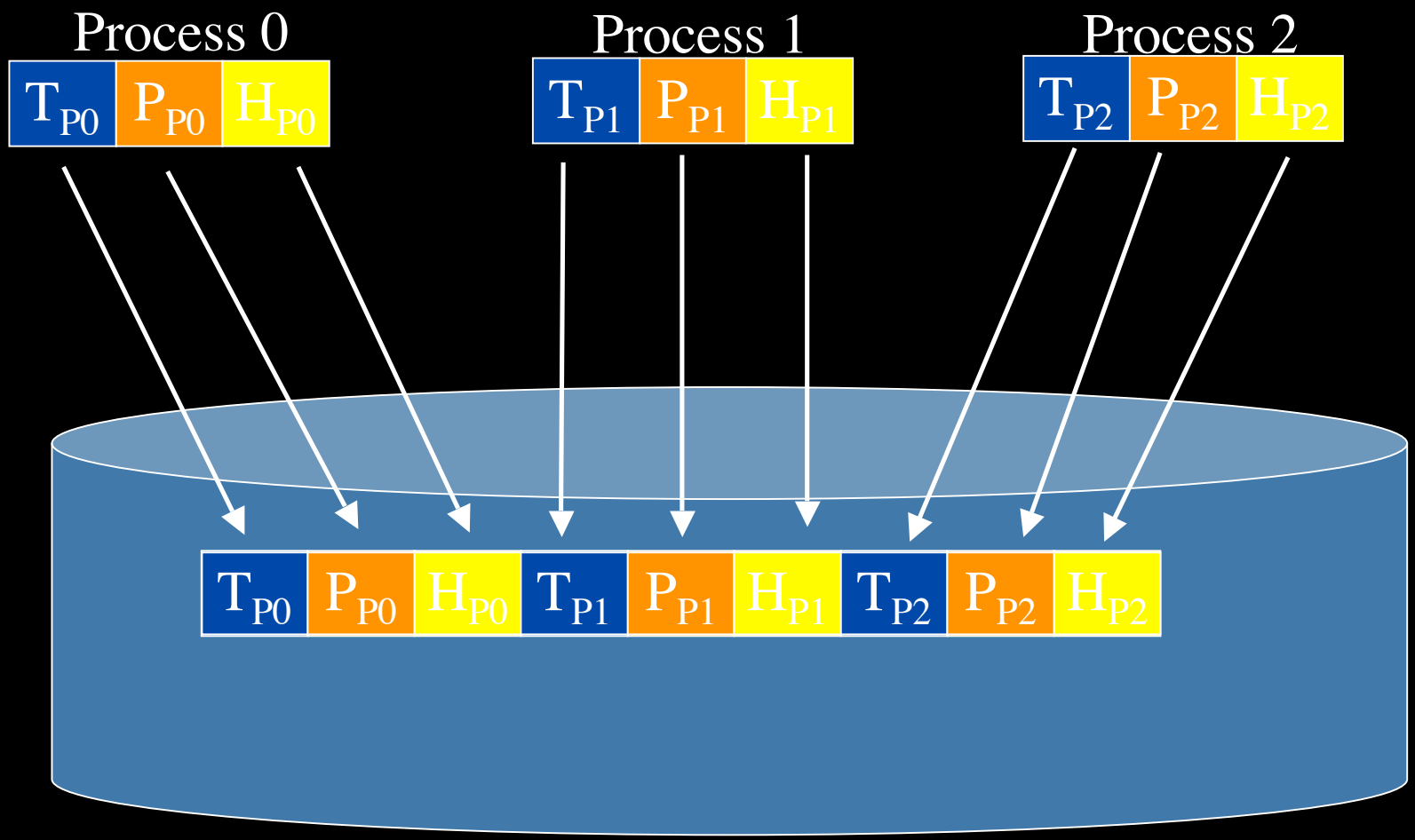
N-to-N example



N-N evaluation

- Advantages
  - Reads and writes can be very fast
    - Writes can be aggregated into large writes only.
  - Files can be stored on local disks
- Disadvantages
  - Multiple simultaneous file creates in a single directory
  - Collecting, managing, and archiving multiple files
    - Running simulations for months creates millions of files to deal with
  - N-to-M restart: Restart on different number of processors
    - Collect all N files
    - Recombine into the mesh
    - Divide and distribute to M processes
  - Visualization of partial results similarly complex

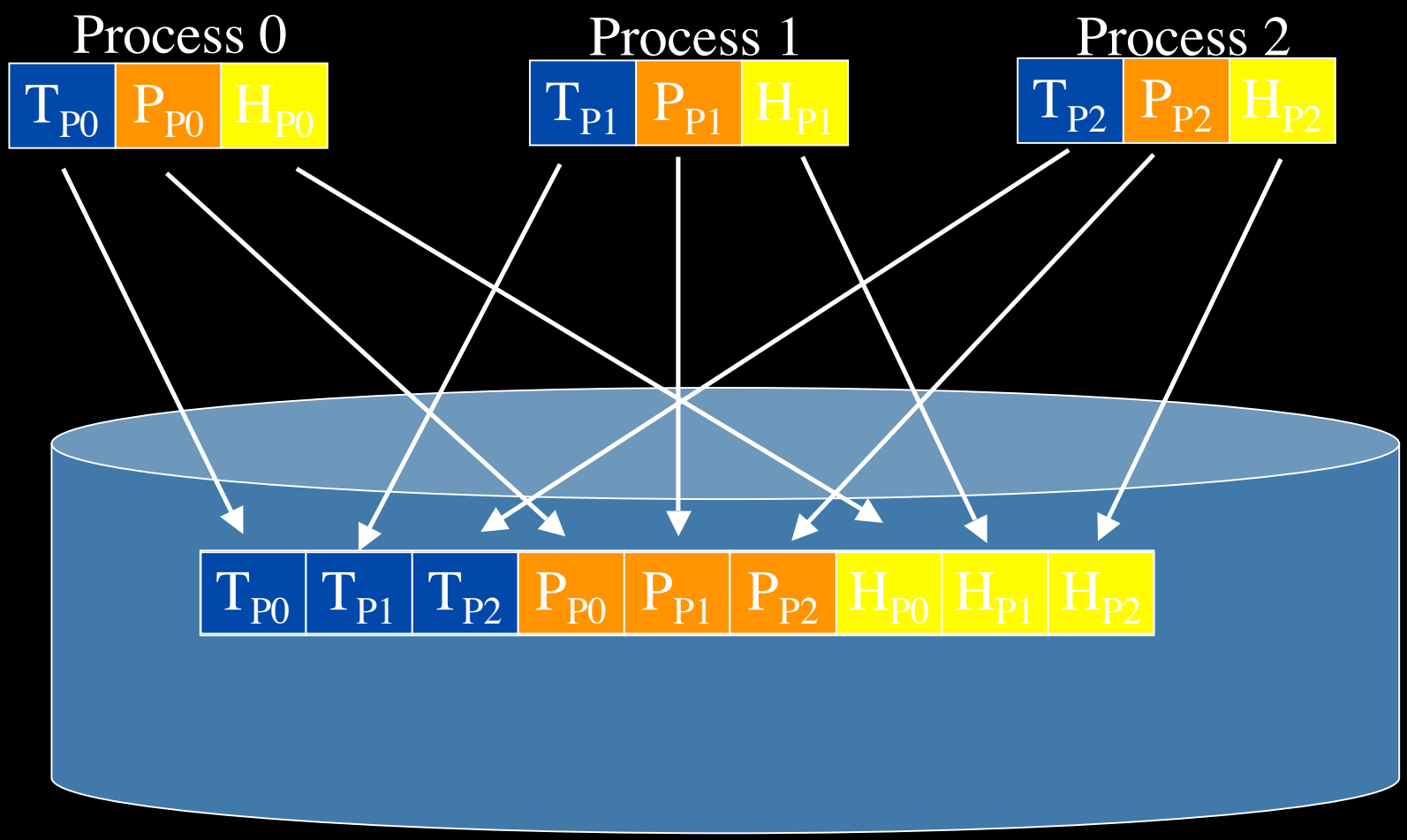
N-to-1 non-strided example



N-to-1 non-strided evaluation

- Advantages
  - Each process has its own non-shared region of file
  - False sharing possible only at borders between processes
  - Larger I/O's reduces frequency of read-modify-write's on RAID5
- Disadvantages
  - N-to-M restart: Complex and costly just like N-N
  - Visualization is similarly problematic

N-to-1 strided example



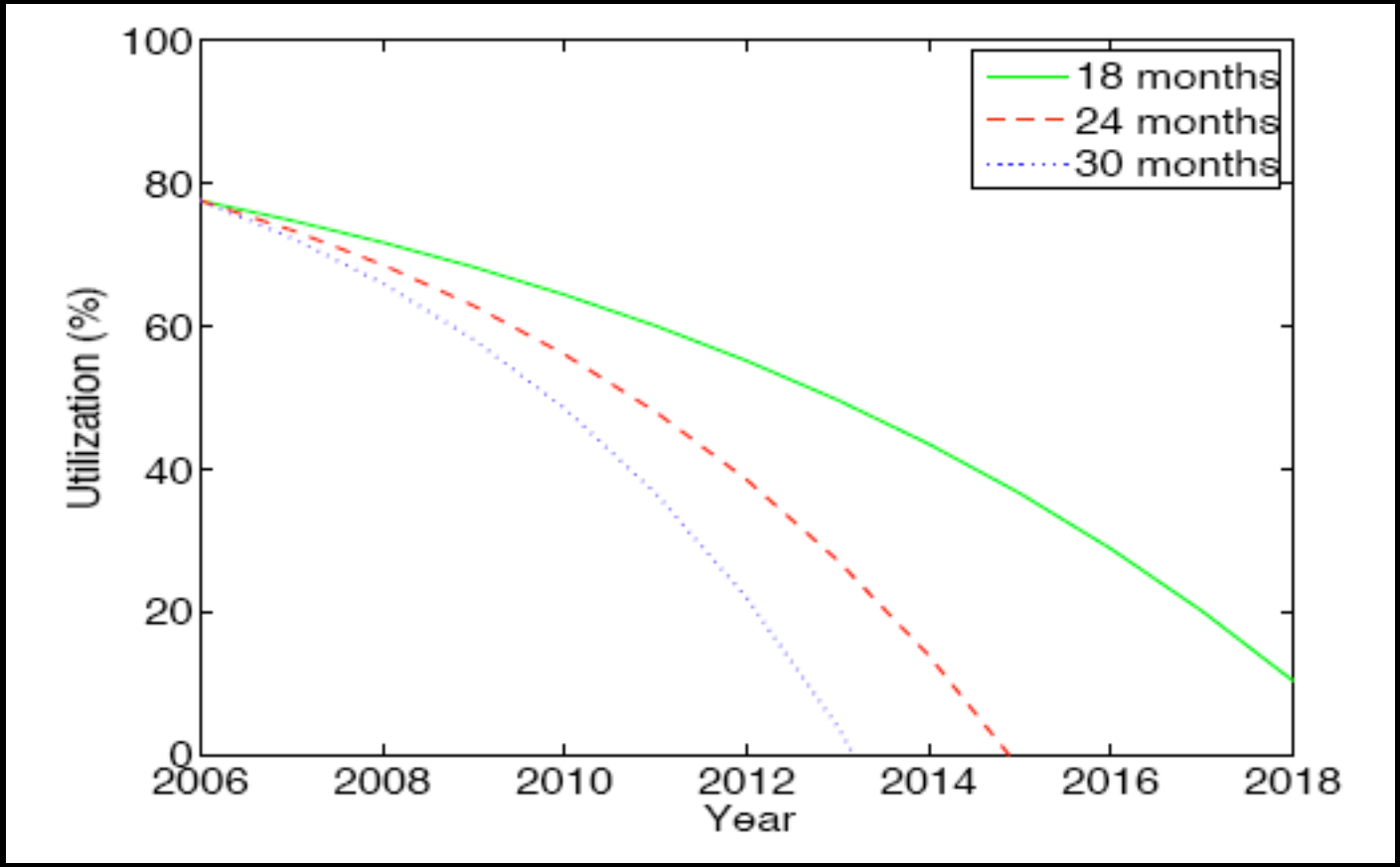
N-to-1 strided evaluation

- Advantages
  - Simplest book-keeping for N-to-M restart
    - Read each element contiguously, resplit for M
  - Simplest formatting for visualization
    - Visualization typically only interested in small number of variables, can read each contiguously
- Disadvantages
  - Small, possibly unaligned writes
  - False sharing
  - Read-modify-write's serialize parallel I/O in our Panasas RAID-5 storage system

Applications seem to want to use N to 1 strided for convenience, for big writes this is not an issue but small writes are problematic

Note: N-1 strided is best for applications, worst for storage system.

Challenges: Mid-term



Bianca Schroeder, Garth Gibson. "Understanding failure in petascale computers."

Questions?

johnbent@lanl.gov

Resources

- HEC FSIO planning site
  - http://institute.lanl.gov/hec-fsio/
- ISSDM site
  - http://institute.lanl.gov/isti/issdm
- PDSI site
  - http://institute.lanl.gov/pdsi

The future holds more capability!

	ZIA	TRINITY
Peak PF	> 2	> 50
Total memory	> 0.5 PB	> 5 PB
Aggregate[(a)] Memory BW	> 1 PB/sec	> 5 PB/sec
Aggregate Interconnect BW	> 1 PB/sec	> 7 PB/sec
Aggregate Bisection BW[(b)]	> 80 TB/sec	> 450 TB/sec
Aggregate Message Rate	> 10 GMsgs/sec	> 80 GMsgs/sec
Aggregate I/O BW	> 1 TB/sec	> 10 TB/sec
Disk Capacity	> 20 PB	> 200 PB
System Power (MW)	5 - 8	10 - 16
Floor Space (sq ft)	< 8,000	< 8,000
MTTI (Job) / MTBF (System) (Both @ Full Scale)	> 50 / > 200 Hrs.	> 50 / > 200 Hrs.

Workflow

- Checkpoint dominates, is vital for MTTI, but is batch
- Restart speed is important to deal with MTTI, but again is batch
- Parallel Data Analysis is mostly interactive and mostly read, often not totally serial
- Archiving – 1 checkpoint archive every 10s to 100s of dumps (dial in the pain), in long runs this is important, and is batch
- Archive of analysis and code is more interactive
- Growing online needs – code, libraries, etc.
- Teraflop workstations may be used for Analysis but people cant have 100 disks in their offices so pNFS may end up helping with that

- Workflow is assisted a LOT by
  - Global Parallel File System (all systems see the data in a scalable way)
  - Global Parallel Archive (all systems see the data in a scalable way)

- Overall workflow is not well understood and we have more “micro-benchmarks” for portions of the workflow. We need to consider more “macro-benchmarks” to model the entire flow.

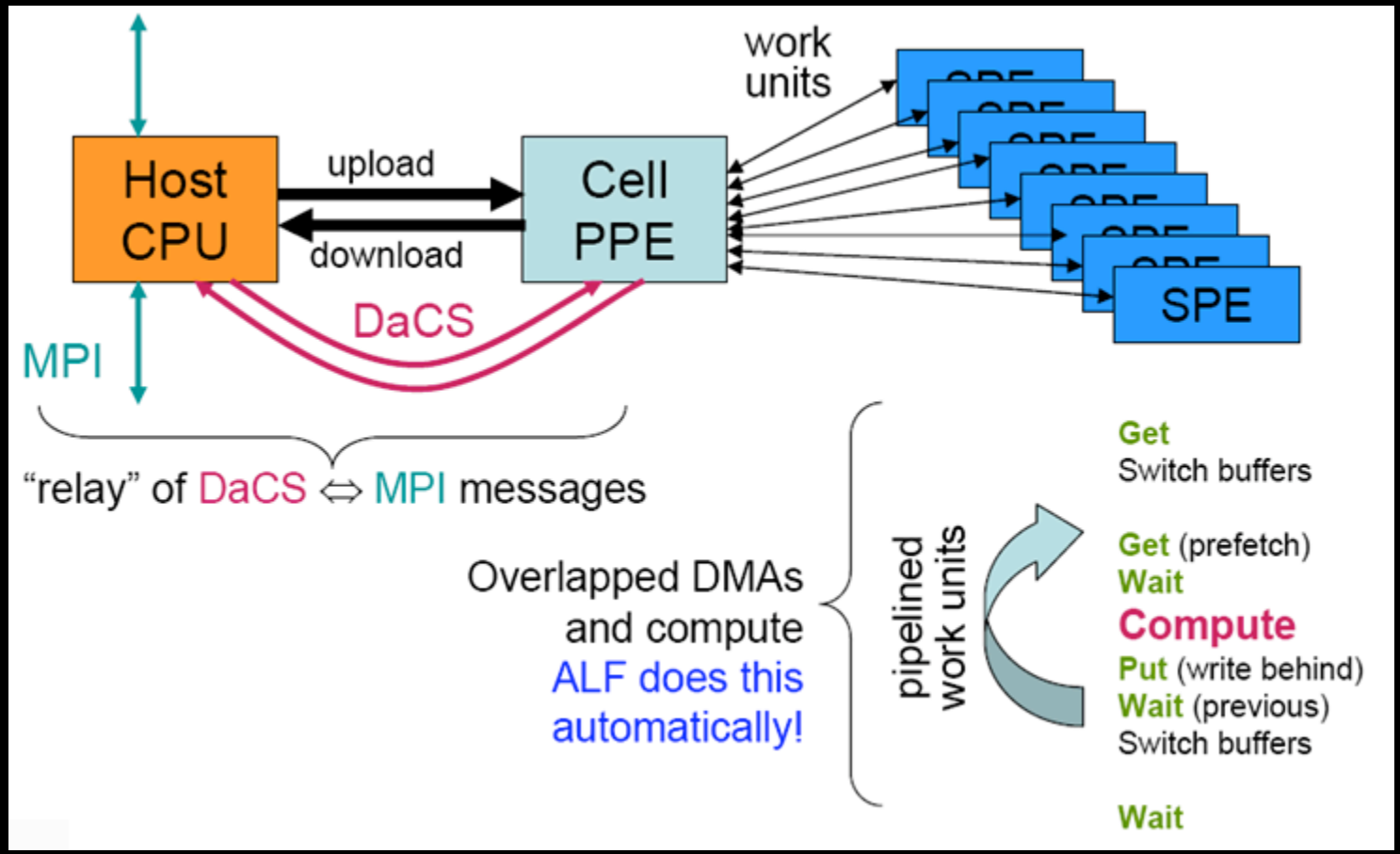
New Device Exploitation

- Flash use
  - Poor $ for BW
  - Write wear
  - Excellent for small unaligned
  - Looking at metadata use and possibly in an IO forwarding environment
  - Looking at removing batteries for NV of the future
- PCM
  - Lower write wear
  - Capacity/speed trade off in the same device

Future Deployment

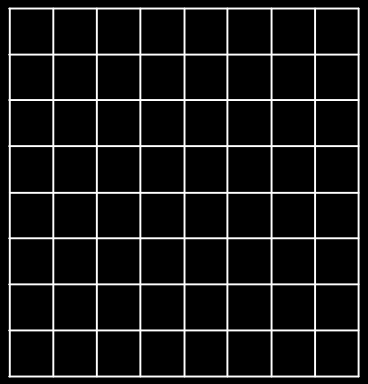
- Likely to go to extensible IB Torus for SAN in future
- If we go IB will need IB subnetting and fail over mechanisms that rival Ether/IP
- Likely to push for ISER server side control over Ether/IB
- If we go with IB need to gateway back to Ether
- Likely to investigate Archives that would fit into the future scalable SAN architecture

Programming Model

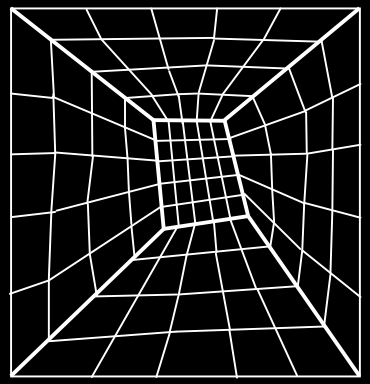


3D Meshes

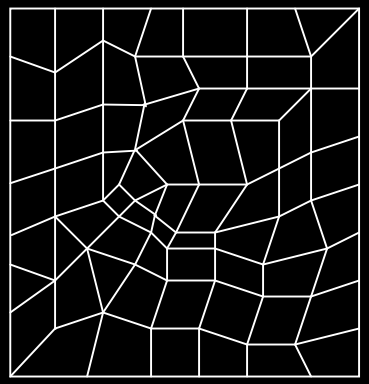
- Spatial meshes in 1-D, 2-D, & 3-D for finite- difference/finite-element numerics



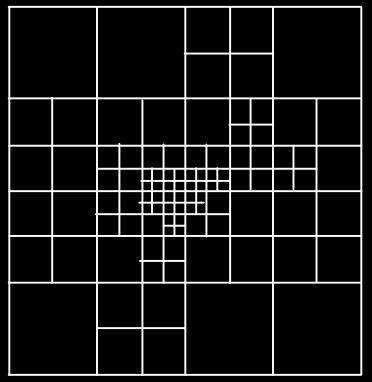
Brick mesh (Cartesian, cylindrical, spherical)



Structured & Block Structured



Unstructured



Continuous AMR Cartesian mesh (cell-by-cell & cycle-by-cycle)

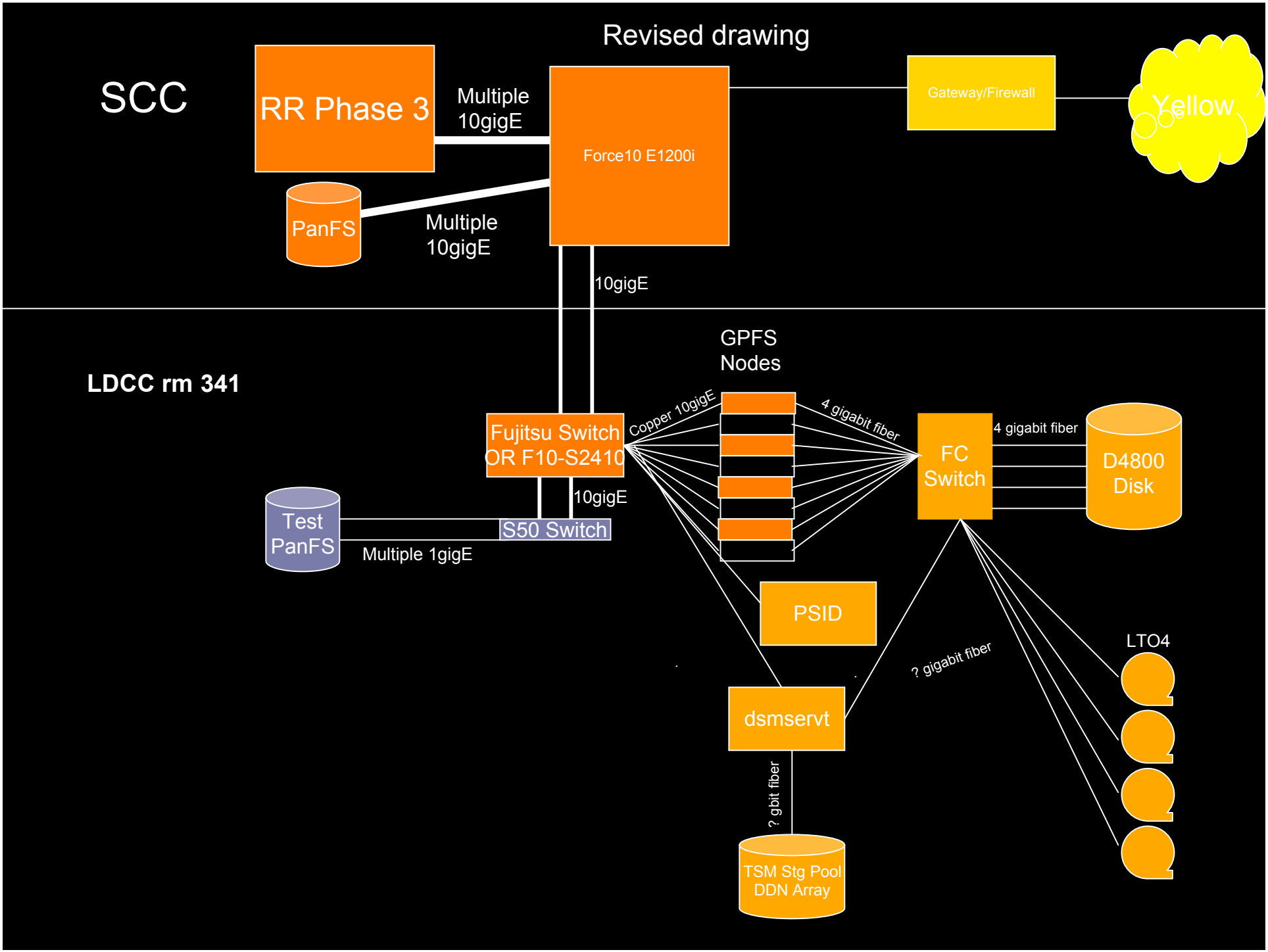
Run on highly parallel supercomputers

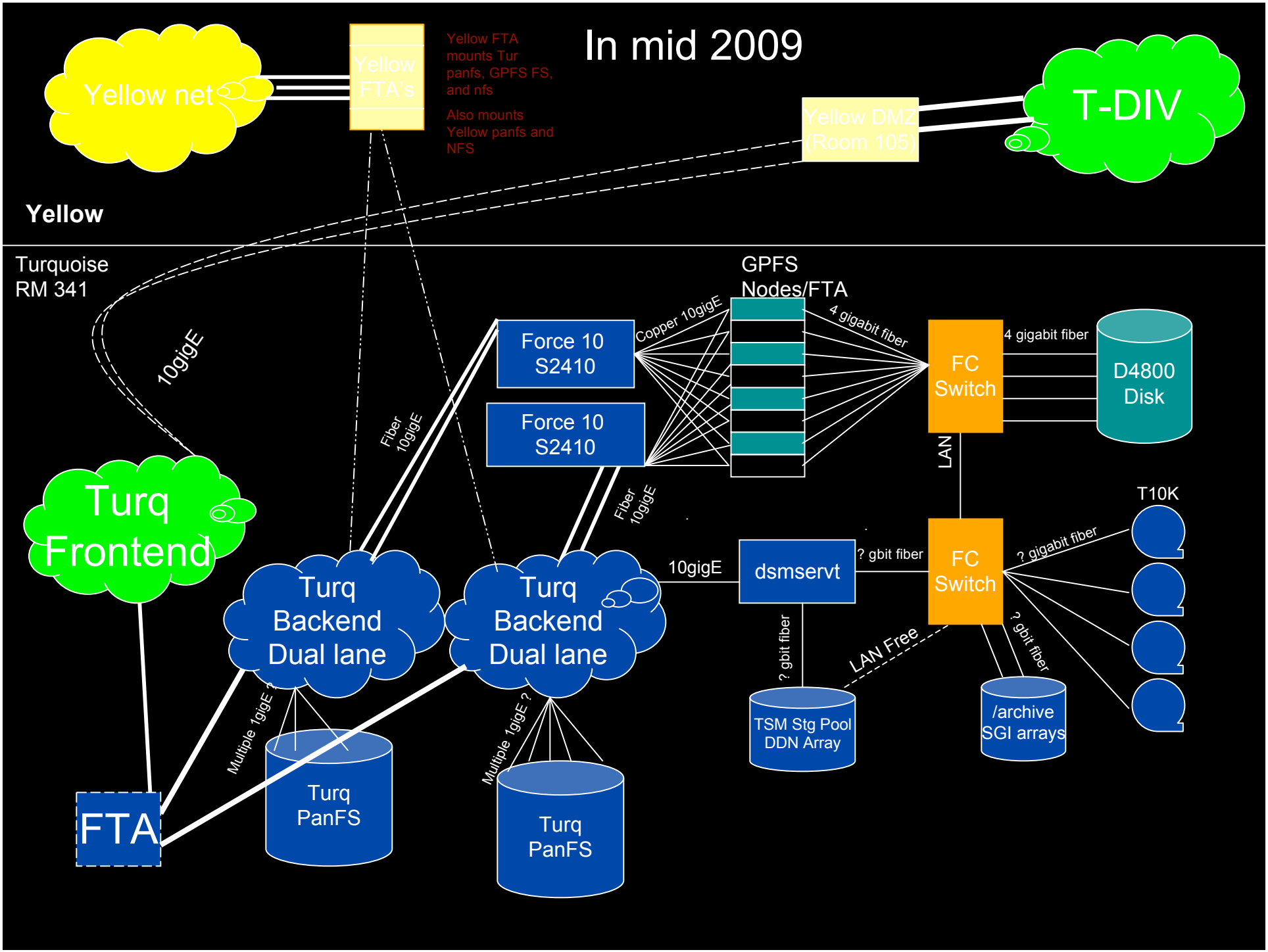
What to checkpoint?

- AMR is split into subdomains
  - Each subdomain contains one or more “elements”
  - Each element of each subdomain has characteristics
    - Temperature, pressure, humidity, etc
- Assign every subdomain to a single process
- Checkpoint: Each process writes all characteristics for all elements within its subdomain
  - Checkpoint files can be very large (order terabytes)
- Restart
  - Read all state, reconstruct mesh, redivide into subdomains, reassign
  - Can be on different processors or even different numbers of processors (this is common)

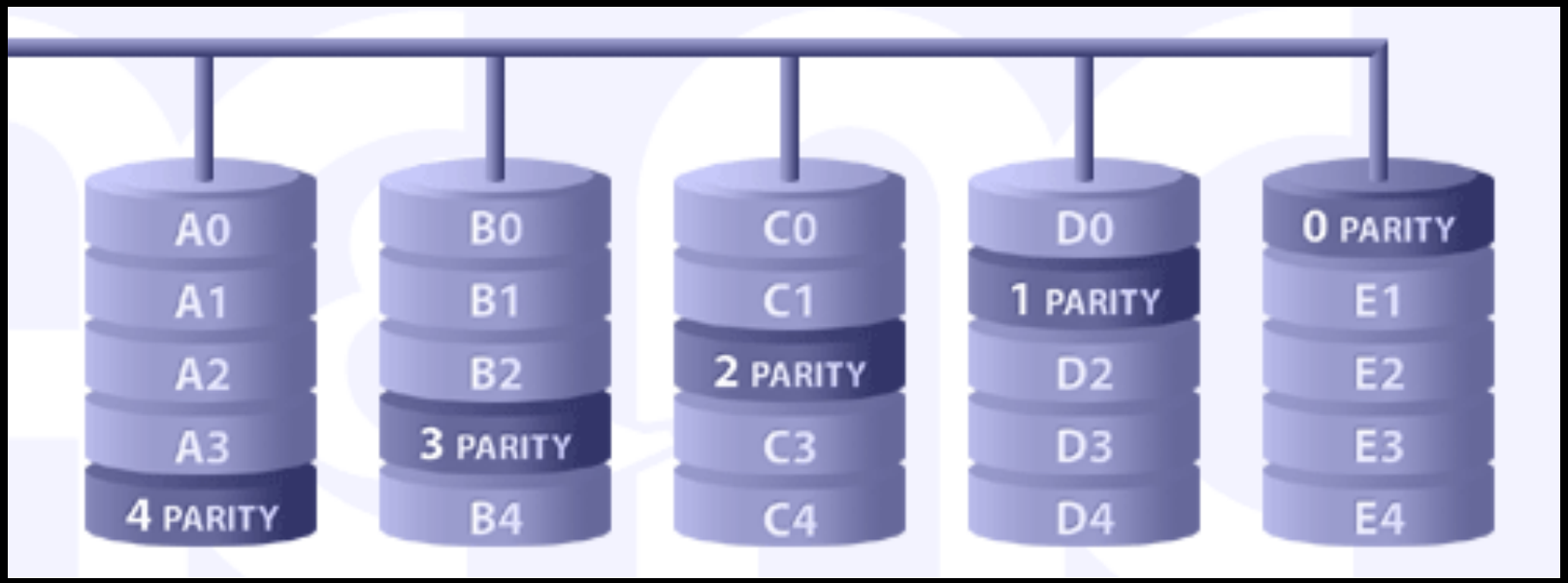
How to checkpoint?

- N-to-N: Each process writes / reads to own file
- N-to-1: Each process writes / reads to single shared file
  - Data can be either non-strided or strided


Revised drawing
SCC
RR Phase 3
Multiple
10gigE
Force10 E1200i
Gateway/Firewall
Yellow
PanFS
Multiple
10gigE
10gigE
LDCC rm 341
GPFS
Nodes
Fujitsu Switch
OR F10-S2410
Copper 10gigE
4 gigabit fiber
FC
Switch
4 gigabit fiber
D4800
Disk
10gigE
Test
PanFS
S50 Switch
Multiple 1gigE
PSID
LTO4
? gigabit fiber
dsmservt
? gbit fiber
TSM Stg Pool
DDN Array


In mid 2009
Yellow net
Yellow FTA's
Yellow FTA mounts Tur panfs, GPFS FS, and nfs
Also mounts Yellow panfs and NFS
Yellow DMZ (Room 105)
T-DIV
Yellow
Turquoise RM 341
10gigE
Fiber 10gigE
Fiber 10gigE
Turq Frontend
Turq Backend Dual lane
Turq Backend Dual lane
FTA
Multiple 1gigE ?
Multiple 1gigE ?
Turq PanFS
Turq PanFS
Force 10 S2410
Force 10 S2410
Copper 10gigE
GPFS Nodes/FTA
4 gigabit fiber
FC Switch
4 gigabit fiber
D4800 Disk
LAN
10gigE
dsmservt
? gbit fiber
FC Switch
T10K
? gigabit fiber
? gbit fiber
? gbit fiber
LAN Free
TSM Stg Pool DDN Array
/archive SGI arrays

RAID 5

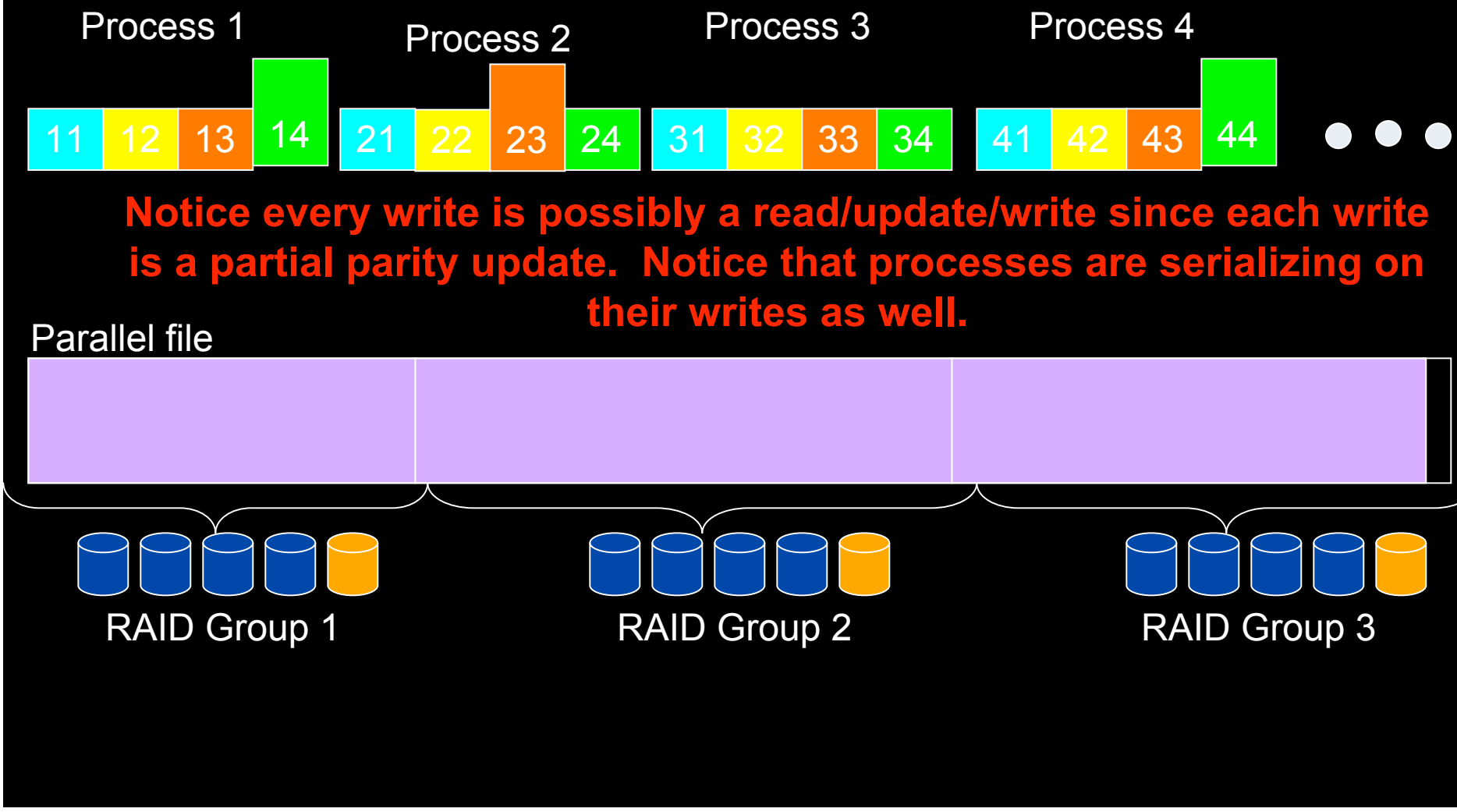


With disk blocksizes getting bigger you must aggregate more and more data to avoid read-modify-write

Panasas read-modify-write

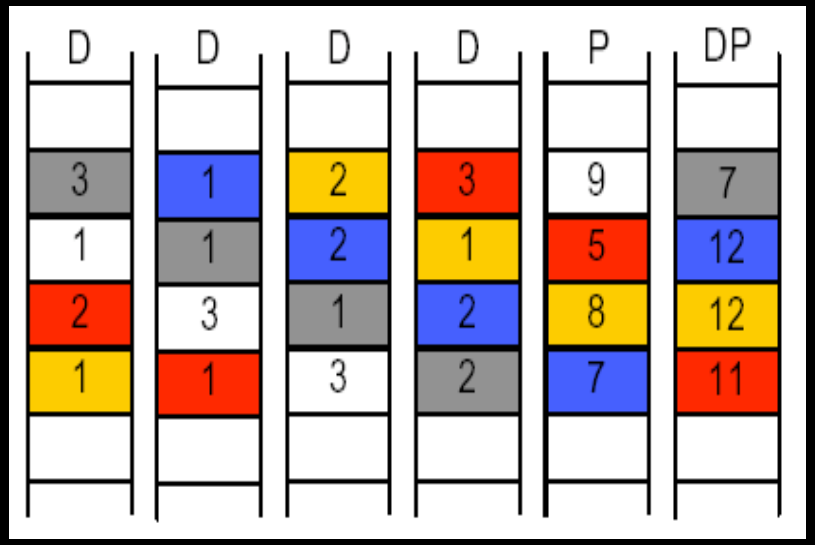
- Panasas clients compute parity
  - Advantages
    - Scalability
    - Almost end-to-end reliability easy
  - Disadvantages
    - Clients must read-modify-write for small writes
    - Multiple clients writing to same RAID stripe serialized (i.e. N-1 strided)

Potential storage implications of N-to-1 strided



RAID 6 (Plus 2) makes it worse

- **Normal XOR parity is calculated straight across the disk blocks**
- **Diagonal parity is calculated on diagonals, there are other methods based on polynomials**
- **You need to have way more data around to do efficient parity calculation**
- **This means you have to aggregate even more data to get efficient writes**



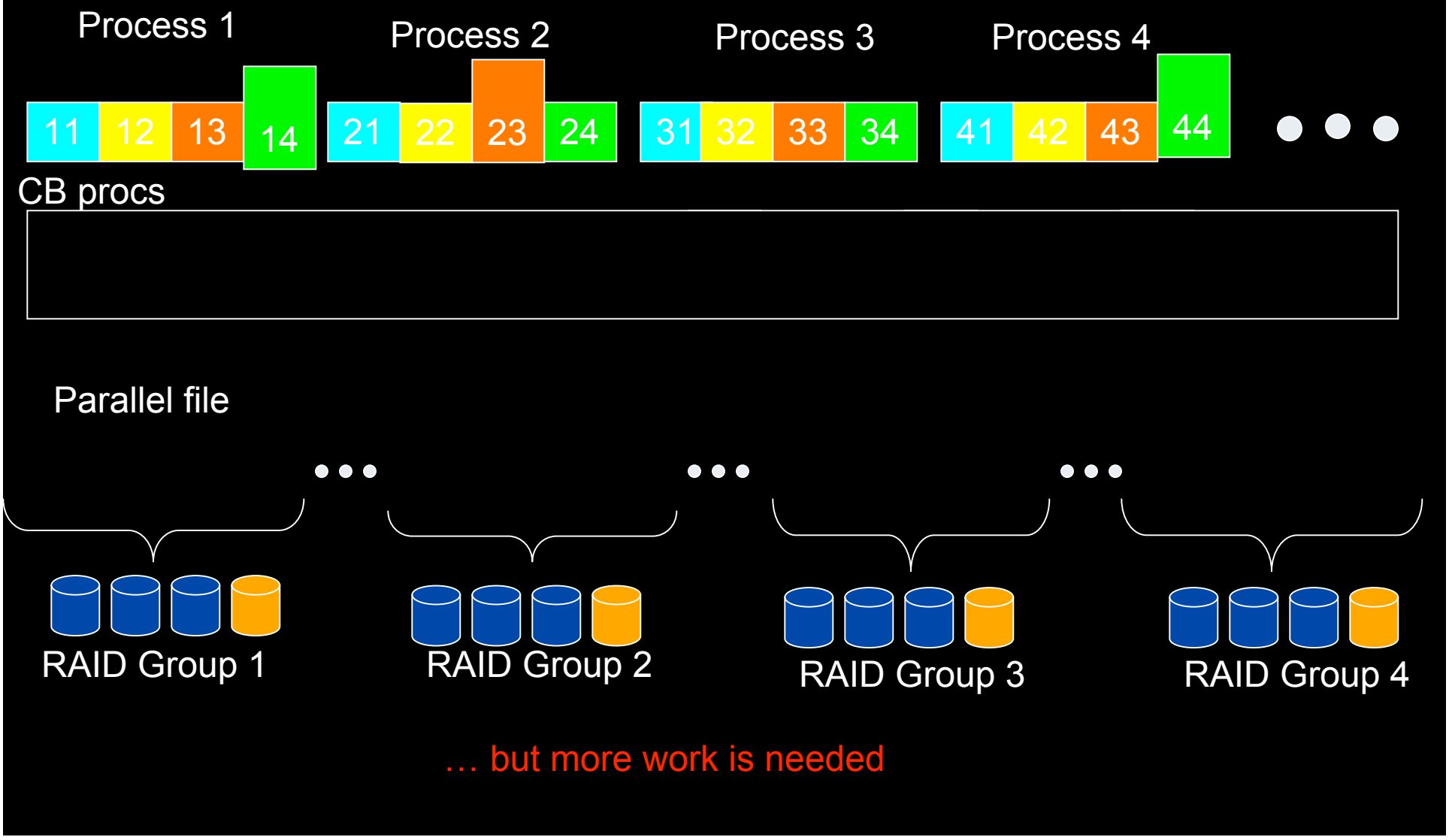
Approaches being worked on for N to 1 small strided

- Middleware aggregation/asyncronism
  - Aggregation and async (Middleware)
    - Current MPI-IO two phased collective buffering
    - Persistent File Domains
    - Dache
- Layout
    - Raid10
- Combination of Layout and asyncronism
    - Raid10 plus async aggregation

Middleware: Work on aggregation/asynchronism for MPI-IO

- Current form aggregates from many nodes to few, but writes occur synchronously
- Persistent File Domain (PFD), same as above, but aggregation/file writes aligned on offset into file
- Dache, adds async writes to the aggregation

Middleware can help



… but more work is needed

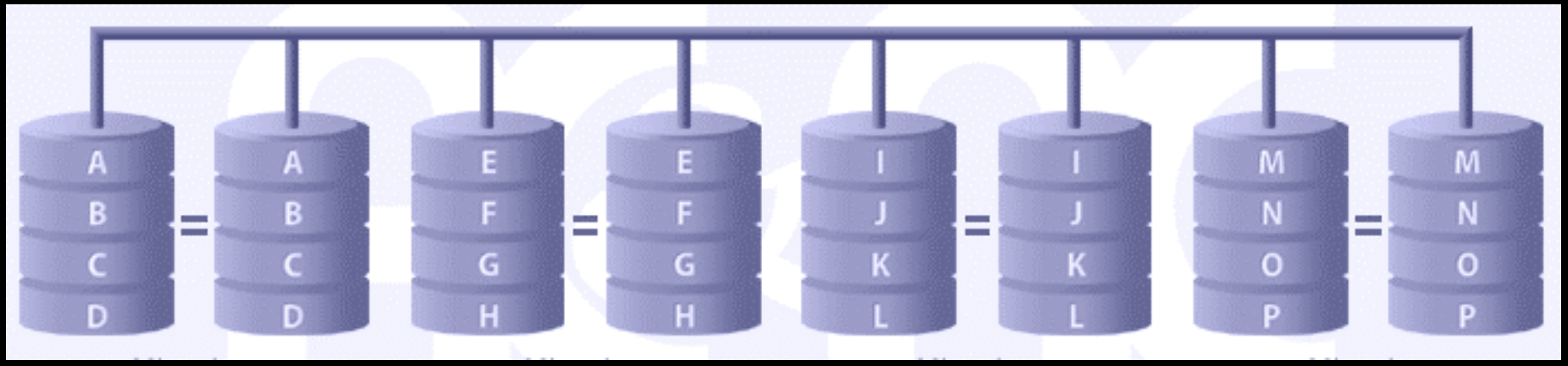
Middleware: Not Complete Solution

a) aggregating/caching to a few nodes helps get bigger writes, but is fundamentally limiting bandwidth to a few nodes writing to the file system
b) aggregating/caching to lots of nodes ends up being an N-Squared problem (because all the compute nodes have to send to all the aggregators at the outer limit)

Middleware can help, but it is NOT the total solution to this problem!

Layout: RAID 10

- If read/update/write and the optimal write sizes for RAID 5 or 6 is at the heart of the problem, don't use raid 5,6
- RAID10 has no read update write, just send your write object to two disks



Layout: Not Complete Solution

a) RAID10 addresses the BW on N to 1 small strided fundamentally!

BUT

b) the need for middleware is not erased though because you will still need async effects to utilize pipes effectively

The solution for now is the combination of middleware and layout!

Last Summer's Student Research

- 1) Parallel I/O Trace
- 2) Multi-dimensional File Systems
- 3) Parallel Multi-dimensional Search with Google Desktop

Andy Konwinski, Berkeley

Milo Polte, CMU

This summer

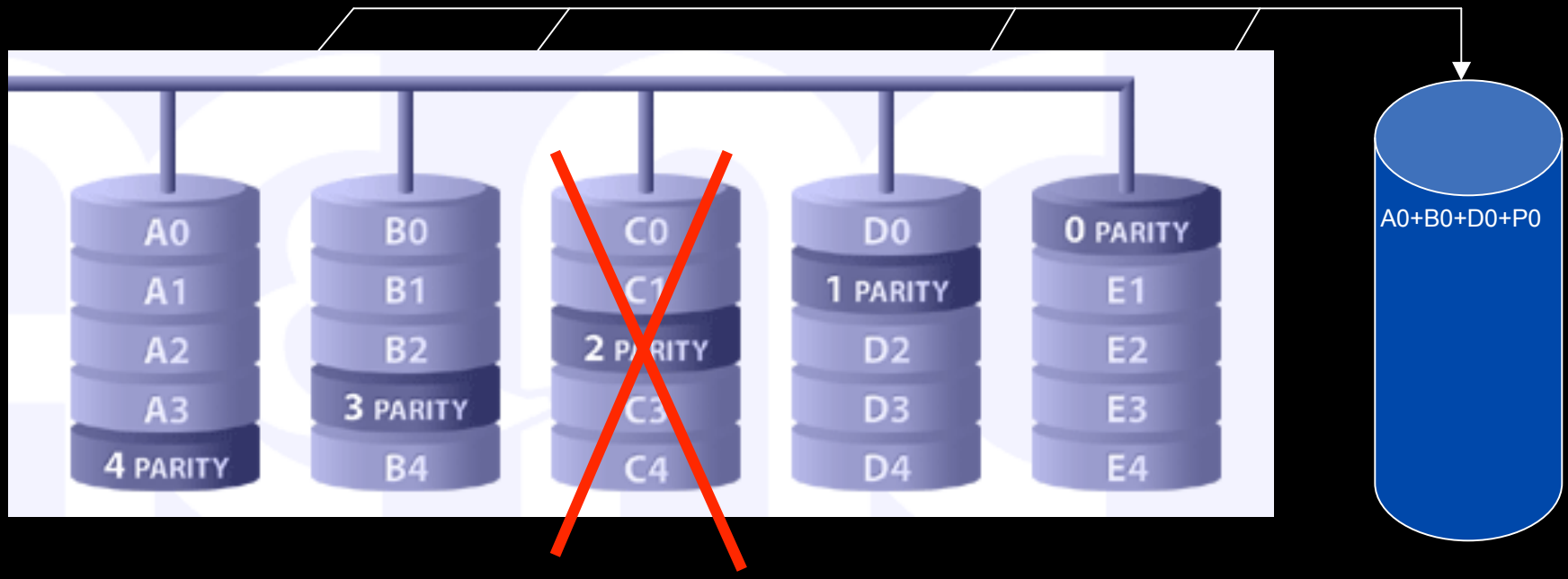
- More tracing
- More parallel search
- Ringbuffer project
- Parallel execution frameworks survey

Panasas Object Storage

- Block storage
  - Disk doesn't understand data semantics
  - File system responsible for organizing data into metadata blocks and data blocks
- Object storage
  - Disks understands data semantics
  - Disks are given "objects" (typically files) and disks themselves maintain metadata, etc.
- Advantages
  - Control (e.g. use RAID-10 for N-1 strided, 5 for N-N)
  - Rebuild times following failures

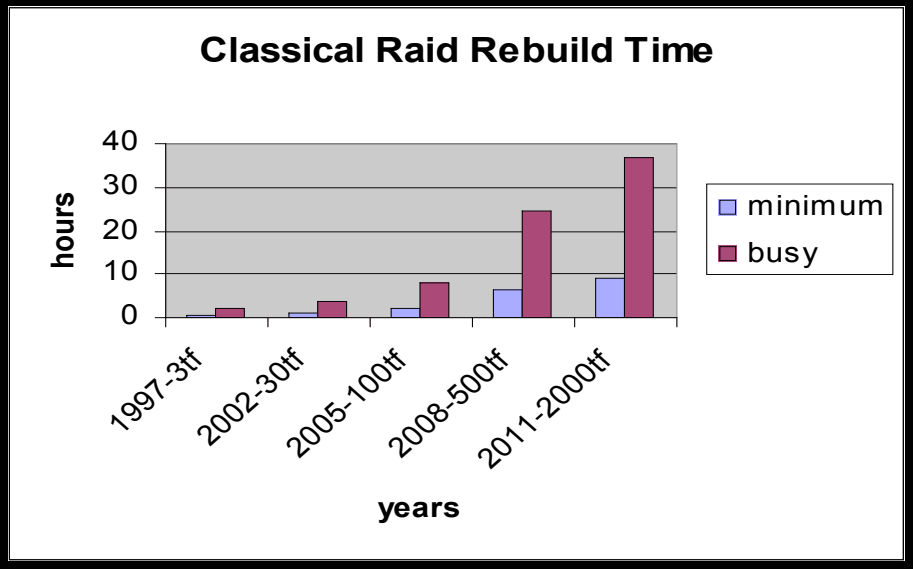
Classical RAID-5 Rebuild

- **Read the remaining disks, XOR, and write the result.**
- **Speed ultimately governed by write speed of target new disk or read of N disks**
- **This is similar for RAID-6 as well**



Classical RAID Rebuild Time

- Density increasing faster than bandwidth
- Rebuild taking longer and longer
- Increased probability of unrecoverable bit error during rebuild



Panasas Rebuild

- RAID groups different for every object
- Rebuilding a failed disk uses different set of disks for each object on that disk
- Only live blocks are rebuilt
- Rebuild objects can be placed anywhere
- Rebuild reads using all N disks, writes using all N also
- UBER only hurts one object, not full disk

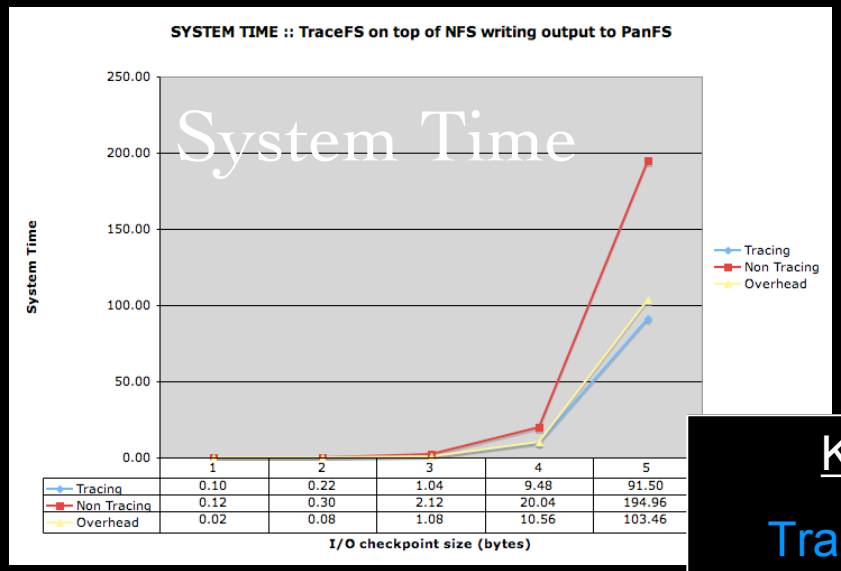
1) Tracing project

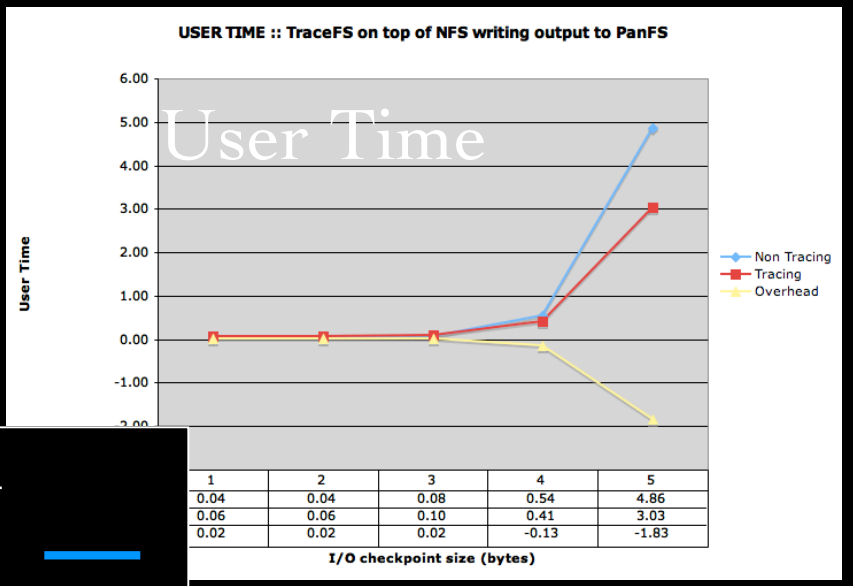
- Comparing LANL-trace, //trace, FS-Trace
- Evaluating for parallel apps
  - Runtime overhead
  - Replay accuracy

Andy Konwinski, Berkeley

- Motivation
  - Enable others to work on our problems
    - Study I/O patterns to improve middleware
    - Replay traces to improve storage systems
  - Understand patterns of our apps to tune storage/app

Overhead



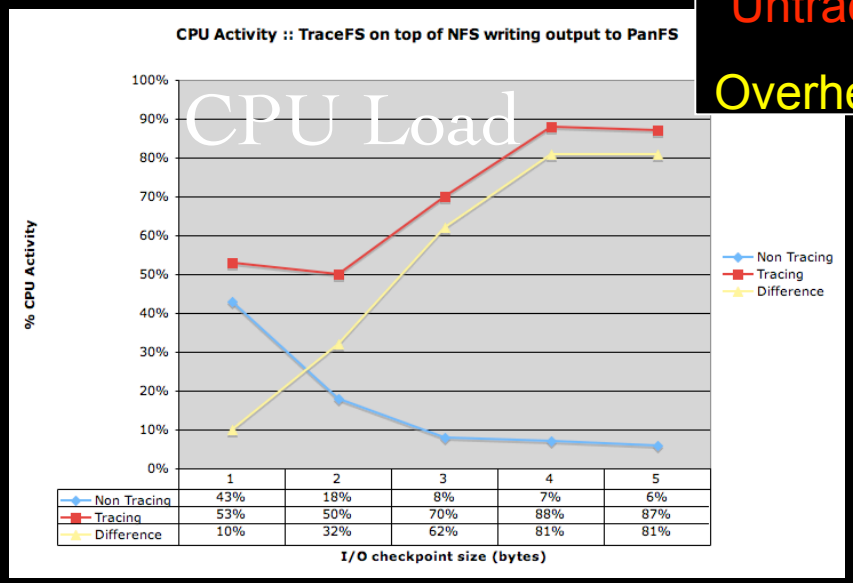


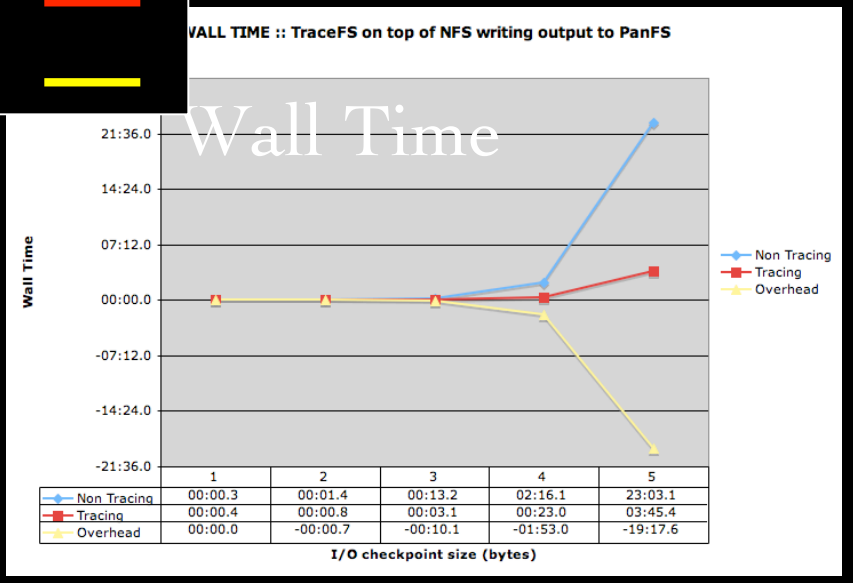
Key:

Traced

Untraced

Overhead





Tracing Conclusions

- Measuring overhead is hard
  - Many variables
  - Lots of ways to skew results
- Overall TraceFS has low overhead
- Timing overhead may not be as important as replay accuracy.

- Published at PDSI workshop (SC07)
  - http://www.pdsi-scidac.org/SC07

2) Multi-Dimensional File System

Milo Polte, CMU

- Create additional indices
  - Users
  - Perms
  - Access times
- Allow search such as
  - Find all pictures taken with cannon camera during date range
  - Find all data with certain humidity and pressure characteristics
  - Find all satellite images of Mexico City on cloudy days
  - Find all old, not recently accessed, large files

Why do we need a multidimensional filesystem?

- Our ability to capture and store data is outpacing our ability to organize and analyze it
  - Data Volumes are doubling each year
  - Scientific instruments are gaining greater precision
  - Automation is creating vast stores of data
- Traditional filesystems index files along a single dimension: That of the filename and path
  - Filenames are frequently irrelevant; analysis needs to be applied to all data with a certain set of attributes not a certain name pattern
- A multidimensional filesystem is one which indexes files based also on their meta-data tags
  - Gives a more expressive way to describe and find files

Design of our system

- Designed to be a prototype of a larger real time knowledge capture system
- Built on top of the Parallel Virtual File System (PVFS) distributed filesystem
  - From Argonne National Labs
  - Open source
  - Used in production systems
- Integrates an sqlite3 SQL database on each of PVFS's meta-data servers
  - Sqlite3 databases used to index and query metadata
  - Embedded solution - low total cost of ownership
  - Indexes all 'normal' metadata (POSIX attributes, file sizes, etc.) stored at the MDS
  - Also allows application-specific metadata to be added for any file indexed by the MDS

Sample Queries

- Queries use an SQL style syntax. Expressiveness limited only by application metadata tags.
- Scientific:
  - All satellite image files taken from a particular telescope and marked by an intelligent program as having a probability > 70% of being a Nebula.
  - All NMR results taken on the folding of a certain protein since Tuesday.

- Administrative:
  - Space saving: Show me the five largest files in the system that haven't been accessed in a month or more.
  - Security: Show me all system files whose content hash doesn't match a list of correct values.

3) Parallel Search w/ MPI and Google Desktop

- Summer class at Colorado School of Mines
- Split large JPEG collection over multiple computers
- Create Google Desktop extension for JPEG's
- Create parallel, striped indices
- Use MPI to parallelize Google Desktop
- Do multi-dimensional file system like searches