OPENFABRICS ALLIANCE

Exascale File Systems

Scalability in ClusterStor's Colibri System

Peter Braam

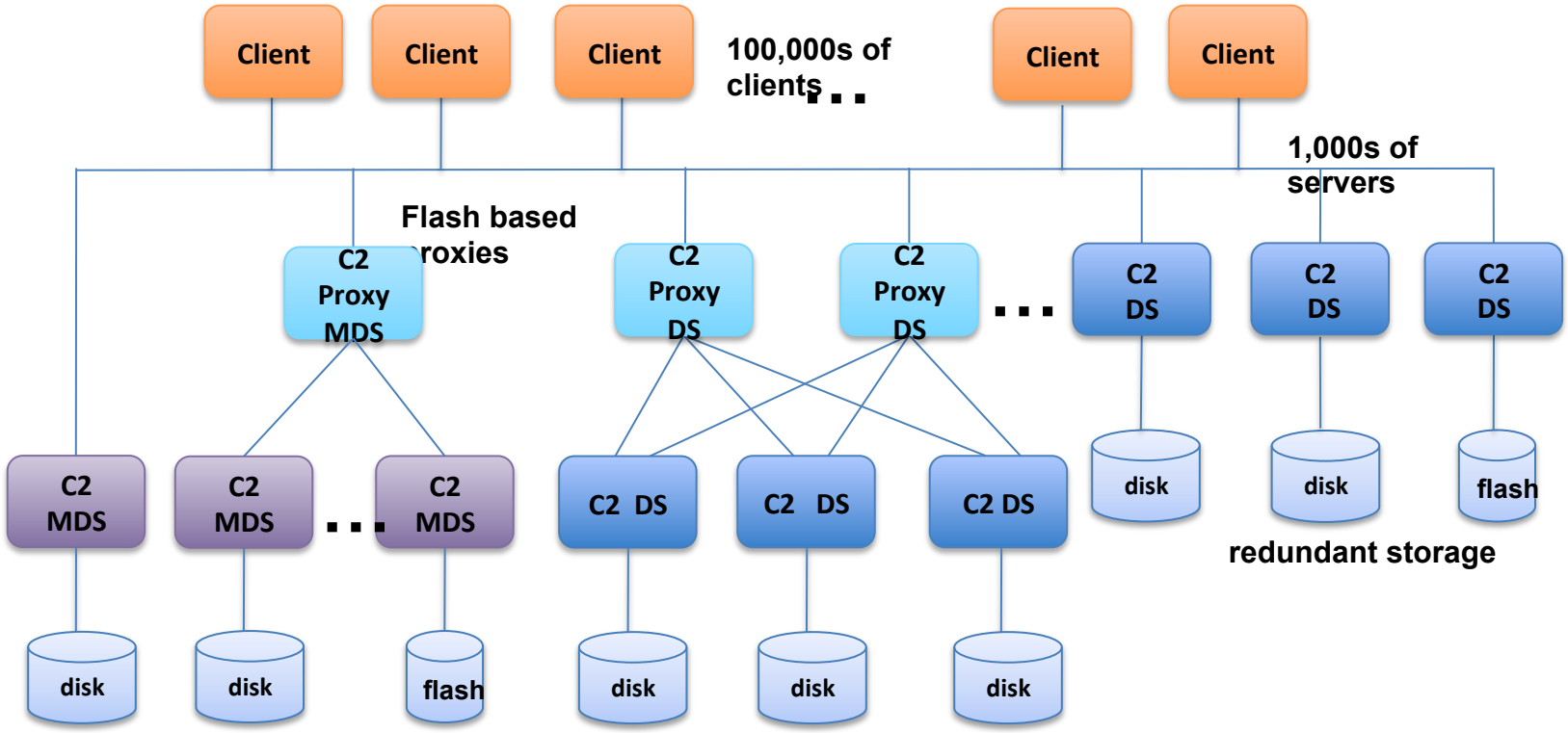
Mar 15, 2010

Content

- System overview
- Management
- Availability
- I/O
- Scalable communications

System overview

Colibri (aka C2) deployment



3rd party storage: RAID arrays, JBOD,
Internal storage, with or without flash

- Complete file service solution
- Clients have "cluster intelligence"
- Scales enormously (HPCS and beyond)

A few elements to start

- Object store is used for everything
- Metadata is in embedded databases
  - Schema breaks away from UFS (finally)
- PCI flash storage: integral design element
- Architecture should scale to 100 PF range
  - After that I see too many disks again….

Management

Management

- What was learned in the past?
- Observing a cluster is crucial
  - FOL: file operations log
  - ADDB: analysis and diagnostics database
  - A data mining & visualization tool
- Simulations
  - Simple language expressing I/O patterns
  - Simulator that processes the FOL traces
  - Editor for FOL for simulations

ADDB & FOL

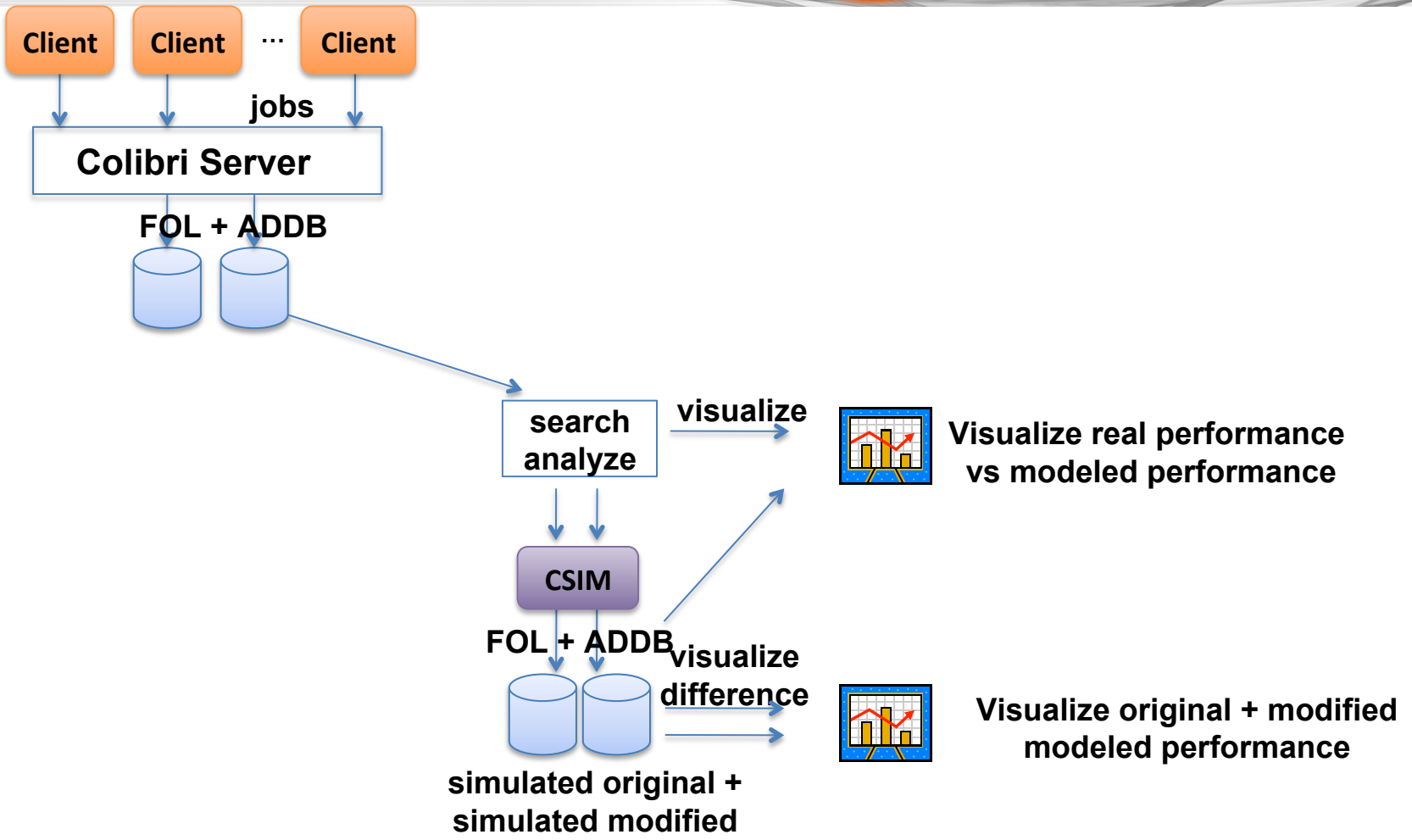
FOL

- File Operation Log
- Contains FOP
  - packets
- Transactional
- Complete
- Useful for data mgmt

ADDB

- For each FOL record
  - One ADDB record
- Contains
  - Analysis & diagnostics data
- Eg:
  - Which client did what?
  - How long did it take
  - What network?
  - Cache hits / disk queues

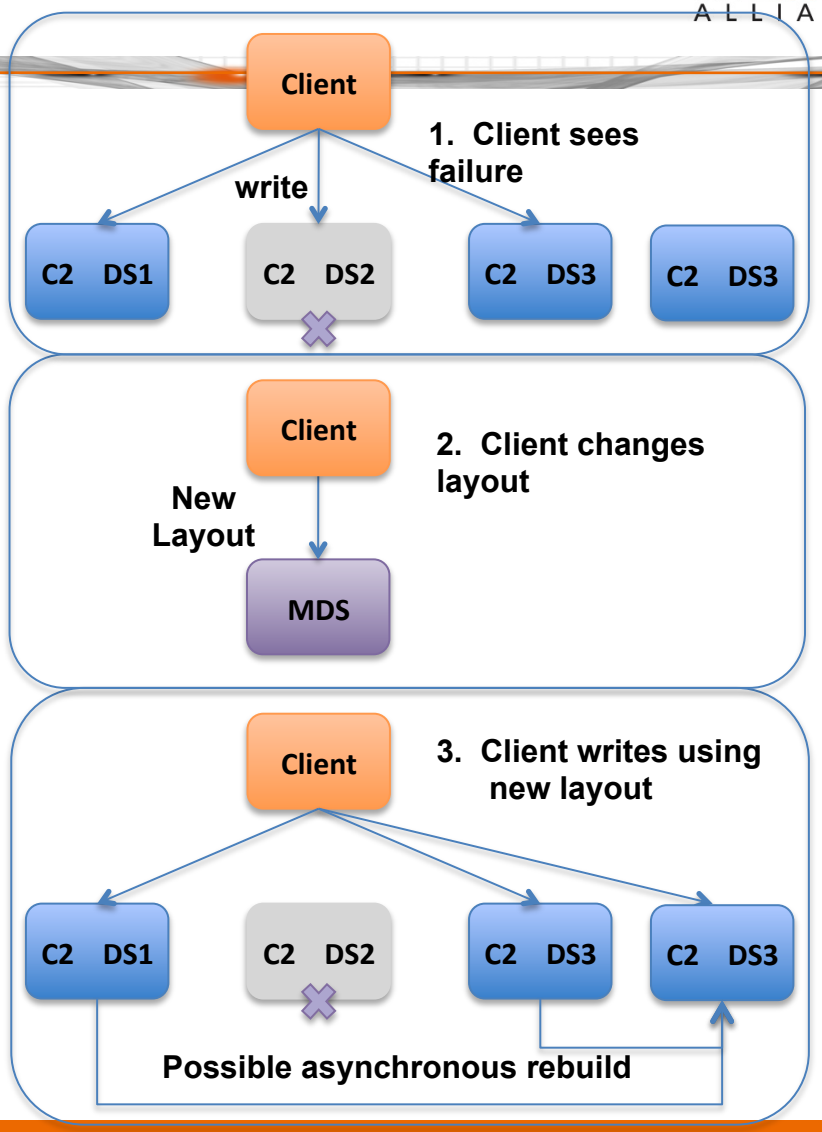
Example - Simulation Environment



Failures & availability

No failover

- Failures will be common
  - in very large systems
- Failover
  - Wait for resource recovers
  - Doesn't work well
- Focus on availability
  - No reply (failure, load)
  - Adapt layout
  - Asynchronous cleanup



Metadata - Data Layout

- Almost all layouts will be regular
  - Use formulas, do not enumerate objects & devices
  - Use references between metadata tables
    - Avoid multiple copies

- After failures, data layout becomes complex
  - Failures can move 1000s of different extents in a file
  - May clean this up asynchronously
    - FOL knows about it

Data placement

- Data layout is central in architecture
  - Redundancy
    - In data servers
    - Across the servers in network
- Parity de-clustered in network and in box
  - All drives:
    - Contain some data of object
    - Contain some spare space
  - 1-5 minute restoration of redundancy for failed drive
    - with bandwidth of cluster

Availability

- System @ full bandwidth 99.97% of the time
  - Disk rebuilds are not the problem
  - Server losses are a problem
- Systems must have network redundancy
  - AKA server network striping
- No purpose for shared storage

Caches and scale

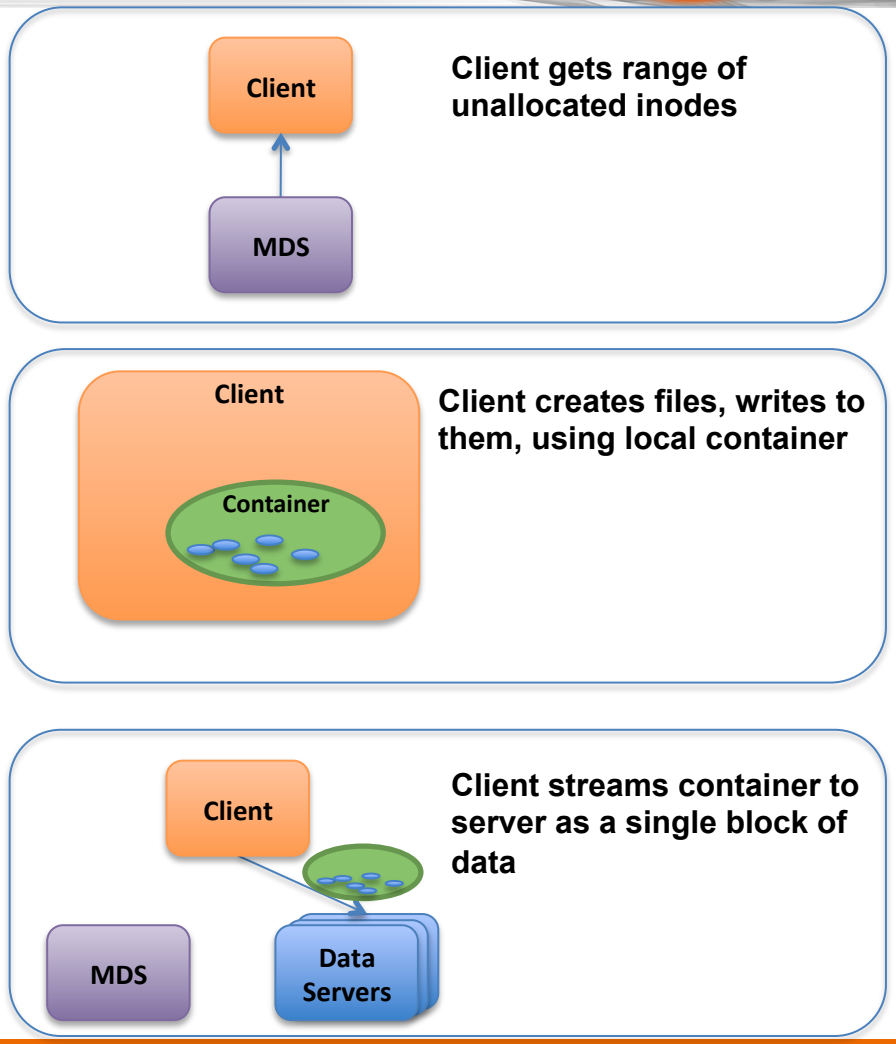
Flash, cache & scale

- {Flash,Disk} based servers = {FBS, DBS}
- 2010
  - Bandwidth: FBS $/GB/sec == 0.25 DBS $/GB/sec
  - Capacity: FBS $/GB == 5x DBS $/GB
  - FBS form factors –> Fusion IO becoming commodity
- Choose
  - FBS for bandwidth, capacity ~ RAM of cluster
    - FBS bandwidth high enough for e.g. fast checkpoint
  - DBS Bandwidth = 0.2 FBS Bandwidth

Containers – forwarding caches

- Containers are like a logical volume
- Can contain
  - Metadata
  - File fragments
  - Files
- One new feature – “include” operation
  - a container moves into a larger container
  - “include” does ***not iterate over contents***
- Metadata more complex
  - Not problematic, we use an embedded DB anyway

Container streaming



Several uses for containers

- WB caches in clients
  - Advantage – no iteration over contents
- Networked caches to act as I/O capacitor
  - Get flash for bandwidth,
    - Capacity ~ RAM of entire cluster
  - Get disk for capacity
- Intermediary storage, e.g. I/O forwarding
  - Physically non-contiguous container with small stuff
  - Streams to contiguous container inside a bigger one
- Data management

Oh, you need to read?

- Physics – disks are slow
- Two common cases:
  1. Everyone reads the same
  2. Everyone reads something different

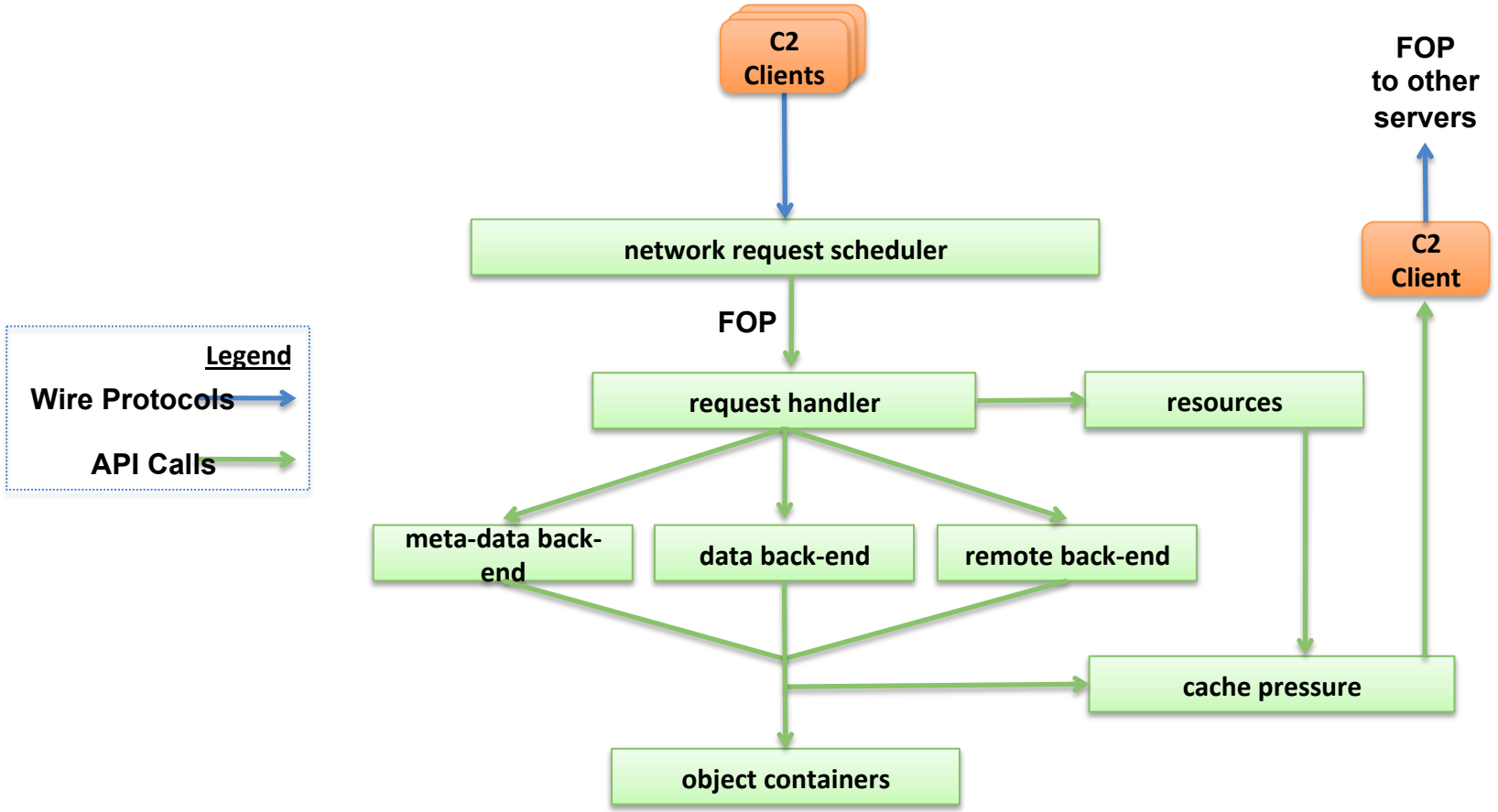
- Case 2 is best handled as follows:
  - Use FOL or otherwise track what files will be needed
  - Pre-stage them in flash
- Case 1 is addressed with scalable comms

Scalable communications

Colibri resources

- Colibri manages resources
  - Locks, file layouts, grants, quota, credits ….
  - Even inodes and data are resources
- Resource communications need to scale
  - E.g. a million processes may need to agree on an adapted file layout
  - All nodes need to read the same data
- Hierarchical flood-fill mechanism (not new)

Architecture decomposition

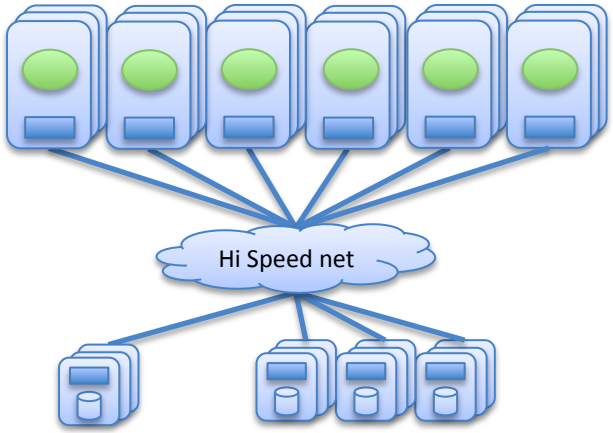


Colibri subsystems and data flow

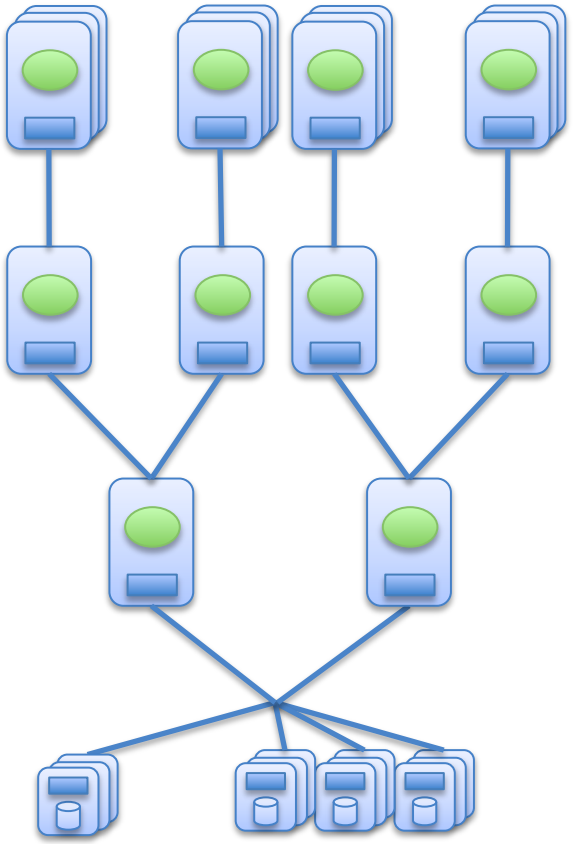
Colibri architecture whitepaper v2 Figure 5

Nov 29, 2009

Scalable communications



Physical Organization of C2 Cluster



Logical Organization for Resource Management

A few other notes…

- ClusterStor market focus is “large data”
- Colibri will be delivered in increments
- The 4th increment is the exascale file system
- The 2nd is a pNFS server
- The 1st, 3rd and 5th are secret ☺

Thank you!