Are SE Architectures Ready For LHC?

Andrew Hanushevsky

Stanford Linear Accelerator Center
Stanford University
3-November-08
ACAT Workshop

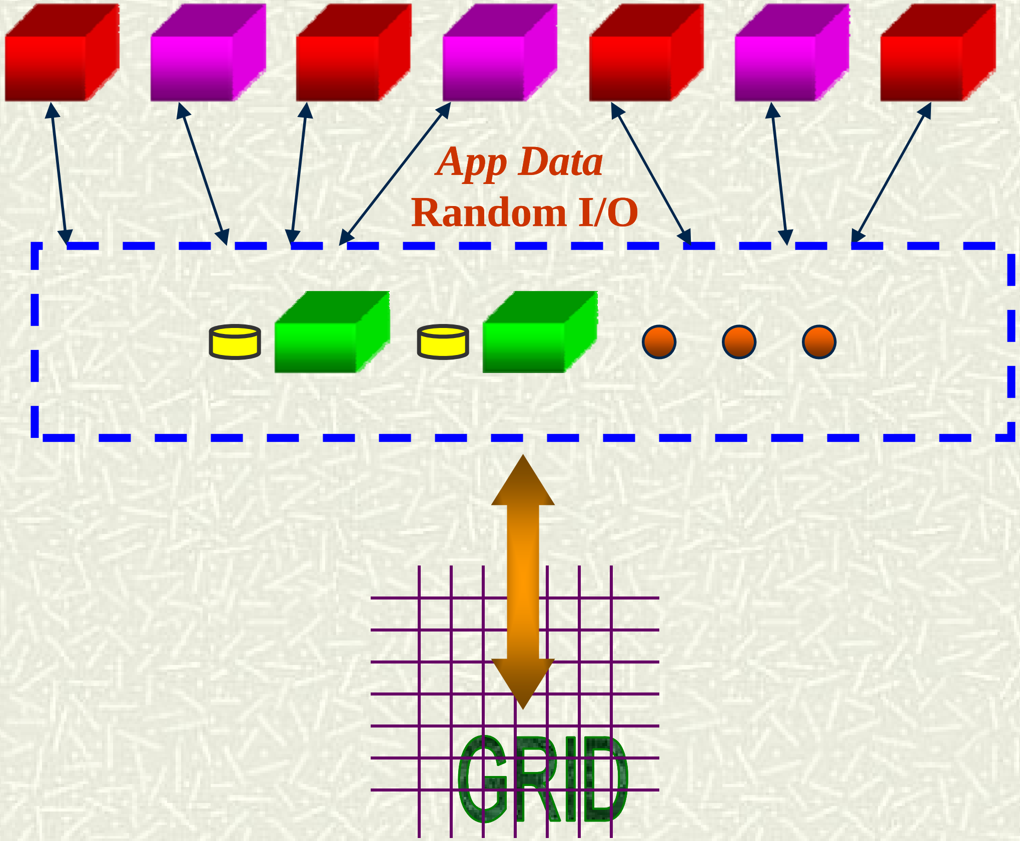
Outline

- Storage Element (SE) Aspects
  - The classic first stab
- Constructing an SE Today
  - GPFS, Lustre, DPM, dCache, Castor, Scalla
    - Considerations
- The SE Spectrum
  - Relevance
  - Practicalities
- Conclusion

Storage Element Aspects

- Provide data to the compute farm (CE)
- Interface with the grid
  - Data in and data out
- Integrate with other systems
  - Offline storage (i.e., MSS)
- All in the context of a “typical” Site
  - Even though many times there is no such thing

General SE Architecture



App Data
Random I/O

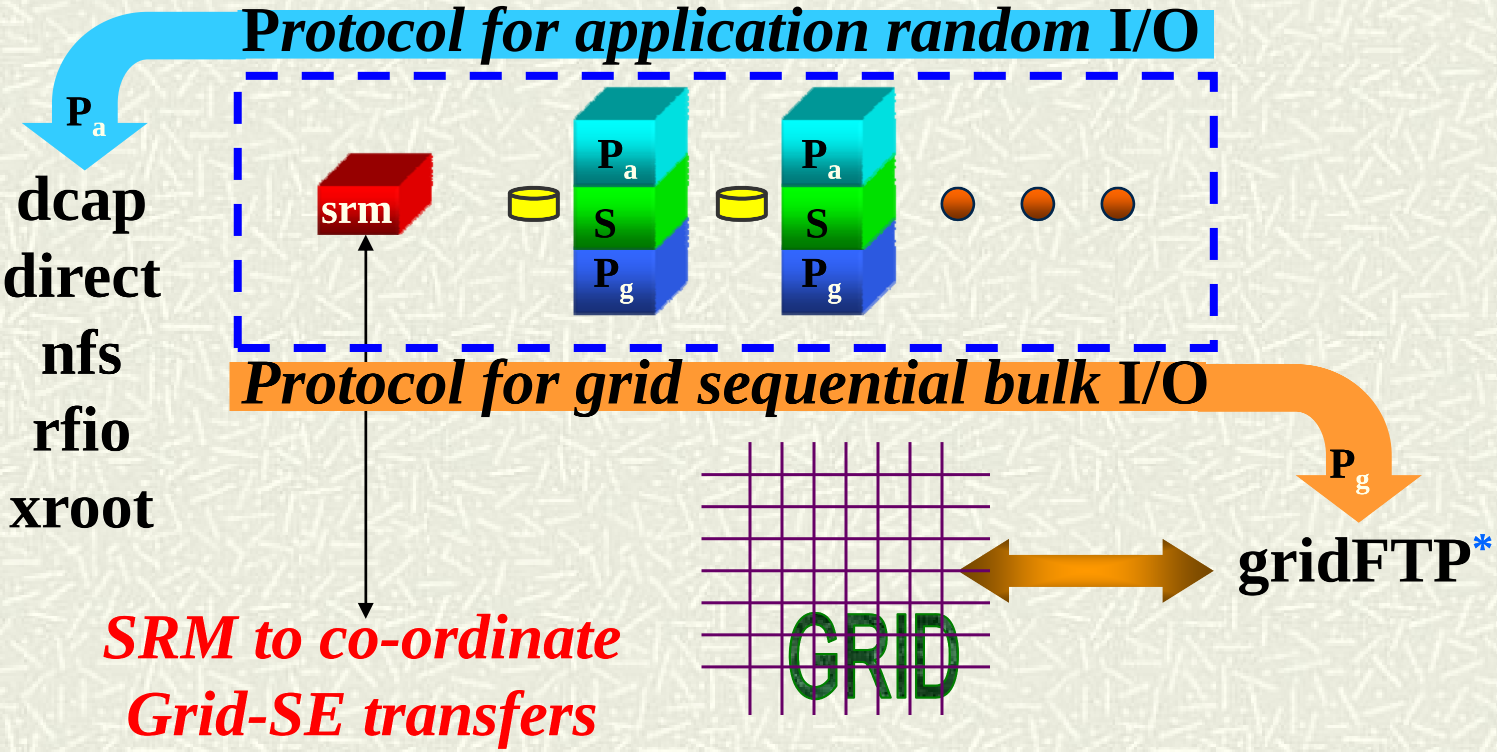
Bulk Data
Sequential I/O

Compute Farm
a.k.a CE

Storage System
a.k.a. SE

Grid Access

Closer Look At The SE



***Protocol for application random* I/O**

P_a

dcap
direct
nfs
rfio
xroot

***Protocol for grid sequential bulk* I/O**

P_g

gridFTP*

SRM to co-ordinate
Grid-SE transfers

***Other non-grid protocols: bbcp, bbftp, scp, xrdcp**

Typical SE Wish List

- Easy to install, configure, and maintain
  - Minimal configuration changes as hardware changes
  - Tolerant of OS upgrades
    - Now more common due to security patches and shortened lifecycles
    - Largely interwoven with pre-requisite software
      - The more pre-requisites the more complicated this becomes
- Easy to debug
  - Integrated monitoring and extensive understandable logging
- Easy to scale upwards as well as downwards
  - Low machine requirements and resources
    - More machines mean more administrative effort

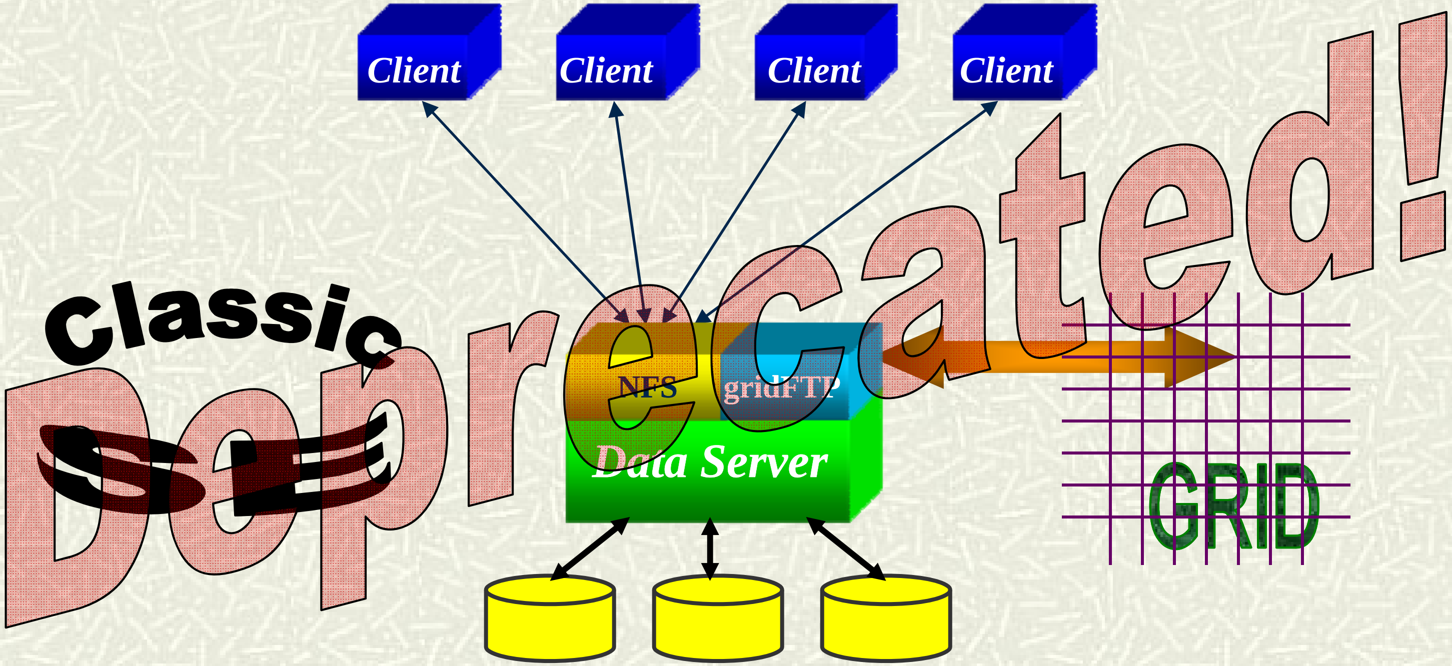
SE Scaling

- Two dimensions of scaling
    - I/O Scaling **(hard limit is network bandwidth)**
        - Adding hardware increases capacity & performance
    - Client Scaling
        - Adding number of simultaneous clients increases through-put
- Would like this to be as close to linear as possible
    - May not be in administrative effort or performance
        - Usually the biggest differentiator in SE architectures

Requirements Reality Check

- I will qualitatively touch upon
  - Entry cost
    - How much hardware is needed at the get-go
  - Software dependencies (***not exhaustive***)
    - Major factor in administrative effort & maintainability
      - Will payroll costs dominate?
  - I/O and client scaling
    - Will it perform efficiently at LHC levels?
  - DataAccess
    - What’s used for analysis and bulk (grid) access

The First Stab



Intoxicatingly Simple!

Classic SE Considerations

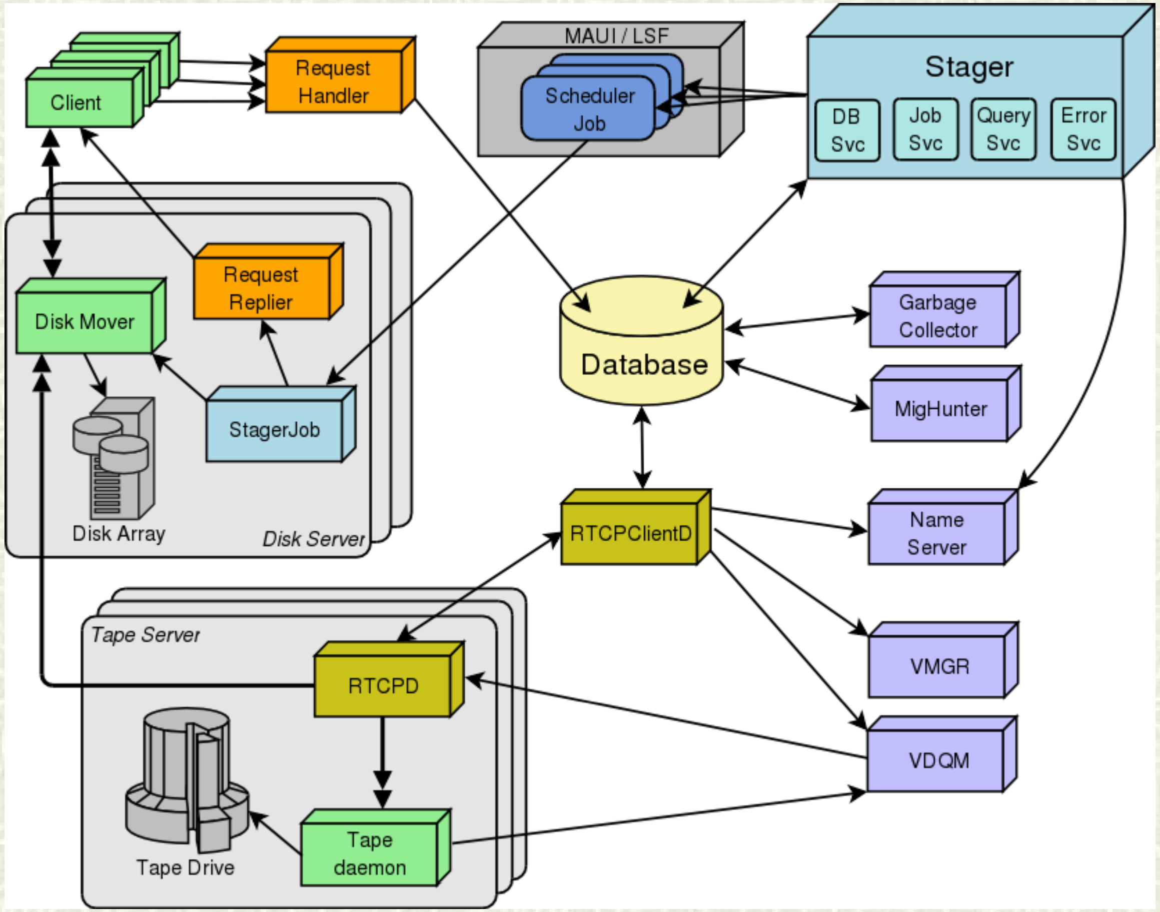
- Entry Cost
  - Very low, almost any hardware that can support the client load
  - Configuration and maintenance is easy
- Software Dependencies
  - None, other than for the SRM (which can be a lot)
- Scaling
  - I/O scaling and client node scaling low (limited applicability)
- Data Access
  - Commonly, NFS
- Other Considerations
  - MSS integration a local matter
  - No database requirements
- Why deprecated?
  - Because there was no SRM at that time
    - World has changed with the appearance of usable BestMan and StoRM

Currently Popular SE Architectures

- Castor
- dCache
- DPM
- GPFS
- Lustre
- Scalla (i.e., xrootd)

The following slides are based on my observations

Castor

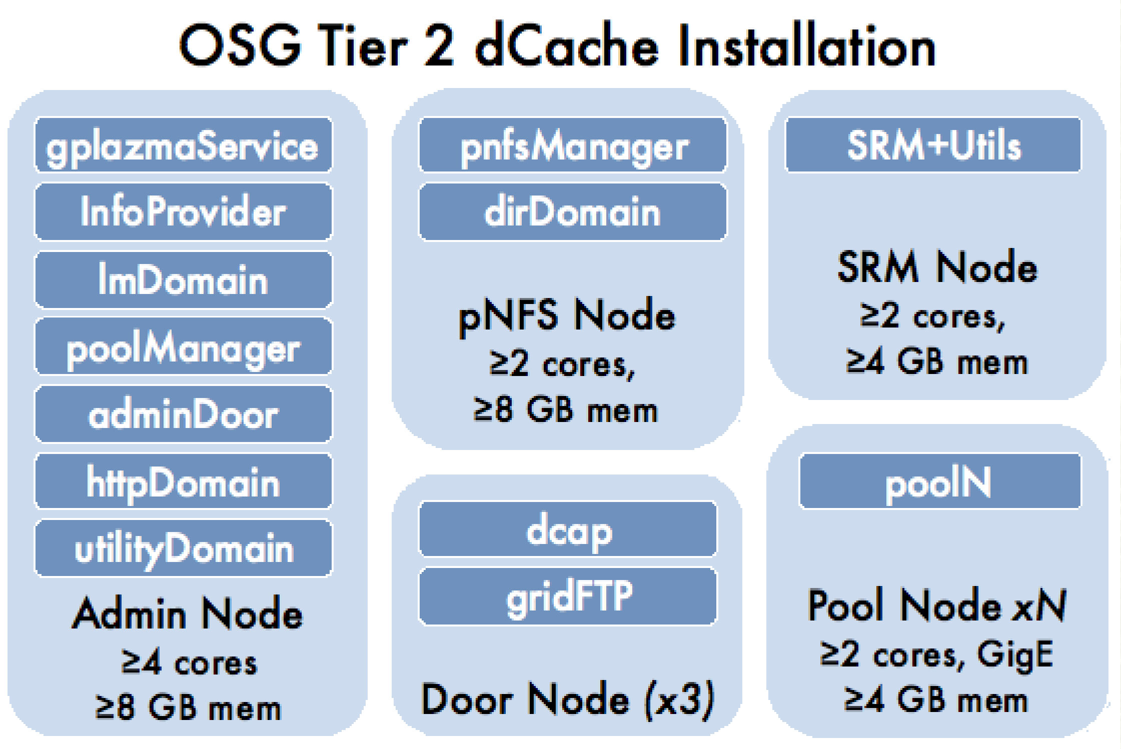


http://castor.web.cern.ch/castor/images/Castor_architecture.png

Castor SE Considerations

- Entry Cost
  - 2-4 Nodes + n Disk Server Nodes + m Tape Server Nodes
  - Configuration is relatively hard
- Software Dependencies
  - Oracle, LSF or Maui, many co-requisite Castor based RPMs
- Scaling
  - I/O scaling average, client node scaling may be below average
- Data Access
  - rfio, integrated xroot, built-in SRM
- Other Considerations
  - MSS integration included
  - Requires database backup/restore plan

dCache

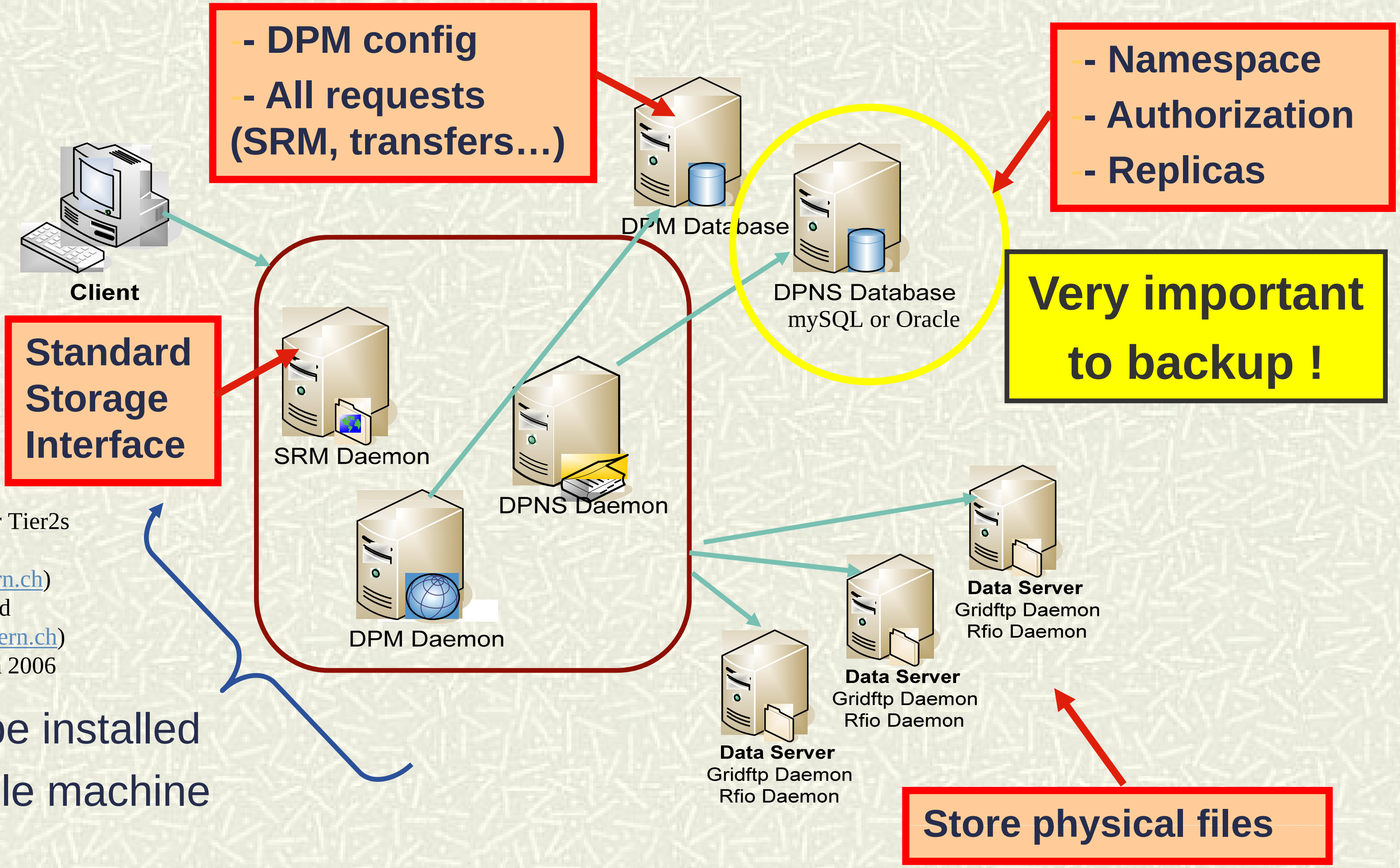


https://twiki.grid.iu.edu/pub/Documentation/StorageDcacheOverviewHardwareLayout/OsgTier2DcacheInstall.png

dCache SE Considerations

- Entry Cost
  - 6-12 Nodes + n Disk Server Nodes
  - Configuration difficulty is medium to hard
- Software Dependencies
  - Java 1.5.0, PostgreSQL 8+, Oracle 10g or DB2 v9
    - Possibly Globus MDS (LCG SE)
- Scaling
  - I/O scaling very good, client node scaling may be below average
- Data Access
  - dcap, minimal native xroot, built-in SRM
- Other Considerations
  - MSS integration supported
  - Requires database backup/restore plan

DPM



- DPM config
- All requests (SRM, transfers…)

- Namespace
- Authorization
- Replicas

Very important to backup !

Standard Storage Interface

Can all be installed on a single machine

Store physical files

DPM Administration for Tier2s
Sophie Lemaitre
(Sophie.Lemaitre@cern.ch)
Jean-Philippe Baud
(Jean-Philippe.Baud@cern.ch)
Tier2s tutorial – 15 Jun 2006

SLAC
NATIONAL ACCELERATOR LABORATORY

DPM SE Considerations

- Entry Cost
  - 2 Nodes + n Disk Server Nodes
  - Configuration is of medium difficulty
- Software Dependencies
  - Java, Globus, mySQL or Oracle
    - Several others available via LCG distribution
- Scaling
  - I/O scaling average, client node scaling may be below average
- Data Access
  - rfio, integrated xroot, built-in SRM
- Other Considerations
  - No MSS support (though in plan)
  - Requires database backup/restore plan

GPFS

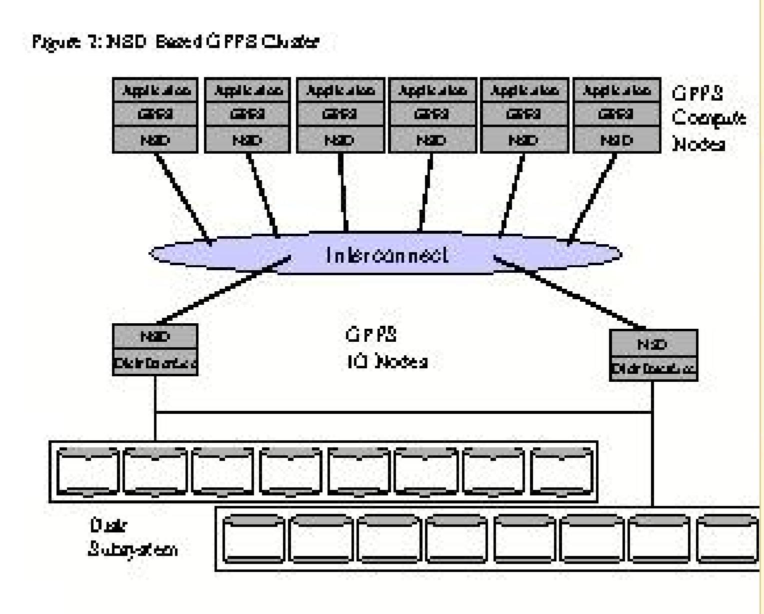
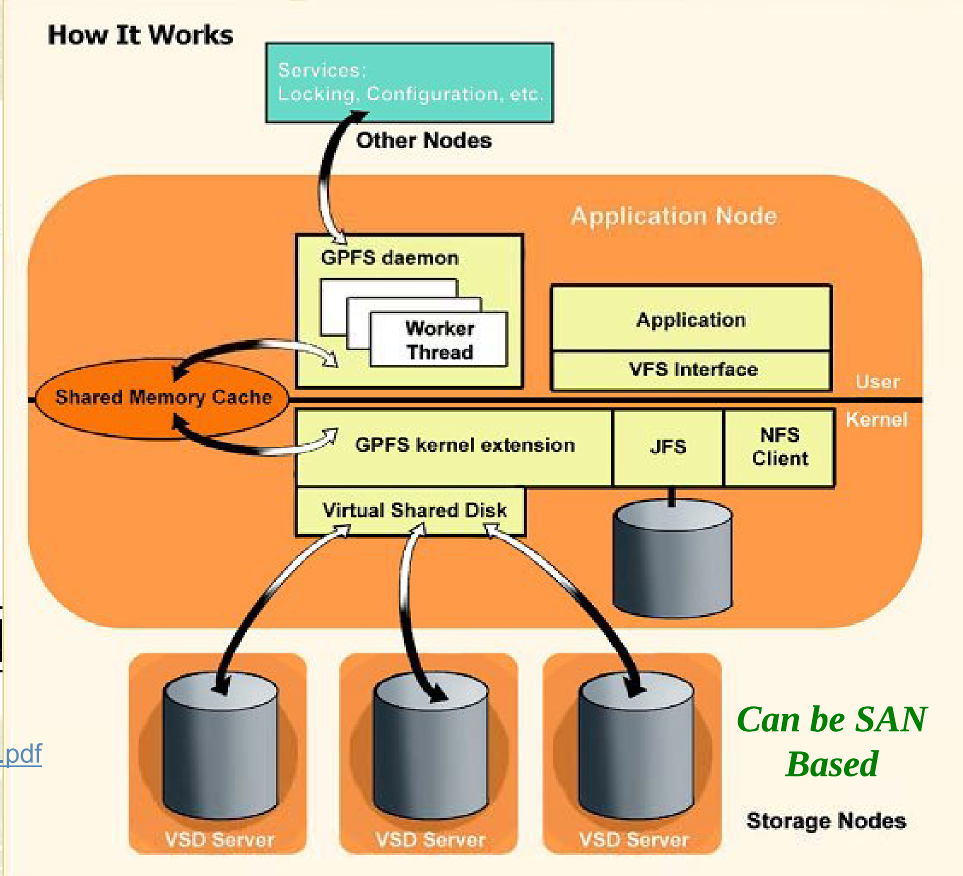


Figure 7: NSD Based GPFS Cluster

http://www.fortuitous.com/docs/primers/Cluster_Filesystems_Intro.pdf



Can be SAN Based

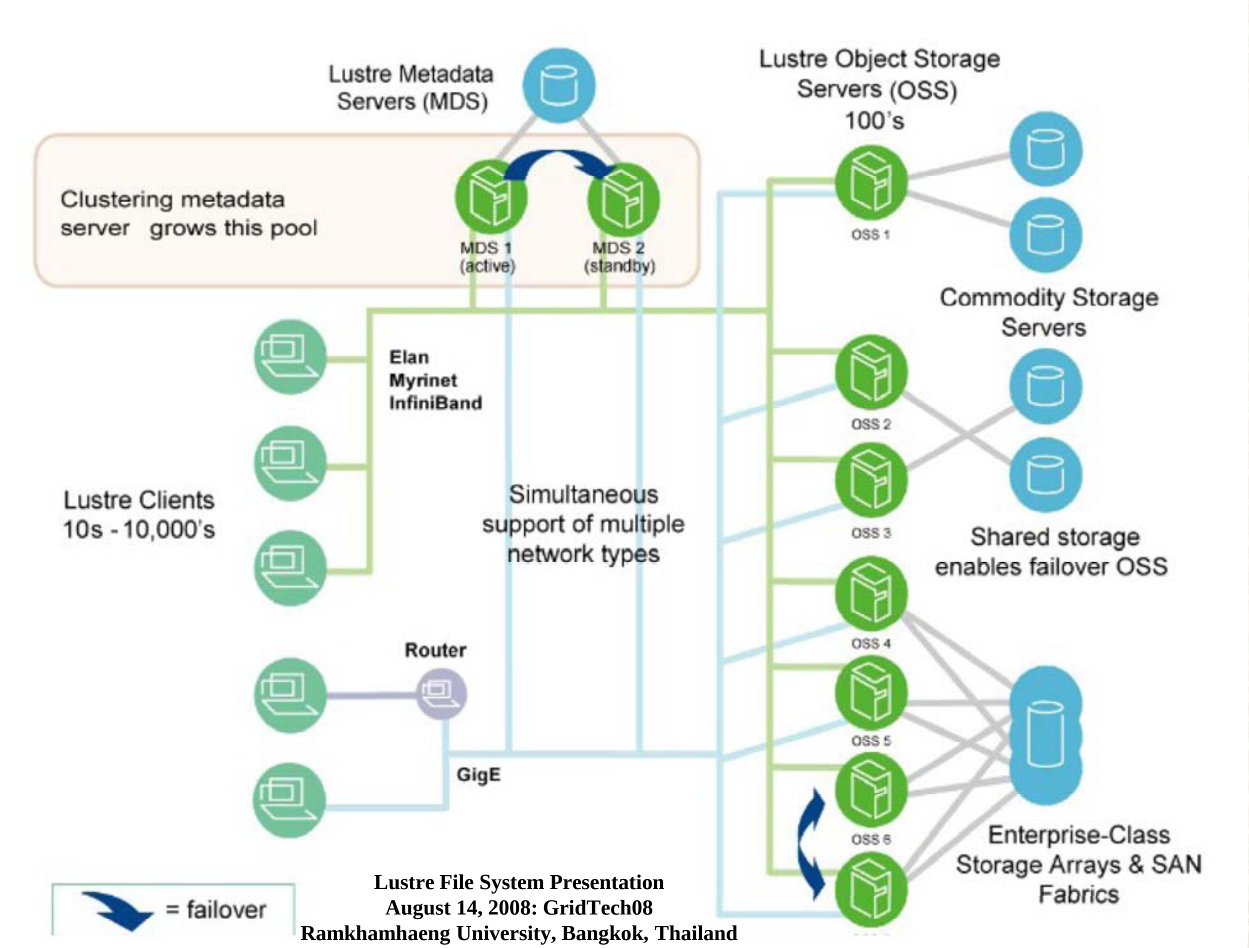
Dominique A. Heger, Fortuitous Technology, (dom@fortuitous.com), Austin, TX, 2006

Architectural and Design Issues in the General Parallel File System, May, 2005, Benny Mandler - mandler@il.ibm.com

GPFS SE Considerations

- Entry Cost
  - 1-2 Nodes + n Disk Server Nodes (0 if SAN based)
  - Configuration difficulty is medium+
- Software Dependencies
  - Linux Portability Layer patches (limited kernel versions)
- Scaling
  - I/O scaling very good, client node scaling above average (500-1,000 nodes documented)
    - Requires tight tuning and relatively robust configuration to achieve this
- Data Access
  - Local VFS, NFS v3, SRM via BestMan or StoRM
- Other considerations
  - MSS integration supported via DMAPI (currently only HPSS)
  - No database requirements
  - Ultimate performance configuration introduces multiple single points of failure
  - Is not free (talk to your IBM representative)

Lustre

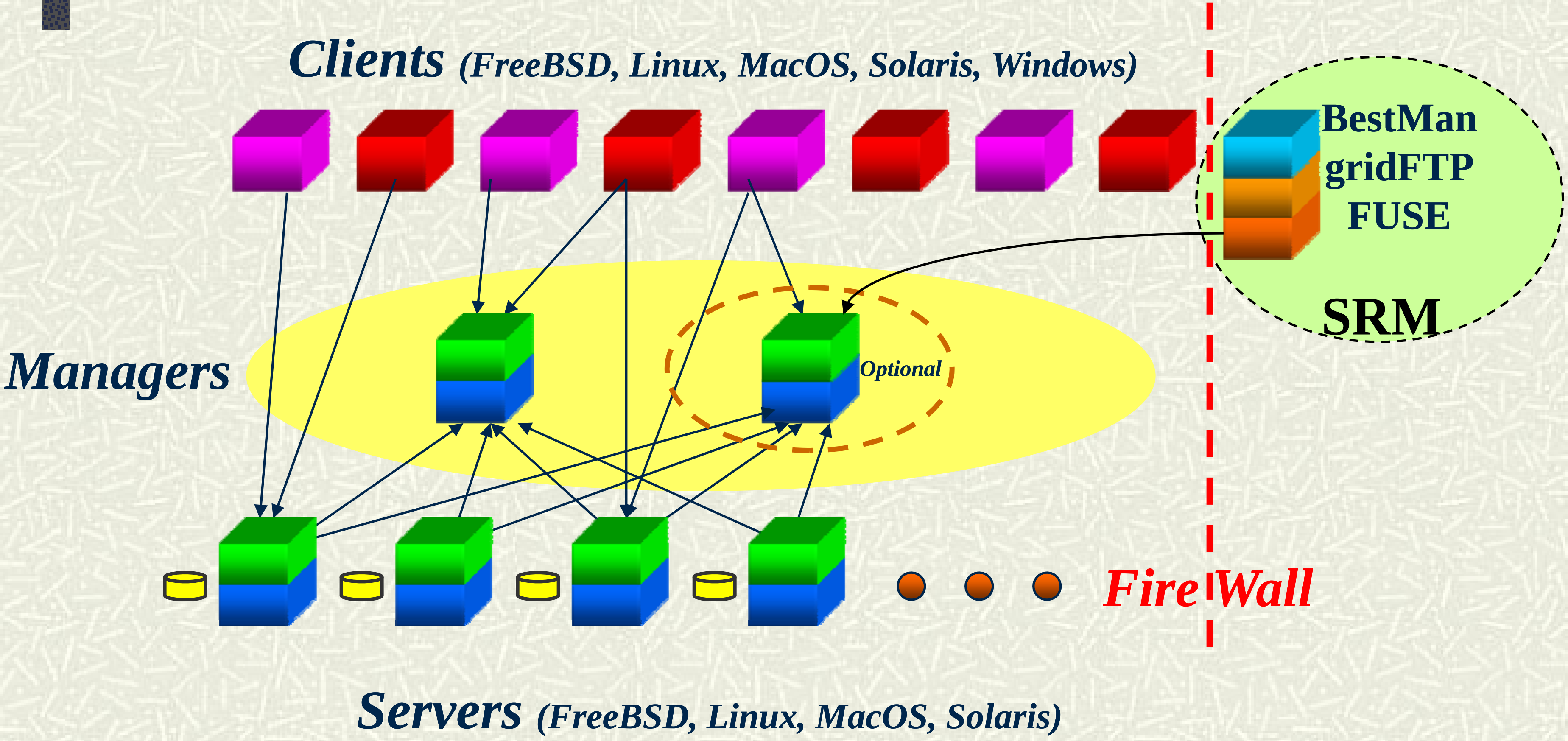

Lustre Metadata Servers (MDS)
Clustering metadata server grows this pool
MDS 1 (active)
MDS 2 (standby)
Lustre Object Storage Servers (OSS) 100's
OSS 1
Commodity Storage Servers
OSS 2
OSS 3
Shared storage enables failover OSS
OSS 4
OSS 5
OSS 6
Enterprise-Class Storage Arrays & SAN Fabrics
Elan
Myrinet
InfiniBand
Lustre Clients 10s - 10,000's
Simultaneous support of multiple network types
Router
GigE
= failover

Lustre File System Presentation
August 14, 2008: GridTech08
Ramkhamhaeng University, Bangkok, Thailand
Kittirak Moungmingsuk kittirak@yahoo.com

Lustre SE Considerations

- Entry Cost
  - 1Node (MDS) + n Disk Server Nodes (OSS)
  - Configuration is medium
- Various Software Dependencies
  - Lustre oriented patches for Linux
    - All servers. All clients if mounting file system (or use preload liblustre.so)
- Scaling
  - I/O scaling excellent, client node scaling very good (1,000+ nodes documented)
    - Requires tight tuning and relatively robust configuration to achieve this
- Data Access
  - Local VFS, SRM via BestMan or StoRM
- Other Considerations
  - No MSS integration
  - Integrated MDS inode database requires backup/restore plan
  - Ultimate performance configuration introduces multiple single points of failure
  - Still needs reliability and stability improvements
    - Developments in pipeline from Sun Microsystems

Scalla (a.k.a. xrootd)



Clients ***(FreeBSD, Linux, MacOS, Solaris, Windows)***

Managers

Servers ***(FreeBSD, Linux, MacOS, Solaris)***

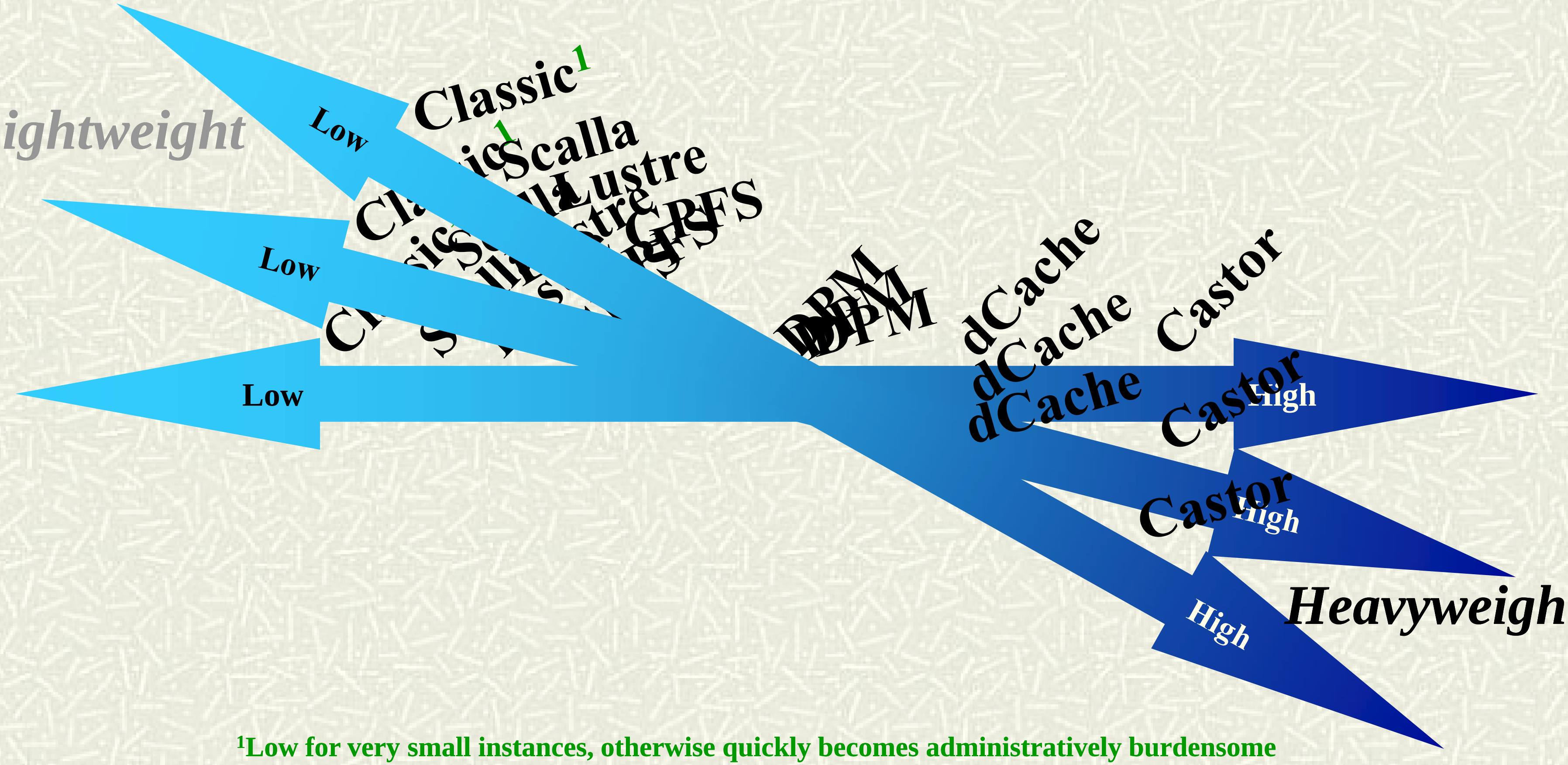
Scalla SE Considerations

- Entry Cost
  - 1-2 Nodes + n Disk Server Nodes
  - Configuration is easy
- Software Dependencies
  - Linux FUSE for full SRM functionality
- Scaling
  - I/O scaling and client node scaling very good
- Data Access
  - xroot, SRM via BestMan
- Other considerations
  - MSS integration supported
  - No database requirements
  - No transactional consistency

SE Resource Requirements Spectrum

In terms of hardware resources and administrative effort

Lightweight



Heavyweight

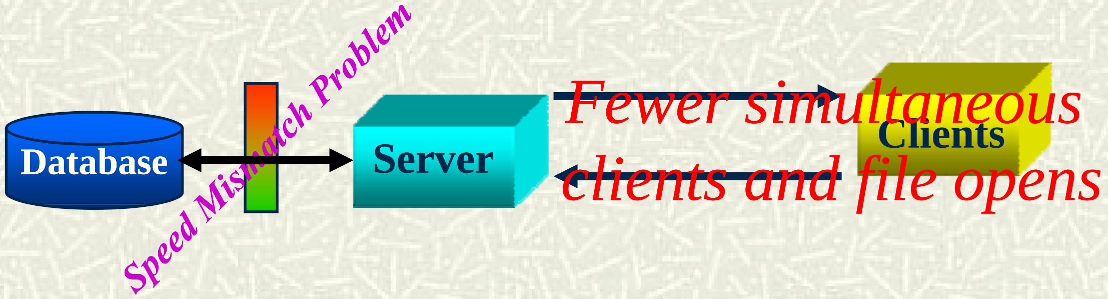
[1] **Low for very small instances, otherwise quickly becomes administratively burdensome**

The Heavy in Heavyweight

- SE’s following IEEE P1244 Storage Reference Model
  - Well intentioned but misguided effort 1990-2002
  - All components are deconstructed into services
    - Name Service, Bit File Service, Storage Service, etc.
    - LFN and PFN are intrinsically divorced from each other
  - Requires database to piece individual elements together
    - Resolution must be done in real-time
- Feature rich design
  - Mostly in managerial elements
    - File placement, access control, statistics, etc.

The Heavyweight Price

- Deconstruction allows great flexibility
  - However, design prone to architectural miss-matches
- Invariably a database in placed in **the line of fire**
  - Every meta-data operation involves DB interactions
    - Impacts the most common operations: open() and stat()
- Databases are not inherently bad and are needed
  - It’s the placement that introduces problems
  - Component latency differences ➡ pile-ups & melt-downs
    - Implies more hardware, schema evolution, & constant tuning



Fewer simultaneous clients and file opens

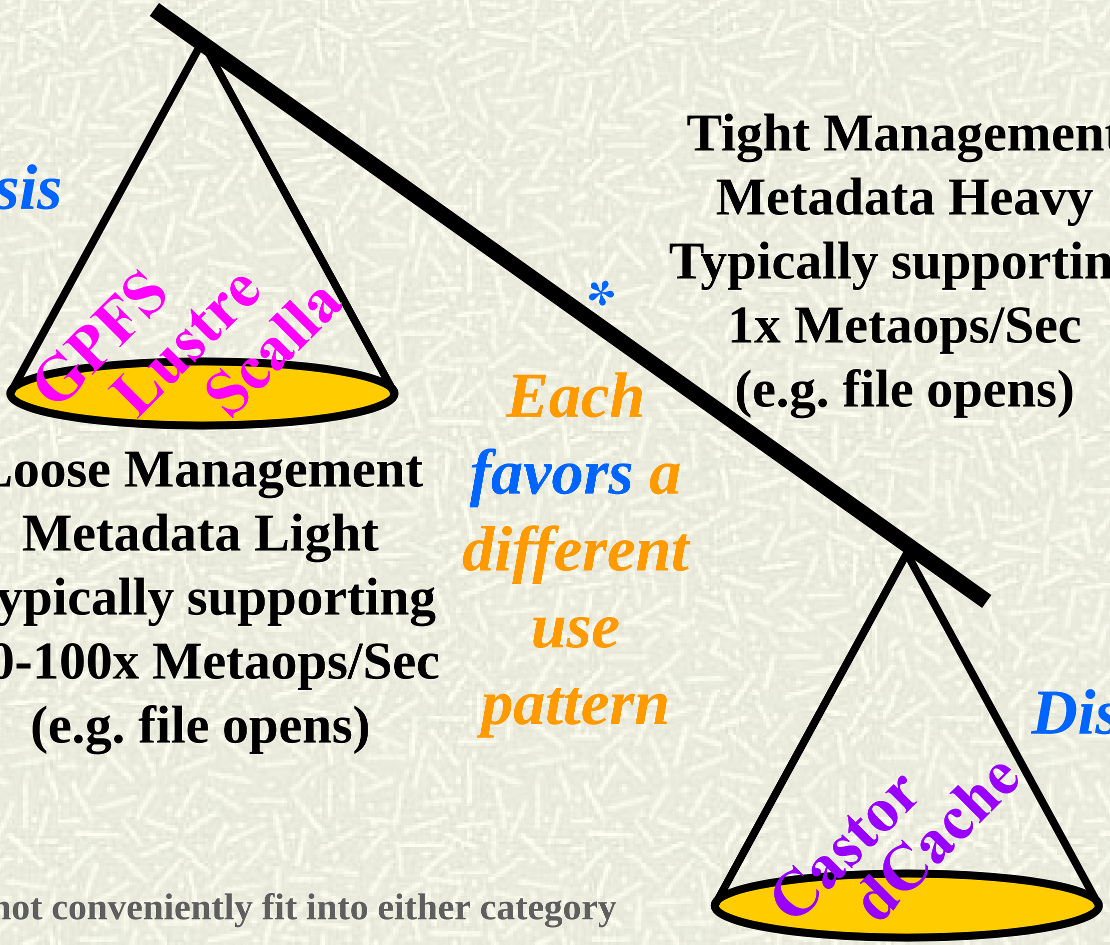
The Light in Lightweight

- Typified by a lean file system
  - Relies on OS primitives for most core functions
    - Latency follows kernel and file system performance
- Limited features to reduce overhead complications
- Components are rarely deconstructed
  - LFN and PFN are intrinsically tied to each other
    - External databases may be required but ***not in*** **the line of fire**
- Naturally supports high transaction rates
  - Job start-up & file open time vastly reduced
    - Avoids pile-up & meltdown problems
  - Less hardware and people are required

The Lightweight Price

- Fewer features
- Less intricate space management controls
    - Challenging to support multiple experiments on same hardware
        - Typically solved by physical partitioning
            - Which in some cases is preferable

The Nutshell Difference



Analysis

Tight Management
Metadata Heavy
Typically supporting
1x Metaops/Sec
(e.g. file opens)

*

Loose Management
Metadata Light
Typically supporting
10-100x Metaops/Sec
(e.g. file opens)

Each favors a different use pattern

Data Distribution

*DPM does not conveniently fit into either category

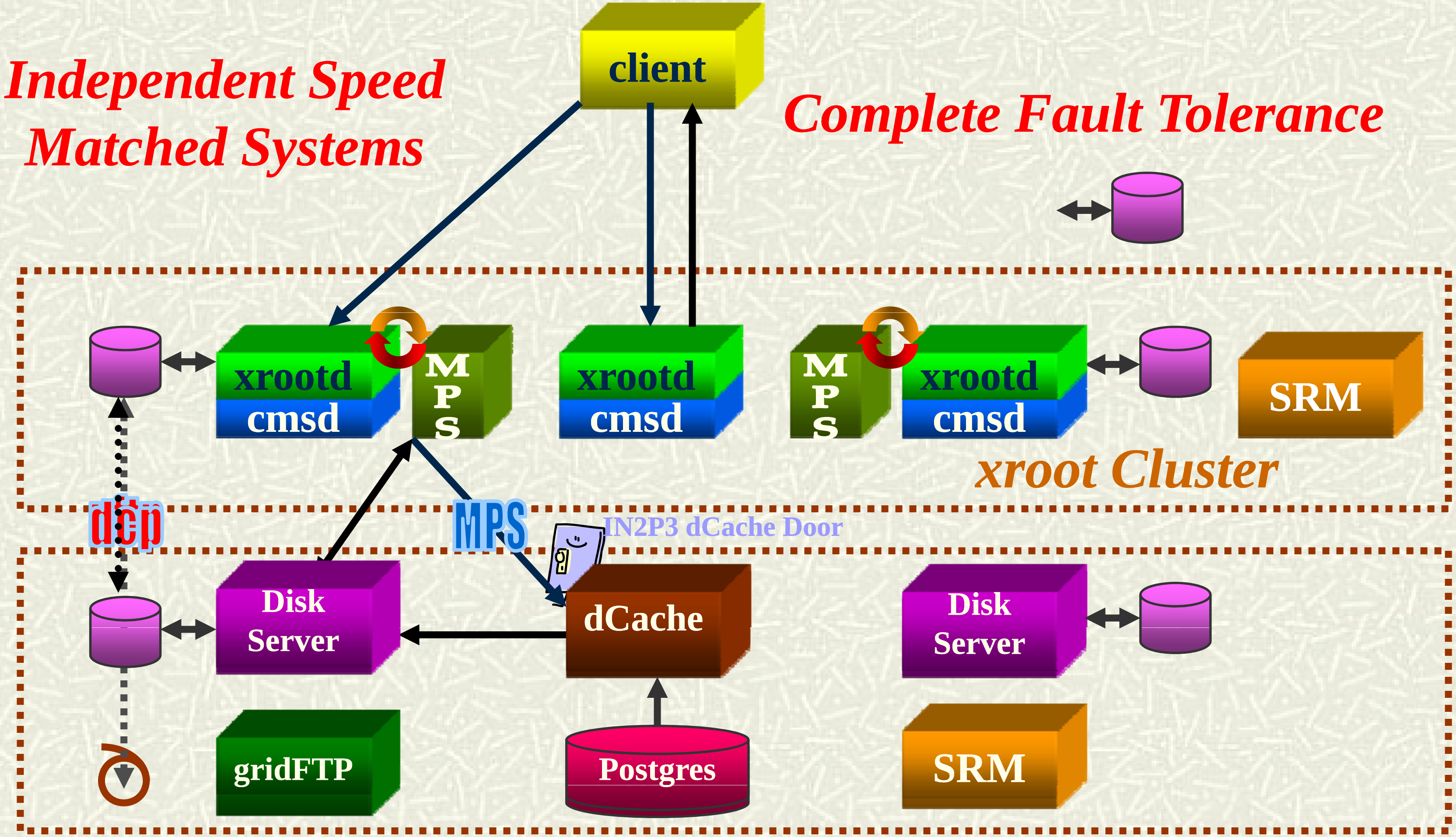
Why Worry?

- Analysis/SE speed mismatches may be fatal
  - Analysis jobs are up-front and swift-footed
  - Typically produce 1000’s meta-ops per second
    - Either due to large number of simultaneous jobs or
    - Reliance on common root-based condition databases
      - 10-100 files swiftly opened by every job
      - 100 jobs can easily produce over 5,000 file opens/second
        - A 100ms latency may cause a deadly 8 minute startup
- May easily overwhelm a heavyweight SE
  - The hard lesson learned from BaBar

Lightweight Only?

- One might think that …
  - Production is 80% analysis 20% distribution
    - Lightweight can more easily scale with analysis
- But heavyweight systems are still needed
  - Mass Storage Systems and Grid data distribution
- Using both requires speed-matching techniques
  - A heavy-light melding strategy can be used
    - Heavyweight systems ***kept out*** of **the line of fire**

Heavy-Light Melding Example



Heavy-Light Melding A Panacea?

- Yes and no
  - Allows analysis workload to avoid high latency paths
    - No pile-up and meltdown
  - Does not address device contention
    - Bulk I/O versus small random I/O
  - You still need a configuration sized for the expected load
- Examples of melding
  - CERN Castor/xrootd (using custom Scalla plug-ins)
  - CNAF Castor/GPFS
  - IN2P3 dCache/xrootd (using native Scalla)

Conclusion

- SE architectures *are* ready for LHC *but*….
- Is LHC ready with the appropriate SE architectures?
  - Probably *not* in some cases
    - Historically true and perhaps practically unavoidable
      - Data distribution usually gets the first round of effort
- Successful production analysis will be the final arbiter
  - Some are hedging with other approaches
    - Virtual MSS (ALICE)
    - Heavy-Light Melding
- At this point there is still a lot to do!

Acknowledgements

- Software Contributors
  - Alice: Fabrizio Furano, Derek Feichtinger
  - CERN: Andreas Peters (Castor)
  - GLAST: Tony Johnson (Java)
  - Root: Fons Rademakers, Gerri Ganis (security), Beterand Bellenet (windows)
  - STAR/BNL: Pavel Jackl
  - SLAC: Jacek Becla (monitoring), Tofigh Azemoon, Wilko Kroeger
- Operational Collaborators
  - BNL, INFN, IN2P3
- Partial Funding
  - US Department of Energy
    - Contract DE-AC02-76SF00515 with Stanford University