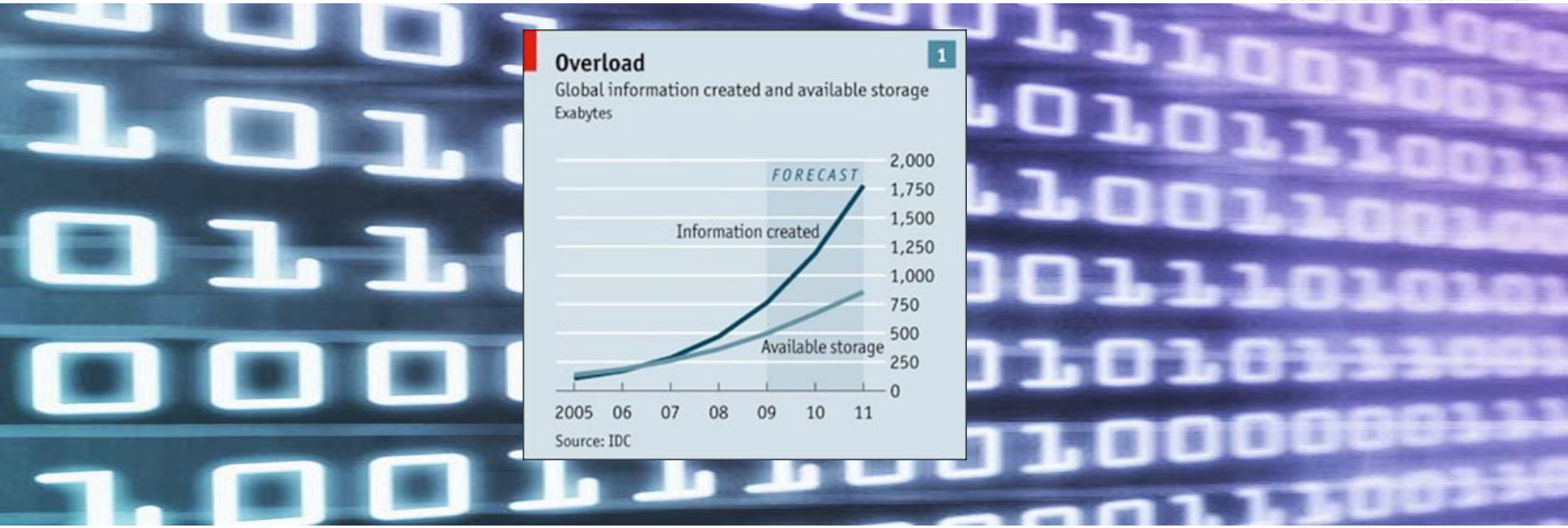
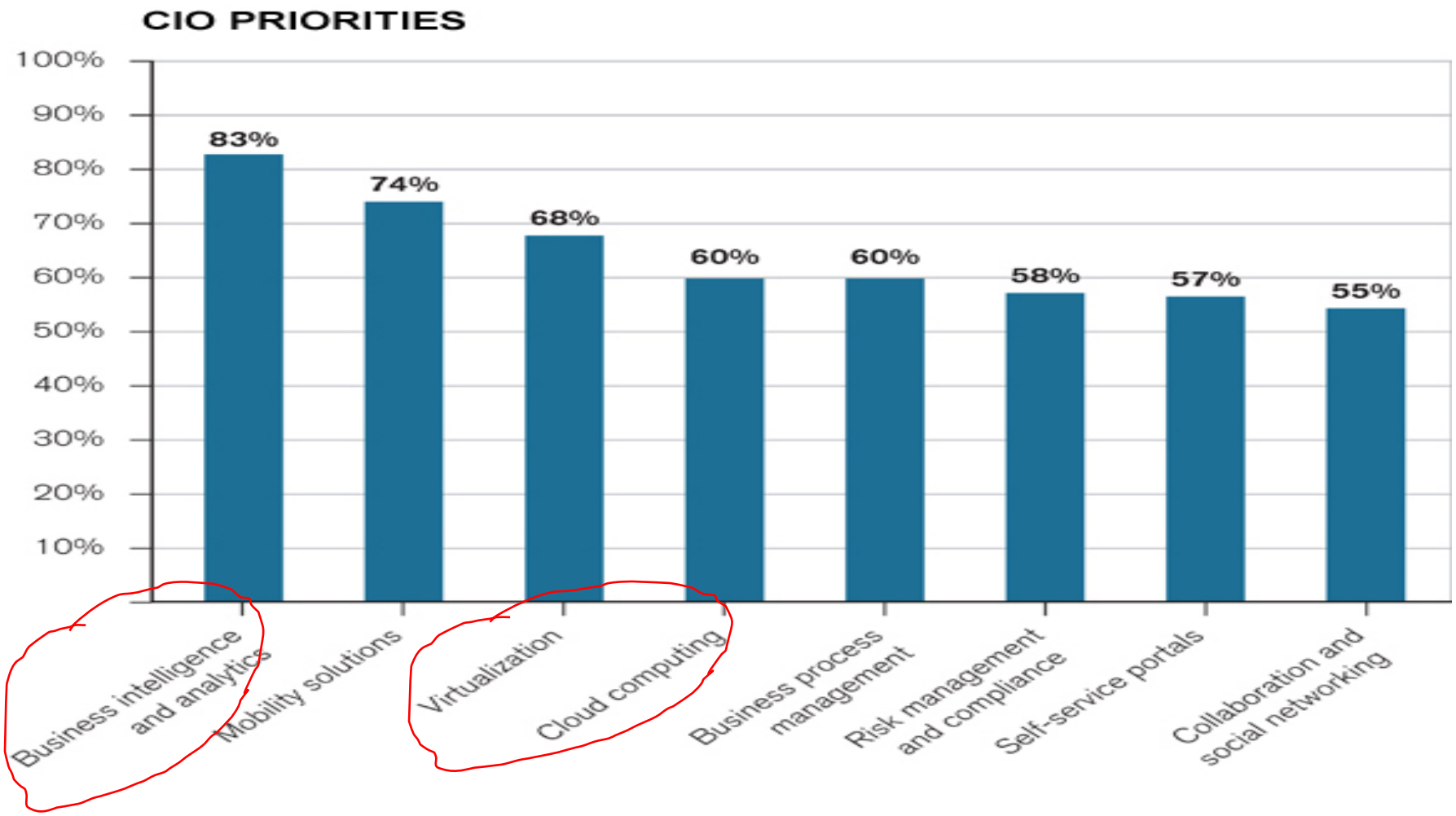



# GPFS-SNC:
# A Cluster File System for Analytics and Clouds

Prasenjit Sarkar, IBM Research

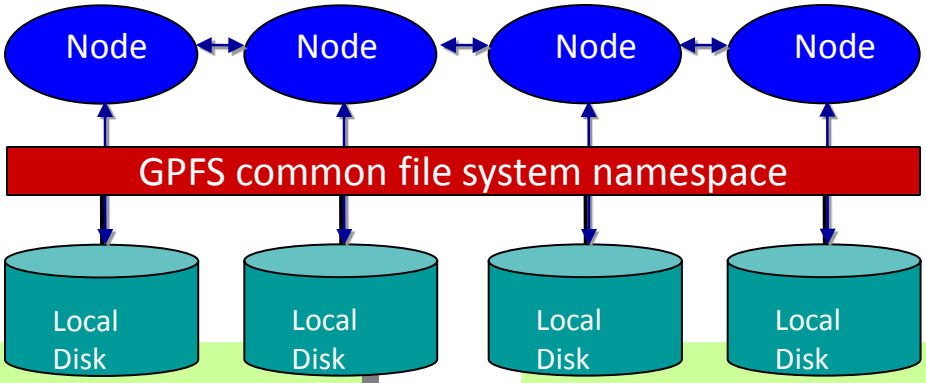
# Analytics and Clouds: Key growth area





# GPFS: Parallel File System






**GPFS Architecture:**

*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.

**Extensions for Shared Nothing Architectures:**

*Locality*
*Write Affinity*
*Metablocks*
*Pipelined replication*
*Distributed recovery*

# GPFS-SNC: Motivation



**Motivation:**

SANs are built for latency not bandwidth

**Advantages of using GPFS:**

High scale (thousands of nodes, petabytes of storage), high performance, high availability, data integrity,

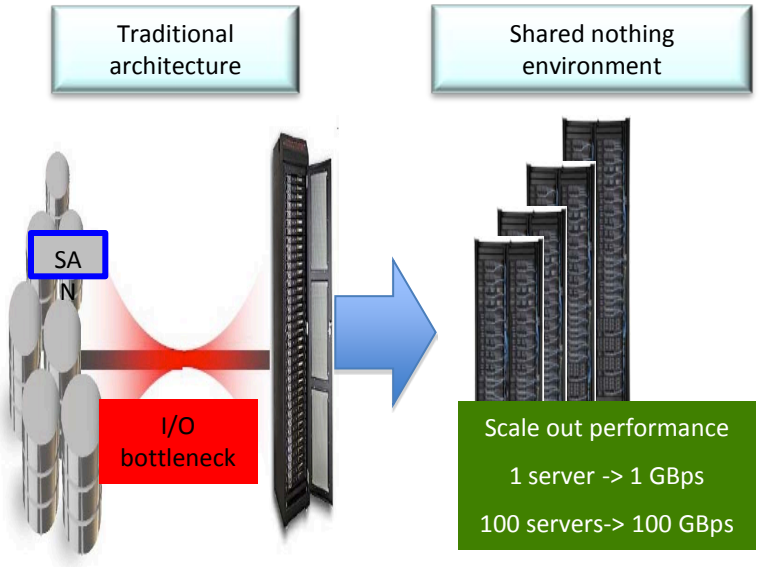
POSIX semantics, workload isolation, enterprise features (security, snapshots, backup/restore, archive, asynchronous caching and replication)

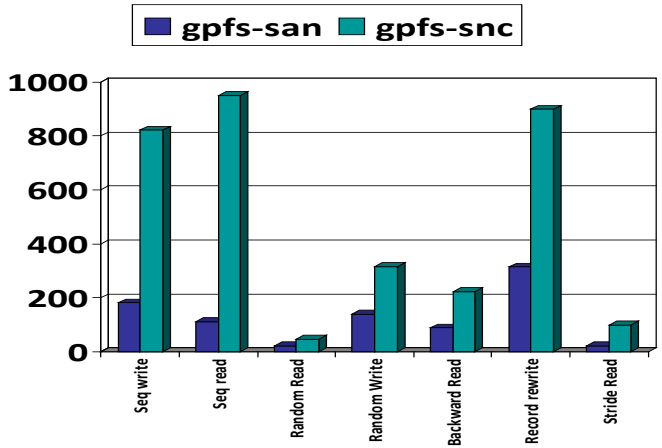
**Challenges:**

Adapt GPFS to shared nothing clusters

Maximize application performance relative to cost

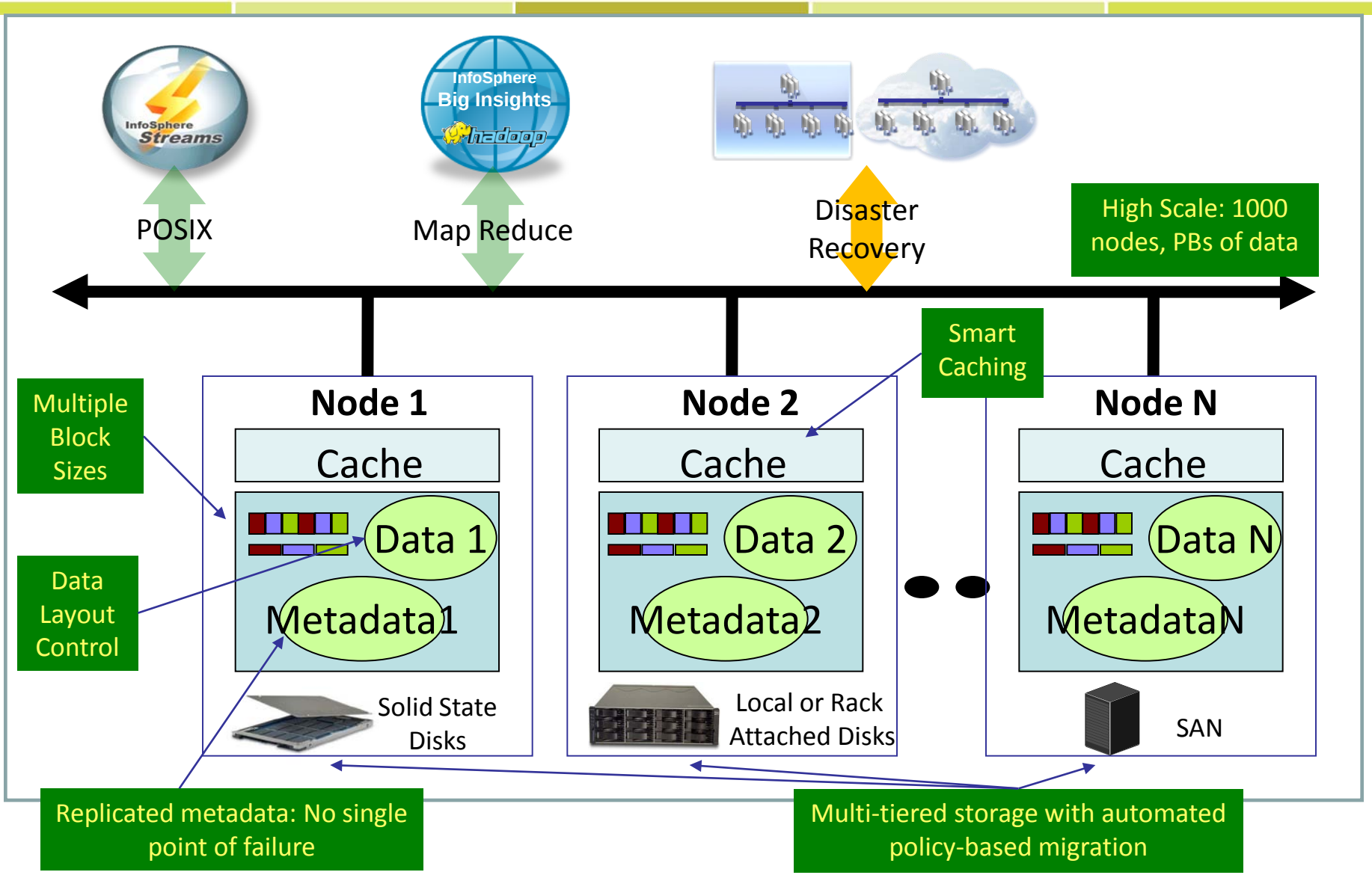
Failure is common, network is a bottleneck



# GPFS-SNC: Architecture

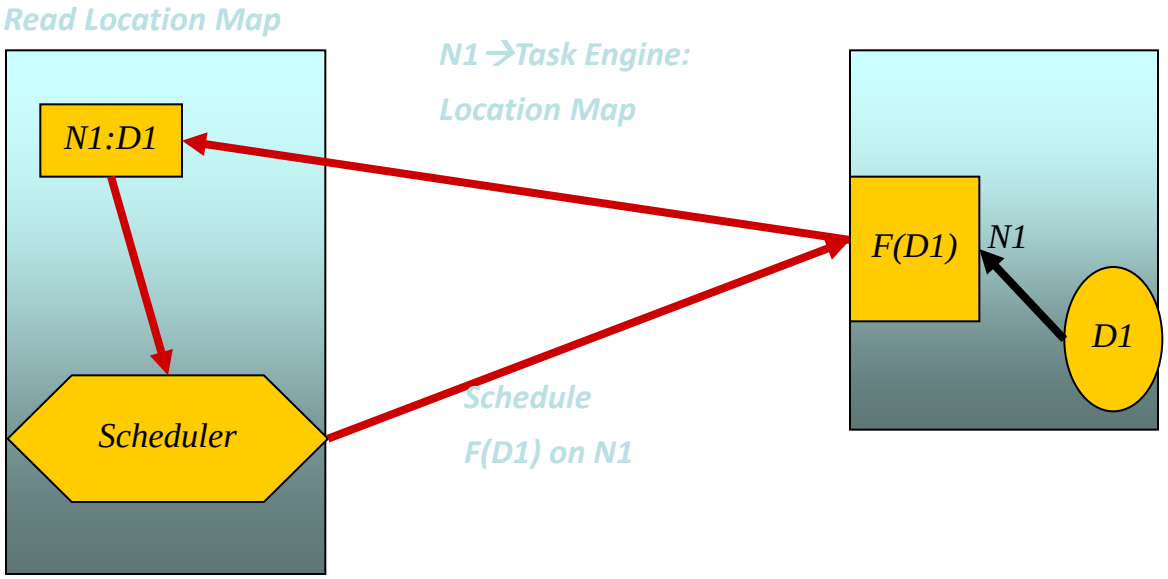


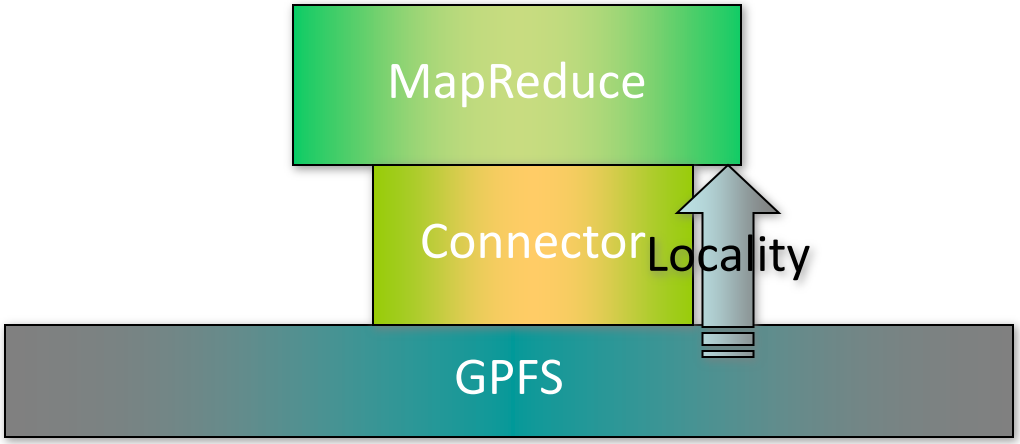


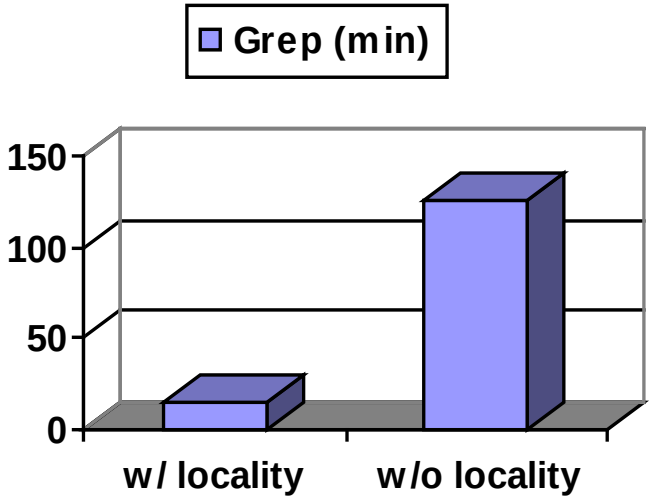
# Locality Awareness



- MapReduce Scheduler assigns tasks to nodes where data chunks reside
  - needs to know the location of each chunk
- GPFS API maps each chunk to its node location
- Map is compacted to optimize its size
- GPFS-Hadoop Connector
  - Adapts Hadoop to transparently use GPFS APIs
  - No changes to Hadoop



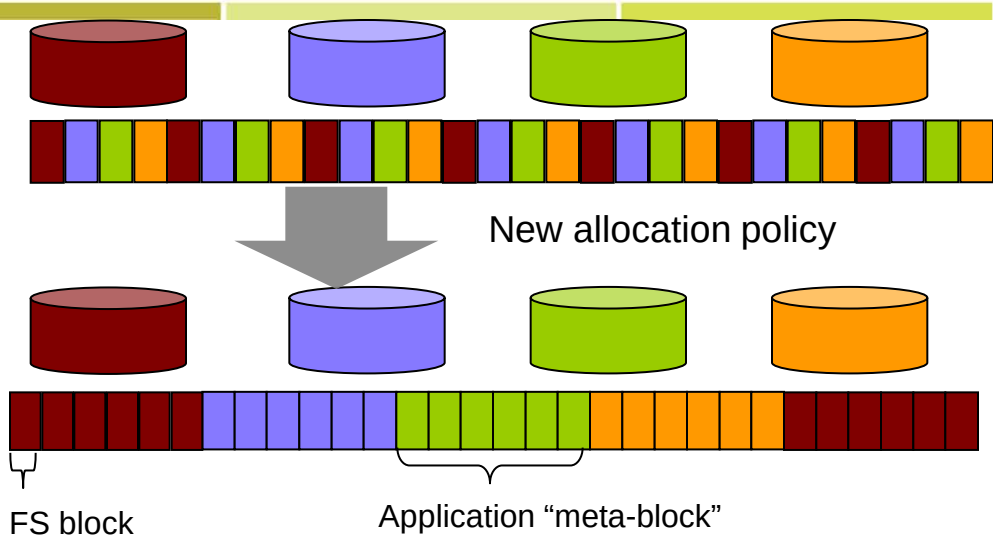



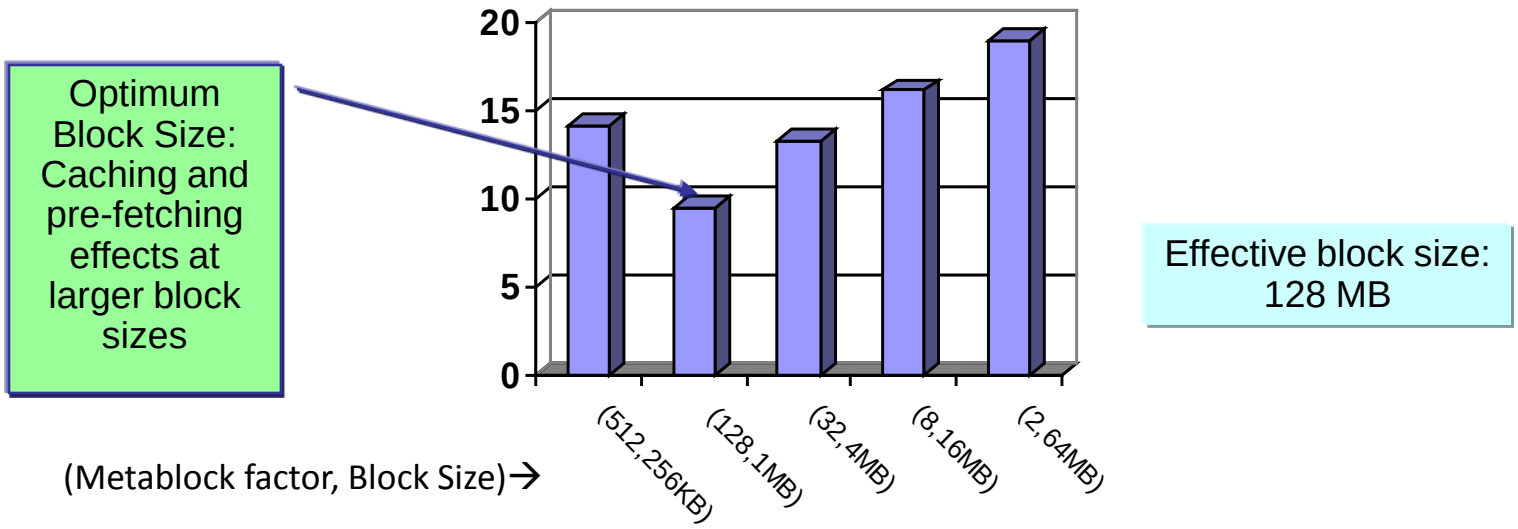



# Metablocks



- MapReduce workloads have two types of access:
  - Small blocks (eg: 512KB) for Index File Lookups
  - Large blocks (eg: 64MB) for File scans
- Solution: Multiple block sizes in the *same* file system
- New allocation scheme
  - Metablock factor for block size
  - Effective block size = block size * Metablock factor
- File blocks are laid out based on effective block size

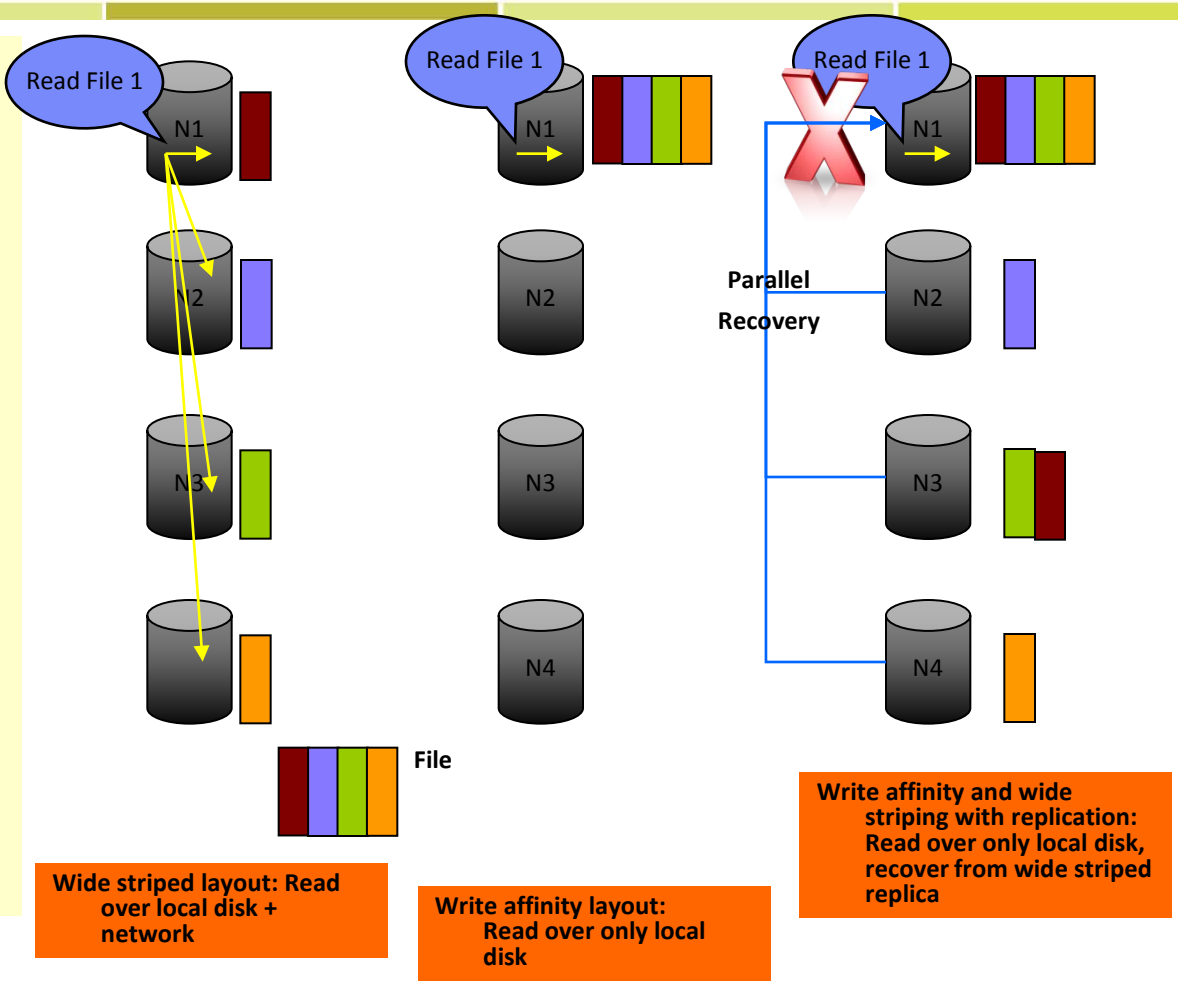


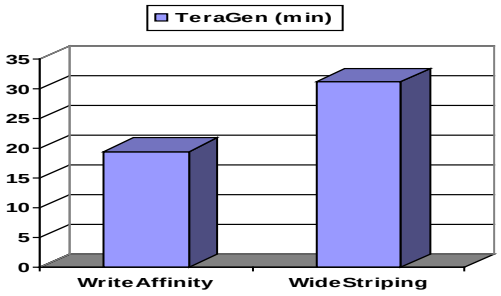



# Write Affinity



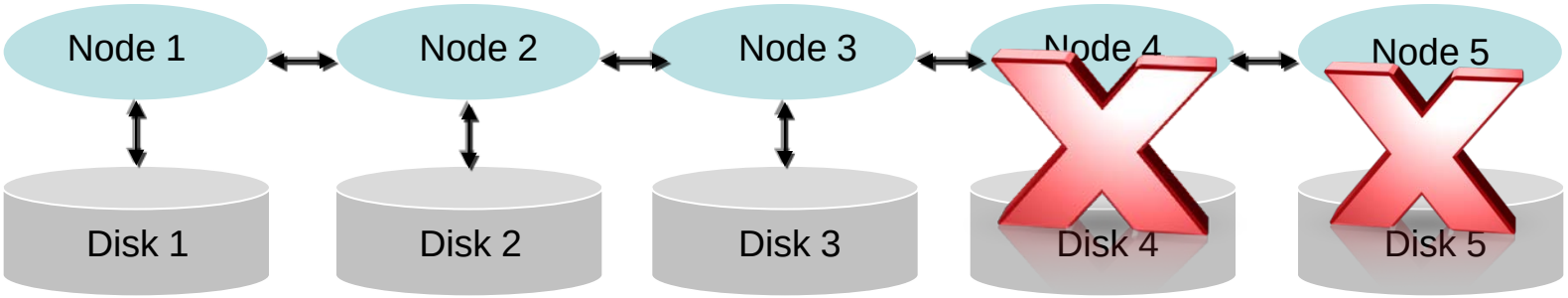
- Wide striping with multiple replicas:
  - No locality of data in a particular node
  - Bad for commodity architectures due to excessive network traffic
- Write Affinity for a set of files (or file-set) on a given node
  - All relevant files will be allocated on the node
- Configurability of replicas
  - Each replica can be configured for either wide striping or write affinity
  - Toplogical support:
    - DataCenter.Cluster.Rack.Node
    - Allows flexibility with networks and failover



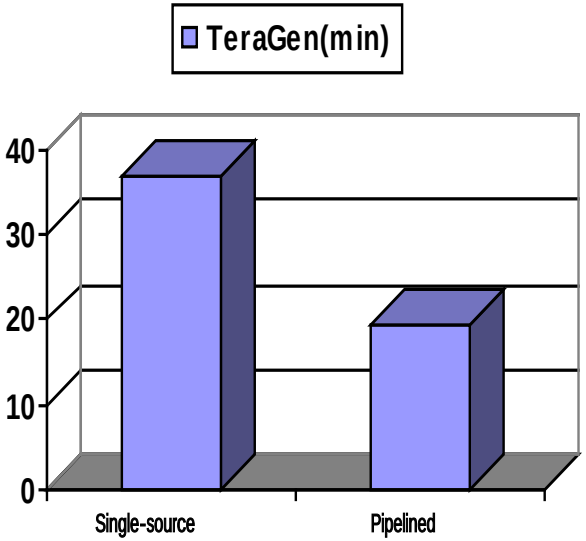



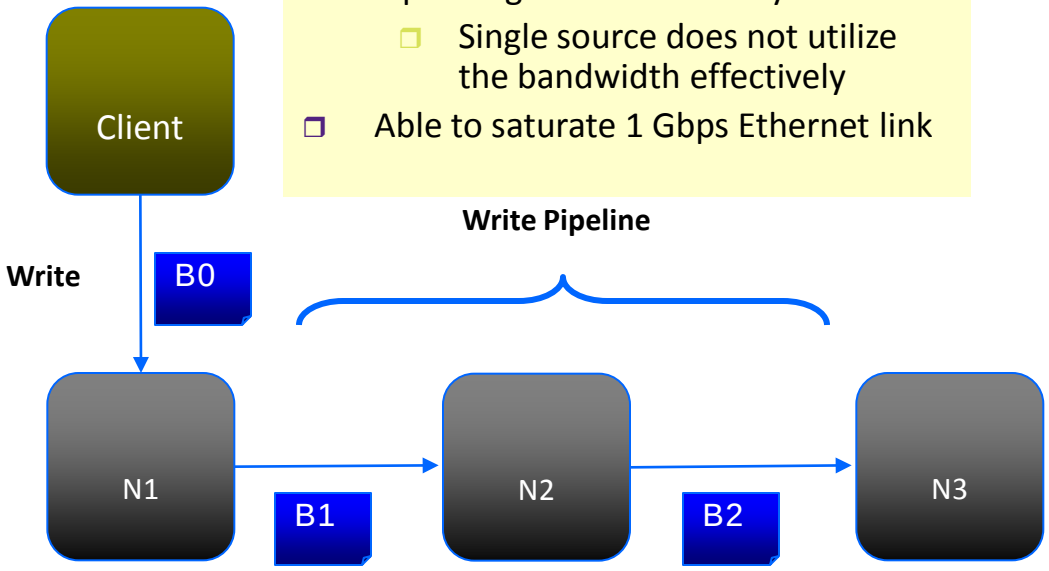
# Pipelined Replication






- Higher Degree of replication
- Pipelining for lower latency
  - Single source does not utilize the bandwidth effectively
- Able to saturate 1 Gbps Ethernet link

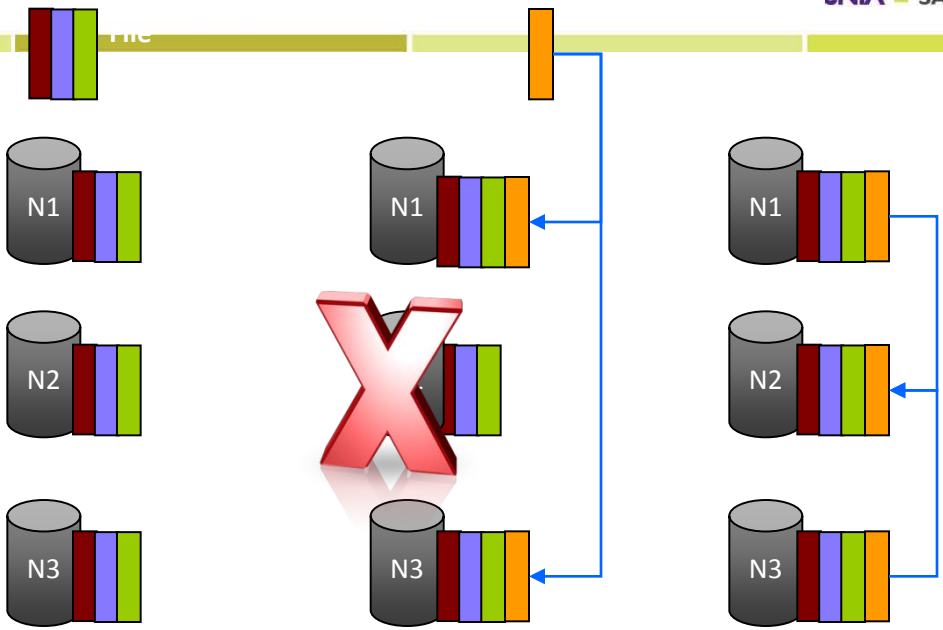


# Fast Recovery



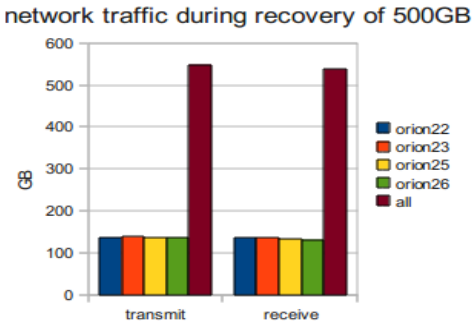
- Handling failures is policy driven
  - Incorporated node failure and disk failure triggers
  - Policy-based recovery
    - Restripe on a node failure or disk failure
    - Alternatively, rebuild the disk when the disk is replaced
- Fast recovery
  - Incremental recovery
    - Keep track of changes during failure and recover what is needed
  - Distributed restripe
    - Restripe load is spread out over all the nodes
    - Quickly figure out what blocks are needed to be recovered when a node fails

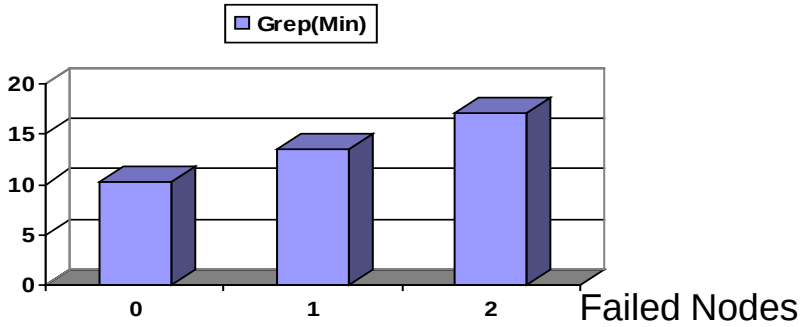



**Well replicated file**

**N2 rebooted:**
**Writes go to N1 and N3**

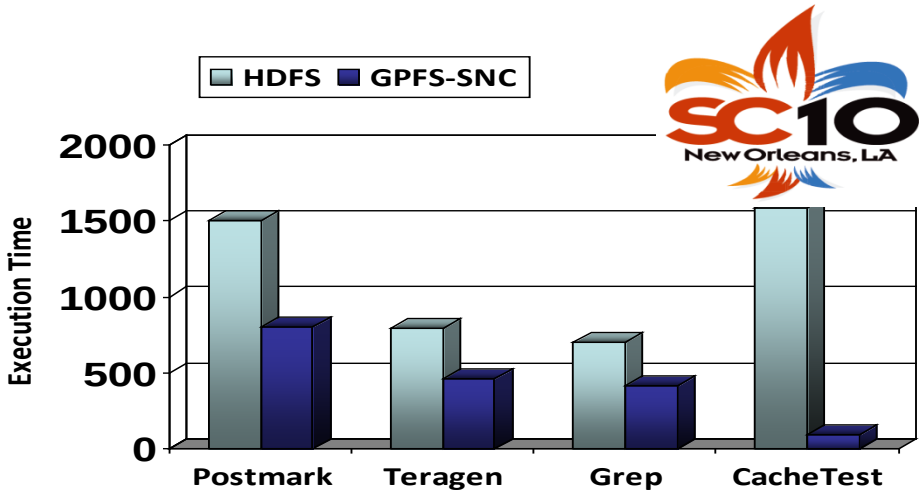
**N2 comes back and catches up by copying missed writes from N1 and N3 in parallel**

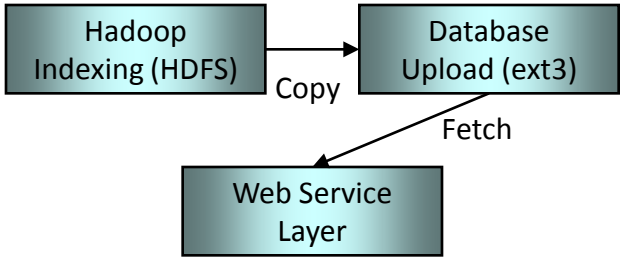


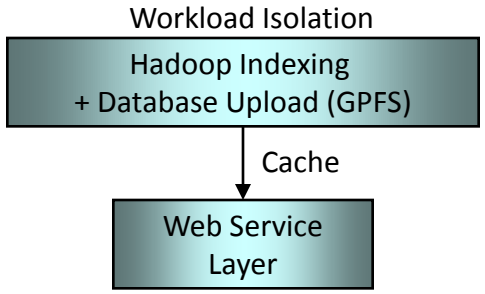



# MapReduce

- Query languages like Pig and JAQL need good random I/O performance
- Sort requires better sequential throughput
  - GPFS is twice HDFS for both of the above
- For document index lookups, client side caching is a big win
  - 17x throughput speedup




*HDFS:*
Extra copy overhead and network fetch, separate clusters for analytics and database

*GPFS:*
Single cluster for analytics and database, no copying required, caching for web layer

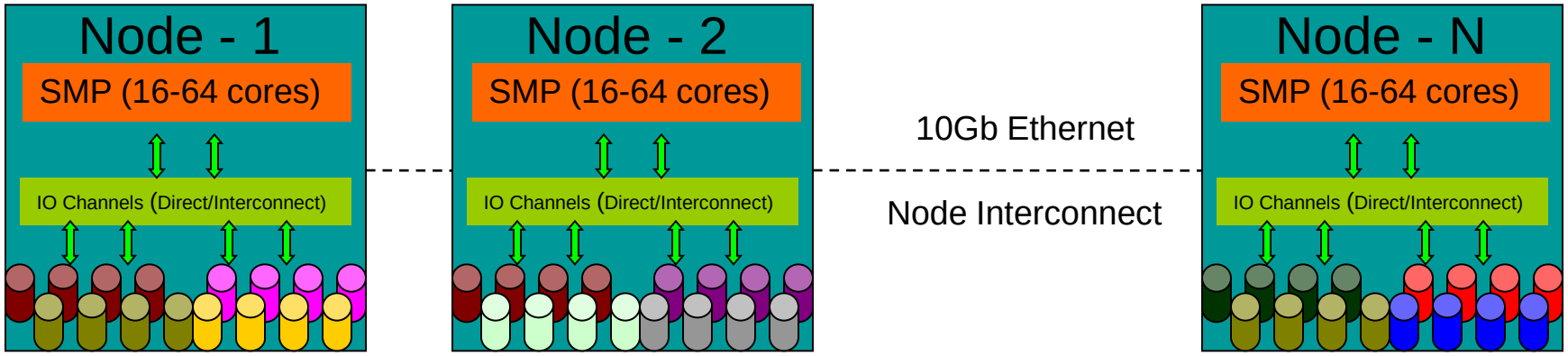
- Proven data integrity
- Replicated metadata services
  - Yahoo keeps 3 copies of 3 versions of HDFS because of unknown data integrity [1]
  - Quantcast deletes files once HDFS is 50% full [2]

[1] Care and Feeding of Hadoop Clusters, Marc Nicosia, Usenix 2009
[2] The Komos Distributed File System, Sriram Rao, Quantcast Inc.

# In-memory Analytic Engines



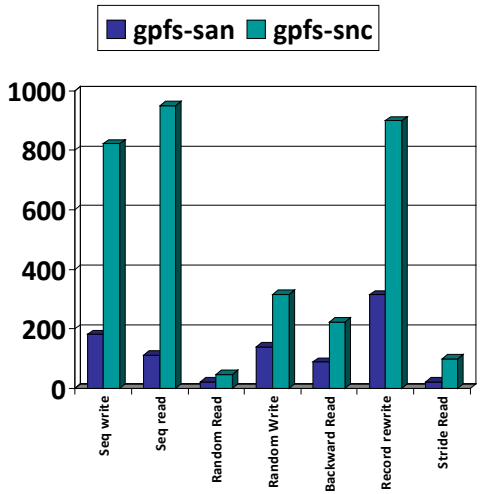
- Why do we need a file system
- Three reasons:
  - Permanent data store
    - Point of ingestion for data
    - Continual process – high bandwidth requirements
  - Logging
    - In-memory transactions are logged to disk or flash
    - High IOPS requirement
  - Failure Recovery
    - Data updates must be replicated for recovery
    - Efficient use of network bandwidth, workload isolation

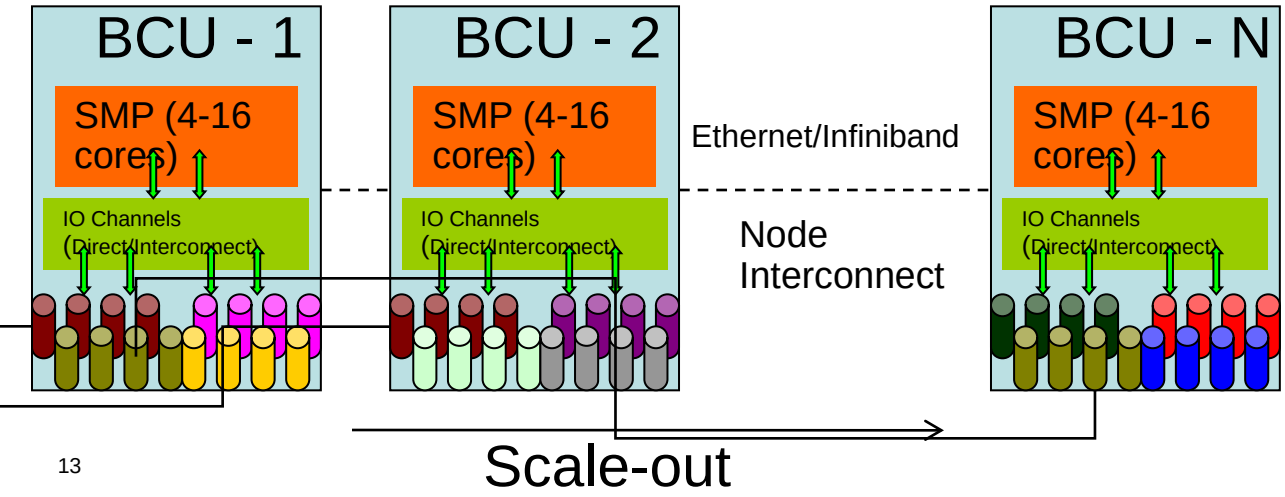


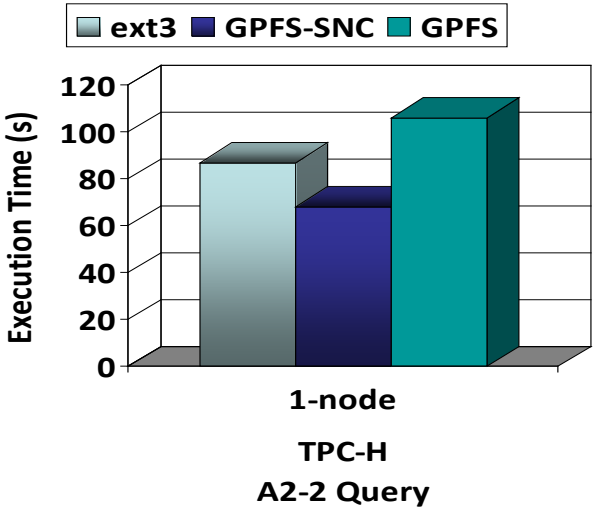
# Scalable Storage for Data Warehousing



- Goal:
  - Provide scale-out storage capability for warehousing
- Motivation: Exploit file system features to accelerate database layer
  - Performance
    - Linearly scale out with the number of nodes
    - Replication: Can saturate Infiniband and 10 GbE networks
  - Benefits:
    - Client-side Caching, Disaster Recovery, Tiering
    - MapReduce techniques
  - Cost:
    - Lower than a SAN configuration

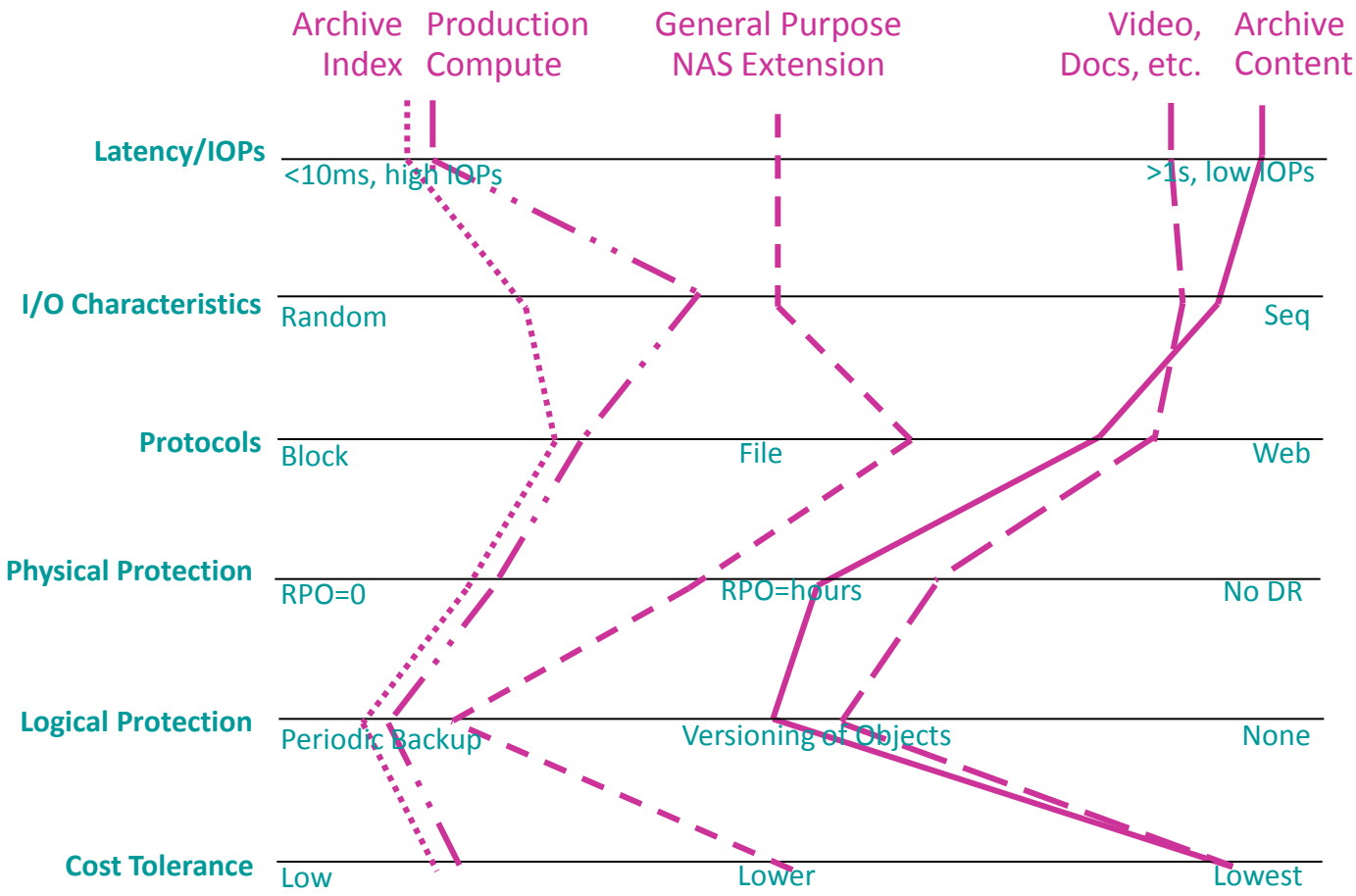


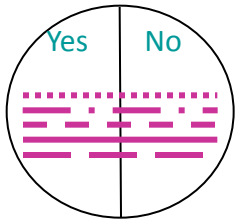
# Storage Clouds: Wide Variety



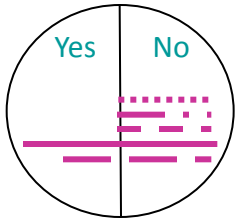



Other Attributes

**Encryption Secure Erase**



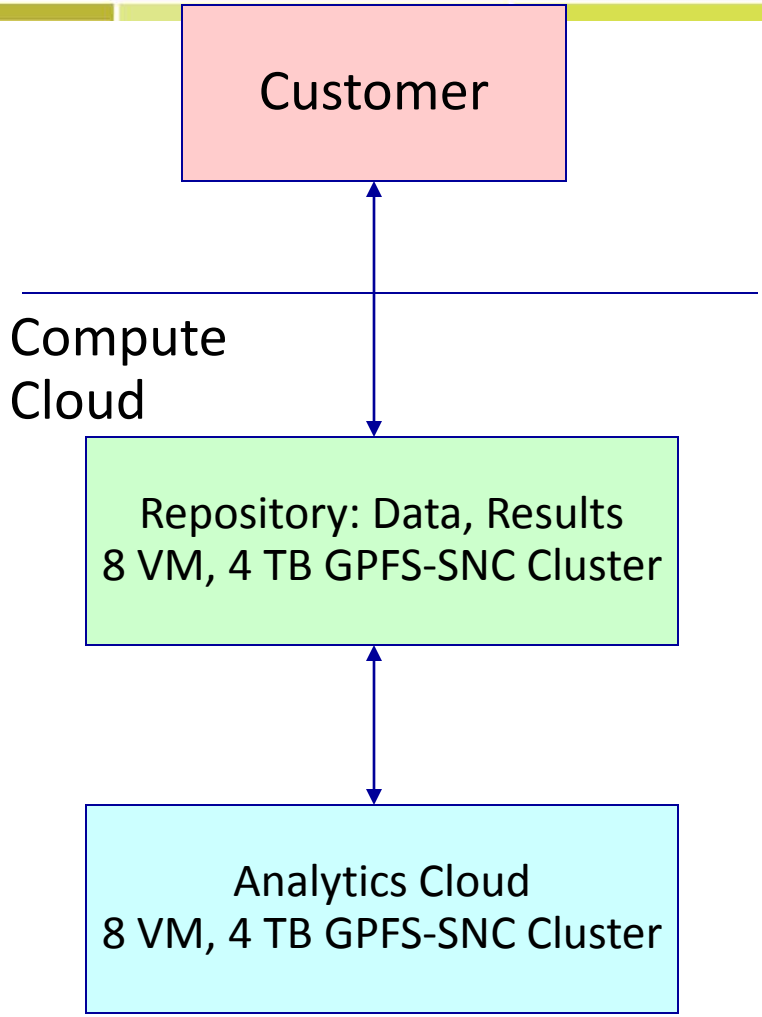

**Guaranteed Immutability**



# Tier-1 Storage for Compute Cloud



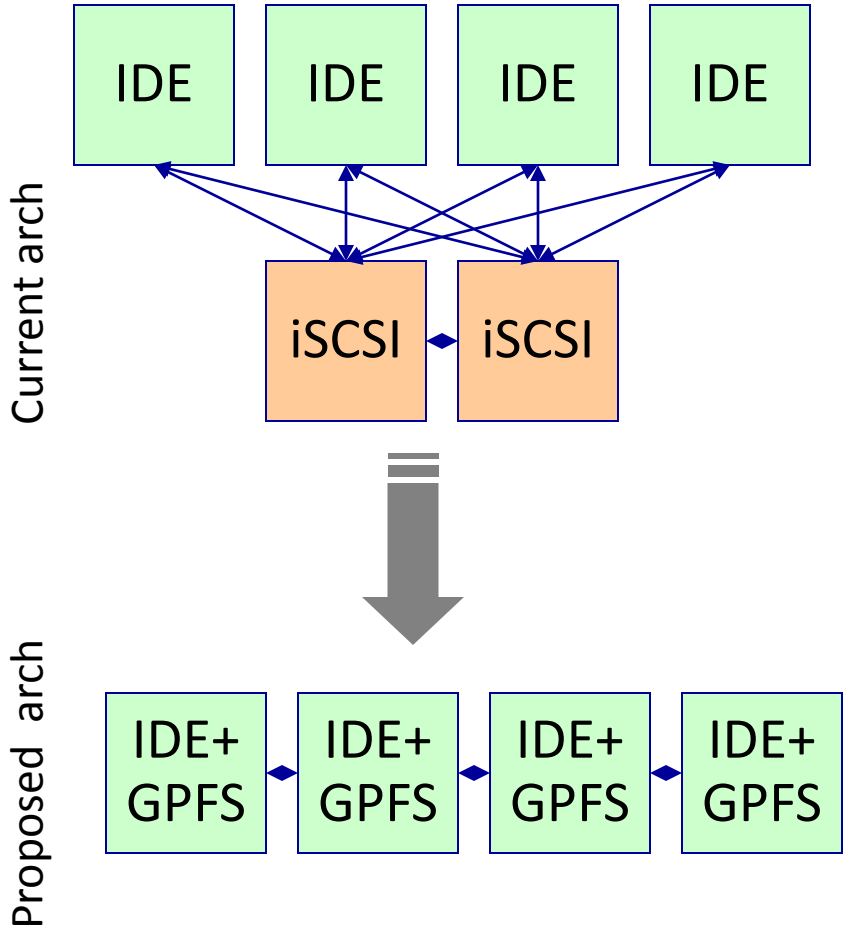
- Based on Generic Compute Cloud
- Data ingested into Compute Cloud onto GPFS-SNC Cluster for staging
  - Referred to as “repository”
- Parallel Copy into analytics GPFS-SNC cluster for processing
- Analytics works in parallel on the data using Compute Cloud resources
- Results of analytics put back into staging area for customer verification



# Image-deployment with GPFS-SNC



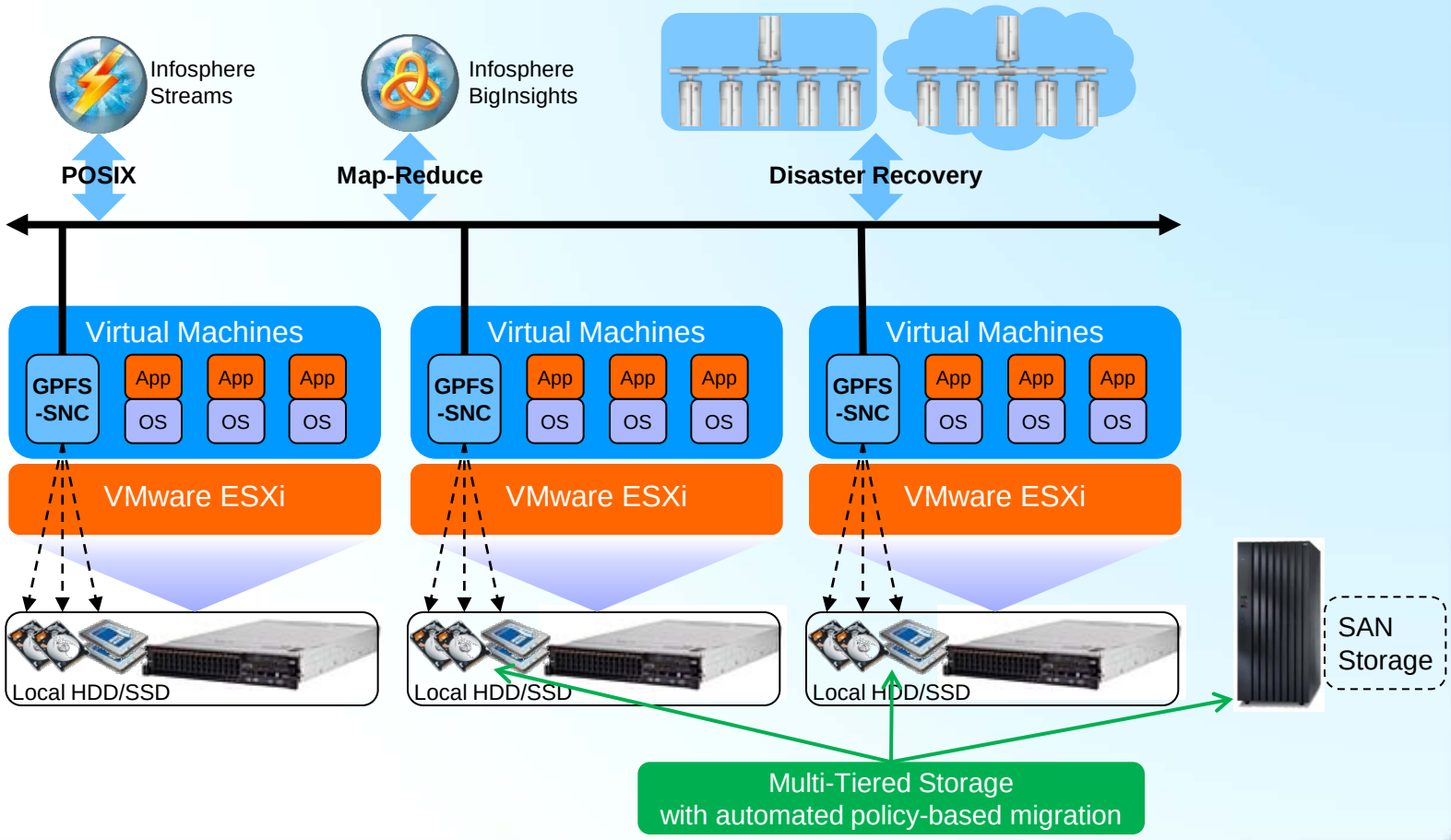
- Advantages of GPFS-SNC
  - No distinct storage and compute nodes: better cloud utilization
  - Automated replication for failure recovery
  - Automatic scalability
  - Intelligent VM placement
  - Instance deployment time reduced from ~6 min to 8 sec with GPFS-SNC and semantic caching




IDE = Image Deployment Engine

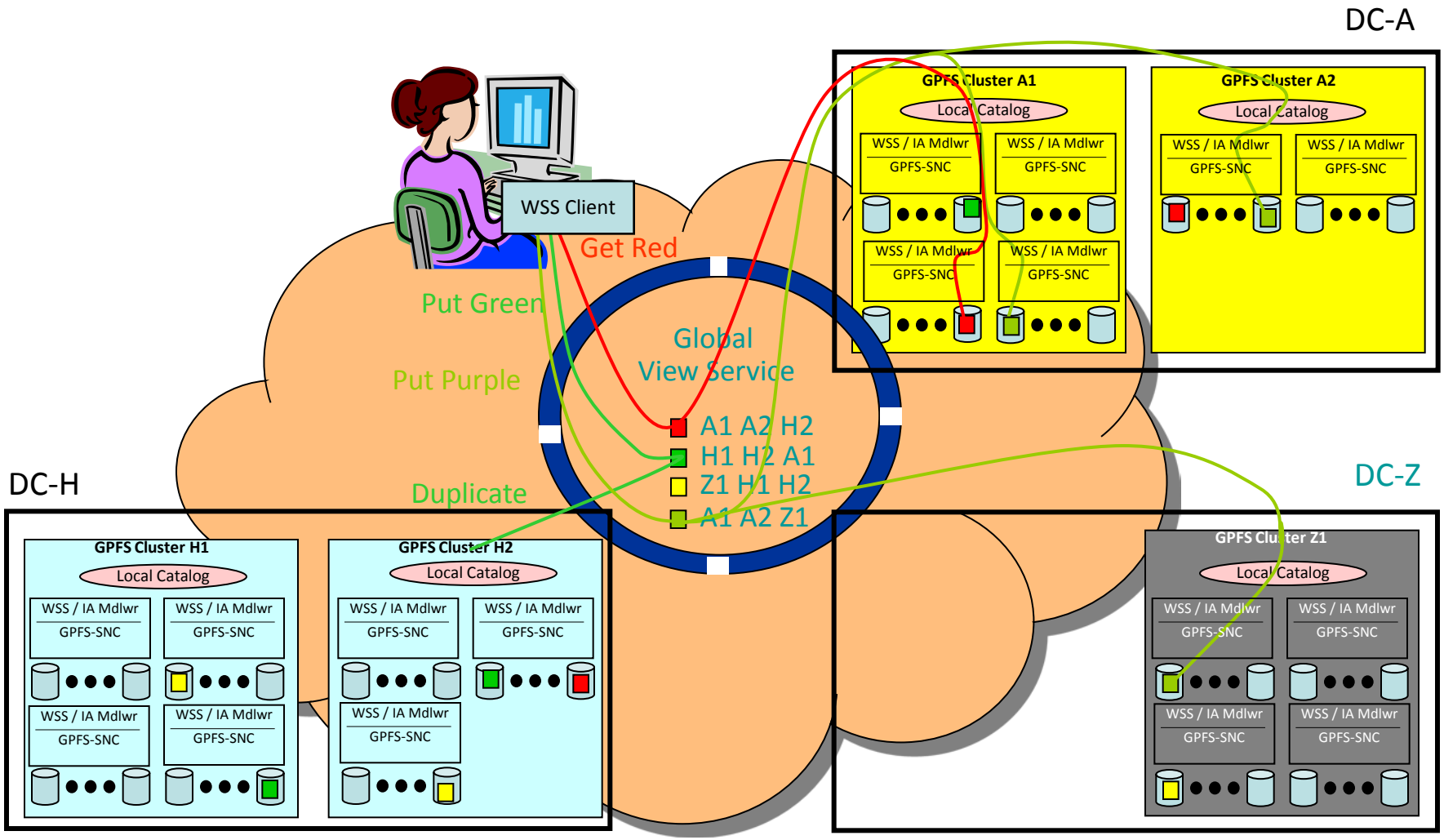
# Scaling VMWare with local disks





# FOCUS: Fixed content archive cloud





# Futures



- Scale and performance: 10,000+ nodes
- Recovery: Scale with number of nodes
- Dynamic caching and tiering: data where its needed
- Space Efficiency: Coding, compression and de-duplication
- Advanced workload isolation