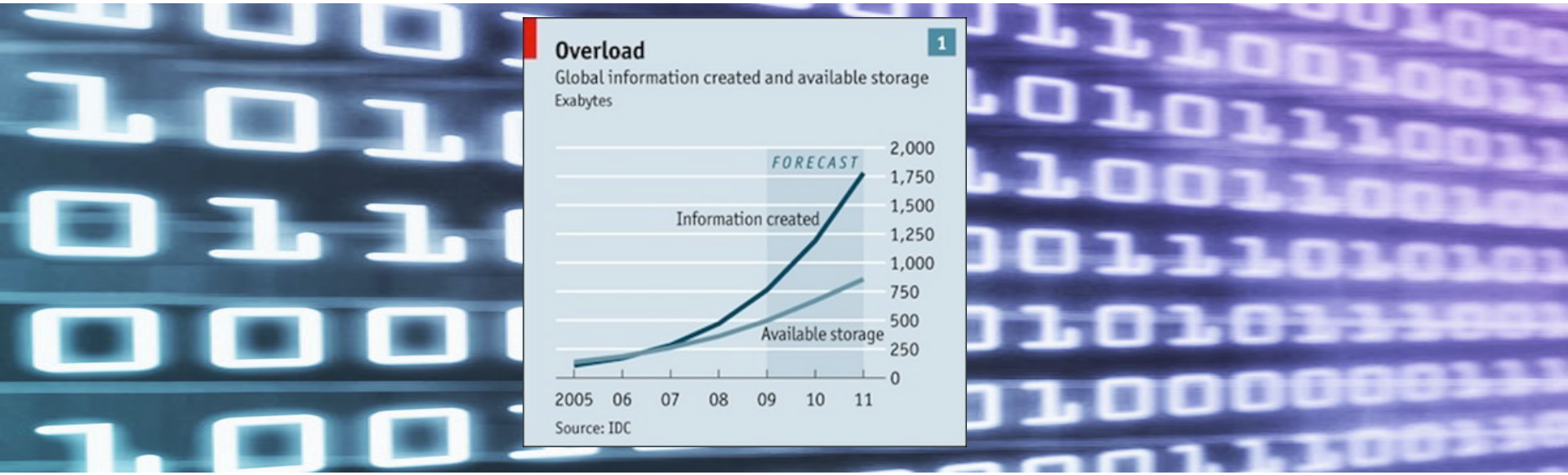


GPFS-FPO

A Cluster File System for Scale-Out Applications

Dinesh Subhraveti

IBM Research

GPFS-FPO: Motivation

Motivation:

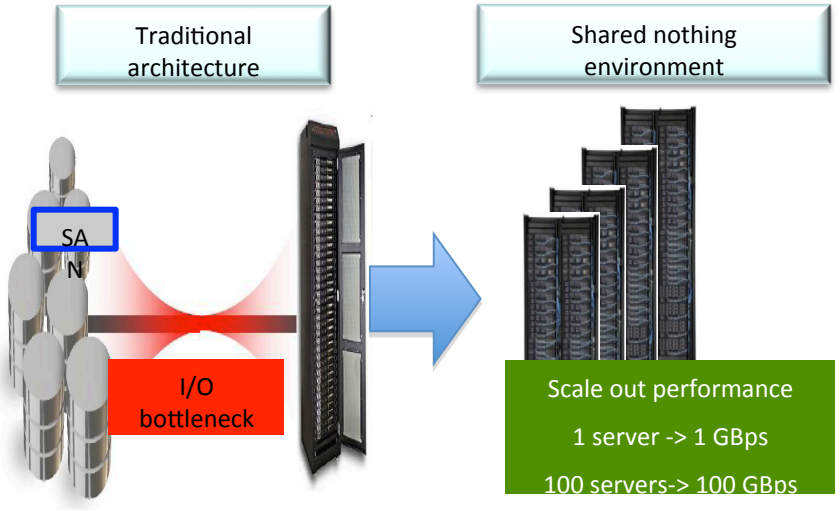
SANs are built for latency not bandwidth

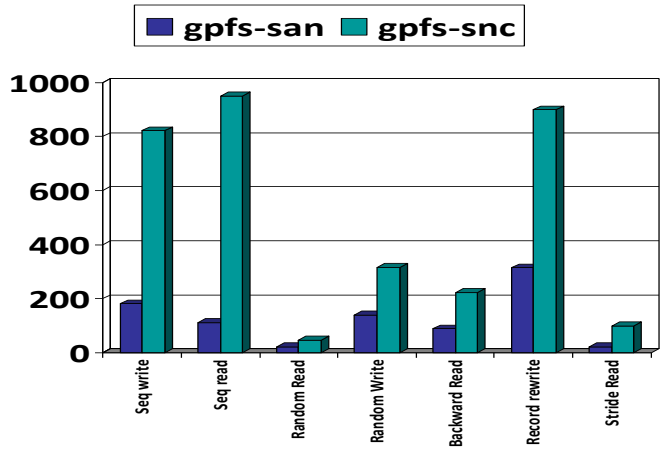
Advantages of using GPFS:

- 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)

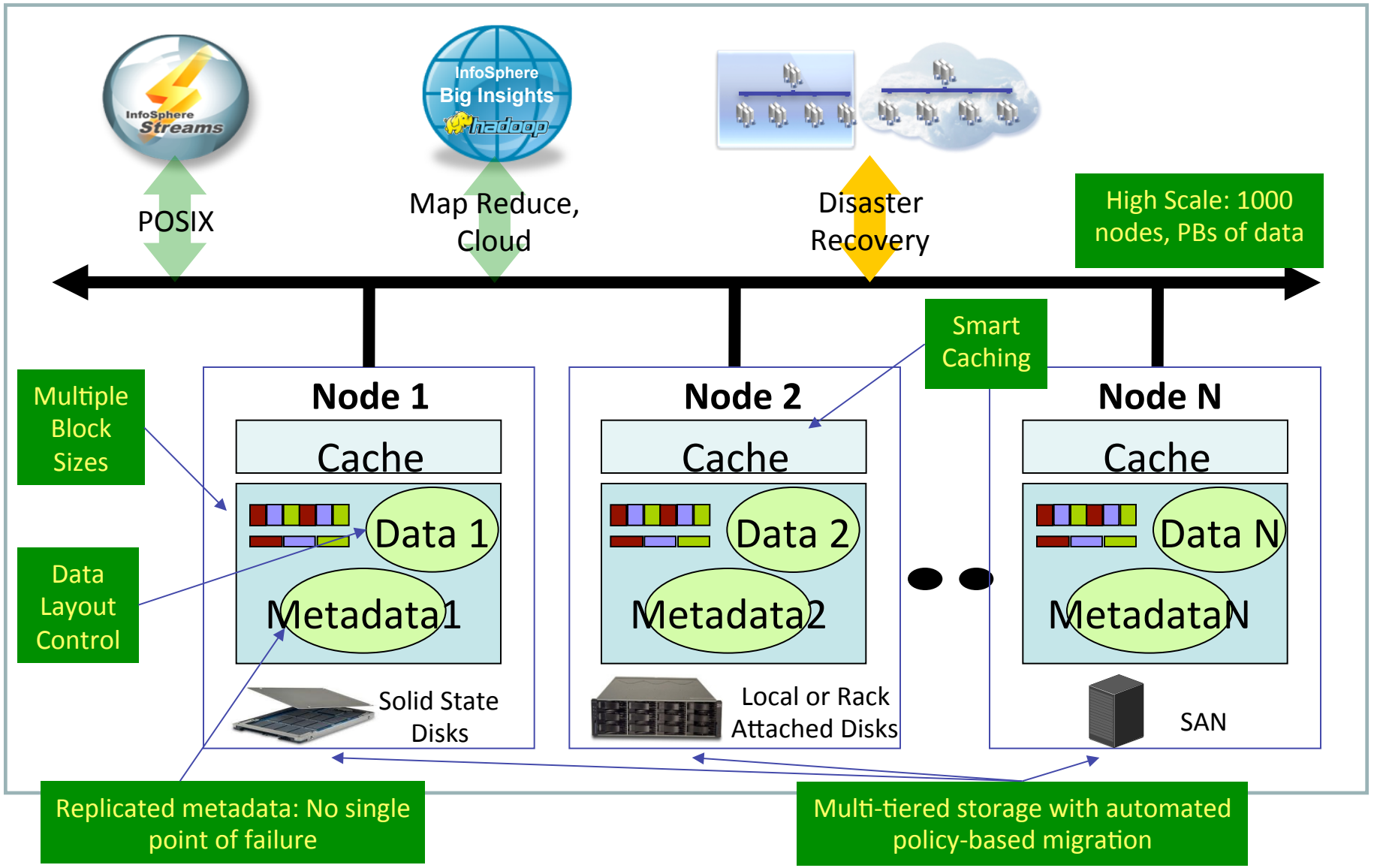
Challenges:

- Adapt GPFS to shared nothing clusters
- Maximize application performance relative to cost
- Failure is common, network is a bottleneck



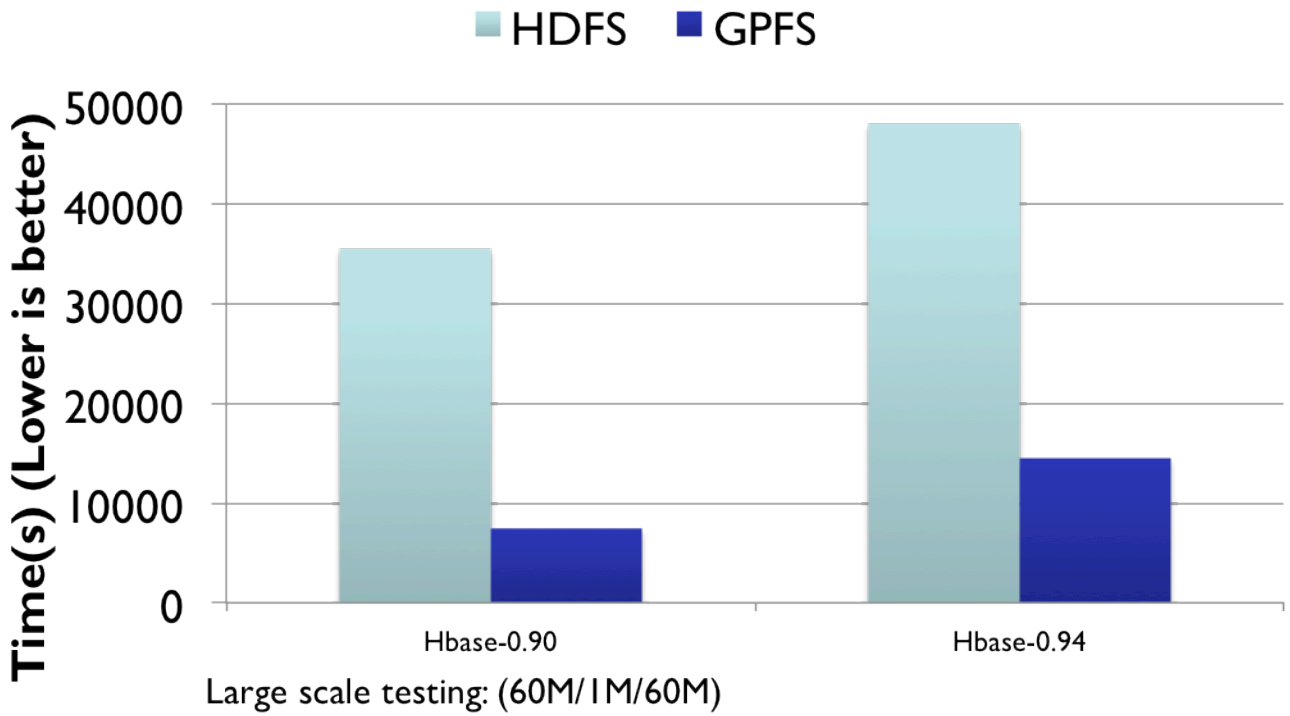


GPFS-FPO: Architecture



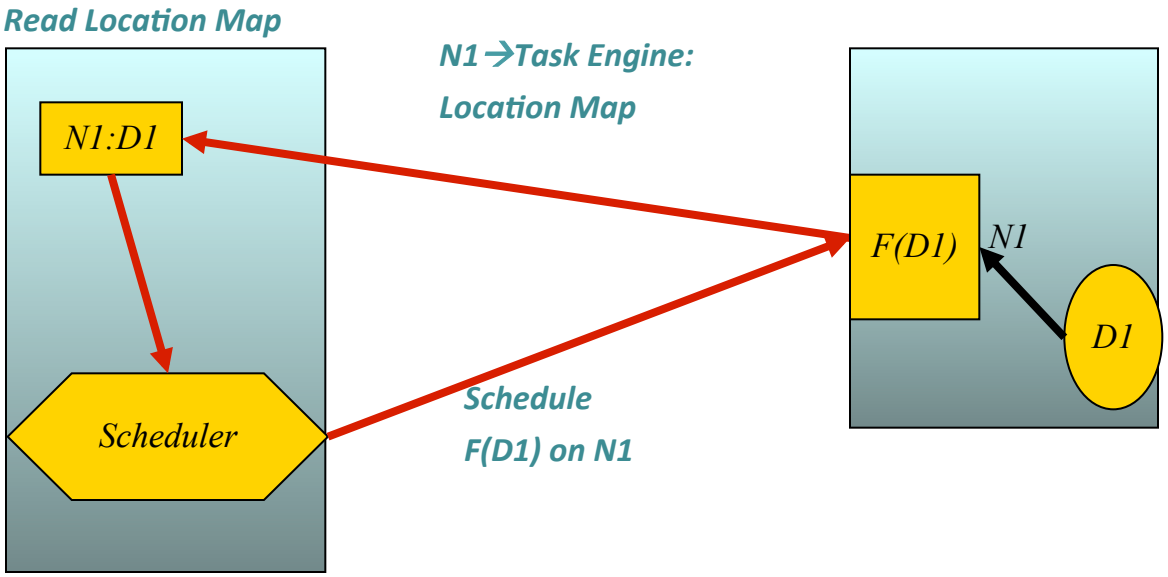
Excellent Performance

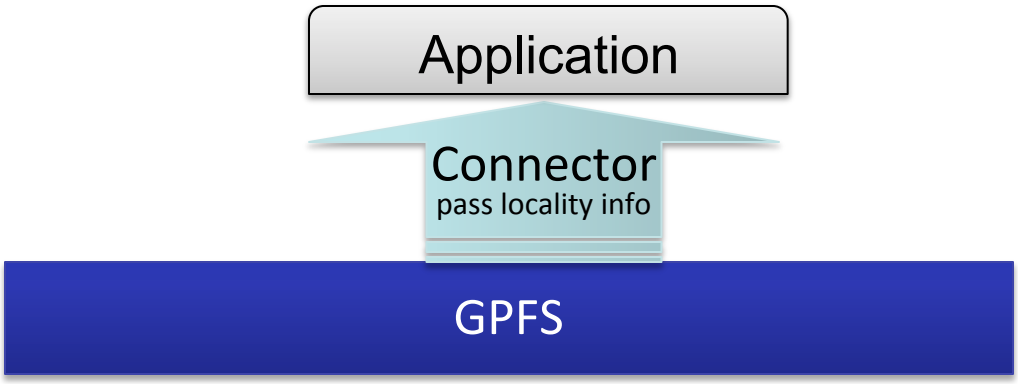
- Hardware Configuration:
  - 4 nodes, per node: 16 cores 96 GB 12 JBOD disks, 10 Gbps network, data disk cache enabled
- YCSB Workload A Configuration
  - YCSB: Region Heap Size 6 GB, Presplit 100, Record count 1M, threads=8, 4 region servers
  - HDFS: 1 metadata 3 data replicas, 48 data disks
  - GPFS: 3 metadata 3 data replicas, page pool 32 GB, system: 256KB block size, data pool 2M block size, data block group factor 128, 48 data disks
  - Hbase: major compaction disabled; minor compaction threshold 3, server threads 50/node
- Similar results for YCSB Workload B-F

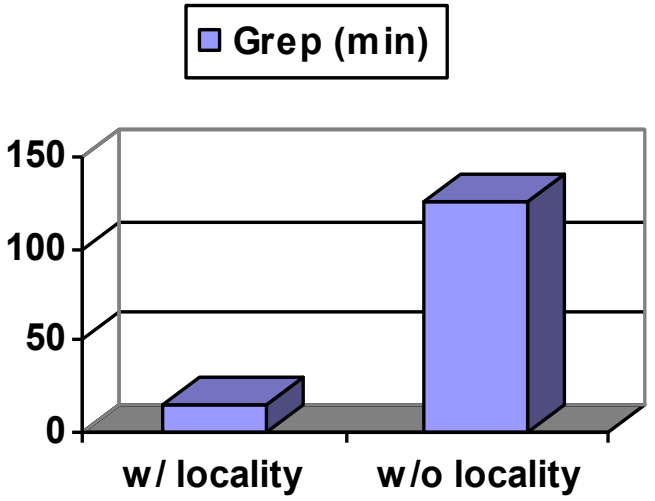


Locality Awareness: Make applications aware where data is located

- 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

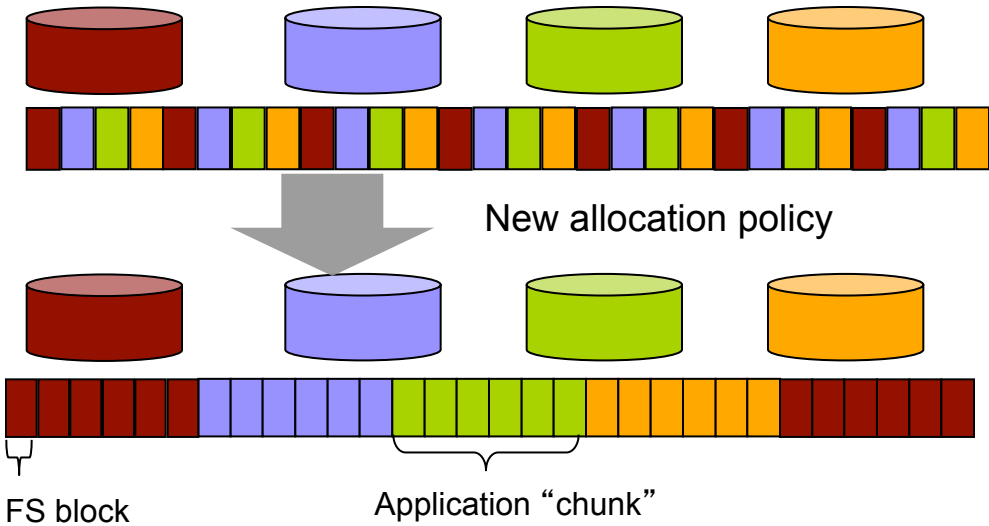






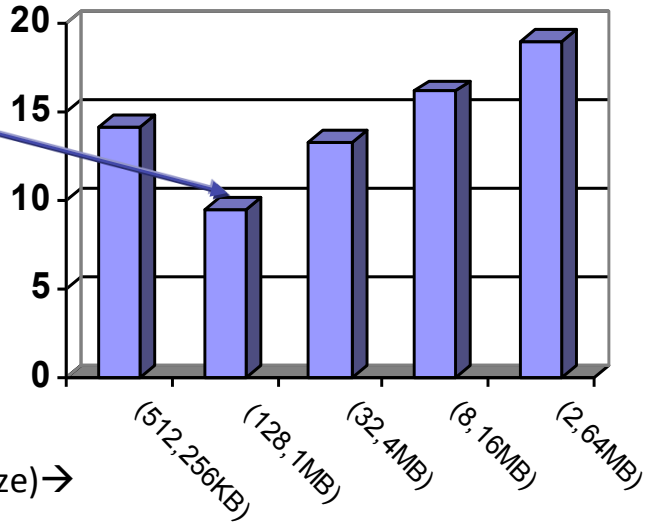
Chunks: allows applications to define own logical block size

- Workloads make two types of accesses
  - 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
  - Block group factor for block size
  - Effective block size = block size * block group factor
- File blocks are laid out based on effective block size



Grep (min)

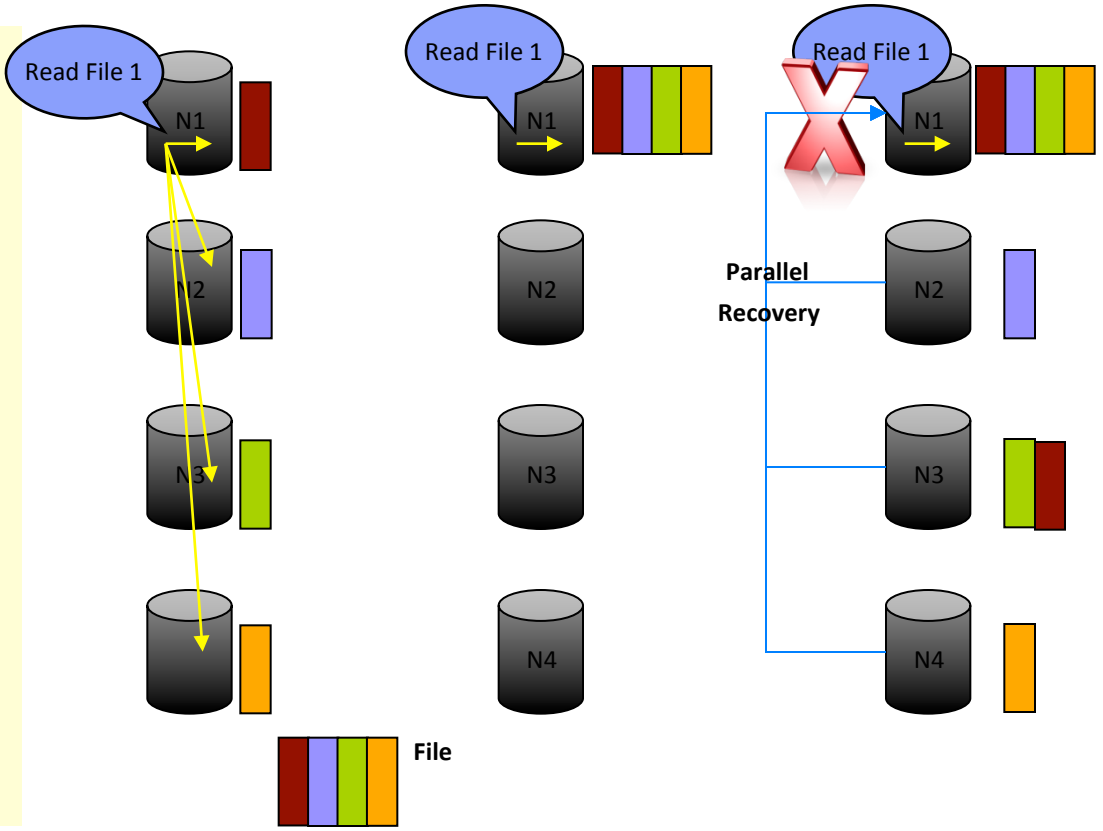
Optimum Block Size: Caching and pre-fetching effects at larger block sizes

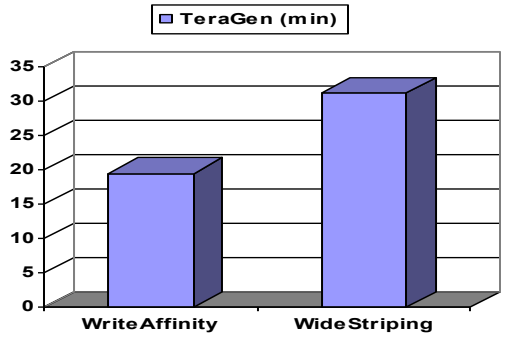


Effective block size: 128 MB

Write Affinity: Allow applications to dictate layout

- 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 file or a set of files on a given node
  - All relevant files will be allocated on the node, near a node or in the same rack
- Configurability of replicas
  - Each replica can be configured for either wide striping or write affinity





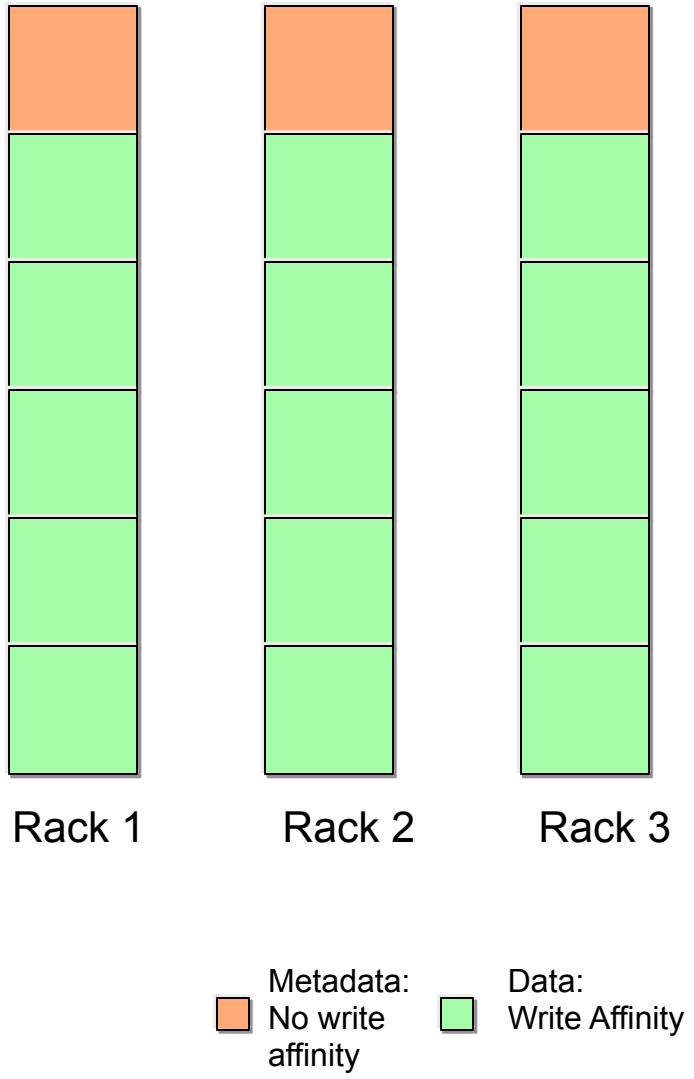
Wide striped layout: Read over local disk + network

Write affinity layout: Read over only local disk

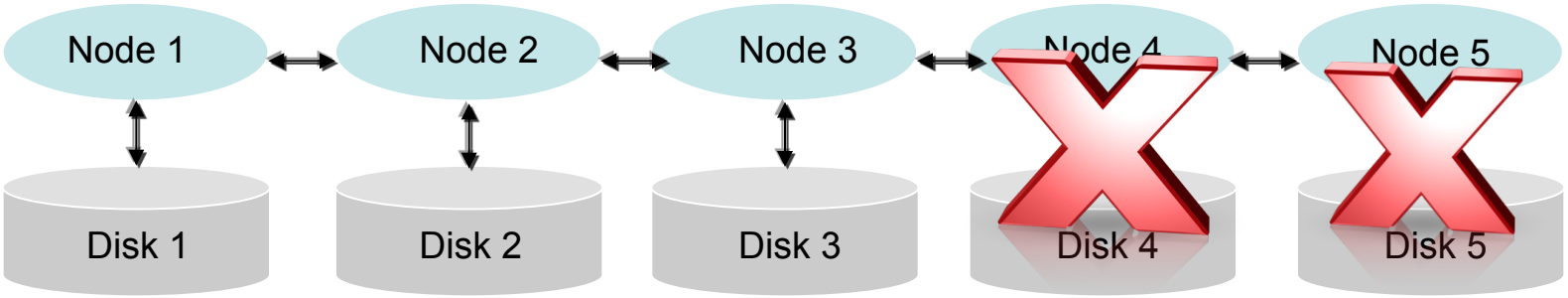
Write affinity and wide striping with replication: Read over only local disk, recover from wide striped replica

Hybrid allocation: treat metadata and data differently

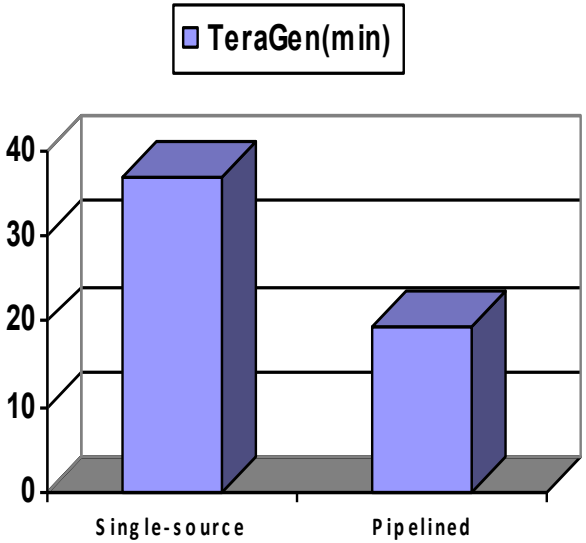
- Observation
  - Data is suited for GPFS-FPO
  - Metadata is suited for regular GPFS
- Make allocation type a storage pool property
  - Metadata pool: no write affinity
  - Data pools: write affinity
- GPFS-FPO: distributed metadata
- No single point of failure but too many metadata nodes
  - Increases probability of any 3 metadata nodes going down → loss of data
  - Assumes replication=3
- Typical metadata profile in GPFS-FPO
  - One or two metadata nodes per rack (failure domain)
  - Random access patterns

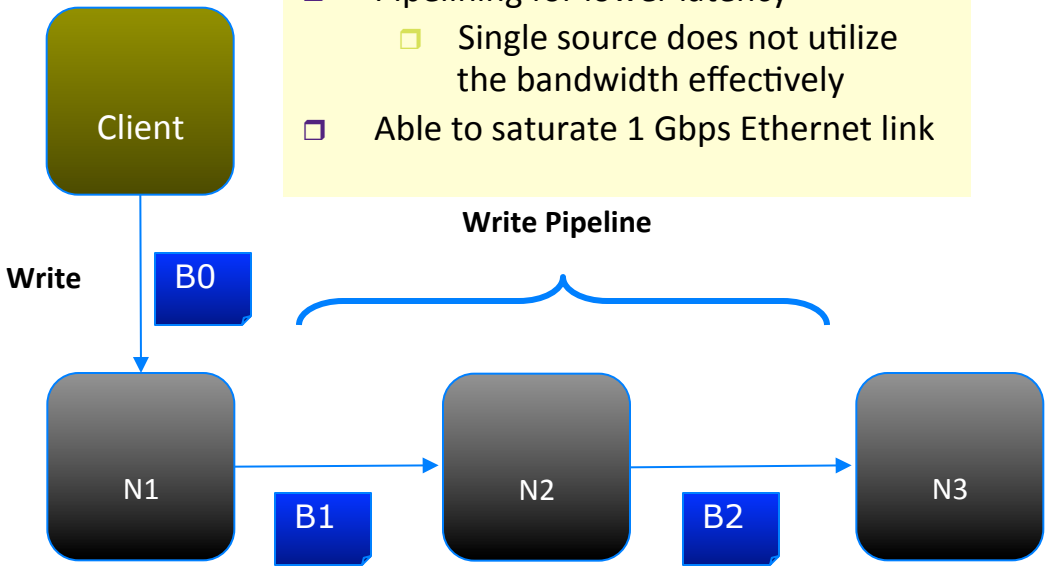


Pipelined Replication: Efficient replication of data



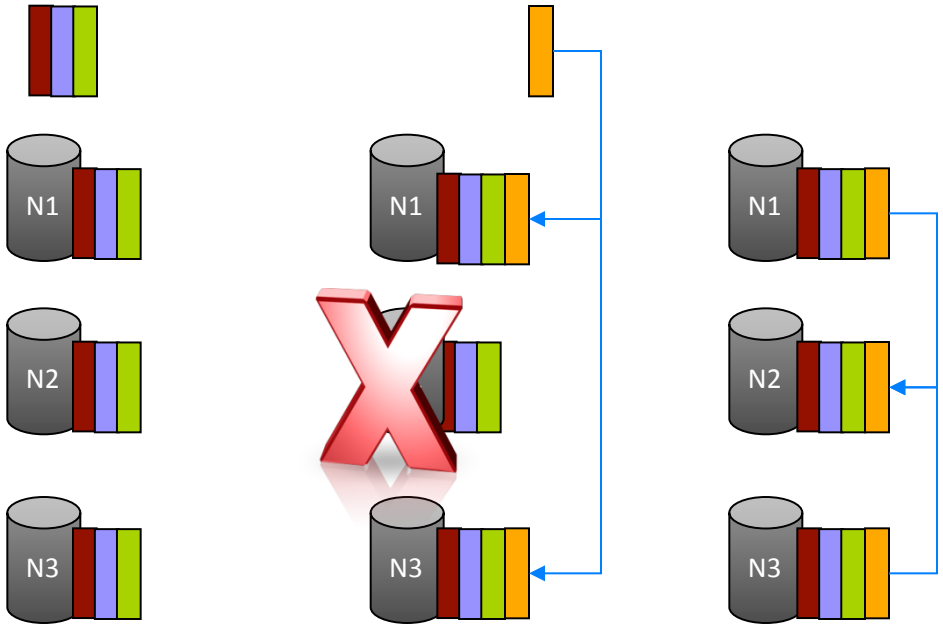
- Higher Degree of replication
- Pipelining for lower latency
  - Single source does not utilize the bandwidth effectively
- Able to saturate 1 Gbps Ethernet link





Fast Recovery: Recover quickly from disk failures

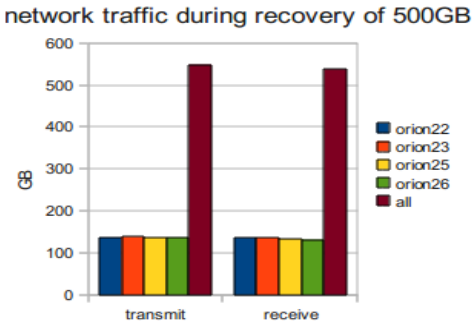
- Handling failures is policy driven
  - Incorporated node failure and disk failure triggers
  - Policy-based recovery
    - Restripe on a node failure or disk failure
    - Alternatively, rebuild the disk when the disk is replaced
- Fast recovery
  - Incremental recovery
    - Keep track of changes 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

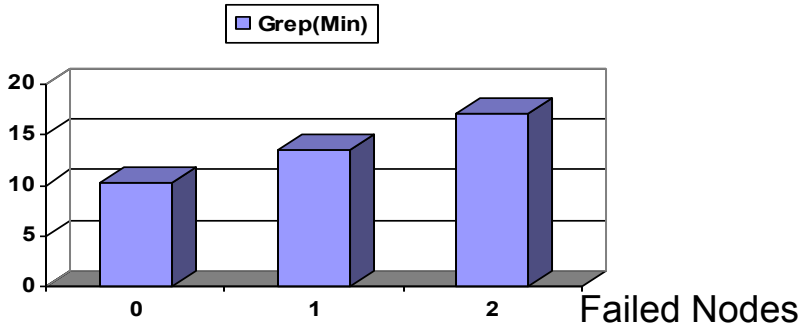


Well replicated file

N2 rebooted:
Writes go to N1 and N3

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





Comparison with HDFS and MapR

File System	GPFS-FPO	HDFS	MapR
Robust	No single point of failure	NameNode vulnerability	No single point of failure
Data Integrity	High	Evidence of data loss	New file system
Scale	Thousands of nodes	Thousands of nodes	Hundreds of nodes
POSIX Compliance	Full – supports a wide range of applications	Limited	Limited
Data Management	Security, Backup, Replication	Limited	Limited
MapReduce Performance	Good	Good	Good
Workload Isolation	Supports disk isolation	No support	Limited support
Traditional Application Performance	Good	Poor performance with random reads and writes	Limited

RoadMap

- 4Q12 : GA
- 1H13 :
  - WAN Caching
  - Advanced Write Affinity
  - KVM Support
- 1Q14:
  - Integrate with Platform Computing Scheduler: DirectShuffle
  - Encryption
  - Disaster Recovery

http://almaden.ibm.com