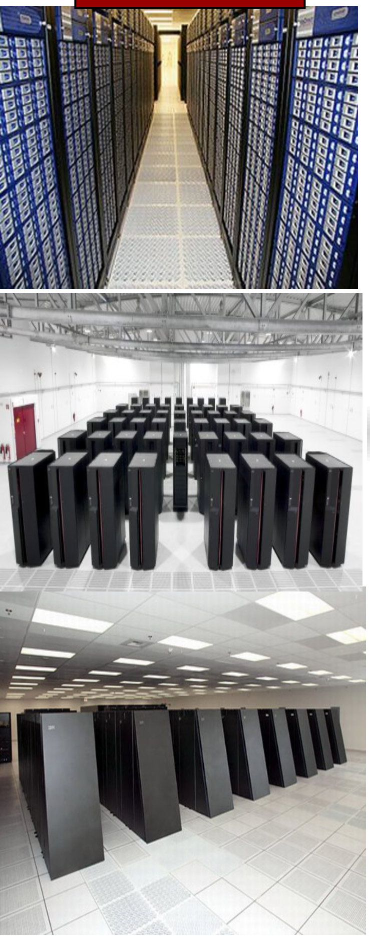
Panache: A Parallel Filesystem Cache for Global file Access

Renu Tewari

(with Dean Hildebrand, Marc Eshel, Manoj Naik, Frank Schmuck, Roger Haskin)

IBM Almaden

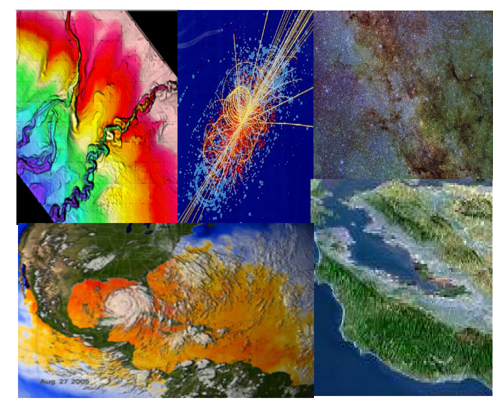
Compute Farms



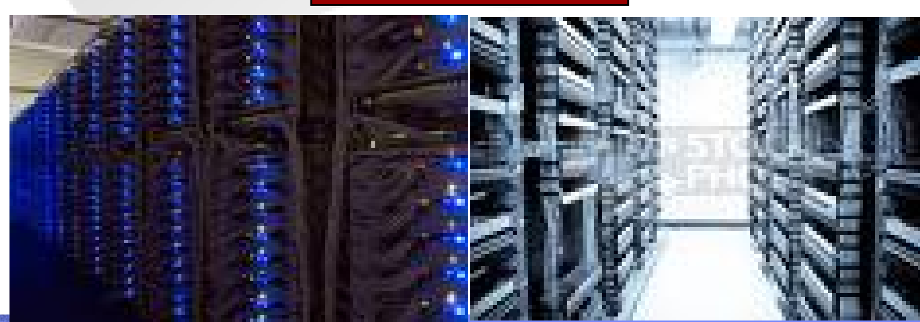
Data Source

Global File Access

Clients



Data Centers



Characteristics

- **High latency links**
  - ~150 ms.
- **Bandwidth across data centers is decent**
  - Across data centers OC-48/192
  - Teragrid 10-30Gb/s
- **WAN Network is not reliable**
  - Multiple exchange points, outages, packet loss
- **Predominantly large files**
  - Virtual machine images, application virtual disks
  - Large satellite images
  - Youtube videos
- **Global access but local speeds**

Global performance to match local?

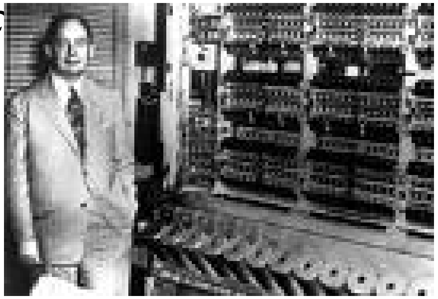
With 25 years of Internet experience,
we've learned exactly one way to deal with exponential growth:

... Caching.

Data has to find 'local' sources near consumers
rather than always coming from the place it was originally
produced -- Van Jacobson, 1995

A. W. Burks, H. Goldstine, John von Neumann "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", 1946.

> *"Ideally one would desire an indefinitely large memory capacity such that any particular word would be immediately available.*
>
> *We are forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly acc*

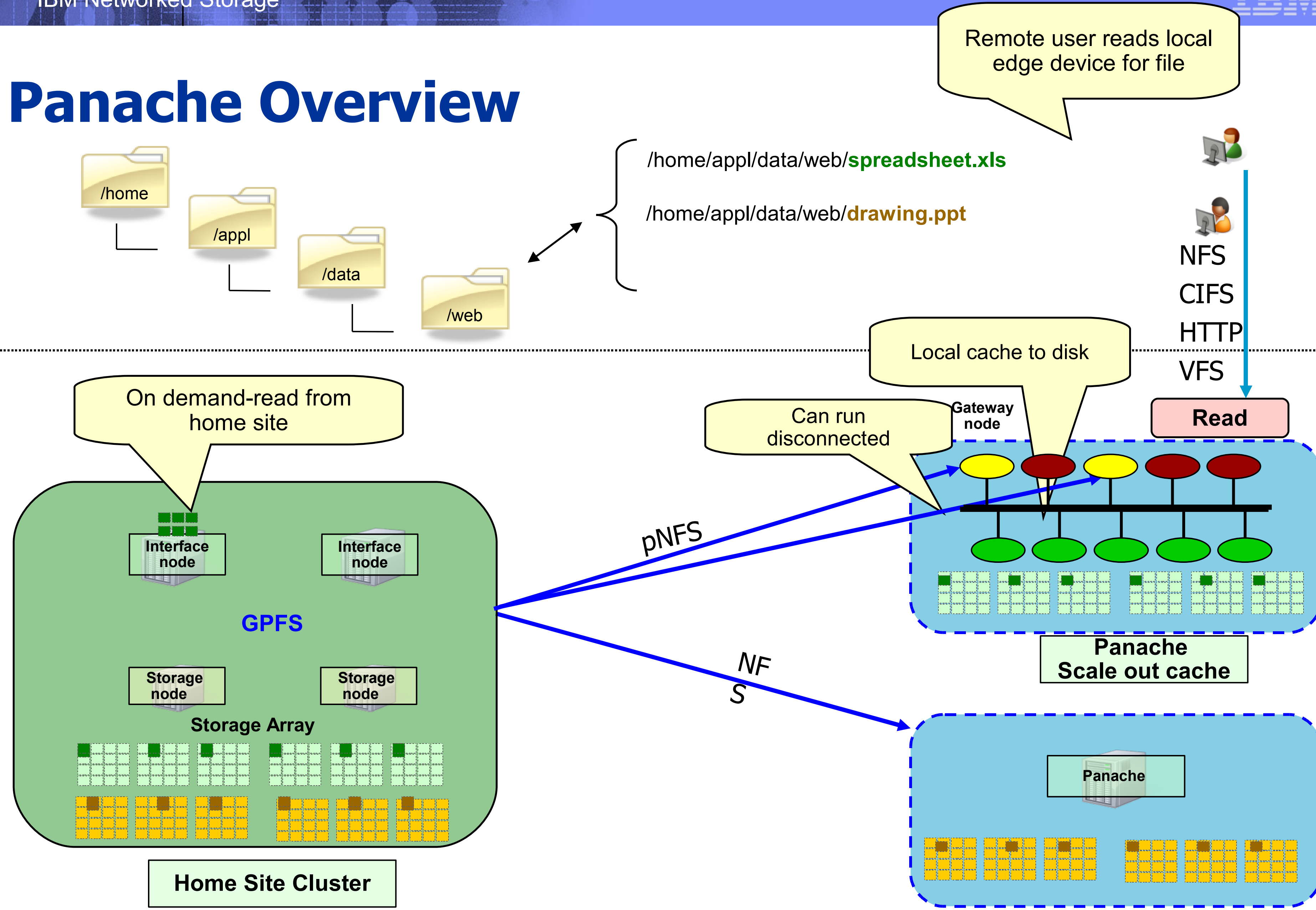


Cache is not only 'local' but also scalable

Outline

1	Panache Architecture
2	pNFS and parallel Reads
3	Asynchronous updates
4	Dependent Metadata operations
5	Namespace Caching
6	Summary and Conclusions

Panache Overview



How did that work

- Independent filesystems
- Separate inode space
- Home FS is unchanged
- Cache is a standalone clustered FS

Inode: 100

Inode attrs:

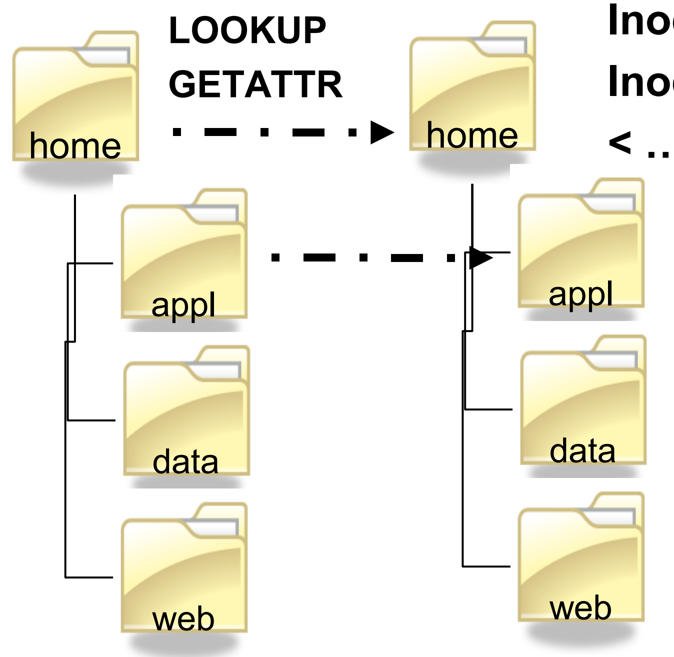
< … >

Remote state:

<id: 1024

attrs: mtime, ctime

>



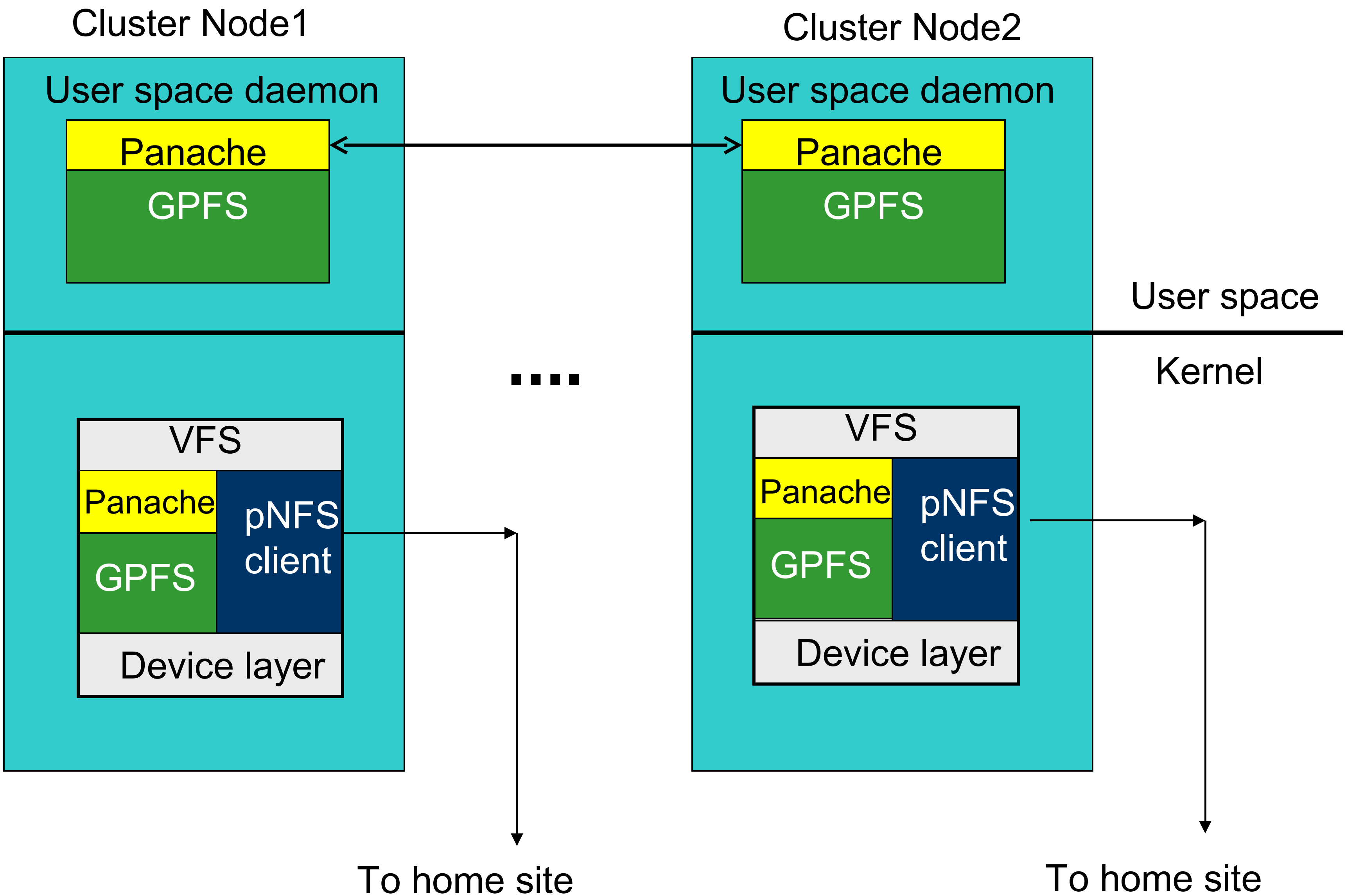
Inode 1024

Inode attrs:

< … >

Cache directory tree **Home directory tree**

Panache Internal Architecture



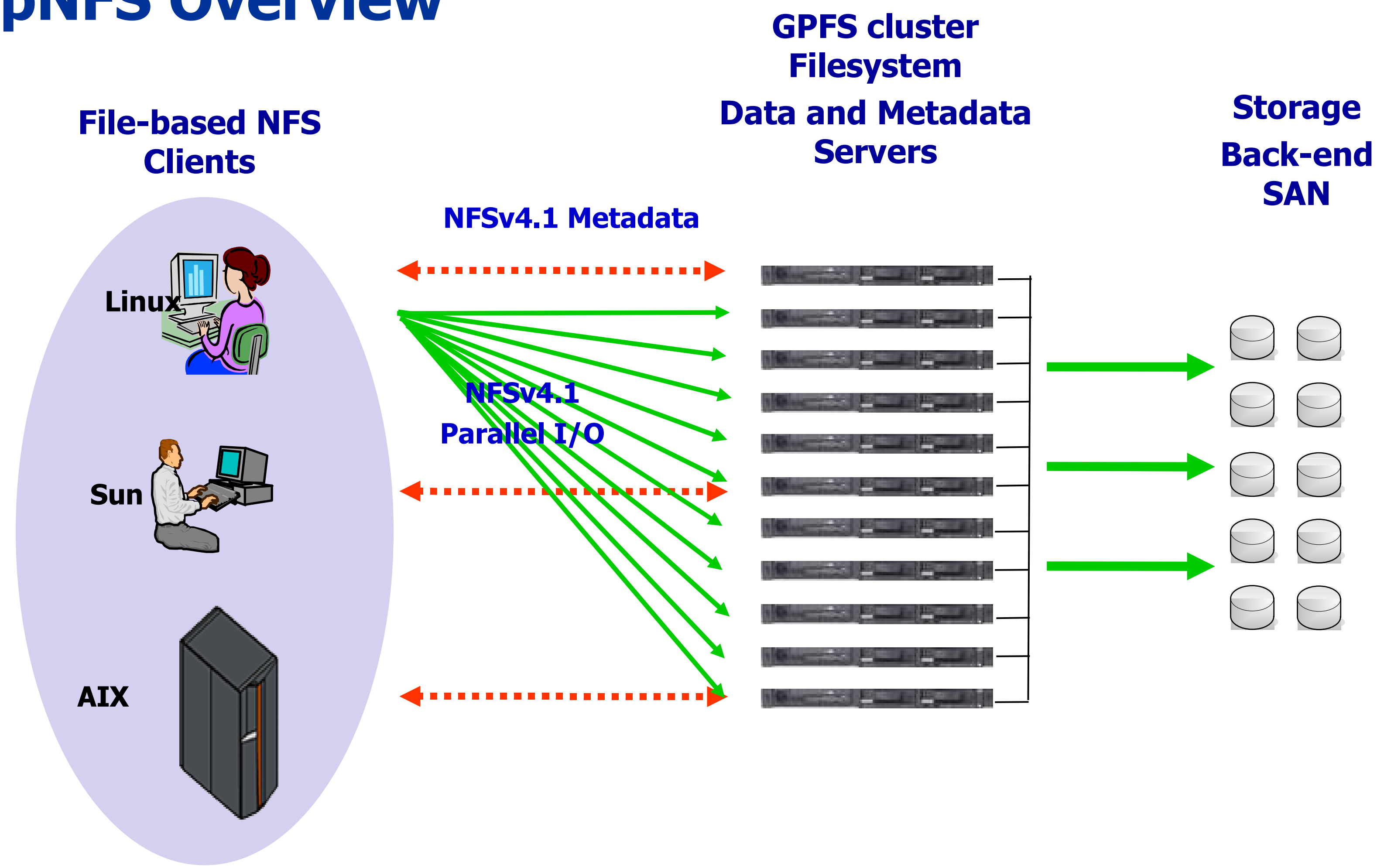
Key Features

- **Miss throughput at WAN bandwidth**
- **Hit throughput matches local access**
- **Writes and metadata updates match local speeds**
  - Asynchronous write back
  - Assume high latency to disconnected network by design
- **All operations are parallel**
  - Parallel ingest
  - Parallel access
  - Parallel update
  - Parallel data write-back
  - Parallel metadata write-back (non dependent operations)

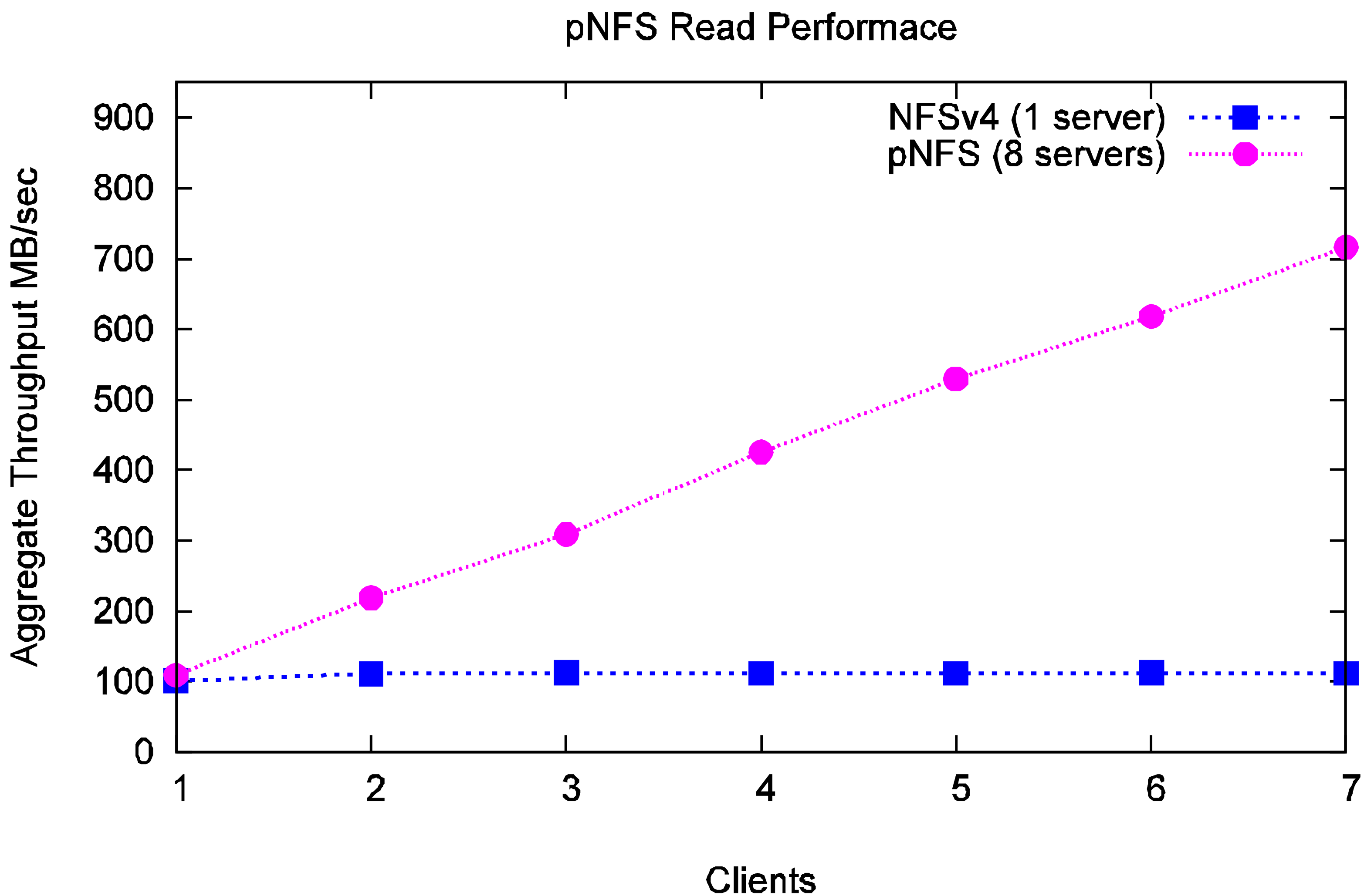
Outline

1	Panache Architecture
2	pNFS and parallel Reads
3	Asynchronous updates
4	Dependent Metadata operations
5	Namespace Caching
6	Summary and Conclusions

pNFS Overview

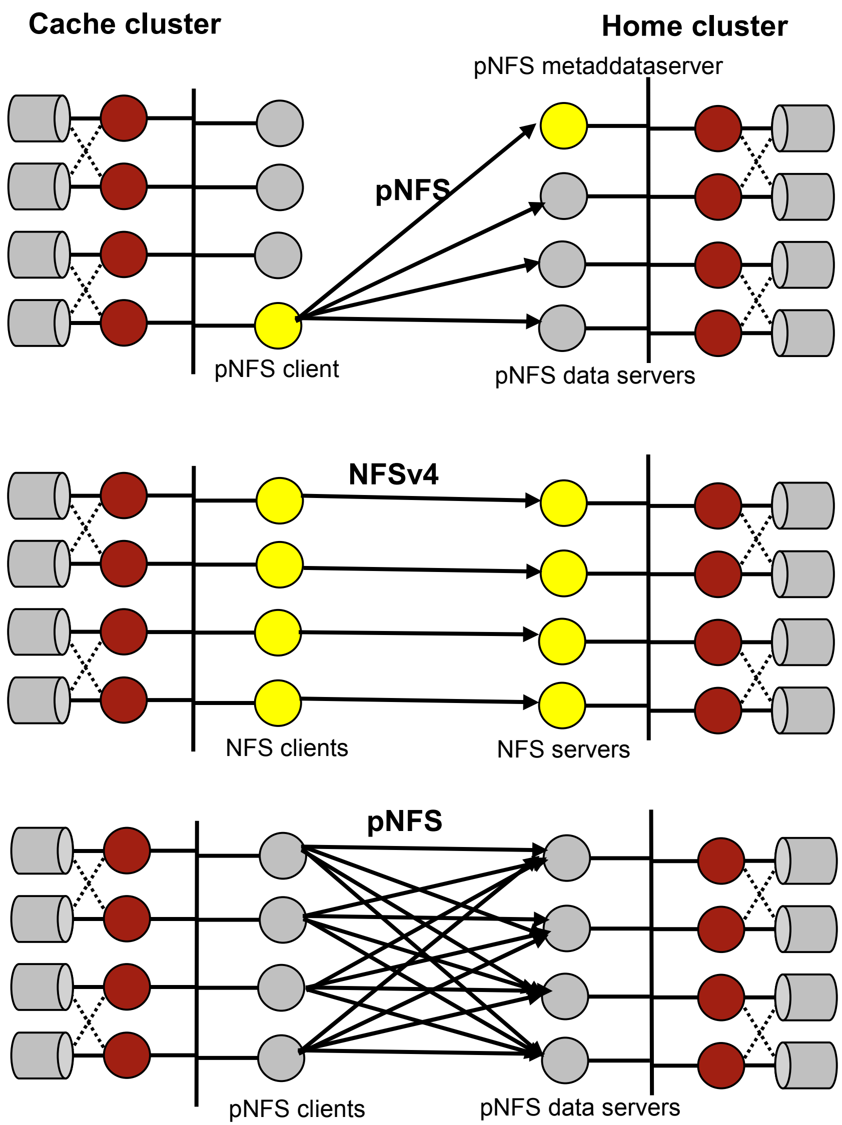


Why use pNFS for Data Transfer

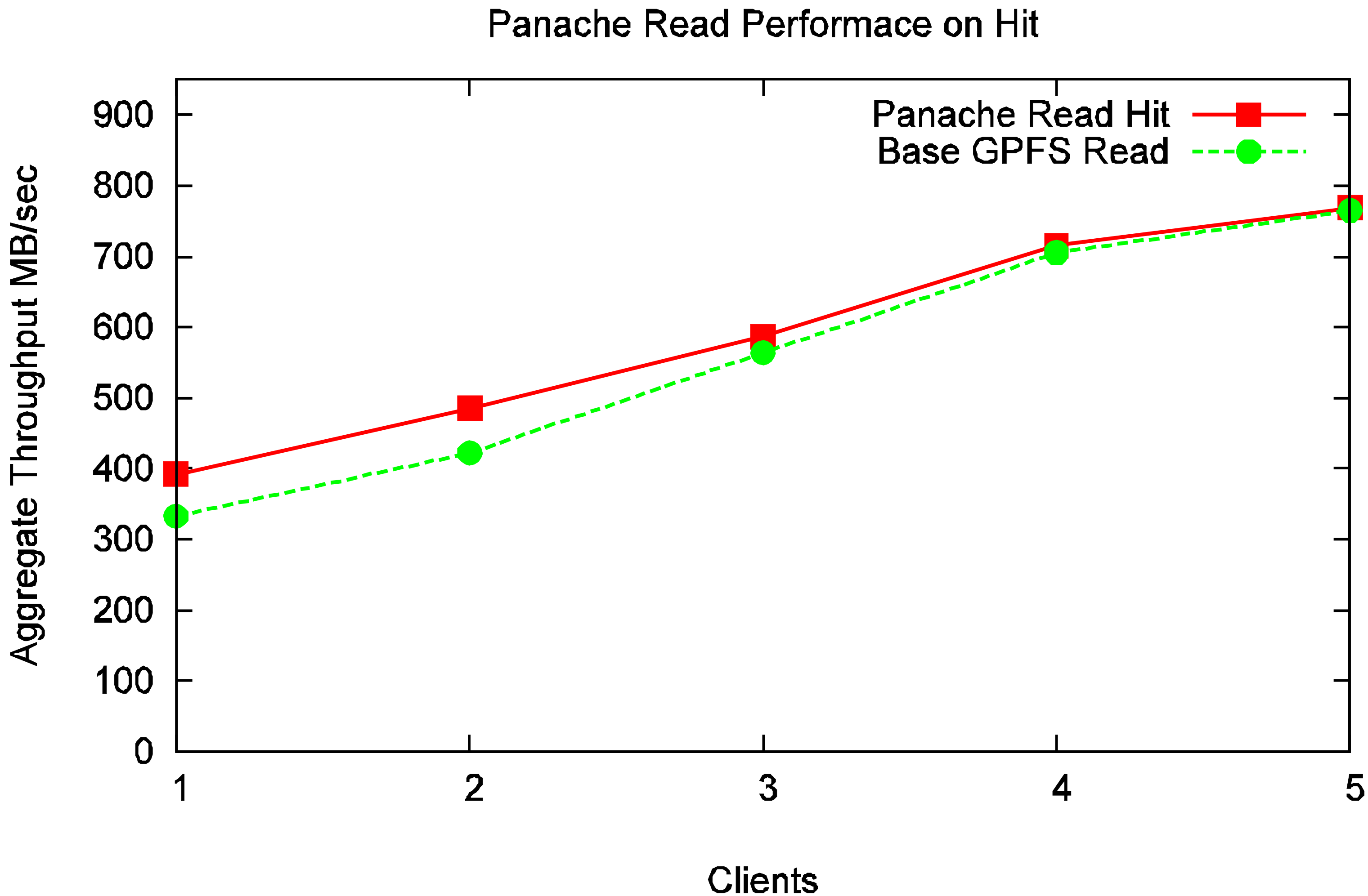


ior reads 8GB files, 8 node server cluster completely saturates the network

Parallel Ingest Options

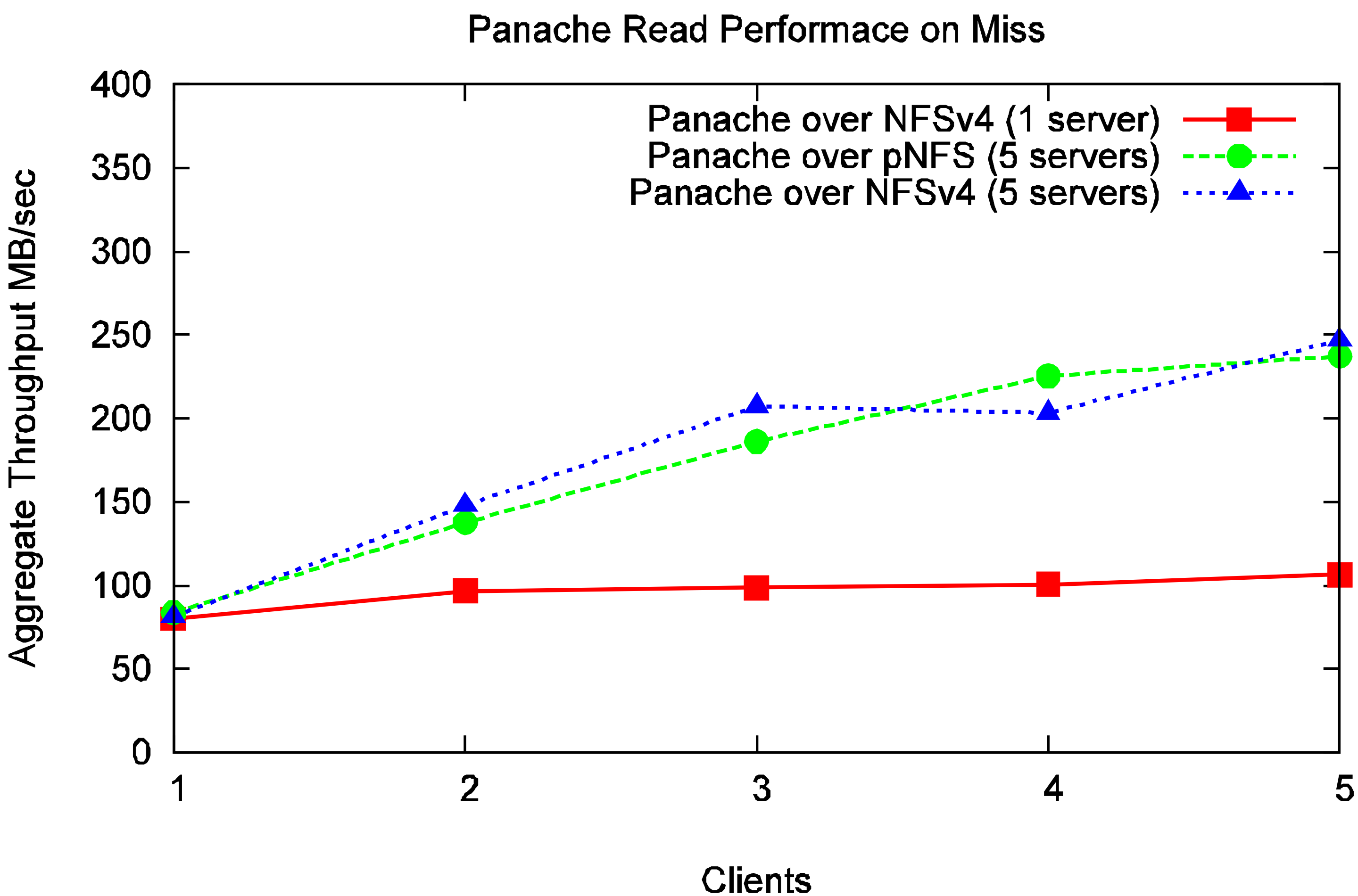


Cache Hit Performance



ior reads 8GB files, 10 node cache cluster (5 app + 5 gw), hits match local access

Cache Miss Performance

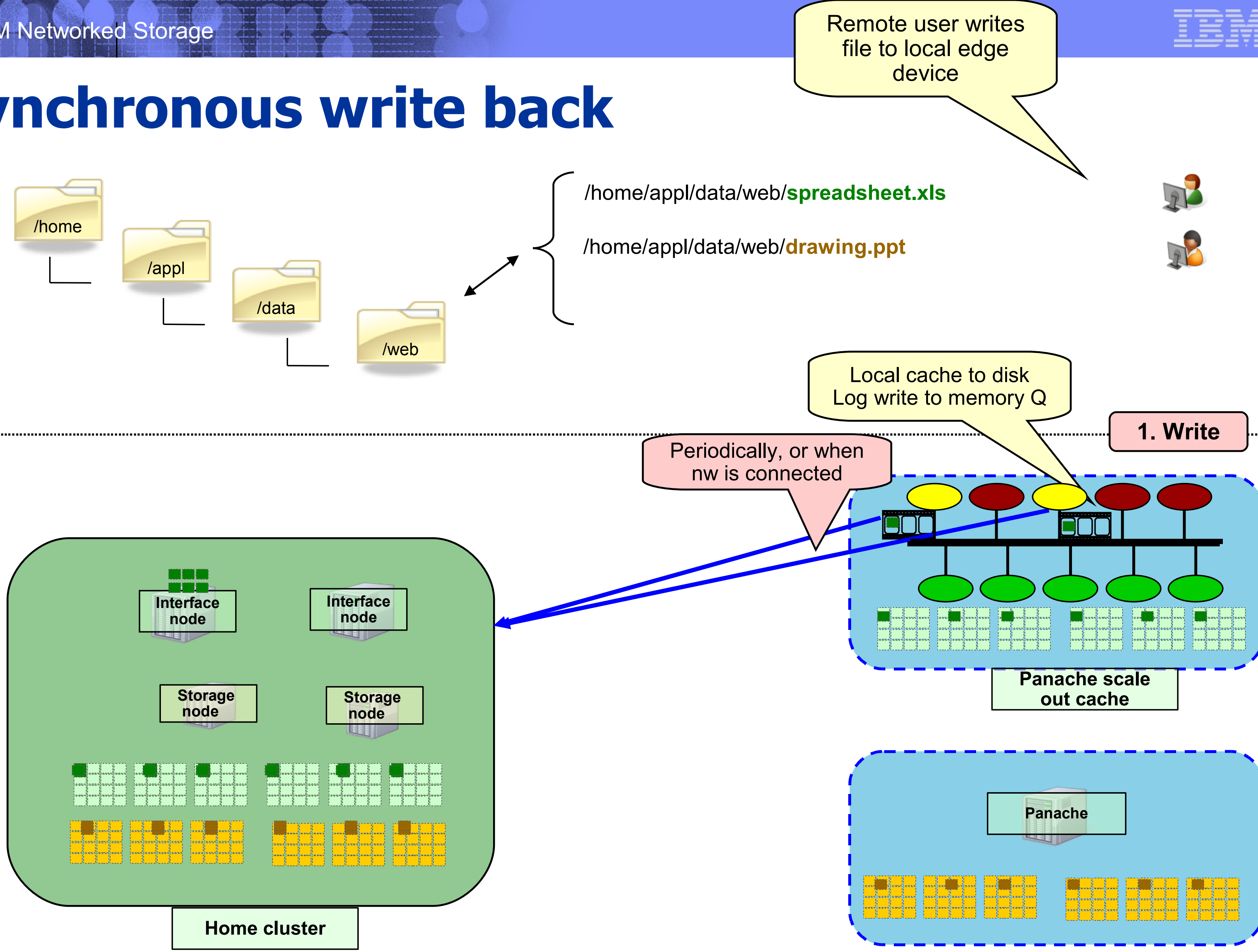


ior reads 8GB files, 10 node cache cluster (5 app + 5 gw), 5 node remote cluster
miss limited by network bandwidth

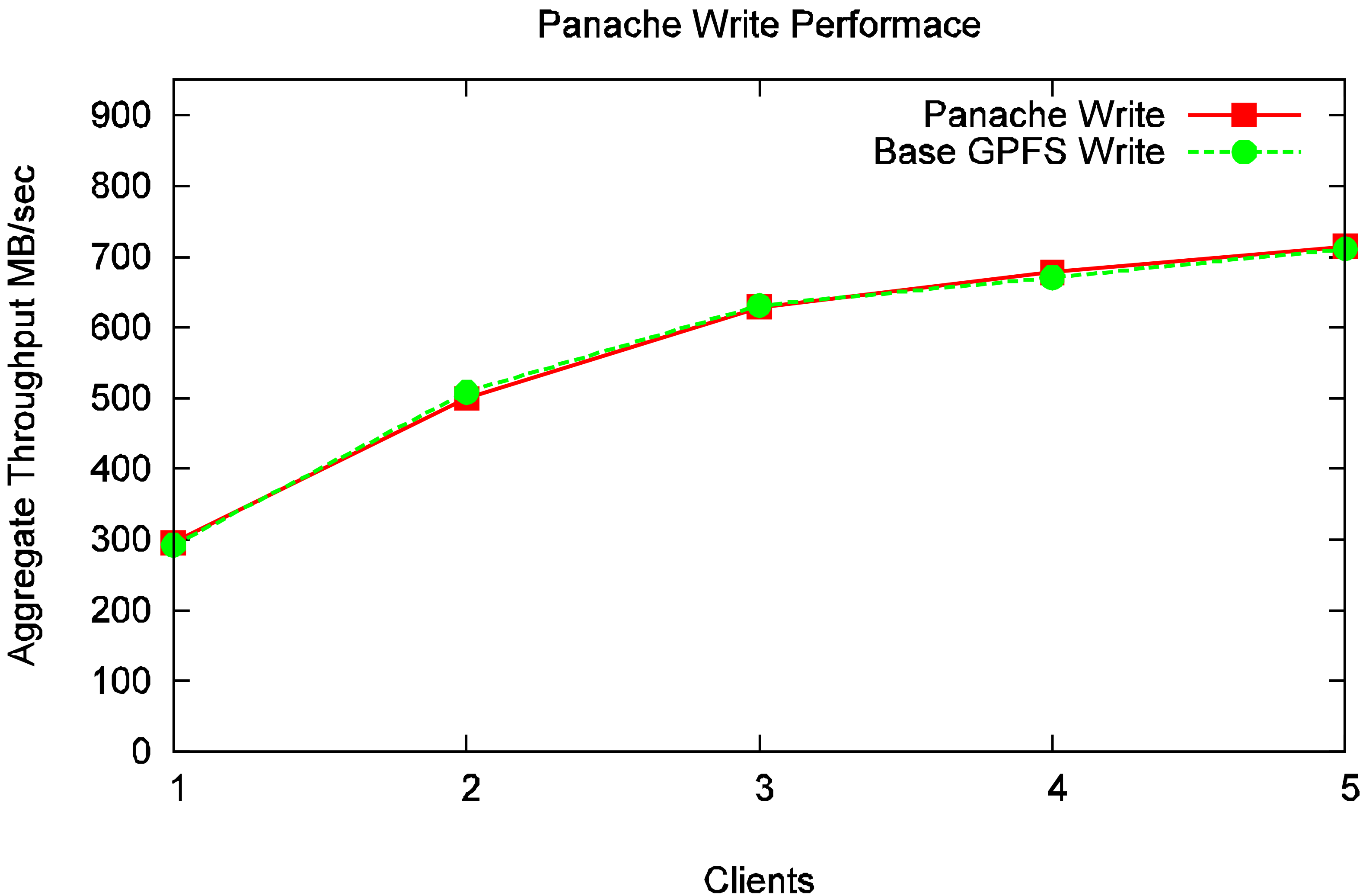
Outline

1	Panache Architecture
2	pNFS and parallel Reads
3	Asynchronous updates
4	Dependent Metadata operations
5	Namespace Caching
6	Summary and Conclusions

Asynchronous write back



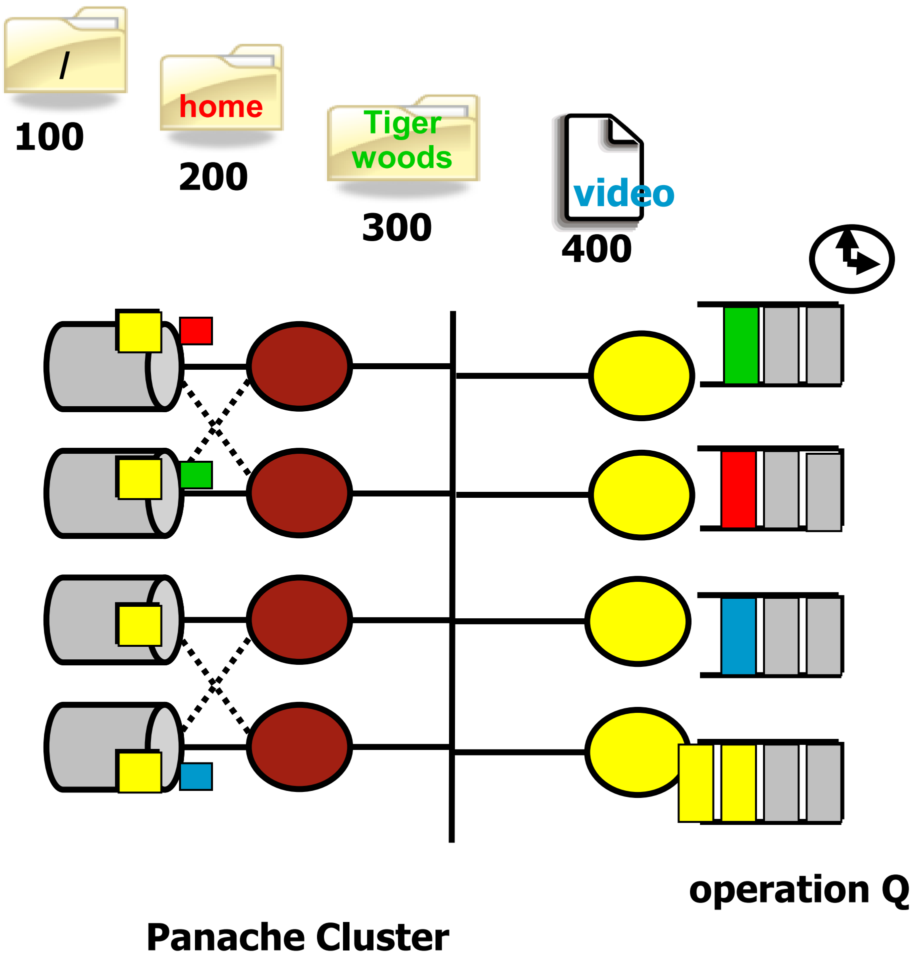
Cache Write Performance

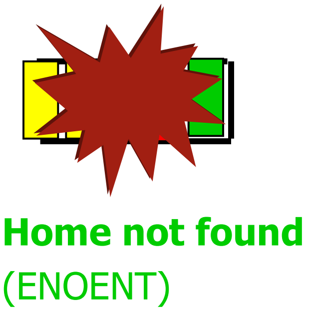


ior writes 8GB files, 10 node cache cluster (5app+5gw), writes match local access

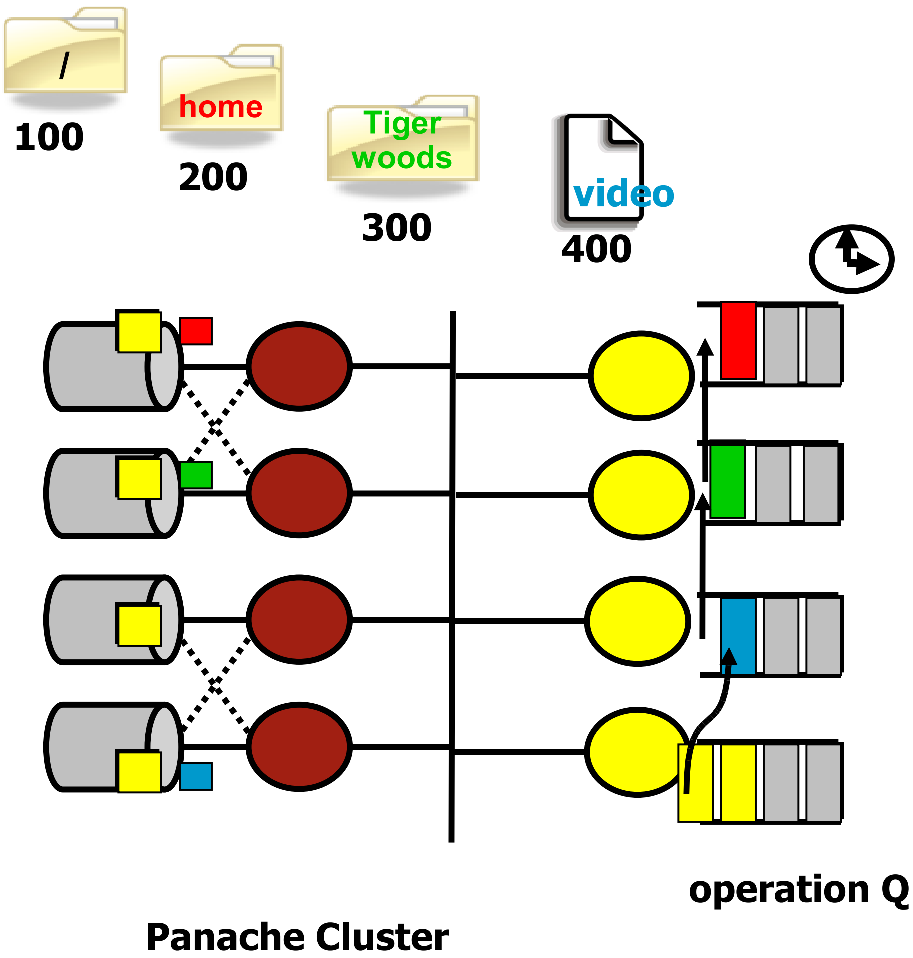
Dependent Metadata Operations

1	mkdir 100, home
2	mkdir 200, Tigerwoods
3	create 300, video
4	write 400, 0, 4M

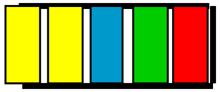




Dependent Metadata Operations with Token Chaining

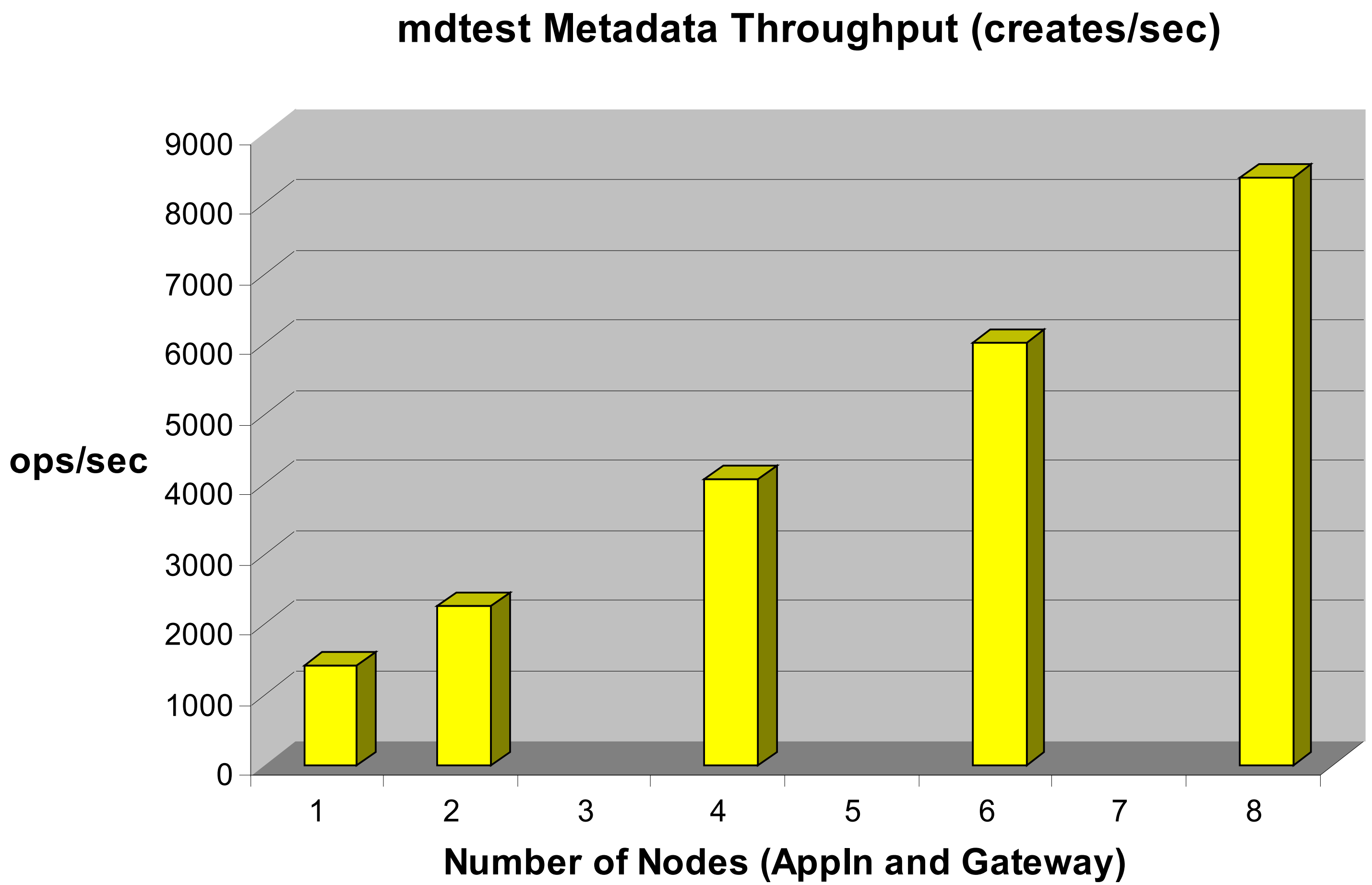


1	mkdir 100, home
2	mkdir 200, Tigerwoods
3	create 300, video
4	write 400, 0, 4M



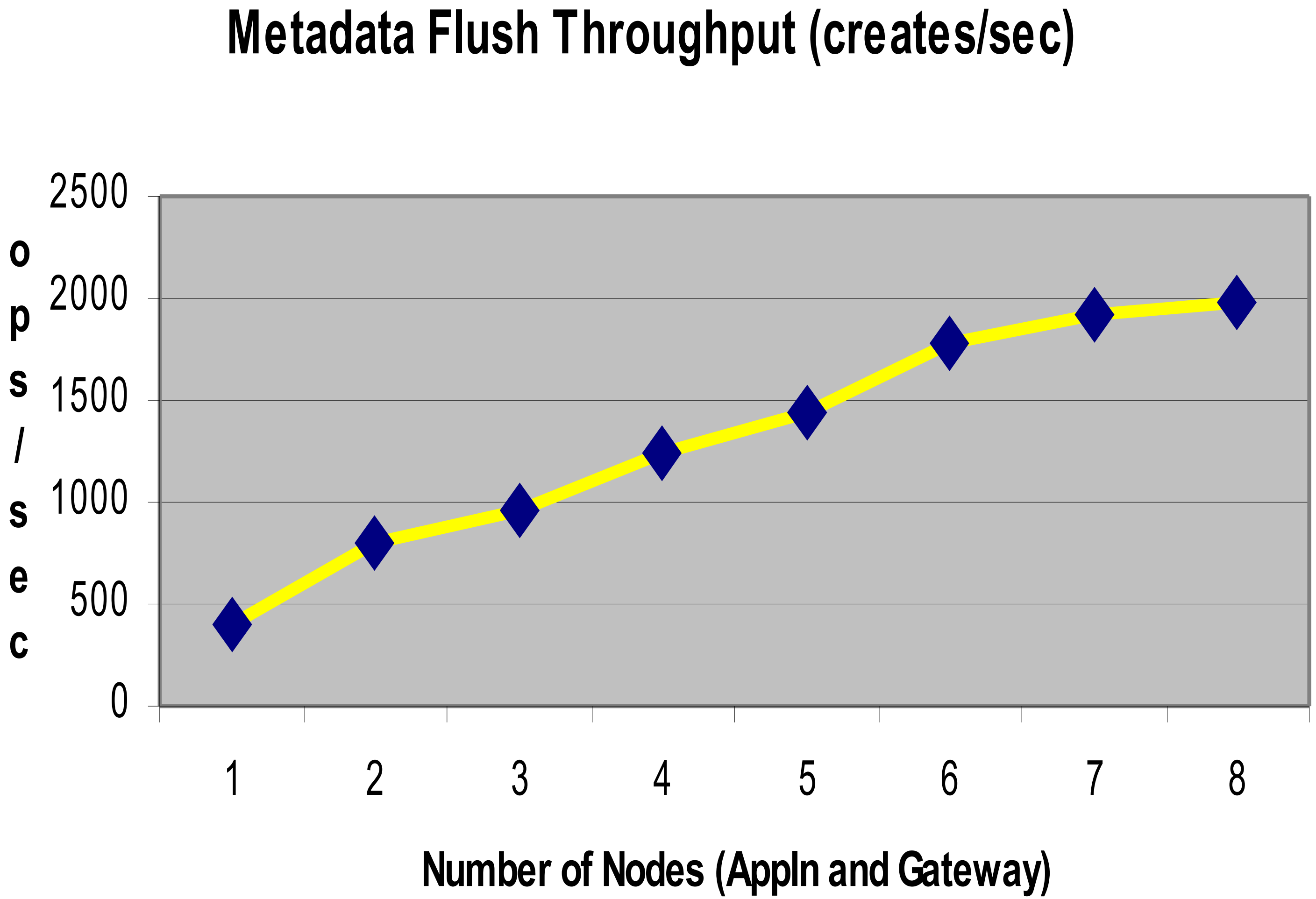
Rename, hard links cross multiple inodes

Parallel Metadata Throughput



mdtest 1000 creates/node, remote cluster size grows in tandem with gw nodes of cache cluster

Metadata Flush Throughput



mdtest 1000 creates/node, remote cluster size grows in tandem with gw nodes of cache cluster

Consistency Definitions

- **Local consistency**
  - Read from a node of the cache cluster returns the last write from any node of the cache cluster
- **Validity Lag** δ
  - Time delay between a read at the cache site reflecting the last write at the remote site
- **Synchronization Lag** μ
  - Time delay between a read at the remote site reflecting the last write at the cache site.
- **Close-to-open consistency**
  - For Open $\delta = 0$
  - For Close $\mu = 0$
- **When disconnected for time T**
  - $\delta \rightarrow$ T
  - $\mu \rightarrow$ T

Conflicts

- **In the absence of cross-site locking**
- **… In the presence of concurrent updates**
- **… Conflicts will happen**
- **Conflict detection**
  - Based on <ctime, mtime, inode#>
- **Conflict resolution**
  - Policy driven per dir tree
  - No data loss …copy to .conflicts dir

Caching Large Namespaces

- **Directory Tree “created” on demand**
- **Readdir (e.g., ls)**
  - <name, inode number>…
- **Readdir plus attrs (e.g., ls –l)**
  - < inode attrs> for each dir entry
- **Inodes allocated but not “created”**
  - Directory entry contains “orphan” inode
  - “create” the inode on a later lookup

Namespace Caching Results

Files per dir	Readdir w/ creates	Readdir w/ orphan inodes	Readdir from cache
100	1.9 s	0.7 s	0.03 s
1000	3.1	1.2	0.09
10000	7.5	2.8	0.15
100000	451.7	25.4	1.2

Summary

- For global file access across data centers
- …cache should be scalable
- Parallel data fetching from/to multiple nodes
  - Miss throughput limited by WAN bandwidth
- Hit throughput matches local data access
- Update throughput matches local access
- All data/metadata updates are asynchronous
  - Handle intermittent network connectivity by design

Questions?