An HDD-based Lustre HSM Implementation Using A Scalable Object Store.

Discussion And Performance Evaluation

Maria Perez Gutierrez, Thomas Favre-Bulle, James Coomer, Shuichi Ihara, Robert Triendl

LAD 2015

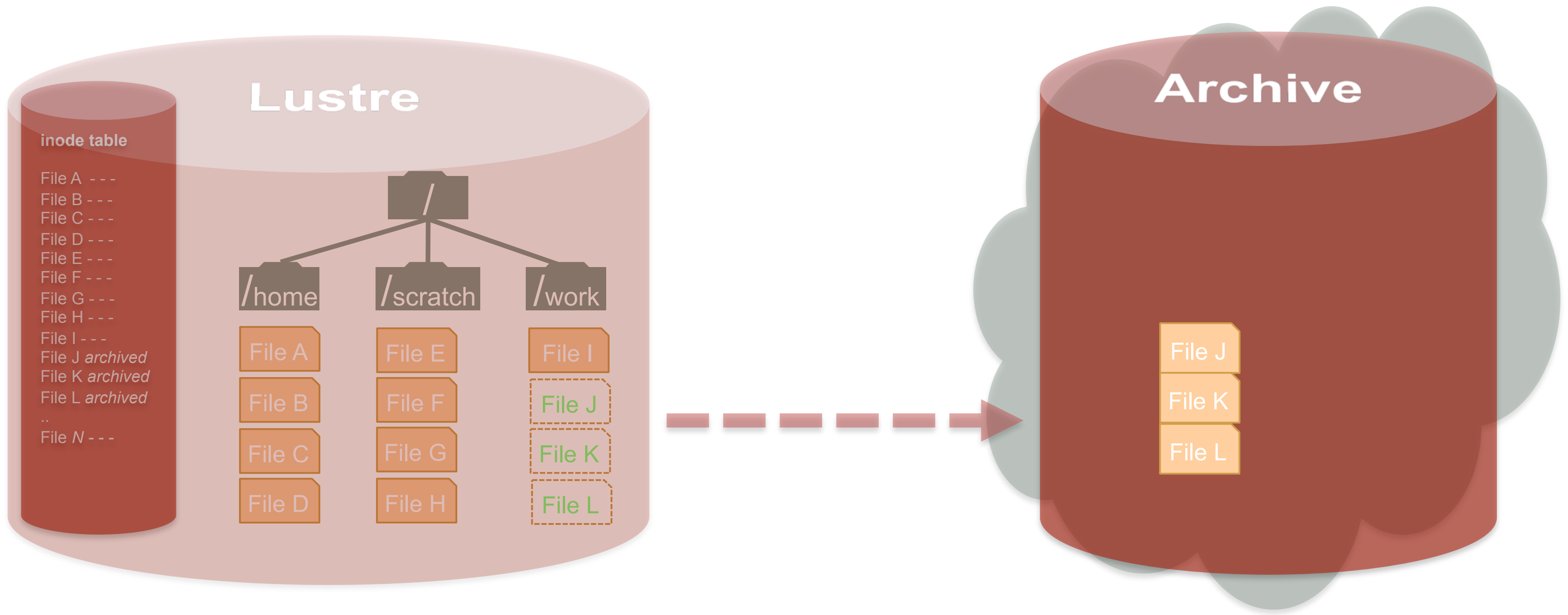
Agenda

- Motivation
- HSM idea
- Why an Object Storage Target
- Object Storage Solution Overview: WOS
- WOS Copytool Implementation
- Test environment & benchmarked Operations
- Results
- Summary

Motivation

- **Current Situation**
  - Robinhood has driven stronger HSM takeup for Lustre
  - Most are implemented to tape (e.g. HPSS) and scale-out NAS
- **But**
  - Object Stores are faster than tape, and simpler to manage than NAS. They also can deliver simple DR
- **So**
  - We implemented a new, simple copytool to an object store (DDN WOS)
- **And**
  - We compare it with other methods

HSM idea



Archive: “Copy blocks for candidate files into Archive” e.g. after last access > 14 days
Release: “Release Lustre blocks for these” e.g. when filesystem is 70% full

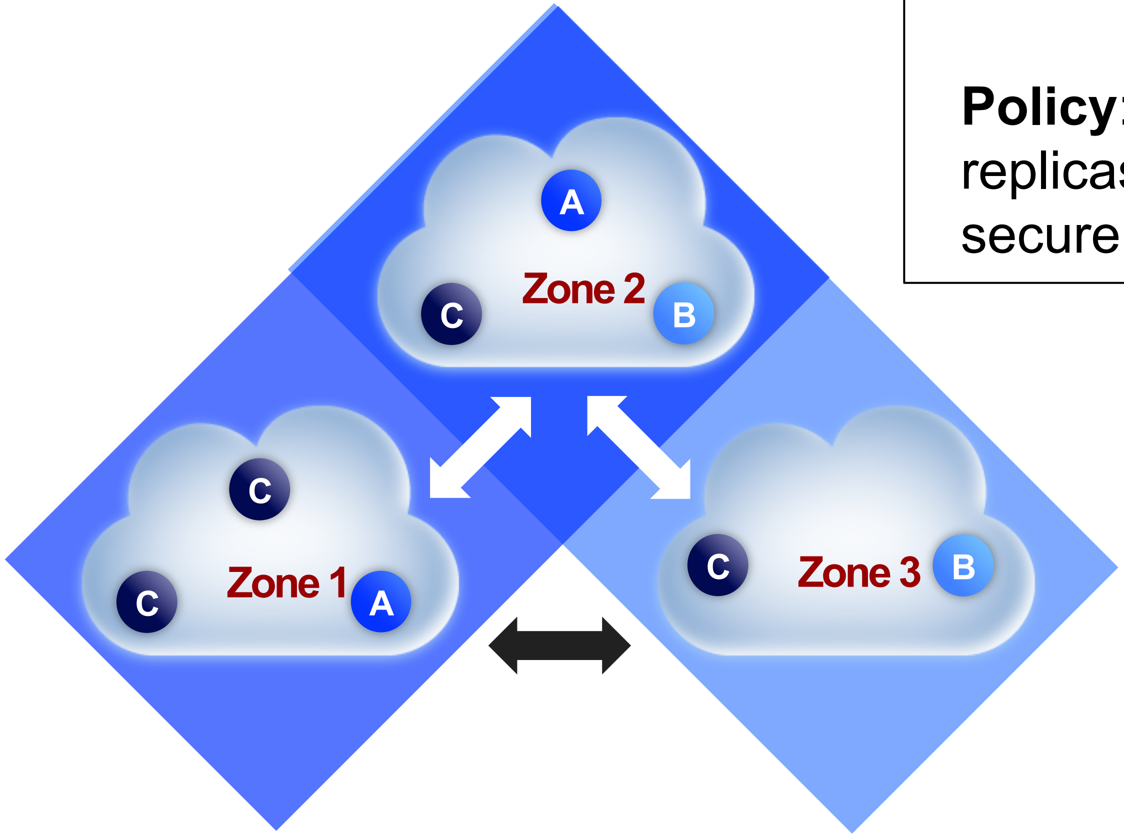
Why an Object Store Target for HSM?

- Eliminate the higher latency, lower throughput, and uncertainty associated with Tape
- Easily implement data distribution to remote sites
- Flexible Policies for data protection and distribution
- Much higher scalability and lower management overhead than an HSM NAS target
- Object stores optimised for lower cost at scale

WOS: Overview

- WOS is an object storage platform that exposes a PUT/GET/DELETE API
- Object Disk Architecture: drives formatted with custom WOS disk object layout, no Linux FS, no fragmentation, fully contiguous object read and write operations for maximum disk efficiency
- WOS storage nodes can be distributed geographically to build a global storage cloud
- Data is stored as objects, with an object ID and metadata in a flat namespace
- PUTs into WOS require a POLICY to be requested
- POLICIES can created to define where the data is replicated to, and how it is erasure coded across drives and sites.

WOS: Overview

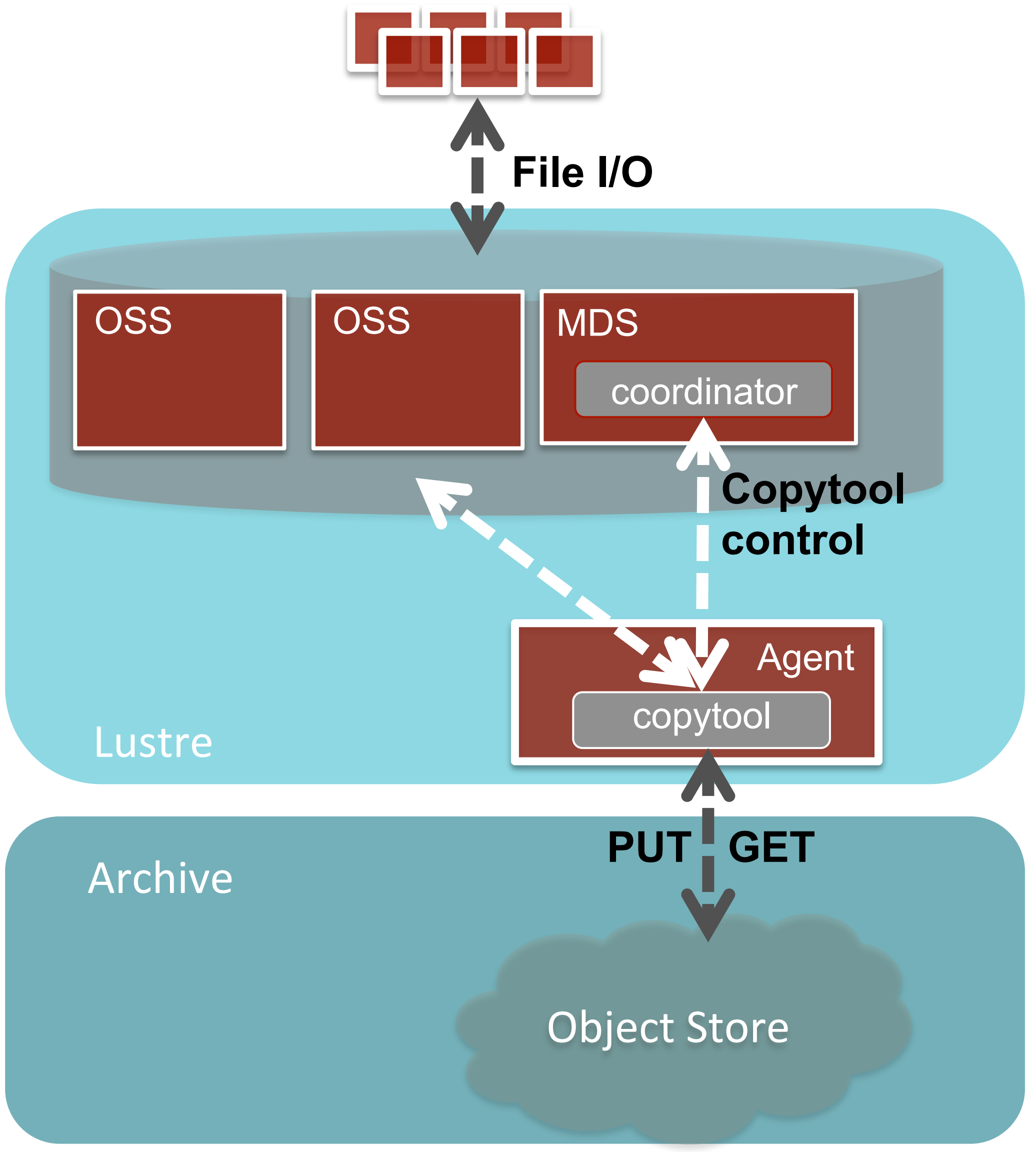


WOS cluster: No. of zones containing nodes.

Policy: defines how an object is stored, no. of replicas in each zone and the method used to secure the data.

Policy A	Zone 1 = 1 Zone 2 = 1
Policy B	Zone 2 = 1 Zone 3 = 1
Policy C	Zone 1 = 2 Zone 2 = 1 Zone 3 = 1

WOS Copytool: Data Flow



WOS Copytool Implementation

- **lhsmtool_wos**
- **Operates similar to POSIX copytool**
- **OPTIONS:**

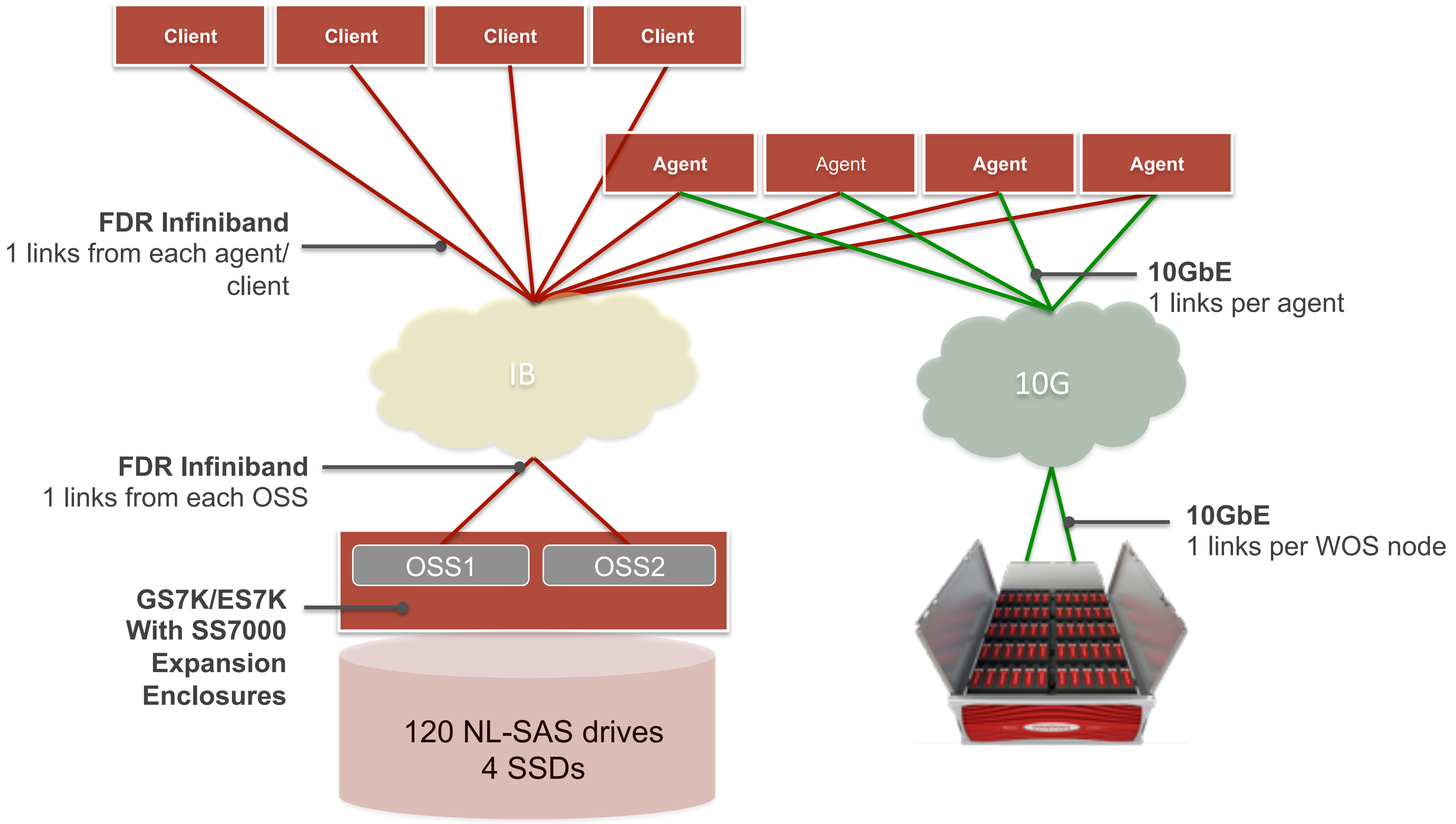
```
-A : archive_id : each copytool agent has a unique archive id
-c : chunk_size: chunk
-d : daemon, run the wos copytool on daemon mode
-h : WOS_IP
-p : WOS_POLICY
-v : verbose
```

- **Internal commands:**
  - `wos_copy put -s INFILENAME -h WOS_IP -p WOS_POLICY`
  - `wos_copy get -o OID -d OUTFILENAME -h WOS_IP -p WOS_POLICY`

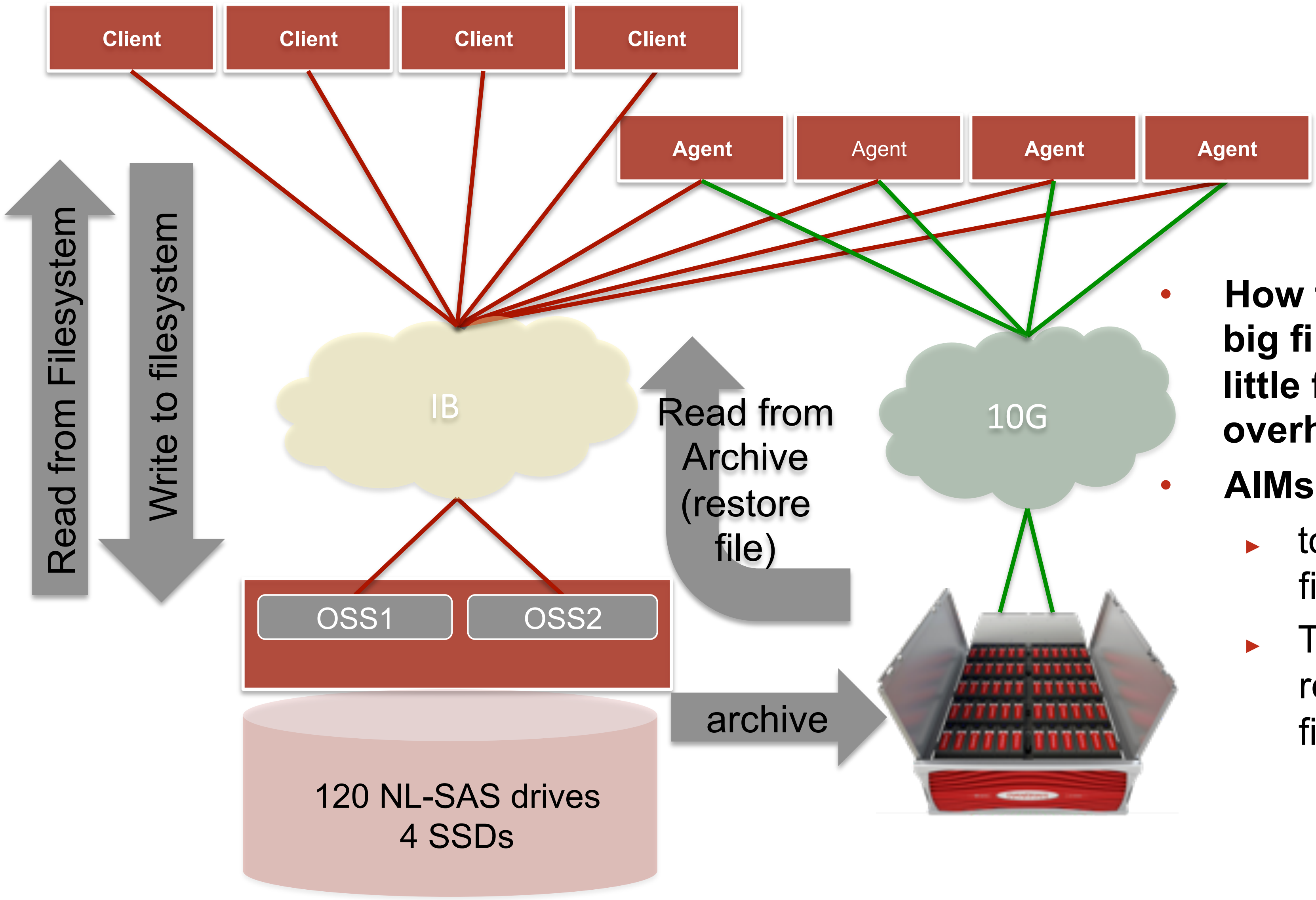
- **An archived file then finds the xattr populated with OID to allow later retrieval**

```
[root@testy test1C]# getfattr -n trusted.oid /lustre/testfile
# file: lustre/testfile trusted.oid="CDDEjBNaBlkKSrCFqBNrgURVpMoD2D6e8hG0FYWG”
```

Test Environment

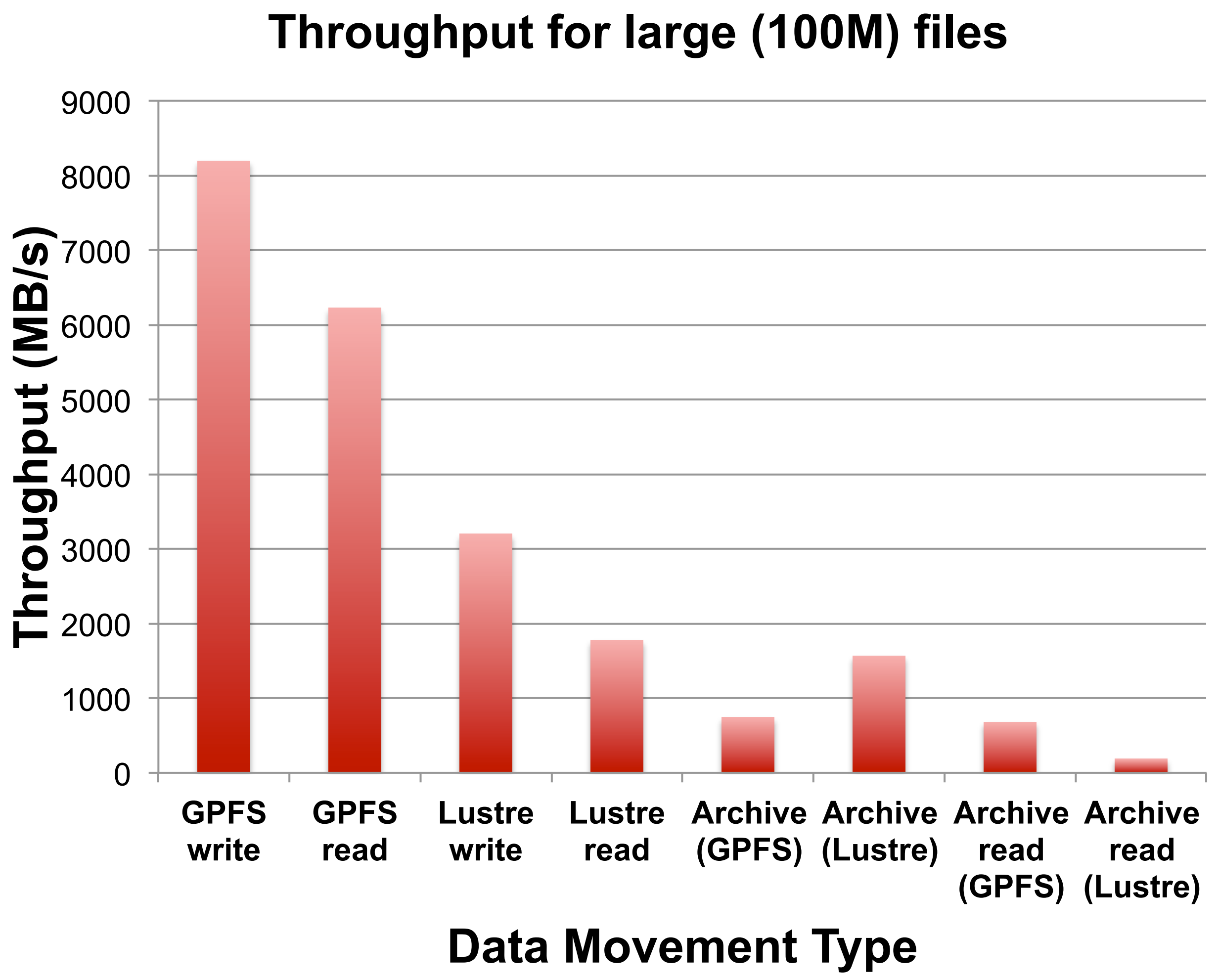


Benchmarked Operations



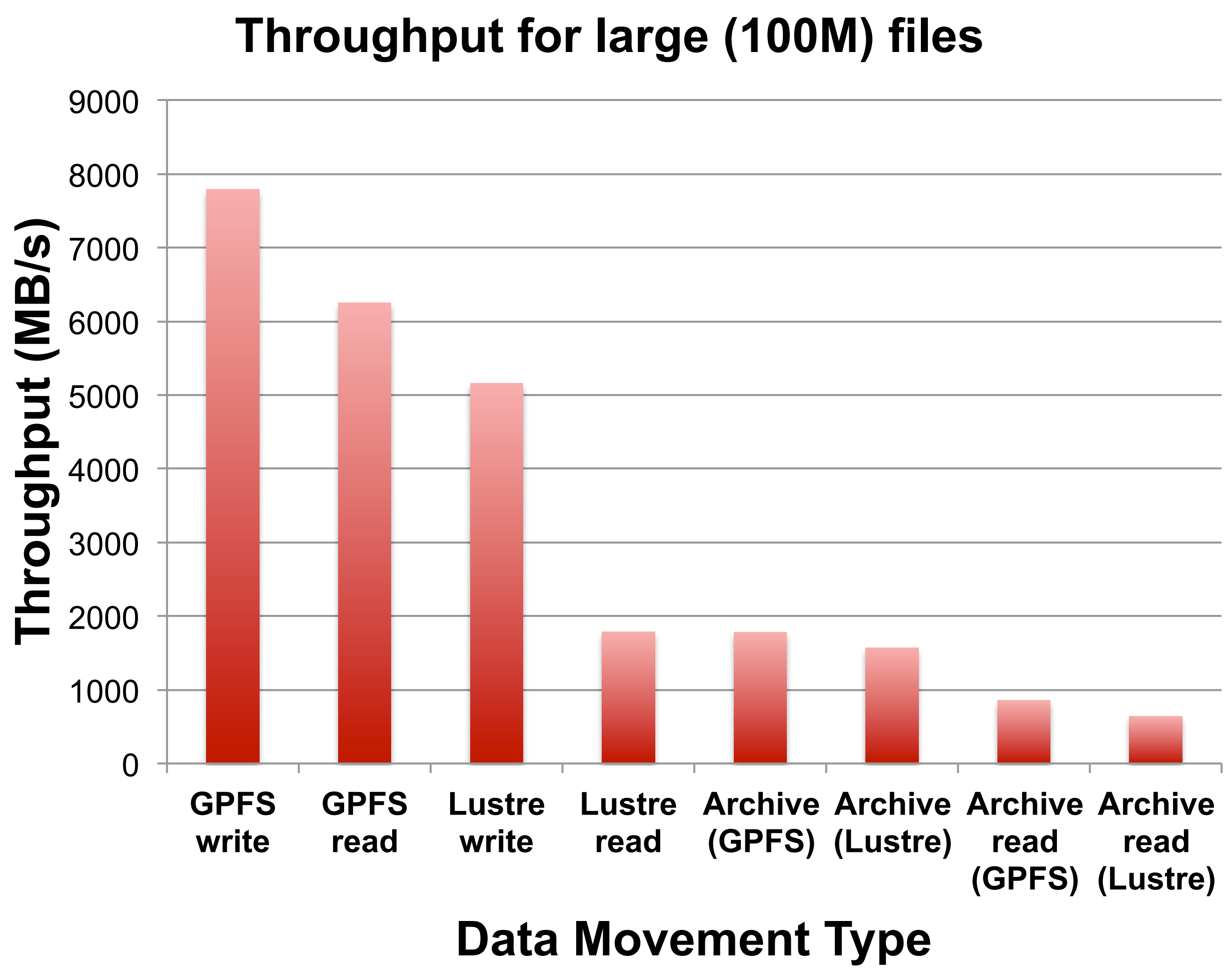
- **How fast can we move big files (throughput) and little files (software overhead for operations)**
- **AIMs:**
  - to keep up with the filesystem IO
  - To provide reasonable read rates for archived files

Results: Large Files - Throughput



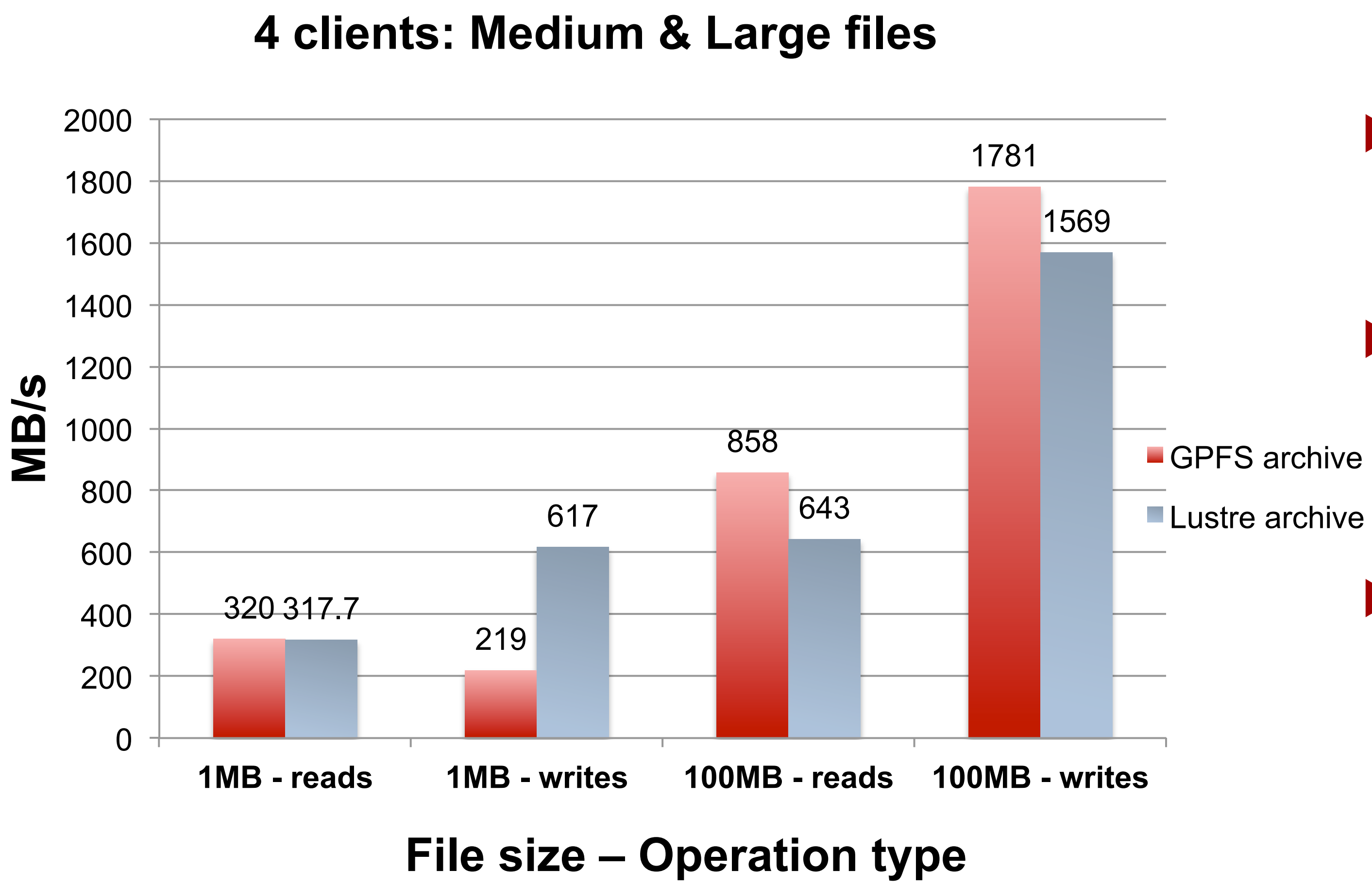
- Using 4 clients/agents, single thread per client/ agent
- Around 1.5GB/s to Archive files into WOS for large files

Results: Large Files - Throughput



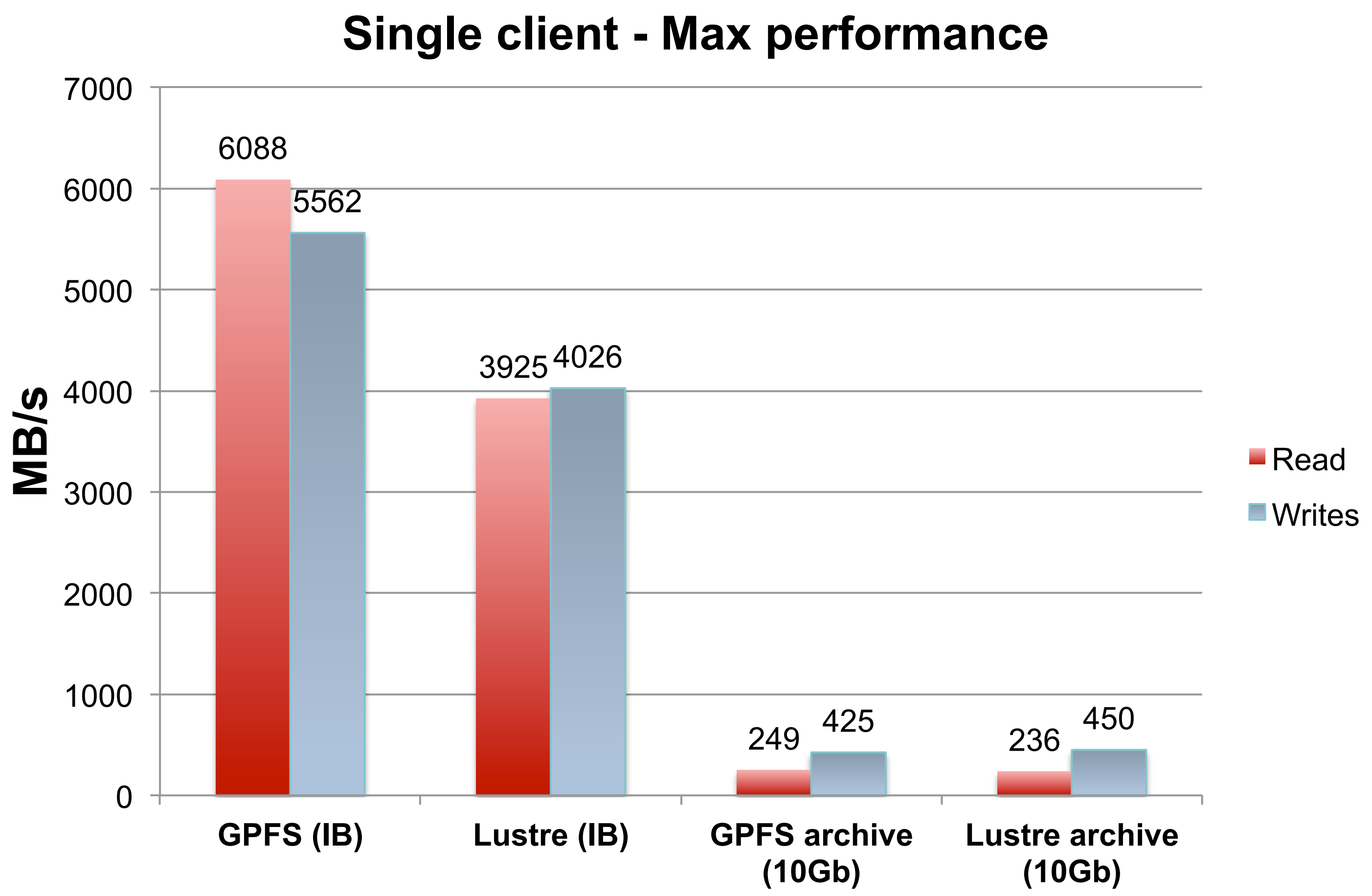
- Using 4 clients/agents, multithread per client/agent
- Around 1.5GB/s to Archive files into WOS for large files
- Aggregate Reads from WOS ~650MB/s

Results: Medium and Large files



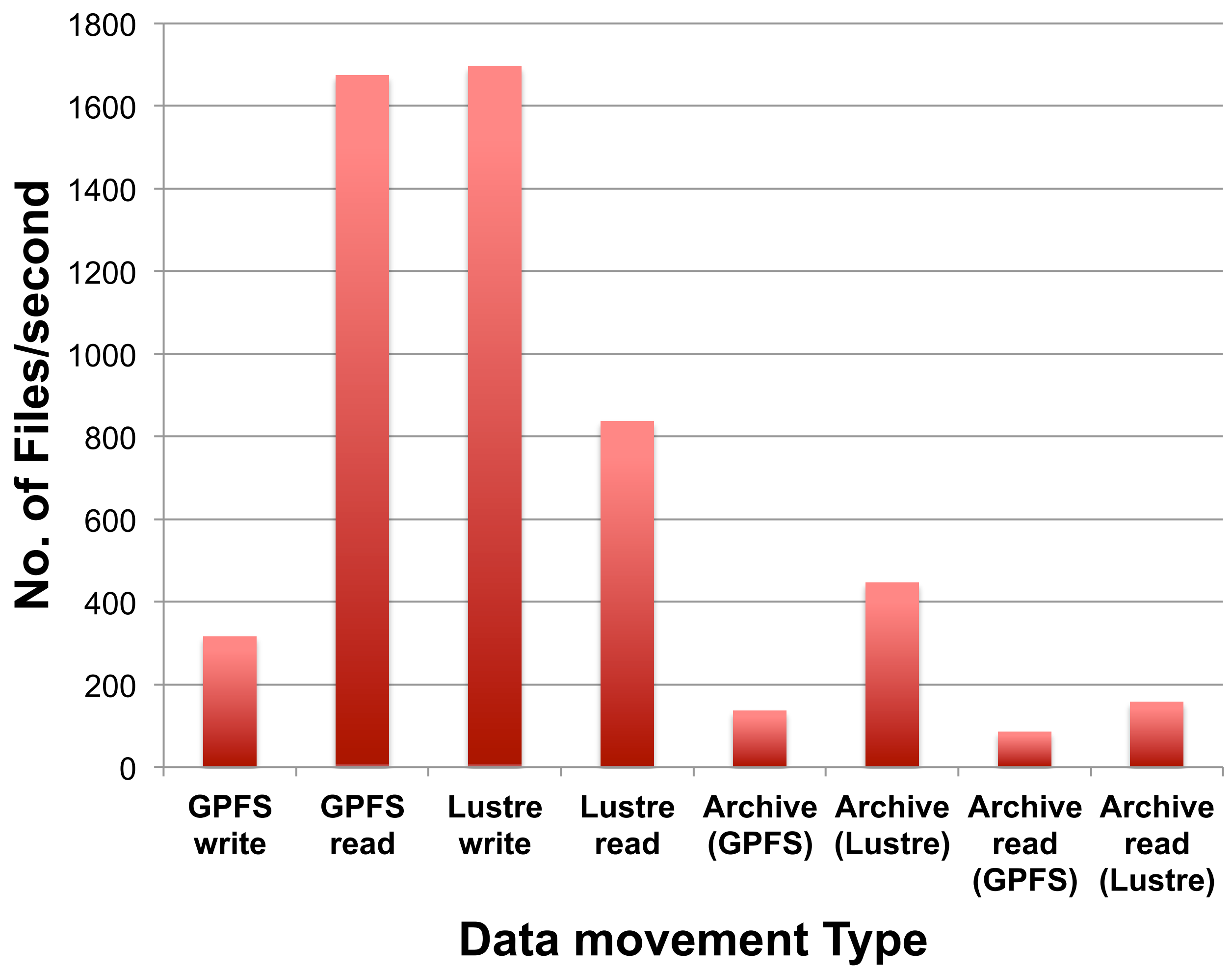
- Using 4 clients/agents, multithread per client/agent
- WOS copytool provides higher bandwidth than GPFS archive solution for 1M files
- Results for large files get close to GPFS based solution

Results: Single client



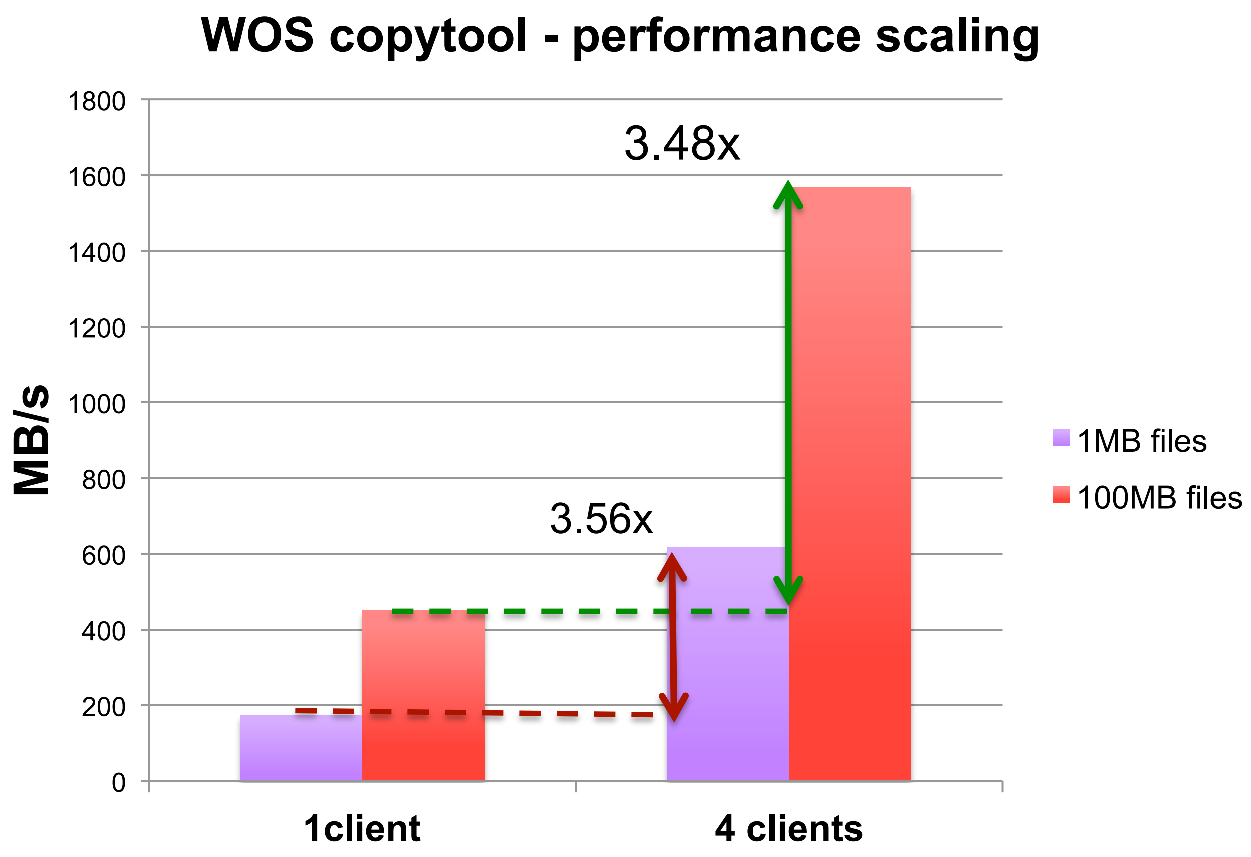
- WOS copytool provides maximum client performances aligned with values provided by the GPFS based solution

Results: Small Files (4k) – Rates



- Using 4 clients/agents, with one thread per client/agent
- Around 400 files moved per second archiving data
- Got higher rates with Lustre Wos-copytool on archiving and recovering data

Performances scaling



Summary

- **Promising data rates to Object Store**
- **Next Steps:**
  - ***Scalability Tests:* Clients and Copytool Agents at scale**
  - ***Explore Data Protection Options***
    - ***Enable DR within Object Store (support a remote namespace)***
    - ***Enable Backup operations to Object Store (immutable file copies to object store)***

Thank you

Questions?