Integration of cloud-based storage in BES III computing environment

Lu Wang
Lu.Wang@ihep.ac.cn
IHEP computing center, Beijing, China

Fabio Hernandez
fabio@in2p3.fr
IN2P3/CNRS computing center, Lyon , France

ZiYan Deng
dengzy@ihep.ac.cn
IHEP experimental physics center, Beijing, China

CHEP2013, Amsterdam, October 2013

Outline

- Context
- Cloud storage for physics data
- Evaluation
- Perspectives
- Conclusion

Context

Cloud storage

- Object storage system

  well documented interface

  on top of standard protocols (HTTP)

  accessible through wide area network

- Advantages

  elasticity, standard protocols, tunable durability by redundancy, scalability, possibility of using lower cost hardware, private or public

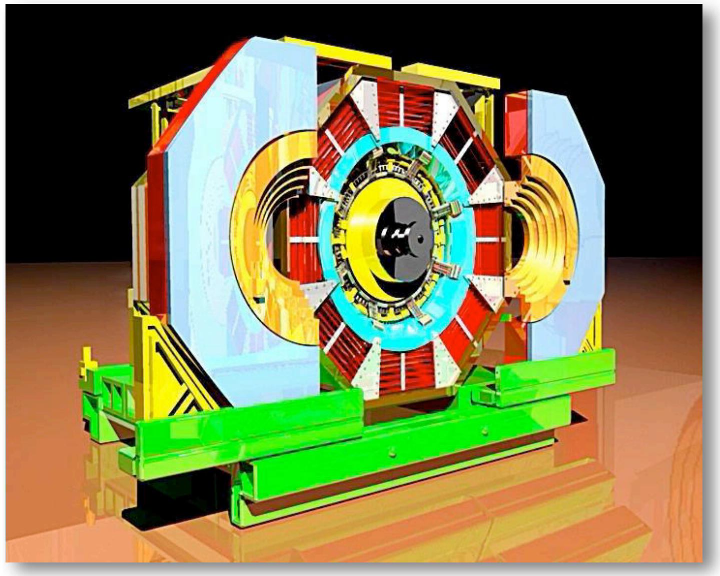
- Significant development over the last few years

  Amazon S3: 2 trillion objects, 1.1M requests/sec (as of April 2013)

- Typical use cases

  well suited for "write-once read-many" type of data: images, videos, documents, static web sites, …

BES III experiment



http://bes3.ihep.ac.cn

- **Collaboration**

  physics in the tau-charm energy region around 3.7 GeV

  world's largest samples of J/ψ and ψ′ events

  53 institutions (Asia, Europe and USA)

- **Data acquisition**

  BEPCII double-ring electron-positron collider at IHEP campus, Beijing

  raw data on tape at IHEP computing center

- **Data processing**

  central role of IHEP computing center both for data processing and derived data repository

  DIRAC-based distributed environment: ~10 external sites, mostly small

  fraction of simulation and analysis performed at external sites

  offline software framework built on top of ROOT for I/O

BES III experiment (cont.)

- Computing resources

  bulk of resources provided on site by IHEP computing center

  current: 4,500 CPU cores, 3 PB disk , 4 PB tape

  expected: up to 10,000 CPU cores, 5 PB disk, 10 PB tape

- Disk storage

  largely based on Lustre

  can fully exploit available hardware

  very effective for high-throughput I/O, not so for handling lots of small files

  requires specialized manpower for keeping the storage infrastructure stable

  not very convenient for sharing data with external sites

  unaffordable for small sites participating to BES III

Motivation

- Can we use cloud storage for physics data?
- How does using cloud storage impact the experiment?
- What of our use cases is cloud storage good for?

Cloud storage for physics data

Cloud storage vs file system

	File system	Cloud storage
Storage unit	file	object
Container of data	directory	container (a.k.a bucket)
Name space hierarchy	multi-level `/dir1/dir2/.../dirn/file`	2 levels container(obj1, obj2,obj..,objn)
File update	allowed	not allowed
Consistency	individual write()s are atomic and immediately visible to all clients	updates eventually consistent
Access protocol	POSIX file protocol `file://dir1/dir2/dir3/file1`	cloud protocol over HTTP `s3://hostname/bucket/object`
Command line interface	`cp`, `mkdir`, `rmdir`, `rm`, `ls`, ...	`s3curl.pl`, `s3cmd`, `swift`, ...

Extending ROOT for cloud storage

- BES III requires ROOT v5.24.00b (Oct. 2009)

  improved built-in support for S3 protocol from ROOT v5.34.05 (Feb. 2013)

- We developed an **extension** to ROOT for supporting cloud protocols

  currently both Swift and S3

  tested against Amazon S3, Google Storage, Rackspace, OpenStack Swift, Huawei UDS

  backwards compatible with all versions of ROOT since v5.24

  no modification to ROOT source code nor to experiment code is required

- Features

  partial reads, web proxy handling, HTTP and HTTPS, connection reuse

  lightweight shared object library (500KB) + TFile plugin

  installable by unprivileged user

Extending ROOT for cloud storage (cont.)

- Usage

experiments can ***efficiently read*** *remote files using cloud protocol as if the files were stored in a local system*

```
TFile* f = TFile::Open("swift://fsc.ihep.ac.cn/myContainer/path/to/myDataFile.root")
```

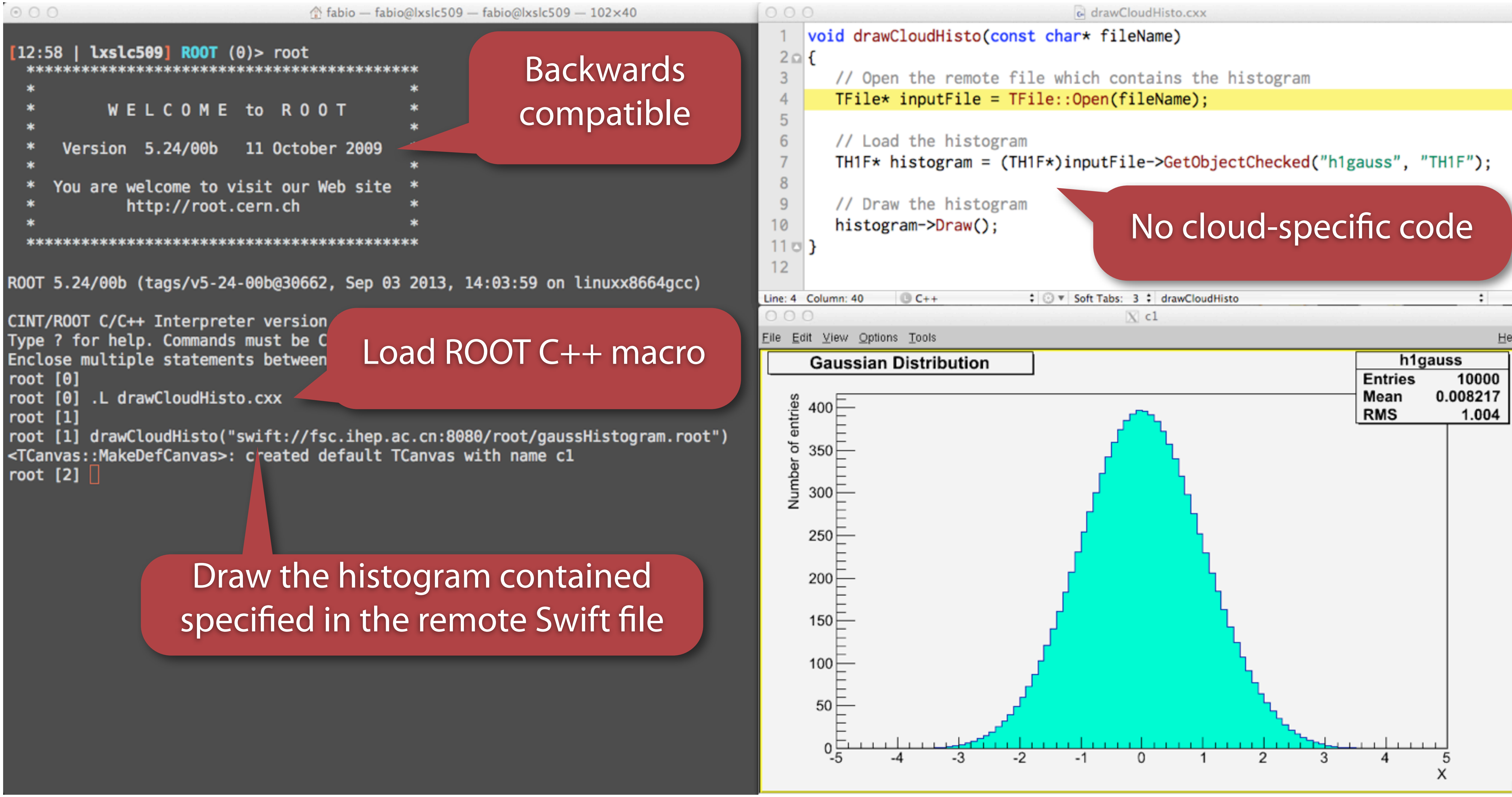
individuals can easily share URLs to their cloud files with other ROOT users

"Look at my plot at `s3://s3.amazonaws.com/myBucket/myHisto.root`*"*

- Source code and documentation on GitHub

`https://github.com/airnandez/root-cloud`

Extending ROOT for cloud storage (cont.)



With this extension, BES III can transparently use cloud storage

CLI-based interface to cloud storage

We developed a CLI-based S3 and Swift client

Advanced functional prototype

Compatible with Amazon, Google, OpenStack Swift, Rackspace, Huawei, …

Exploits GO programming language built-in concurrency

Small size, stand-alone executable, so installable on the fly

Works on MacOS, Linux and (soon) Windows

Source to be open after cleaning

```
$ mucura --help
Usage:
   mucura <command> [opts...] args...
   mucura --help
   mucura --version

Accepted commands:
   lb      list existing buckets
   mb      make new buckets
   ls      list bucket contents
   rb      remove buckets
   up      upload files to a bucket
   dl      download files
   rm      remove files

Use 'mucura help <command>' for getting more information on a specific command

$ mucura dl http://s3.amazonaws.com/mucura/path/to/myFile /tmp
```

Filesystem interface to cloud storage

- Useful to expose cloud storage as a local file system

  usual Unix file manipulation commands work transparently (e.g. cp, ls, tar, …)

  POSIX-based applications work (almost) unmodified

- Evaluated S3fs, a FUSE-based file system designed for Amazon S3 backend

  https://code.google.com/p/s3fs

- Features

  files and directories have their corresponding objects named with their full path in S3fs

  directories implemented as empty objects to store their metadata

  download whole file to local cache on open(), subsequent operations act on the local copy

  new or modified files are uploaded on close()

See backup slides for details

Filesystem interface (cont.)

- S3fs limitations

  one mount point can only expose a single bucket

  downloads the whole remote object on open()

  renaming of directory is not supported: potentially expensive operation

  only supports Amazon S3 backend : can be tweaked to (partially) work with others, though

- Interesting interface in particular for human users, for instance for navigating the name space

- Desirable features for supporting BES III use cases

  single mount point for multiple buckets

  on-demand partial download

  support other cloud protocols

Evaluation

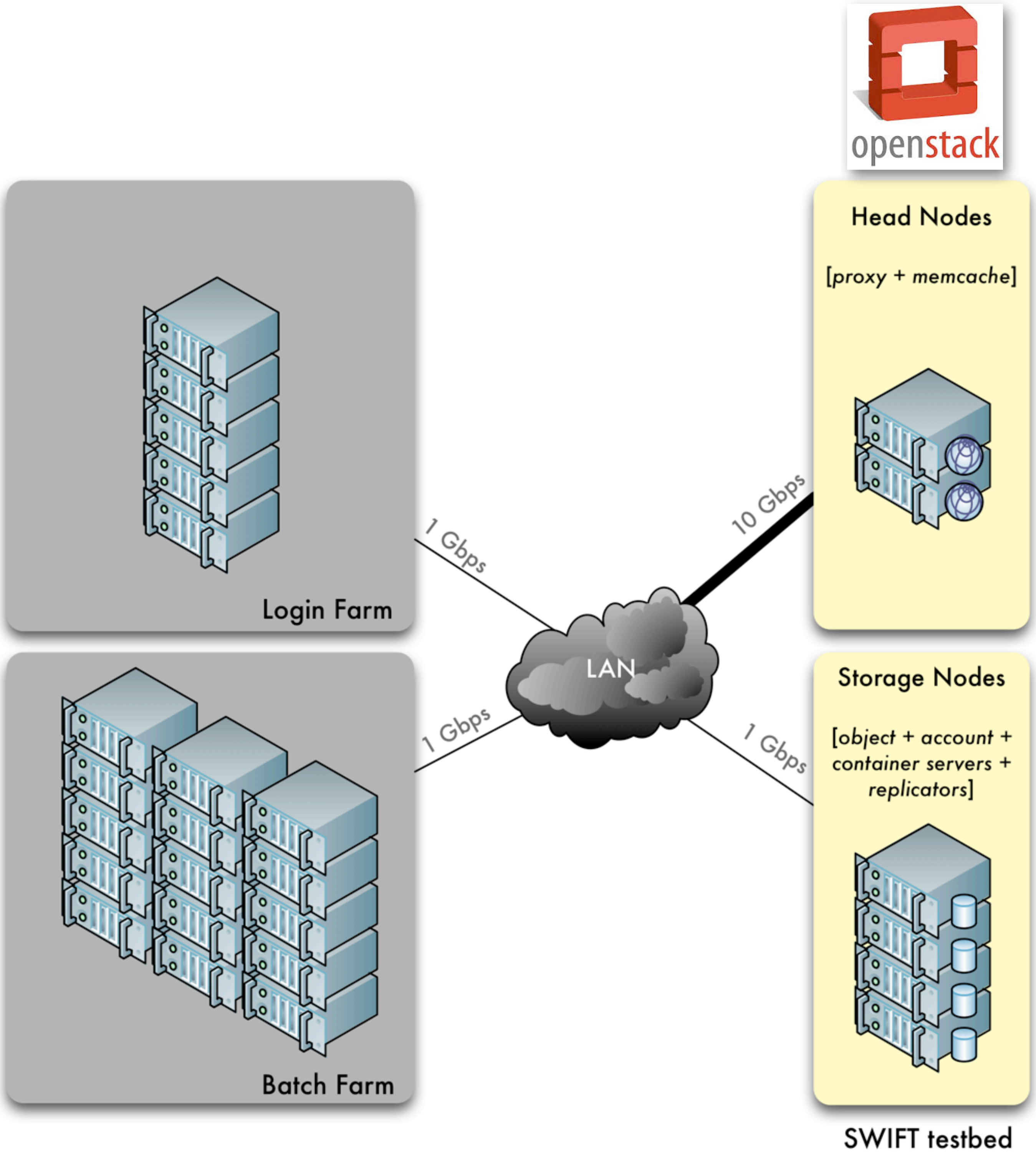
Goals

Use cloud storage and measure:

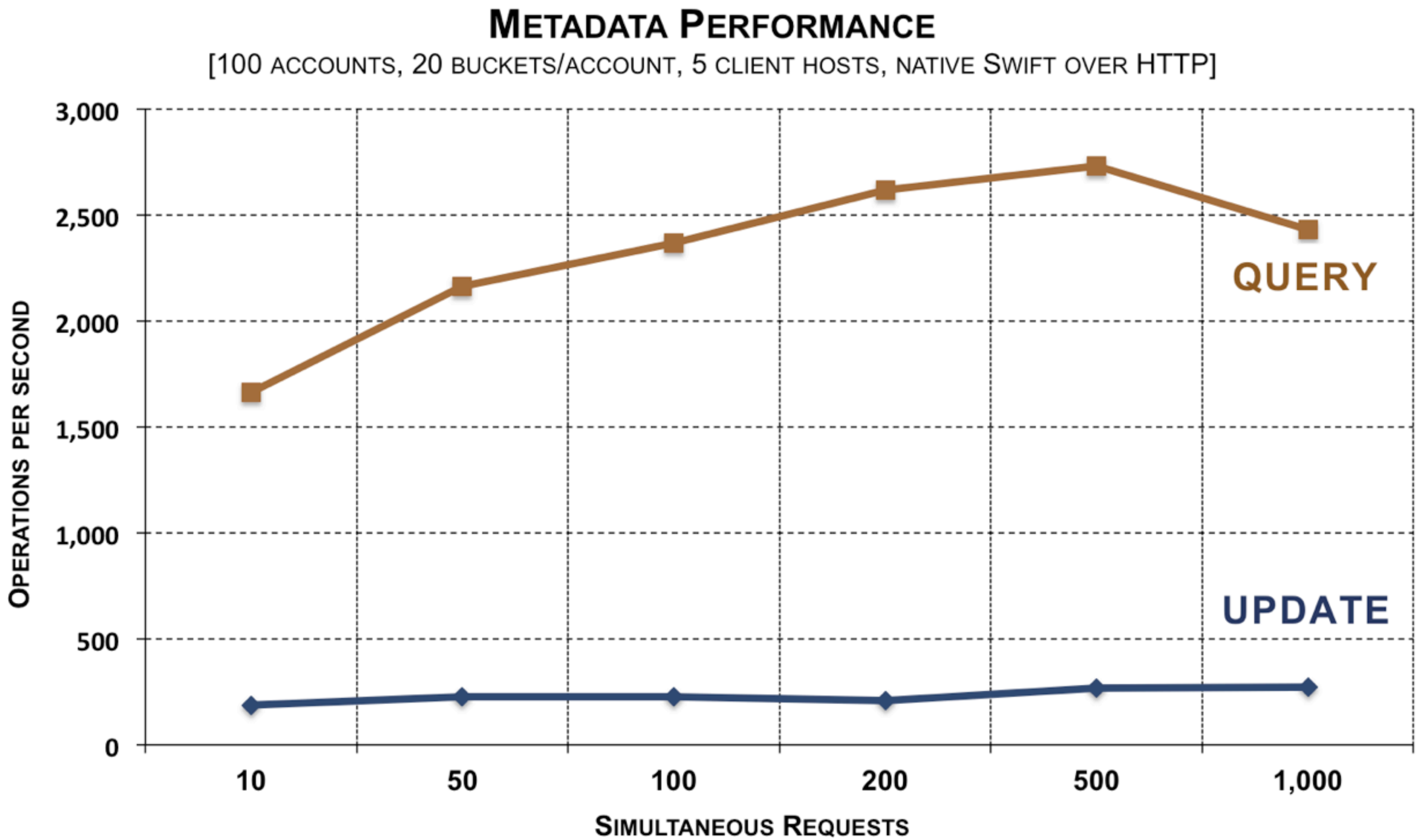
- Performance with small-sized files
- Efficiency of access protocol
- Performance and scalability when used by real BES III jobs

OpenStack Swift test bed

- Head Node x2
  10Gb Ethernet, 24GB RAM, 24 CPU cores
- Storage Node x4
  1 Gb Ethernet, 24GB RAM, 24 CPU cores
  3 x 2TB SATA disks
- Aggregated raw storage capacity: 24TB
- Max read throughput： 480MB/s
- Access protocols
  native Swift
  Amazon S3 (partial support with 'swifts3' plugin)
- Software
  OpenStack Swift v1.7.4
  Scientific Linux v6

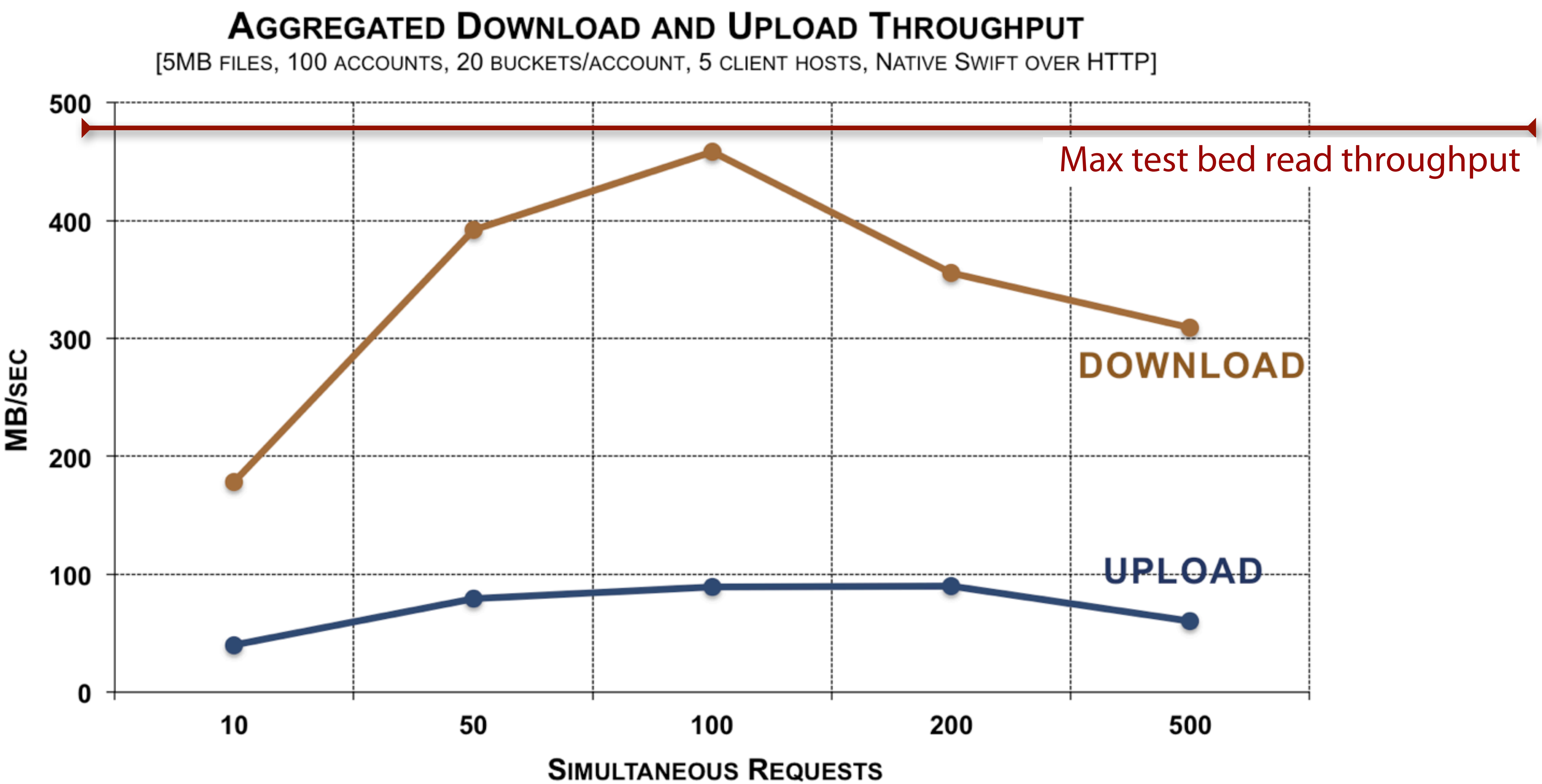


Metadata performance



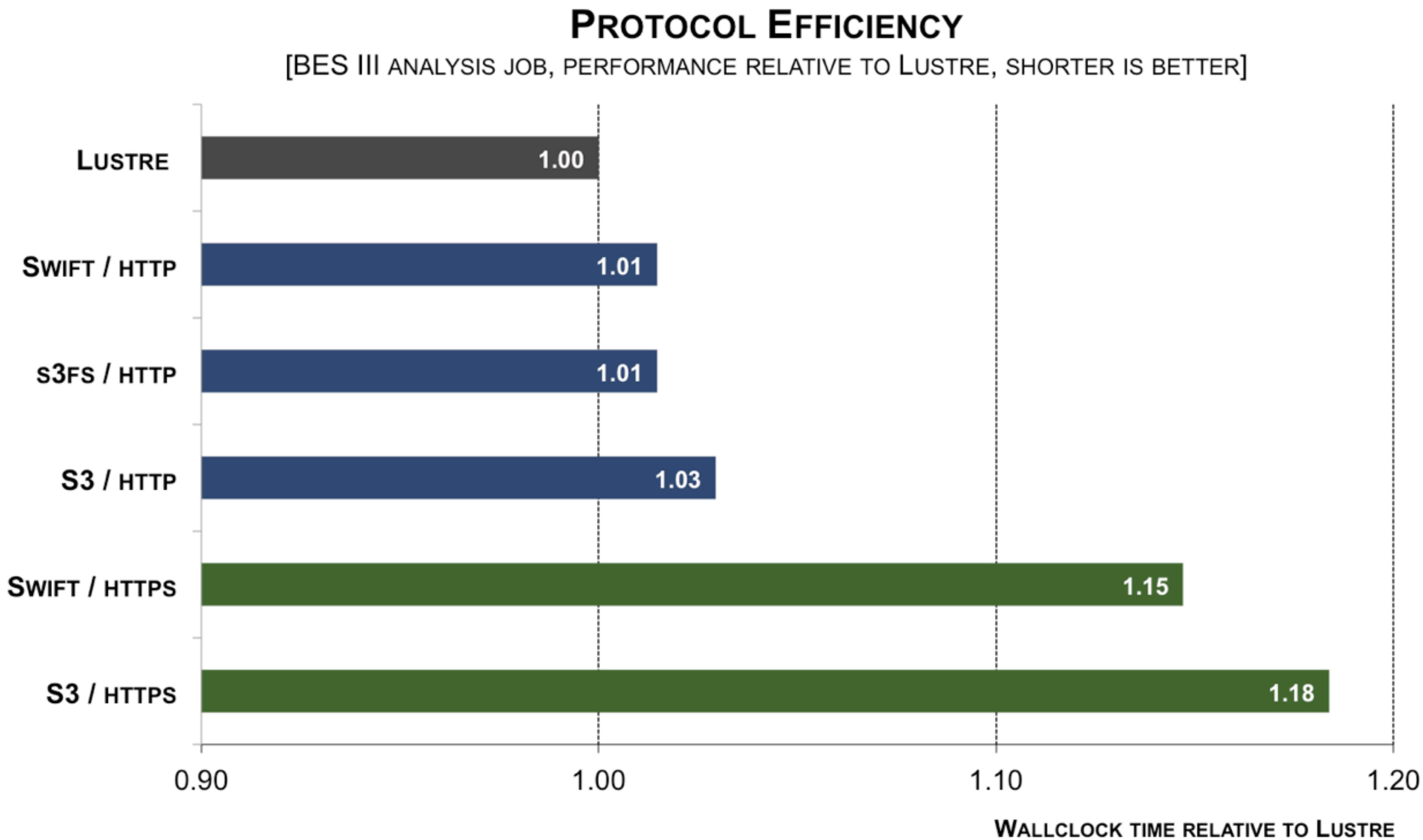
Significant gap between update and query performance
Very low update performance relative to Lustre

Throughput with small-sized objects



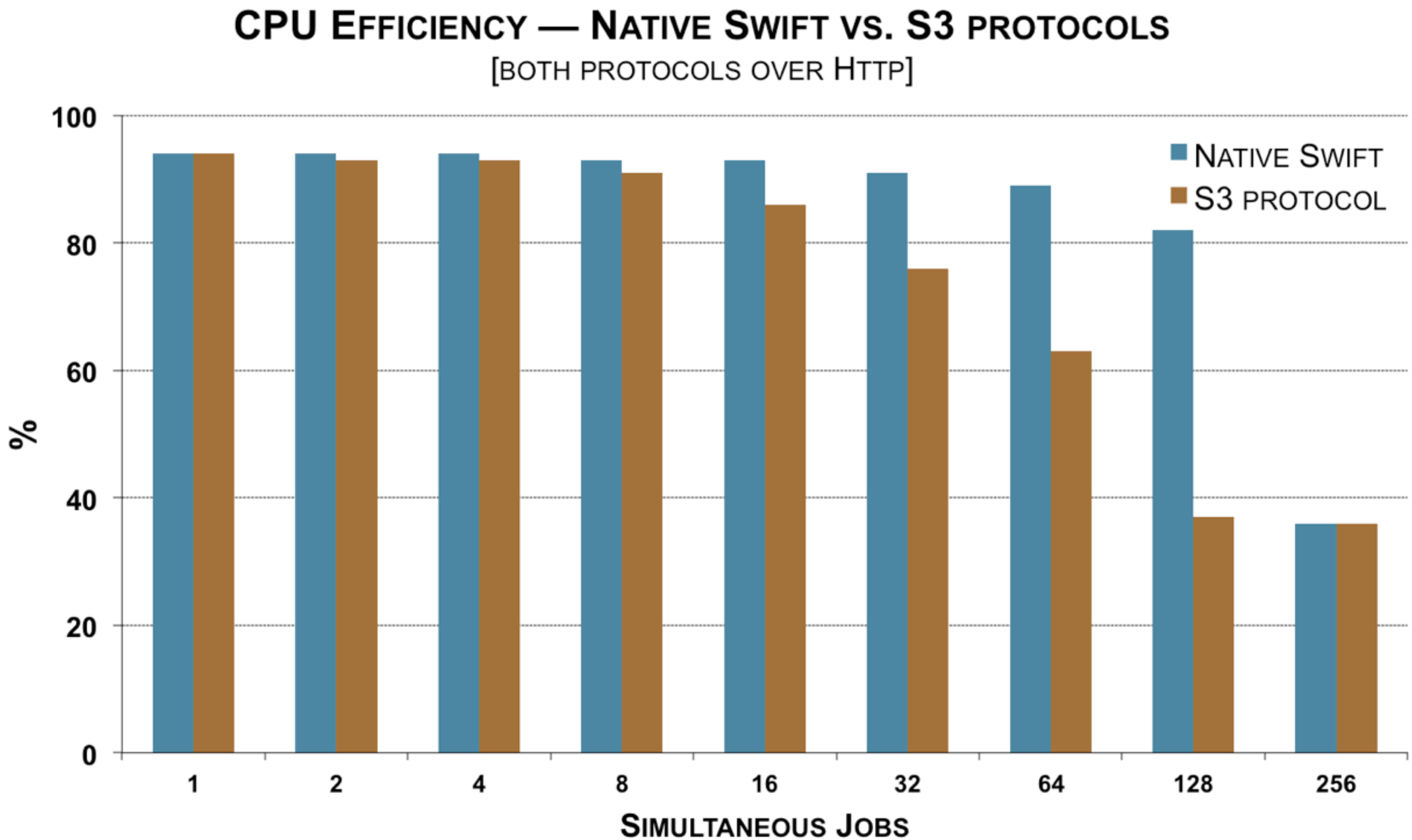
Replication impacts write performance

Efficiency with real jobs



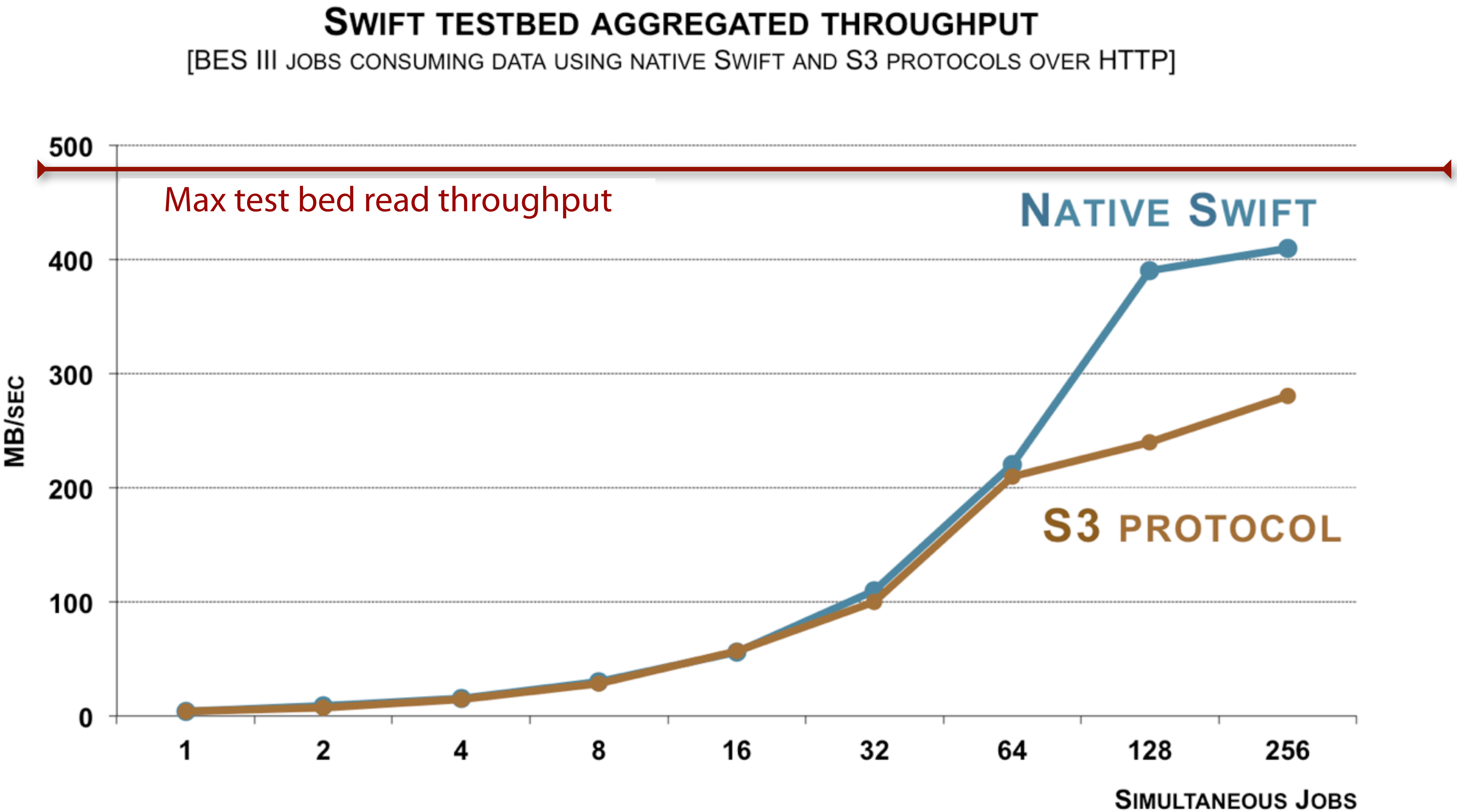
Low overhead of both native Swift and S3 over HTTP
Noticeable penalty when using HTTPS

Efficiency with real jobs(cont.)



With native Swift protocol, up to 128 jobs can be fed to stay above 80% CPU efficiency. Each job consumes 3.7MB/sec

Throughput with real jobs



Swift delivers up to 85% of test bed max read throughput

Perspectives

Potential uses of cloud storage for BES III

- Storage of physics data at external sites

  demonstrated good scalability and efficiency of BES III physics analysis jobs

  deployable in-house or rented

  can exploit less expensive hardware

  allegedly requires less manpower for operations

- Storage backend for small files

  individual user files (software, analysis results, plots, papers, …)

  accessible not only when on campus but remotely over wide area network

  requires friendly client-side interface for interactive use, not fully available at present

- Data sharing among participating sites

  data could be transferred using both "pull" and "push" models or accessible in place

Conclusions

Conclusions

- Demonstrated the possible usage of cloud storage for physics data

  without modification to the experiment's existing software

  thanks ROOT !

- We are convinced of the high potential of cloud storage as a backend for file repositories targeting individuals and groups

  more work needed for improving the situation of the client-side tools

  tests with real users still to be performed

- Sites with limited manpower may consider cloud technologies in their storage strategy

  to lower hardware and operation costs

  to make sharing of data with remote sites more convenient

Questions & Comments

Backup Slides

Notes on Swift configuration

- Tune the number of simultaneous processes of every Swift software component (a.k.a. workers)
- Configure the IPv4 stack of the machines in the Swift cluster so to avoid a high number of TCP connections in TIME_WAIT state
- Configure the verbosity level of Swift logs
- Need to carefully plan for the configuration of the "ring" (the mechanism used by Swift for distributing the storage load among the nodes)
- Use an dedicated client-facing layer for handling SSL termination (i.e nginx)

Estimation of test bed throughput

- Each client write implies 3 disk write operations on separate disks on different storage nodes. Each client read implies single disk read operation
- We compute the max throughput as follows:

$$T_{maxread} = min(T_{headnic} \cdot N_{head}, T_{diskread} \cdot N_{disk}, T_{storagenic} \cdot N_{storage})$$

$$T_{maxwrite} = min(T_{headnic} \cdot N_{head}, T_{diskwrite} \cdot N_{disk}/N_{rep}, T_{storagenic} \cdot N_{storage}/N_{rep})$$

- $T_{headnic}$ and $T_{storagenic}$ are the network bandwidth per head and storage node, respectively (1250 MB/sec, 125 MB/sec)
- N_{head} and $N_{storage}$ are the number of head and storage nodes, respectively (2 and 4)
- N_{disk} is the number of disks in the test bed (12)
- N_{rep} is the configured replication factor (3)
- $T_{diskread}$ and $T_{diskwrite}$ are the read and write throughput of a single SATA disk, as measured by IOZone (40MB/s and 95 MB/sec)
- $T_{maxread}$ = 480 MB/sec and $T_{maxwrite}$ = 380 MB/sec

Notes on using S3fs with Swift

- Software

  server side: Swift v1.7.4 with swift3 module enabled

  client side: s3fs v1.19, fuse v2.7.4, Scientific Linux v5

- S3fs modified to make HTTP requests compatible with swift3

  `http://bucket.domain.org/object → http://host.domain.org/bucket/object`

- Supported functionality

  readdir, mkdir, cp file, delete file, rename file, fopen, fread, fwrite (first time)

- Unsupported functionality

  chown, chmod, rmdir, stat, df

  swift3 ignores all the optional request headers starting with "x-amz-meta-" issued by s3fs

I/O patterns of BESIII job