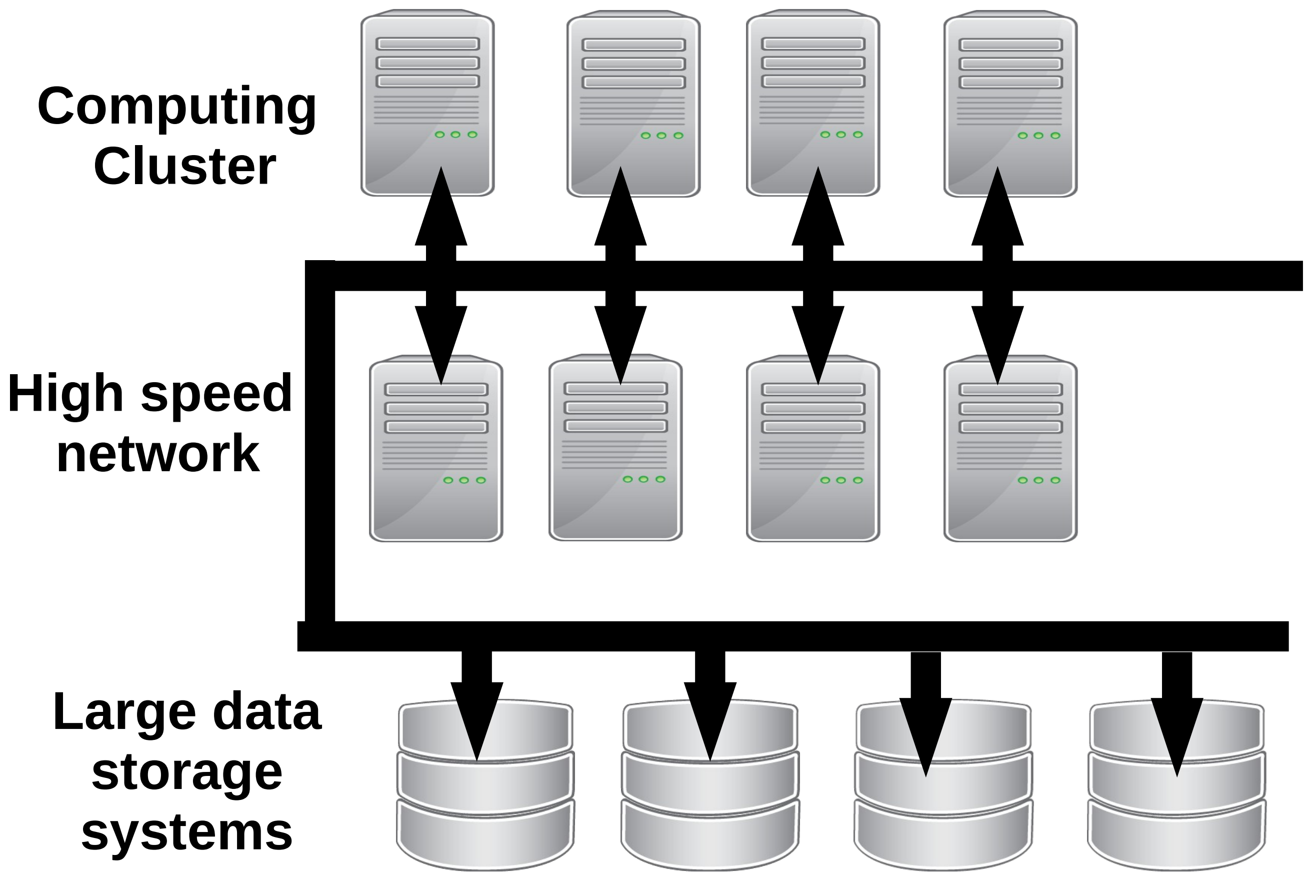
Efficient HTTP based I/O on very large datasets for high performance computing with the Libdavix library

Authors
Devresse Adrien (CERN)
Fabrizio Furano (CERN)

Typical HPC architecture



Remote I/O requirements in HPC

→ Low latency

→ High throughput

→ Parallel access

→ Reliability

HPC I/O protocols

A protocol Zoo...

the globus toolkit

GridFTP

iRods

lustre

hadoop HDFS

dCap

XRootD

ceph

IBM GPFS

pNFS.com

GlusterFS

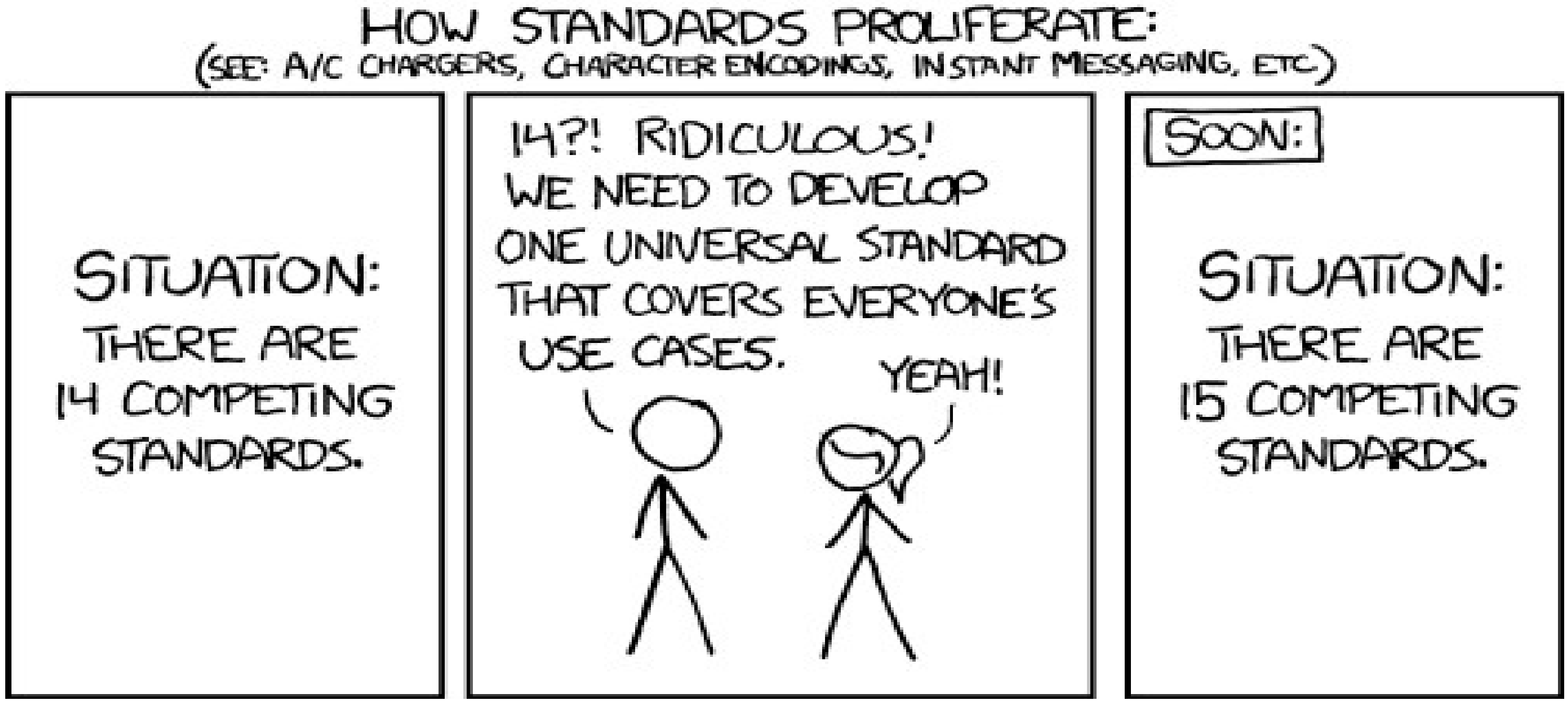
AFS

Very specific protocols

- They are often specific to a storage system
- They use advanced caching strategies and optimizations
- They are optimized for the previous HPC requirements
  - High throughput
  - Parallel access
  - Low latency

Standards : Should we re-define one more ?

Classic problem



Crazy Idea

Why not using HTTP ?

Is this madness ?

First, it seems crazy

- Text based, Stateless
- No multi-plexing
- Suffers of the TCP slow start mechanism
- No standard « fail-over » mechanism
- No multi-sources / multi-streams
- Incomplete support for partial I/O

Not so crazy

HTTP is

- Widespread
  - has a rich ecosystem and powerful actors
- A protocol that scales
  - HTTP Caching is easy to deploy
- Today most Storage Systems provide an HTTP gate
- Flexible, Extensible

What we did

- **Created a tool-kit for HPC I/O with HTTP protocol named DAVIX**

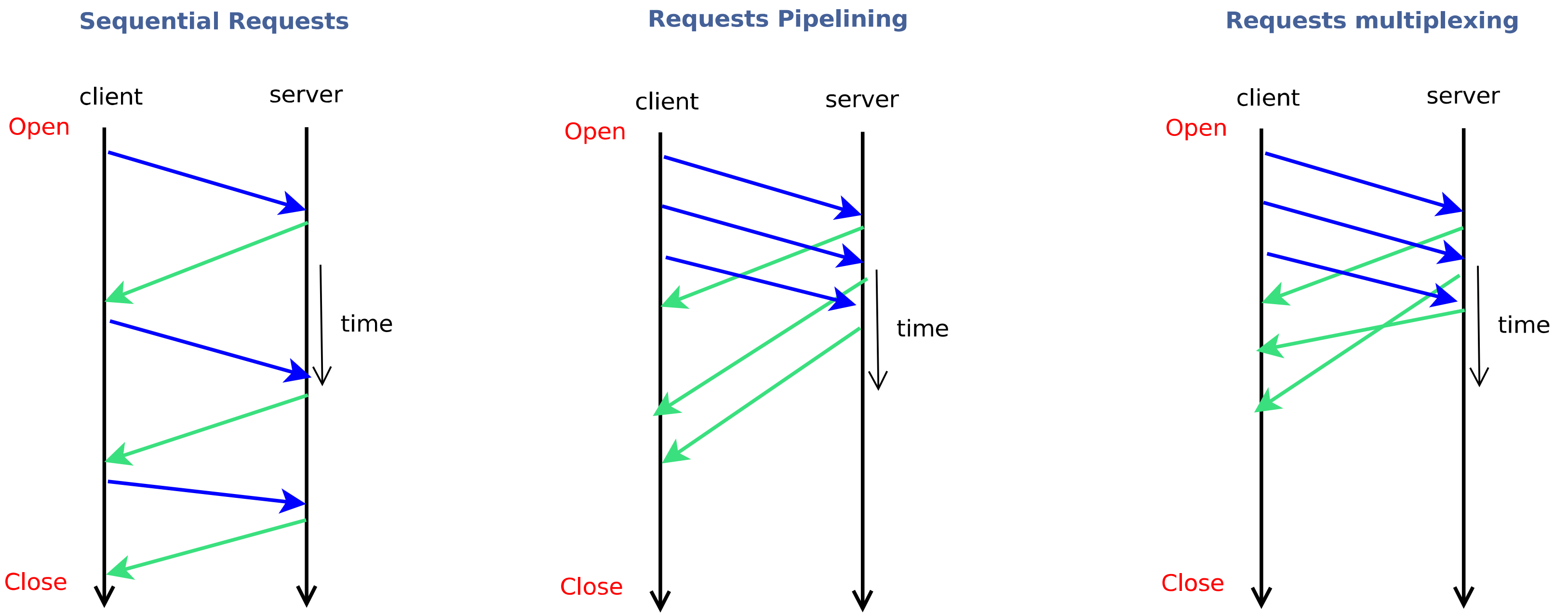
- **Apply several optimizations to make HTTP a competitive protocol in term of performance with HPC specific protocols.**

- **Benchmark it with a High Energy physics data analysis work-flow.**

Problem : Parallelism and persistent connection

	HPC I/O protocol	HTTP
Operation Multiplexing	YES	Pipelining
Persistent connection	YES	KeepAlive

Multi-plexing vs pipelining

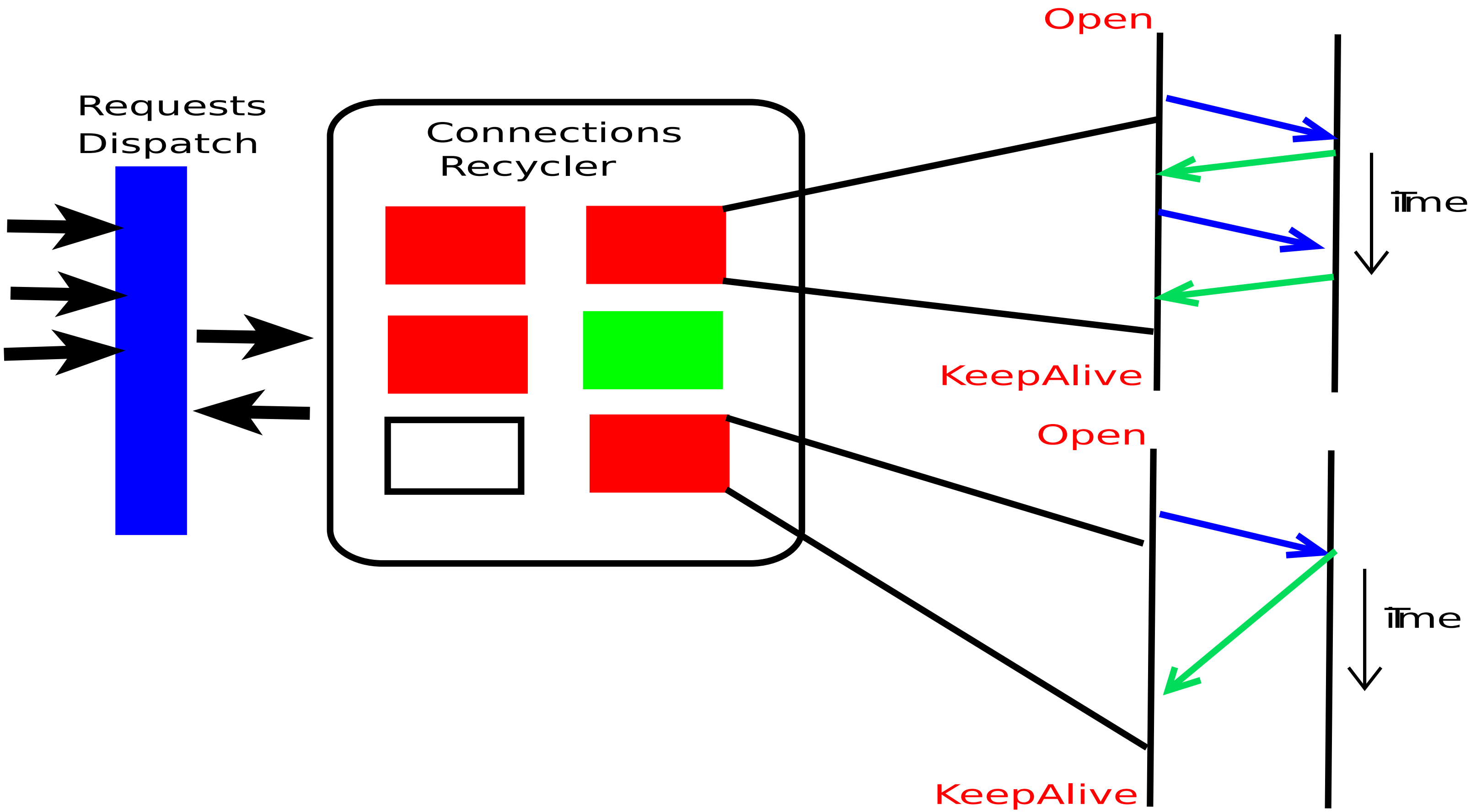


→ **Request Pipelining are in order**
→ **Request Pipelining introduces latency**

Optimization: recycling and request dispatch pattern

- **Maximize the usage of each TCP connection with KeepAlive**
- **Dispatch parallel queries to different execution queues using a session pool pattern**

Optimization: recycling and request dispatch pattern



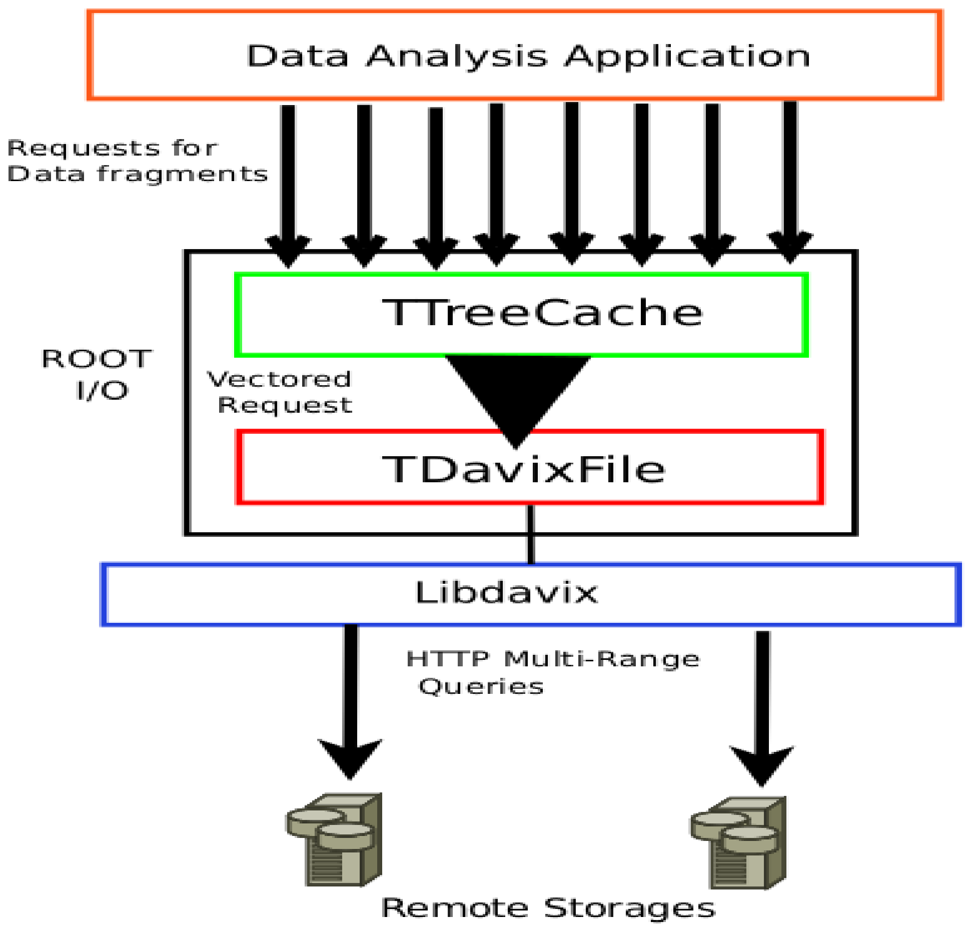
Optimization: Bulk query system with branch prediction

- High Energy physics data are compressed
  - Significant number of little data chunk

- We vectorize sequential I/O operations into Bulk operations
  - Based on HTTP Multi-part content type
  - Vector size > 10000 chunks

- We use a cache with informed prefetching “TTreeCache”
  - reduce the number of network queries.

Optimization: Bulk queries system with Informed Prefetching



Details of the benchmarks (1)

- Execute a HEP analysis job based on the ROOT data analysis framework

- Each job reads 12000 events in a 700 MBytes in a compressed file remotely

- We use for remote I/O
  - XrootD toolkit with the XrootD protocol

  - DAVIX with the HTTP protocol

Details of the benchmarks (2)

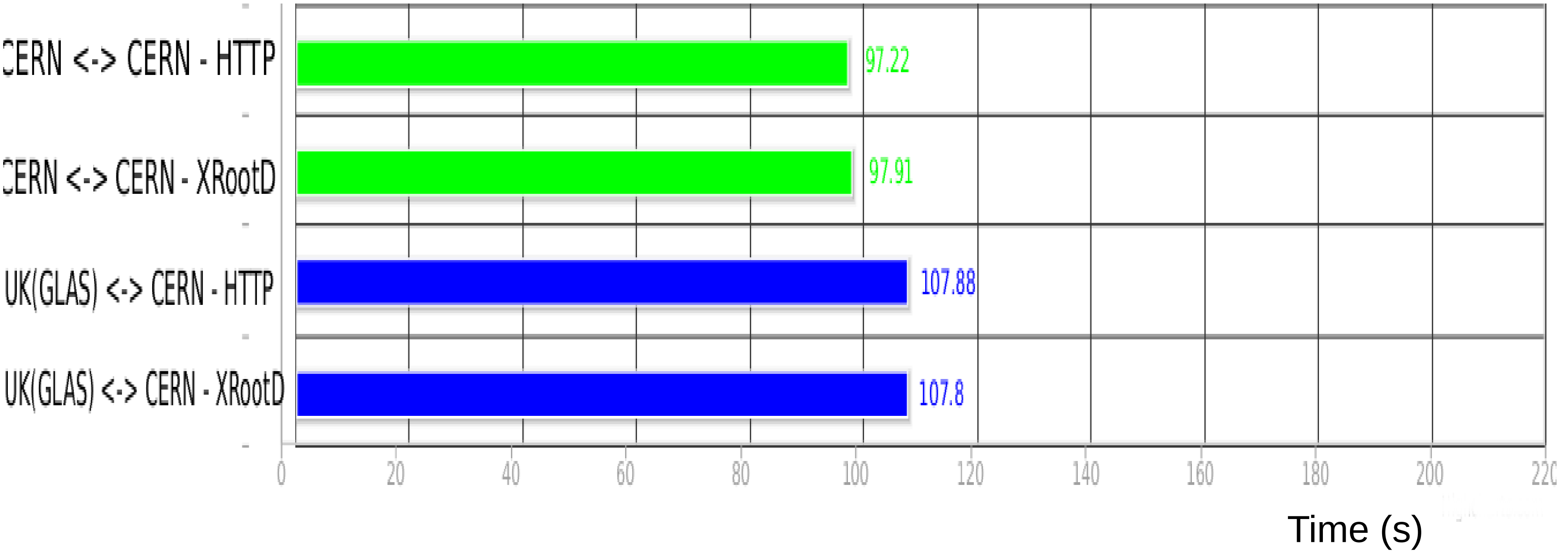
- **Each job is executed with the HammerCloud grid testing framework**

- **Results obtained on 576 run over 12 days**

- **Tests executed against Disk Pool Manager 1.8.8 storage system**
  - **4 Core Intel Xeon CPU**
  - **32 GB of RAM**
  - **1 Gigabit network link**

Performance after optimizations

Average Execution time of the Job



CERN ↔ CERN : Analysis over LAN access

CERN ↔ UK: Analysis over European PAN Network

Problem: Reliability and replicas

- **We are in a world wide distributed environment**
  - **Data object replicas are spread in different datacenters and stored with different Storage system technologies**

- **HTTP is a 1-1 client server protocol**
  - **No recovery in case of server failure**

Metalink and HTTP

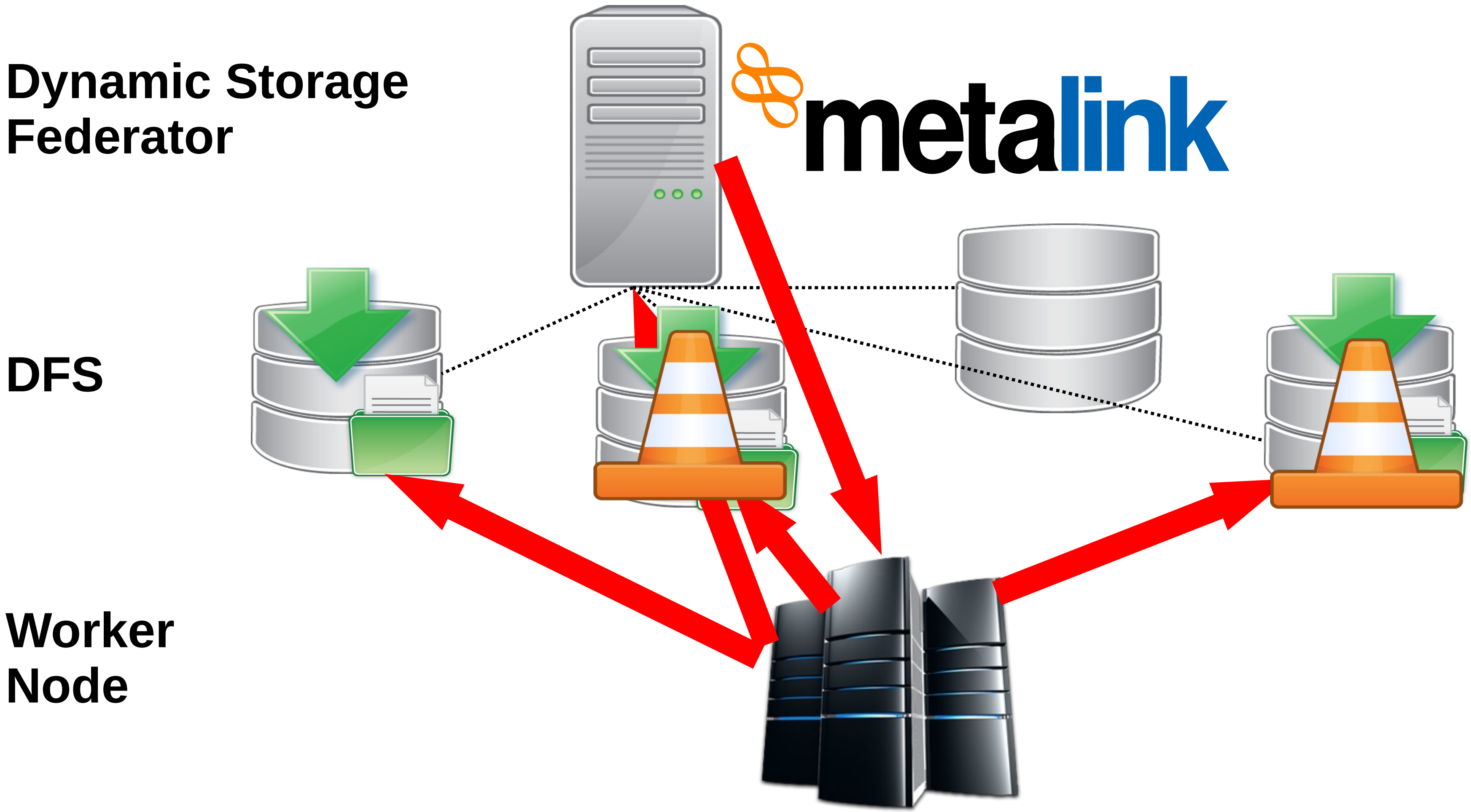
- **Metalink is a standard file format supporting replicas and meta-data descriptions**

- **We use metalink for transparent recovery in case of server unavailability**

```
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
  <published>2009-05-15T12:23:23Z</published>
  <file name="example.ext">
    <size>14471447</size>
    <identity>Example</identity>
    <version>1.0</version>

  <file name="example2.ext">
    <size>14471447</size>
    <identity>Example2</identity>
    <description>
    Another description for a second file.
    </description>
    <hash type="sha-256">2f548ce50c459a0270e85a7d63b2383c5523...</hash>
    <url location="de"
         priority="1">ftp://ftp.example.com/example2.ext</url>
    <url location="fr"
         priority="1">http://example.com/example2.ext</url>
    <metaurl mediatype="torrent"
         priority="2">http://example.com/example2.ext.torrent</metaurl>
  </file>
</metalink>
```

Optimization: Metalink and HTTP for transparent recovery



Optimization: Metalink and HTTP for transparent recovery

- Transparently recovers from a server failure as long as one replica is available world wide

- Multi-stream from different sources based on HTTP

Conclusion

- **HTTP can compete with HPC specific protocols for data analysis use cases.**
- **HTTP weakness in HPC can be compensate with informed prefetching, session recycling and Large bulk operation support.**
- **Reliability of I/O over HTTP in Distributed environment can be greatly improved with Metalink support.**

About DAVIX

- **Offers a I/O and a file management API**
- **Shared Library C++ & set of tools**
- **Already released**
  - **Open Source**
  - **Integrated with the ROOT Analysis framework**
  - **Used by the File Transfer Service of of the Worldwide LHC Computing Grid**

Informations

About Davix

http://dmc.web.cern.ch/projects/davix/home

About our HTTP dynamic federation

https://svnweb.cern.ch/trac/lcgdm/wiki/Dynafeds

About the ROOT analysis framework

http://root.cern.ch/drupal/

Questions ?