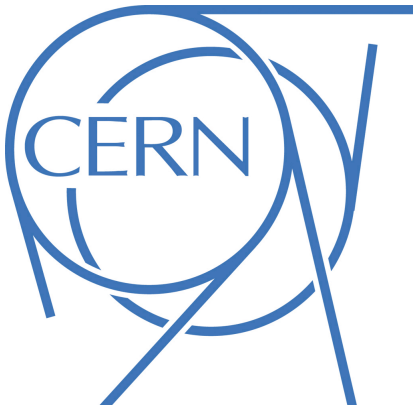
Application Characteristics that Best Fit the CernVM FileSystem and the Frontier Distributed Database Caching System



Dave Dykstra (Fermilab)
Rene Meusel (CERN)

EGI Community Forum—Spring 2014

Outline

- Introduction to Frontier & CVMFS
- Application characteristics with Frontier
  - Those that work well
  - Those that don't work well
- Application characteristics with CVMFS
  - Those that work well
  - Those that don't work well
- Potential enhancements to cover more applications
- Conclusions

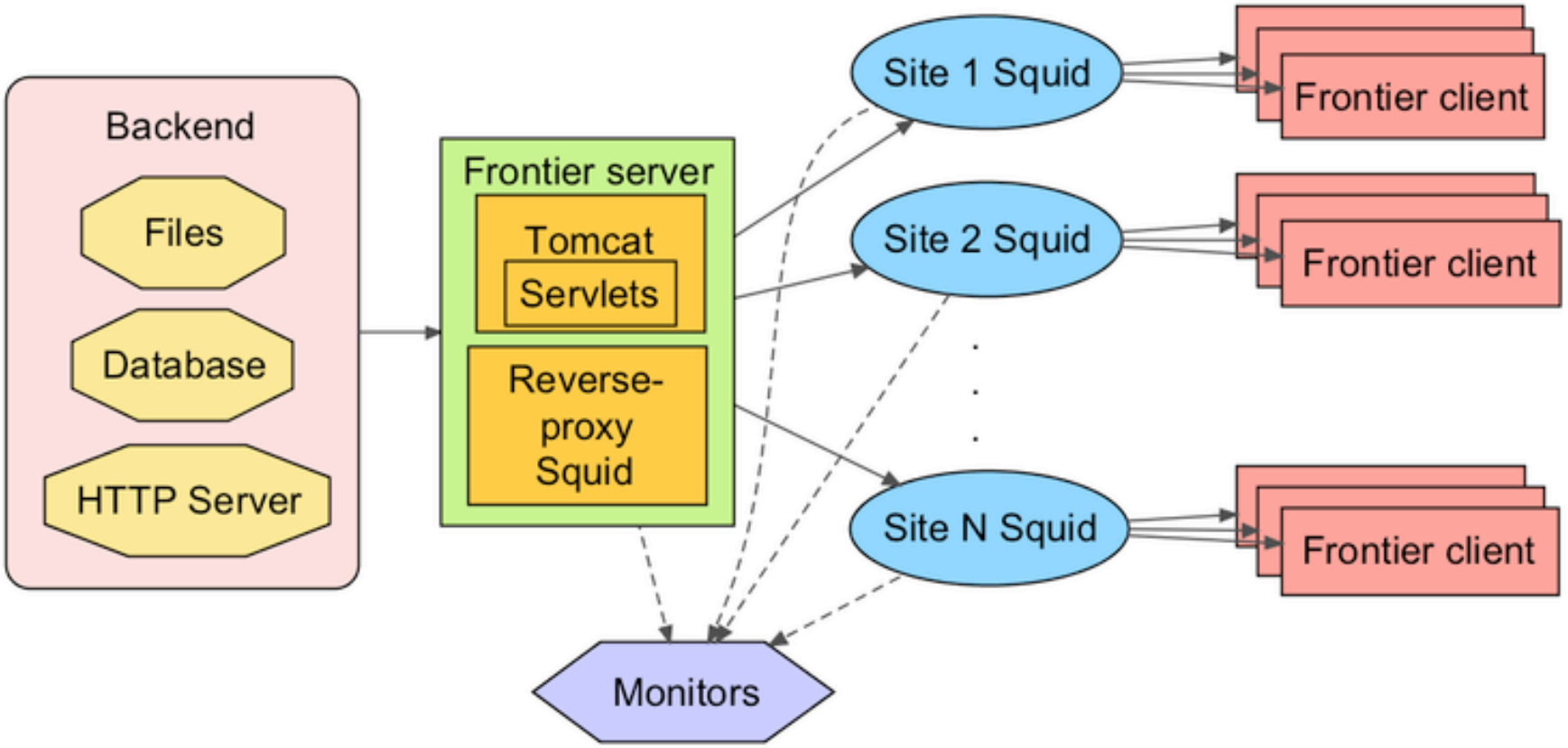
Intro to Frontier & CVMFS

- Frontier & CVMFS both based on using http proxy caches (squids) at the sites where jobs are run
- Both support high volume of simultaneous reads by related jobs, using small number of low volume http servers
- Frontier optimized for HEP calibration data stored in changing databases, CVMFS optimized for code executed by jobs
  - Both can and have been used for additional applications, some work well and some don't
- Both cryptographically verify authenticity and integrity (mandatory in CVMFS, optional in Frontier)
- Both distribute data read-only and have no cryptography-based access control
- Both compress data for transport

Intro to Frontier & CVMFS cont'd

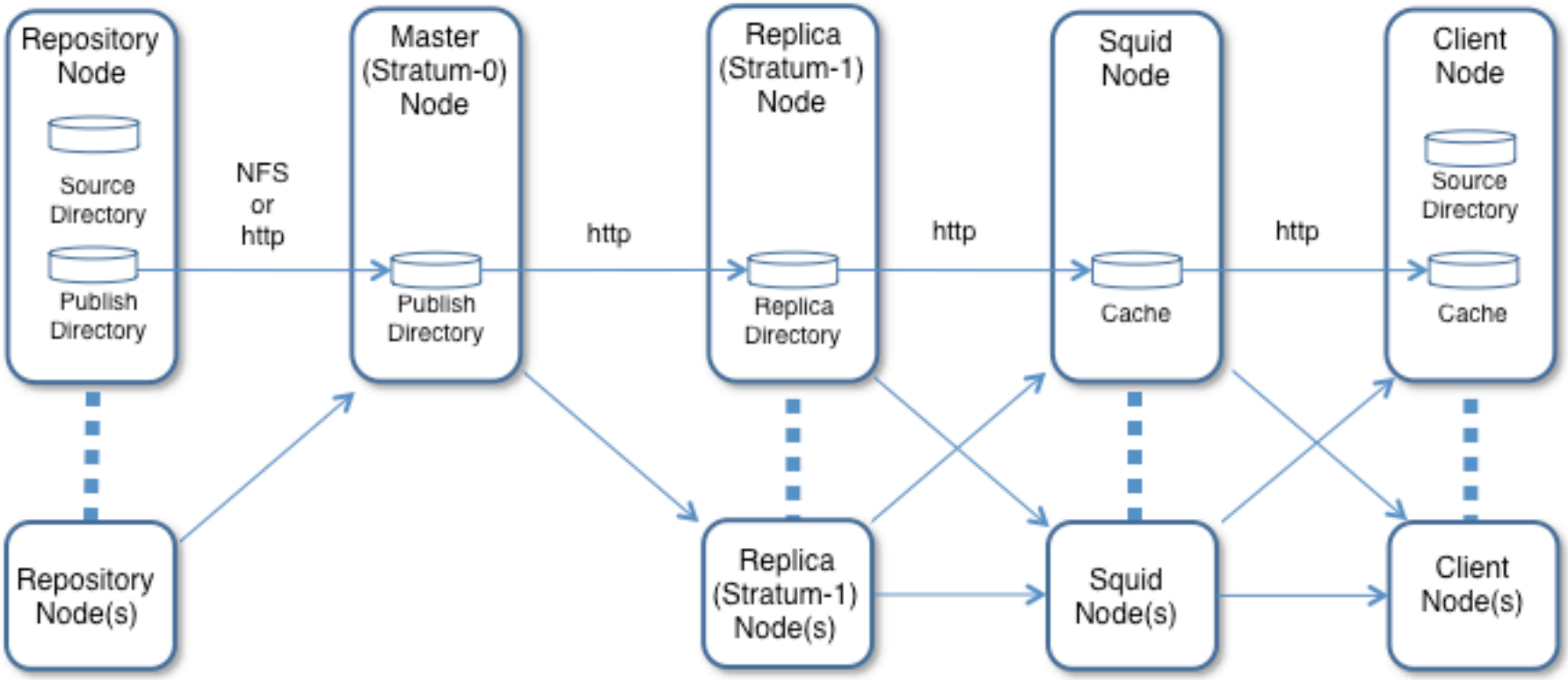
- Both have fault tolerant clients and can do load balancing between proxies
- Both support Proxy Auto Config
- Frontier client application interface is a library, CVMFS client interface is a POSIX filesystem
- Frontier has several powerful mechanisms for cache coherency to manage changing data, CVMFS cached data never changes other than two small files
- Frontier has extendable plugin architecture for application-specific intelligence on the server side
- Frontier server does connection sharing, queuing, sends keepalives

Intro to Frontier



- Frontier has long history: CDF, CMS, ATLAS
- CMS sees 140-to-1 concentration in requests, 1000-to-1 concentration in data on site squids compared to server

Intro to CVMFS



- Files stored compressed, indexed by secure hash of contents
- Metadata stored in file catalogs, put in storage like other files
- Client is fuse-based, with local cache on client node shared between all jobs

Frontier & CVMFS in practice

- Jobs tend to read more data from CVMFS (~GB) than Frontier (~100MB) but because of the shared worker node cache the average squid load is lighter for CVMFS
  - Still, sometimes CVMFS demands more for a period of time when jobs read different data in a short amount of time
- Sites usually supply squid capacity around a gigabit/s per thousand job slots
  - Sufficient most of the time, if large groups of jobs read the same data so there's a high cache hit ratio
  - Most squids are deployed with disk-based caches that are significantly slower than a gigabit/s and depend on kernel filesystem RAM buffer caches & high hit ratio to reach full speed

Application characteristics that work well with Frontier

- Groups of jobs read the same data close together in time
- Data may change quite frequently
  - There are several very good options for handling cache coherency
- If job starts are staggered, less peak bandwidth is needed
- If job starts are synchronized (like the CMS Online High-Level-Trigger) then more squid bandwidth can/should be supplied by using more squids arranged in a hierarchy

Application characteristics that don't work well with Frontier

- Very large (>200MB) data objects
  - Can overload squids if there are many jobs starting around the same time
  - Can be mitigated by carefully spacing job starts
- Jobs that read different data and have little sharing
  - ATLAS "overlay" jobs and CMS "lumi" jobs do this
  - Can be managed by limiting the number of jobs run at once and by giving them their own "servlet" configured with a low number of parallel connections to back end
  - This still performs better than remote database access

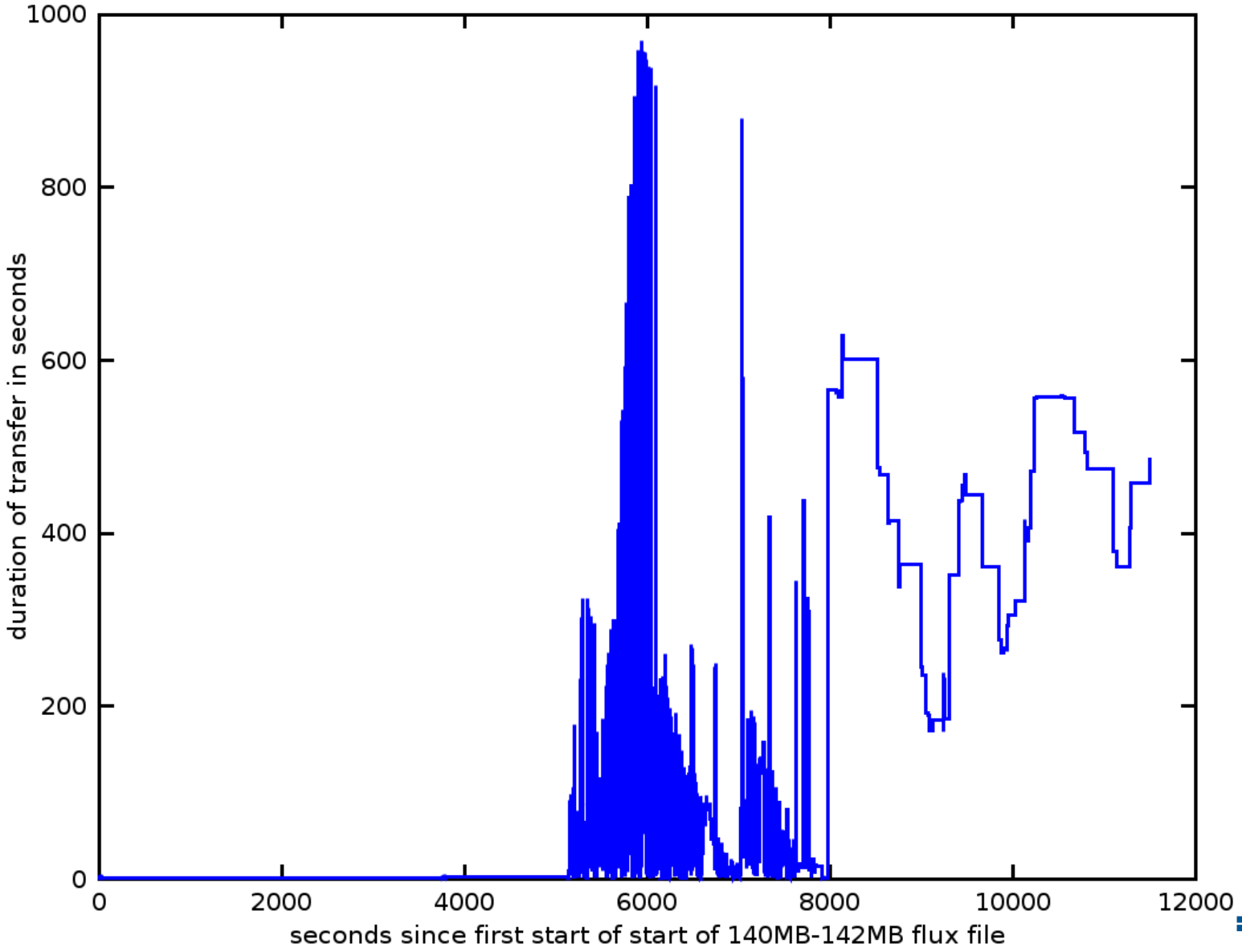
Application characteristics that work well with CVMFS

- Groups of jobs that read the same files close together in time
  - This is generally the case when the files are software
- Files that don't change very frequently
  - Deleted files are not automatically removed from repository
- Groups of jobs that don't use up a large amount of client cache
  - Grid sites configure clients for the software case and so the general recommendation is 20GB of cache regardless of the number of job slots

Application characteristics that don't work well with CVMFS

- Frequently deleting files (e.g. nightly builds)
- Large files overload squids
  - CVMFS 2.1 chunks them, helps especially if access spread through life of job
- Compressed archives (tarballs, zip files)
  - Changing one file inside archive causes whole new copy in repository & stratum 1s
- Large amount of data read at the beginning of the job
- Data files accessed in random order

Transfer times for 128 jobs, 1.8GB per job, 15GB dataset



Fermilab

CVMFS enhancements to cover more applications

- Automated cleanup of garbage files left over in repository coming in a CVMFS release soon
- ‘Alien” cache in CVMFS 2.1.17 may be able to used to automatically share a cache between clients onto existing high-speed file servers (e.g. dCache, Hadoop FS)
- Discussing possible mechanisms for clients to share data peer to peer

Conclusions

- Frontier & CVMFS are great at what they're designed for
- They can be useful for other applications on grids & clouds if care is taken
- CVMFS is being extended to be useful in more situations

- To try either one out, come to hackathon later this afternoon

- More info:
  https://twiki.cern.ch/twiki/bin/view/Frontier/FrontierOverview
  http://cernvm.cern.ch/portal/filesystem/techinformation
    - especially "Limitations on Repository Content"