Scalla: Structured Clustered Architecture for Low Latency Access

Andrew Hanushevsky and Daniel L. Wang*

SLAC National Accelerator Laboratory, Menlo Park, CA, USA

21 May 2012

Outline

Motivation

Initial problem: high-energy physics data in 2002

- Scalability problems with ODBMS (@1PB)
  - High concurrent load (1000s)
  - Large data set (2 $\times$ disk size)
- Unnecessary DBMS overhead

Wishlist

- "Simple," file-access
  - High-throughput, WAN capable
  - Mostly read-only (15% create/write)
  - MSS integrated
- Simplified admin $\rightarrow$ Auto fault-recovery, cluster-configuration
- Bottleneck free $\rightarrow$Burned by lock collisions, metadata servers

General need for distributed, load-balanced tertiary-backed file access

What is Scalla

Scalla:

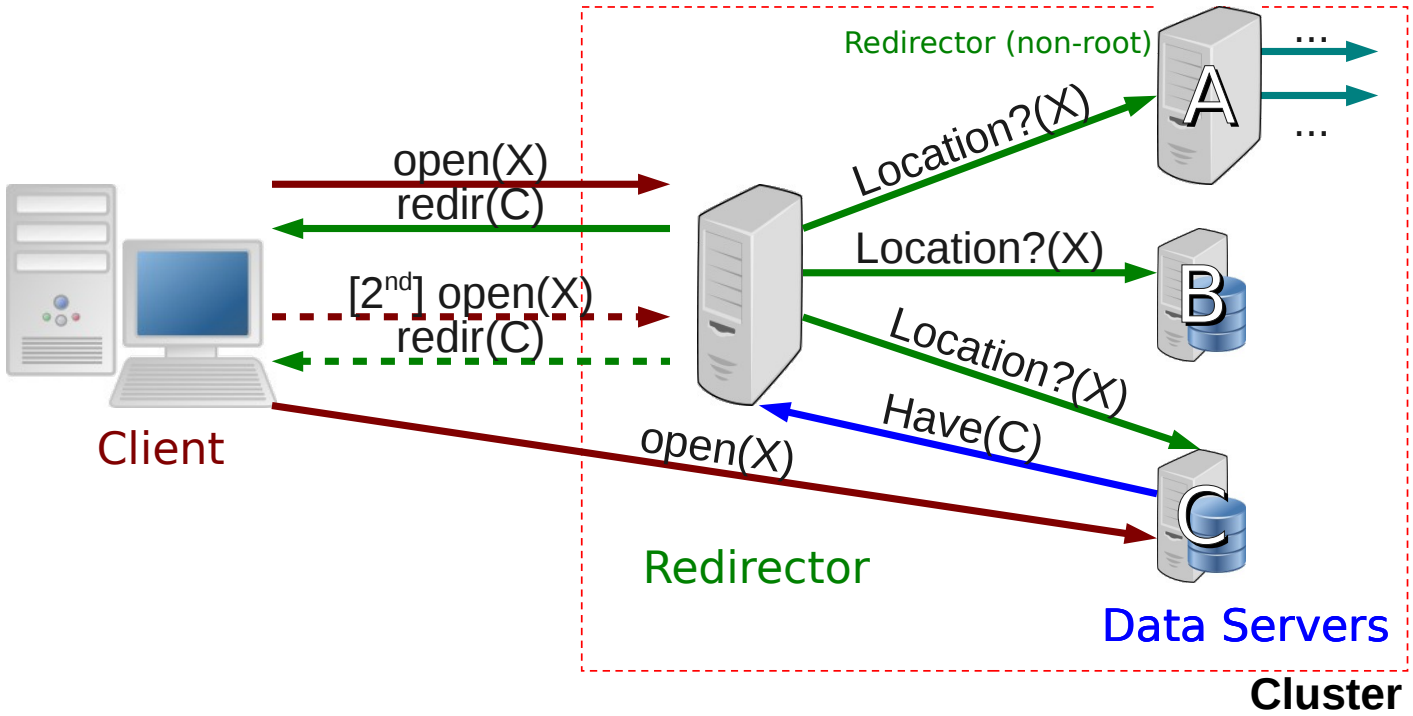
a file access system built as a tree of redirectors and servers

- Decentralized*, distributed → scalabilty, no bottlenecks
  *Except for redirector
- Fault-tolerant → auto-mgmt handles failures and overloads
- Almost stateless
  Cache-only → cheap state changes
  No state for un-requested files
- Minimizes centralized load (redirector)
  Single purpose: redirect

...up to 1000 $\frac{redirects}{sec}$ w/ 400MHz UltraSparcII
...using special data structures

Cluster architecture

64-ary trees of redirectors, servers(leaves)



N servers $\longrightarrow \approx \frac{N}{64}$ redirectors and $\lceil \log_{64} N \rceil$ hops

Data structures

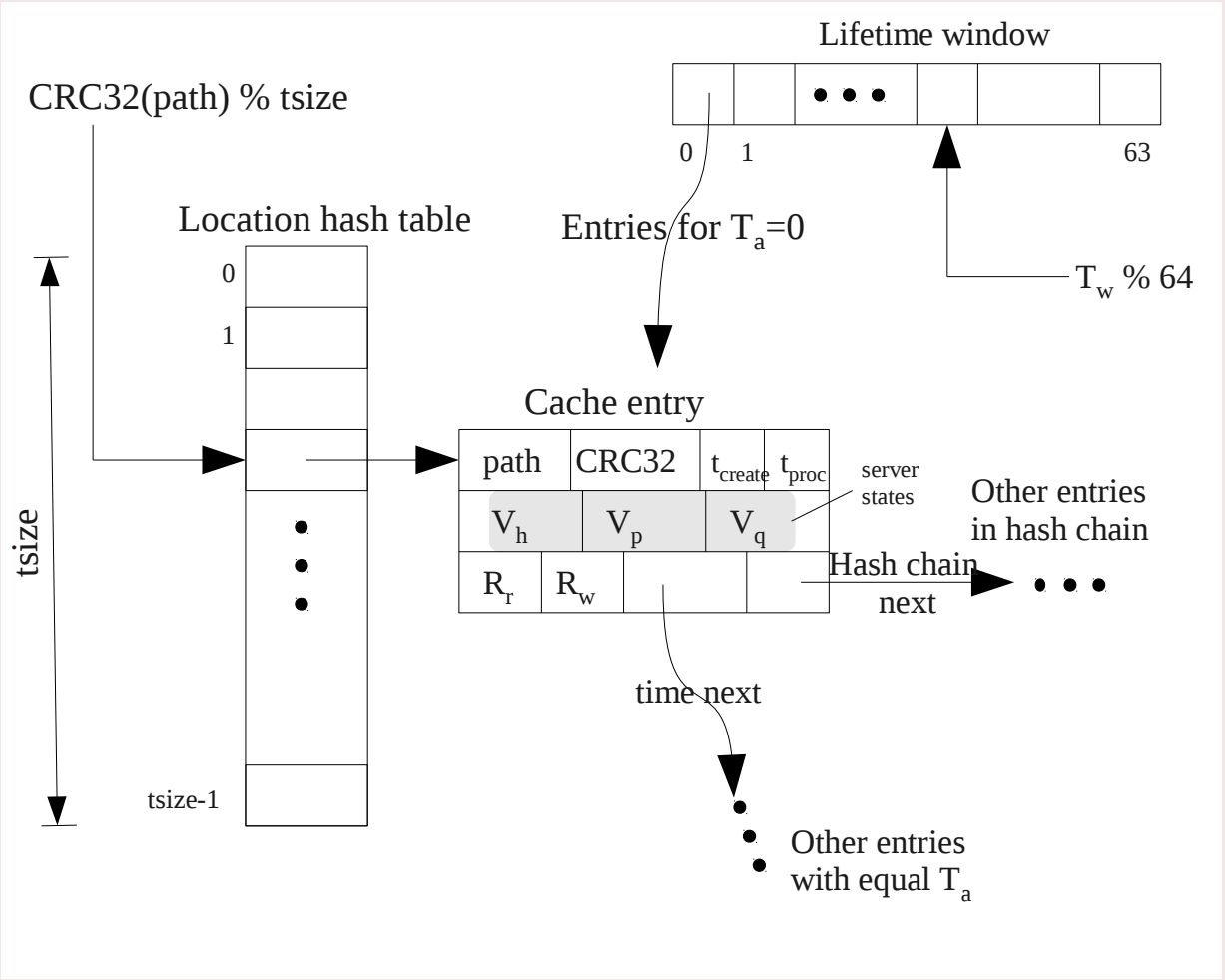
All lookups start at redirector→redirector must be fast

Location cache is the secret sauce

- Leverage past lookups
- Time-based eviction → bounded staleness
- Bit vectors → CPU/cache efficiency
- Lock contention/minimization, some lock-free ops → low-latency, high concurrency
- Integration of lookup, server/location state, aging → efficiency

Caching structure

Location cache with time-based eviction



V_h: has

V_p: in prep

V_q: need query

R_r: idx read resp

R_w: idx write resp

T_w: clock ($\Delta t = \frac{L_t}{64}$)

t_{create}: creation (Δt)

T_a: t_{create}%64

Leverage word-length bit vectors to minimize cost, maximize cache friendliness

Implemented ideas

Idea: Tree flood, ACK-only

(efficiently query many servers)

- "request, rarely-respond"
- All servers searched in $\lceil \log_{64} n \rceil$ hops (3 hops $\rightarrow$ 256K)
- Extremely small lookup cost per-hop
- Simplified file state: down = overloaded $\rightarrow$ not available

  Creation: add'l delay (5 seconds) req'd to avoid collision

Implemented ideas

Idea: Client-directed refresh/avoidance

(reports stale info to the redirector)

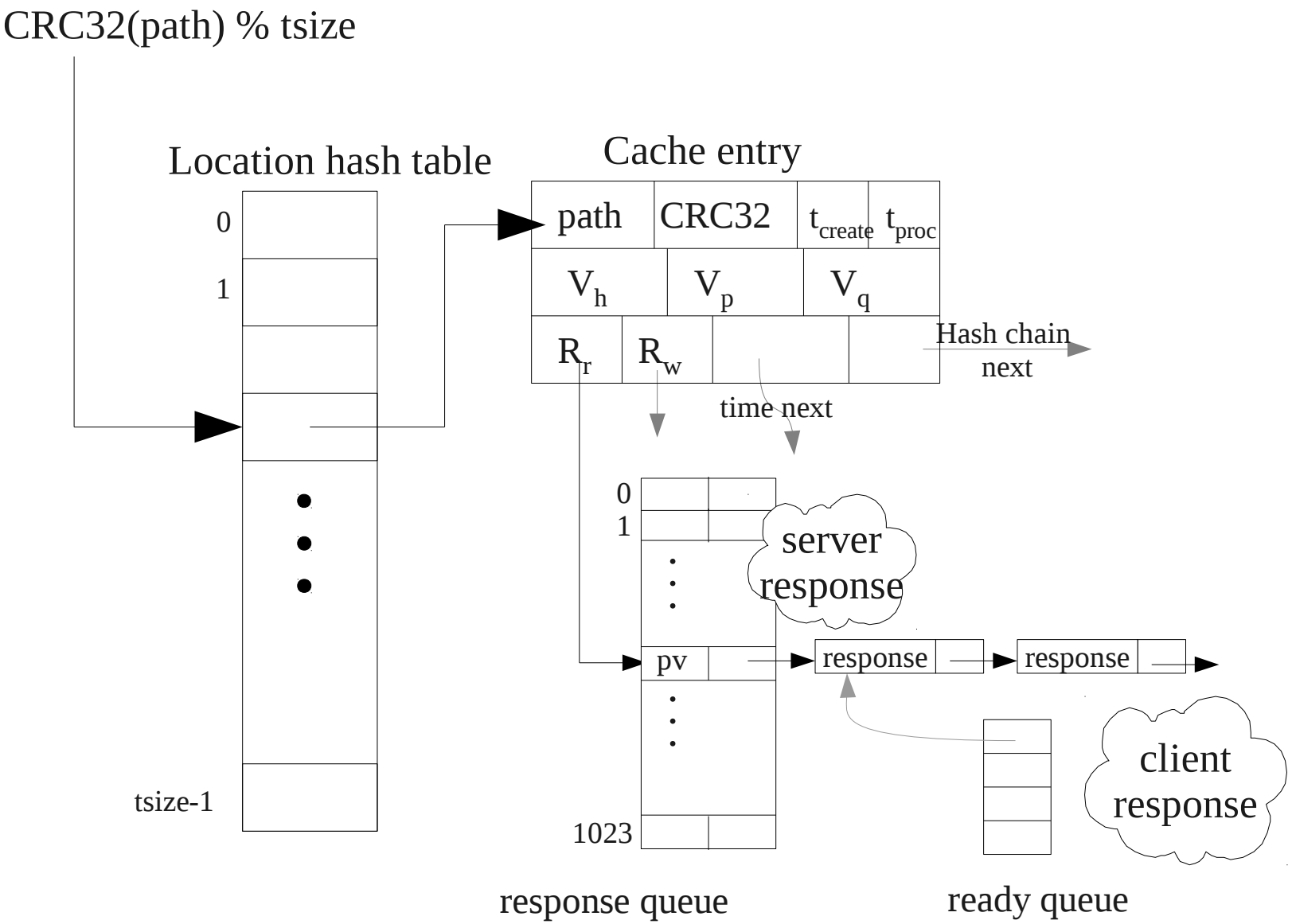
1. Client redirected to server
2. Client receives denial (or timeout) from server X
3. Client requests new lookup from redirector excluding X

Idea: Lookup coalescing

(prevents multiple flood queries for same file)

- Using response queues (R_r and R_w in cache entry)
- Processing deadlines in cache entries

Lookup response processing



Usage: File service

Widely deployed w/ original + 3rd party impl

- FGRST: @SLAC 1.6PB, 47 servers, 2 redirectors, rw)
- BaBar: 3 clusters, 75 servers, 5 redirectors, 1.5PB
- LHC

  ALICE: 43 sites, ≈200 servers
  ATLAS production: 12 US sites

  @SLAC 2.2PB, 3k-4k client jobs, 25 servers,

  US federated storage: 19PB
- STAR @ Brookhaven: 456 nodes

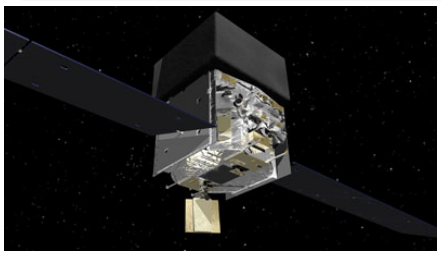
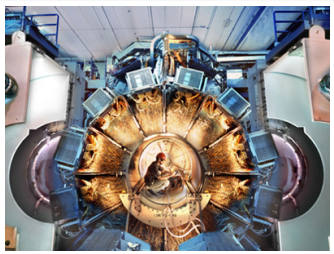
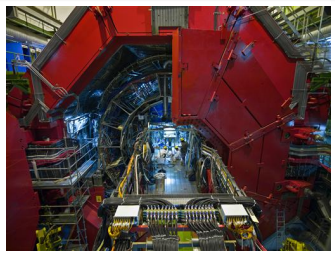
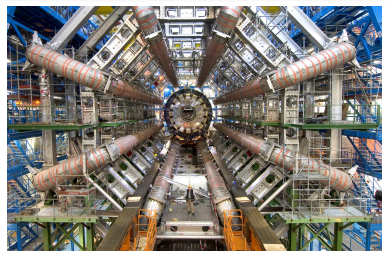
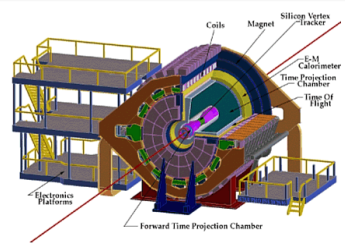






Image credits: NASA, SLAC, CERN, ATLAS Experiment (c)2012 CERN, BNL

Usage: Distributed query dispatch

LSST's Qserv: distributed database

- data partitions → file namespace
- Queries, results
  →open/write/read/close
- Reuse fault-tolerant, mature distributed framework
- Tested: 150 servers, 1+2 redirectors, burst up to 9000

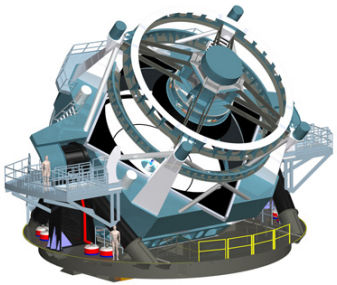
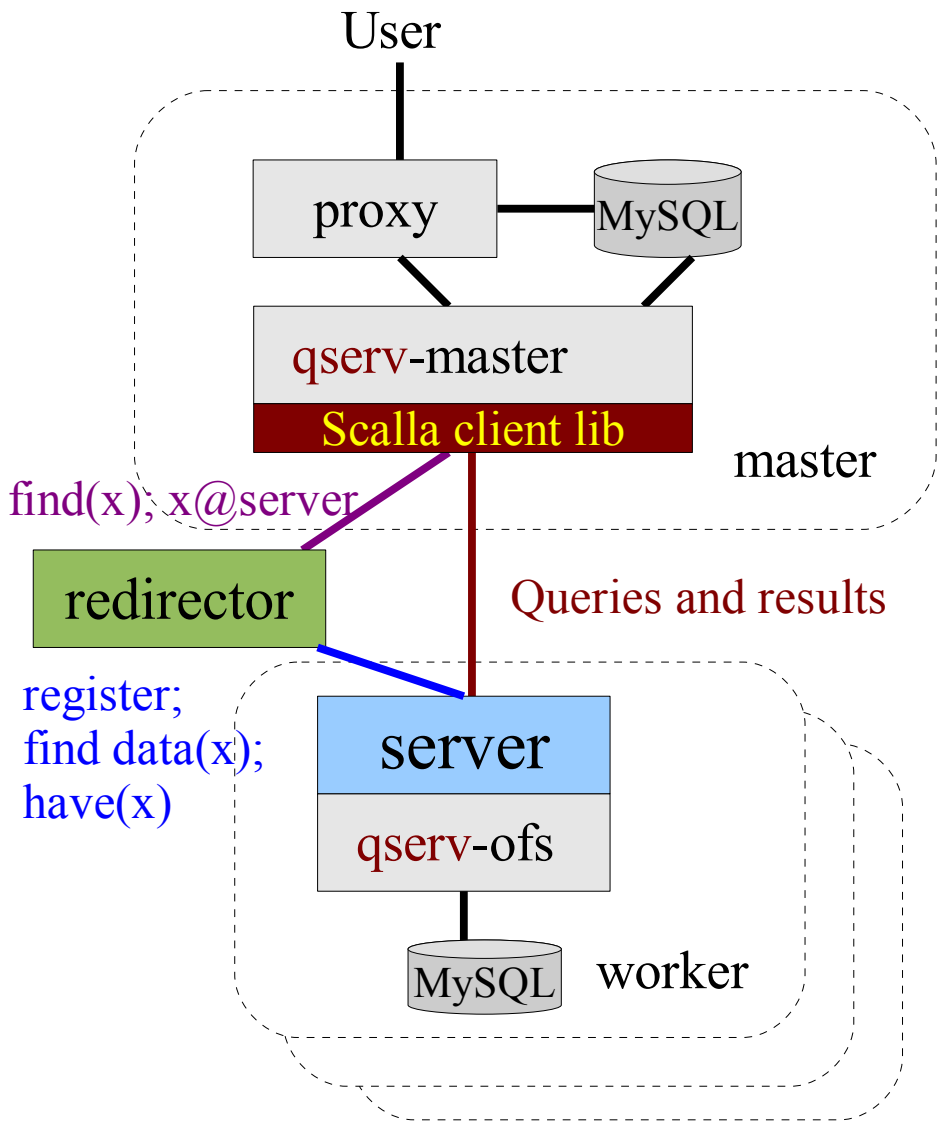


Image credit: LSST Corp / NOAO



Tradeoffs

- Not optimized for home directories
  - No built-in file listing, except with FUSE
- Thicker clients
  - Client mediates lookup and connections w/servers.
  - Most processing: clients + leaf servers (minimal centralization)
  - Complex client protocol (but small enough for feature-phone)
- Decentralized architecture
  - Difficult to gather system state, statistics
  - Neighbor-only connection

Summary

- Scalla provides scalable fault-tolerant file service
- Efficiency provided by integrating cluster mgmt w/ location cache
- Providing HEP/astro/physics file access for 10 years
- Providing clustering/messaging for distributed db (Qserv)

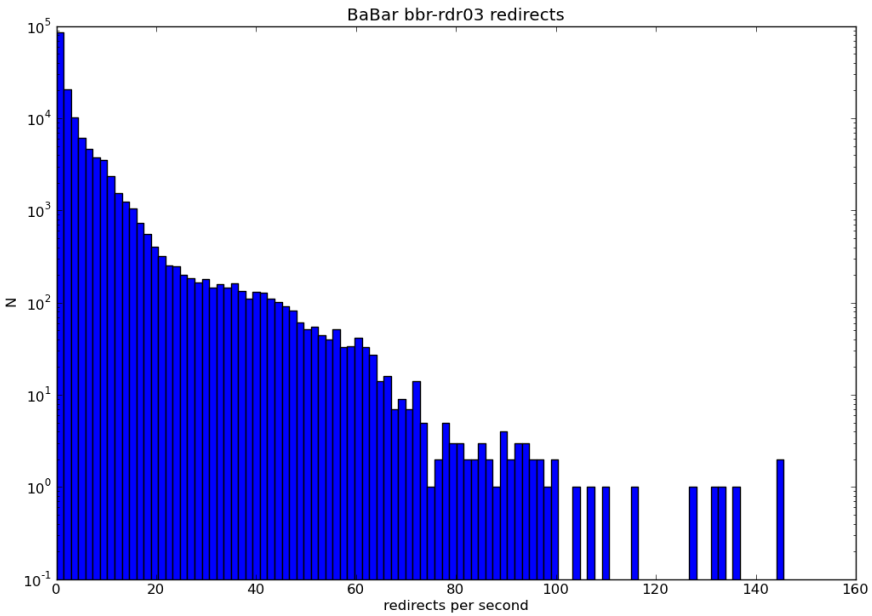
Thank you!

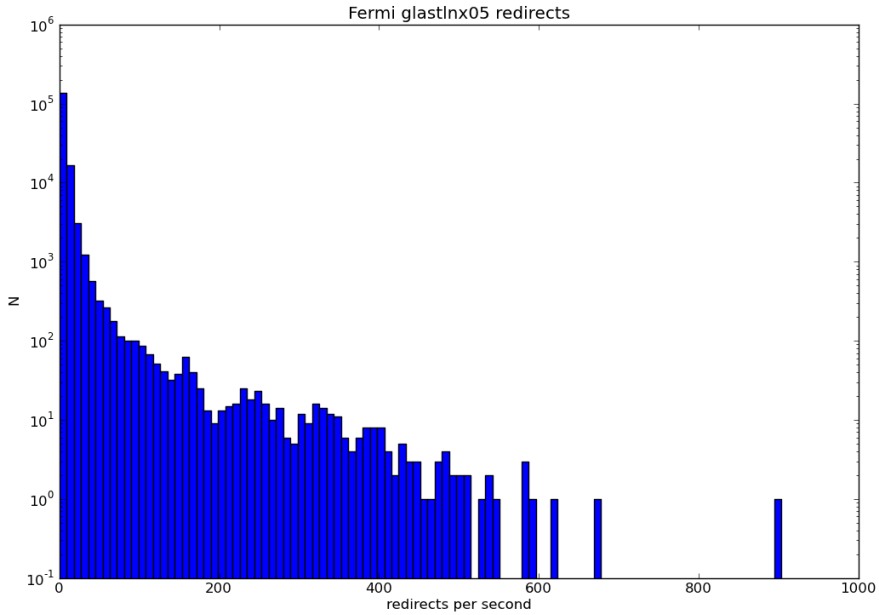
Questions?

- download, refs, slides:
  - `http://xrootd.slac.stanford.edu`
- email:
  - Andy `abh@slac.stanford.edu`
  - Daniel `danielw@slac.stanford.edu`

Thanks!

A day of redirection





Example paths

LHC's ATLAS

root://ccsrb15:1094//pnfs/in2p3.fr/data/atlas/atlasdatadisk/fdr08_run1/AOD/fdr08_run1.0003050....

Qserv path examples

- Sharded query write

  xroot://qsm@mgr:1094//q/rplante_PT1_2_u_pt12prod_im3000/7505

- Sharded result read

  xroot://qsm@172.23.36.70:1094//result/10dbdd8da1a39908cd12b529fd79a7c4