The Power of HTTP Proxy Caches

Dave Dykstra, Fermilab
dwd@fnal.gov

Work supported by the U.S. Department of Energy under contract No. DE-AC02-07CH11359

Introduction

- Talk goal: encourage use of HTTP proxy caches
- Outline:
  - What's good about them
  - Application requirement: REST compliance
  - Frontier database caching
  - Caching file transfers
  - CernVM FileSystem

What's good about HTTP caches

- HTTP is designed for internet-sized scaling
  - Minimal overhead: request/response
  - Designed to be cached
- HTTP caches can be easily inserted whereever repeated requests occur, for better scaling
  - Can be chosen by client (“proxy”) or by server (“gateway”)
- HTTP caches require little maintenance
- Multiple implementations to choose from
  - My favorite is squid

Application requirement: RESTful

- REpresentational State Transfer
- Defined by Roy Fielding in his PhD dissertation
- General architectural style derived from using a subset of http strictly according to HTTP RFCs
  - Roy was a principal author of HTTP RFCs
- Enables scaling, caching

REST essentials

- Essentials for HTTP-based cacheable protocol:
  - Stateless service
    - Every request independent
    - No cookies
    - No https (digital signatures for authentication are OK)
      - Unless just for tunneling to scalable private-net server farm
  - Use HTTP methods as originally intended
    - Don't use POST to pass in complex parameters
    - Use GET with a separate URL for every “resource”
  - Set cache expiration times
    - Varies by application

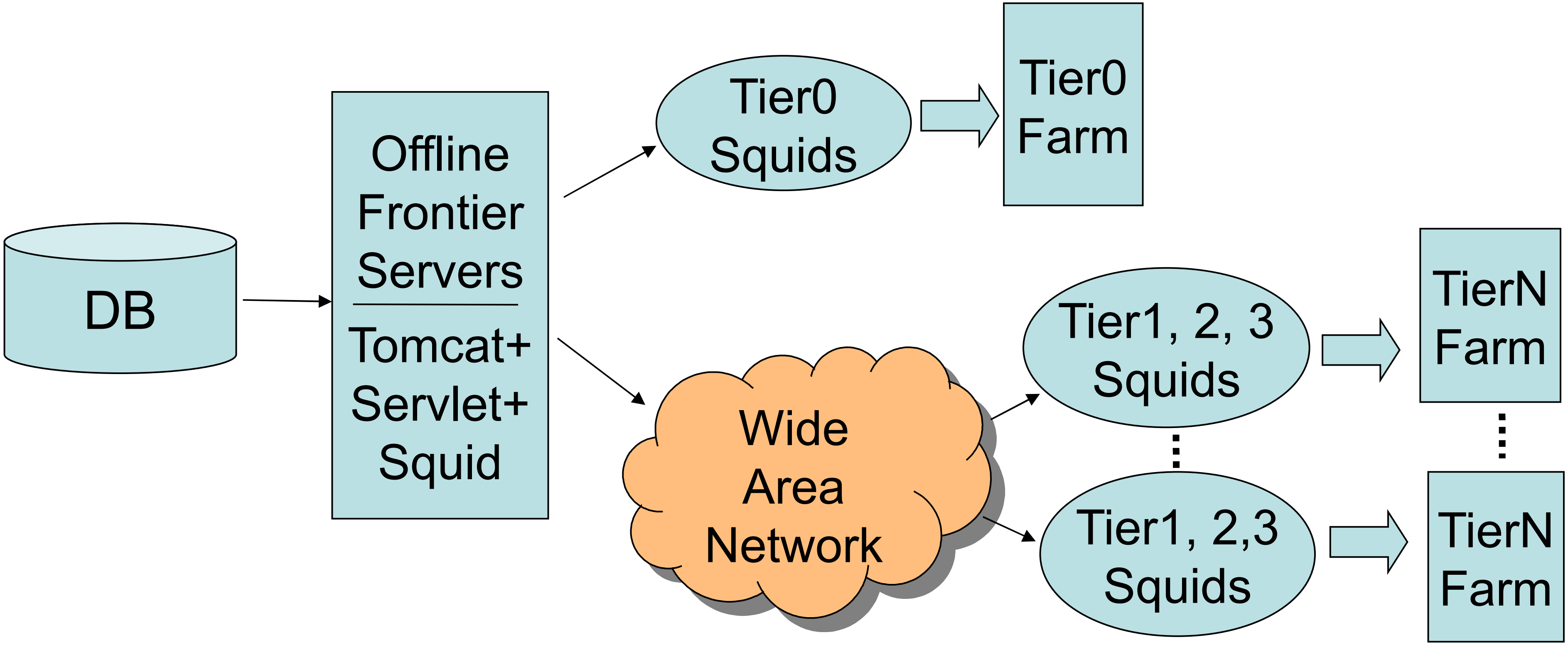
REST importants

- Important but less essential for an http-based cacheable protocol:
  - Use Last-Modified/If-Modified-Since or ETag/If-None-Match
    - Enables revalidating cache with simple NOT MODIFIED response if nothing changed
    - If answer has changed, it is returned immediately with no protocol overhead
  - Don't use '?' in URL
  - Deploy with sufficient caching proxies

Frontier example

- Distributes read-only database SQL queries
  - Updates are done with a different protocol (like most of the RESTful cacheable systems I have seen)
- Developed for High Energy Physics “Conditions” data with many readers of same data distributed worldwide
  - Used in production by CMS Offline & Online, ATLAS Offline Analysis
- Ideal for caching

CMS Offline Frontier example

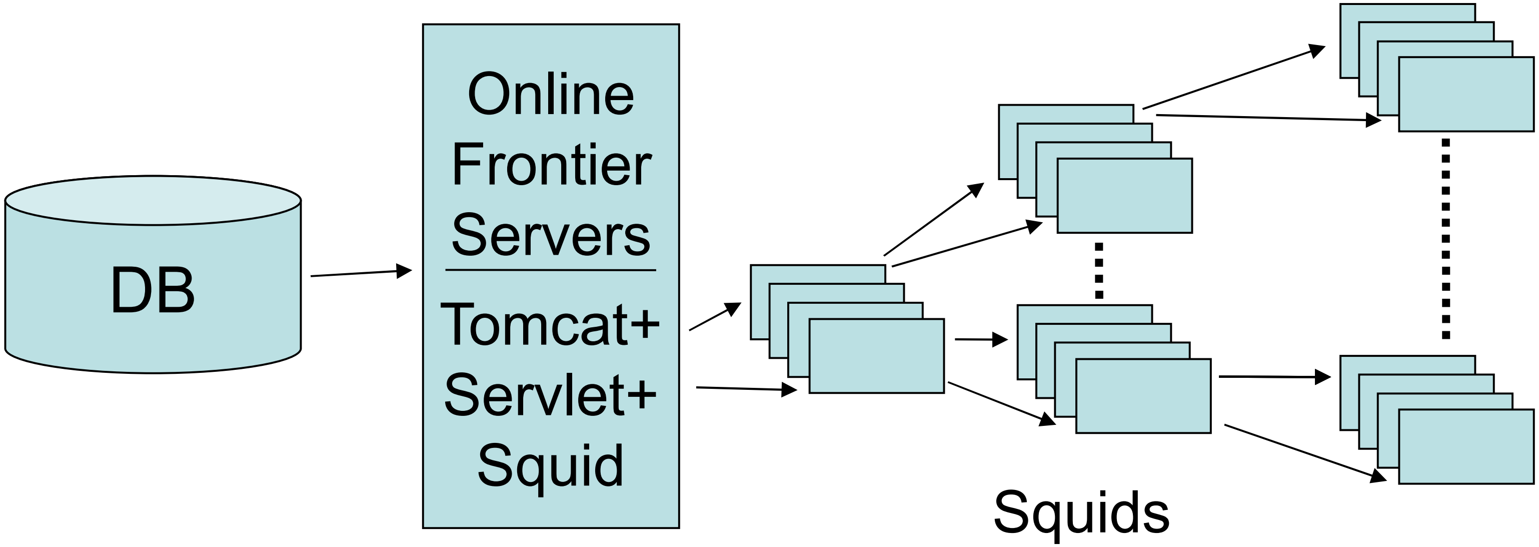


- Many copies of frontier_client in jobs on the farms
- Jobs start around the world at many different times
- Cache expirations vary from 5 minutes to a year

CMS Offline Frontier stats

- Average of 250 job starts per minute worldwide
- Average 500,000 total Frontier requests per minute, aggregate average total 500MB/s
- The 3 central squids only get 6,500 total requests per minute, and 0.7MB/s
  - Factor of 77 improvement on requests and 715 on bandwidth
- Vast majority of jobs read very quickly because results are already cached in local squids

CMS Online Frontier example



- Blasts data to all 1400 worker nodes in parallel
- Hierarchy of squids on worker nodes
- Frontier servlet sends “Cache-Control: max-age=30”

CMS Online Frontier stats

- Roughly 100MB of data loaded to all 1400 nodes in parallel in about 30 seconds, effectively an aggregate of almost 5GB/s
- Cache expires in 30 seconds so every run start verifies that every query is up to date
  - Most of the time, most of it is up to date so very little is actually transferred over the network

Caching file transfers

- Simple http file transfer, via for example apache and wget, can also benefit from same caching
- Last week Frontier servlet & client were expanded to also transfer and cache files
  - Advantages: robust retries on failures, and easy to add to existing server & squid infrastructure

CernVM-FS

- CernVM File System (CVMFS) designed to distribute slowly changing filesystem of software
  - Mostly only additions, not changes
- HTTP URLs secure hash of **contents** of files
  - Once cached, they never change
  - Detection of tampering is trivial
- Indexes map filenames to hashes
  - Digitally signed to prevent man-in-the-middle attack
- Accesses from local squid almost as fast as local disk

Summary

- Use HTTP proxy caches in your applications
  - Whenever the same information is needed in many places
  - Use locally deployed standard caches
    - Already deployed at most sites participating in LHC experiments
- Deploy squids at all OSG sites
  - Suggest two types: production & opportunistic