Update on
The Frontier Distributed Database Caching System

SCD Cross-Division Project Meeting
July 24, 2012

Dave Dykstra
dwd@fnal.gov

Outline

- What is Frontier?
- Who uses it?
- CMS Offline & Online deployment
- My involvement
- CMS Offline performance
- Squid infrastructure
- Future plans
- Relationship to NoSQL databases

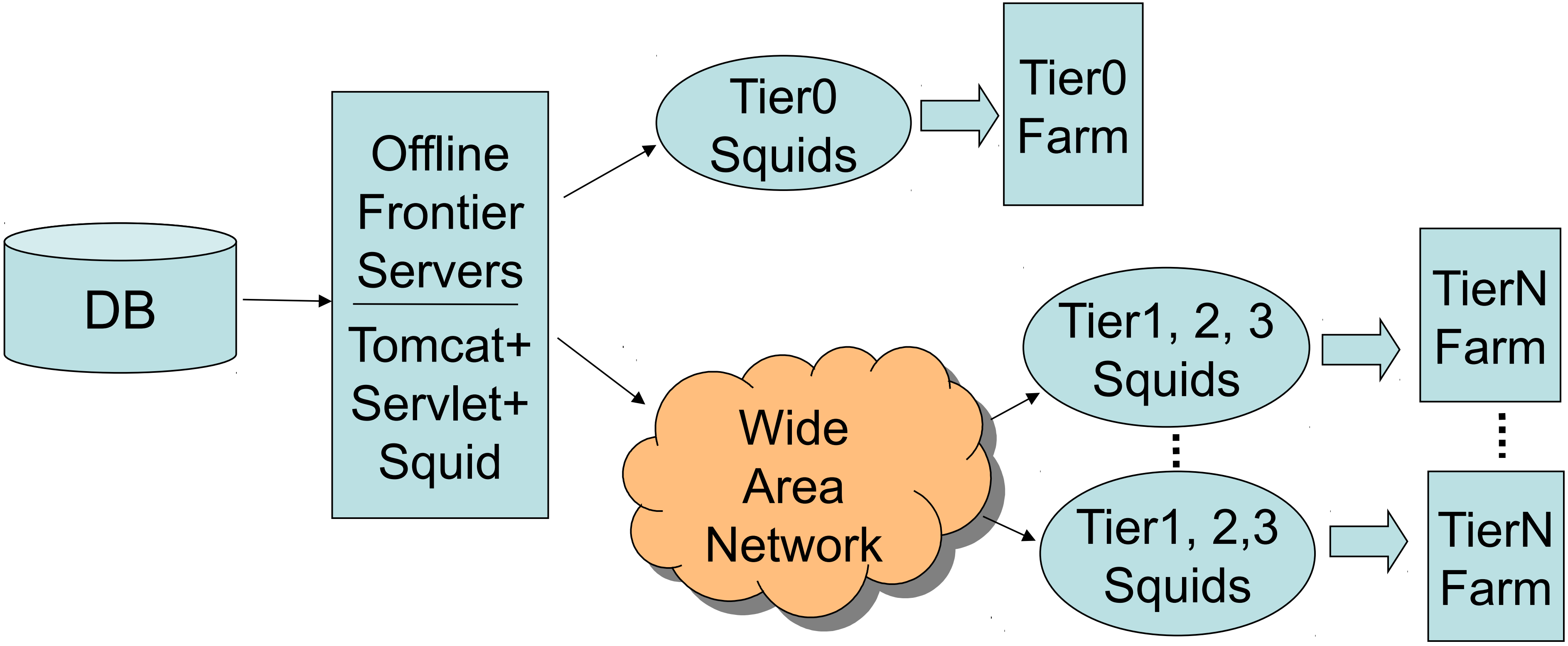
What is Frontier?

- Read-only cached distributed access to public relational databases
- Ideal for loading the same data to many parallel jobs at close to the same time
  - Primary use is for HEP conditions data
- RESTful protocol
  - http according to RFCs, to enable efficient caching
- Uses standard web proxy caches at all sites
  - Squids, monitored by standard tools
- C/C++ client, java servlet in Tomcat server

Who uses Frontier?

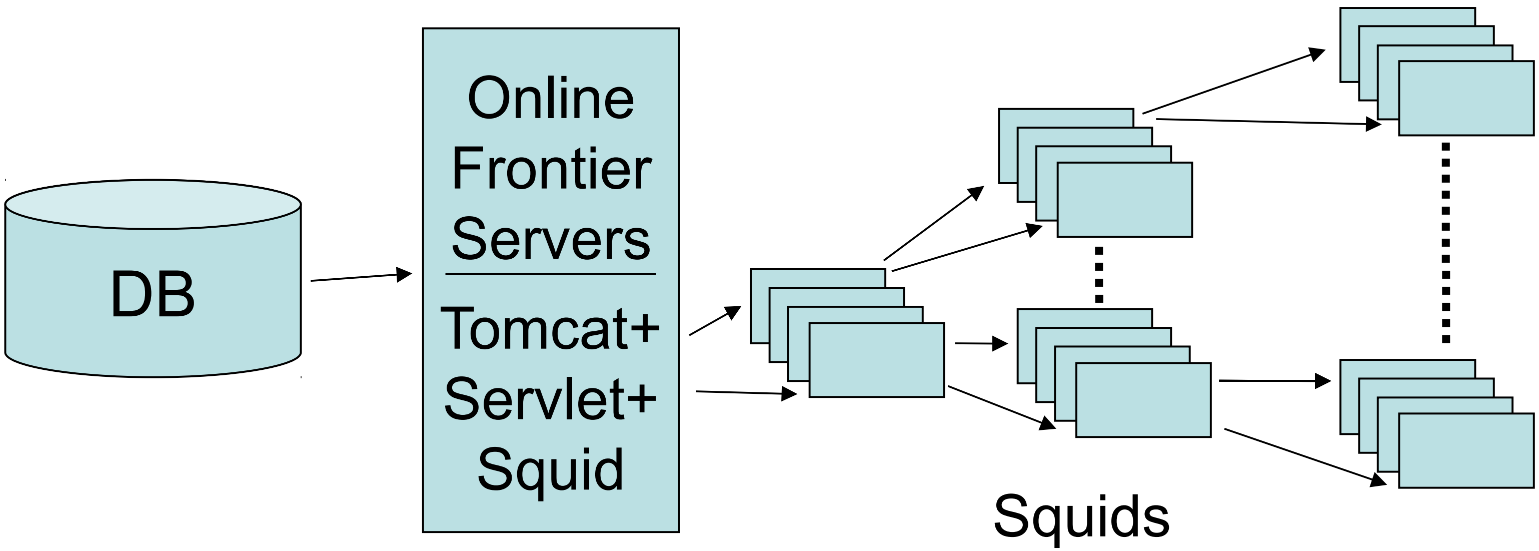
- Invented for CDF who still uses original version
- CMS adapted it for use under CERN's Common Relational Access Layer (CORAL)
  - sql queries, easily switch between database types
- CMS has always used it for all conditions access, both Offline & Online
- ATLAS uses it for most of their Offline conditions
  - Started out planning to use only Oracle but found they couldn't do without Frontier

CMS Offline Frontier deployment



- 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 Online Frontier deployment



- Blasts data to all 1400 worker nodes in parallel
- Hierarchy of squids on worker nodes
- 30 second cache expirations

My involvement

- I have made many improvements over 6 years
- Many performance improvements
- Major innovation: modification time tracking
  - Enables better cache consistency: cached items can be checked for update more frequently without overloading infrastructure
  - Unchanged items are quickly revalidated & reused
- Numerous other smaller features, bug fixes & production hardening (surviving errors)
- I have also been quite involved in operations, especially architecture of CMS services

CMS Offline Frontier/Squid Performance

- For Tier 0, 1, & 2 (not counting Tier 3):
  - Average 500,000 total Frontier requests per minute, aggregate average total 500MB/s
    - Bursts at sites are much higher than average
- The 3 central server Squids at CERN only get 4,000 average requests per minute, 0.5MB/s
  - Factor of 125 improvement on requests and 1000 on bandwidth (not counting Tier 3)
    - Difference mainly from modification time tracking
- Vast majority of jobs read very quickly because results are already cached & valid in local Squids

Squid infrastructure

- An important component is the operation of the worldwide Squid network
  - General purpose for REST-compliant applications
  - Robust monitoring in place
  - Generally requires very few interventions
- Used by other applications, particularly CVMFS
  - Caching remote file system, especially well-suited for software distribution
- Expect more applications in the future

Future plans

- Add backup DB for CMS Offline
- Add squids for failovers at CERN & Fermilab
  - To protect central servers from potential overload
- Add digital signatures on answers
- Add Web Proxy Auto Discover (WPAD) support
- Change retry strategy to include all addresses in a DNS round-robin
- LHCb still planning to use Frontier/Squid
- Intensity Frontier experiments not interested

Relationship to NoSQL DBs

- NoSQL DBs have more flexible key-value hierarchies instead of relational row-column
  - The are also well-known for distributability
- Frontier adds distributability to SQL for many readers of same data ("slashdot problem")
- LHC experiments use Hadoop, Cassandra, MongoDB, and CouchDB
  - Hadoop has most potential value
  - CouchDB is RESTful so could also use http proxy caches

Summary

- Frontier is very effective at distributing cacheable relational database data
- It's general enough to be applied in other applications
- Its Squids are useful for many other purposes
- More information: http://frontier.cern.ch
  - Including at bottom of home page my CHEP papers on modification time tracking, use of RESTful protocols for HEP, and comparison of Frontier to NoSQL databases