Life in extra dimensions of database world or penetration of NoSQL in HEP community

Valentin Kuznetsov (Cornell, USA), Dave Evans (FNAL, USA), Simon Metson (Bristol, UK)

History

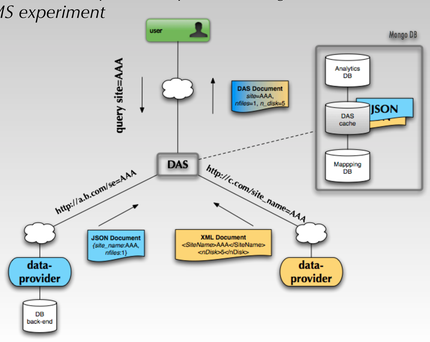
- Current generation of CMS software consists of
  - C++ (framework), python (web, data and workload management), Java (web & data services), perl (web services)
- Major data-services, such as PhEDEx, Data Bookkeeping System, Run Summary are based on RDBMS
  - production: ORACLE, development: MySQL and SQLite
- R&D of tools based on NoSQL stores began in 2009

Reasons to bring NoSQL stack

- Application growth
  - heterogenous environment; evolving data models; big data scale
  - RDBMS may not be best solution, e.g. trade ACID in favor of BASE properties, data evolution/aggregation
  - underlying schema constraints can become real bottleneck
  - high availability vs consistency
  - real time features, e.g. analytics, aggregation, map-reduce
- Horizontal scalability & sharding
  - usage of commodity hardware; search across multiple servers/indexes; distributed map-reduce operations
- Administrative and maintenance cost
  - choice of RDBMS has long term commitment, licensing issues, off-site deployment

DAS w/ MongoDB

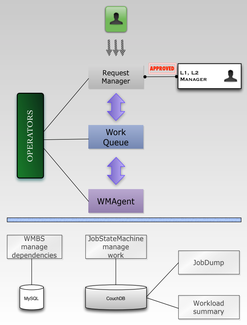
An intelligent cache in front of CMS data-services; fetch and aggregate data on demand upon user queries; next generation of data discovery in CMS experiment



- MongoDB is schema-less document oriented database
  - documents stored as binary JSON; read/write operations are very fast due to memory-mapped files
  - it supports native drivers; multiple indexing; data collections; in-place updates
- Document based queries (on par w/ SQL)
  - Flexible query language; map-reduce; aggregation
- Horizontal scaling via replication and sharding
  - mirrors via LANs and WANs
- Open source; native support for different OSes
- DAS uses MongoDB as cache storage
  - documents from different data-services can be stored and queried together (similar to federated database but does not require a schema)
  - we achieved 20k/7k docs per second for read/write I/O

WMAgent w/ CouchDB

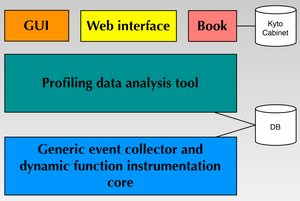
Data and Workflow management tool for job submission and execution engine; dispatch and manage CMS jobs



- Effective key-value store; data in JSON
- MySQL stores job definitions and dependencies, while CouchDB handles job progress, job summaries and output reports
- RESTful HTTP API - in common w/ service we write, no need to maintain DAO's
- Limited relationships between data, map-reduce data look-up is sufficient
  - incremental index building maintains performance
- Replication is built in and very simple
- Back-up in CouchDB is simple due to append only file format
  - can either replicate DB to another node or write DB file to CASTOR
- CouchDB is written in Erlang: high concurrency is natively supported
  - open source; clustering solution exists; commercial support is available

IgProf w/ KyotoCabinet

Main tool for performance tuning of CMS software (core framework)



- O(100) profiles/build, O(100M) of keys
- IgProf uses SQLite and KyotoCabinet
  - SQLite to store build profiles
  - Kyoto to analyze profile results (compare multiple one)
- Kyoto is a library of routines for managing a database
  - choose your DB type based on you app
- It is key-value store
  - very fast, elapsed time to store/search 1M records ~1 sec
  - multi-thread safe, supports transactions and ACID properties
  - written in C++; provides APIs for different languages

Production experience

- DAS aggregates data across dozens of data-services (DBS, Phedex, RunSum, CondDB, etc.)
  - we are able to query over distributed databases using common Query Language and data-service APIs
  - it divides data-service development from common UI/QL; accepts data-service security policies; use different data-formats (JSON/XML/CSV) and legacy/ production APIs
  - **MongoDB** used as dynamic cache, **1M records/day (50GB in size)**
- WMAgent spawns jobs across the globe
  - job bookkeeping is distributed by nature, but replication among data centers is trivial task
  - **CouchDB** is used as distributed storage, **8 instances, 6M+ docs (300GB in size)**
- IgProf keeps our software builds healthy by analyzing in real-time different configurations, measuring and analyzing application memory and performance characteristics
  - **KyotoCabinet** is integrated with release builds; it is used to store build profile results and compare build results/configurations; **zero administration cost**
- CMS maintains NoSQL solutions without central IT support
  - we heavily utilize CERN VM for deployment/integration tests
  - we deploy NoSQL products as any other piece of CMS software
  - 2 years of running shows no significant problems with NoSQL stack

Benefits of (No)SQL

- Schema-less document oriented storage provides new degree of freedom at application level and allowed us to build sophisticated tools
  - shown applications do not require ACID properties and rather favor flexibility of data storage and its high-availability in distributed environment
- Key-value stores can be fast, compact, portable alternative to RDBMS where schema is overkill
- Minimal administrative cost and maintenance

Conclusions

- NoSQL solutions have found a niche in CMS software stack
  - one size does not fit all use cases
- Nicely co-exists with existing RDBMS solutions
  - complement them rather then compete
- Requires minimal administrative and maintenance cost
  - all NoSQL solutions serve as application back-end
  - deployed in virtualization environment (CERN VM)
- Their size is an order of magnitude less of RDBMS usage, but quickly growing