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 PMDGE, Data Bookkeeping System, Run Summary are based on RDMS
  - production: ORACLE, development: MySQL and SQLite
  - IgProf of tools based on NoSQL stores began in 2009

- Current generation of CMS software consists of:
  - JobStateMachine
  - WorkManager
  - JobDump
  - JobDump
  - CERN DB

- Current generation of CMS software consists of:
  - Request
  - AOD
  - OPERS
  - Manager
  - Book

Reasons to bring NoSQL stack

- Application growth
  - heterogeneous environment; evolving data model; big data scale
  - RDMS 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 operations
  - Horizontal scalability & sharding

- Usability of commodity hardware; search across multiple servers/indexes; distributed map-reduce operations

- Administrative and maintenance cost
  - choice of RDMS 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

  - IgProf uses SQLite and KyotoCabinet
  - O(100) profiles/build, O(100M) of keys

- WMAgent w/ CouchDB

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

  - CouchDB is written in Erlang: high concurrency is natively supported
  - Replication is built in and very simple
  - Back-up in CouchDB is simple due to append only file format
  - can either replicate DDB to another node or write DB file to CASTOR

- IgProf w/ KyotoCabinet

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

  - 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

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; accepts data-service policies; use different data-formats (JSON/XML/CSV) and legacy/production APIs
  - MongoDB is 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

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 RDMS 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 RDMS 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 RDMS usage, but quickly growing