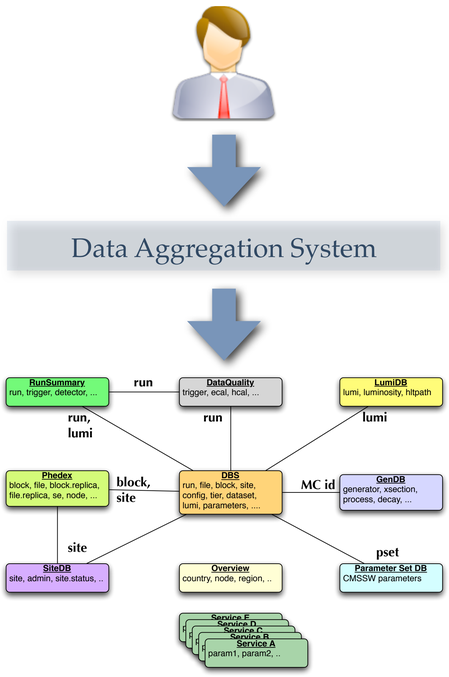
Data Aggregation System, an information retrieval on demand over relational and non-relational distributed data sources

Valentin Kuznetsov, CMS::LHC::CERN, Cornell

Motivations ...

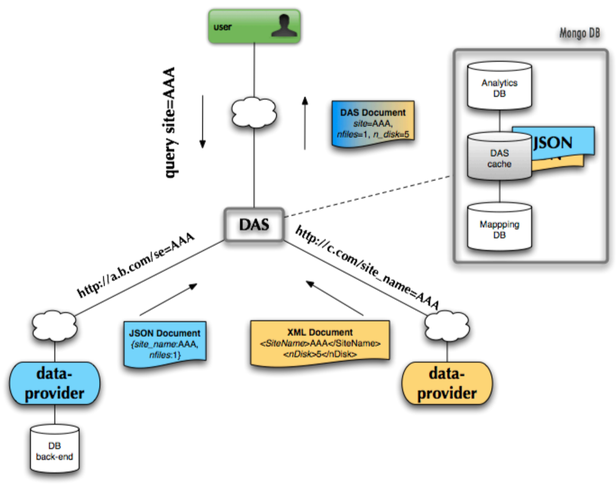
- A user wants to query different data services without knowing of their existence
- A user wants to aggregate information from different data services
- A user has domain knowledge, but needs to query X services, using Y interfaces and deals with Z data formats to get the data



DAS @ CMS :: LHC :: CERN

- 40 countries, 172 institutions, more then 3000 scientists
- CMS experiment produces a few PB of real data each year and we collect ~TB of meta-data
- CMS relies on GRID infrastructure for data processing and uses 100+ computing centers word-wide
- CMS software consists of 4M lines of C++(framework), 2M lines of python (data management), plus Java, perl, etc.
  - ORACLE, MySQL, SQLite, NoSQL

DAS workflow



DAS operation

- DAS works with 15 distributed data-services
  - their size vary, on average O(100GB)
- DAS uses 40 MongoDB collections
  - caching, mapping, analytics, logging (normal, capped, gridfs cols)
- DAS inserts/deletes O(1M) records on a daily basis
- We operate on a single 64-bit Linux node with 8 CPUs, 24 GB of RAM and 1TB of disk space, sharding is not enabled

DAS in nutshell

DAS provides a novel approach to aggregating data from multiple sources without applying any requirements on data providers. Once data is accessible on a web DAS can handle the rest.

DAS leaves data management up to the data-providers. It is true that they know better how to handle, preserve and secure their data.

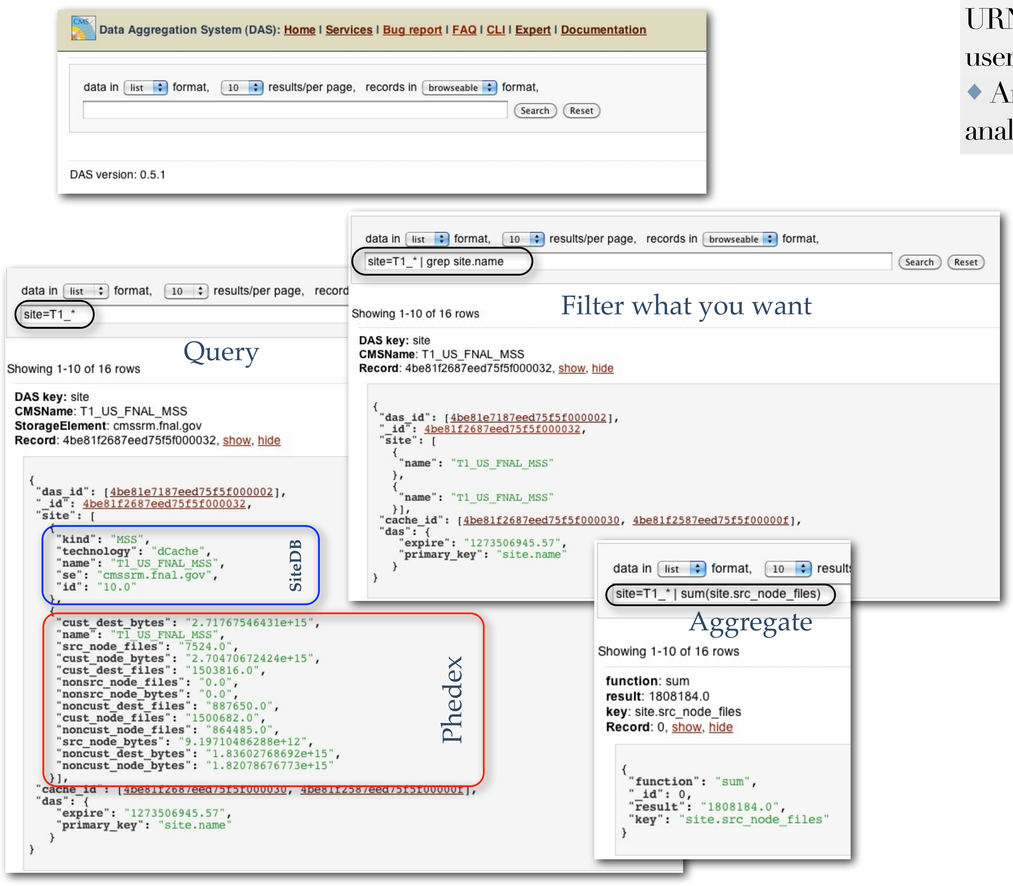
DAS is agnostic to data content. Thanks to NoSQL document-based database MongoDB we're able to store any type of meta-data documents provided by data-providers.

DAS provides a free text-based query language to ease data-lookup. It should be as simple as you search on Google.

DAS uses filters and aggregators to help you navigate through your data.

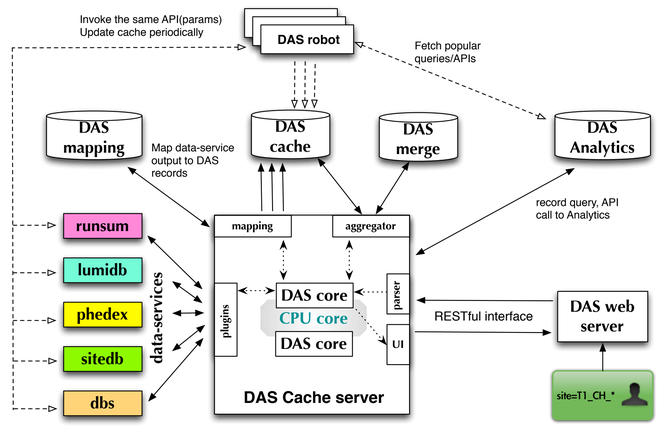
DAS has been developed in CERN-CMS to deal with broad variety of existing distributed data services, majority of them are RDMS based.

DAS interfaces




```
shell# das_cli  --query="summary dataset=/a/b/c | grep dataset.nevents"
```

DAS architecture



- Web server/CLI tool to communicate w/ end-users
- Cache server to handle requests flow
- Cache DB to store results from data-providers
- Merge DB to store aggregated results from DAS cache
- Mapping DB to keep info about data-providers (URIs, URNs, expire timestamps) and DAS keys used by end-users
- Analytics DB to keep track of user requests and query analysis

Logic

To simplify data look-up we used

- Presentation maps for data records, which shows a snapshot of data content, e.g. for site record we only show site name and SE info
- Filters in form of standard UNIX pipes, which select a sub-set of data record
- Aggregators, e.g. sum, count, which allow get snapshots of data
- Map-reduce functions for more sophisticated data analysis

Data from data-providers were converted into common JSON data-format

Data notations has been centralized across multiple data-services by using DAS maps

Tradeoffs

- **Query collisions:** DAS does not own the data and there is no transactions, we rely on query status and update it accordingly
- **Index choice:** initially one per select key, later one per query hash
- **Storage size:** we compromise storage vs data flexibility vs naming conventions
- **Speed:** we compromise simple data access vs conglomerate of restrictions (naming, security policies, interfaces, etc.), but we tuning-up our data-service APIs based on query patterns

Summary

- DAS is data agnostic intelligent cache and aggregation service
- It uses pluggable architecture and allows to work with distributed data-providers without a-prior knowledge of data, schemas, policies and their implementation details
- Data can be aggregated in any dimensions
- Horizontal scale is available