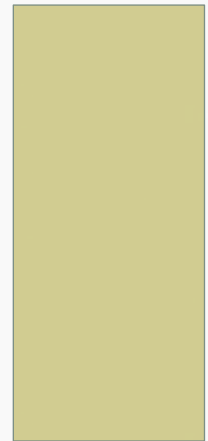


PERSISTENT DATA STORAGE IN CHAOS: SOME CONSIDERATIONS

GIORGIO GAMBOSI – LOREDANA VIGLIANO



MOTIVATIONS

- **Framework**

- Huge amounts of data, such as the ones produced by physics experiments
- Data are produced at very high rate

- **Needs**

- Historical DB to be managed and queried
 - Time series
 - Conditions on data (ex.: out of range)
- High performance, especially in inserting data
- Data access through abstraction layer, to provide physical and logical independence

ACTIVITIES

- **Providing access**
 - defining actions (insert, queries) relevant to CHAOS
 - identifying reliable, efficient OS tools to store and manage data
 - dealing with data modeling issues
- **Integration in CHAOS**
 - providing high-level API's for data access
 - DBMS transparency/independence

DATA ACCESS

- 2 levels
 - Basic (elementary) queries
 - Limited number
 - Implemented on top of storage system API's
 - Higher level queries
 - Composition of elementary queries
 - Composition formalization
 - Composition implemented on top of basic queries

DATA ACCESS (LOWER LEVEL)

- **Query characterization**

- Set of simple & general basic queries
- `CHAOSQL_value([set_of_devs], time_interval, rate)`
 - Returns for each device in `set_of_devs`, the time series of all measured values, at the given rate
- `CHAOSQL_condition(device, condition, time interval, rate)`
 - Returns for the given device, all values measured at times (in the given `time_interval`) when the given condition is verified, at the given rate
- Up to now. Maybe more?

- **Query resolution algorithms implementation**

DATA ACCESS (UPPER LEVEL)

- **Query characterization**
 - Composition mechanism of elementary queries (syntax/semantics)
 - Implementation of complex query resolution algorithms
 - CHAOSQL_fetch(complex_query)
 - Complex_query: AND/OR expression. Composition of queries

DATA STORAGE TOOLS

- Framework has many similarities with systems for managing/dealing with/exploiting web and social networks data
 - Big data (web pages, SN users profiles, logs, ...)
 - High data production rate
 - Statistics and data warehouse/mining
- What we know from those cases?
 - Relational DBMS too complex/too slow
 - Scalability is important

NOSQL

- **NoSQL**

- Key-value
- Simple schemata
- No join
- No transaction
- Large size
- Scalability (Map-reduce)
- Efficiency
- Introduced to deal with big data produced in social networking

ACTIVITIES AND NOSQL

- **NoSQL systems evaluation**
 - We need to compare different NoSQL systems
 - Creating a benchmark
 - Evaluation
 - Efficiency
 - Scalability
 - Functionalities

SW LAYER

- **Abstraction**

- Design and development of sw layer
 - Provide unique interface for data access
 - Independence from underlying DBMS
 - Extendibility to more DBMS

- **Query resolution**

- AND/OR expressions of lower level queries