Database works update

Cristian De Santis
on behalf of
SuperB Distributed Computing Group

4th SuperB Collaboration Meeting
La Biodola, Isola d'Elba
May 31st - June 5th

Presentation Outline

- Introduction
- Database porting to PostgreSQL
- Quality study
- High availability study
- Future works
- Conclusions

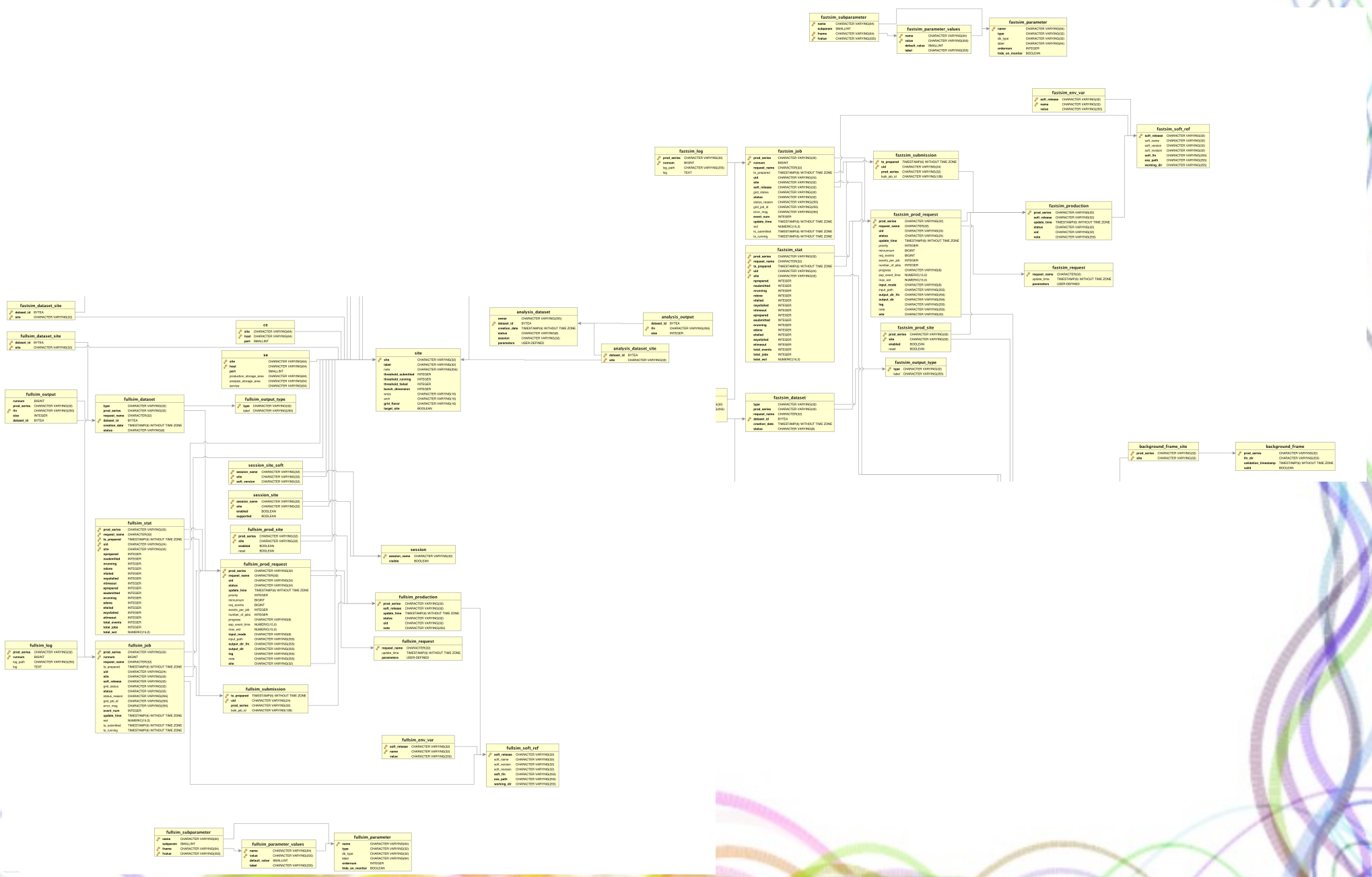
PostgreSQL Porting

- Porting from MySQL (5.1) to PostgreSQL (9.1) decided after the 2nd SuperB Collaboration Meeting
- PostgreSQL is more SQL compliant and, exploiting its hstore data-type, allows to solve some major architectural issues concerning the dataset management of physical parameters
- Extensive tests to check PostgrSQL and HTTP REST interface system robustness have been carried
- During the stress, test up to 100 users*s^{-1} have been created. Each user has carried out a connection and 8 insert/update operations
- Stress test results were good, being the system capable to sustain 10000 DB transactions (being a transaction 1 connection+8 insert/update) in ~100s (~900 operations*sec^{-1})

Quality Study - Introduction

- Normalization analysis of the book-keeping database (*sbk5*) carried out
- sbk5 relies on PostgreSQL 9.1 and exploits *hstore* datatype (n-tuple *key->value*)
- Normalization study concerning database compliance to first three normal forms (NF1, NF2 and NF3)
  - NF1 - Table faithfully represents a relation and has no repeating groups
  - NF2 - No non-prime attribute in the table is functionally dependent on a proper subset of any candidate key
  - NF3 - Every non-prime attribute is non-transitively dependent on every candidate key in the table. The attribute that don't contribute to the description of primary key are removed from the table
- User defined data-types (*e.g.* hstore) not considered in standard normalization theory. *Ad hoc* analysis needed
- sbk5 has a very complex structure (see ERD)

Quality Study – Entity Relationship Diagram



Quality Study - Analysis

- Four logical hierarchical levels have been identified for fastsim/fullsim book-keeping tables:
  - Production
  - Request
  - Submission
  - job+log+output+stat
- Few minor corrections have been recommended in order to make *sbk5* NF1, NF2 and NF3 compliant:
  - few column deletion
  - renaming some columns (*e.g.* uid)
- Specific considerations about *hstore* (thanks to Stefano Dal Prà @ CNAF)

Quality Study - Results

- According to this normalization study, the overall quality of *sbk5* is reported to be very good
- *hstore* fields are accessed by queries on single couple key->value so they are not NF1 compliant (waste of resources). But *hstores* are "*rows with many attributes that are rarely examined*" (~100 updates every 6 months for *sbk5*)
- Trade-off: *hstore* are kept (de-normalized wasting resources) because of its ease of access at very low frequency (only human interaction)
- After some check, suggested modifications will be carried out very soon

High-Availability Study

- High avalaibilty study have been planned for database systems
- Two possibilities:
  - master-slave or master to multiple slaves (Slony-I)
  - clustering
- PostgreSQL has is own built-in Write Ahead Logging-based replication which imposes some constraints:
  - every node must run the same PostgreSQL version
  - everything must be duplicated (specific parts of the changes that are going on cannot be replicated)
  - nothing extra that changes data can run on a WAL-based replica

Slony-I - Introduction

- Slony-I is a *master to multiple slaves* replication system for PostgreSQL supporting cascading (*e.g.* a node can feed another node which feeds another node) and failover
- Slony (or another trigger based replication system) is a better choice than the WAL based replication in PostgreSQL in a lot of practical use-cases:
  - master and slave are on different hardware platforms
  - some additional tables for reporting needed on slave
  - multiple databases on master but only some of them needed to be replicated to the slave
  - For security reasons different table permissions are needed on slave
  - Possibility to take master down for hardware maintenance but after that the master has to take over from the slave without having to re-copy the entire database?
  - replication from A==>B and then have B replicate to C and D?
  - Automatic (Data Definition Language) DDL replication not needed

Slony-I - Limitations

- Slony-I does not automatically replicate:
  - Changes to large objects (BLOBS)
  - Changes made by DDL commands
  - Changes to users and roles
- The main reason for these limitations is that Slony-I collects updates using triggers, and neither schema changes nor large object operations are captured by triggers
- There is a capability for Slony-I to propagate notably DDL changes if you submit them as scripts via a specific script

Slony-I - Tests

- Slony-I has been installed on sb-serv02 (master) and sb-serv03 (slave)
- Master already configured but some minor issues on sb-serv03
- Some mistakes found in Slony-I documentation
- Listen paths between master and slave have still to be defined according to CNAF network
- Failover policies will be carefully planned
- Tests will be carried out on a benchmark database (pgbench)
- Tests are expected to last 2 or 3 months

Future Work

- Documentation writing
- Complete the ongoing work on Slony-I master/slave system
- Study of clustering systems (looking for real-world success stories)
- DB refactoring (at mid-term as in HEP) considering new possible approaches:
  - split in: data placement, analysis, production
  - modelization with no-SQL databases
- Interest in some R&D solutions (XLDB conference in September could be an opportunity for new ideas)

Conclusions

- Database performance and quality analysis completed obtaining excellent results
- High avalaibility studies still ongoing
- Future refactoring and R&D planned