Book-keeping DB work status

Cristian De Santis
on behalf of
SuperB Distributed Computing group

Presentation Outline

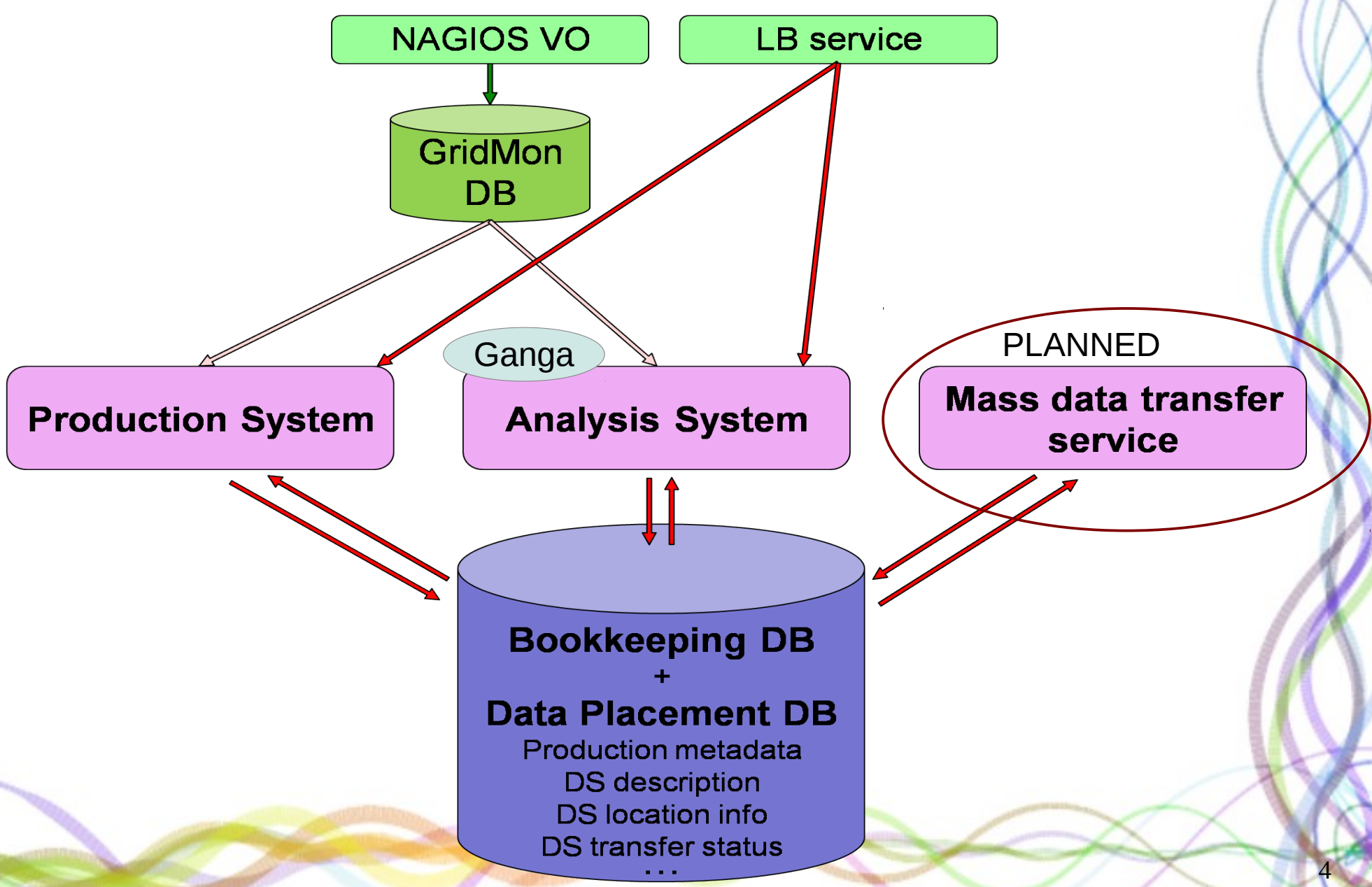
- Introduction
- Database porting to PostgreSQL
- PostgreSQL/HTTP stress test
- Future work
- Conclusions

Use cases

- At present time, the book-keeping database is a key element of all SuperB use cases:
  - Official Simulation Production
  - Analysis/reduction
  - Personal Simulation Production

- Dataset management
  - Monitor, research, creation, deletion, status management
  - Dataset transfer

High level scenario



PostgreSQL Porting - Motivations

- 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
- The new version of the book-keeping database should include the data-placement database too
- From version 9.0 PostgreSQL includes a *"module which implements the hstore data type for storing sets of key/value pairs within a single PostgreSQL value. This can be useful in various scenarios, such as rows with many attributes that are rarely examined, or semi-structured data. Keys and values are simply text strings."*
- Few changes to HTTP/REST interface (http://phprestsql.sourceforge.net/) required

PostgreSQL Porting – Workflow (1)

- As first step, the MySQL *sbk4* book-keeping database has been reproduce under PostgreSQL with few minor changes. An *ad-hoc* HTTP REST interface has been adapted too from the original MySQL one
- Compatibility between the two *sbk4* versions has been successfully tested
- A new book-keeping database (*sbk5*) has been developed from sbk4 under PostgreSQL 9.1 in order to have a robust dataset management both for production (official/personal) and analysis use cases

PostgreSQL Porting – Workflow (2)

- Beside hstore data-type, some other powerful PostgreSQL features have been exploited: its procedural language (PL/pgSQL) and schemas for a better management of user privileges. An extensive use of views and trigger procedures has been done too
- The gridmon database has been ported under PostgreSQL
- Everything can be managed via phpPgAdmin web interface
- HTTP/REST interface and Severus job wrapper have been refurbished to work with Psql
- Minor changes have been necessary for Ganga and Nagios

Book-keeping Dataset Records

Dataset parameters are hstore types.

dataset_id | parameters

4f394214a328d55f29000096 | "dg"=>"DG_4", "tcl"=>"MixSuperbBkg_NoPair", "evt_tot"=>"176320000", "evt_file"=>"80000", "generator"=>"B+B-_Btag-SL_e_mu_tau_Bsig-HD_SL_Cocktail", "prodscript"=>"SLRecoilCocktail", "prod_series"=>"2010_September_311", "analysis_type"=>"SemiLepKplusNuNu"

4f394214a328d55f2900005b | "dg"=>"DG_4", "tcl"=>"MixSuperbBkg_NoPair", "evt_tot"=>"3000000", "evt_file"=>"30000", "generator"=>"B+B-_Kstar+nunu_SL_e_mu_tau", "prodscript"=>"KplusNuNu", "prod_series"=>"2010_September_307", "analysis_type"=>"SemiLepKplusNuNu"

Example of Ganga output:

j.inputdata.getDataset(prod_series='2010_September_311', prodscript='generic')

```
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
| id | prod_series        | prodscript | generator    | dg   | tcl                 | analysis_type | status |
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
| 0  | 2010_September_311 | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | closed |
| 1  | 2010_September_xyz | Generics   | B+B-_generic | DG_4 | MixSuperbBkg_NoPair | HadRecoil     | closed |
+----+--------------------+------------+--------------+------+---------------------+---------------+--------+
choose dataset: 0

Chosen dataset details:
analysis_type: HadRecoil                              |    generator: B+B-_generic
creation_date: 2012-02-13 18:10:49.885510             |    id: 0
dataset_id: 4f394214a328d55f2900003b                  |    occupancy: 121915466273
dg: DG_4                                              |    occupancy_human: 113.5GiB
evt_file: 50000                                       |    prod_series: 2010_September_311
evt_tot: 94500000                                     |    prodscript: Generics
evt_tot_human: 94.5M                                  |    status: closed
files: 1890                                           |    tcl: MixSuperbBkg_NoPair
```

Dataset parameters are defined and stored into the BK DB at the production definition time

Psql/REST Stress Test

- Extensive tests to check PostgrSQL and HTTP REST interface system robustness have been carried out by means of the Tsung tool (http://tsung.erlang-projects.org/)
- Tsung can simulate users in order to test the scalability and performance of IP based client/server applications in order to do load and stress testing of servers. It can be distributed on several client machines and is able to simulate hundreds of thousands of virtual users concurrently
- The REST interface has been configured to establish permanent DB connection in order to save connection slots
- 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 on a sbk4 mock-up database which mimic the real behavior of a production job

Tsung Configuration

```
<tsung loglevel="warning">
 <clients>
  <client host="localhost" use_controller_vm="true"/>
 </clients>
 <servers>
  <server host="localhost" port="8080" type="tcp"/>
 </servers><load loop="2" duration="1" unit="hour">
 …
 <arrivalphase phase="4" duration="3" unit="minute">
  <users maxnumber="10000" interarrival="0.01" unit="second"/>
 </arrivalphase></load>
 <sessions><session probability="100" name="rest_session" type="ts_http">
  <setdynvars sourcetype="random_number" start="1" end="1000000">
   <var name="runnum"/>
  </setdynvars>
  <transaction name="rest_bench_insert_job">
   <request subst="true"><match do="restart" when="match">ERROR.*</match>
    <http url="/resttest/fastsim_job" method="POST" version="1.1" contents="prod_series=2010_July_test\nrunnum=%%_runnum
%%\nrequest_name=d89187c085a819c0647be220caa4feb8\nts_prepared=2011-08-29
18:10:14\nuid=manzali\nsite=VICTORIA-LCG2\nsoft_release=V0.2.5
311\nstatus=submitted\nevent_num=1\nprodscript=HadRecoilCocktail\ngenerator=B0B0bar_Btag-
HD_Cocktail\ndg=DG_4\ntcl=PacProduction">
    </http>
   </request>
  </transaction>
…
  </session>
 </sessions>
</tsung>
```

Tsung Report (1)

Main Statistics

Name	highest 10sec mean	lowest 10sec mean	Highest Rate	Mean	Count
connect	5.46 msec	0.169 msec	737.6 / sec	3.11 msec	267341
page	0.33 sec	57.06 msec	92.2 / sec	0.20 sec	33330
request	40.23 msec	7.03 msec	737.1 / sec	24.46 msec	267341
session	0.33 sec	58.89 msec	91.7 / sec	0.20 sec	33330

Transactions Statistics

Name	highest 10sec mean	lowest 10sec mean	Highest Rate	Mean	Count
tr_rest_bench_finalize	0.12 sec	21.39 msec	92.2 / sec	74.57 msec	33330
tr_rest_bench_insert_job	57.89 msec	7.27 msec	95.8 / sec	28.43 msec	34031
tr_rest_bench_update_job	0.16 sec	28.22 msec	91.9 / sec	0.10 sec	33330

Network Throughput

Name	Highest Rate	Total
size_rcv	1.30 Mbits/sec	59.00 MB
size_sent	1.74 Mbits/sec	78.78 MB

Tsung Report (2)

Counters Statistics

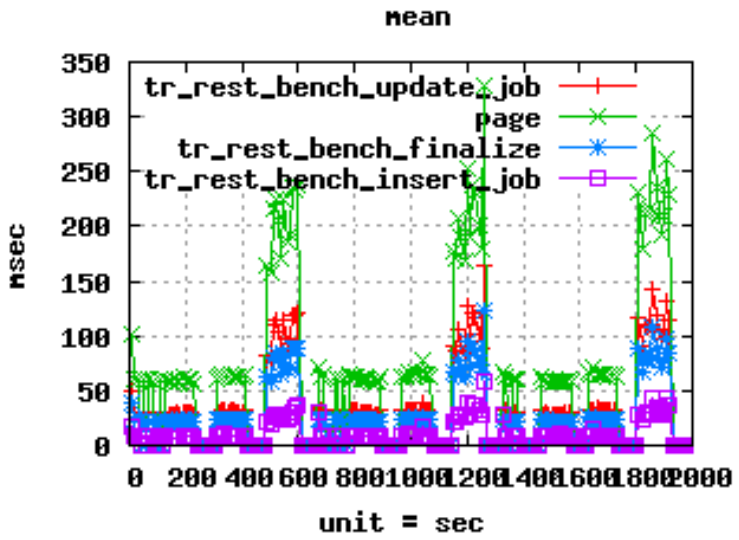
Name	Highest Rate	Total number
match	5.2 / sec	701
match_restart	5.2 / sec	701
nomatch	92.3 / sec	33330

Name	Max
connected	30
finish_users_count	33330
newphase	11
users	58
users_count	33330

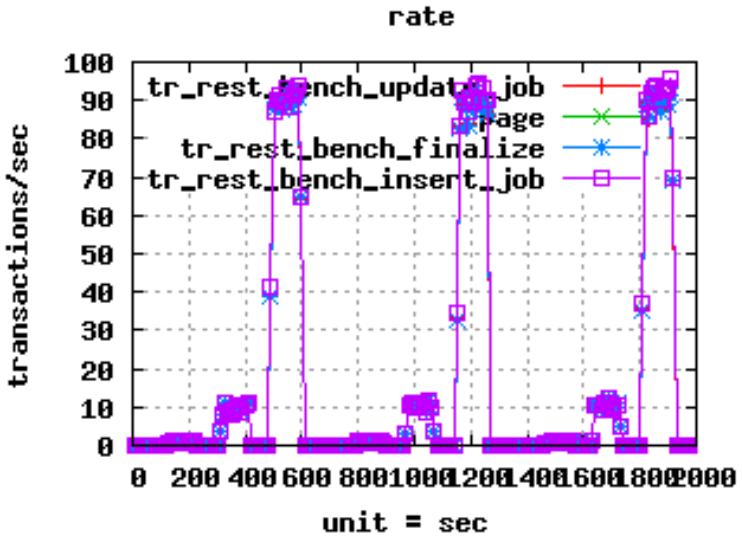
HTTP return code

Code	Highest Rate	Total number
200	5.2 / sec	701
201	276.1 / sec	99990
204	460.2 / sec	166650

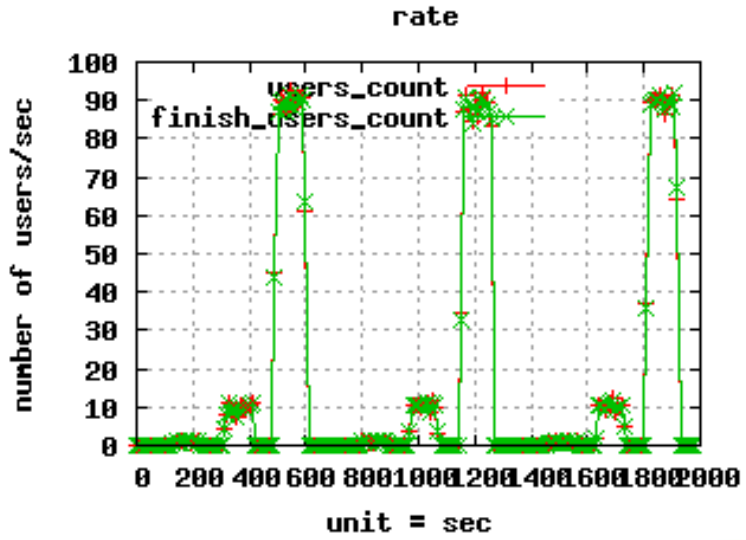
Tsung Report (3)



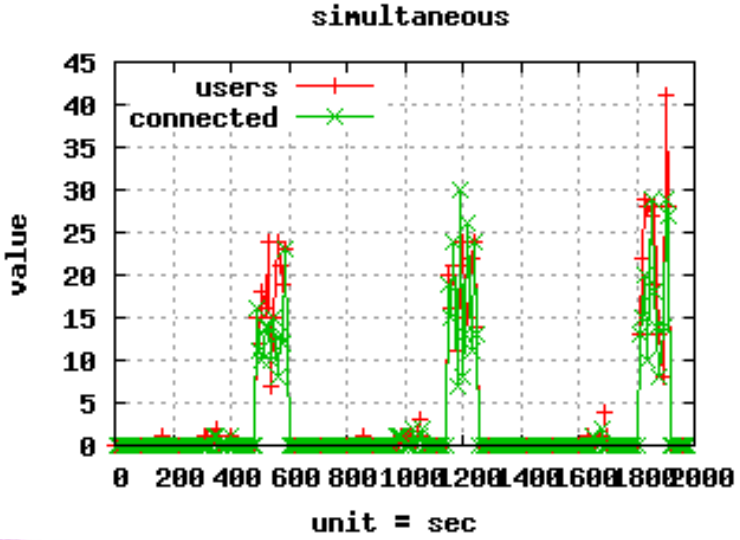
RESPONSE TIME - TRANSACTIONS



THROUGHPUT - TRANSACTIONS



RESPONSE TIME – NEW TSUNG USERS



SIMULTANEOUS USERS – DB USERS

Psql/REST Stress Test Results

- 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})
- Further tests are still on-going in order to obtain better performances by means of a fine tuning of the configuration parameters both for PostgreSQL and Apache servers and kernel parameters too

Future Work

- documentation writing
- detailed quality study (normalization)
- study for a porting in a cluster environment or a *master to multiple slaves* replication system is planned

Conclusions

- A complete porting from MySQL to PostgreSQL has been successfully carried out
- The new system (bookkeeping DB+HTTP interface) is capable to handle the required loads
- A study for PostgreSQL and Apache parameter optimization is on-going but preliminary results are promising
- System quality analysis and future clustering/replication developments are planned