Super-B Databases: Introduction

Igor Gaponenko
SLAC National Accelerator Laboratory, USA

Super-B Computing R&D Workshop
Ferrara, Italy
March 11, 2010

Our plan for today

- **Review and discuss the past and current (state of the art) experience with databases in HEP**
- **Consider trends and risks in a projected timescale of Super-B**
- **Identify those issues which might be relevant for Super-B:**
  - Name and agree upon **most critical** database areas for which we do **NOT** have an immediate answer:
    - either because the area hasn't been well developed/understood, or specific/unique requirements of Super-B, or technology transitions, or computing landscape movements
  - Define R&D to address these problems
- **What's next in this session:**
  - A preliminary analysis (personal view)
  - Talks on the database experience from LHC and BABAR
  - A discussion

Databases in HEP

- **10+ (probably up to 15) years of experience enabled by:**
  - Stable and improving industry standards (SQL, ODBC, etc.)
  - Multiple affordable implementations (proprietary and open-source):
    - **RDBMS**: ORACLE, MySQL, PostgreSQL, etc.
    - **ODBMS**: Objectivity/DB
  - Expertise build-up within HEP
- **Substantial progress in:**
  - Identifying most suitable application domains/areas; successful implementations
  - Learning how to use databases effectively (performance, scalability, evolution, etc.)
- **Some (mostly recently) progress in:**
  - Dealing with distributed databases
  - Application Domain Programming interfaces (good example: COOL, CORAL)

Databases in HEP: known issues

Similar problems for most HEP software

- **Social & Organizational:**
  - Limited cooperation and a knowledge transfer across experiments
  - Very little software reuse (with a few exceptions like ROOT)
  - Duplication of efforts
  - Many HEP databases (schema & contents) is a product of a collective development/use:
    - HEP is much less "controlled" environment compared with industry
    - Developers == Users
    - A tricky business of keeping in sync: **database schema** <-> **contents** <--> **applications**
- **Spatial:**
  - Staying in sync with distributed data production, processing and analysis
- **Temporal:**
  - HEP experiments typically last for many years which makes them extremely susceptible to changes in the underlying databases area (fading/emerging technologies, schema evolution, etc.)
    - This problem is not always appreciated and/or understood, as well as methods which needs to be employed to deal with this.

…known issues (contd.)

- **Technological:**
  - For a variety of reasons, RDBMS (and so ODBMS) alone isn't really a match for many HEP database applications, hence (except really trivial "all in one table" scenario) we end up with:
    - Complex schemas
    - Costly SQL/C++ translation (which has to be keep in sync with schema)
    - Multiple layers on top of vendor APIs
    - Poor performance, limited scalability
    - Various workarounds and solutions to compensate for missing (but desired) functionality
    - And multiple versions of all of those above at any given moment of time
  - Non-existing or limited "out of a box" solutions for distributing RDBMS across sites
  - Security models of many popular DBMS aren't always consistent with our requirements & practice:
    - Are we ready to enter a "special" password each time we're going to update a calibration…from a batch job?

New Computing Landscape

- **New trends & risks affecting Super-B:**
  - A rapid expansion of multiple forms of the distributed computing (**GRID**, **Cloud**, **Folding@Home**)
  - Virtualization
  - An explosion of parallelism at many levels (**clusters**, **multi-core CPUs**, **GPUs**)
- **Consequences:**
  - Dramatically increased complexity of data processing/analysis systems:
    - The famous Moor's law is still in an effect, but now it translates into a number of processes (run on separate "cores"). Will the present database systems scale to handle x10, x100, x1000 more clients in the not so distant future?
    - The propagation of information across a highly distributed system has a limited speed and it's not trivial to deal with. How to ensure a consistency of this process and an overall data integrity?
- **What would be an effect of new technologies? New opportunities or new borders?**
  - No doubt, all of that will change ways we define/use databases for HEP.

What can we do then?

- **(obviously) Better be proactive than reactive**
  - Analyze trends and risks
  - Think on how to mitigate negative consequences(?)
- **The main issue here is how to preserve the "investments" into the code, procedures, people training?**
  - The classical solution from the Software Engineering: "…one more level of indirection solves all problems (for now)…".
  - In our case this recipe would translate into:
    - Properly identified abstraction layers
    - Going from domain specific concepts to a technology (**not an opposite!**)
    - Quality programming & documentation

Suggested R&D topics (ideas)

- **Focus on:**
  - Abstraction layers, Programming Interfaces, Tools
  - Distributed Databases
  - Usage and management models and scenarios

- **Other questions:**
  - Should we consider databases in a broader context of a consistent ("holistic"?) approach to the Super-B Computing Model?
  - What Super-B can inherit from BABAR and LHC?