Next-Generation Databases

Miguel Branco *on behalf of the RAW team*

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

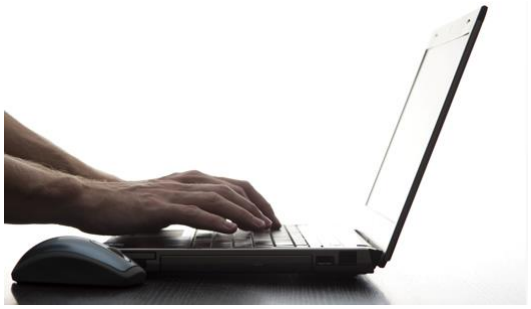
EPFL

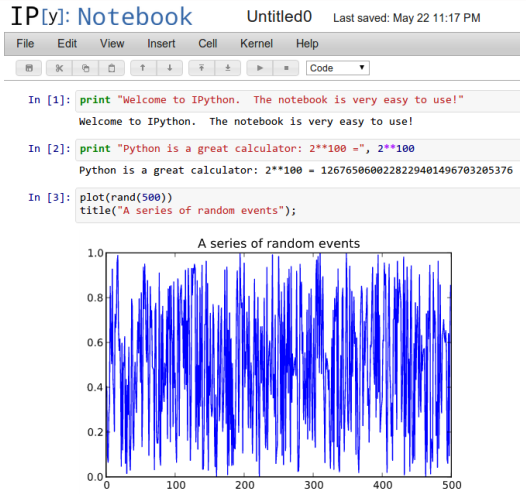
Trends

- More complex hardware
  - Multicores, GPUs, Cloud, NUMA*, PoP+SoC**, …
- More complex questions
  - “Last month sales” → “Next month sales”
- More complex apps
  - Distributed, Service-oriented, Rack-aware, ...
- More data analysts
  - Easy-of-use, Interactivity, Collaboration, ..
- More data
  - Volume, File Formats, ...

** Non-uniform memory architectures*

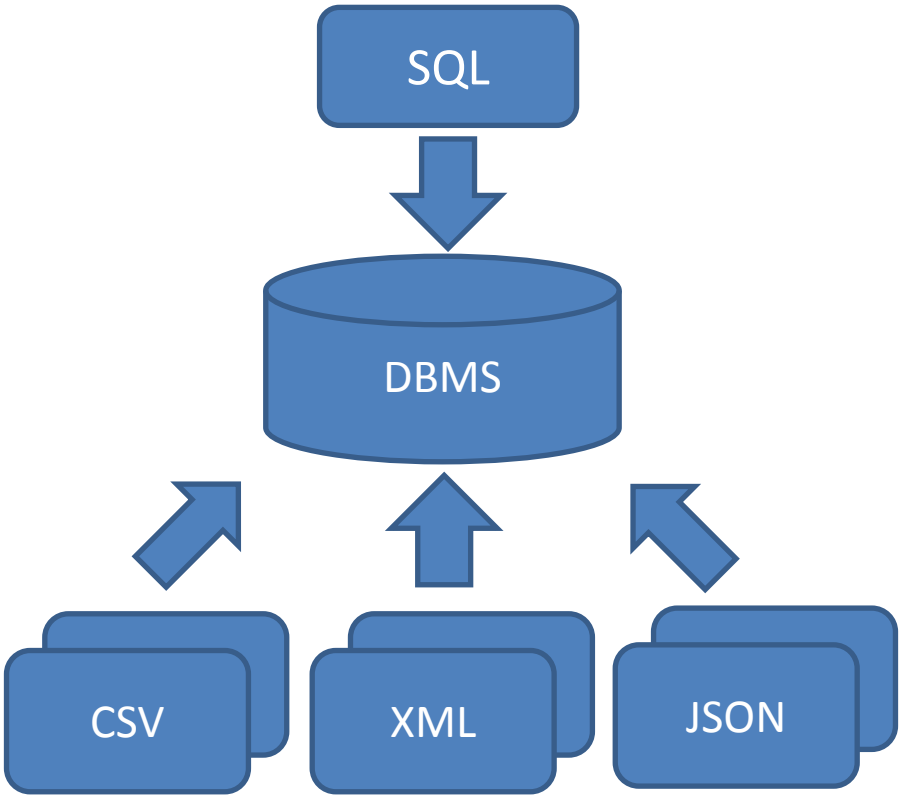
*** Package on Package, System on a Chip*





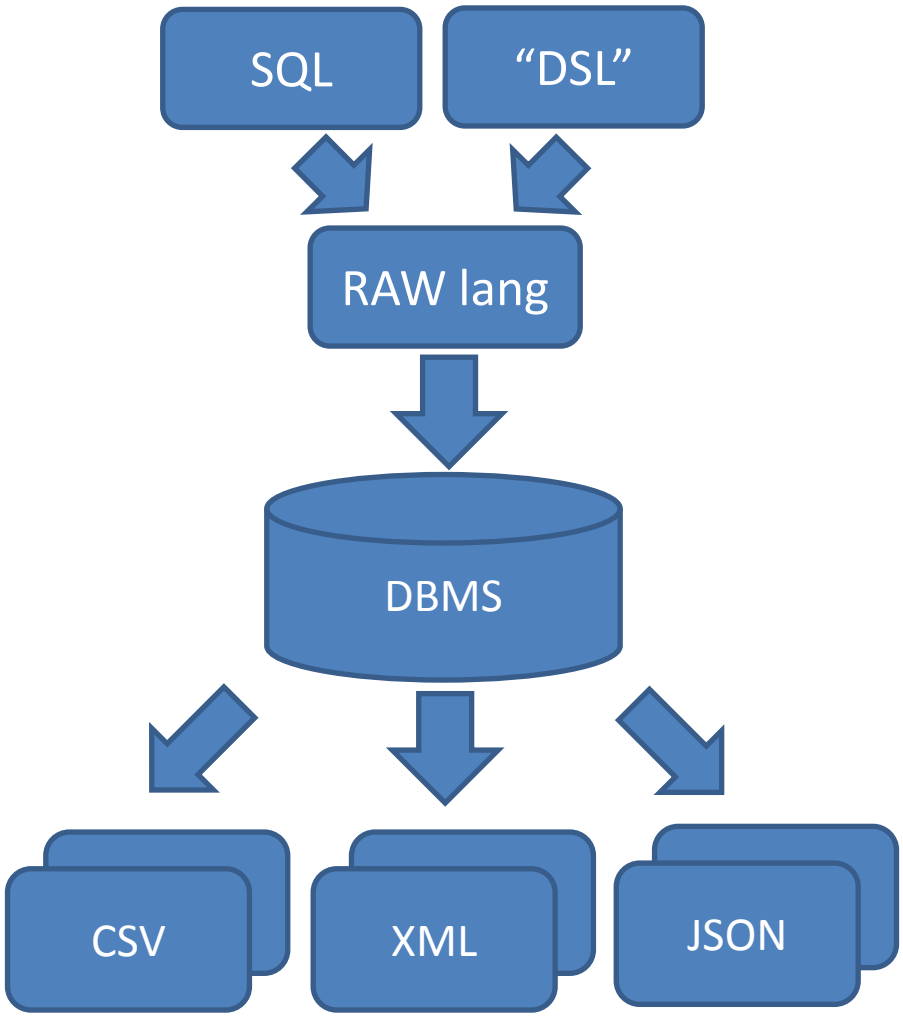
- No data loading
  - No “physical” data copy: support existing file formats
- No database tuning
  - Instead, self-tuned based on actual usage patterns
- Not restricted to tables
  - Add support for trees, vectors, matrices, …
- Not just SQL
  - Instead, enable domain-specific languages

Traditional Database



Data adapts to the query engine

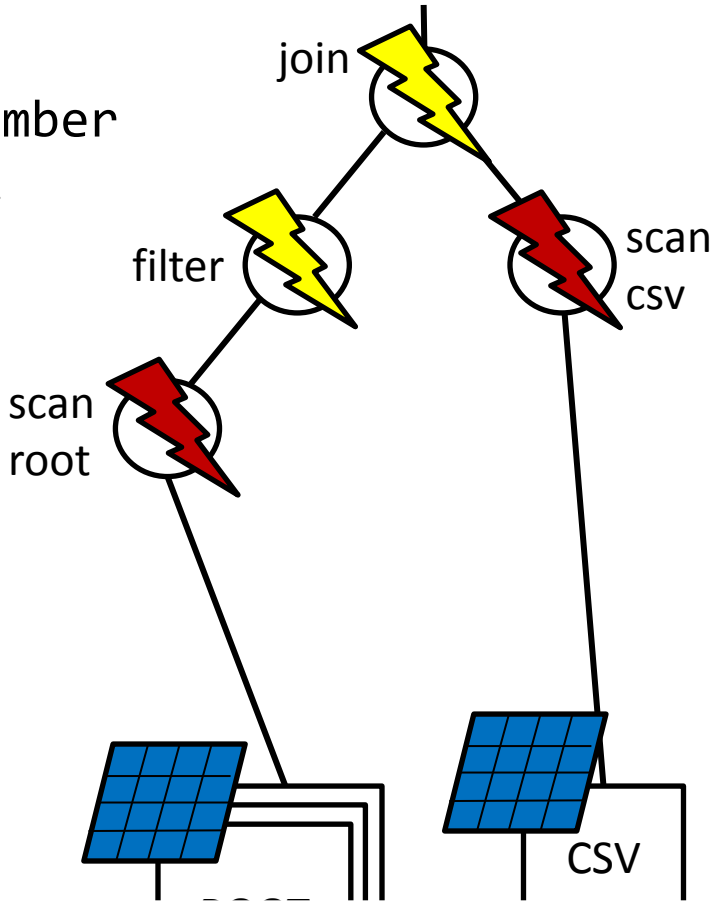
RAW



Query engine adapts to the data

How RAW adapts to data

```
SELECT event.jet…
FROM csv, root
WHERE csv.RunNumber = root.RunNumber
AND root. EF_2mu13 == TRUE AND …
```



Code Generate the Access Paths

Code Generate the Query

Build Position and Data Caches

Adapt to format, file instance and query

just-in-time

Adapting to schema & query

✓ Remove overhead of generic operators

GENERAL-PURPOSE

[CSV input]

```
∀col:
  if col needed:
     if col isInt
        readInt();
     if col isFloat
        readFloat();
     if ...
  else:
    skipField();
```

JUST-IN-TIME

```
readInt();
readInt();
skipField();
readFloat();
skipRestLine();
```

Adapting to format

- Unroll Columns

```
∀col:                    readInt();
if col needed:           skipField();
  if col isInt      ⇨    readFloat();
  ...                    skipRest();
```

- Free navigation in files

- fieldLength:10
- tupleLength:100
- Need fields 2 & 5 of 2nd row

⇨

```
moveTo(110);
readInt();
moveTo(140);
readFloat();
```

- Embedded indexes/existing APIs

- Bitmaps, R-Trees etc.
- readNextField() vs. readField(filename,id)

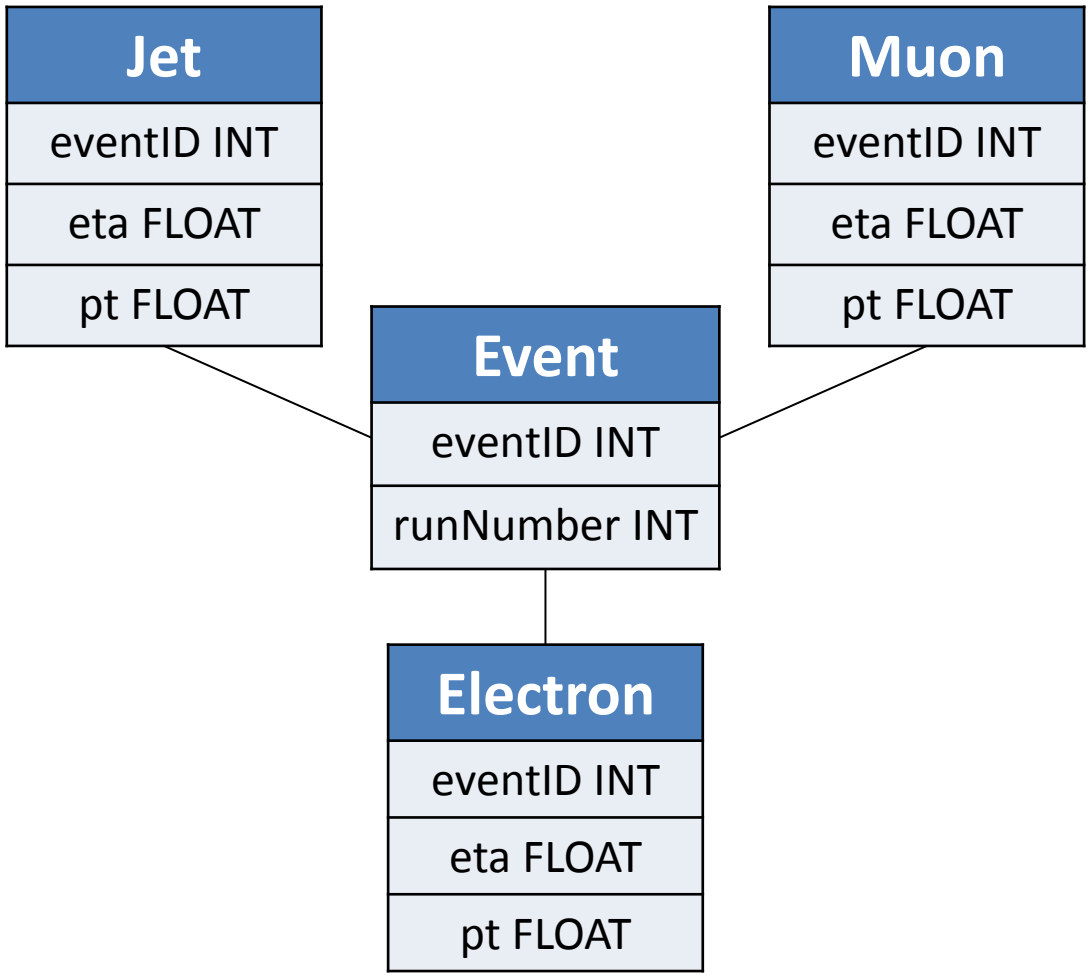
DATA-INTENSIVE APPLICATIONS AND SYSTEMS

HEP analysis: Data

ROOT - C++

```
class Event {
    class Muon {
        float pt, eta;
        …
   }
   class Electron {
        float pt, eta;
        …
   }
   class Jet {
        float pt, eta;
        …
   }
   int runNumber;
   vector<Muon> muons;
   vector<Electron> electrons;
   vector<Jet> jets; }
```

RAW



HEP analysis: Queries

"Identify events of interest → Filter out background events → Plot aggregated results in a histogram"

ROOT - C++

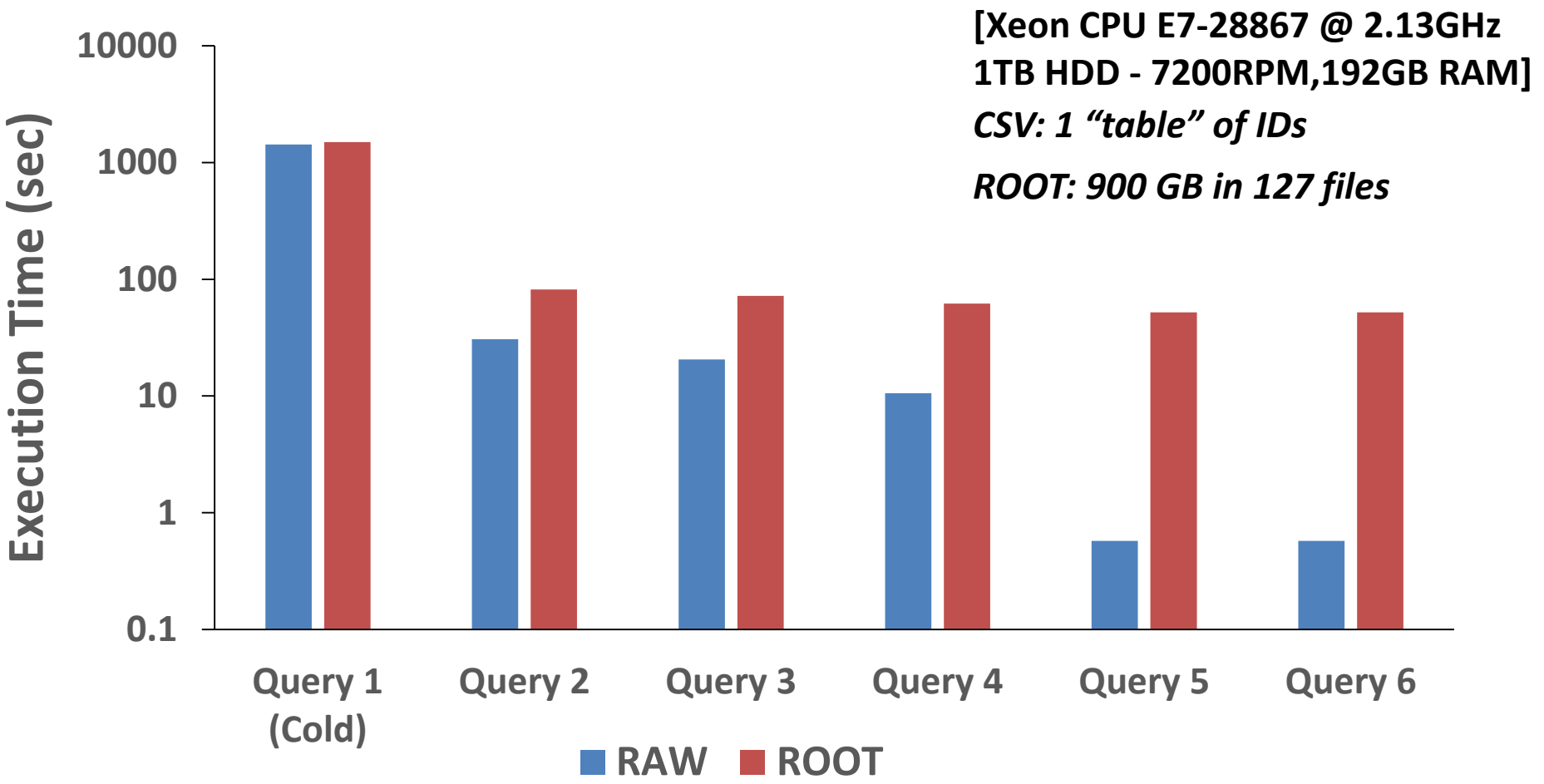
1000+ lines of C++

```
for (unsigned int imuon = 0 ;
     imuon<((*curr_entries)[jentry].mu_pt)->size();
     imuon++) {
      if (((*curr_entries)[jentry].
           mu_ptcone20)->at(imuon) < 0.1 *
           ((*curr_entries)[jentry].
           mu_pt)->at(imuon) &&
          ((*curr_entries)[jentry].
           mu_pt)->at(imuon) > 20000. &&
          fabs(((*curr_entries)[jentry].
           mu_eta)->at(imuon)) < 2.4 &&
          …
}
...
```

RAW

```
SELECT event
 FROM root:/data1/ATLAS/*.root ,
        csv:/data1/ATLAS/events.csv
 WHERE
 (          csv.id = event.id AND
            event.EF_e24vhi_medium1 OR
            event.EF_e60_medium1 OR
            event.EF_2e12Tvh_loose1 OR
            event.EF_mu24i_tight OR
            event.EF_mu36_tight OR
            event.EF_2mu13) AND
            event.muon.mu_ptcone20 < 0.1 *
            event.muon.mu_pt AND
            event.muon.mu_pt > 20000. AND
            ABS(event.muon.mu_eta) < 2.4 AND
.....
```

RAW vs. the ROOT framework



Declarative queries + up to 90x improvement

RAW for High-Energy Physics

- End-users:
  - Performance (JIT, codegen, vectorwise, …)
  - Easy-to-use (declarative) query language
- Infrastructure Providers:
  - Data kept in original location & file format
  - Declarative query language → More optimization opportunities
    - "Event" caches

http://dias.epfl.ch/RAW

Thank You!