Petaminer: Efficient Navigation to Petascale Data Using Event-Level Metadata

P. Hamill
Tech-X Corporation
5621 Arapahoe Ave, Suite A, Boulder, CO 80303, USA
E-mail: `paulh@txcorp.com`

J. Cranshaw, D. Malon and A. Vaniachine[1]
Argonne National Laboratory
9700 S. Cass Ave, Argonne, IL, 60439, USA
E-mail: `cranshaw@anl.gov, malon@anl.gov, vaniachine@anl.gov`

MySQL is a standards-compliant, high-performance, relational database with a pluggable storage engine architecture that enables third parties to create low-level interfaces to data stored in different formats, permitting more optimal operation in various use scenarios. The Petaminer is a joint project with a private company whose purpose is to exploit a feature of MySQL to provide a direct "MySQL interface" to ROOT files. We report on the development of a custom MySQL storage engine to directly access the LHC experiment's event TAG metadata stored in ROOT files. ROOT is a structured, extensible analysis framework and data architecture for storage and retrieval of arbitrarily scaled, distributed data. ROOT supports high-rate streaming of data as well as storage distributed across multiple file systems and domains. It was designed to contain physics event data among other uses, and so has numerous features that make it ideal for this purpose, including hierarchical, column-oriented data structures, dynamic schema evolution, and support for binary float numbers. The ability to access column-oriented event data stored in distributed ROOT files using standard MySQL tools offers the potential to improve the efficiency and accessibility of metadata-keyed queries for LHC data analysis. The Petaminer project demonstrated feasibility of using ROOT TTree files as a pluggable storage engine for MySQL. We present first results of our feasibility studies of creating a column-oriented MySQL storage engine that uses the ROOT API to access TAG metadata directly from ROOT files, and uses ROOT-FastBit to map MySQL indexing operations to FastBit indices. The prototype storage engine demonstrates both query functionality and improved performance. As a result, Petaminer enhances MySQL with column-oriented storage capable to support binary float numbers with bitmap indexing. Work is in progress to extend the functionalities of the Petaminer software.

XII Advanced Computing and Analysis Techniques in Physics Research
Erice, Italy
3-7 November, 2008

[1] Speaker

1. Introduction

HEP experiments at the LHC store Petabytes of data in ROOT files described with TAG metadata. The LHC experiments have challenging goals for efficient access to this data. Physicists need to be able to compose a metadata query and rapidly retrieve the set of matching events. Such skimming operations will be the first step in the analysis of LHC data, and improved efficiency will facilitate the discovery process by permitting rapid iterations of data evaluation and retrieval.

To address this problem, the Petaminer project [1] have prototyped a custom MySQL storage engine to enable the MySQL query processor to directly access TAG data stored in ROOT TTrees [2]. In addition to providing an efficient SQL query interface to the data stored in ROOT TTrees, the Petaminer engine links MySQL index-building capabilities to FastBit [3] bitmap indexing of the data in ROOT TTrees, further optimizing to TAG query performance.

Column-oriented databases are an emerging technique for achieving higher performance than traditional row-oriented databases, especially in large scale data-mining scenarios. As ROOT TTrees are inherently column-oriented, reading them directly will provide improved performance over traditional row-oriented TAG databases.

1.1. TAG databases

TAG databases are a strategy enabling raw event data to be tagged with descriptive metadata. Although capitalized by convention, TAG is not an acronym; in the context of this work, TAGs are defined as name-value pairs representing mutable event metadata such as geometry, energy, source, quality, or arbitrary descriptive designations. Such in-situ metadata allows massive amounts of data to be flexibly categorized and processed. A particularly important goal of TAG is enabling the storage of massive amounts of raw data in central locations that have sufficient storage, processing, and network infrastructure to handle it, while also permitting remote scientists to work with the data by using TAG metadata to select smaller-scale, higher-quality data that can be downloaded and processed at locations with more modest resources, a process known as event skimming. The TAG schema will evolve over time as additional metadata is identified. At 1 kB per event, ATLAS TAGS data are expected to grow by approximately 10 TB per year [4]. Other HEP experiments are expected to use metadata tagging schemes similar to ATLAS TAGs to support event skimming.

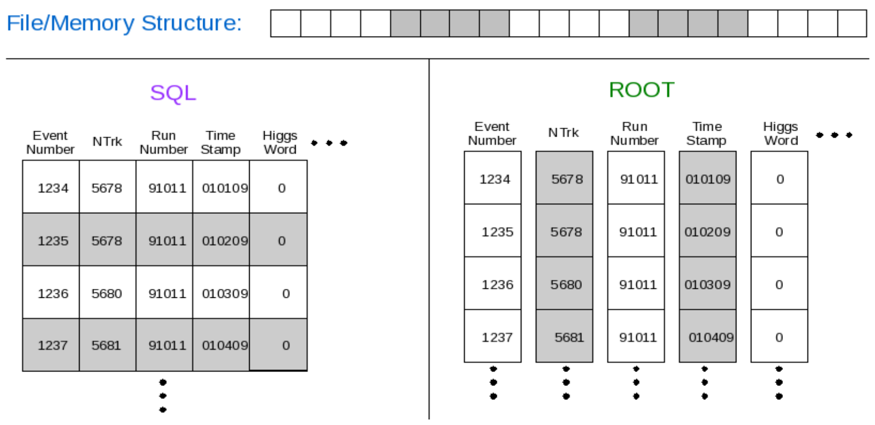
When a reconstructed event is written to the ATLAS Event Store comprised of POOL/ROOT files, the metadata about the event (a “tag”) is exported along with references to the event data to a ROOT TTree file. In an Oracle database it can be captured as tables, e.g.:

- Five data tables with different physics streams: Muon, Egamma, Minbias, Jet and Bphys
- Six runs/partitions per table: 52280, 52283, 52290, 52293, 52301, and 52304

A row's columns consist of references to AOD and ESD files, physics quantities, and trigger bitmasks for Event Filter, L1 and L2 triggers. Indexing makes selection of rare events highly efficient, even at Terabyte scales. However, the need to copy data from ROOT into database tables hampers scalability of operations, due to the need to periodically reload very large tables, TAG metadata schema evolution requiring corresponding changes to table schemas, and dealing with database tables' physical size limitations. In contrast, Petaminer avoids such problems by reading data directly from the source ROOT files.

1.2. Row versus column-oriented data formats

Row-oriented data storage is optimized for fast writes and permits row-level locking for updates. In contrast, a column-oriented format is optimal for fast reads and data mining. Column-oriented databases are an emerging technique for achieving higher performance than traditional row-oriented databases, especially in large scale data-mining scenarios.



Event Number	NTrk	Run Number	Time Stamp	Higgs Word
1234	5678	91011	010109	0
1235	5678	91011	010209	0
1236	5680	91011	010309	0
1237	5681	91011	010409	0

Event Number	NTrk	Run Number	Time Stamp	Higgs Word
1234	5678	91011	010109	0
1235	5678	91011	010209	0
1236	5680	91011	010309	0
1237	5681	91011	010409	0

Figure 1. Row vs. column data layout.

Figure 1 compares row-oriented data layout, conventionally used by SQL databases, to column-oriented layout, used by ROOT. In a column-oriented layout, all values of each attribute are stored contiguously, so that during SELECT operations, attributes matching the WHERE clause parameters can rapidly be scanned without having to traverse the entire dataset. Also, column-orientation permits compression of columns via lossless run-length encoding, so that large attribute value sets need much less space, further reducing query times.

2. Approach

To address these problems of efficient data access and data migration, we are developing a custom MySQL storage engine permitting the MySQL query processor to directly access TAG data stored in ROOT TTrees without intermediate duplication in conventional database tables. This approach permits using FastBit indexing to further tune and improve performance.

FastBit is a LBNL-developed generic bitmap indexing tool that permits optimal searches through large datasets containing heterogeneous data types. ROOT-FastBit [5] is middleware that enables FastBit to index and query ROOT file contents. The Petaminer project has updated ROOT-FastBit v0.8 to interoperate with the current release of FastBit (v1.0.2) for purposes of integrating ROOT-FastBit into the storage engine, so that it can directly create and use FastBit indices of ROOT TAG metadata.

The prototype architecture forwards MySQL database operations to the Petaminer storage engine, which makes ROOT API calls to access data directly from ROOT files, without intermediate storage of data in MySQL tables. Figure 2 shows the system architecture.

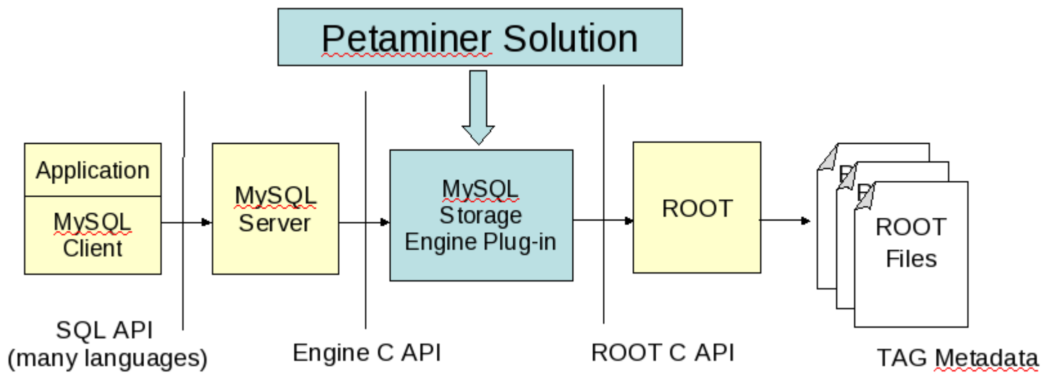


Figure 2. High level software architecture of Petaminer.

The custom storage engine is implemented as a *handlerton* which maps API operations to custom storage engine methods. Development of the Petaminer engine initially focused on functionality to open and read ROOT file contents, specifically targeted at the n-tuples containing TAG metadata. The Petaminer engine internally maps ROOT data structures to analogous database constructs such as tables and columns (Table 1). Using this logical mapping, SQL queries handled by the storage engine are translated into ROOT file access operations.

Table 1. An example of logical data mapping between ROOT and MySQL

ROOT Data	MySQL Data

TFile **MyData.root** {
TTree tree {
TBranch<Int_t> **a1**
TBranch<Float_t> **a2**
}
}
	TABLE **MyData** {
	a1 INT,
	a2 DOUBLE
	}

In addition to processing data reads, the Petaminer software maps standard MySQL database indexes to FastBit bitmap indexing. This is done by making the Petaminer storage engine capture index parameters, and using ROOT-FastBit to create a FastBit index on the same ROOT attributes. Consider this SQL command to create a simple two-column table, each of which has an index:

```
CREATE TABLE  MyData (a1 INT,  a2 DOUBLE, INDEX idx1(a1), INDEX idx2(a2));
```

In this case, Petaminer issues these ROOT-Fastbit calls to create the bitmap indices:

```
TFile* file = new TFile("MyData.root");
TTree *tree = (TTree*)file->FindObject("tree");
TBitmapIndex index;
index.Init();
char indexLocation[16] = "data/test";
index.ReadRootWriteIndexFile(tree, indexLocation);
index.BuildIndex(tree, "a1", indexLocation);
index.BuildIndex(tree, "a2", indexLocation);
```

3. Status and Plans

Our main goal in the first phase of the Petaminer project has been achieved: queries submitted to MySQL via a standard user interface are passed to the Petaminer storage engine and retrieve TAG metadata from a ROOT file. This basic functionality permits any MySQL-compatible client to issue standard SQL queries to a MySQL server with the Petaminer storage engine and read results directly from ROOT files. The regular MySQL query processor is used, so many standard SQL selection, ordering, and processing functions are available, although all operations are read-only. The integration of ROOT-FastBit has demonstrated the feasibility of making MySQL index operations map to FastBit indexing. In preliminary tests, FastBit indexed queries across randomly distributed numeric ROOT attributes have demonstrated an order of magnitude improvement over unindexed queries, on average, a result consistent with [5].

The Petaminer work completed thus far has identified numerous areas for future work on improving the prototyped functionality and adding new features. Additional work on index optimization is expected to produce greater query speed improvements. The project also has demonstrated the feasibility of making ROOT TFormulas map to MySQL User-Defined Functions (UDFs.) This functionality would permit TFormulas to be included in user's SQL statements to perform advanced ROOT operations on query results. In addition, a lightweight Web-based interface will be created for querying TAG databases, in order to provide improved usability and global accessibility. To assure security we will use the grid-enabled MySQL software developed by the SHIELDS project [6].

4. Acknowledgements

We gratefully acknowledge the invaluable advice and participation of ROOT team members Rene Brun and Philippe Canal. We also thank Jérôme Lauret and Valery Fine for their interest and discussions of software applications in Nuclear Physics. This work is supported in part by the U.S. Department of Energy, Division of High Energy Physics, under Contract DE-AC02-06CH11357 and Grant No. DE-FG 02 08ER85058.

References

[1] https://ice.txcorp.com/trac/petaminer

[2] R. Brun, F. Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch.

[3] Kesheng Wu. *FastBit: an efficient indexing technology for accelerating data-intensive science*, J. Phys.: Conf. Ser. 16, pp 556-560.

[4] J. Cranshaw, A.T. Doyle, M. J. Kenyon, D. Malon, H. McGlone, C. Nicholson, *Integration of the ATLAS tag database with data management and analysis components*, 2008 J. Phys.: Conf. Ser. 119 042008.

[5] K. Stockinger, K. Wu, R. Brun, P. Canal, *Bitmap Indices for Fast End-User Physics Analysis in ROOT*, Lawrence Berkeley National Laboratory Paper LBNL-58426, July 26 2005.

[6] A. Vaniachine, S. Eckmann and W. Deng, *Securing distributed data management with SHIELDS,* in proceedings of *XXI International Symposium on Nuclear Electronics & Computing (NEC'2007)*, JINR Publishing, Dubna, 2008, pp 446-449.