Event Collections as AIDA ITuples

SQLTuple

- Requirements on Event Collections API
- Architecture:
  - Technology Choices
  - SQLTuple Package Decomposition
  - FreeHEP AIDA Storage
  - SQLTuple Extensions:
    - ITuple Extensions
    - Collection Management
- Interoperability:
  - Web Service
  - Metadata Management
  - Pool Compatibility
  - C++ Interface
- Performance
- Summary

SQL storage
for FreeHEP implementation
of ITuple AIDA ITuple interface
compatible with Pool Event Collections Metadata
(the rest is pure AIDA)

http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/SQLTuple
http://java.freehep.org
http://aida.freehep.org
http://lcgapp.cern.ch/project/persist/metadata

Requirements on Event Collections API (Event MetaData)

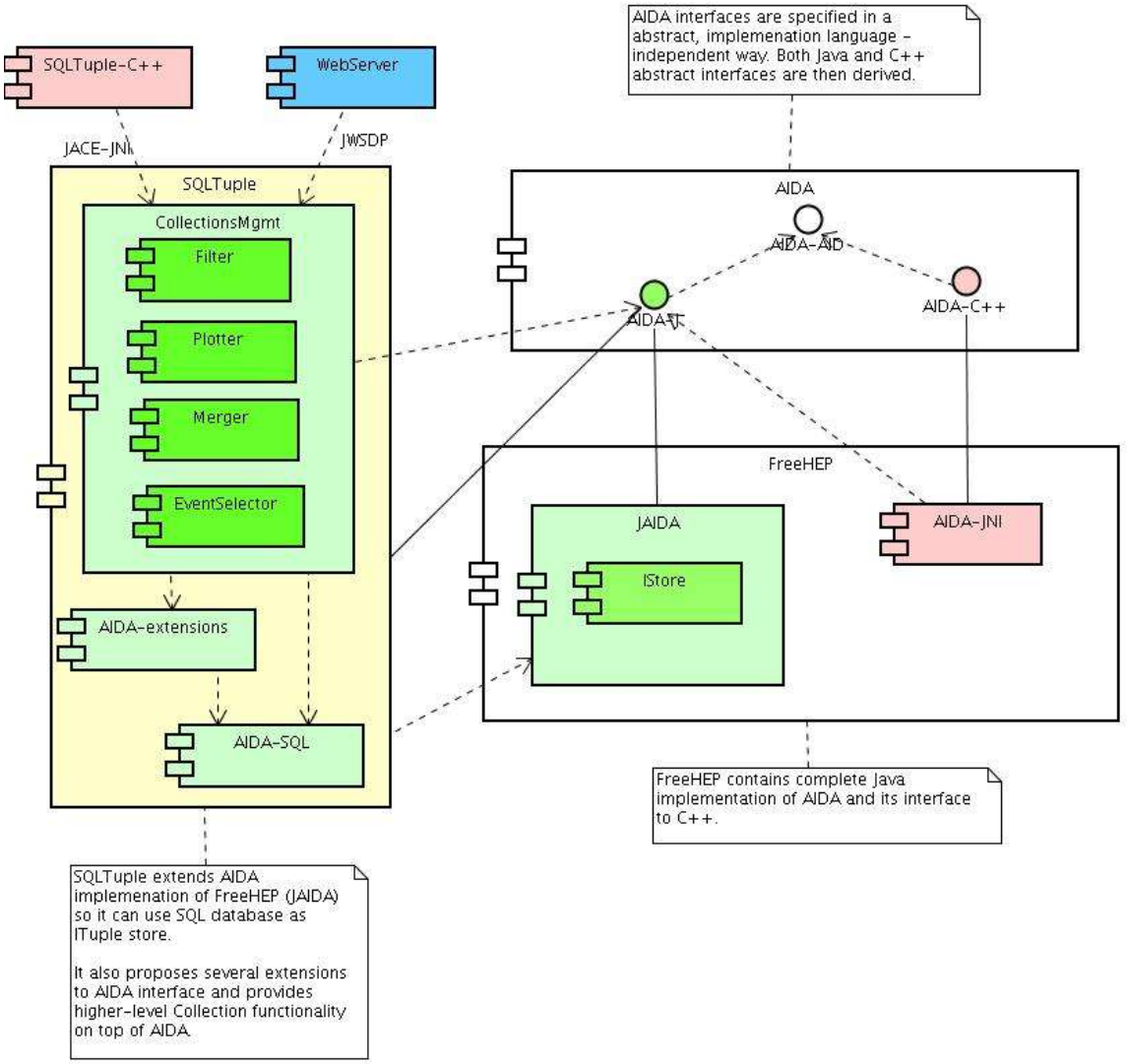
- **Collections Management** functionality should be provided (replications, filtering, merging, spliting,...).
- **Collections Navigation** functionality should be provided (searching, looping,...).
- **Analysis** functionality should be available (histogramming, combining, cutting,...).
- All functionality should be available in **multi-language** environment (Java, C/C++, Python,...).
- All functionality should be available in a **platform-independent** environment.
- API should be reusable with **other storage technologies** (SQL databases, XML files, Root files,...).
- **SQL syntax** should be **hidden**, user should use her native environment (Java, C/C++, Python,...).
- **SQL functionality** should be **used** (performance, advanced functions when available,...).
- **Any** SQL **database** should be supported (MySQL, PostgreSQL, Oracle, embedded DBs,...).
- **Distributed** (Grid) environment should be possible (WebService, OGSA,...).
- Compatibility with **Pool** Event Metadata should be possible.
- **Performance** overhead over native SQL should be negligible.

Technology Choices

- Following technologies have been chosen to satisfy all requirements:
  - **AIDA** for API (Event MetaData are NTuples)
  - **FreeHEP** implementation of AIDA (the most complete; most storage technologies supported; Java, C++, Python and PNuts API)
  - **JDBC** for access to SQL databases (10^3x more users than OTL)
  - **WebService** for distributed and multilanguage interface (real standard)
  - **Java** for native implementation (Java has all needed properties and functionality; it is easy to export Java to other languages)
  - **JACE** for interface to C++ (from C++, it is usually easier to call Java than another C++)

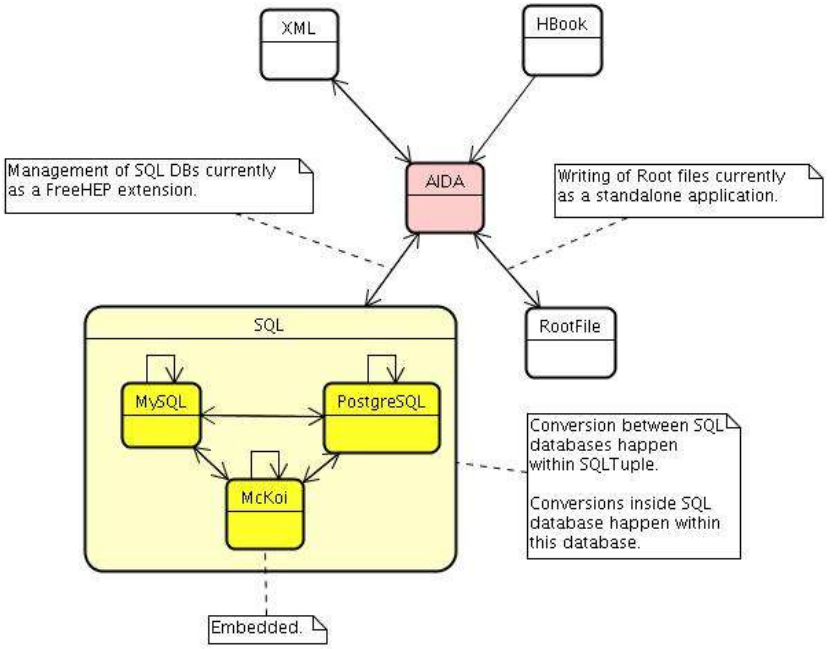
SQLTuple Package Decomposition

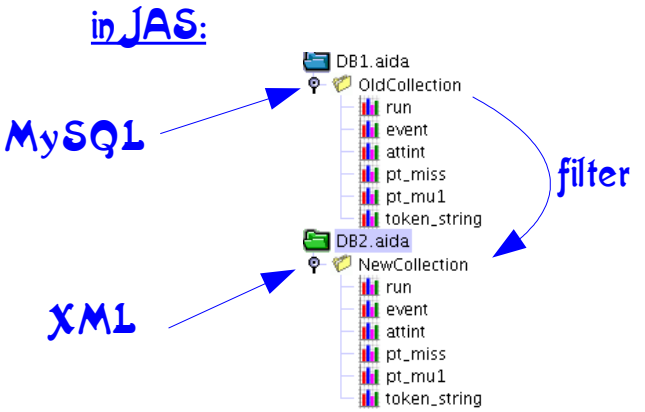
- SQLTuple extends existing FreeHEP implementation of AIDA by implementing SQL Storage.
- It uses the same API as other AIDA FreeHEP storage technologies, like XML or Root files.
- SQLTuple provides several extensions to existing AIDA inerfaces:
  - Extensions to ITuple interface (proposed to be include in AIDA standard).
  - Collections Management utilities build on top of AIDA:
    - Global operations as Filter, Plotter and Merger,
    - Navigation operations as EventSelector.
  - C++ proxies for Collection Management (can be extended to other components).
  - Web Service of Collection Management (can be extended to other components).



FreeHEP AIDA Storage

- AIDA ITuples can be currently stored using many technologies using FreeHEP AIDA implementation:
  - AIDA (compressed) XML files,
  - Root files (reading integrated, writing currently via standalone application),
  - HBook files (only reading),
  - SQL databases (MySQL, PostgreSQL and McKoi directly supported; Cloudscape and Hypersonic tested).
  - ...
- All AIDA standard operations are supported.
- ITuples (=Collections) in ITree can be mounted, linked, copied, moved,... as in the (distributed) filesystem. Then, they can be manipulated from the code, GUI or command line.
- All database operations profit from the native technology (e.g. copying of SQL ITuples is performed within SQL database if possible).





SQLTuple Extensions to ITuple

- Persistency (proposed as AIDA standard):
  - AIDA storage API (IStore) is not yet standardised.
  - Existing IStore implementations support well file-based storage technologies. They have to be updated to accommodate real databases.
  - The only agreed standard for AIDA persistency is currently AIDA XML format. Other possibilities are not yet agreed as a common format.
- ITuple API (proposed as AIDA standard):
  - Richer access methods.
  - Mapping between ITuple rows and Objects (like Pool AttributeSet).
- ITuple management Utilities (layer on top of AIDA).

Collection Management

- **Filter** creates new (sub)Collection/(sub)Replica from existing one applying a filter to attributes.
- **Merger** physically merges two Collections into new one.
- **Plotter** plots selected attributes from Collection.
- ...
- **EventSelector** returns a set of OIDs of Events from Pool Collection using a filter on attributes:

```
EventCollection collection = new EventCollection(url, filter_string, user, passwd);
EventIterator iterator = collection.iterator();

String oid;
while (iterator.next()) {
  oid = iterator.oid();
  ...
  }

collection.close();
```

- Just several examples, others can be easily added.
- Works accross all supported storage technologies.
- Java, C++ and WebService interfaces exist.

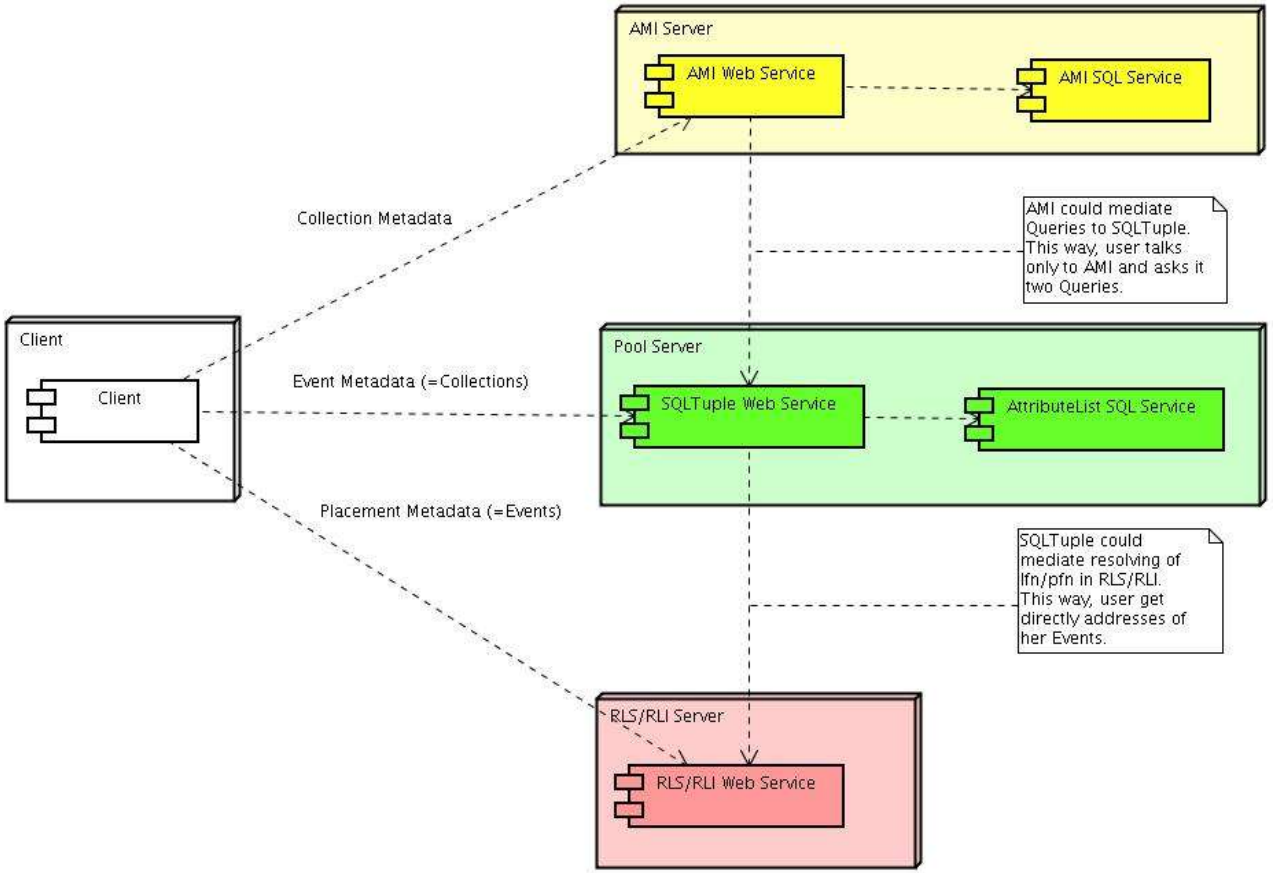
Web Service

- Collection Management utilities are exported using JWSDP WebService server.
- Other AIDA functionality can be easily added.
- Clients can be created from WSDL WebService descriptor in almost any language (even in C++).
- SQLTuple WebService can collaborate with other WebServices (like AMI), forming **Distributed Heterogenous Collections Database**.
- SQLTuple WebService can be reimplemented as an OGSA WebService.

```
/** Web Client to SQLTuple EventSelector WebService.
  * All other code is created automaticaly from WSDL. */

// Get remote EventSelector
EventSelectorWS selector = new EventSelector_Impl().getEventSelectorWSPort();
selector.setProperty(ENDPOINT_ADDRESS_PROPERTY, "http://WebServer.there.net:8080/SQLTuple/EventSelector");

// Select set of OIDs
Object[] oids = selector.select("jdbc:mysql://SQLServer.here.net/Tuples/TestCollection",
                                "pt < 6",
                                "user",
                                "passwd");

// Loop over OIDs
for (Object o : oids) {
  ...
  }
```

Metadata Management

- All Metadata can be accesed in a distributed, language-neutral way using WebServices.
- One WebService could mediate access to other WebServices so user would talk only to one interface. So, for example, user would supply two query strings to AMI and AMI would forward the second one to SQLTuple.
- Local replicas of selected (sub) Collections can be created (using embedded technologies) to enable disconnected work.
- WebServices should be placed close to data (idealy on the same node).



Pool Compatibility

exists

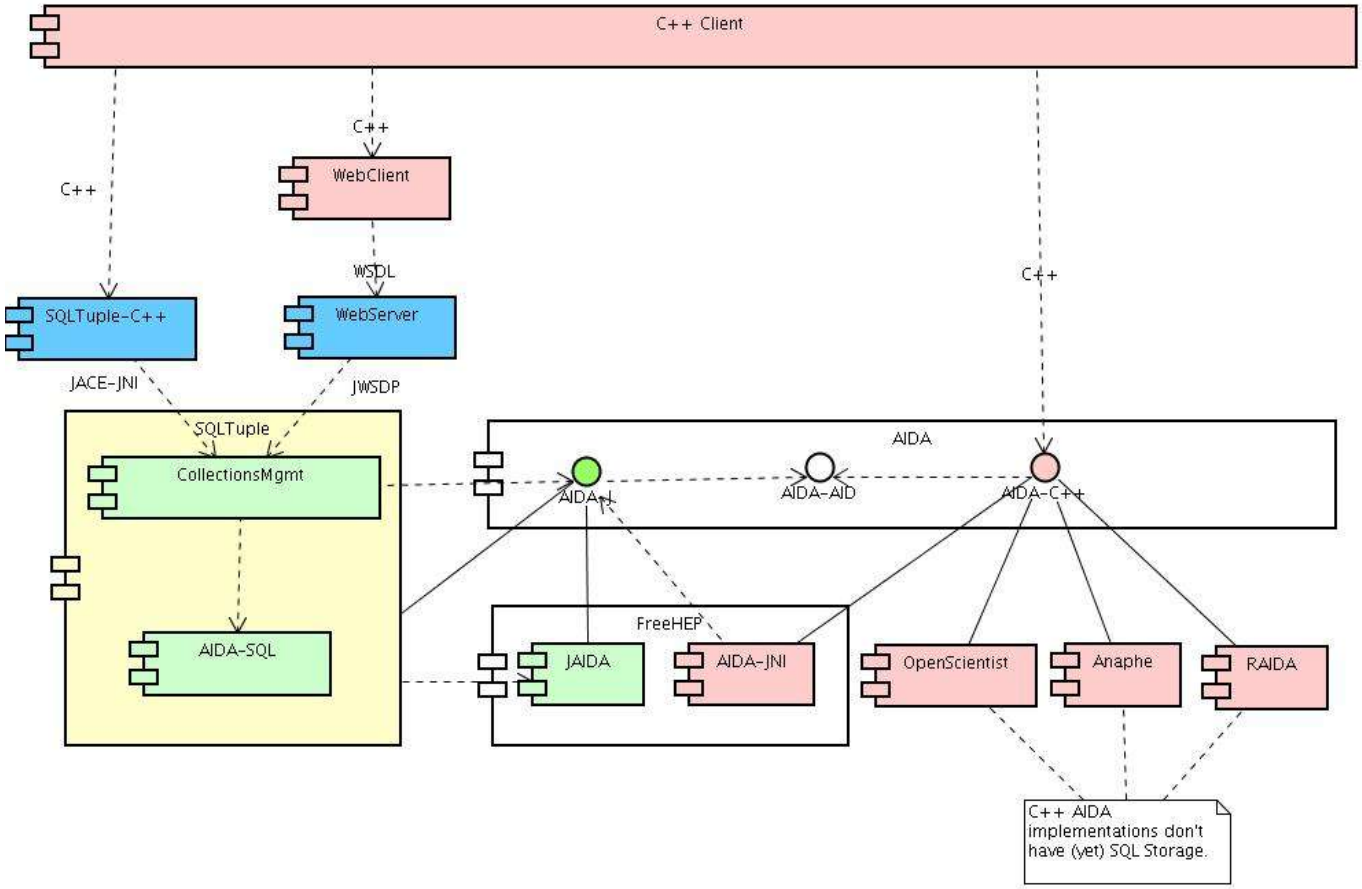
- The default SQLTuple behavior is to map SQL table directly to AIDA ITuple.
- All SQL commands are defined in a properties file *StmtSrc.properties*, user can provide customized file which allows different mapping of SQL to ITuple (i.e. different Schema).
- Pool AttributeSets are stored in two tables: attributes themselves in one, OIDs in another (LinkTable):
  - Customized *StmtSrc.properties* file is used to access Pool Collections.
  - Pool-aware Collection Management classes are used to access Pool-specific Collections (PoolFilter, PoolMerger, EventSelector).

to be implemented

- Pool SQL tables are not rich enough **to provide portability across applications and databases**:
  - Mapping between SQL and native types is not unified and can't be deduced from the database content. Conservating reading is not possible without additional information.
  - Schema are not available without reading actual database.
  - Different databases don't use the same SQL dialect.
  - OIDs (Tokens) can't be interpreted outside Pool.
- Pool Collections should put Schema definition next to data (into SQL database), it should contain two tables:
  - SQL commands (to be) used to access data (i.e. content of *StmtSrc.properties*).
  - AttributeSet Schema (i.e. mapping to SQL types, default values, comments,...).
- Standalone Service for creation/interpretation of OIDs (Tokens) should be available.

C++ Interface

- SQLTuple is written in Java (and SQL) to profit from mature infrastructure, largely non-exiting in other languages (JDBC, FreeHEP,...).
- Many interfaces to C++ are available:
  - FreeHEP AIDA implementation itself implements C++ AIDA interface via AIDA-JNI package (used, e.g., by Geant4).
  - Direct C++ proxies to Collection Management utilities are created using JACE package.
  - JWSDP WebService can be transparently used by any WSDL C++ Web Client.

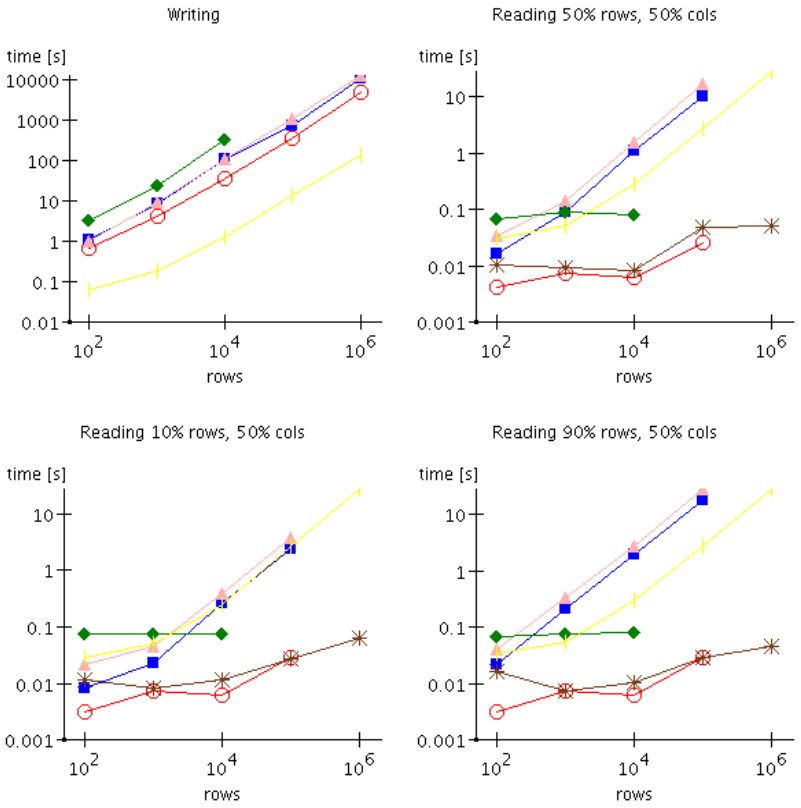


Interfaces to other languages (Python, PNutes, ...) are provided too.

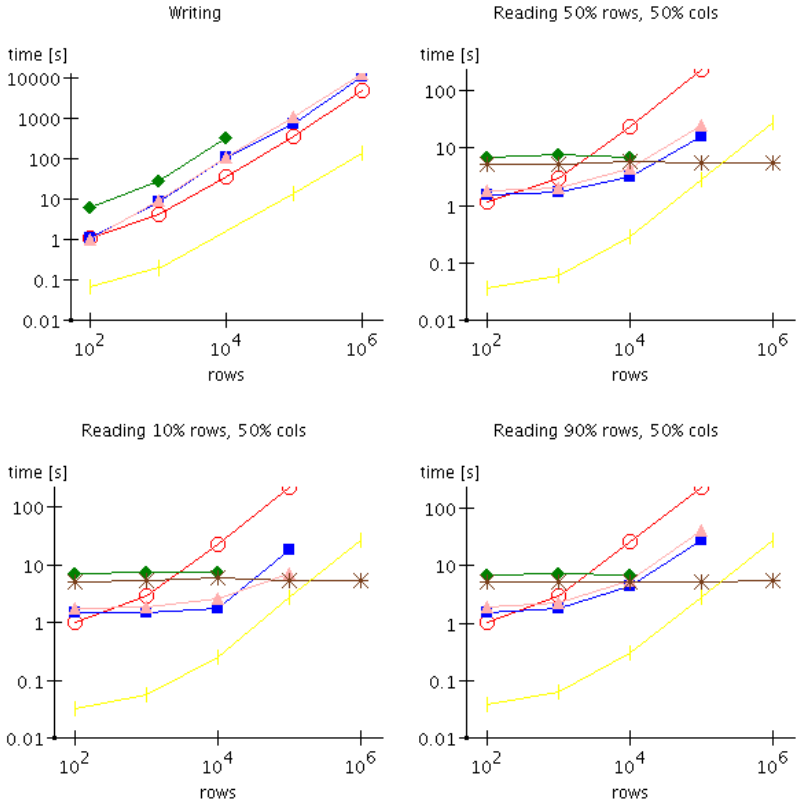
Performance

- Detailed CPU benchmarking suite is provided to evaluate all supported storage formats with different access patterns (reading subset of columns, selecting subset of rows by filter), results are available on the Web.
- However, comparison of different technologies can only be done on real applications in a real use because simplified benchmarks are always trapped by differences in optimization strategies:
  - Client-side and Server-side caches
  - Indexing
  - Hollow variables
  - Lazy loading
  - Dynamic optimization
  - Memory management
- Generally, CPU time needed to read an ITuple can be divided into two parts:
  - Constant overhead spent only once per ITuple, it is used to understand ITuple Schema, perform selections and prepare structures in memory.
  - Access time proportional to the amount of data read in.
- File-based, embedded and simple storage technologies seem to be more performant for flat access (when user reads everything or at least knows in advance what she will need).
- SQLTuple (and JDBC) brings in negligible overhead, most of the time is spend in low level access code.
- Size of stored ITuples depends linearly on number or entries and is (for 100000 rows * (50 floats + 50 ints)):
  - XML: 54MB
  - MySQL: 44MB
  - PostgreSQL: 21MB
  - McKoi: 223MB
  - Root: 42MB (25MB when compressed)

Performance - sample



Without constant overhead



With constant overhead

Summary

AIDA, extended by SQLTuple, is suitable API for Event Collections.
FreeHEP provides all necessary foundadions.

- **SQLTuple** presents any SQL data as standard AIDA ITuples so it
  - can be transparently reused within any AIDA-compatible application,
  - can transparently reuse any other AIDA-based storage technology.
- **SQLTuple** is platform independent.
- **SQLTuple** supports any relational database.
- **SQLTuple** offers multi-language access (Java natively, C++ via proxies, any language via WebService).
- **SQLTuple** is compatible with Pool Event Metadata.
- **SQLTuple** provides high level Collection Management Utilities.
- **SQLTuple** can be used in a distributed (Grid) environment.
- **SQLTuple** reuses: Java, FreeHEP, AIDA, JDBC, JACE, JWDSP, MySQL, PostgreSQL, McKoi, Log4J, Ant.
- **SQLTuple** is curently available in version beta3.