Argonne NATIONAL LABORATORY

The future of event-level information repositories, indexing, and selection in ATLAS

Elizabeth Gallas (on behalf of Jack Cranshaw)

for the ATLAS Collaboration

U.S. DEPARTMENT OF ENERGY

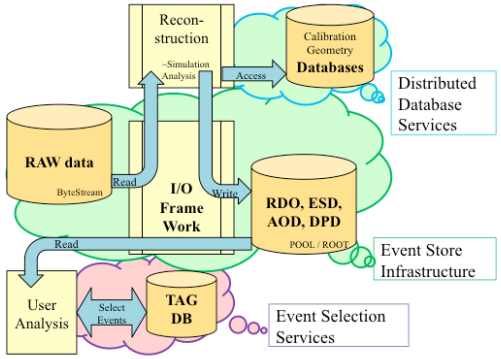
The ATLAS Collaboration

- ATLAS is a general purpose detector, so the content of the data being taken is fully configurable through the trigger system. ATLAS computing uses primarily dedicated computing and storage resources at the detector, at CERN, and at various grid sites distributed around the world. Although the primary computing environment is Linux, the programming environment uses many different languages, although python and C++ are the most common.
- These data are grouped as events which go through multiple stages of processing and multiple format changes.
- *The LHC schedule*
  - **2009-2013 (Run 1)** Data was taken at multiple energies with a data rate of 200-500 Hz during live times. Equivalent amounts of simulation data were also generated.
  - **2013-2015 (LS1)** This is a break in LHC running to bring the machine up to design luminosity. It is also a time when ATLAS is conducting upgrades of both the detector and computing systems.
  - **2015-2018 (Run 2)** The LHC will run at 14 TeV* COM energy and above design luminosities. The plan is that ATLAS will be taking data at 1 kHz, and that the events will be larger and more complicated due to increased pileup.

The ATLAS Computing Model and the TAGS project

- When the ATLAS Computing Model was first written down in the early 2000s, an event level metadata system was included as one of the data products. The metadata would be gathered in the final production step and used by physicists to access analysis object data (AOD) and upstream data.
- Due to perceived bandwidth constraints, it was foreseen that the event data would be relatively static and that jobs would be sent to the data, the MONARC model. Since then the data distribution model has evolved to be more flexible.
- The TAGS project was charged with implementing the system for gathering, querying and retrieving event data. The metadata content was defined by a working group in 2006 to include metadata viewed as useful for event selection.
- The plan was that the TAG data would be uploaded into a central database where it could be used to configure and guide jobs using the navigational information included in its production.

Navigating the event store



- **Navigation:** From the beginning of the ATLAS computing model, the Event Store group has included a navigation capability within ATLAS data products. This was done using tokens which contained information on the object and its position within a file containing event data.
- **Event Pointers:** Tokens were general, but when used to point to an event header, they become event pointers which could be stored outside the files to provide navigation to individual events.
- **Provenance:** In addition, event headers were designed to keep pointers to the event in the previous data product in the processing chain.
- **Abstraction:** With sets of data as large as those at ATLAS, some abstraction layer which allows users to avoid working with individual files or directories was needed. The following methods were used during Run 1.
  - Groups of files with metadata: *Datasets*
  - Groups of events with group metadata: *Event Collections*
  - Groups of events with event metadata: *TAGS*

Reality and the TAGS project

- The ATLAS data handling system underwent intense evolution during the running period from 2009-2013. This was driven both by experience and new technologies/ideas. This affected how the TAG data was used.
  - **Use cases which diminished**
    - *Selection by physicists:* Physicists migrated away from the standard ATLAS data products such as AOD and ESD and the Athena framework. They moved to doing things directly in ROOT. Since the TAG data did not help to select downstream data, the use case for direct use by physicists became weak.
  - **Use cases which appeared**
    - *Monitoring:* The TAG data were one of the few places that certain data quality concerns such as split luminosity blocks, trigger count consistency, etc. could be checked. Also, the physics content was sufficient that the TAG data could be used for fast physics monitoring.
    - *Event picking*: For rare events, as well as for some targeted reprocessings, there were users who just wanted to use the navigational capabilities after doing a selection on some other data product to produce an event list.
    - *Using the Grid as a database:* In the beginning, the TAG files which were generated from the AOD were seen as a temporary data product which could be discarded after upload to the database. As the grid processing model at ATLAS developed, it was discovered that these files could be useful as a grid-resident database of TAG navigational data both by themselves and in tandem with the Oracle database.

TAG data structures

- One could say that all physics analysis at ATLAS depends on event metadata in the sense that any calculated quantities (electron pt's, invariant masses, …) attached to an event are event metadata. Consequently event metadata naturally follow a data model similar to what physicists use.
- For TAG data this meant that the data structures tended to look like ntuples (when stored in ROOT format, they were stored in a TTree) or simple tables. For data which could be compiled by a single process, this simplicity was also appealing. It mapped easily onto data stored in ROOT or in a relational database like Oracle.
- During Run 1, simple data structures of this sort were used, but various issues cropped up over time.
  - The metadata within the table was used in different ways, but it was hard to optimize a single data structure for use cases with different requirements.
  - Metadata was identified as useful which became available at different times, so the single process model proved insufficient.
  - The software stability and user requirements were more volatile than expected.

The Event Index project

- Eventually the opportunity to improve the systems built for event metadata for the TAG project presented by the long shutdown spawned a new project: the *Event Index*. This project has several charges.
  - **Data structure improvements**
    - *Staged data:* Provide a model which accommodates metadata arriving at different times.
    - *Forward references:* Provide a model which allows metadata to reference data products which were made after the metadata creation.
  - **New technology evaluation**
    - New technologies for storing and processing data have become available since the start of the TAGS project. Event Index is evaluating a prototype in Hadoop.
  - **Improved integration with data processing**
    - *High availability:* More timely acquisition and publishing of event metadata.
    - *Transparent use:* Improve integration as input to data processing.
  - For details on this project, I hope some of you visited the poster displayed on Monday.

Data Structure Evolution I: Separate the Navigation from the Metadata

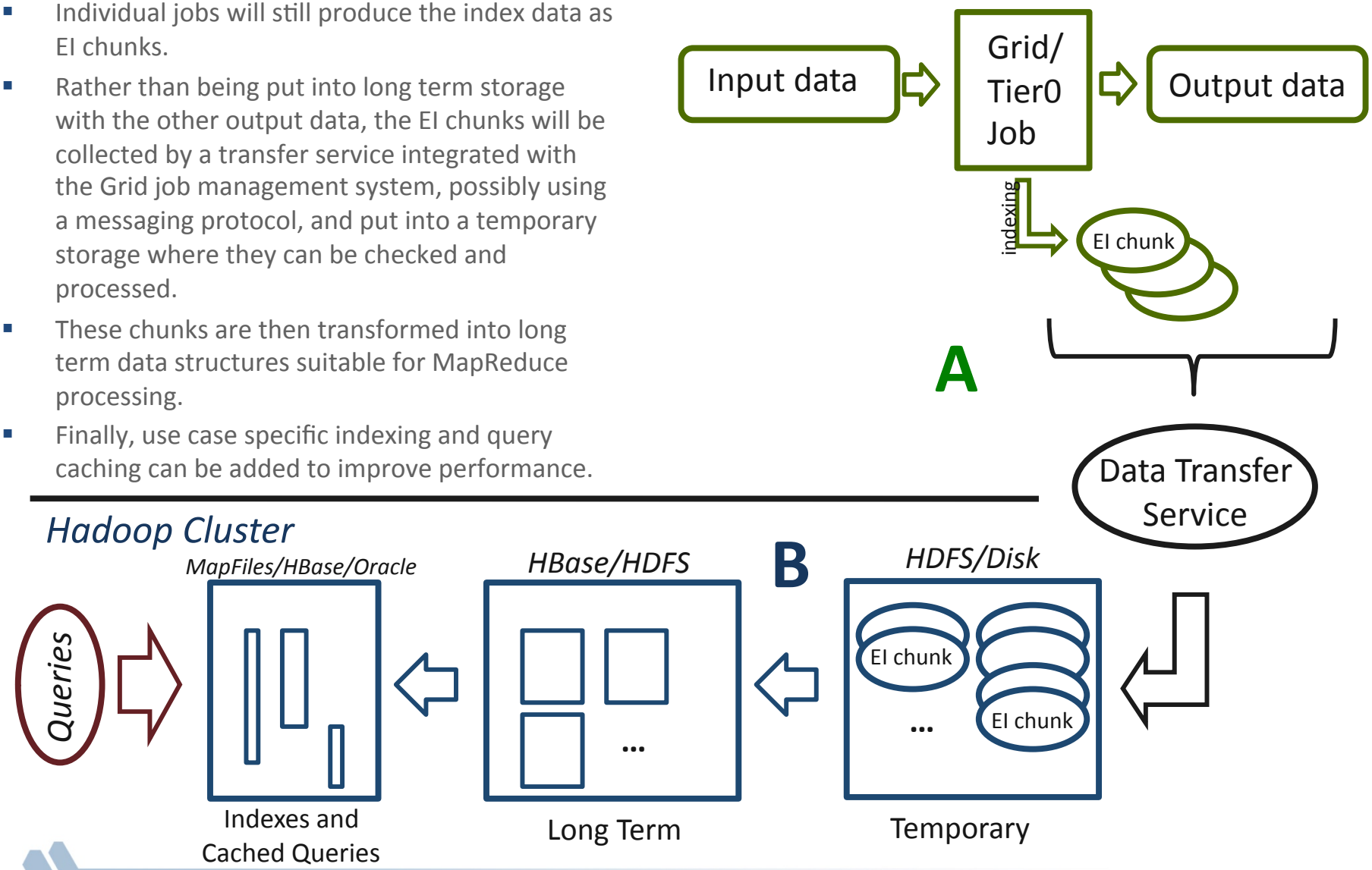
- This is perhaps the simplest and key change that needs to be made. Rather than a single table with (navigation, event id, trigger decisions, physics quantities, …) we vertically partition this into 2 (or more 'tables'). This has several advantages
  - *Fast availability:* The navigational data has no required metadata, so it can be published at the same time that the data it referenced is published.
  - *Forward referencing:* The metadata uses abstract event id's, so it is not limited to the files available when it was produced.
  - *Improved query performance:* The navigational and metadata components can be optimized for different use cases.
- One can take this a step further and different types of metadata as well, for example separating versioned (mutable) and non-versioned (immutable) metadata.
- Finally this improves efficiency as it allows the metadata to be built up as needed rather than defined a priori.

New Database Technologies

- Oracle has been the primary storage used for the TAG database, but new technologies have become available since Oracle was adopted circa 2003. One that we are investigating is Hadoop. It has several capabilities which we think are particularly adapted to use by an event metadata system.
  - Schema flexibility.
  - Excellent scaling with commodity hardware. (Cheaper).
  - A large selection of open source tools with a growing user community.
  - Multiple processing models from pure map-reduce to quasi-relational.
- Note that this reduction in cost is really a transfer of costs
  - Query processing has to be written by us.
    - This can also be viewed as an advantage as we have more control.
  - Optimization of query execution has to be handled by us.
- Initial evaluation is still underway to implement current TAG capabilities in an Event Index in Hadoop.

The general idea

- Individual jobs will still produce the index data as EI chunks.
- Rather than being put into long term storage with the other output data, the EI chunks will be collected by a transfer service integrated with the Grid job management system, possibly using a messaging protocol, and put into a temporary storage where they can be checked and processed.
- These chunks are then transformed into long term data structures suitable for MapReduce processing.
- Finally, use case specific indexing and query caching can be added to improve performance.



Status

- Several areas of development are being pursued in parallel
  - A. Data Collection/Gathering
    - Development has started between the Event Index developers and the Grid job management system (Panda) to enable the gathering of the Event Index chunks. There are a lot of details here such as caching, messaging, partitioning, verbosity, frequency, etc. which are being worked out.
  - B. Storage Technology and Data Structures
    - Using TAG data from 2011 we have created a realistic set of data within HDFS. One of the advantages of Hadoop is that there are a multitude of open source tools available with different capabilities and strengths. We have been focusing primarily on two alternatives.
      - HBase using python and Thrift
      - MapFiles using java
- The planned timetable
  - Jan-Oct 2013: tests of data formats, schemas, performance of upload, search and retrieve data on a reduced dataset (1 TB)
  - Oct-Dec 2013: implementation of the chosen solution on the CERN Hadoop cluster; adaptation or development of external services; upload of all existing data
  - Jan-Jun 2014: commissioning of the new system; performance optimization
  - Jul-Dec 2014: commissioning with new cosmic-ray data

Data Structures Evolution II: New ways of indexing

- In the TAG model, everything is organized by event in a single monolithic, ntuple structure. The metadata seems better organized by metadata values, and it may be amenable to an inverted index structure. Here's an example for the trigger.

Run	Datablock	Trigger	Events
1	1	Trig1	(1,4,5,6,8)
		Trig3	(1,2,3,7,9)
	2	Trig1	(12,17,18)
		Trig4	(10,15)
		Trig5	(11,13,14,17)
		Trig8	(12,16,18)

- The results for a selection of any combination of triggers becomes a group of set operations on sorted sets within lumiblock boundaries. With a system like Hadoop this should be both fast and scalable.

New Opportunities

- If the Event Index works as hoped with the new technology and the new data structures, then the ability to index data both stored in the database and stored on the grid could provide interesting new capabilities.
  - ATLAS is developing a more event-oriented processing model where processes receive events from an event server rather than files. An event index could be quite useful for these sorts of jobs.
  - There are a variety of data preservation activities and publication processes which may benefit from an event index.
    - Sharing and monitoring overlap between data products.
    - Replotting or reanalyzing data from a previous analysis using reprocessed data (new reconstruction, new calibrations, new fits, …)
    - Easily export data for archiving or publication.

Backup Materials

		Oracle	PROOF	SciDB	Hadoop	HBase
Predecessor		RSI	ROOT	XLDB conference series	Google MapReduce	Google BigTable
Created		1982	2003	2009	2005	>2004
Hardware restrictions		High reliability, high availability	commodity hardware			
File System		Oracle RDBMS	POSIX	DMAS	HDFS	
Storage Unit		Partition / Block	File / Basket	Chunk (user defined)	File / Block (64MB)	
Grouping Unit		Table	Dataset	Array	File / Directory	Table
Compression						
User Defined Types		No	Yes			
User Defined Functions		Yes	Yes			
Query Language(s)		SQL • insert • delete • update • select • where • join • order • group	C++ API, Python API	ArrayQL, C++ • subsample • trim • slice • reshape • Sjoin • Cjoin • concatenate • project • filter • aggregate • limited MR	All JVM compatibles • Map reduce	All JVM compatibles • Put • Delete • Scan • Get
Data Model	**Storage Type**	Relational	Column	Multi-dimensional	Wide Column	
	Unit	Tables	Tree	nested arrays	key-value	key-(ragged array)
Features	**Updatable?**	Yes	No	Add a history dimension	No	Versioning
	Constraints?	Yes	No	Yes	Primary key only	Yes
	Indexing?	Yes	Local		Third party	Yes
	Query Optimizer?	Provided	User	For array operations	Third party	
	I/O optimized for?	Random access	Configurable		Sequential access	
	Sparse storage?	No (user designed)	No	Yes	Yes	Yes
	Runtime extend?	No	No		No	Yes
	NULL support?	Native	User defined	Native Extendable	User defined	User defined
	Data Dictionary?	Native	Available	Yes	Third party	Yes
	ChangeLog?	Yes	No	No		Versions
	Provenanc	No	No	Yes*		Versions