



# **Grid-enabled Data Access in the ATLAS Athena Framework**

**D. Malon, E. May, A. Vaniachine (Argonne),  
S. Resconi (Milan),  
J. Shank, S. Youssef (Boston University)**

**CHEP'01  
Beijing**

**3-7 September 2001**

# Outline

- **Background: general physics experiments approach to computing**
- **Athena background**
  - **Event Selection**
  - **Output**
- **Globus replica catalog**
  - **Logical and Physical filenames**
- **Athena and the Globus replica catalog**
  - **Registration...Retrieval.**
- **GDMP**
- **Virtual Data**
- **Future**



## Background

- **Even in grid-oblivious environments, simulation, reconstruction, and analysis programs must register their output so that it can be found by later jobs**
- **Later jobs must, at least implicitly, navigate this “registry” to map a collection name or other input specification into sufficient physical location information to support access to persistent data**
- **Physics experiments have traditionally delivered far richer functionality than this baseline, providing bookkeeping and metadata databases well beyond the capabilities of current standard grid tools**



## **Background II**

- **The underlying abstractions, though--registration of output data, and location and (potential) transfer of input data--can be supported even with today's modest grid capabilities**
- **This work explores approaches to connecting grid implementations of these abstractions to the current ATLAS computing environment**



# Athena

- **Athena is the common control framework in ATLAS for simulation, reconstruction, and analysis**
- **Based upon the LHCb Gaudi framework**
- **Its approach to transient/persistent separation implies, among other things, that Athena users and even most package developers should neither know nor care whether data come from the grid or from local filesystems, nor whether data reside in object or relational databases or ROOT or Zebra files**



## Event selection in Athena

- Athena job configuration includes specification of input events to be processed--set in job options files or scripts
- Simple example for the purposes of this talk:
  - EventSelector.InputCollection = “collection name”;
  - EventSelector.InputDatabase = “database name”;
- (one doesn't really need both--more on this later)
- In ATLAS baseline storage technology (Objectivity/DB), “database” does in fact correspond to a database, but the mechanism applies as well to other supported technologies, in which “database” is replaced with the appropriate unit of physical storage
- More advanced selection criteria (e.g., filter predicates) may be passed as strings to Event Selectors



## Event output in Athena

- Athena jobs often create event collections as output
- Users name these collections in their job options by setting the `OutputCollection` property of the corresponding persistence service
- May also supply placement directives; here, we describe a simplified model in which users name the database into which output should go
- Simple example using Objectivity/DB:  
`NaiveObjyCnvSvc.OutputCollection = "output collection \ name";`  
`NaiveObjyCnvSvc.OutputDatabase = "name of database";`



## Event output in Athena II

- **Production operations are not so simple**
  - single stream's output may go to multiple files
  - file names may be auto-generated to provide consistent access and naming
  - output database sizes (hence, database file boundaries) may be tuned by policy rather than by explicit user choice
  - ...
- **We limit our attention here to the explicit named database case**



## A simple scenario

- **Athena jobs at Site A build collections** of generator (Pythia, ISAJET, ...) events or collections of fast simulation (Atlfast) events, store results in local Objectivity/DB federation
- **Users at Site B require these event collections** as input to their simulations or analyses
- **Site B users configure their jobs exactly as though required input data were local**

### **INTERNALLY:**

- An internal Athena service, the Event Selector, checks the local federation catalog for the requested data
- If database file is not found, grid services are queried to locate the data and trigger its delivery



# Globus replica catalog

- Current (alpha) release is LDAP-based
- Fundamental to the implementation are **“logical collection” objects** and their associated **“location” objects**
- Logical collections have **“logical filenames” as attributes**, as do location objects
- Interpretation is that filename attributes of a collection serve to list the collection’s constituent files; filename attributes of a location serve to list those files in the collection that are available at that location
- Files may be replicated in several locations
- Locations may host only a subset of a collection’s files
- Attributes of locations include access protocol, hostname, port, and (base directory) path



## Logical files and physical filenames

- “Logical file” objects can also be created, but these are not fundamental
  - entirely optional, and serve as loci for attributes like file size and file metadata that might prove useful to higher-level services
- An artifact of this implementation is that *physical filenames are not stored*; rather, it must be possible to generate them from attributes of a location, plus logical filename
- Default strategy? Treat path attribute of location object as base directory, append logical filename to this to get fully-qualified pathname
- Together with location’s protocol/host/port attributes, this suffices to build URL sufficient for data access



## Athena and Globus replica catalog: registration

- When an Athena job creates an event collection in a physical database file, grid-enabled collection registration is straightforward:
  - add filename to the (replica catalog) collection
  - add filename to location object describing Site A
  - (can use OutputDatabase from job options as filename)
- Command-line equivalent of what needs to be done is, approximately,  
globus-replica-catalog ... -collection -add-filenames XXX  
globus-replica-catalog ... -location "Site A" -add-filenames \  
XXX
- (The "... " elides LDAP URL of the collection, and authentication information)



## Athena and Globus replica catalog: retrieval

- When client at Site B specifies this collection as input, the Athena job's Event Selector first tries to find the database locally; on failure, consults Globus replica catalog
  - (can use InputDatabase for this)
- Command line equivalent is, approximately,  
`globus-replica-catalog ... -collection -find-locations XXX`
- This returns list of locations hosting a replica of the requested file; one such location corresponds to Site A
- Physical filename is built from location object attributes and logical filename
- Data transfer (matching the location object's protocol attribute) is initiated
- Local access is then retried (successfully(!?))



## Details, details...

- **After transfer but before local retry, local Objectivity/DB federation must be made aware of the newly imported data**
  - **Objectivity/DB ooattachdb for this purpose**
- **Lack of true logical filename support in current Globus replica catalog is largely what impels us to use InputDatabase name rather than the user's event collection name**
- **Even here some care is needed, since Objectivity by default appends a suffix to a user-specified database name**
  - **Since, in this implementation, building the physical filename is under our control, we can handle this internally**



# Globus replica management

- **Service layer built upon replica catalog services, intended to provide additional functionality and robustness, and to insulate clients from the catalog itself**
- **A simpler scenario, in which data transfer is handled by the replica management layer, is easy to describe, but we have not yet made use of the alpha release of these services for two small reasons:**
  - **only gridFTP is supported--places requirements on database testbed servers**
  - **because replica management constructs physical filenames internally, logical-to-physical filename mapping is even more restrictive than our OutputDatabase/InputDatabase approach**



## **Grid Data Management Pilot (GDMP)**

- **Developed initially for CMS and now a part of both EU DataGrid and Particle Physics Data Grid projects**
- **Different approach to grid-enabled data distribution (comments refer to Version 1.2.2):**
- **Sites run server processes to make database catalogs available to remote sites**
- **Remote sites subscribe to these catalogs**
- **An export catalog of newly-added databases may be built and published to subscribing sites either periodically or by explicit request (gdmp-publish-catalog)**



## **GDMP continued**

- **Remote sites may choose to implement a subscription mechanism, whereby all new databases, or all databases matching a filter criterion, are imported**
- **Alternatively, user jobs at a remote site may inspect the import catalog and choose to import data based upon their own selection filters**
- **A single GDMP request triggers a grid-enabled database file transfer AND handles the ooattachdb step required to notify the local Objectivity/DB federation of the newly-arrived data**



## Athena and GDMP

- If Site A policy is to publish catalogs on a periodic basis, Athena jobs creating output databases need do nothing
- Otherwise, such jobs may internally call `gdmp_publish_catalog`
- For retrieval, just as with the Globus replica catalog, Athena jobs may try first to get the data from the local database catalog
- On failure to find data locally, `gdmp_replicate_file_get` with an explicit `-sourcefile` (or sufficiently clever filter-setting) suffices to trigger both the transfer and the `oattachdb`



## Toward virtual data

- **GriPhyN project (Grid Physics Networks) distinguishes itself from other grid projects by emphasis on virtual data**
- **Idea, loosely, is that if X, Y, and Z are data products, and if f, g, and h are transformations on those data products to produce new data, then a user who requires f(X) should not care whether the data are retrieved from storage or computed on demand--the grid should be able to choose the appropriate strategy**
- **In ATLAS, Athena job options files/scripts can be thought of as defining transformations on an input specification**
- **A catalog of named job options files, possibly parameterized, would be a natural candidate for an early implementation of the proposed GriPhyN transformation catalog**



## Virtual data and Athena

- **Straightforward to specify by means of Athena Event Selector properties, instead of a selection predicate, the logical name of a job options file  $f$  (think “Calibrate1”, “DefaultReconstruct,” with parameters allowed) and an input collection  $X$  (“Raw Run N”) against which to run the job**
- **Net effect: trigger a preliminary Athena job that runs  $f$  on the data  $X$  to produce the input required by the triggering job, unless  $f(X)$  were already registered in a data catalog and available “nearby”**
- **An effort to realize this scenario, using Athena-based fast simulation (Atlfast), is a part of ATLAS GriPhyN plans**



## Some closing thoughts

- **Both Athena and the associated grid tools are in their infancies--this work is really just a first exploration, matched to the still-modest capabilities on both sides**
  - **this work should not be construed as the long-term ATLAS plan for grid-enabled offline software**
- **There has been progress in all of the products and projects mentioned in this talk, and many of them are evolving in ways that address the shortcomings we have encountered**
- **There are several interesting ongoing grid activities in U.S. ATLAS (and ATLAS worldwide) that are NOT represented in this talk or at this conference**
  - **cf. Magda (formerly DBYA) from BNL, GRAPPA, and much testbed activity--apologies to omitted projects**

