ATLAS Distributed Analysis: Model, Status and Plans

ARDA Workshop

David Adams
BNL

March 8, 2005

Contents

Relevant projects

Model

- Datasets
  - Dataset catalogs
- Transformations
  - Transformation descriptions
  - Transformation catalogs
- Schedulers
  - Analysis services
- Jobs
  - Job catalogs
- Clients

Current status

- ADA Architecture
- DIAL 1.00
- ADA 1.00

Demo

Integration with gLite

- Workload management
- Data management
- Software management
- Metadata
- Security
- Service discovery

Relevant projects

ADA = ATLAS Distributed Analysis

- Project to deliver a distributed analysis system for ATLAS

DIAL = Distributed Interactive Analysis of Large datasets

- Provides a C++ implementation of the ADA model
- Original goal to demonstrating the feasibility of the title

GANGA = Gaudi/Athena aNd Grid Alliance

- Joint LHCb/ATLAS project to develop a framework for distributed analysis
- ATLAS emphasis is providing a user interface

ARDA = A Realization of Distributed Analysis

- Project to help each of the four LHC experiments deliver prototype distributed analysis systems based on gLite

GLite

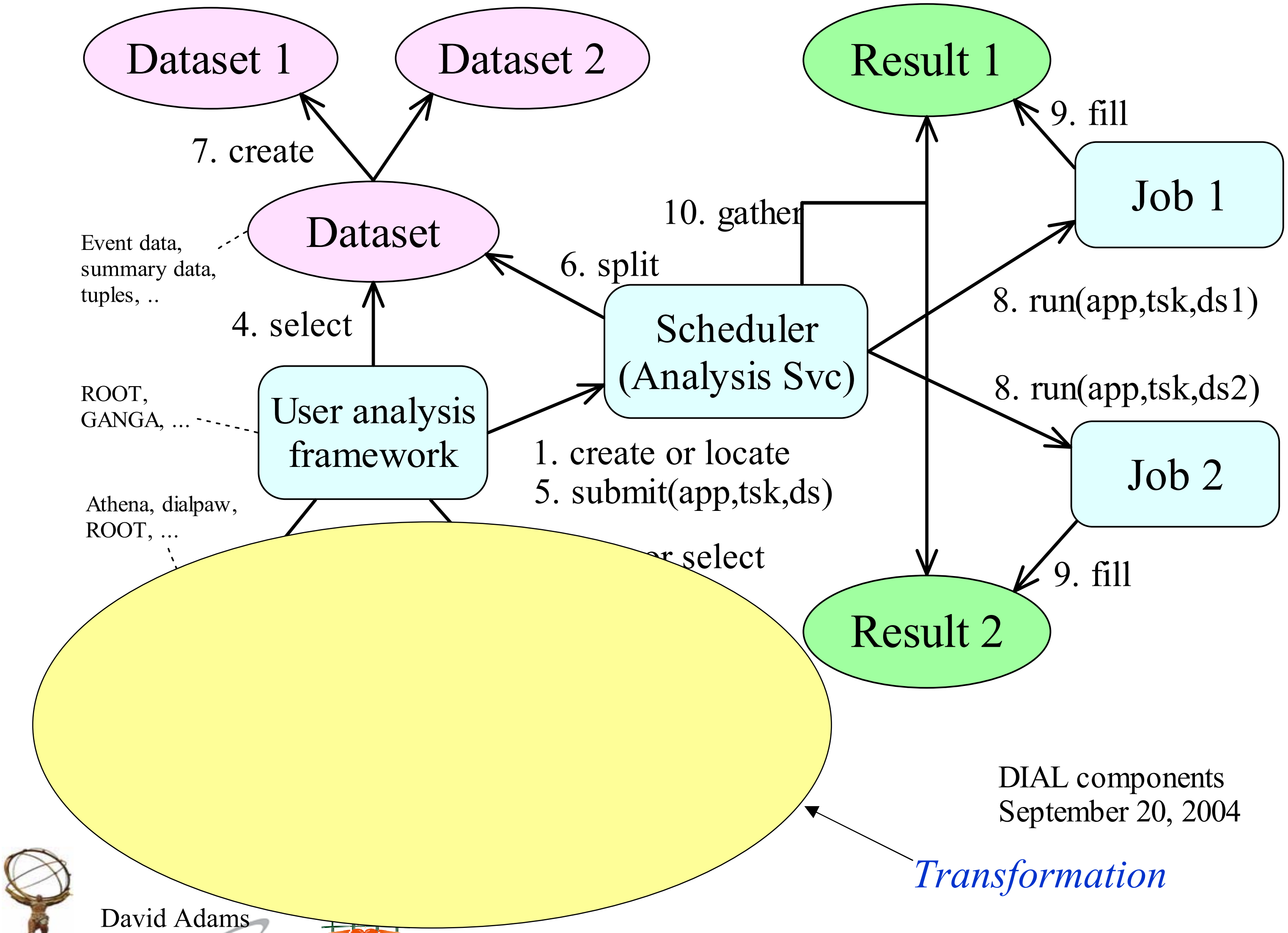
- EGEE middleware development project

Model

ADA model

- Data are described with datasets
- A transformation describes an operation that can act on a dataset to produce a new dataset
- A job is an instance of a transformation acting on a dataset
- An scheduler accepts job requests and carries out the processing
  - Split input dataset
  - Apply transformation to each sub-dataset
  - Merge output datasets
  - Often run as an “analysis service”
- Users interact using various clients
  - Create, examine, and locate datasets and transformations
  - Define and submit jobs
  - Monitor jobs

Model (cont)



Dataset 1

Dataset 2

7. create

Dataset

Event data, summary data, tuples, ..

4. select

6. split

10. gather

Result 1

9. fill

Job 1

Scheduler (Analysis Svc)

8. run(app,tsk,ds1)

8. run(app,tsk,ds2)

ROOT, GANGA, ...

User analysis framework

1. create or locate
5. submit(app,tsk,ds)

Job 2

Athena, dialpaw, ROOT, ...

r select

9. fill

Result 2

DIAL components
September 20, 2004

Transformation

Datasets

Dataset properties fall into many categories

- Immutable intrinsic properties
  - Included in XML description of the dataset
  - These are listed below
- Metadata
  - To aid in dataset selection
- Management
  - Ownership, access control and lifetime
- Provenance
- Placement
  - Where a physical copy (e.g. the files) of a dataset may be found

These properties are recorded in catalogs

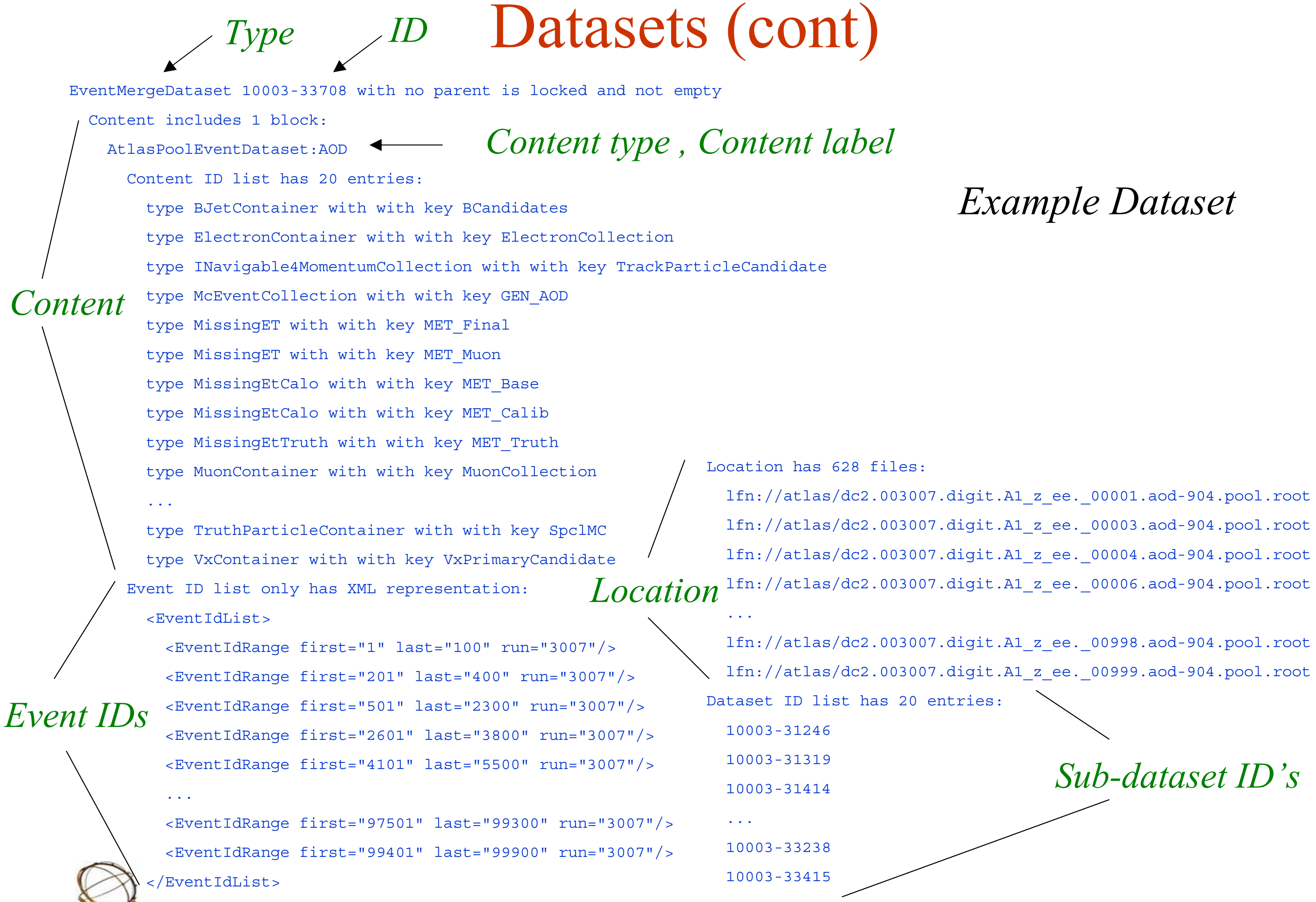
- Simple MySQL tables right now
- Want to move to web service interfaces

Datasets (cont)

Dataset intrinsic properties

- Identity
- Content description
  - List of event ID's
  - Content label or labels (ESD, AOD, etc.)
  - Detailed content description (tracks, jets, electrons, ...)
- Location
  - Typically a list of LFN's
- Sub-datasets
  - Expressed as a list of sub-dataset ID's
  - Datasets are hierarchical
- Mapping
  - How content is mapped to location
  - To enable splitting of the dataset
  - Typically handled with sub-datasets

Datasets (cont)



```
EventMergeDataset 10003-33708 with no parent is locked and not empty
  Content includes 1 block:
    AtlasPoolEventDataset:AOD
      Content ID list has 20 entries:
        type BJetContainer with with key BCandidates
        type ElectronContainer with with key ElectronCollection
        type INavigable4MomentumCollection with with key TrackParticleCandidate
        type McEventCollection with with key GEN_AOD
        type MissingET with with key MET_Final
        type MissingET with with key MET_Muon
        type MissingEtCalo with with key MET_Base
        type MissingEtCalo with with key MET_Calib
        type MissingEtTruth with with key MET_Truth
        type MuonContainer with with key MuonCollection
        ...
        type TruthParticleContainer with with key SpclMC
        type VxContainer with with key VxPrimaryCandidate
      Event ID list only has XML representation:
        <EventIdList>
          <EventIdRange first="1" last="100" run="3007"/>
          <EventIdRange first="201" last="400" run="3007"/>
          <EventIdRange first="501" last="2300" run="3007"/>
          <EventIdRange first="2601" last="3800" run="3007"/>
          <EventIdRange first="4101" last="5500" run="3007"/>
          ...
          <EventIdRange first="97501" last="99300" run="3007"/>
          <EventIdRange first="99401" last="99900" run="3007"/>
        </EventIdList>
```

```
Location has 628 files:
  lfn://atlas/dc2.003007.digit.A1_z_ee._00001.aod-904.pool.root
  lfn://atlas/dc2.003007.digit.A1_z_ee._00003.aod-904.pool.root
  lfn://atlas/dc2.003007.digit.A1_z_ee._00004.aod-904.pool.root
  lfn://atlas/dc2.003007.digit.A1_z_ee._00006.aod-904.pool.root
  ...
  lfn://atlas/dc2.003007.digit.A1_z_ee._00998.aod-904.pool.root
  lfn://atlas/dc2.003007.digit.A1_z_ee._00999.aod-904.pool.root
Dataset ID list has 20 entries:
  10003-31246
  10003-31319
  10003-31414
  ...
  10003-33238
  10003-33415
```

Dataset catalogs

DR = Dataset repository

- Catalog holding XML descriptions indexed by ID
  - Presently implemented as a single MySQL table
- Web service interface created but not yet deployed
- Do not expect/allow queries on intrinsic properties
  - All datasets (many more than DSC)

Dataset catalogs (cont)

DSC = Dataset Selection Catalog

- List datasets of interest to physicists
- Primary index is a unique dataset name
  - So datasets can be referenced by name
- A dataset ID is associated with each available dataset
  - Full dataset description can be obtained from the DR
  - Dataset reference by ID is immutable; by name is not
- Metadata attached to enable dataset selection
  - Replicate some intrinsic properties (content label, event count, …)
  - Provenance information
  - Add other properties such as quality, relevant physics groups, …
- Current (very rudimentary) table may be browsed at
  - http://www.atlasgrid.bnl.gov/dialds/dlShowMain.pl
  - Similar catalogs in AMI—these will merge

Dataset catalogs (cont)

DPC = Dataset placement catalog

- Lists the storage sites where placement datasets may be found
  - Placement dataset is one for which all LFN's are archived or staged at one or more single sites
- So jobs may be directed to sites where data is present
  - Without checking individual files
- This does not yet exist in ADA
  - Coming soon
- Will also want dataset placement service to copy dataset to a site and record result in DPC
  - Makes use of FTS, RC (or LSE wrapper)

Dataset catalogs (cont)

DMC = Dataset Management Catalog

- Does not yet exist
- Used to determine lifetimes for datasets
  - Essential for analysis activities where intermediate results and most user results are temporary
- Proposal is to introduce a claims mechanism
  - Multiple claims can be attached to a dataset
  - Each claim has an expiration time
  - Dataset may be deleted when all its claims are released or expired
  - Extend the same mechanism to files
    - File claims may derive from dataset claims
    - E.g. claiming a dataset may claim all the logical files associated with that dataset
    - Or claiming a dataset placement may claim all the replicas associate with that placement

Transformations

Definition

- A transformation acts on a dataset to produce a new dataset
  - Multiple input or output datasets can be handled by simple merging
- The new dataset is called the result of the transformation

Impose the following requirements

- A transformation can be written in a way that a runs on all ADA sites (any grid) and produces equivalent results
- A transformation applied to sub-datasets followed by merge of results is equivalent to applying the transformation directly to the original dataset
- The description of a transformation should distinguish processing common to all input data from that specific to the input data
  - Processing system is not required to repeat the former for each job
  - E.g. compilation and event processing
- Non-expert users can easily configure transformations

Transformations (cont)

ADA model

- Transformation is made up of application and task
- Application holds the scripts that carry out the processing
- Task carries the files used to configure the job
- Typical usage
  - User selects an application and example task
  - User extracts and modifies the files in the task
  - Modified files are used to create a new task
  - Application, task and dataset define a job

Application holds two scripts

- build_task carries out the common processing
  - Task files are made available in a build directory
- run carries out the processing specific to the input dataset
  - The build directory from the firs step is accessible (readonly)
- At present, the merge is implicit

Transformations (cont)

Assumed environment for transformation scripts:

- Posix commands run from /bin/sh
  - ls, echo, cat, …
  - Should we allow other shells (python, perl, …)?
- C/C++ compiler
  - Version is part of platform specification
- pkgmgr to provide access to other software
- pkgmgr_util – scripts to aid in setting up software packages
- Nothing else
  - CMT, ATLAS, DIAL are all accessed through pkgmgr

Transformations (cont)

Accessing software

- Assume one command from pkgmgr
  - pkgmgr locate tagname
- Returns the directory holding the software associated with tagname
- If the package is not available, the command fails
  - Most likely, the job also fails
- Transformation script sets up the software
  - If needed
  - Script *runcom* simplifies setup in a subprocess
    - To avoid polluting environment of top process
- Site selects the implementation of pkgmgr
  - One reference implementation (pkgmgr-simple) is available
- If site does not provide pkgmgr, it could be shipped with job
- Site choice to enable installation on demand

Transformations (cont)

Data access

- Present applications use DIAL-provided commands for locating input files and creating output datasets
  - – dataset_property –n dsname files
    - > LFN→PFN, stage files
  - – make_atlaspool_dataset –c AOD –m somefile.root
    - > PFN→LFN, archive, register
- DIAL would like common means for file access (DQ, LSE)
  - – To simplify implementation of the above commands
- Transformation developer is allowed to use other means to get and put data
  - – But input and output are dataset XML objects

Transformation descriptions

Application and Task have XML descriptions

Application description

- At present just a name and version
  - These map to an AFS directory where the scripts are found
- Likely will soon move to a model where the scripts are embedded
  - Name and version are moved to metadata catalog
  - Add unique ID

Task description

- Task files are embedded
- Could extend this to allow LFN’s for binaries and other large files
  - Would have to add standard interface for accessing replicas
  - Can obtain functionality at present with file holding LFN names

Do we want a description for transformations?

- Or just use Application and Task ID’s?

Transformation catalogs

ADA has not yet deployed transformation catalogs

- Some interfaces have been defined in DIAL
- Some empty tables have been created in MySQL

Anticipate the following

- Application and task repositories
  - Descriptions indexed by ID
- Application selection catalog
  - Map name and name+version to ID
  - Attach metadata to aid in selection
- Task selection catalog
- Placement catalogs might be desirable
  - Or just check for presence of required software
- Management catalogs may not be needed
  - Transformations live for ever or keep those required for provenance

Schedulers

Scheduler interface

- Accept job submission from user
  - Application, task, dataset and preferences
- Return job status on request
  - Fraction of events and sub-jobs completed
  - Status of each sub-job
  - Partial and, ultimately, complete results

DIAL provides schedulers for local processing

- Fork, LSF and Condor job submission
- Easy to extend to other batch systems

DIAL also provides a scheduler web service

- Usually called the "analysis service"
- And a client to this service which has the scheduler interface
  - Local and analysis service submissions have the same look and feel

Schedulers (cont)

Job processing

- Input dataset may be split
  - Now split is done by going down one level in the dataset hierarchy
  - Interface exists to plug in other splitting strategies
    - Based on available resources, desired response time, …
  - Splitting strategy could be specified in job preferences
  - Each sub-dataset is used to define a sub-job with same transformation
- Sub-jobs typically sent to another scheduler
  - Local scheduler or one making use of a grid WMS
    - Likely no further splitting
  - Or job may be handed to another analysis service
    - Choice may be based on dataset and software placement and on available compute and storage resources
    - Different jobs may go to different sites
    - Further splitting likely

Schedulers (cont)

Job processing (cont)

- Scheduler maintains a result dataset for each job
  - A sub-jobs complete, their results are merged into this result
    - > At present this is done in the scheduler
    - > Work in progress to submit jobs to do this
    - > Effect is a *dynamic* DAG
- If any sub-jobs fail, the overall job is marked as failed
  - Clear need at present to
    - > Resubmit failed jobs
    - > Kill and resubmit jobs that run too slowly

User requests are typically directed to an analysis service

- Service can be tuned and assigned resources to meet demand
- Processing continues when user disconnects

Analysis services

Service discovery

- Need means for users to discover the appropriate service for job submission
- Service receiving request may want to discover more appropriate services to handle part or all of the request
- Information services to
  - Find analysis services
  - Select those meeting requirements for
    - > Available CPU and storage
    - > Archived or staged data
    - > Installed software

Jobs

A job is an instance of a transformation acting on a dataset

Jobs are hierarchical

- Job may hold a collection of sub-jobs
- Sub-job collection may be dynamic
  - Depending on the order in which jobs complete
  - To recover from failures
  - Not yet in DIAL

Typically each job has a manager and client

- Manager is overseeing the processing
  - Analysis service, batch system, WMS or CE
- Client is waiting for the result of the job
  - User, analysis service or WMS

DIAL provides an XML description of each job

- Dynamic: changing until the job is complete

Jobs (cont)

Job description includes

- Identity of application, task and dataset
- Identity of partial or complete result
- Submit, start, stop and update times
- State (running, completed, failed, …)
- Identities of sub-jobs

Scheduler holds job descriptions

- Of each job that it manages
- Does regular updates of jobs that are not complete
  - Polling or waiting for signal
- Need a persistent record of these descriptions
  - Updated by job manager (scheduler)
  - Maintain availability after scheduler termination or restart
  - Monitor jobs without contacting scheduler

Jobs (cont)

```
CompoundJob 501-108370 is running
Application: atlasopt 1.00   Transformation          Example job
Task 102-194
Dataset 10003-33708 with 62680 events   Input dataset
Job preferences ID 0-0
Run host: atlasgrid09.usatlas.bnl.gov
Job directory: /usatlas/u/dial/local/jobs/MasterScheduler/00/00/01/f5/00/01/a7/52
 create time: 2005 March 06 15:22:32
  start time: 2005 March 06 15:23:22 (50 sec elapsed)
 update time: 2005 March 06 15:33:07 (635 sec elapsed)
There are 20 subjobs
  12 running
  8 done
  0 failed
  0 killed
  8 included in result
Events processed: 24800 (39%)
       in result: 24800 (39%)
Result dataset 10003-38319 is not an event dataset   Output dataset
```

Job catalogs

Clear need for job repository

- Job descriptions indexed by unique ID
- Per site or per scheduler instance to record all jobs
- Component of provenance/history service

Provenance catalog

- Require that we record the provenance and full processing history of any published dataset
- Support queries (unlike repository)
  - Application, task and datasets
  - Processing site and/or scheduler instance
- Job ID provides link to full description in repository

Job history

- Success rates, processing times, environment, resources consumed
- Aids in the prediction of job behavior

Clients

Goal of clients

- To provide users with easy access to ADA
- Provide standard interfaces to combine pieces of the system
  - E.g. to enable application scripts to fetch and store data

ROOT-based client

- Almost all DIAL classes are imported into ROOT and available at the ROOT command line
- Convenient integration with usual analysis tools
  - E.g. from ROOT, submit a job, monitor its progress and view histograms of partial results

Clients (cont)

DIAL provides a number of command line tools

- dial_submit to submit and monitor a job
- dataset_property to examine the properties of a dataset
- make_XXX_dataset to construct and dataset and move a file into the FMS (file management system)
- uidtest to validate connectivity and authentication

Web pages

- A PHP interface to the ADA catalogs
  - E.g. to select a dataset
- Expect more, e.g. to monitor services
  - Or job submission from AMI

Clients (cont)

A python interface is available

- Developed as part of the GANGA project
- Uses lcgdict to wrap most DIAL classes
- Can enable users to interact with ADA from GANGA or interactive athena
- Long-term ambition to remove DIAL C++ dependency and provide lightweight client

GUI

- GANGA is developing a GUI using the python binding
- Enable job submission
  - Select application, task and dataset
  - Modify task and record new version
  - Submit job
- Job monitoring (need GANGA 4)

Current status

Summarized in architecture diagram (cartoon)

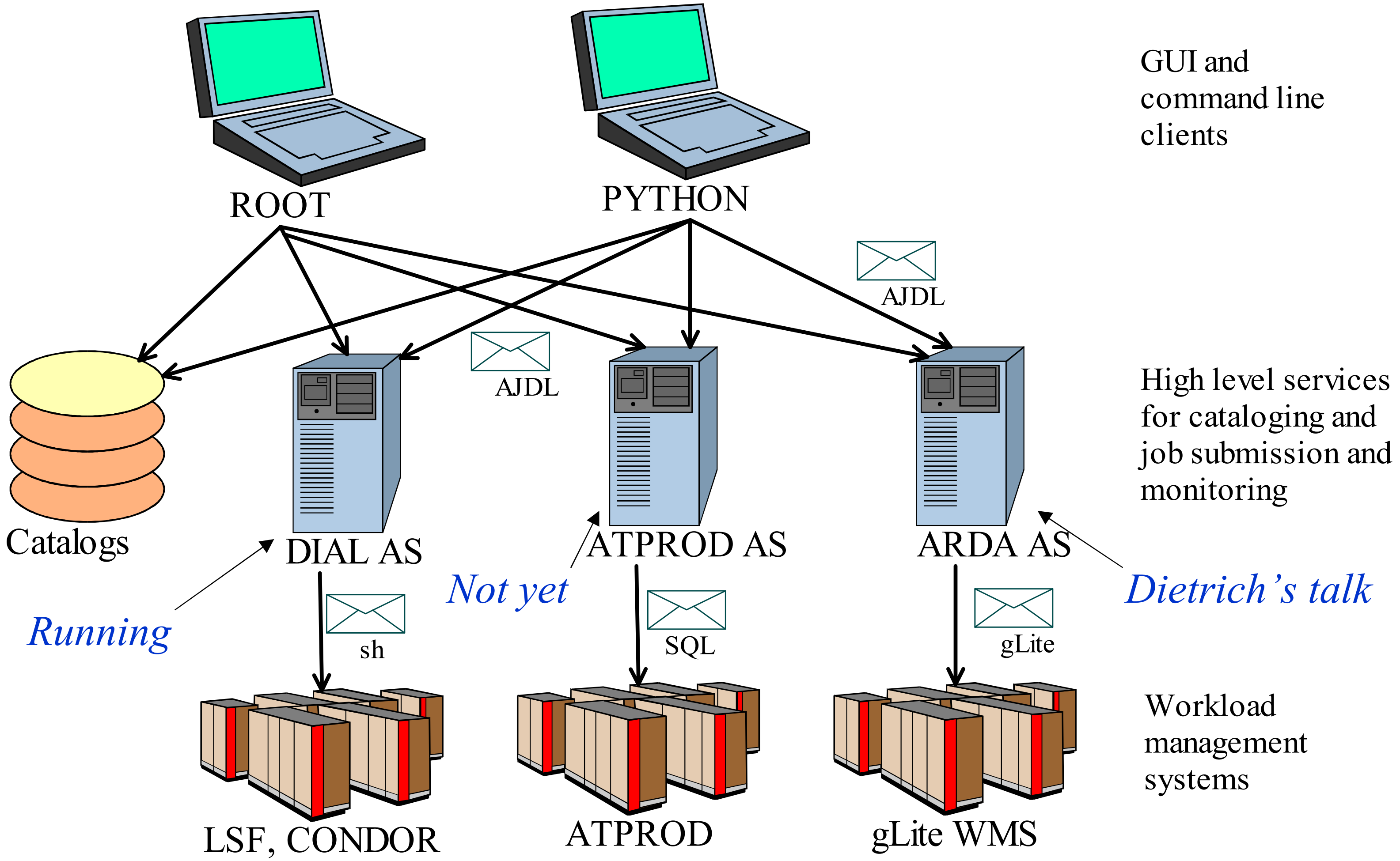
Recently made DIAL release 1.00

ADA release 1.00 built on this

- Adding transformations and datasets

Dietrich will describe integration with gLite release 1

ADA Architecture



GUI and command line clients

High level services for cataloging and job submission and monitoring

Workload management systems

DIAL 1.00

DIAL release 1.00 made in mid-February

- Changes described in the following sections

Portable build system

- Enables builds at other sites and on other platforms
- Based on GNU tools: make, autoconf, auotmake, libtool
- Modules described using CTEST interface (as before)

Package management system

- Interface pkgmgr
- Reference implementation: pkgmgr-simple
- Used by build system to find software
- Used at run time to locate dial and external software
  - DIAL server, clients and utilities
- Used by transformations to find software on worker nodes

DIAL 1.00 (cont)

Dataset location described with file reference URL’s

- Typically LFN that may be located in many ways:
  - DQ, magda, RLS, …
- Still may be a “physical file”
  - file:myfile, nfs:/home/myfile, srm://someserver/atlas/myfile, …

Upgrade external software

- Root 4.02.00
- ATLAS 9.4.0
- XercesC 2.6.0
- Gsoap and gsoap-gsi to latest versions

Improvements in robustness of magda file transfer

- Still some problems here

Distribution for RH73 and RHE3 (certified on SLC3)

ADA 1.00

ADA uses DIAL framework but includes much more

- Described below
- More work required here for this release to be useful

ATLAS-specific catalogs

- Progress in defining schema and interface for DSC
  - Dataset selection catalog
  - Where users go to find datasets

Atlas datasets

- At present a few small samples and one large AOD dataset
- Multi-level structure
  - Single files
  - Sensible grouping for single sub-job (e.g. 50-100 AOD files)
  - Site placement
  - Full dataset (all data meeting some conditions)

ADA 1.00 (cont)

Transformations

- Clarified the environment in which jobs run
  - Posix plus pkgmgr
  - Use pkgmgr to locate other software
    - > DIAL, ATLAS, CMT, …
- Now support compound output datasets
  - Any combination of HIST, NTUP, AOD, ESD, …
- Progress on existing transformations: atlasreco and aodhisto
- New transformation: atlasopt
  - Event data ➔ histograms and ntuples
  - User provides atlas release and job options
- Work with physicists doing analysis to define others
  - E.g. from arbitrary changes in user development area

Demo

Install DIAL

- http://www.usatlas.bnl.gov/~dladams/dial/releases/1.00
- Already done at CERN and BNL

Select application

- http://www.usatlas.bnl.gov/~dladams/dial/releases/1.00/apps.html

Install the corresponding demo

- http://www.usatlas.bnl.gov/~dladams/dial/releases/1.00/dial/dial_root/demos/demo6/README

Use query page to select a dataset

- http://www.atlasgrid.bnl.gov/dialds/dlShowMain.pl

Submit job

Monitor progress

Display results (partial or complete)

Integration with gLite

ADA model is a higher level of abstraction than gLite

- Datasets and compound jobs instead of files and simple jobs
- Will or should gLite deliver something at the dataset level?
  - Probably not
  - If so, on what time scale?
- Presumably gLite would like consensus view from LHC experiments
  - Can be difficult to get this within any single experiment

Consider topics separately

- Workload management
- Data management
- Software management
- Metadata
- Security
- Service discovery

Workload management

Compound jobs

- Can gLite handle ADA compound jobs?
  - DAG is dynamic
    - > merge jobs as they finish; not in predefined order
    - > Substitute new jobs for jobs the fail or hang
  - Often want all processing for placement dataset at the same site
    - > WMS should place DAG—not individual jobs
    - > No need to move data between sites
    - > Result is naturally at the same site as the input

Partial results

- Need to get results back before compound job has completed

Interactive analysis

- Fast response time for job execution and fetching output data

Workload management (cont)

Job placement

- ADA wants to place jobs based on dataset placement
  - Not on file placement
  - Does gLite have place for VO input to job placement?
- Also place jobs based on software placement

Data placement

- For some jobs, need to move data
- Want to stage data in any case
  - Like to specify acceptable formats
    - E.g. rfio, glite, dcap and posix
- Jobs should start as soon as but not before the above

Software installation

- Check software presence before submission
- Install if necessary

Data management

Datasets

- Do we find any support in gLite for the concept?
- If so, is it consistent with ADA model?

FMS (file management system)

- In any case our DSMS (dataset management system) makes extensive use of the FMS
- ADA expects site-based VO file replica catalogs
  - Listing only the files archived or staged at the site
- Expect FTS (file transfer service)
  - Site based
  - To queue and prioritize transfers to the site
- SRM-like staging with claims (pinning)
- Proposal: sites should have LSE interface

Data management

LSE

- Main functionality
  - Put: create logical file and replicate from input PFN
  - Stage: make replica available for access with specified protocol
  - Copy: archive replica
- Input to all is one or a list of LFN’s
- Similar to SRM but with LFN’s replacing SURL’s
- Interface (ideally very thin) over
  - Replica catalog
  - File transfer service
    - > Managing incoming data movement
  - Local staging (SRM)
- Enables interoperability between sites
  - E.g. Phedex, DQ and gLite DMS

Software management

Locate

- ADA expects sites provide pkgmgr interface to locate software
  - Or include this as part of submitted job

Placement

- Need means to query the existence of software at a site
- And means to install it when absent

Specification

- Above require means to specify software packages
  - Name, version, platform
  - Platform-independent aliases
    - > E.g. atlas-10.0.0 points to atlas-10.0.0-slc3

Metadata

Whose service?

- Should ADA be using gLite, ARDA or ATLAS-specific services
- Don’t envision queries returning many entries
  - So web services are OK/preferred?

Security

GSI

- Do we agree on use of GSI with transport level security?
- Does this make our services interoperable?
  - DQ reports issues with different versions of gsoap
- Mechanisms for authorization

User ID's

- ADA would like to have mechanism to map DN's to VO ID
  - Rather than using DN itself as ID
- This VO ID would be the basis for accounting, access control, prioritization, etc.

Also need support for roles

- Consistent across grids

Service discovery

ADA needs mechanism for service discovery

- Should be consistent across grids

Conclusions

ADA model is evolving and crystallizing

- Experience managing an end-to-end system
- Interaction with other projects
  - ARDA, gLite, ATLAS production, …

Serious use by ATLAS physicists scheduled for next month

- When “Rome” data is available

ATLAS ARDA prototype

- Requires deployment of gLite on significant resources
- Interesting ATLAS data must be placed at these sites
- Required software (ATLAS, DIAL) must be installed and registered with pkgmgr
- Which gLite services should we use and how?
  - As described by Dietrich?