

Distributed analysis in the ATLAS experiment

D.L. Adams^a, F. Brochu^b, K. Harrison^b, R.W.L. Jones^c, G. Rybkin^d, C.L. Tan^e

^a *BNL, Upton, NY 11973-5000, USA*

^b *Cavendish Laboratory, University of Cambridge, CB3 0HE, UK*

^c *Department of Physics, University of Lancaster, LA1 4YB, UK*

^d *Department of Physics, Royal Holloway, University of London, TW20 0EX, UK*

^e *School of Physics and Astronomy, University of Birmingham, B15 2TT, UK*

Abstract

A distributed system is being developed to support analysis activities in the ATLAS experiment, which will investigate proton-proton interactions at the Large Hadron Collider at CERN, Geneva, and will require processing of data volumes of the order of petabytes per year. The ATLAS Distributed Analysis (ADA) system will enable access to distributed data and resources for a worldwide community of hundreds of physicists. It must work across major Grid developments such as LCG, Grid3 and NorduGrid, and emerging new systems, such as gLite and OSG. This paper describes the components of ADA, and reports on the system's status.

1 Introduction

The ATLAS experiment [1] will investigate high-energy proton-proton interactions at the Large Hadron Collider (LHC) [2], due to start operation at the European Laboratory for Particle Physics (CERN), Geneva, in 2007. Interactions with potential physics interest will trigger readout of the ATLAS detector, and data will be recorded for an estimated 10^9 triggers per year. Analysis of the triggered interactions, including associated simulation studies, will require offline processing of a data volume of 5-10 petabytes.

ATLAS involves a collaboration of more than 1800 physicists, from over 150 institutes in 34 countries, spread over 6 continents. The experimental data, recorded at CERN, will be shared between national and regional centres for processing. This includes collaboration-wide production, where interactions are reconstructed, and are tagged according to their characteristics; and analysis, where small work groups and individual physicists are interested in reconstructed interactions with particular tags.

The ATLAS Distributed Analysis (ADA) system [3] is being developed to support the experiment's analysis activities, providing for a worldwide user community that must access distributed data and resources. ADA is required to work across major Grid developments such as LCG (LHC Computing Grid) [4], Grid3 (US Application Grid Laboratory for Science) [5] and NorduGrid (evolved from the Nordic Testbed for Wide Area Computing and Data Handling) [6], and emerging new systems, such as gLite (Lightweight Middleware for Grid Computing) [7] and OSG (Open Science Grid) [8]. ADA must also provide provenance tracking, to ensure that an analysis carried out by one physicist can be readily checked by another.

This paper outlines the ADA model, gives a description of the system components, and reports on the current status.

2 ADA overview

ADA is based on a client-service architecture, where the main components (Fig. 1) are analysis services to manage processing, catalogue services to record data and its provenance, clients with which the user interacts with these services, and an Abstract Job Definition Language (AJDL) used to format messages between clients and services. ADA additionally has components that deal with data transfer, software management, and monitoring.

The first release of ADA benefits significantly from work in related projects:

- The ADA model, many service implementations and a number of client tools have come from DIAL [9].
- An analysis service has been contributed by ARDA (A Realisation of Distributed Analysis for LHC) [10].
- Monitoring and a client including a Graphical User Interface (GUI) have been provided by GANGA [11].

3 AJDL

In AJDL, a job is represented as a transformation that acts on a dataset. A transformation, here, is essentially a prescription for building and running an application (for example, the ATLAS reconstruction), and a task definition, which is the user's specification of how the application should be configured. The task definition can include parameter values, file references, and source code to be compiled into a shared library that the application should load. At the lowest level, a dataset is typically a collection of files and some description of their content.

AJDL additionally allows a user to give preferences for how a job is processed. These preferences can indicate, for example, where and when a job

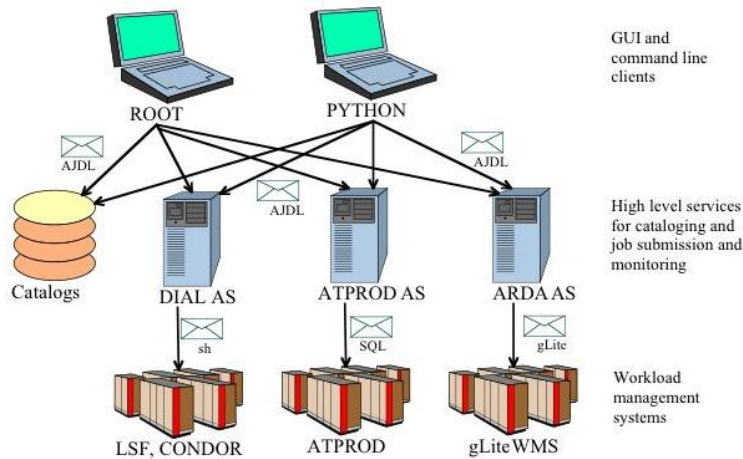


Figure 1: ADA architecture

should run; the priority to be assigned to the job; if, and how, job splitting should be performed.

ADA makes use of the AJDL implementation of DIAL, where the various components (application, task, different types of dataset, preferences) are defined in terms of C++ classes. These implement methods that permit AJDL objects to be converted to, and created from, descriptions formulated in XML.

4 Analysis services

An analysis service in ADA is essentially a Workload Management System (WMS) accessed via a web-service interface that accepts jobs represented in AJDL. Where possible, and not in conflict with user preferences, the analysis service splits a job into subjobs, distributing among them the data from the originally specified dataset, then submits the subjobs to the WMS. As subjobs complete, the analysis service merges their results and makes the merged result available to the submitter. A hierarchy of analysis services can be formed by using a second analysis service as the backend WMS to the first.

The first release of ADA runs analysis services providing access to LSF [12] and Condor [13] resources at Brookhaven National Laboratory (BNL) in the US, and to LSF and gLite [7] resources at CERN.

Work is in progress to add an analysis service that provides access to the system developed in ATLAS for large-scale production activities. The ATLAS Production System [14] uses a central database that is filled with job requests in a particular format, and on each of the ATLAS Grid flavours (Grid3, LCG, NorduGrid) runs services that process the requests, update the database when there are changes in job status, and catalogue the data produced. The analysis service built on top of this system will create job requests in the database, and periodically query status until the submitted jobs complete.

5 Software management

Software needed at a worker node to apply the transformation specified in a job request is located using a lightweight package manager [15], which is an interface to a database that stores details of available software packages. This package manager is implemented in Python and included in the ADA release. It allows packages to be registered in, or deleted from, the the database, and provides commands for obtaining information such as: the current platform, the path to the directory where a particular package is installed, and the list of all registered packages.

The same package manager is used for locating external software in the ADA build process, and by ADA clients.

6 Catalogue services and data management

ADA catalogue services are web-service interfaces to various databases and storage systems. For each of the AJDL components of application, task and dataset, ADA provides a selection catalogue and a repository; and for datasets it additionally provides a replica catalogue.

The selection catalogues can be queried to obtain information on the properties of stored AJDL objects that satisfy some search criteria. For all types of object, these properties include a name and a unique identifier. The repositories store XML descriptions of AJDL objects, and allows a copy of an object representation to be retrieved by specifying the object's identifier.

The dataset replica catalogue stores one-to-many mappings from the identifiers of logical datasets to the identifiers of copies.

Data management in ADA relies on the DIAL file-catalogue interface, which provides for the storage and retrieval of physical files identified by logical names. The interface has been implemented for file systems (NFS and AFS) and for the MAGDA (Manager for Grid-based Data) system [16]. The longer-

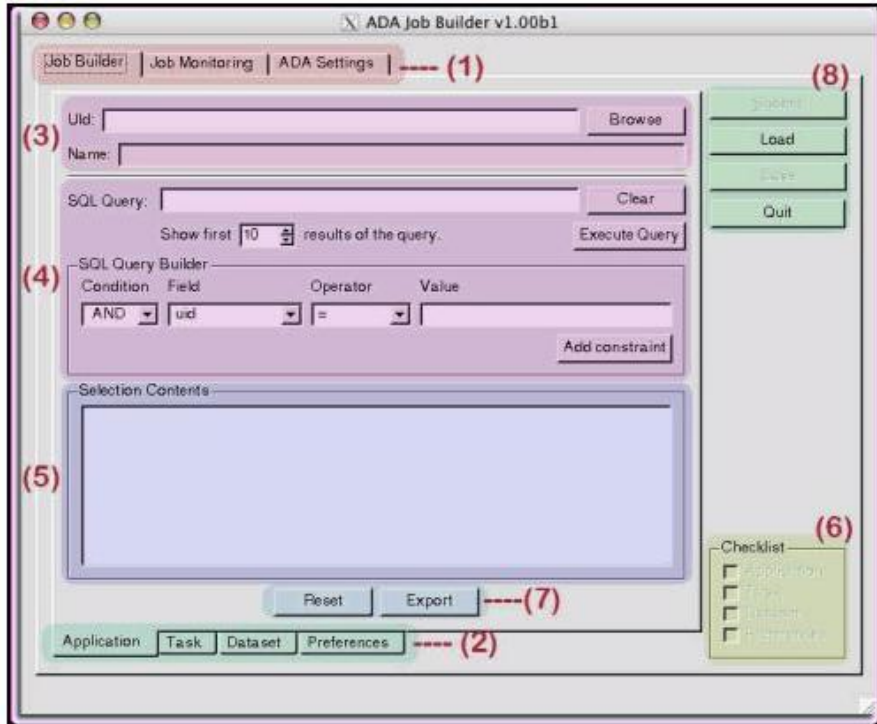


Figure 2: Screenshot of ADA GUI main window. Features shown include: tabs for the main functions of Job Builder, Job Monitor, and General Setup (1); for the Job Builder, tabs for the AJDL components of application, task, dataset and preferences (2), a checklist of defined components (7), and action buttons (8); for the application component, possibilities for selecting the required application by browsing locally (3), or by querying the catalogue services (4, 5).

term plan is to make use of the data-management tool DON QUIJOTE [17], currently under development in ATLAS.

7 Monitoring

Monitoring is a loosely used term that can mean anything from watching something of interest to actively, even automatically, applying complex actions to some object based on its state. Monitoring in ADA is performed using the GANGA [11] monitoring component, a pragmatically simple but flexible component designed with the above loose terminology in mind. This component can provide simple updates of state, but also allows more complex actions, defined by the developer, to be associated with state changes.

Like ADA, the GANGA monitoring component is based on a client-service architecture. The monitoring client is a generic component that communicates with one or more monitoring services, which can be local or remote. Each monitoring service is a specialised component, with knowledge of how to query a particular backend system. Monitoring clients don't actively request information, but register their interest with relevant monitoring services. Each service periodically queries its associated backend, caches the results, and alerts registered clients of any changes.

In the case of ADA, two monitoring services have been implemented. One queries all of the other ADA services for their states, and changes reported to a permanently running client are published on a web page. The other monitoring service queries the analysis services for information on submitted jobs, and is of interest to clients connected with a user interface.

8 User interfaces

Users can interact with ADA services from the command line, from the ROOT [18] analysis framework, from a Python environment, or through a Python-based Graphical User Interface (GUI).

The command-line tools, based on executables compiled from C++ programs that make use of DIAL classes, provide basic functionality for job definition, submission and monitoring.

The ROOT interface gives access to DIAL classes via dictionaries generated by parsing the DIAL headers with the ROOT tool ACLiC. Users can query the ADA selection catalogues, create and examine applications, tasks and datasets, submit jobs to the analysis services, and monitor job progress. Result datasets of analysis jobs often include files of ntuples and histograms, which can readily be displayed and manipulated in the ROOT environment.

PYDIAL [19] has been developed in the context

of the GANGA [11] project to provide Python tools for accessing ADA services and catalogues. It currently wraps the DIAL C++ classes, using tools from the LCG SEAL [20] project, and implements Python functions that simplify their use. The longer-term plan is to have a pure Python implementation providing similar, or better, capabilities.

PYDIAL can be used as a standalone user interface to ADA, or can be used indirectly from the GANGA job-management system. In both cases, the resultant functionality is similar to that from ROOT.

PYDIAL is also the basis of the ADA GUI [21] (Fig. 2), which offers intuitive, user-friendly tools for querying the ADA selection catalogues, creating and editing jobs, monitoring job status, and examining results.

9 Status and outlook

A first release of the ATLAS Distributed Analysis (ADA) system has been made, including all of the components described. Catalogue services have been running stably at BNL for several months, and analysis services have been running stably at both BNL and CERN. Various transformations useful for analysis activities have been written, and datasets for a range of ongoing physics studies have been catalogued. The ADA client tools have been used successfully to formulate job requests in AJDL, and to run jobs using analysis services available both locally (submission from CERN to CERN, and from BNL to BNL) and remotely (submission from CERN to BNL, and vice versa).

Physics groups have been making first tests with ADA, and have seen significant reductions in the time required to obtain results. As an example, the time for the analysis of one group, running on a sample of 146000 simulated interactions, is reduced from about 3 hours for a job submitted to the local batch system to 6.5 minutes for a job submitted to the LSF analysis service at BNL. In the latter case, the job is split into 80 sub-jobs, and the completion time achieved indicates that, unsurprisingly, there are overheads in creating the sub-jobs, running them, and merging their output. The contributions to these overheads have not been studied in detail, and there is scope for improvement.

The immediate challenges for ADA are to increase the resources available, in particular by adding analysis services that provide access to large-scale Grids, and to optimise the system's response time.

Although ADA has been developed to meet the needs of the ATLAS experiment for performing data analysis with distributed resources, it is noted that it is a generic system. It is potentially useful for other groups, who would have to define the types of transformations and datasets relevant to their domain, and might make different choices for the analysis services to be run, and the underlying data-management tools.

Acknowledgments

We are pleased to acknowledge support from GridPP in the UK, and from PPDG in the US.

References

- [1] ATLAS Collaboration, *Atlas - Technical Proposal*, CERN/LHCC94-43 (1994); <http://atlas.web.cern.ch/Atlas/>
- [2] LHC Study Group, *The LHC conceptual design report*, CERN/AC/95-05 (1995); <http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/>
- [3] <http://www.usatlas.bnl.gov/ADA/>
- [4] <http://lcg.web.cern.ch/lcg/>
- [5] <http://www.ivdgl.org/grid2003/>
- [6] <http://www.nordugrid.org/>
- [7] <http://glite.web.cern.ch/glite/>
- [8] <http://www.opensciencegrid.org/>
- [9] D. Adams et al., *DIAL - Distributed Interactive Analysis of Large datasets*, Proc. 2003 Conference for Computing in High Energy and Nuclear Physics, La Jolla, California, USA; <http://www.usatlas.bnl.gov/~dladams/dial/>
- [10] <http://lcg.web.cern.ch/LCG/activities/arda/arda.html>
- [11] U. Egede et al., *Ganga user interface for job definition and management*, Proc. 4th UK e-Science All-Hands Meeting, Nottingham, UK (2005); <http://ganga.web.cern.ch/ganga/>
- [12] <http://www.platform.com/products/LSF/>
- [13] <http://www.cs.wisc.edu/condor/>
- [14] F. Brochu et al., *Grid-based production and operations in the ATLAS experiment*, Proc. 4th UK e-Science All-Hands Meeting, Nottingham, UK (2005); <http://http://www.nordugrid.org/applications/prodsys/>
- [15] <http://www.pp.rhul.ac.uk/~rybkine/packagemgr/>
- [16] <http://www.atlasgrid.bnl.gov/magda/info/>
- [17] <http://atlas.web.cern.ch/Atlas/GROUPS/DATABASE/project/ddm/>
- [18] <http://root.cern.ch/>
- [19] <http://ganga.web.cern.ch/ganga/pydial/>
- [20] <http://seal.web.cern.ch/seal/>
- [21] <http://ganga.web.cern.ch/ganga/ADAJB/>