The ATLAS Production System Evolution: New Data Processing and Analysis Paradigm for the LHC Run2 and High-Luminosity

To cite this article: F H Barreiro et al 2017 J. Phys.: Conf. Ser. 898 052016

View the article online for updates and enhancements.
The ATLAS Production System Evolution: New Data Processing and Analysis Paradigm for the LHC Run2 and High-Luminosity

F H Barreiro\textsuperscript{1}, M Borodin\textsuperscript{2,5}, K De\textsuperscript{1}, D Golubkov\textsuperscript{3,5}, A Klimentov\textsuperscript{4}, T Maeno\textsuperscript{4}, R Mashinistov\textsuperscript{4}, S Padowski\textsuperscript{4}, T Wenaus\textsuperscript{4} on behalf of the ATLAS Collaboration

\textsuperscript{1} University of Texas at Arlington, Arlington, TX 76019, USA.
\textsuperscript{2} The University of Iowa, Iowa City, IA 52242, USA.
\textsuperscript{3} State Research Center Institute for High Energy Physics, Protvino, Russia
\textsuperscript{4} Brookhaven National Laboratory, Upton, NY 11973, USA.
\textsuperscript{5} National Research Center “Kurchatov Institute”, Moscow, Russia

E-mail: mikhail.borodin@cern.ch

Abstract. The second generation of the ATLAS Production System called ProdSys2 is a distributed workload manager that runs daily hundreds of thousands of jobs, from dozens of different ATLAS specific workflows, across more than hundred heterogeneous sites. It achieves high utilization by combining dynamic job definition based on many criteria, such as input and output size, memory requirements and CPU consumption, with manageable scheduling policies and by supporting different kind of computational resources, such as GRID, clouds, supercomputers and volunteer-computers. The system dynamically assigns a group of jobs (task) to a group of geographically distributed computing resources. Dynamic assignment and resources utilization is one of the major features of the system, it didn’t exist in the earliest versions of the production system where Grid resources topology was predefined using national or/and geographical pattern. Production System has a sophisticated job fault-recovery mechanism, which efficiently allows to run multi-Terabyte tasks without human intervention. We have implemented “train” model and open-ended production which allow to submit tasks automatically as soon as new set of data is available and to chain physics groups data processing and analysis with central production by the experiment. We present an overview of the ATLAS Production System and its major components features and architecture: task definition, web user interface and monitoring. We describe the important design decisions and lessons learned from an operational experience during the first year of LHC Run2. We also report the performance of the designed system and how various workflows, such as data (re)processing, Monte-Carlo and physics group production, users analysis, are scheduled and executed within one production system on heterogeneous computing resources.

1. Introduction

The Production and Distributed Analysis (PanDA) \cite{1} software system handles the workload management of the ATLAS experiment \cite{2} on distributed Worldwide LHC Computing Grid (WLCG) resources. Major upgrades to PanDA were made during the shutdown between LHC Run 1 and Run 2, based on the experience gained from Run 1. The ATLAS Production System works at Exabyte scale...
of data with more than 250,000 CPU cores in use simultaneously. This paper will present a brief overview of some of the software components that manage the variety of ATLAS workflows implemented in PanDA.

1. Production system components
The ATLAS Production System is designed to run jobs on a diverse collection of computing resources, for a large number of users, with many different and challenging workflows. To meet this complexity, the system consists of several independent centralized service layers (Figure 1):

- PanDA server is a multithreaded application responsible for generating jobs on the worker nodes.
- JEDI (Job Execution and Definition Interface) is a layer responsible for Task-level workflow management. The key feature of JEDI is dynamic job definition: optimizing job parameters depending on available resources. Dynamic job definition plays a crucial role in running multithreaded jobs and for non-Grid resources such as High-performance computing (HPC) and Clouds.
- The Database Engine for Tasks (DEFT) is a system responsible for task definition as well as for storing task state.
- The Web user interface which allows the main system users, ATLAS production managers, to interact with the task management system. A production manager represents a group activity in the ATLAS experiment and defines workflows for their group: simulation, first processing, reprocessing, derivation and high level trigger reprocessing. Each group has its own data flow and different requirements for tasks.

![Figure 1. Production system components](image)

2. DEFT data model
Each step of analysis, data processing or simulation workflow in ATLAS is treated as a separate transformation that transforms an input dataset into an output dataset using a set of parameters, for example the software version. Using this concept of transformation which worked well during Run 1 data taking at the LHC, we developed a data model (Figure 2) for DEFT that covers all current workflows and is flexible for new workflows in the future. The data model is the core of all development and achieves two main goals: rapid task definition and bookkeeping. It is implemented in several abstraction levels, which are supersets of each other:

- The production request is the highest level, which combines production of similar tasks, for instance MC (Monte-Carlo) production of some physics group or reprocessing campaign. Production requests include general information for all future tasks, as the energy, manager and description. The production request status is used for request manipulation.
• The slice is a combination of some related steps. A slice could be the child of another slice; therefore it allows the creation of a tree-like structure. It doesn’t have any unique meta information, but is heavily used for bookkeeping.

• A step is a leaf instance that contains parameters that are required for a task definition. An initiated step becomes a task after applying the task definition process.

Figure 2. Part of the DEFT data model

Every instance (production requests, slices, steps) could play the role of a template. In this case only a few of the parameters are set. The template approach is frequently used since most production requests are created and modified by different users.

3. Production system workflows

ATLAS has dozens of workflows used by many different groups of users. All workflows have a similar flow of actions, shown in Figure 3. First, user submits data to the production system: it could be through an external source, such as a structured google spreadsheet, or internal source, like a web form or through previously created templates. After data is submitted, production managers begin preparation of the request: define missing parameters by using patterns from templates. After the request is ready, the production manager submits the request. In this stage a task is initiated using the task definition. If the task could not be defined, an error is created and the user is notified. Defined tasks are run by JEDI and can be monitored and manipulated by the production system web interface. The user receives a notification if the task could not be submitted. After a task is done, postproduction is initiated, whose primary purpose is to fill metadata and delete datasets that are no longer needed. We have attempted to minimize the preparation time for task definition since it is critical for overall system performance.

The main types of production in ATLAS are MC simulation, reprocessing and derivation production. Each of them has a specific workflow that is implemented within the framework of the production system.
3.1. MC workflow

MC simulation is the biggest and most complex part of the production system. Dozens of physics groups request MC simulation for hundreds of different physics studies.

MC production uses strictly structured google spreadsheet as input. It contains transformation description, number of events and comments. Uploaded spreadsheets are automatically converted to system instances: slices and steps. After that, physics management can decide whether to approve the request or not. If the request is approved, MC production managers prepare the production request by using patterns that correspond to the MC campaign. Patterns are defined at the beginning of each campaign and are reused for many requests, reducing the production request definition time and minimizing incidence of errors. MC request patterns are flexible and can be modified. For instance, during 2016 a new step was requested for all MC workflows, so default values were added to all patterns and applied to newly created requests.

3.2. Reprocessing workflow

The reprocessing workflow in ATLAS is used for processing real data with new software, conditions and calibrations. Reprocessing requests are defined by a few reprocessing coordinators. The reprocessing campaign usually includes several validation requests and one main request. The reprocessing workflow normally has a tree structure, where one step could be parent of many other steps. Based on this, requests could be defined using a special interface or by cloning an existing request as a pattern.

3.3. Derivation production workflow

Derivation production [3] is the last step of all workflows created centrally in ATLAS. Derivation production requests can be submitted either by production managers or by the physics groups directly. In order to provide faster and flexible procedures for defining reprocessing tasks, several alternate methods to create these requests were implemented: as ‘open ended’, as ‘train’, as a ‘child’ of other requests, or from scratch. Creating requests from scratch was used for all derivation tasks at the beginning, but was replaced by faster methods later, and are used only for creating new patterns now.

For open-ended request, the derivation coordinator defines containers and steps for each container. As soon as the dataset appears in the container, the Production System creates a step and defines a task from it. It is used to run derivation on real data produced from pre-processing [4]. Physicists heavily
use processed data from derivation, and the speed of task definition is important. Another added improvement is if the first task finishes successfully, similar tasks in the same request are defined without any scout jobs\(^1\): this speeds up completion of the task even more.

Another way to define production requests is the so-called ‘train’ approach. In this case, the derivation coordinator defines patterns for each campaign, which contains possible output types for each step and the parameters needed for the task definition. When the pattern is defined, the group contact person creates production request by choosing input datasets and corresponding outputs for them. If several groups select the same outputs, which could be run in the same task, the system merges them.

Creating the request as a child follows a similar workflow as ‘train’: requests are defined by group contact persons using existing pattern, but instead of using input dataset, steps are created as a child for tasks from MC or reprocessing campaign. This allows running derivation production tasks as fast as possible, because jobs could be defined as soon as enough input is ready, even before the parent task is fully completed. Production system follows parent tasks and in case of failure or redefinition by someone, the Production System automatically redefines the child task too.

### 3.4. Other workflows
The Production System is also used for High Level Trigger, MC validation and Tier 0 [4] spillover. High Level Trigger and MC validation have well defined workflows that are repeated for all requests with few parameter changes. For creating this type of production requests, a special interface has been implemented where the user just needs to select and fill requested fields.

Tier 0 is the first processing of real data normally performed on CERN computing resources. Sometimes these resources are not enough. Therefore the Production System was extended to run first processing tasks on the ATLAS grid with the same workflow that previously only ran on Tier 0.

### 4. Production System task definition
After a user creates the steps and submits a request, the Production System defines the tasks. During this stage several important actions take place. First, the Production System sets default parameters, e.g. memory and CPU limits are rarely defined by the user, and therefore set by DEFT. ATLAS has many default parameters for complex workflows: so a special interface to view and change them is available through the Production System. After parameters are set, the system checks for their consistency and compatibility. For instance, it checks that the input dataset has enough events or that defined parameters can be used for the chosen software release. At the same time it checks if similar tasks were already produced and, if so, it rejects the task because it could produce duplicate events. Searching for duplicated task is not easy, since a similar task could have been defined in subtly different ways, e.g. having a parent as a separate dataset or a separate container or some other task. During task definition, when the Production System detects that some events from the input dataset were already used, DEFT sets an offset and the new task uses or generates only non-duplicated events. In case a task could not be defined, DEFT will create a ticket with a detailed description and notify the manager of the task. If all checks are successfully passed, DEFT submits the task to JEDI.

### 5. Production System web UI
The Production System web UI has a modular structure, which allows reusing its components for different workflows. Hence the interface to submit or modify requests is the same for all described workflows (Figure 4).

\(^1\) Scout jobs are the first jobs of a task. They are used to measure job parameters such as memory consumption and processing time.
The web UI is used to monitor and manipulate tasks. It provides the following features:

- **Bookkeeping**: stores metadata, includes arbitrary hashtags, aggregates statistics that may be used for fine-tuning of running and historical tasks.
- **Monitoring**: user can easily follow the progress of running tasks.
- **Error Handling**: task could fail because of several permanent (e.g. bug in software) or temporary (e.g. storage is down) reasons. To quickly understand the root of the problem and to fix it by redefining the task is one of the major features of the Production System.
- **Chaining one production to another.**
- **Changing task parameters.**

### 6. Production System deployment

The Production system has a client-server architecture (Figure 5). On the client side it uses the Model-view-viewmodel (MVVM) approach that allows the creation of a modular, reusable interface addressing ATLAS requirements. It has two types of servers:

- **frontend servers** for generating web pages and to provide simple, interactive operations.
- **backend servers** for cron jobs and heavy operations.

The communication between frontend and backend is implemented using message queues.

ProdSys2 uses CERN Single-sign-on (SSO) for authentication and authorization [5]. It provides a familiar, standard login mechanism for most users, and allows users to set permission control on the application side.
7. Conclusion

ATLAS is a big experiment with a huge number of workflows. It is a challenging task to create a system that satisfies all the current requirements and is flexible to future requirements. The design decisions that were used in ProdSys2 data model, interface and deployment, have proven to be appropriate. In average, more than 2000 tasks were defined every day in 2015 and 2016 by hundreds of users. Future work will focus on the implementation of new workflows and improving the system usability.

8. Acknowledgment

We wish to thank all our colleagues who contributed to the ATLAS data processing and Monte-Carlo simulation activities. This work was funded in part by the U.S. Department of Energy, Office of Science, High Energy Physics under Contracts No. DE-SC0012704, No. DE-AC02-98CH10886 and Contract No. DE-AC02-06CH11357. NRC KI team work was funded by the Russian Ministry of Science and Education under Contract No 14.Z50.31.0024

References

[1] De K et al. 2015 The future of PanDA in ATLAS distributed computing J. Phys. Conf. Ser. 664 062035
[2] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider J. Inst. 3 S08003
[3] Catmore J, Cranshaw J, Gillam T, Gramstad E, Laycock P, Ozturk N and Stewart G A 2015 J. Phys. Conf. Ser. 664 072007
[4] Nairz, A et al. 2010 ATLAS Operations: Experience and Evolution in the Data Taking Era J.Phys Conf. Ser. 331 072034
[5] SSO at CERN - https://sso-management.web.cern.ch