Proceedings of the 9th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2021), Dubna, Russia, July 5-9, 2021

DEVELOPMENT OF THE ATLAS EVENT PICKING SERVER

E.I. Alexandrov [1,a], I.N. Alexandrov[1], D. Barberis[2], F.V. Prokoshin[1], A.V. Yakovlev[1] on behalf of the ATLAS Software and Computing Activity

[1] *Joint Institute for Nuclear Research, Joliot-Curie 6, RU-141980 Dubna, Russia*

[2] *Università di Genova and INFN, Via Dodecaneso 33, I-16146 Genova, Italy188300*

E-mail: [a] aleksand@jinr.ru

During LHC Run 2, the ATLAS experiment collected almost 20 billion real data events and produced about three times more simulated events. During physics analysis it is often necessary to retrieve one or more events to inspect their properties in detail and check their reconstruction parameters. Occasionally it is also necessary to select larger samples of events in RAW format to reconstruct them with enhanced code. The new Event Picking Server automates the procedure of finding the location of the events using the EventIndex and submitting the Grid jobs to retrieve the individual events from the files in which they are stored.

Keywords: Scientific computing, BigData, EventIndex

Evgeny Alexandrov, Igor Alexandrov, Dario Barberis, Fedor Prokoshin, Aleksandr Yakovlev

ATL-SOFT-PROC-2021-022
14 September 2021

1. Introduction

The ATLAS experiment is one of four LHC accelerator experiments at CERN [1]. The ATLAS EventIndex [2], the catalogue of all ATLAS recorded and simulated events, has been working since 2015 and improvements are produced permanently in correspondence with the increasing data flow. The main use case for the ATLAS EventIndex is event picking, i.e. finding one or more events in the many billion events generated by the ATLAS experiment, stored in several million files, and extracting them. One of the latest addition to the EventIndex project infrastructure is the development of a new Event Picking Service, in order to automate the process of event picking for cases when the number of requested data is large.

Several physics analysis workflows can use massive event picking to select a set of interesting events from the wealth of ATLAS data and reprocess them with enhanced algorithms or save additional variables that can help downstream analysis steps. One example in 2019 was the "γγ→WW" analysis. This analysis required the extraction of 50k events in RAW format out of the 18 billion events in Run 2 (about 10 million files). All the steps to look up the events in the EventIndex, submit the PanDA [3] event picking jobs, monitor them and retry them when timing out (because of long tape staging delays) were executed manually. A second round of that analysis is now requiring 136k events.

The main goal of the Event Picking Service is to perform all these actions automatically. A user would only have to supply to this service all relevant information to find the requested events such as a file containing the run and event numbers, data format of requested data, project name, trigger stream in case raw data are requested and the version of the requested events if other than raw data are in the request.

2. Architecture of the Event Picking Service

After analyzing the current request and taking into account the possibility of other similar requests the following requirements to the Event Picking Service have been formulated:

- The service consists of two parts: the Web Server and the Daemon, which are independent and communicate using a database
- Only the Web Server communicates with Client
- Only the Daemon communicates with protected systems (like Panda and Rucio [4])
- The Daemon has a flexible workflow. The workflow can be modified only using the database
- The results of all steps of the workflow are stored in the database
- The input data to any step is the output from the previous one, input to the first step is the request by the client

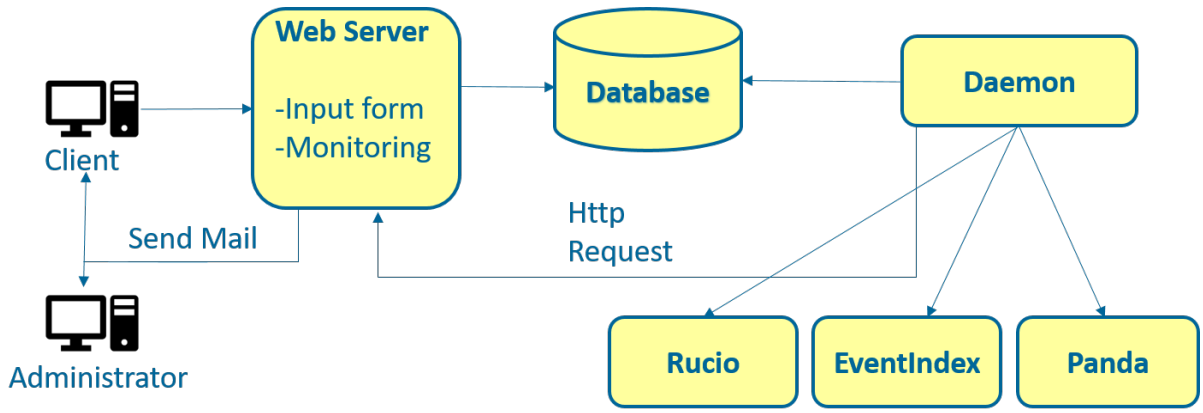

Figure 1. The architecture of the Event Picking Service

The architecture of the Event Picking service is present in Figure 1. A client generates a request to the server using a form which is built in an HTML page. The client receives an identifier for their request which can be used to get the details of that request. The Web Server sends the new request to the database. The Daemon monitors the database for a new request. When it detects a new request, it starts performing tasks for that request according to the workflow taken from the database. Input and output data, logs and results of all tasks are placed in the database. All this data can be viewed by a user or an administrator using a web server. After all tasks corresponding to the workflow are completed or critical errors are received during their execution, the Daemon sends an HTTP request to the web server. In response to this request, the web server receives all the necessary information upon request from the database and generates a message to the user (or administrator) about the results of the request.

3. Workflow

The Event Picking Service is not intended for solving one specific task, but for a whole type of event picking. In this case, the steps needed for different requests may be different. As practice shows, even during the execution of one request, it becomes necessary to make changes to the workflow. As a result, it is important to develop the dynamic ability to change the workflow.

After analyzing typical requests, two types of tasks were identified. The first type are tasks that use full input data. Tasks of this type must be performed sequentially. The second type are ones that only use a portion of the input. These tasks can be performed in parallel. Usually, whole chains of sequential tasks are parallel, rather than individual tasks. Within this chain, the task uses all the data from the previous task, but the first task in this chain uses only a portion of the data from the parent task. In the Event Picking Service, the first type of task is called *Job*. Chains of sequential tasks that can be run in parallel are called *Chain*.

The Event Picking Service has two types of workflow: *Job* workflow and *Chain* workflow. Both workflows are stored in database in separate tables. The basic idea behind these workflows is that they have an input state, an output state, and the name of the method that will execute the task. The service searches the workflow using the input state for the selected task. Using the workflow, it dynamically starts the required code and upon completion changes the status to the output state of the workflow.

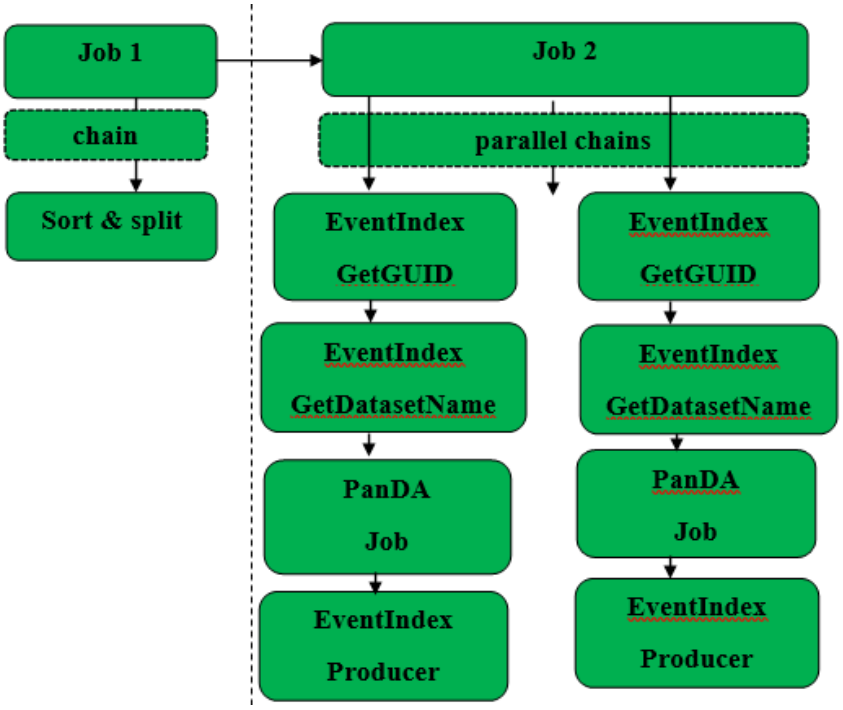

Figure 2. The workflow of Event Picking for “γγ→WW” analysis

Figure 2 shows the event collection workflow for the "γγ → WW" analysis. The first job sorts the entire input pair of run-event numbers. The second task splits it by different values of runs and starts all the split data in parallel mode. For each run, the following tasks are performed: get the GUID from the runs and event number, get the dataset template using the GUID, start copying the event using Panda, check the output files to make sure all events have been copied.

When using the service at stage 2, for some file datasets, it was necessary to manually intervene in the service to resolve errors. Most of these errors can be resolved automatically after upgrade of the service and we intend to do this at the next steps of updating the service.

4. Database

As was mentioned earlier, the Event Picking Service has two main components: Web Server and Daemon. The interaction between these components is carried out using the PostgreSQL database [5]. In addition, the database stores input and output data, the states for all request, jobs, chains or tasks, logs and all workflow. Figure 3 shows the structure of the database for the Event Picking Service. All the main elements of the workflow have their tables in the database for storing the current state, a link for input / output data and so on. Input and output data are stored in separate tables. Logs are stored only for real tasks.

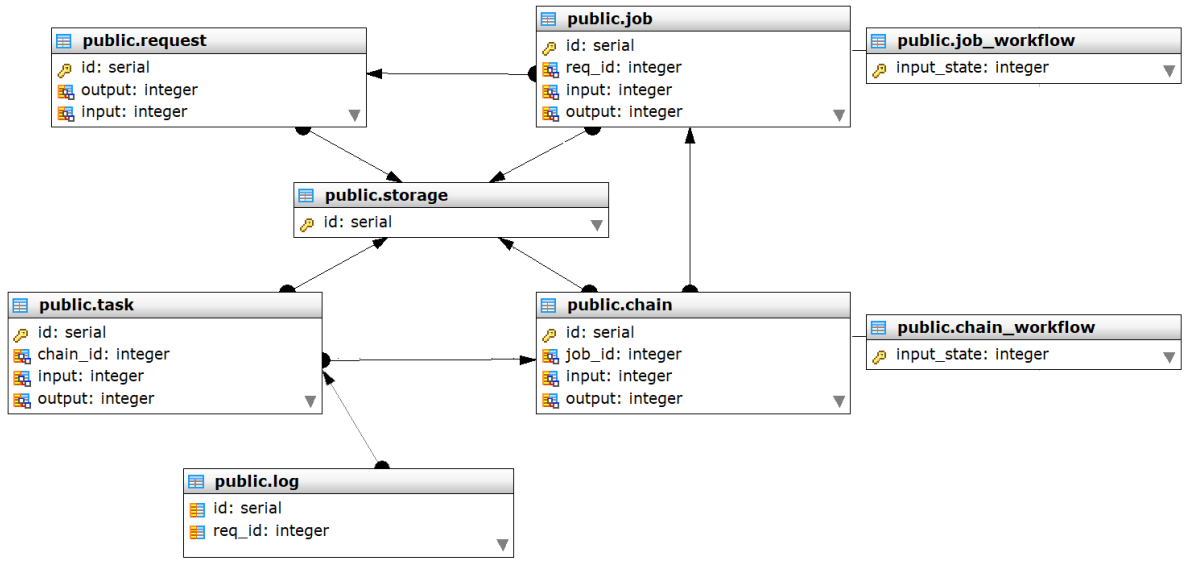

Figure 3. The structure of database for the Event Picking service

5. Daemon

The Daemon is responsible for performing all actions not directly related to the user. This requires access to ATLAS protected systems like Panda, Rucio and EventIndex. The Daemon of the Event Picking Service is written as a Java application and must run continuously. All new data or commands are retrieved from the database.

The structure of the Daemon is shown in Figure 4. The main class is the Root controller which monitors new requests and creates first job of any new request. The job controller keeps track of new jobs and finds a workflow for any new job and starts executing it. The chain controller finds the chain workflow and starts the execution of the task corresponding to that workflow.

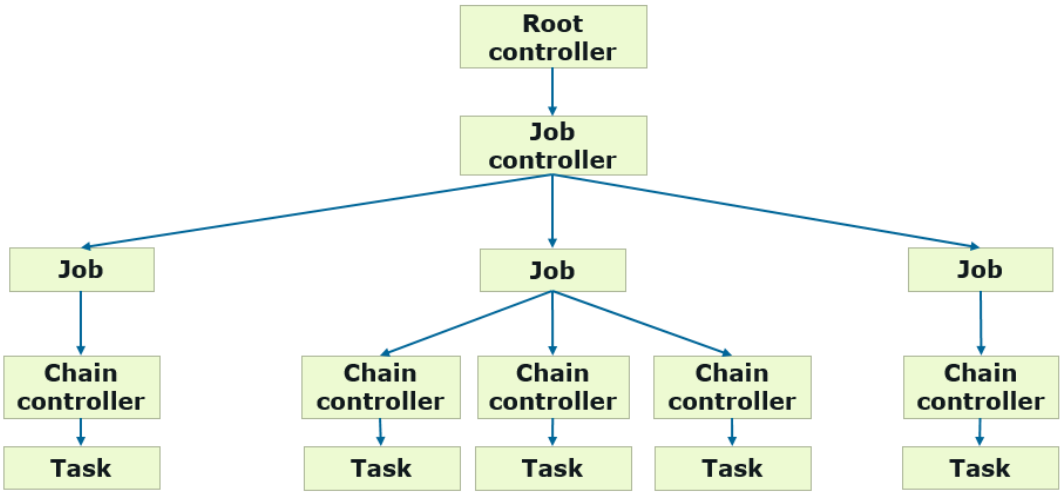

Figure 4. The structure of the Event Picking Daemon

6. Web Server

The web server is implemented on Apache Tomcat and is based on the Web Application Lego Toolkit [6]. It does not need to be installed on the same machine as the Daemon. Its only requirement is database access.

The Web Server has two forms, one for creating a new request and one for monitoring. To create a new request, the user must specify several parameters and upload the file with run-event number pairs. In the monitoring page the user can see the list of all requests with the states of the results. The user can see the details of selected request as well (Figure 5). The details include the result dataset name (in case the request was completed correctly) and information for each job. If the user selects any job he will see the information about the chains and tasks of these chains. This page has also error logs for the tasks.

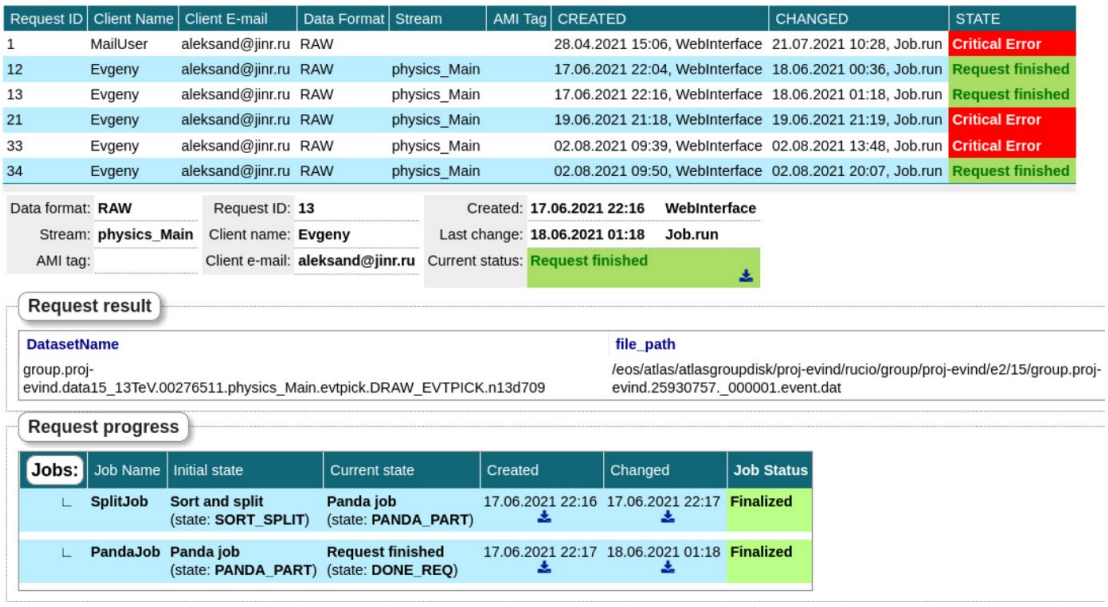

Figure 5. Monitoring of requests for the Event Picking Service

6. Conclusions

A prototype of the event picking service has been developed. The service was used for the second stage of the "γγ→WW" analysis (136k events). The experience of using the service in this stage showed that some errors required manual intervention. The Event Picking service will be modernized to fix such problems.

References

[1] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 S08003 doi:10.1088/1748-0221/3/08/S08003

[2] Barberis D et al 2015 The ATLAS EventIndex: architecture, design choices, deployment and first operation experience, J. Phys.: Conf. Ser. 664 042003, doi:10.1088/1742-6596/664/4/042003

[3] Barreiro Megino F et al 2017 ATLAS WORLD-cloud and networking in PanDA, J. Phys.: Conf. Ser. 898 052011, doi: 10.1051/epjconf/201921403025

[4] Barisits, M. et al. 2019 Rucio: Scientific Data Management, Comput Softw Big Sci, doi: 10.1007/s41781-019-0026-3

[5] PostgreSQL: https://www.postgresql.org/

[6] Sokolov I et al 2021 WALT Platform for Web Application Development, these proceedings