Getting the Most from Distributed Resources With an Analytics Platform for ATLAS Computing Services

Ilija Vukotic[1]
University of Chicago
5620 S Ellis Ave Chicago IL 60637, USA
E-mail: `ivukotic@uchicago.edu`

Robert W. Gardner
University of Chicago
5735 South Ellis Avenue, Chicago, IL 60637, USA
E-mail: `rwg@uchicago.edu`

Lincoln A. Bryant
University of Chicago
5735 South Ellis Avenue, Chicago, IL 60637, USA
`E-mail: lincolnb@uchicago.edu`

To meet a sharply increasing demand for computing resources for LHC Run 2, ATLAS distributed computing systems reach far and wide to gather CPU resources and storage capacity to execute an evolving ecosystem of production and analysis workflow tools. Indeed more than a hundred computing sites from the Worldwide LHC Computing Grid, plus many "opportunistic" facilities at HPC centers, universities, national laboratories, and public clouds, combine to meet these requirements. These resources have characteristics (such as local queuing availability, proximity to data sources and target destinations, network latency and bandwidth capacity, etc.) affecting the overall processing efficiency and throughput. To quantitatively understand and in some instances predict behavior, we have developed a platform to aggregate, index (for user queries), and analyze the more important information streams affecting performance. These data streams come from the ATLAS production system (PanDA), the distributed data management system (Rucio), the network (throughput and latency measurements, aggregate link traffic), and from the computing facilities themselves. The platform brings new capabilities to the management of the overall system, including warehousing information, an interface to execute arbitrary data mining and machine learning algorithms over aggregated datasets, a platform to test usage scenarios, and a portal for user-designed analytics dashboards.

38th International Conference on High Energy Physics
3-10 August 2016
Chicago, USA

[1]Speaker, on behalf of the ATLAS collaboration

1. Introduction

The ATLAS experiment at the CERN LHC [1] uses a Distributed Computing (ADC) system composed of a number of different services distributed over more than one hundred institutions. Most of the ADC monitoring data are collected and stored in the ATLAS Oracle database at CERN. During LHC Run 1, ADC Monitoring group provided a number of custom made dashboards that enabled us to get high level information. These included: data management, sites, distributed data management (DDM) & storage accounting, data processing, data accounting, databases, etc. and were mainly used by the ADC shifters. Each time a user needed a new plot or other change in the dashboards, it took at least few days and involved at least two more people. For the LHC Run 2 the ADC monitoring group has a number of new requirements that the old system could not fulfill. On top of the already covered requirement of quickly giving a high level, global picture of the system state, we need to understand the interplay of different systems and services. These requirements demand all of the data to be in one place and the system be able to quickly cross-reference different data sources. To be able to debug systems we need a way to easily drill down to the most detailed information available. To do so the old system requires authorization and direct Oracle database access. Improvements in ADC heavily depend on doing complex analysis or testing different models on the historical data, so a programmatic access to all the data is an important requirement for the new service. Finally we need to be able to continuously or periodically analyze both raw and derived information in order to provide near-real time alarm and alert services.

2.Data sources

The data collected in the new system can be divided in two categories: data originating from CERN and the data coming from outside. The former include PanDA [2] data coming from the Oracle database, and Rucio [3] services data coming from logs of its servers. The latter include world-wide distributed sources like PerfSONAR [4] network monitoring service, federated data access service FAX [5] monitoring, and even individual users' analysis programs (input data usage patterns). A brief description of several of the largest services is included here:

- **PanDA** - all grid jobs and tasks are described by 96 and 45 metadata fields, respectively. These are crucial to accounting, job performance analysis, brokering decisions analysis, alerting and error reporting, that can be done per site, cloud, user, activity, etc. The PanDA data amount to more than 5GB per day in more than 1.3 million records.
- **Rucio** - all of the ATLAS Distributed Data Management (DDM) actions are performed and logged by Rucio servers. All data transfers using the reliable file transfer service FTS [6], client interactions, API calls, system states and errors can be extracted, monitored and analyzed from the logs. Long term storage of these data is essential to optimizing the current data distribution and modeling new data placement algorithms. Around 15 million (~15GB) or records generated per day should be kept for the long term. On top of that, around 80 million (~65GB) records of debugging or short-term data are kept for one month.

- **PerfSONAR** - the widely-deployed test and measurement infrastructure that is used by science networks and facilities around the world to monitor and ensure network performance will in addition be used in both job and data brokering decisions. The four most important metrics from ~350 sites generate around 4 million (2.4 GB) records per day.

3.Architecture of the analytics system

The data flow in the new ATLAS analytics system can be seen in Figure 1. The two central components are: a large Hadoop cluster at CERN and two Elasticsearch (ES) clusters - one at the University of Chicago and one at CERN. While most of the data exist in both Hadoop and ES, it is not required. The data coming from the ATLAS Oracle database is imported directly using Apache Sqoop into the Hadoop cluster. From there data are periodically cleaned up (obsolete variables removed, strings variables parsed), enriched (with info from other sources) and sent to both Elasticsearch clusters using Apache Pig scripts. Rucio log data are imported using Logstash. The data coming from outside CERN get imported using Apache Flume or custom made collectors subscribed to the ActiveMQ queues. All of the scripts mentioned above run at a small dedicated cluster of 5 virtual machines. In order to test performance and feasibility of doing analytics in the cloud, part of the data is shipped in Apache Avro format to the Google object store and from there imported into the Google BigQuery.

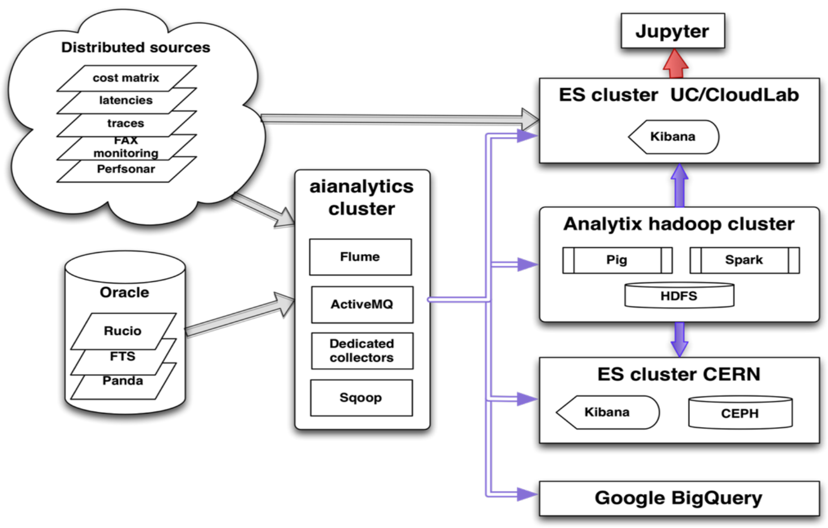

Figure 1. Simplified schema of the ATLAS ADC Analytics System.

3.1 Hadoop

A large 25-node Hadoop cluster at CERN serves as our primary data store and runs periodic MapReduce jobs doing data cleaning, reshuffling (eg. combining all the information related to one Panda task in one record), and ES indexing. Certain types of studies are also only possible in this environment. However, due to slow turn-around times (because of network attached storage) and complexities of Pig scripting, this is sparsely used. To that end, the

Apache Spark processing engine has also been installed as it promises very fast turn-around times and a simple Python API.

3.2 Elasticsearch

While we have an Elasticsearch cluster at CERN, the one currently used in production is at the University of Chicago. It consists of eight nodes in total: five data nodes, each with four 800 GB solid state disks, as well as three master nodes that also provide client access and indexing. This configuration proved capable of simultaneously supporting 10 kHz indexing and 1 kHz of simple queries. For a graphical user interface, we use Elastic Kibana for easy data exploration, visualization, and dashboard creation. Some examples of Kibana dashboards can be seen in Figure 2.

3.3 Jupyter

To make it faster and more convenient to do complex data analysis tasks on data stored in UC Elasticsearch, we provide a dedicated Jupyter service on a powerful node in the same local network. This node is additionally equipped with two NVidia Tesla K20c GPU cards. Most popular scientific computing and machine learning Python packages are installed (NumPy, scikit, PANDAS, SciPy, matplotlib, Theano, Keras). We opted for a simple, unauthenticated access Jupyter instance in order to simplify code reuse, and make it easy for beginners to learn from others. Currently there are ~ 20 active users (mainly managers of different ADC systems).

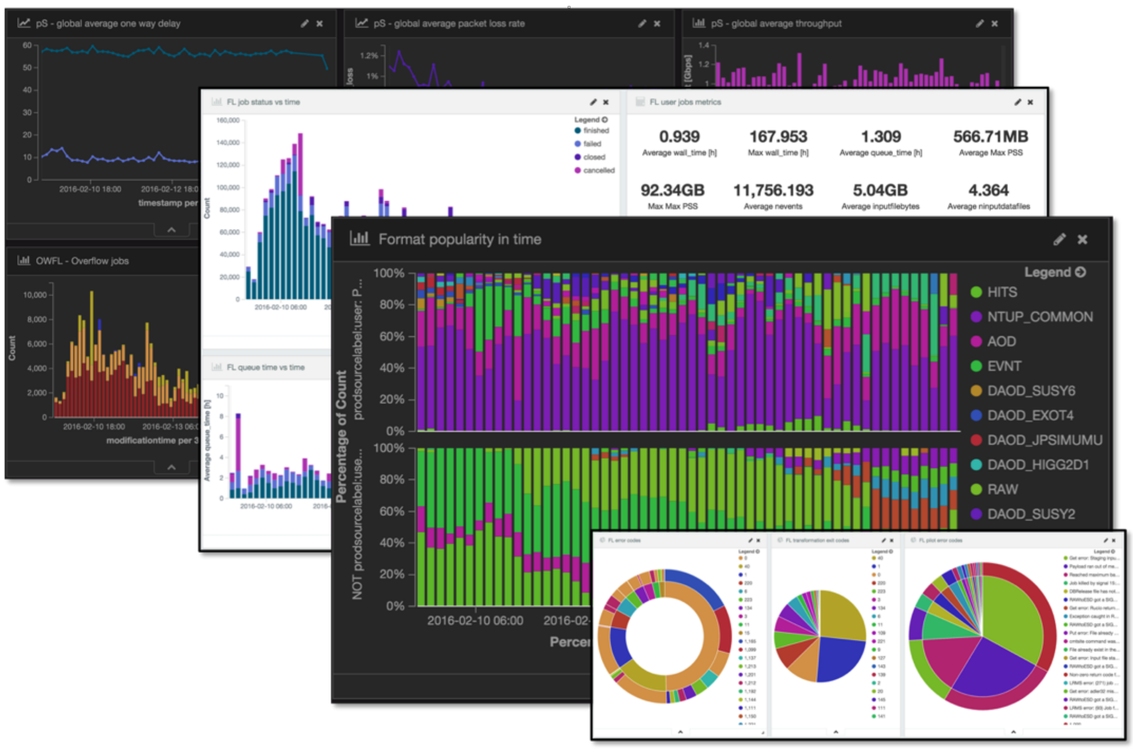
pS - global average one way delay
pS - global average packet loss rate
pS - global average throughput
FL job status vs time
FL user jobs metrics
0.939 Average wall_time [h]
167.953 Max wall_time [h]
1.309 Average queue_time [h]
566.71MB Average Max PSS
92.34GB Max Max PSS
11,756.193 Average nevents
5.04GB Average inputfilebytes
4.364 Average ninputdatafiles
OWFL - Overflow jobs
FL queue time vs time
Format popularity in time
Legend
HITS
NTUP_COMMON
AOD
EVNT
DAOD_SUSY6
DAOD_EXOT4
DAOD_JPSIMUMU
DAOD_HIGG2D1
RAW
DAOD_SUSY2
Percentage of Count
2016-02-10 06:00

Figure 2. A few examples of Kibana dashboards used for monitoring and reporting.

4. Examples of analytics tasks

A large number of simple, mostly accounting tasks has been done using Kibana (eg. per user resource consumption). More complex investigations include: understanding performance of different CPU types in different ATLAS data processing tasks, optimization of ATLAS event data files content by analyzing variables accessed by users, modeling proxy cache [6] behaviour with different combinations of caching models, sites, and data accesses, understanding timing of different processing steps, memory usage, and CPU efficiencies at different sites, generation of alarms and alerts upon network anomaly detection, optimizations of data and job brokering decisions (see Fig. 3), neural network based detection of high network utilization, etc.

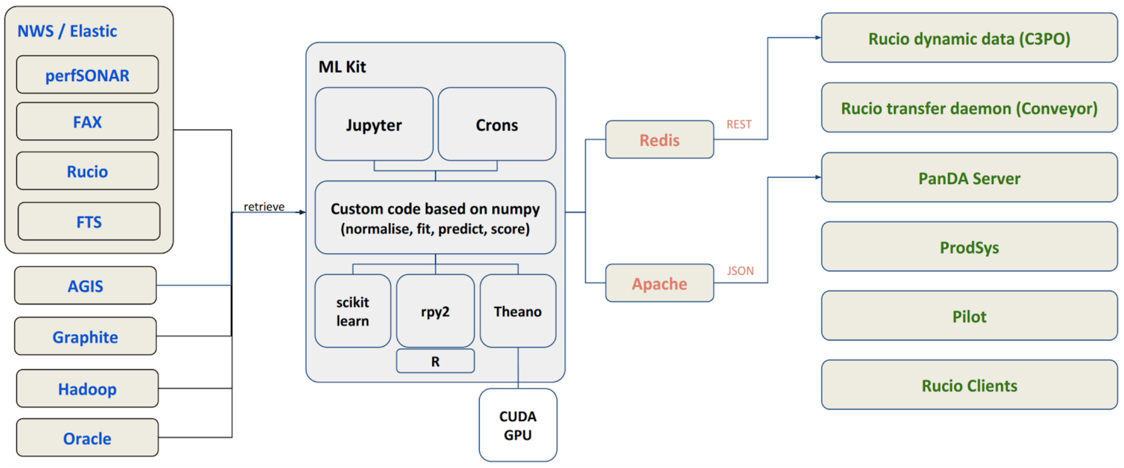

Figure 3. Machine learning method trained on historical data could be used to optimize data and job brokering using near real-time analytics data. Here one potential scheme of such a system is shown.

5. Conclusions

The new stack of Big Data tools (Hadoop, Elasticsearch, Flume, Sqoop, Logstash, Kibana, Jupyter) provides an excellent platform for the ATLAS Distributed Computing analytics needs. It is horizontally scalable (as both Hadoop and Elasticsearch are), very performant and easy to use and develop for. It provides simplicity and freedom in creating custom dashboards and specialized GUIs. Additionally it provides a backend for the complex analytics tasks and can replace other full fledged services for alerting, reporting, etc.

References

[1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider* , JINST 3 13617 (2008) S08003

[2] T. Maeno, *PanDA: distributed production and distributed analysis system for ATLAS*, *Journal of Physics: Conference Series* **0119** (062036) 006 [iopscience.iop.org/1742-6596/119/6/062036]

[3] Serfon, C, et al., *Rucio, the next-generation Data Management system in ATLAS*, Nucl. Part. Phys. Proc. **273-275** (969-975) [10.1016/j.nuclphysbps.2015.09.151]

[4] Hanemann A, et al., *PerfSONAR: A Service-Oriented Architecture for Multi-Domain Network Monitoring*, International Conference on Service Oriented Computing, Amsterdam, The Netherlands [10.1007/11596141_19]

[5] Gardner, R. et al., Data federation strategies for ATLAS using XRootD, 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2013), Amsterdam, The Netherlands [10.1088/1742-6596/513/4/042049]

[6] Aylon A. A, et al., *FTS3: New Data Movement Service For WLCG*, 2014 J. Phys.: Conf. Ser. 513 032081, [iopscience.iop.org/1742-6596/513/3/032081]

[7] *XRootd, disk-based, caching proxy for optimization of data access, data placement and data replication*, J. Phys. Conf. Ser. **513**, 042044 (2014)