Predictive analytics tools to adjust and monitor performance metrics for the ATLAS Production System

F H Barreiro Megino[1], M Borodin[2], D Golubkov[3,4], M Grigorieva[4,5], M Gubin[5], A Klimentov[4,6], T Korchuganova[5], T Maeno[6], S Padolski[6], M Titov[4] on behalf of the ATLAS Collaboration

1 University of Texas at Arlington (US)
2 University of Iowa (US)
3 Institute for High Energy Physics (RU)
4 National Research Centre Kurchatov Institute (RU)
5 National Research Tomsk Polytechnic University (RU)
6 Brookhaven National Laboratory (US)

E-mail: maksim.gubin@cern.ch

Abstract. Having information such as an estimation of the processing time or possibility of system outage (abnormal behaviour) helps to assist to monitor system performance and to predict its next state. The current cyber-infrastructure of the ATLAS Production System presents computing conditions in which contention for resources among high-priority data analyses happens routinely, that might lead to significant workload and data handling interruptions. The lack of the possibility to monitor and to predict the behaviour of the analysis process (its duration) and system's state itself provides motivation for a focus on design of the built-in situational awareness analytic tools.

ATL-SOFT-PROC-2017-059
20 October 2017

1. Introduction

The ATLAS [1] Production System (ProdSys) [2] is an automated scheduling system that is responsible for central production of Monte-Carlo data, highly specialized production for physics groups, as well as data pre-processing and analysis using such facilities as grid infrastructures, clouds and supercomputers. Its current processing rate is around 2M tasks per year (that corresponds to 365M jobs per year). ProdSys evolves to accommodate a growing number of users and new requirements from the ATLAS Collaboration (the ATLAS experiment at the Large Hadron Collider, CERN), physics groups and individual users. ATLAS Distributed Computing in its current state is the aggregation of large and heterogeneous facilities, running on the WLCG (Worldwide LHC Computing Grid) [3], academic and commercial clouds, and supercomputers.

There are two key components of ProdSys that represent the core of the system: i) Database Engine for Tasks (DEfT) is used to formulate the tasks, chains of tasks and task groups (i.e., production request), complete with all necessary parameters; ii) Job Execution and Definition Interface (JEDI) is a subsystem (also part of the workload management system PanDA [4]) that performs task-level workload management (i.e., brokerage and execution), dynamic job definition and execution (for resource usage optimization) [5].

Having information such as an estimation of the processing time or possibility of system outage (abnormal behavior) helps to monitor system performance and to predict its next state. The current cyber-infrastructure of ProdSys presents computing conditions in which contention for resources among high-priority data analyses happens routinely, that might lead to significant workload and data handling interruptions. The lack of the possibility to monitor and to predict the behavior of the analysis process (its duration) and system's state itself caused us to focus on design of the built-in situational awareness analytic tools.

The proposed suite of tools aims to estimate completion time (so-called "Time To Complete", TTC) for every (production) task, completion time for a chain of tasks, and to predict the failure state of the system based on "abnormal" task processing times. Its implementation is based on Machine Learning methods and techniques, and besides the historical information about finished tasks it uses ProdSys job execution information and the state of resource utilization (real-time parameters and metrics). The next planned step after task duration prediction is task profiling, detecting anomalous task behavior, and predicting task chain execution duration.

2. Approaches to estimating completion time for tasks

Several approaches were considered for predicting task execution times (i.e., task execution durations) on the basis of simulation modeling, and on the basis of machine learning.

Within the simulation modeling approach, tools for modeling distributed computing systems such as SimGrid, CloudSim, SPECI, CDOSIM and DCSim were considered for use. [6, 7] Based on the provided functionality, the SimGrid (a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments) was chosen for further consideration.

Within the framework of the machine learning approach, modeling based on deep neural networks and ensemble methods, such as random forests and gradient-boosted trees, were considered. After performing the preliminary analysis of the approaches, we decided to implement the methods in the following order: apply gradient-boosted trees and random forest regression methods for task duration prediction (use minimal set of parameters), refine predictions by developing a framework allowing to concurrently use several different predictive models (that are built with different "extended" sets of parameters of used methods) and compare their precision, adding the deep learning-based model to the set of models, and choosing the ones that give better precision while keeping acceptable performance. [8]

The decisive factor in choosing the machine learning-based approach over the simulation-based approach was the rate of growth and change of the WLCG and rapid increase of its computation load, making simulations impractical, as the scale and complexity of ProdSys exceeds the capabilities of the best grid modeling systems currently available. [2, 9]

A machine learning-based approach, unlike a simulation-based one, uses historical data to describe functioning of the system instead of its exact structure description for model training, and therefore allows for gradual increase of prediction precision with increase of the size of data sample, in turn allowing the researcher to choose a desired trade-off between model scale and precision, while a simulation-based approach features sharp precision drop when the used model is incomplete or outdated, necessitating using a full scale model of the system.

3. Framework for prediction of the task execution duration

The imprecise nature of models used to predict task durations, as well as the need to provide users with task duration estimates for all steps of the task lifecycle, preferably starting before the task is completely defined, to simplify the planning of data processing, necessitated creating a task duration prediction framework that allows using several models simultaneously, monitoring their precision and performance, adding and removing new models without impact on the framework

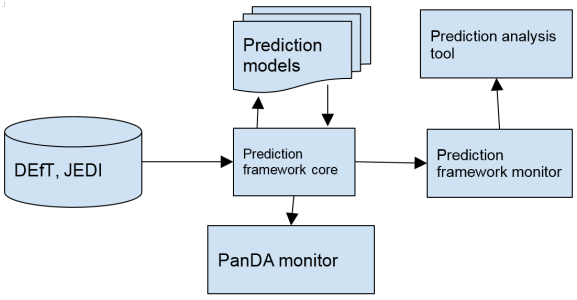

Figure 1. Task duration prediction framework structure

functionality, updating models according to a predefined schedule, and gathering statistics to be used in further improving of the framework.

3.1. Task duration prediction framework components

The structure of the task duration prediction framework is displayed on Figure 1.

The *prediction framework core component* provides the system's core functionality. It maintains a connection to a set of prediction models (by the following blocks: collector and predictor) and allows to operate with new models (e.g., extend the set with newcomers) and control their status in the system, serves as a source of prediction data for PanDA monitor, as well as generates and sends logs of its functioning to the prediction framework monitor. Input data for model creation is collected from ProdSys components (i.e., DEfT/JEDI).

Prediction framework monitor accumulates the logs provided by the prediction framework core, and allows operators to review logs, as well as to configure conditional alerts to be sent out in case of certain conditions being met.

Prediction analysis tool implements several metrics that can be used to control the stability, performance and precision of various models based on the logs gathered by the prediction framework monitor, predictions made by the models, and factual task durations.

Generating a prediction involves the following steps: the framework core component collects data with task parameters from the DEfT/JEDI database, converts them into a format accepted by the models, and then sends it to the prediction models using the task duration predictor API. After receiving predictions, the prediction framework core sends the results to the PanDA monitor to be displayed. Every step of the process (e.g., receiving a request for prediction generation, polling the models, the model response times and predicted values) is stored using the prediction framework monitor.

Results of the prediction process conducted by the framework are displayed by the PanDA Monitor [10] (that shows the profile of the corresponding task with produced predictions).

3.2. Prediction models

The requirement of providing users with an estimate of task execution duration for every step of task lifecycle forces use of a tiered approach to task duration prediction that means using a certain type of the model for the particular stage of the task lifecycle.

3.2.1. Preliminary prediction. It is often useful for the data processing planning to have a rough estimation of data processing duration even during the preliminary planning stage, when the user does not yet know how much resources can be allocated to the task, when the task will be scheduled for for execution, and the exact size of the input data. Initial "rough" estimation of the task execution duration is produced by the classifier component, that returns an average duration of tasks of the same class as the requested one (based on historical data). The class of the task is defined by the following properties: *project name*, *production step*, and *provenance*.

3.2.2. "Cold" model. Models of the "cold" category use numerous task parameters for their creation. Those parameters are known at the stage of task definition, such as type of the input data, size of the input data (expressed as the number of processing jobs), type of processing to be performed on the input data, and type of the processing queue that will be used by the task. Limiting the model to the parameters that are known at the time of task definition allows the model to give prediction at the early stage of task lifecycle, but it also means that its prediction will lack the precision, as the model does not account for the state of processing environment.

3.2.3. "Warm" model. After the execution of the task begins, it launches up to 10 "scout" jobs to measure several important performance metrics, such as amount of RAM an average job uses, and an average job duration, that allows to better gauge the load that the task will affect on the WLCG, as well as the state of WLCG itself. The knowledge of these metrics allows for significant improvement of task duration prediction precision. "Warm" model category is based on utilization of these metrics. Models of this category make use of all parameters available to the previously defined category, as well as metrics provided by running scout jobs.

3.2.4. "Hot" model. Models of the "hot" category are aimed aimed at making predictions of the remaining runtime for a task that is being processed. These models are created on (and can make use of) every metric available to models of "cold" and "warm" categories, as well as parameters of task jobs, such as average job duration, trends in job duration, and occurrence of failed jobs, to predict task duration with the highest precision.

3.3. Implemented and planned model types

The current implementation of the preliminary prediction model is based on a simple classifier. It is planned to switch over to a machine learning based model if the classifier's precision proves insufficient for practical use. The "cold", "warm" and "hot" models currently in use are based on the Gradient-Boosted Trees regression method from Spark MLlib machine learning library. [11]

During the next step of the framework development, random forest and deep learning-based models will be added to the models pool for performance and precision evaluation, followed by selecting the most precise models with acceptable performance for use in predicting task durations.

3.4. Planned extensions of task duration prediction framework

The preliminary research has been performed to gauge feasibility of detecting anomalous behaviour of running tasks. We have found that tasks of the same type tend to have similar patterns of job execution. The execution profile of a typical task is shown in Figure 2. The red line shows when the jobs comprising the task finish, the green dots show the start times of the corresponding jobs. By viewing the task profile it is often possible to notice anomalies visually long before a task becomes overdue.

Another promising direction of the framework development is task chain duration prediction. Due to the way tasks are implemented in ProdSys, the next task in the chain can often starts execution before the previous task finishes. This leads to the fact that duration of the task chain is shorter than the sum of tasks' durations, and preliminary tests with Gradient-Boosted Trees-based models showed that predicting task chain duration with high precision is possible, conditional upon knowing task durations with high precision. As currently task duration prediction has low precision for tasks that are not currently running, the implementation of task chain duration prediction was delayed for now.

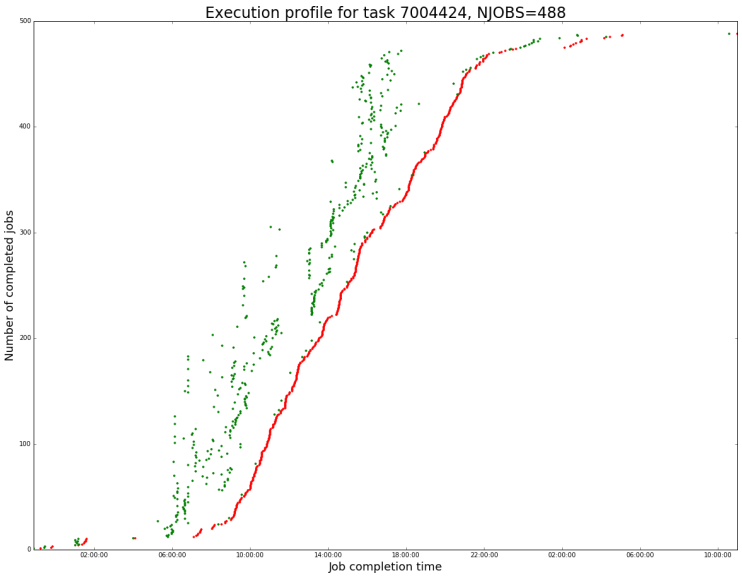

Figure 2. Execution profile of a typical task

4. Conclusion

Adding predictions of the task execution duration to the features of the ProdSys monitoring will allow users/operators to notice task execution delays as they happen, and take measures to resolve them. Combined with the anomaly detection framework, this will further improve efficiency of ProdSys data processing and reduce the data handling interruptions.

Acknowledgments

We wish to thank all our colleagues from ATLAS experiment, PanDA team and CERN IT.

TPU team research was funded from Government Task "SCIENCE" Grant Number 2.9223.2017/8.9

NRC KI work was funded by the the Russian Ministry of Science and Education under contract 14.Z50.31.0024.

References

[1] Aad G, Abat E, Abdallah J, Abdelalim A, Abdesselam A, Abdinov O, Abi B, Abolins M, Abramowicz H, Acerbi E *et al.* 2008 *Journal of Instrumentation* **3** S08003–S08003

[2] Borodin M, De K, Garcia J, Golubkov D, Klimentov A, Maeno T, Vaniachine A *et al.* 2015 *Journal of Physics: Conference Series* vol 664 (IOP Publishing) p 062005

[3] Bird I 2011 *Annual Review of Nuclear and Particle Science* **61** 99–118

[4] Maeno T, De K, Klimentov A, Nilsson P, Oleynik D, Panitkin S, Petrosyan A, Schovancova J, Vaniachine A, Wenaus T *et al.* 2014 *Journal of Physics: Conference Series* vol 513 (IOP Publishing) p 032062

[5] De K, Golubkov D, Klimentov A, Potekhin M, Vaniachine A *et al.* 2014 *Journal of Physics: Conference Series* vol 513 (IOP Publishing) p 032078

[6] Casanova H 2001 *Cluster computing and the grid, 2001. proceedings. first ieee/acm international symposium on* (IEEE) pp 430–437

[7] Malhotra R and Jain P 2013 *International Journal* **3**

[8] Caruana R and Niculescu-Mizil A 2006 *Proceedings of the 23rd international conference on Machine learning* (ACM) pp 161–168

[9] Fujiwara K and Casanova H 2007 *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools* (ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)) p 12

[10] Klimentov A, Nevski P, Potekhin M and Wenaus T 2011 *Journal of Physics: Conference Series* vol 331 (IOP Publishing) p 072058

[11] Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S *et al.* 2016 *The Journal of Machine Learning Research* **17** 1235–1241