Operational Intelligence for Distributed Computing Systems for Exascale Science

Alessandro Di Girolamo[1], *Federica* Legger[2], *Panos* Paparrigopoulos[1], *Alexei* Klimentov[6], *Jaroslava* Schovancová[1], *Valentin* Kuznetsov[3], *Mario* Lassnig[1], *Luca* Clissa[8,9], *Lorenzo* Rinaldi[8,9], *Mayank* Sharma[1], *Hamed* Bakhshiansohi[5], *Marian* Zvada[7], *Daniele* Bonacorsi[8,9], *Simone* Rossi Tisbeni[10], *Luca* Giommi[8,9], *Leticia* Decker de Sousa[8,9], *Tommaso* Diotalevi[8,9], *Maria* Grigorieva[4], and *Sergey* Padolski[6]

[1]CERN, Geneva, Switzerland
[2]INFN Turin, Italy
[3]Cornell University, USA
[4]Moscow State University, Moscow, Russia
[5]DESY
[6]Brookhaven National Laboratory (BNL), USA
[7]University of Nebraska-Lincoln, Lincoln, NE, USA
[8]University of Bologna, Bologna, Italy
[9]INFN Bologna, Italy
[10]INFN-CNAF Bologna, Italy

Abstract. In the near future, large scientific collaborations will face unprecedented computing challenges. Processing and storing exabyte datasets require a federated infrastructure of distributed computing resources. The current systems have proven to be mature and capable of meeting the experiment goals, by allowing timely delivery of scientific results. However, a substantial amount of interventions from software developers, shifters and operational teams is needed to efficiently manage such heterogeneous infrastructures. A wealth of operational data can be exploited to increase the level of automation in computing operations by using adequate techniques, such as machine learning (ML), tailored to solve specific problems. The Operational Intelligence project is a joint effort from various WLCG communities aimed at increasing the level of automation in computing operations. We discuss how state-of-the-art technologies can be used to build general solutions to common problems and to reduce the operational cost of the experiment computing infrastructure.

1 Introduction

The Operational Intelligence (OpInt) project starts as a joint effort of several HEP experiments. We currently focus on the operational challenges of the distributed computing infrastructures of the Worldwide LHC Computing Grid (WLCG)[1], and discuss possible technological solutions. Machine Learning (ML) models applied to the prediction of intelligent data placements and access patterns can help to increase the efficiency of resource exploitation and the overall throughput of the experiments distributed computing infrastructures. Time-series analyses may allow for the estimation of the time needed to complete certain tasks, such as processing a certain number of events or transferring a certain amount of data. Anomaly detection techniques can be employed to predict system failures, leading for example to network congestion. Every year thousands of tickets are submitted to ATLAS[2] and CMS[3] issue tracking systems, hence further processed by the experiment operators. Recording and analyzing shifter actions can be used to automate tasks such as submitting tickets to support centers, or to suggest possible solutions to repeating issues.

1.1 The WLCG computing infrastructure

The computing infrastructure of the LHC experiments is based on the WLCG, and relies on distributed facilities that continuously process LHC data, produce simulations of physics processes at the LHC, and provide data access and computational power to the physics community for analysis. In total, the LHC experiments rely on more than 400 PB of disk storage, 700 PB of tape storage, almost 1 million CPU cores, and both dedicated and shared network infrastructures. Such infrastructure is distributed across more than 200 computing centres worldwide under different administrative domains. These include off-the-shelf resources with commodity hardware, such as a typical university computing cluster, and a growing number of non standard resources, such as HPCs and opportunistic/volunteer resources, as well as commercial and scientific clouds.

1.2 Operational model

HEP computing systems are operating in a complex, heterogeneous, and dynamically changing environment. The operation and maintenance of the computing infrastructure must guarantee on the one hand constant and optimum utilization of the available resources, and on the other hand high data throughput according to the physics goals and priorities set by each experiment. Unstable behavior of a distributed computing environment can arise at various levels:

- *Application:* anomalous job execution time due to rare and unusual workflows, wrong configuration parameters, missing software dependencies, runtime exceptions, software defects;
- *Middleware:* authentication, authorization, data transfers, schedulers, computing elements, data flow policies;
- *Network:* data transfer rate between computing sites/elements;
- *Resource:* memory/storage/CPU issues, low packet I/O rates.

Many workload and data management routine operations are already automated. This includes job dispatching and scheduling, brokerage and re-brokerage, recovery of failed jobs affected by sporadic issues, fair-share based access to resources, replication of popular data, recovery of lost data, and deletion or archival of unused data. A variety of automatic tests are periodically executed to check the health of the infrastructure. Failures of critical elements can be automatically handled in some cases, for example by automatic restart of services, and faulty resources can be automatically excluded from the global infrastructures. However, most issues still need to be spotted manually by teams of operators, and some of them need the intervention of experts to be resolved. Typically this is handled by the experiments by having dedicated teams of operators and experts for the central workload and data management services, and local support for each distributed facility or groups of geographically close ones or belonging to the same administrative domain. Such activities may be organised in shifts to cover the various time zones.

Operators of the various teams are constantly checking several sources of aggregated information such as test results, system performance metrics, or system logs to spot possible issues and take appropriate actions. Often failures in one subsystem or facility can generate failures in a multitude of other systems, causing avalanche effects. Therefore operators need to be able to correlate the various sources of information to find the root cause of the issue. Once the cause is found, it may either be solved by the operators themselves following documented procedures, or escalated to the relevant experts. The escalation is currently being done either in the form of email, chat, meeting, or most frequently by using a ticketing system such as JIRA, SNOW, or GGUS.

While it is difficult to quantify how many Full Time Equivalents (FTE) are doing such tremendous efforts daily, WLCG experiments claim to have around 100 persons for a total of 50 FTEs involved at various levels in global computing operations. Thousands of tickets are filed and handled every year. Similarly, a case study from the telecom industry reports that complex customer issues may involve hundreds of engineers, and several days to identify a root cause and to apply network fixes [4].

2 The OpInt Framework

In general, there is a lack of a common tool capable of: *aggregating* various sources of monitoring data and alerts, *extracting* relevant information and producing useful metrics, *keeping* a history of past problems and solutions, *proposing* corrective actions based on past experiences, and *automate* corrective actions for recurrent issues. We propose to address the above issues by developing a smart platform with the following features:

- pipeline for aggregation of instrumentation data streams;
- data collection and storage of the aggregated data;
- intelligent algorithms able to spot anomalies in the incoming data streams;
- recommendation system to predict actions to be taken to cure arising issues;
- integrated view for human operators of the current metrics and alerts;
- collection of feedback from human operators on recommendation, to be used to improve future suggestions;
- knowledge base containing a history of the issues, suggestions, and resolution actions;
- ultimately automation of corrective actions without human intervention;
- provide feedback to upstream systems to improve error reporting.

We will evaluate and adapt to our needs the architecture shown in Figure 1. We foresee the following layers:

- *Data providers:* at the bottom of the stack, these are the various subsystems, components, services of the computing infrastructure;
- *Data sources:* this is a storage layer for monitoring and logging information produced by the data providers;
- *Processing:* this layer is responsible for all operations such as data cleaning, sanitation, anonymization, aggregation, necessary to prepare the data for further processing by the layers above;
- *Analytics:* this layer is responsible for extracting insights from the data, such as performance metrics, alerts, and may trigger further processing;

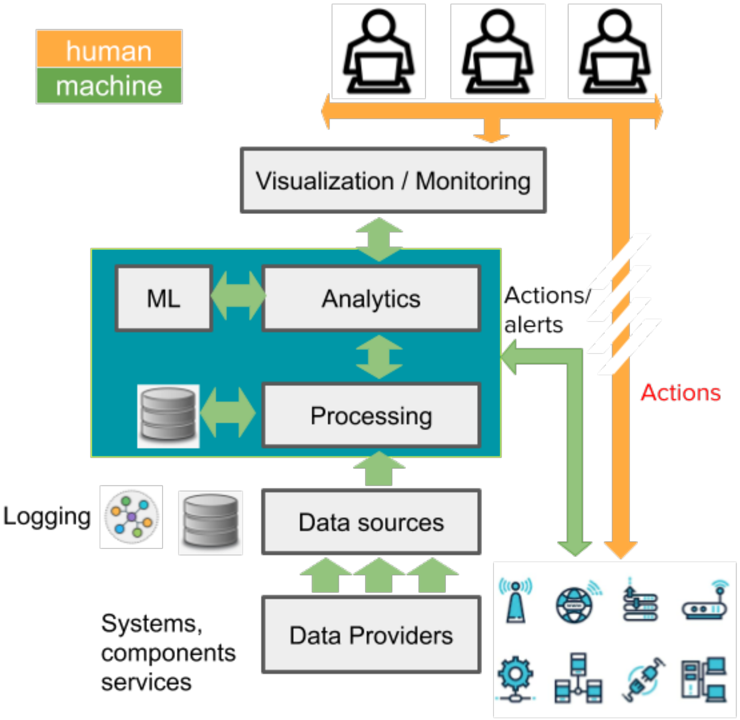

Figure 1. The proposed architecture of the Op Int Framework.

- *Visualization:* this is the user interface, which can be of the form of web interfaces, chat bots, emails, to expose the results to human operators, and collect their feedback on the suggestion of the framework.

Each individual layer of the data processing pipeline should be independent and encapsulated via containerized solutions. This technology can provide a high level of abstraction from underlying hardware resources and mitigates various maintenance issues related to the chosen technology stack and its inter-dependencies. Each layer should clearly define data formats and APIs which can take care of data transformation and delegation. Such choices should be defined based on the chosen technology middlewares, for example a JSON based schema can be a good candidate for data exchange between the Analytics and Visualization layers, while DataFrames or NumPy arrays may be good candidates for the data exchange between Processing and Analytics layers. The layer encapsulations should guarantee transparent data flow between each layer and chosen middleware such that it can be easily replaceable with newest tools available in the IT world.

3 Challenges

There are challenges that need to be undertaken in optimising operations for data management, workflow management, and sites. The Operational Intelligence effort is addressing them by developing a set of general techniques, and tailoring them to the specific challenges.

3.1 Data management operations

One of the major problems in data management operations is to tackle the root cause of job failures related to missing input data, i.e., jobs could not start due to the input data not being available on time at the required compute site. The objective for Operational Intelligence is to design an intelligent predictive algorithm to send alarms in case of steady state violations, e.g., degraded transfer performances between storage systems. This means harmonizing and analyzing the data transfer metrics, which have already been collected and stored in a suitable analytics platform, and looking for potential correlations among the available metrics. These correlations can be used to send specific alarms and operational *actions* if potential anomalies are detected. The shifter/expert can then validate the proposed actions, which in turn sends feedback to the algorithm to improve over time.

In particular, we developed two alternative pipelines to address this problem using an unsupervised learning approach, thus enabling the discovery of new, unknown failure patterns. Both approaches analyze log messages through a Word2Vec model [5] that learns how to represent all tokens as Vector Space Model (VSM), thus allowing to vectorize log messages and discover their similarities. Considering the composite and diverse nature of the computing environment, we started analyzing the lowest-level source of information, i.e. data from the File Transfer Service (FTS) [6]. This makes the strategy as general as possible, and not limited to dedicated subsystems or experiment-specific configurations. A clustering algorithm is applied to group related errors, with the possibility of multi-stage clusterization to refine the model outcome.

The first pipeline is designed for (but not limited to) online processing and it exploits the DBSCAN algorithm [7] to gather similar messages together[1]. This choice allows the detection of clusters of varying shapes, which is particularly useful when dealing with outliers. A representative log pattern is then extracted from each cluster so that the shifter has to inspect just one error per group rather than all the single messages. The second approach is thought for offline processing and it is inspired by the work in [8]. During training, a K-Means algorithm [9] is adopted for clustering. The results are then validated by experts to build a knowledge base where an error category is attached to each cluster together with the solution. In production, the Word2Vec representation of a new message is compared to all the known issues and assigned to the closest category. If the minimum distance is greater than a threshold, a new issue is created and manual intervention is required. To validate our approaches we are building a reference dataset with labels for error categories and solving actions, which will ease the comparison of alternative algorithms and make the investigation of novel techniques sustainable.

3.2 Workload management operations

The succeeding rate for the payload execution in distributed computing environments may suffer from any of the factors noted in Section 1.2. Each failure event may require a unique approach to detection and investigation. Another impact on operations complexity is the large variety of leveraged systems. The sources of an issue can be anywhere among hundreds of computing elements, thousands of servers and many software layers. Additionally, non-infrastructure errors may occur due to defects in the software or user mistakes. To address these operational challenges, we started several projects for the automation of workload problem detection, and to optimise the information delivery channels.

[1] https://pypi.org/project/clusterlogs/

3.2.1 Jobs Buster

The Jobs Buster project aims at identifying workload problems and at taking automatic decisions on how to address spotted issues by using ML techniques. We are currently focusing on reliable issue detection functionality. Accomplishment of this goal itself could sufficiently reduce human efforts spent on workload operations. Issue detection has two primary tasks: identify complete sets of failed jobs (or delayed in execution) due to the same reason within each set, and find the root cause for each failure reason to precisely point to the problem.

We developed the following approach to extract a minimal feature set and their values to spot job failures: using a sample of failed and succeeded jobs we build a prediction model which is able to "forecast" the lost wall time for any job in the analysed sample. We use the job description retrieved from the workload management system as features which impact the job behaviour. Examples of such descriptors are: computing element where the job is running, software release version used to build the executable, the physics group the submitter belongs to, the memory requirements supplied in the job definition and others. Once the model is built we extract from it the principal factor responsible for the job misbehaviour. We then create a sub sample of jobs with the same principal factor and repeat the same procedure. This procedure points us to the next principal factor. Such recursive procedure is necessary because the same principal factor, for example a particular computing element, may correspond to both successful and failed jobs, and the failed jobs may still be due to multiple reasons: some jobs may have a memory leak, others a missing library dependency. The failure report generated by Jobs Buster is available in the monitoring framework of the ATLAS experiment [10], and experts are currently assessing its functionality.

3.2.2 Unified information delivery system

One of the most important requirements of successful operations is the ability to detect the right message and route it to the right person within a predefined time. With increasing scale of the computing resources and system complexity, the current monitoring systems must play a more proactive role in data processing and targeting. One of the directions of the Operational Intelligence initiative is to establish approaches for efficient information delivery and to minimize efforts on its adaptation for particular needs.

We deployed a message-based monitoring architecture which allows to encapsulate information producers as services which are completely isolated from the information distribution and visualization components, and execute them at the place where the operational data is actually generated. We use the Neural Autonomic Transport System (NATS) [11] for message delivering within system components. The messages are converted into metrics and injected into the VictoriaMetrics [12] system which is used as Prometheus [13] back-end. These metrics can be visualized in Grafana dashboards, and we rely on AlertManager [14] to notify users through various channels, such as Slack or email groups, about potentials anomalies. The system is used to target different CMS operators, which may be interested in different sets of metrics, for example all jobs failed at a certain site, failed jobs belonging to a certain campaign or task, or job failures with certain exit codes.

3.3 Site Operations

One peculiar instance of the Operational Intelligence efforts is focused towards the optimization of the computing operations at the distributed WLCG computing centers. Despite the diverse levels of maturity and automation of the adopted solutions, site operations relies still heavily on human efforts in the form of reactive maintenance. The richness of the already collected and available data (for example in log files) is not yet fully exploited to extract actionable insight. Solutions to reduce the need for manual intervention are being explored, relying on analytics components as well as data science methods (such as machine and deep learning).

Currently, both supervised and unsupervised approaches are used [15], as well as anomaly detection and log template extraction approaches [16], and various techniques adopted from Natural Language Processing (NLP) research. All methods are based on a quick prototype and validation cycle, in order to timely develop, train, and test models against new unseen log data being collected in real time, with the goal of validating one or more approaches as valuable to the early detection of symptoms of future failures at sites. The aim is to reduce the latency between the time a problem is detected and the time it is actually addressed (even aiming at anticipating its occurrence thus mitigating the impact of a possible failure), and to increase the quality and focus of operator interventions by equipping them with additional details, and correlations to other anomalies.

The ultimate goal is to eventually be able to address a variety of common issues that prevent an effective use of the resources of the computing sites, and move the bar from a run-to-failure approach to a series of designed diagnostic tools for ML-enforced predictive maintenance at sites.

4 Conclusions

The Operational Intelligence effort is the result of joint activities from various WLCG communities, aiming at reducing the manpower cost of operating distributed computing systems. This challenge is tackled at several angles: on one side we are developing intelligent algorithms to spot and predict issues, on the other we are developing a smart platform to support operators and computing experts in their daily tasks by exploiting the predictions of the studied ML models.

5 Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement LHCBIGDATA No. 799062. The work dedicated to the development of log clusterization methods was partially supported by Russian Science Foundation grant No.18-71-10003.

References

[1] I. Bird, Computing for the Large Hadron Collider, *Annual Review of Nuclear and Particle Science*, **61**, 99-118 (2011)

[2] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, *Journal of Instrumentation*, **3**, S08003 (2008)

[3] The CMS Collaboration, The CMS Experiment at the CERN LHC, *Journal of Instrumentation*, **3**, S08004 (2008)

[4] P. Tapia *et al.*, "Implementing Operational AI in Telecom Environments", Tupl White Paper (2018)

[5] T. Mikolov et al., *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781 (2013).

[6] E. Karavakis, A. Manzi, O. Keeble, M. Arsuaga Rios, “FTS improvements for LHC Run-3 and beyond”, 2020 in preparation for the proceedings of CHEP 2019 Conference, Adelaide (Australia)

[7] M. Ester, et al. *A density-based algorithm for discovering clusters in large spatial databases with noise*, Kdd, **Vol 96 No. 34**, p. 226-231 (1996)

[8] Q. Lin, et al., *Log clustering based problem identification for online service systems*, Proc. 38th Int. Conf. on Software Engineering Companion. ACM, p. 102-111 (2016)

[9] D. Arthur, S. Vassilvitskii, *k-means++: The advantages of careful seeding.* Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms, **8**, p. 1027-1035 (2007).

[10] Jobs Buster, *https://bigpanda.cern.ch/oi/jobsbuster/?hours=12&jobtype=prod*

[11] NATS *https://nats.io/*

[12] VictoriaMetrics *https://victoriametrics.com/*

[13] Prometheus *https://prometheus.io*

[14] Prometheus AlertManager *https://prometheus.io/docs/alerting/alertmanager/*

[15] L. Giommi *et al.*, “Towards Predictive Maintenance with Machine Learning at the INFN-CNAF computing centre”, Proceedings of International Symposium on Grids & Clouds 2019.

[16] L. Decker de Sousa et al., “Big Data Analysis for Predictive Maintenance at the INFN-CNAF Data Center using Machine Learning Approaches”. Proceedings of the 25th Conference of Open Innovations Association FRUCT 2019. IEEE p. 448-451.