SMOOTH MIGRATION OF CERN POST MORTEM SERVICE TO A HORIZONTALLY SCALABLE SERVICE

C. Aguilera-Padilla, S. Boychenko, M.Dragu, M.A. Galilee, J.C. Garnier, M. Koza, K. Krol, R. Orlandi, M.C. Poeschl, T.M. Ribeiro, M. Zerlauth, CERN, Geneva, Switzerland

Abstract

The Post Mortem service for the CERN accelerator complex stores and analyses transient data recordings of various equipment systems following certain events, like a beam dump or magnet quenches. The main purpose of this framework is to provide fast and reliable diagnostic to the equipment experts and operation crews to help them decide whether accelerator operation can continue safely or whether an intervention is required. While the Post Mortem System was initially designed to serve the CERN Large Hadron Collider (LHC), its scope has been rapidly extended to include as well External Post Operational Checks and Injection Quality Checks in the LHC and its injector complex. These new use cases impose more stringent time-constraints on the storage and analysis of data, calling for a migration of the system towards better scalability in terms of storage capacity as well as I/O throughput. This paper presents an overview of the current service, the ongoing investigations and plans towards a scalable data storage solution and API, as well as the proposed strategy to ensure an entirely smooth transition for the current Post Mortem users.

INTRODUCTION

The Post Mortem (PM [1]) service has been providing data collection, storage and analysis of LHC event data since 2008. The PM system usage grew significantly from the first use cases covering the understanding of the conditions in which the machine aborted its operations: fault on an equipment, powering event, etc. It currently covers the Injection Quality Checks (IQC [2]) from the SPS to the LHC and the Extraction Post Operational Checks (XPOC [3]) to verify that the beam extraction from the LHC was correctly executed. A new use case was just implemented to analyze every single cycle of the SPS. A possible integration of LINAC4 is under study. The initial design of the PM systems was to collect data on every beam dump, e.g. every 8 hours under normal operation conditions, or multiple times an hour when problems arise. The newer use cases call for real-time constraints. The analysis result for XPOC must be delivered within 10 seconds, an SPS cycle analysis must be given within 7 seconds in order to use the result on the next cycle. A LINAC4 cycle lasts about 1 seconds so an analysis that interlocks the next cycle requires real-time constraints for the data transfer protocol and for the analysis framework.

In its current design, the PM system provides a good vertical scalability, meaning that resources can be added to the nodes with minimum downtime and impact on the service availability. This way it could satisfy the original use case and withstand the first extensions. The increase in data throughput to the PM system and the SPS use case integration both demonstrated that its horizontal scalability must be improved to sustain all its current and future use cases as well as to accommodate larger events and analysis processes.

The first Section presents the current functional design of the Post Mortem service. The second Section presents the architecture of the PM system to support the functional design and identifies the current shortcomings. The third Section presents the improvements we propose, the steps that are planned to achieve the migration, and the first results of the migration.

FUNCTIONAL BEHAVIOR

The LHC control devices are continuously recording information about their current state in a rolling buffer which they are ready to dump on demand from an external event, or on their own trigger. The data dump is identified by the device information and by an “event time stamp” given by the device. Typical events are beam dump, powering faults, self triggers, etc. If the data dump could not be performed completely and the data were not serialized to the permanent storage despite the fallback mechanism, the device is warned so that it can store the data locally on the front-end controller storage and try to dump them later on.

All the collected data dumps are forwarded to the Post Mortem event-builder. The event-building is data driven. When it receives a first data dump coming from a control device, it sets a data collection time window. All the subsequent data dumps arriving during this time window with an event time stamp close to the one of the first one will be associated to the event. The event-builder identifies and characterizes the type of the event according to the data dump patterns. If the event-builder only received data from power converters and quench protection systems, it is considered a powering event. If it received data from every LHC system, this is a global event which is in average 190 MB.

The event is then analyzed by a hierarchy of analysis modules. Modules can be dedicated to the analysis of a single type of system, or they can perform correlation between multiple systems.

ISBN 978-3-95450-148-9

ARCHITECTURE

Control devices around the LHC use a Post Mortem client library to create and dump data to the Post Mortem service. The data are first received by Front-End servers that are dedicated to a system type, e.g. one Front-End server will collect all the data from the Quench Protection System, another one will collect all the data from the LHC Beam Dump System, as shown in Figure 1. The destination Front-End system is selected by the client library.

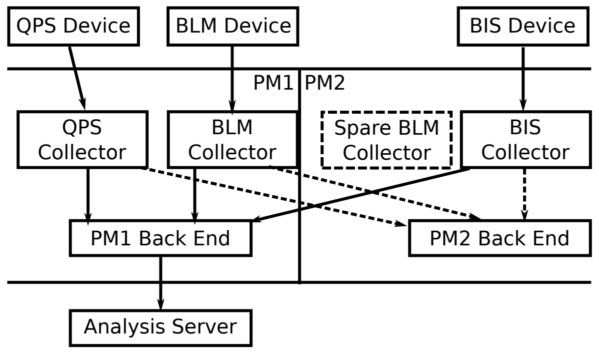

Figure 1: The Post Mortem data collection. The dashed lines represent the redundant collection path to the secondary Back-End server. The event-builder and the analysis tools are only attached to the main PM1 Back-End server. Devices can also dump to spare Front-End servers if the main one is not reactive.

Static Load Distribution

At first, the data flow was optimized between the redundant server nodes so that the load coming from the Front-End servers dump requests would be more or less evenly distributed. The data sizes, patterns and frequencies changed with time. The initial load-distribution is not adapted anymore and one server is clearly more stressed than the other. For a global event, in the current situation, one node collects in average 125 dumps that amounts to 40 MB of compressed data, while the other node collects in average 4073 dumps that amount to 150 MB. The total size of a global event collection varies depending on the cause of the event as some devices send buffers only when they are concerned. Other types of events follow different distributions of data per nodes. In addition, the PM system accepts a continuous flow of data. Due to the way the load distribution is implemented, it is difficult to scale it to cover optimally all use cases.

Monolithic Implementation

The production Front-End and Back-End servers are currently implemented in C++. The application protocol guaranteeing to the device that its data dump was completely stored is based on synchronous calls. The device dumping process will know that the data are completely written if the call to dump data returns successfully, e.g. without any exception.

The data collection servers are implemented in such a way that they do not handle data dumps in parallel but rather serializes them. The bigger the event, the more time is spent on collecting the data. It happens very rarely that a dumping device receives an RDA timeout while dumping data.

Data Consistency

Front-End servers store the data on a fast access non-permanent file system. They forward the data to Back-End servers that will store it on a permanent storage and notify the event-builder. There are two Back-End servers for redundancy. The Back-End processes enrich the data with information, like the reception time on the back-end system, or the path to the data on the file system. While the PM service ensures data storage redundancy, the data consistency cannot be ensured by standard tools due to the information added by the Back-End processes.

Data Storage Exposition

The file system is exposed via NFS to all the Post Mortem users, e.g. the analysis services and the equipment experts running custom analysis written in LabVIEW or Java. Until now it was not possible to perform analysis in another technology. Clients access the file system structure directly, either to use the raw data or to convert it into other formats and store it back on the Post Mortem storage. Therefore, PM data are duplicated to serve different use cases.

When the file system is under heavy load it causes delays in the file writing, which subsequently causes delays in the handling of the dump request by the data collection servers, which ultimately can trigger a time out in the dumping device.

Shortcomings

We identified the following shortcomings in the current architecture:

- Static load distribution
- Lack of data consistency verification
- Storage architecture and data format exposed to the end-users
- Data duplication
- File system performance limits and back pressure
- Monolithic implementation of data collectors
- Neither delivery nor ordering guaranty of data dumps

All this leads to performance limitations of the PM system that impact the implementation of new use cases. This calls for upgrades of the current system. The shortcomings are mainly due to in-house design of redundancy and distribution paradigms. The goal of the upgrade is to re-use available technologies to achieve fully horizontally scalable data collection, storage and analysis.

ISBN 978-3-95450-148-9

The data transfer from devices to the event-builder relies entirely on the CERN Control's MiddleWare (CMW [4]) RDA protocol. The current data collection uses RDA2 based on CORBA [5], with an on-going migration to RDA3 based on ZeroMQ [6]. The migration from RDA2 to RDA3 is on-going at the level of control devices. As long as some devices rely on RDA2, the current PM infrastructure must be maintained. There are currently 999 PM clients based on RDA2 and 915 based on RDA3. Multiple control devices can share the same RDA.

The challenge is therefore to upgrade the PM system while keeping it backward compatible for clients that would migrate to RDA3 slowly.

ROADMAP

The long term plan is to make the Post Mortem service horizontally scalable in all aspects: data collection, storage and analysis. This requires a drastic modification of the storage system. To modify the data format and the data storage, it is necessary to encourage users to use a common API to read and dump data as a first step. That way the data format and storage will be considered as a black box and it will be then possible to modify it at will.

Data Collection

The client API used to dump data is providing proper abstraction to the users. The API is implemented in C++ and in Java. It exposes the RDA serialization of the data to the user. This means a user knows RDA is used to dump data, and it knows that the data are serialized with CMW/RDA for the transport to the Post Mortem service. The client API currently supports RDA2 and RDA3 protocols. The exposure of the RDA protocols through this API is not a big issue, as user processes already rely heavily on these technologies. The client API is now integrated into the FESA framework [7], which is used by most of the control devices at CERN.

The missing feature for the data collection is dynamic load distribution, meaning that a client would dump Post Mortem data to any of the available data collection servers. This feature must be integrated into CMW/RDA3 before being used in the PM system. Alternatives could be to rely on other message protocols, like the underlying ZeroMQ protocol. The challenge is not to introduce a single point of failure.

Having introduced a dynamic load distribution, the client library will not be coupled anymore to specific PM targets. This means that a client will not dump to a specific data collection process anymore, but to any PM data collection process. This will increase the availability of the service and its maintainability. It will also allow the PM system not to rely on the Front-End/Back-End process anymore, but on a single layer of data collection process, as shown in Figure 2.

Data Access

The APIs to access data are implemented in Java or based on files. They both rely on the NFS exposure of the data.

A REST (Representational State Transfer) API is currently being implemented. The aim of this API is to serve multiple language technologies. Many users would like to perform analysis in Python or MATLAB. It will also provide data to the LabVIEW analysis framework. This way, no user will directly depend on the data serialization format or the file system. The data are exposed in plain JSON [8] or in Apache Avro [9]. The REST API is built incrementally. It currently provides the minimum amount of features for users to search for data using a few basic criteria. The aim of the first release is to provide the features that users implemented on top of the storage system, e.g. extracting directly one signal from the data without reading it entirely. This will encourage them to use the API and hide the file details. The overhead of accessing data through the REST API is very low. It adds an average delay of 8 ms to the file access, and a worst case delay of 73 ms when retrieving today's largest PM data.

The long term goal for the Post Mortem API is to enable complex queries, e.g. "Get all the Quench Protection Data when the Beam Energy was greater than 6.5 TeV". This can be achieved by extending the first release, adding indexing and correlating features. Open source products will be studied more thoroughly to address this requirement.

Storage

Once the data will be accessed only through the Post Mortem APIs and not directly by the users, the main upgrade of the Post Mortem storage will be possible.

The goal here is to provide complete data redundancy and integrity checks, as well as horizontal scalability. The requirements for the Post-Mortem storage are:

- Data replication
- High availability
- Error detection and correction
- High storage capacity
- High read and write throughput

The PM data which can be considered as semi-unstructured data. The structure is defined by the users and evolves with time. Quench Protection System data from 2008 does not look like data from 2015. In these conditions, it is difficult to restructure the data in a different format to store it. The data can only be stored in the same structure it was sent, as an immutable snapshot. Distributed object storage systems can therefore be considered as a good alternative to the current file system implementation. Distributed file systems are also a good option as they share many features with distributed objects storage systems.

Numerous solutions are available on the market. The Ceph [10] object storage, the Gluster [11] and HDFS [12] file systems are being considered for this upgrade. The plan

ISBN 978-3-95450-148-9

is to make an inventory of these technologies and to evaluate them according to our storage requirements.

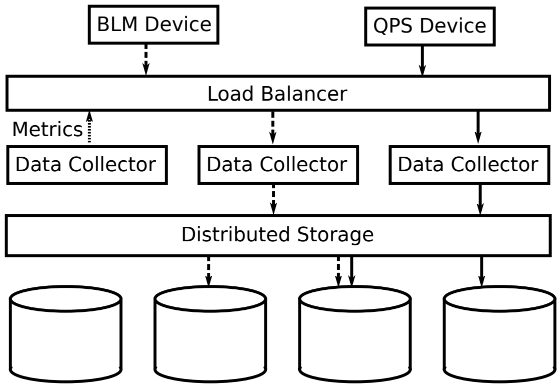

Figure 2: The targeted simplified Post Mortem architecture. Devices dump Post Mortem data to Data Collectors that are part of a load-balancing scheme. This way the load can be distributed among the Data Collectors of the Post Mortem cluster. The Data Collector that processes the dump writes the data to the distributed storage. The data are automatically replicated by the storage infrastructure.

The way to serialize files will also evolve. The plan is to enhance the usage of PM data for analysis. Considering that users may run analysis only on parts of the data dumps, the files must be splittable. Compression is also required to optimize the occupied space in the storage system. Indeed the signals often contain zeros and therefore have a high compression rate. After a thorough evaluation of multiple technologies, Apache Avro has been selected. The files can be compressed and still splittable at the same time, as the compression is performed on blocks of the Avro format, and not on the entire file itself. Avro also brings the flexibility required for the data structure to evolve. There is no need to generate specific classes to interpret the data, as the data themselves can embed the schema required to interpret them. The data will therefore still be self-described. The fact that Avro is a key technology for Mapreduce [13] jobs is another motivation for the choice of this file format.

Upgrade Steps

In order to perform an upgrade as smooth as possible for the users, the following steps are necessary:

1. Provide APIs to deprecate direct storage access
   - (a) Unique API for all languages to access data
   - (b) Data dumps from both CMW RDA2 and RDA3
2. Commission new data storage and serialization in parallel to operations
   - (a) Keep the current storage and its APIs running
   - (b) Automatic or semi-automatic data collection forwarding to new storage
   - (c) Run spare data analysis tools on new storage using same compatible API
3. Decommission old PM system
   - (a) Transfer all data to new storage
   - (b) Move production APIs to new storage, keeping them backward compatible
   - (c) Switch off

The first step is currently in the development phase and incremental releases are already performed for production usage. Storage solutions are being studied in parallel. As the PM system will not expose the storage nor the data format, it will be easier to perform migrations in the future.

CONCLUSION

The current Post Mortem system is still fulfilling its use cases successfully, and a second instance has been deployed to perform the SPS Quality Checks. The restart of the LHC after the first long shutdown and the growing need of more accurate analysis on higher data resolution in a deterministic time has pushed the Post Mortem system to its limits. The limitations of the current systems are clearly identified and the work already started to scale horizontally the Post Mortem system. This entire upgrade is the opportunity to make the system more maintainable and ease the future upgrades, providing a clear segmentation of the different areas of concern: APIs, storage architecture, data format, data analysis. It is also the opportunity to bring new features, e.g. allowing any language to work on PM data. Improving the storage layer is furthermore the opportunity to move toward data analytics on Post Mortem data, and enabling correlation of data from other CERN data storage as presented in [14].

REFERENCES

[1] O. Andreassen et al., “The LHC Post Mortem Analysis Framework”, proc. of ICALEPCS 2009, Kobe, Japan.

[2] V. Kain et al., “Injection Beam Loss and Beam Quality Checks for the LHC”, proc. of IPAC10, Kyoto, Japan.

[3] N. Magnin et al., “External Post-Operational Checks for the LHC Beam Dumping System”, proc. of ICALEPCS 2011, Grenoble, France.

[4] A.Dworak et al., “The new CERN Controls Middleware”, proc. of CHEP 2012, New-York, NY, USA.

[5] http://www.omg.org/spec/CORBA/3.3/

[6] http://zeromq.org/

[7] M.Arruat et al., “Front-End Software Architecture”, proc. of ICALEPCS 2007, Knoxville, TN, USA.

[8] http://www.json.org/

[9] http://avro.apache.org

[10] http://www.ceph.com/

[11] http://www.gluster.org/

[12] http://hadoop.apache.org/

[13] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, proc. of OSDI 2004, San Francisco, CA, USA.

[14] S. Boychenko et al., “Toward a Second Generation Data Analysis Framework for LHC Transient Recording”, WEPGF046, these proceedings, ICALEPCS’2015, Melbourne, Australia (2015).

ISBN 978-3-95450-148-9