This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/NSSMIC.2016.8069637, 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)

Optimised Lambda Architecture for monitoring WLCG using Spark and Spark Streaming

U. Suthakar[1], L. Magnoni[2], D.R. Smith[1], and A. Khan[1]
[1] Brunel University London, College of Engineering, Design and Physical Sciences, UK
[2] CERN, European Laboratory for Particle Physics (CERN), Geneva, Switzerland

I. Introduction

THE Worldwide LHC Computing Grid (WLCG) provides storage and computational power for four LHC experiments (ALICE, ATLAS, CMS and LHCb). The WLCG infrastructure is a highly distributed and heterogeneous platform with various middleware characteristics, job submission and execution tools, and diverse methods of transferring and accessing the dataset. The high computation activity and distributed nature of the WLCG make the system very complex. Hence, it increases the probability of failures or inefficiencies. Efficient monitoring is necessary in order to present a comprehensive strategy to recognise and resolve any issues within such a dispersed infrastructure. It is also important for the effective utilisation of computational and data resources.

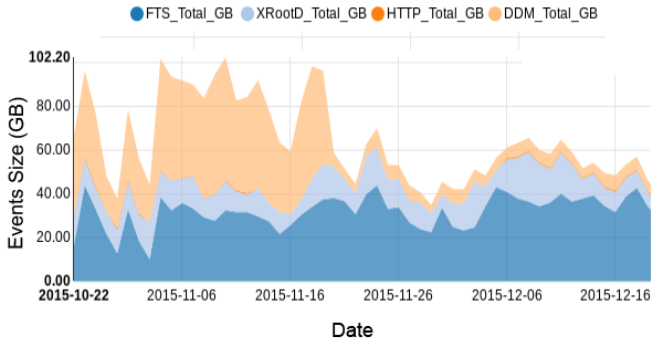

Fig. 1. WLCG data activity events size.

The original solutions for monitoring are based on the Oracle Relational Database Management System (RDBMS) for data storage and processing, but recent developments evaluate Lambda Architecture [3], in particular, data storage and batch processing for processing large-scale monitoring datasets using Hadoop and its MapReduce framework [1]. However, this approach does not support real-time monitoring. Therefore, this paper presents an optimised Lambda Architecture using Apache Spark technology, which involves modelling an efficient way of joining batch computation and real-time computation without the need to add complexity to the User Interface (UI). Three models were explored for WLCG Data acTivities (WDT) (e.g. average transfer rate from site A to site B over time): pure streaming, pure batch computation and the combination of both batch and streaming.

II. Background

The Experiment Dashboard system provides common solutions for monitoring data transfers, job processing and site and service status [2]. Monitoring events, metadata of the jobs and data transfers are collected and analysed to produce summary plots used by operators and experts to evaluate WLCG computing activities. However, due to the high volume and speed of the events that are produced as shown in Figure 1, the traditional methods are not optimal. Therefore, Lambda Architecture [3], an architecture leveraging many different technologies for supporting Big Data, was employed in order to support the WDT use case [1]. However, the complexity of combining and synchronising many technologies is a very significant concern.

The Lambda Architecture that is presented in [1] has demonstrated that it works well for monitoring. In particular, the WDT use case has shown that it outperforms the traditional architecture. However, with the three-layer structure, it comes at a price when integrating all three layers together to serve a single purpose goal, which is to monitor the infrastructure in real-time. Having different technologies for each layer will be difficult to integrate, implement and maintain. Hence, there is a requirement for a single solution that will accommodate and integrate batch processing as well as the real-time processing of monitoring data seamlessly. Apache Spark [4] is a new parallel processing paradigm similar to MapReduce, but with better analytical performance by exercising in-memory computation and therefore supporting iterative computation. It also supports data streaming, SQL-like commands, interactive command line, machine learning and Graphx. Having Spark batch and streaming under a stack is useful in optimising the Lambda Architecture. The Spark streaming and batch computations adapt the Resilient Distributed Dataset (RDD), an abstract data collection that is distributed across nodes for parallel processing. Therefore, transformation and computation logics can be reused between batch and streaming jobs.

III. Architecture and Design

The main requirement of any monitoring system is that it should be able to provide information about the infrastructure at least in near-real-time so that an appropriate action can be taken. Therefore, the following methods were designed and implemented:

1. Pure stateless batch computation, which can be scheduled to run at a preferred time interval, thus, it will not

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI: 10.1109/NSSMIC.2016.8069637, 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)

have any knowledge of the previous jobs. However, this does not support the real-time computation but Spark framework provides in-memory computations. Therefore, the execution time can be compared with the MapReduce framework that was used in [1]. The batch computation can also be used for historical computation.

2. Pure stateful streaming computation, which will carry out incremental computation on continuously streaming data, thus, it maintains the state of the computed statistics. It also has a checkpoint mechanism to dump the state to a disk; in case of job failure it can pick up from where it stopped. This method on its own is enough for real-time computation. Therefore, the complexities of merging multiple technologies as in the Lambda Architecture approach can be eliminated.
3. A combination of batch and streaming computation: the pure streaming is enough but there is the potential to get duplicate events from the message brokers due to system failures. Having pure streaming computation cannot address this issue as the raw events are dropped when they are processed. The state of the streaming job cannot keep the unique ID of the events once they are aggregated by the statistics group by key (e.g. sites). Incorporating batch computation can correct the inaccurate statistics as it will recompute whole datasets from the storage layer, such that it can eliminate duplicate events. Therefore, having a streaming job to do the continuous calculation and scheduling the batch job to run every hour to override the results to validate the statistics seems appropriate. To support this approach, the monitoring events were duplicated with one sent to an Hadoop Distributed File System (HDFS) for batch computation and the other streamed straight into the streaming receiver. However, there are a lot of complexities addressed in synchronising the two approaches together such as: informing the streaming job about the newly available data computed by batch job so that it can pick it up and override the streaming state as well as the serving layer that is used for storing computed statistics for serving the UI, and a mechanism to eliminate the network communication bottleneck at the serving layer to make sure that only the newly streamed data are updated/inserted into the serving layer.

IV. Results and Discussion

To evaluate how accurately the architecture was able to compute the WLCG site throughput in time-series all three proposed methods were tested. As shown in Figure 2, the stateless batch job was scheduled to run every 5 minutes to carry out batch computations on the data stored in an HDFS. However, the plot highlighted in Figure 2 shows that some data are missing. This is due to the latency of the batch computation and the unavailability of the data when the job started.

Figure 3 represents both the pure streaming and the combination of batch and streaming methods. Both methods are showing the computation in real-time as highlighted in the plot. This shows that both of these methods are capable of providing up-to-date statistics that are more beneficial to the

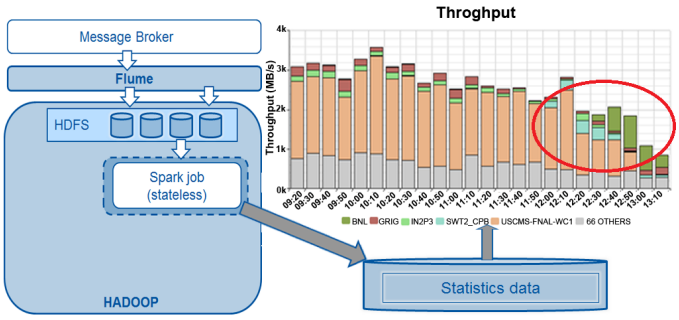

Fig. 2. The Spark batch computations for WLCG monitoring.

users in comparison with the pure batch computation. On average $\sim$ 15k events were received and processed by the streaming job per minute. Also, the Spark batch computation performed faster than the MapReduce job presented in [1] due to the use of in-memory processing so the intermediate results were cached into memory in comparison with the latter, which utilises the disk for reading and writing.

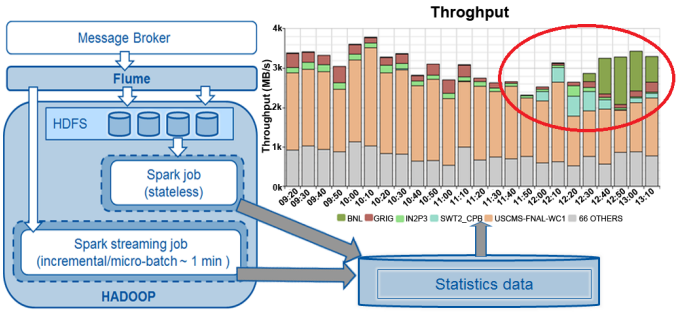

Fig. 3. The Spark batch and streaming computations for WLCG monitoring.

V. Conclusion

The three methods for effective monitoring presented in this paper outperform the RDBMS based system that is used by the WLCG in terms of execution time and scalability. In particular, the streaming approach provides an up-to-date state of the infrastructure. The WLCG group has adopted the optimised Lambda Architecture, which is a combined batch and streaming approach, and it has been used for monitoring the WLCG Data activities Dashboard since October 2015 [5].

References

[1] L. Magnoni, U. Suthakar, C. Cordeiro, M. Georgiou, J. Andreeva, A. Khan and D.R. Smith, Monitoring WLCG with lambda-architecture: a new scalable data store and analytics platform for monitoring at petabyte scale, *Journal of Physics: Conference Series 664 (5) (2015).*

[2] J. Andreeva , B. Gaidioz, J. Herrala, G. Maier, R. Rocha, P. Saiz and I. Sidorova, International Symposium on Grid Computing, *ISGC 2007 pp 131139 ISBN 9780387784168 ISSN 1742-6596.*

[3] N. Marz and J. Warren, Big Data: Principles and best practices of scalable realtime data systems, *Manning Publications ISBN 9781617290343.*

[4] Apache Spark, *Available: http://spark.apache.org [Online; accessed 20-04-2016].*

[5] WLCG Data activities Dashboard, *Available: http://dashb-wdt-xrootd.cern.ch/ui [Online; accessed 27-04-2016].*