An Overview of Monitoring Tools for Big Data and Cloud Applications

Gabriel Iuhasz
Institute e-Austria
Timisoara, Romania
iuhasz.gabriel@info.uvt.ro

Ioan Dragan
Institute e-Austria
Timisoara, Romania
idragan@ieat.ro

ABSTRACT

In his paper[1] we make a short overview of current state of the art monitoring tools for both cloud and big data frameworks. A hot topic for research in recent years is manipulation of big data, cloud computing and a combination of them. The focus on these topics is due to the problems that are posed by the manipulation of big data in cloud environments as well as the potential that they expose for obtaining better results. However, in order to effectively create, test and deploy new algorithms or frameworks one needs also suitable monitoring solutions. In this paper we aim on creating a critical overview for some of the most important monitoring solutions existing on the market. Besides that we also present the relevant metrics used for monitoring the cloud as well as big data applications, with the focus on cloud deployment scenarios for big data frameworks.

1. INTRODUCTION

During recent years a large number of companies and organizations tried to move part or even entire infrastructure in the cloud. This trend can be observed due to the new easy ways of delivering applications, process large workloads and even empower the end-user to work with some model of cloud.

Although the trend is so that lots of organizations tend to move their applications to the cloud, there are still problems that have to be addressed in order to ease usage of cloud technologies. One of the problems that has to be addressed is related to the way one can monitor the cloud in order to provide the user with aggregated information about cloud behavior in different situations. The major problem that arises in the context of monitoring is which metrics have to be monitored. Besides this problem one has to figure out what sort of monitoring solution fits the cloud model deployed.

This paper focuses on identifying key characteristics of a monitoring solution design for the DICE[2] project which aims to offer model-driven engineering accessible to big data application developer through an automated tool chain [23].

In the following we make a short overview of different monitoring solutions that can be applied for different big data and cloud deployments. We will start by presenting some basic information on Hadoop [7] and hadoop based big data frameworks. Afterwards we make a short overview of architectures and current technologies used for monitoring. Lastly we describe key characteristics we have identified in the context of the DICE project that will ultimately govern the design and implementation of the DICE Monitoring and data-warehousing solution.

2. CLOUD COMPUTING AND BIG DATA

Big Data is at the forefront of current distributed system research. It is geared towards data analytics on a never before seen scale. This in turn means that monitoring cloud deployments of big data frameworks is of paramount importance both for providers and consumers of such services. Firstly, it is important to monitor key performance indicators (KPI) of both the platform and application. Secondly, metrics related to Quality of Service (QoS) and Service Level Agreement (SLA) supply both the Provider and Consumer with metrics related to the overall quality and usability [19].

In this paper we focus mainly on monitoring solutions related to big data platforms. In particular we want to highlight the performance monitoring and how this can be used to fine tune a particular deployment. In the next sections we detail a few of the more representative big data frameworks and some of their most relevant KPIs.

Open Service for Lifecycle Collaboration (OSLC) [14] provides a standard way of specifying the operations supported by a performance monitoring provider. Using this standar one can easily integrate with different performance monitoring solutions that exist in the wild. The performance monitoring provider has to define different records that can be further used for testing and for deployment lifecycle. Individual resources as well as the relationship between them have to be defined as well as relations with resources from outside of the performance monitoring domain have to be

[1]This manuscript was submitted for review at MICAS 2015 worksop, affiliated with the 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2015), Timisoara, Romania.

[2]http://www.dice-h2020.eu/

defined. The OSLC specifications defines the HTTP-based RESTful interface in terms of classic HTTP methods: GET, POST and DELETE.

Hadoop[3] is an open source platform that is capable of storing and processing large scale data sets. One of the main features of hadoop is its ability to be scaled up to thousands of nodes. It is largely based on the MapReduce programming model [24]. Hadoop includes a distributed file system called HDFS and YARN which is a framework for schedueling and cluster resource management. The Hadoop ecosystem also includes tools such as Hive (data warehouse infrastructure), HBase (storage of large tables), Pig (dataflow language) and Spark (fast compute engine). All of the previously mentioned tools and frameworks extend Hadoops basic functionality in one way or another.

3. MONITORING TOOLS

3.1 Monitoring Architecture

On a cloud based deployment of big data platforms cross layer monitoring is a key factor. Application components can be distributed not only on different Virtual Machines (VMs) but also on different cloud layers. In these cases the monitoring parameters should cut across all cloud layers on which application components are deployed in order to give a complete picture of the current application status. Typically application are deployed on one or more of the following layers: Software as a Service (SaaS), Platform as a Service (PaaS) and/or Infrastructure as a Service (IaaS).

On IaaS typically we want to monitor resource utilization such as CPU usage and states, Hard Disk utilization, Memory usage and status as well as additional network parameters. In contrast at Paas and SaaS level parameters include byte throughput metrics, status of system services, uptime, availability etc. For example, in the case of a Hadoop deployment we have metrics such as MapReduce processing time, Job Turnaround, Shuffle operations etc.

The type of resources that are monitored is highly dependant on the application type. For example data transfer quality and rate is important for any video streaming application while a batch processing application will only care about basic process and network latencies.

There are several types of monitoring solutions currently in use or in development. In the case of centralized monitoring, all resource states and metrics are sent to a centralized monitoring server. These metrics are continuously pulled from each monitored component. It is easy to see that this approach while allowing a more controlled management of any cloud application has several drawbacks. First, it has a single point of fail over and lacks scalability. This means that at a certain stage the monitored application will exceed the capability of the central monitoring server and the only solution in case of centralized monitoring is that of vertical scaling. Moreover high network traffic can also lead to bottlenecks which in turn can lead to faulty or incomplete monitoring data. A decentralized approach can alleviate these problems.

[3]https://hadoop.apache.org/

In a decentralized architecture no component is considered more important than another. In Structured Peer-to-Peer systems the central authority is defused thus eliminating the central point of failure. Unstructured Peer-to-Peer network overlay is meant to be distributed. However, the search directory is not centralized. And lastly we have the hybrid Peer-to-Peer sysytems in which super peers can serve as localized search hubs for small network portions /citeAlhamazani15.

Next we will detail some of the most used monitoring tools and platforms in the context of cloud computing and big data. Some of these platforms have been adopted from HPC scenarios while others have been designed specifically for this task.

Hadoop Performance Monitoring UI [7] provides an Hadoop inbuilt solution for quickly finding performance bottlenecks and provide a visual representation of the configuration parameters which might be tuned for better performance. Basially it is a lightweight monitoring UI for Hadoop server. One of its main advantagest is the availability in the Hadoop distribution and the eas of usage. On the other hand it proves to be fairly limited with regard to performance. For example, the time spent in gc by each of the tasks is fairly high.

SequenceIQ [17] provides a solution for monitoring Hadoop clusters. The architecture proposed in [6] and used in order to do monitoring is based on Elasticsearch [4], Kibana [9] and Logstash [10]

The architecture proposed by [6] has the main objective of obtaining a clear separation between monitoring tools and some existing Hadoop deployment. For achieving this they use three Docker containers.

In a nutshell the monitoring solution consist of client and server containers. The server container takes care of the actual monitoring tools. In this particular deployment it contains Kibana for visualization and Elasticsearch for consolidation of the monitoring metrics. Through the capabilities of Elasticsearch one can horizontally scale and cluster multiple monitoring components. The client container contains the actual deployment of the tools that have to be monitored. In this particular instance it contains Logstash, Hadoop and the collectd module. The Logstash connects to Elasticsearch cluster as client and stores the processed and transformed metrics data there.

Basically the proposed solution consists of a collection of tools that are used in order to montior different metrics from different layers. One of the main advantages of this solution is the ease of adding and removing different components from the system. Another interesting aspect of this architecture is the ease with which one can extract different informations from tools.

Hadoop Vaidya [8] (Vaidya in Sanskrit language means "one who knows", or "a physician") is a rule based performance diagnostic tool for MapReduce jobs. The mechanism behind Vaidya is to perform post analysis steps for map-reduce jobs. For this purpose it parses and collects different execution statistics from job history and different configuration files.

In order to obtain differnet performance problems, Vaidya, performs a set of predefined rules agains the collected job execution statistics. Since the system is rule based, each of these rules adresses a specific problem that could arise in a map-reduce job. After each rule is applied it provides the user with advice for the future. As output the tool produces an XML report that is based on the evaluation of individual test rules.

Ganglia [5], is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. The main target for Ganglia is federation formation of clusters and is based on a hierarchical design. Ganglia heavily relies on technologies as XML for data representation, XDR for data transport as well as RRDtool that is used for storing data as well as data visualization. Due to its design it manages to achieve very low per-node overhead as well as high concurrency.

The tool is designed such that it is robust and easy to port to different operating sustems. While developed it was ported to a vast set of poerating systems and processor architecture. Currently it is in use for thousands of clusters around the world. One of its major pluses it the cappability to scale up in order to handle clusters that consists of thousands of nodes. Currently being used to connect clusters from different university campuses around the world.

The *Apache Ambari* [1], is yet another tool that aims at making Hadoop management simpler. Ambari project is developing software that for different tasks as provisioning, managing and monitoring Apache Hadoop clusters. At its core, it also provides an easy to use and intuitive Hadoop management web user interface trough RESTful APIs.

Apache Chukwa [2] is an open source data collection system for monitoring large distributed systems. Chukwa is built on top of the Hadoop Distributed File System (HDFS) and Map/Reduce framework. Since it uses these technologies it is easily scalabel and robust. Besides collecting the monitoring data, it also provides a powerfull toolkit that allow users to monitor, display and analyze results of different runs in order to better understand the collected data. The tool is released under Apache 2.0 licence.

Datastax [3] provide a solution, *OpsCenter* [16], that can be integrated in order to monitor Cassandra installation. Using OpsCenter on can monitor different parameters of the Cassandra instance and also different parameters provided by the actual machines on which it runs. Also, OpsCenter exposes an interactive web UI that allow administrators to add/remove nodes from the deployment. An interesting feature provided by the OpsCenter is the automatic load balancing. For integration of OpsCenter with other tools and services an developer API is provided.

MMS [12] is one way of monitoring and keeping under control a MongoDB deployment. MMS is a cloud service developed by the developers of MongoDB. It provides an integrated and easy way to provision, monitor, backup and scale MongoDB on the infrastructure of choice. It also includes out of the box support for Amazon Web Services (AWS). Deployment of an instance of MongoDB using AWS can be done by a couple of mouse clicks.

Server Density [18] is a tool that provides an interesting way of monitoring different systems. Alongside with the monitoring itself, it provides a way to clearly visualize the data. One of the main advantages of Server Density is it's versatility. Monitors from basic user profile up to high throughput processing of over 30TB/month of time series data. It also comes with an interesting licencese, that is "All you can eat 15 days free trial! while 1 server and 1 check costs 10$/month"

Applications Manager [11] enables the users to monitor the performance of Cassandra and also perform administration tasks of all the nodes in a cluster in a centralized manner. One can collect different statistical data from the Java Virtual Machines (JVM) that run in a cluster. Besides Cassandra the tools offered by Manage Engine cover a wide variety of other applications that can be monitored. When it comes to NoSQL DB's it can also monitor MongoDB instances and others.

Nagios [13] monitoring platform supports multi-layer monitoring. Due to its plugin based architecture, Nagios provides a way of monitoring both cloud based resources as well as inhouse infrastructure. In order to achieve this it uses SNMP monitoring network resources. The architecture of Nagios requires a centralized server in order to collect the monitoring data. However, it is possible to create a hierarchy of Nagios servers that mitigate the disadvantages of a centralized server.

OpenNebula [15] is a monitoring solution that provides management of data centers. It relies on SSH to connect to and gather monitoring data. It's main design goal is to monitor physical infrastructure as well as private cloud deployments.

4. MONITORING REQUIREMENTS

In the DICE project there is a need for a monitoring and data warehousing solution. It must be able to collect and serve monitoring data from a variety of big data frameworks. Its main function is to aggregate data and to serve it to a variety of different DICE tools. In essence the monitoring solution has to be able to collect monitoring data across multiple cloud layers. We have decided basted on the state of the art analysis that the best solution is one based on the so called ELK stack - Elastic Search , Logstash and Kibana [26]. The ELK stack is also used by SequenceIQ as presented in Section 3.

Cloud monitoring tools have a wide range of open research issues [20]. These include but are not limited to: Elasticity, Scalability, Resiliance, Reliability, Adaptability, Autonomicity, Accuracy and Timeliness. The most important for DICE monitoring platform is related to scalability. As we stated in Section 2 a Hadoop deployment can be comprised of thousands of nodes. Each node produces logs from system resource utilization as well as Hadoop services. This has the potential to create a huge workload for the Logstash server (especially when grok parsing is used) and can lead to unreliable data and even data loss. The good news is that logstash server is scalable in the sense that more than one logstash server can be used with an elasticsearch instance.

Moreover, elasticsearch in itself is relatively easy to scale.

In some instances the logstash pipeline may exceed the elasticsearch clusters capabilities to assimilate the incoming data. This scenario requires a buffer between the two components in the form of a queueing service. Logstash is capable of throttling incoming data flow if it exceeds the index rates however, this is not enough. By adding a message queue we can prevent data loss. If a logstash instance fails to read incomming data this data can be redirected to an open instance. There are a great number of message queue services which can be used such as: Redis[22], Kafka[25] and RabbitMQ[21]. Logstash provides plugins for both input and output, that enable almost seamless integration with these types of tools. Kafka will provide a distributed backbone and data pipeline that enables the integration into the DICE monitoring and data warehousing solution of advanced machine learning and anomaly detection engines.

In the context of monitoring systems timeliness refers to the fact that events should be available on time for their intended use [29]. In essence this means that a monitoring framework has to be able to handle short time intervals between metrics. This enables the creation of up to date and most importantly accurate information about a systems current state.

Last but certainly not least the DICE monitoring platform has to be autonomous in the sense it has to be able to self-manage its distributed resources by automatically reacting to unpredictable changes [28]. Most big data platforms that are deployed on a cloud infrastructure are on demand deployments. This means that a monitoring platform has to be able to be deployed alongside it. Another usecase that requires a certain degree of autonomous behaviour is when an already deployed big data platform has to be monitored. In this case there has to be a mechanism that allows the discovery of all the running services in the deployment and of the underlying hardware (or VM whichever the case). This can be done using a software agent which once uploaded into a target VM (i.e. via SSH). This software is able to detect all running services and forward relevant performance metrics and logs to the logstash server. One example of such an agent is the chef-client used in the Chef configuration and management tool [27].

5. CONCLUSION

In this paper we have briefly described current cloud computing and big data frameworks monitoring challenges and available platforms. We have highlighted which open research questions are of paramount importance in the monitoring solution that will be implemented for the DICE project. Namely we have identified that scaling, autonomy and timeliness are the key challenges which we have to tackle during the design and implementation of the DICE monitoring and data-warehousing solution.

6. REFERENCES

[1] Apache ambari. https://ambari.apache.org.

[2] Apache chukwa. chukwa.apache.org.

[3] Datastax. http://www.datastax.com.

[4] Elasticsearch. https://www.elastic.co.

[5] Ganglia. http://ganglia.info.

[6] Hadoop monitoring. http://blog.sequenceiq.com/blog/2014/10/07/hadoop-monitoring/.

[7] Hadoop toolkit. https://code.google.com/p/hadoop-toolkit/wiki/HadoopPerformanceMonitoring.

[8] Hadoop vaidya. http://hadoop.apache.org/docs/r1.2.1/vaidya.html.

[9] Kibana. https://www.elastic.com/producs/kibana.

[10] Logstash. http://logstash.net.

[11] Manage engine. https://www.manageengine.com/.

[12] Mms mongodb. https://mms.mongodb.com/.

[13] Nagios. https://www.nagios.org/.

[14] Open services for lifecycle collaboration. http://open-services.net.

[15] Opennebula. http://opennebula.org/.

[16] Opscenter. http://www.datastax.com/what-we-offer/products-services/datastax-opscenter.

[17] Sequenceiq. http://sequenceiq.com/.

[18] Server density. https://www.serverdensity.com/plugins/mongodb.

[19] G. Aceto, A. Botta, W. De Donato, and A. Pescapè. Survey cloud monitoring: A survey. *Comput. Netw.*, 57(9):2093–2115, June 2013.

[20] K. Alhamazani, R. Ranjan, K. Mitra, F. Rabhi, P. P. Jayaraman, S. U. Khan, A. Guabtni, and V. Bhatnagar. An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art. *Computing*, 97(4):357–377, Apr. 2015.

[21] S. Boschi and G. Santomaggio. *RabbitMQ Cookbook*. Packt Publishing, 2013.

[22] J. L. Carlson. *Redis in Action*. Manning Publications Co., Greenwich, CT, USA, 2013.

[23] G. Casale, D. Ardagna, M. Artac, F. Barbier, E. Di Nitto, A. Henry, G. Iuhasz, C. Joubert, J. Merseguer, V. Munteanu, et al. Dice: Quality driven development of data intensive cloud applications. 2015.

[24] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[25] N. Garg. *Apache Kafka*. Packt Publishing, 2013.

[26] R. Kuc and M. Rogozinski. *Mastering ElasticSearch*. Packt Publishing, 2013.

[27] M. Marschall. *Chef Infrastructure Automation Cookbook*. Packt Publishing, 2013.

[28] R. Mian, P. Martin, and J. L. Vazquez-Poletti. Provisioning data analytic workloads in a cloud. *Future Gener. Comput. Syst.*, 29(6):1452–1458, Aug. 2013.

[29] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, pages 141–150, New York, NY, USA, 2011. ACM.