A Framework for Big Data as a Service

Quang Hieu Vu [1], Rasool Asal [1,2]
quang.vu@kustar.ac.ae, rasool.asal@bt.com
[1] Etisalat BT Innovation Center (EBTIC), Khalifa Univeristy, UAE
[2] British Telecommunications, UK

***Abstract*—Cloud computing has become popular and grown to be a dominant computing model as a result of several advantages it has compared to other computing models. In addition to having desirable properties such as agility and flexibility, cloud computing helps users to save a significant amount of expense that is related to infrastructure investment, management and maintenance. Many types of services have been hosted in different stacks of the cloud from low-level stacks such as Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) to high-level stacks such as Software as a Service (SaaS) and Data as a Service (DaaS). Given that anything can be developed and deployed as a service in the cloud and an increasing demand for big data processing, a new type of cloud computing service, Big Data as a Service (BDaaS), has been introduced recently. In this paper, we will present a high-level design of a general framework for BDaaS. But prior to that, we will discuss properties and challenges of building such framework.**

***Index Terms*—Big data; Cloud computing; Service; Framework;**

I. Introduction

Cloud computing is a computing model that provides users on-demand access to a shared pool of computing resources. In this way, compared to traditional computing models, cloud computing provides users several advantages such as agility and flexibility. Besides, with the pay-as-you-go model, cloud computing has fundamentally changed the way users pay for computing resources and allowed them to have a significant cost saving. In particular, cloud computing helps users to save a large amount of expense that is related to infrastructure investment, management and maintenance. In cloud computing, anything can be a service in which the most popular ones are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These are basic services coming from the three main stacks of the cloud platform.

Big Data as a Service (BDaaS) is a relatively new cloud computing service that comes together with the growing trend of big data. The basic idea of BDaaS is to provide services for big data analysis. Even though a number of BDaaS have been introduced in literature, they only support limited functionality. In particular, most existing BDaaS target to only provide either high-level big data analysis services (i.e., a type of SaaS) or a deployment of big data platforms [1] (i.e., a variance of IaaS), which has a fixed setup on the storage and processing model. These types of services are not suitable to users who need extra customization on the data processing. As an effort to provide more flexibility to users, in this paper, we propose a design of a general framework for BDaaS, which is able to provide services for big data at different levels of cloud stacks from IaaS to PaaS and SaaS.

There are many challenges of building a general framework for BDaaS that maximizes benefits of the cloud platform and provides users adequate features together with ultimate flexibility. In this paper, we identify challenges and propose a high-level design of the framework in four steps. Initially, we analyze basic requirements of the framework. Then, based on the requirements, we list main properties of the framework. After that, challenges in building the framework to support each of the properties are identified. Finally, a general framework is designed to address these identified challenges. To summarize, these four steps of building a general framework for BDaaS makes our two following major contributions:

- An analysis of basic requirements and then main properties of a general framework for BDaaS based on which core challenges are identified.
- A high-level design of the framework with detailed discussion on each of the framework's components.

The rest of paper is organized as follows. In Section II, we show related work. In Section III, we respectively analyze basic requirements, identify main properties and discuss challenges of building a general framework for BDaaS. In Section IV, an initial design of the framework is introduced and its components are discussed one by one in details. In Section V, we make a conclusion and present our future work.

II. Related Work

In this section, we show related work in three categories. First, we introduce work that discuss benefits, features and challenges of hosting big data in the cloud. After that, we present work that discuss specific solutions and applications for big data in the cloud. Finally, we discuss various cloud based frameworks targeting big data.

A. Benefits, features and challenges of big data in the cloud

Before targeting big data in the cloud, it is necessary to understand benefits, features and challenges of such a target. A number of work have been introduced for this purpose. In particular, [2] presented a study of existing states and potential future opportunities for big database management systems in the cloud, focusing on update intensive applications and decision support systems for deep analytics. On the other hand, [3] shown different case studies of big data processing in the cloud from which open issues, challenges and research directions are discussed. Besides, [4] discussed big data management in the cloud from the perspectives of both cloud provider and cloud consumer while [5] discussed how big data problems can be minimized in the cloud using Hadoop architecture. Similar to our work, an overview of the three basic big data services: big data infrastructure as a service, big data platform as a service and big data software as a service was introduced in [6]. This work, however, simply presents the concept of these three services without any further discussion of how to design and provide these services in the cloud.

B. *Solutions and applications for big data in the cloud*

Given several benefits of hosting big data in the cloud, a number of specific solutions and applications have been developed. Specifically, [7] explored service-oriented enterprise architecture for big data applications in the cloud while [8] proposed a universal storage architecture for big data in the cloud, which is able support different data models across cloud clusters. In a different approach, [9] proposed a system architecture to address challenges of running big data workflow management systems in the cloud. Focusing on the management and usage parts, [10] introduced a general dataflow-based approach for performance analysis of big data cloud and a specific implementation for the approach in Hadoop while [11] presented an optimization algorithm for timely, cost-minimizing of moving geo-dispersed big data to the cloud. On the other hand, [12] introduced an approach to reduce the cooling energy cost for big data analytics cloud.

C. *Frameworks for big data in the cloud*

There have been a number of work whose target is to build frameworks for big data in the cloud. Specifically, [13] proposed a cloud framework to support large-scale data analytics and visualization with a case study on climate data of various scales. [14] presented a cloud based framework that integrates R to provide rich data statistical and analytics functions for big data mining and analyzing services. [15] introduced a big data cloud framework called distributed GraphLab, which is an extension of GraphLab [16], for machine learning and data mining. CDH (Cloudera Distribution Including Apache Hadoop) [17] is a typical framework for Apache Hadoop distribution. All these frameworks are not general since they target specific applications or issues in the cloud. The closest work to ours is [1]. Similar to ours, this work discusses challenges and presents an approach to build a framework for BDaaS. The main difference, however, is that this work is limited to support big data PaaS while our work aims to provide not only big data PaaS but also big data IaaS and big data SaaS. Furthermore, our framework is designed from the point of view to support all basic properties, in addition to defining links among services in the three basic stacks to make them work as the whole in the system.

III. Properties and Challenges of a general Framework for BDaaS

In this section, we will first discuss basic requirements of the framework from which the set of important properties required for the framework is derived. Then, based on this set properties, we will show challenges of building such a framework.

A. *Requirements and Properties of the Framework*

We analyze requirements of a general framework for Big Data as a Service (BDaaS) from two basic points of views. On the one hand, being developed as a cloud computing service, the first and also the most important requirement of the framework is to preserve all properties of the cloud computing model of which agility and flexibility are the most critical ones. Besides, as in any cloud platform, the design needs to be practical for implementation and usage. On the other hand, since BDaaS targets data processing, data security is an important requirement to be addressed. Furthermore, since the size of data is big, it is necessary to have a mechanism to support optimization in data job allocation (i.e., data processing optimization). Based on this requirement analysis, we list important properties of the framework and discuss these properties in details as follows.

- *Agility*: This property implies the ability to provide dynamic configuration to grow and shrink data clusters easily. By using the term "data clusters", we emphasize that the property is supported not only at the data storage but also at the data processing stage.
- *Flexibility*: As there could be different types of data storages (e.g., Hadoop File System or Google File System) and different models of data processing (e.g., MapReduce or Spark), this property allows user to link different data storage types to different data processing models.
- *Security*: This property provides the ability to support both security and fault-toleration for data in different basic states, e.g., at rest in the data storage, in motion when data is transferred across nodes for computation, and in processing when it is being processed.
- *Practicability*: This property requires a design of the framework to provide complete supporting features including search and discovery for services hosted in the framework, different payment models for using the services, and management for Quality of Service (QoS).
- *Efficiency*: This property mainly involves optimization in resource scheduling, in particular data processing job allocation in the framework. The optimization part needs to deal with two main aspects: optimization in resource consumption and optimization in data transfer.

B. *Main challenges in the Design*

In the previous part, we presented a list of basic properties for the framework. In this part, we discuss challenges we face in the design of the framework with respect to each of the above listed properties. Details are as follows:

- *Agility*: While it is easy to provide agility for data space in storages, the main challenge is how to scale computational resources for data processing. Besides, it is also a challenge to redistribute data jobs efficiently when data size grows or shrinks.

- *Flexibility*: The challenge here is to answer the question: how many stacks are there in the framework and where BDaaS stacks are located in the cloud stack as well as how to combine and link different components across different stacks together.
- *Security*: Providing data security and fault-tolerance for data in storages is not very difficult. However, it becomes more and more challenge when we need to support security and fault-tolerance for data in motion and then data in processing.
- *Practicability*: Even though people can say that existing service description, payment, and QoS models used in current cloud computing services can be employed in the framework, the real challenge is how to adapt them to take into account unique features of BDaaS.
- *Efficiency*: There are basically two approaches for optimization: static and dynamic. In the context of BDaaS, it is desirable to support both static and dynamic optimizations and in this case, the big challenge comes from dynamic optimization.

IV. Framework high-level design

Big data processing systems usually have two basic layers: distributed file system and data processing. However, given that the data processing layer only provides basic operations such as Map and Reduce in the MapReduce model, many supporting libraries are built on top of that. Thus, we design a three-level stack for our framework in which the bottom two stacks are the same as those in existing big data processing systems: data storage and processing engine while the top stack is for supporting libraries and also data analytics services. Looking from the cloud computing point of view, these three stacks are actually similar to the three basic stacks of a cloud platform: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Together with the resource scheduling, these three stacks form the heart of our framework. Supporting for these core stacks are four other components: Service Description and Discovery, Payment Management, QoS Management, and Data Security and Fault-Tolerance. An overview of our framework is shown in Figure 1. In the rest of this section, we will respectively present each of the stacks and components of the framework.

A. Data storage

In our design, the data storage stack is situated at the IaaS stack of the cloud. Basically, a service request for a data storage in our framework is similar to a request occurred at the IaaS stack. The difference, however, is that while an IaaS request is only linked to a number of virtual machines (VMs), a request in our framework further links to virtual disks working as a storage for the data. In particular, a data storage request consists of two elements: a number of virtual disks (and their capacity) and a number of data nodes (i.e., VMs) in which each data node connects to a number of virtual disks in the storage. In addition, when requesting data storage, users are able to specify any type of storages available in the framework (e.g., Hadoop File System [18] or Cassandra database [19]) and that specific storage type will be provided.

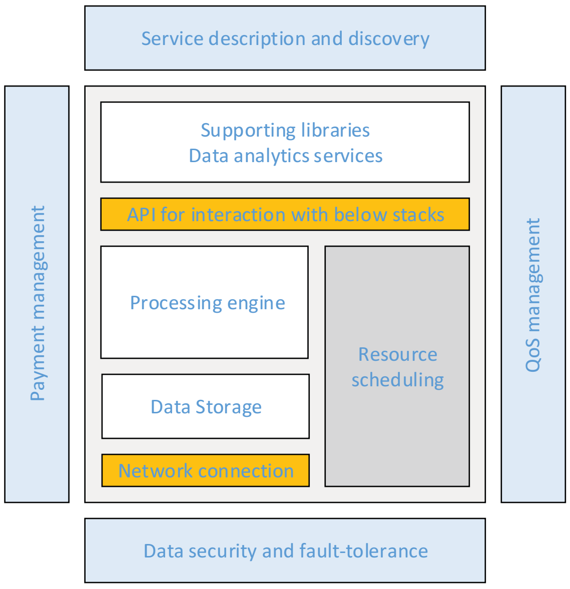

Fig. 1. Architecture of the Big Data as a Service framework

B. Processing engine

A service request of a processing engine can be treated either separately or in combination with a data storage request. On the one hand, when the request is treated separately, the framework provides a link to connect the processing engine with the underlying data storage. In this case, it is important that the framework checks the compatibility between the processing engine and the data storage since a specific processing engine may only work with a certain number of data storages. On the other hand, when the request is combined with a storage request, it is similar to a PaaS request in the cloud in which users do not need to describe the link between the storage and the processing engine. Instead, they are deployed as the whole to serve the request. For example, a combination request of a Hadoop MapReduce will deploy a Hadoop cluster that includes both the MapReduce processing engine and the Hadoop Distributed File System. As in the case of a data storage request, different processing engines are available for selection in the framework (e.g., MapReduce [20] or Spark [21]).

C. Supporting libraries and analytics services

This stack is setup as a SaaS in the cloud framework. Service requests in this stack often involve supporting libraries and end-user analytics services. The purpose of this stack is to provide service developers a way to deliver usable products directly to end-users. While a deployment of supporting libraries and analytics services in this stack depends on the processing engine and data storage in the below stacks, as in any cloud SaaS, users using the services only care about the services themselves without any need to know the below processing engine and data storage. Instead, service developers take full responsibility for all activities below the stack. To support service developers, the framework provides a set of standard APIs for the communication with the below stacks. An example of an analytics service that can be deployed in this stack is Google's big query [22] while examples of supporting libraries include Mahout [23] that provides core algorithms for clustering, classification and collaborative filtering and Giraph [24] that supports iterative graph processing.

D. Resource scheduling

This resource scheduling component is different from the inner job scheduling part of processing engines. While the inner job scheduling part takes responsibility for scheduling tasks of processing engines (e.g., Map and Reduce tasks of the MapReduce engine), our scheduling component is used to manage and schedule jobs and activities across VMs from different data clusters. As discussed before, the resource scheduling component supports both static and dynamic scheduling. In dynamic scheduling, the framework monitors workload of virtual machines (VMs) where processing engines are running and performs re-allocation of jobs if necessary. We design the dynamic resource scheduling based on the idea of SkewTune [25]. However, we cannot reuse the whole idea of SkewTune as it is limited to the Hadoop MapReduce. Instead, we execute job re-allocation based on the performance of VMs as well as information on the network connection between data clusters and VMs (i.e., data nodes).

E. Service description and service discovery

In a previous work, we proposed DEMODS [26] as a description model for Data as a Service (DaaS). In this framework, we extend DEMODS to provide service description for big data services. For this purpose, we make two basic changes in the service description model. The first change is to reflect the existence of the processing engine. It means that instead of having only two levels of service description as in DEMODS, we now have three layers of description: data storage, processing engine, and service. In addition to adding the new layer, we also make some changes in the two existing layers to reflect that the services are for big data. On the other hand, the second change is to provide a specification to link service description at the top stack to processing engine and data storage in the bottom stacks.

F. Payment management

Since we treat data storage and processing engine separately, we employ two different pay-as-you-go models for services in these two stacks. One payment model charges cost per data node and data size of virtual disks connecting to the data node (i.e., the storage cost) while the other model calculates cost based on computational resources consumed by the processing engine. Additionally, the two cost models are linked together to determine the cost of data transfer between data nodes and the processing engine. In cases the request comes from the top stack with supporting libraries or analytics services, the cost is calculated based on requests triggered at the below data storage and processing engine stacks.

G. QoS management

For QoS management, the center task is to evaluate the quality of services. In our framework, given a service, we first need to get all properties that reflect the QoS (e.g., properties of a basic data analytics service can include data availability, data security, and service response time). Then, we employ the solution presented in this paper [27] to monitor all of these properties. Based on the monitoring results, at any point of time, the service quality can be determined. While there are a number of options that can be employed for QoS management (e.g., actions to improve QoS to enforce service level agreements), in our current design of the framework, QoS management is simply linked to payment management because in many cases it is preferable that the cost of using a service depends on the quality of that service.

H. Data security and fault-tolerance

For data security, we employ a number of strategies. First, we provide strict access control policies for data at rest and encrypt data that is simply archived in the storage (we assume that data having a low frequency of access is archived data). Then, when data is sent across data nodes, a secure encrypted communication channel is used. Finally, we employ techniques in trusted computing [28] to provide trusted engines that maintain certain degrees of security for data in processing. In addition to data security, to provide fault-tolerance, we link this component with the QoS management part to predict potential failures and employ proactive policies to perform preemptive actions (e.g., when the data failure rate tends to increase, the framework can replicate data in the storage or re-allocate data processing jobs to avoid disruption in providing data and services).

V. Conclusion and Future Work

In this paper, we have discussed requirements and properties of a general framework for Big Data as a Service (BDaaS) and introduced an initial high-level design for such a framework. In general, the framework is designed with three basic stacks: data storage, processing engine, and supporting libraries (or analytics services), which respectively correspond to IaaS, PaaS, and SaaS stacks in a cloud platform. These three stacks are linked together and supported by a resource scheduling to form the heart of the framework. Surrounding these basic stacks are four functional components: service description and discovery, payment management, QoS management, and data security and fault-tolerance, each of which serves a specific purpose for the framework.

In the future, we plan to convert this high-level design of the framework into a low-level design before implementing a proof-of-concept prototype for the design. For the implementation, we intend to employ and customize OpenStack [29], the most popular open-source cloud platform at the moment. Once the prototype is finished, we will aim to deploy it inside EBTIC to serve our own in-house big data analytics services before going for further tests in BT data centers.

References

[1] J. Horey, E. Begoli, R. Gunasekaran, S.-H. Lim, and J. Nutaro, "Big data platforms as a service: Challenges and approach," in *Proceedings of the 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2012.

[2] D. Agrawal, S. S. Das, and A. E. Abbadi, "Big data and cloud computing: Current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT)*, 2011, pp. 530–533.

[3] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big Data Processing in Cloud Computing Environments," in *Proceedings of the 12th International Symposium on Pervasive Systems, Algorithms and Networks (ISPAN)*, 2012, pp. 17–23.

[4] H. Ludwig, "Managing Big Data Effectively - A Cloud Provider and a Cloud Consumer Perspective," in *Proceedings of the 18th IEEE International Conference on Enterprise Distributed Object Computing Conference (EDOC)*, 2014.

[5] M. Adnan, M. Afzal, M. Aslam, R. Jan, and A. M. Martinez-Enriquez, "Minimizing big data problems using cloud computing based on Hadoop architecture," in *Proceedings of the 11th Annual on High-capacity Optical Networks and Emerging/Enabling Technologies (HONET)*, 2014, pp. 99–103.

[6] Z. Zheng, J. Zhu, and M. R. Lyu, "Service-Generated Big Data and Big Data-as-a-Service: An Overview," in *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, 2013, pp. 403–410.

[7] A. Zimmermann, M. Pretz, G. Zimmermann, D. G. Firesmith, I. Petrov, and E. El-Sheikh, "Towards Service-Oriented Enterprise Architectures for Big Data Applications in the Cloud," in *Proceedings of the 17th IEEE International on Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, 2013, pp. 130–135.

[8] Q. Zhang, Z. Chen, A. Lv, L. Zhao, F. Liu, and J. Zou, "A Universal Storage Architecture for Big Data in Cloud Environment," in *Proceedings of the IEEE Green Computing and Communications (GreenCom), Internet of Things (iThings/CPSCom), and Cyber, Physical and Social Computing*, 2013, pp. 476–480.

[9] A. Kashlev and L. S. Lu, "A System Architecture for Running Big Data Workflows in the Cloud," in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2014, pp. 51–58.

[10] J. J. Dai, J. Huang, S. Huang, B. Huang, and Y. Liu, "HiTune: Dataflow-based Performance Analysis for Big Data Cloud," in *Proceedings of the USENIX Annual Technical Conference (USENIXATC)*, 2011, pp. 7–7.

[11] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving Big Data to The Cloud: An Online Cost-Minimizing Approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[12] R. T. Kaushik and K. Nahrstedt, "T*: A data-centric cooling energy costs reduction approach for Big Data analytics cloud," in *Proceedings for the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012, pp. 1–11.

[13] S. Lu, R. M. Li, W. C. Tjhi, K. K. Lee, L. Wang, X. Li, and D. Ma, "A Framework for Cloud-Based Large-Scale Data Analytics and Visualization: Case Study on Multiscale Climate Data."

[14] F. Ye, Z. Wang, F. Zhou, Y. Wang, and Y. Zhou, "Cloud-Based Big Data Mining amp; Analyzing Services Platform Integrating R," in *Proceedings of the International Conference on Advanced Cloud and Big Data (CBD)*, 2013, pp. 147–151.

[15] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Distributed GraphLab: A Framework for Machine Learning in the Cloud," *Proceedings of the VLDB Endowment (PVLDB)*, vol. 5, no. 8, pp. 716–727, 2012.

[16] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J.-M. Hellerstein, "GraphLab: A New Parallel Framework for Machine Learning," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[17] "Cloudera," http://www.cloudera.com.

[18] "Hadoop," http://hadoop.apache.org/.

[19] "Cassandra," http://cassandra.apache.org/.

[20] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[21] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Fast and Interactive Analytics over Hadoop Data with Spark," *The USENIX Magazine*, vol. 37, no. 4, pp. 45–51, 2012.

[22] "Google's big query," https://cloud.google.com/bigquery/.

[23] "Mahout," http://mahout.apache.org/.

[24] "Giraph," http://giraph.apache.org/.

[25] Y. C. Kwon, M. Balazinska, B. Howe, and J. Rolia, "SkewTune: Mitigating Skew in Mapreduce Applications," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2012, pp. 25–36.

[26] Q. H. Vu, T. V. Pham, H. L. Truong, S. Dustdar, and R. Asal, "DEMODS: A Description Model for Data-as-a-Service," in *Proceedings of the 26th Internation Conference on Advanced Information Networking and Applications (AINA)*, 2012, pp. 605–612.

[27] C. A. Ardagna, E. Damiani, R. Asal, and Q. H. Vu, "On the Management of Cloud Non-Functional Properties: The Cloud Transparency Toolkit," in *Proceedings of the 6th International Conference on New Technologies, Mobility and Security (NTMS)*, 2014, pp. 1–4.

[28] "Trusted computing," http://www.trustedcomputinggroup.org/.

[29] "OpenStack," https://www.openstack.org/.