ISSN 2395-695X (Print)
ISSN 2395-695X (Online)

International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

AN INTEGRATED PROTECTION SCHEME FOR BIG DATA AND CLOUD SERVICES

Ms. K.Yasminsharmeli. IInd ME (CSE),
Mr. N. Nasurudeen Ahamed M.E., Assistant Professor
Al-Ameen Engineering College, Erode, Tamilnadu, India
yasminsharmeli@gmail.com

Abstract

Hardware and software resources are provided to the users from the cloud computing environment. Processor, memory and storage spade are shared under hardware resources. Service components are provided in the software resource sharing environment. The software services utilize the data values that are maintained under the hardware service provider environment. Big data services are provided to the users from the cloud environment. Public and private clouds are integrated to construct the cross cloud services.

The big data values are processed over the services provided under the public cloud services. Different service provider provides the service components for the big data process. Service provider selection is a complex task in the big data process. Service execution properties are used in the service selection process. Response time, throughput and cost factors are considered in the service provider selection process. History records are maintained under the service log files. The history records are analyzed to find out the suitable service provider for the big data process. History Record based Service Optimization Method (HireSome-II) scheme is applied for the service provider selection process. Aggregation functions are adapted to summarize the history records. K-means clustering technique is also applied to group up the history records. Privacy is provided for the history records to protect the user data access information.

The big data process and cloud service operations are integrated with History Record based Service Optimization Method (HireSome-II) scheme. Data protection is handled with the integrated protection scheme. History records and big data values are protected with anonymization techniques. The system performs the data classification using the public cloud services and big data from private cloud environment.

1. Introduction

Today huge amount of digital data is being accumulated in many important areas, including e-commerce, social network, finance, health care, education, and environment. It has become increasingly popular to mine such big data in order to gain insights to help business decisions or to provide better personalized, higher quality services. In recent years, a large number of computing frameworks [2], [10] have been developed for big data analysis. Among these frameworks, MapReduce is the most widely used in production because of its simplicity, generality, and maturity. We focus on improving MapReduce in this paper.

Big data is constantly evolving. As new data and updates are being collected, the input data of a big data mining algorithm will gradually change, and the computed results will become

stale and obsolete over time. In many situations, it is desirable to periodically refresh the mining computation in order to keep the mining results up-to-date. For example, the PageRank algorithm computes ranking scores of web pages based on the web graph structure for supporting web search. The web graph structure is constantly evolving; Web pages and hyper-links are created, deleted, and updated. As the underlying web graph evolves, the PageRank ranking results gradually become stale, potentially lowering the quality of web search. It is desirable to refresh the PageRank computation regularly.

2. Related Work

Data privacy preservation has been extensively investigated [5]. We briefly review existing approaches for local-recoding anonymization and privacy models to defense against attribute linkage attacks. In addition, research on scalability issues in existing anonymization approaches is shortly surveyed. Clustering techniques have been leveraged to achieve local recoding anonymization for privacy preservation. Xu et al. studied on the anonymization of local recoding scheme from the utility perspective and put forth a bottom-up greedy approach and the top-down counterpart. The former leverages the agglomerative clustering technique while the latter employs the divisive hierarchical clustering technique, both of which pose constraints on the size of a cluster. Byun et al. formally modeled local recoding anonymization as the k member clustering problem which requires the cluster size should not be less than k in order to achieve k-anonymity, and proposed a simple greedy algorithm to address the problem.

The inconsistency issues of local recoding anonymization in data with hierarchical attributes and proposed KACA (K-Anonymization by Clustering in Attribute hierarchies) algorithms. Aggarwal etal. [11] proposed a set of constant factor approximation algorithms for two clustering based anonymization problems, i.e., r-GATHER and r-CELLULAR CLUSTERING, where cluster centers are published without generalization or suppression. Existing clustering approaches for local-recoding anonymization mainly concentrate on record linkage attacks, specifically under the k-anonymity privacy model, without paying any attention to privacy breaches incurred by sensitive attribute linkage. On the contrary, our research takes both privacy concerns into account. Wong et al. proposed a top-down partitioning approach based on the Mondrian algorithm specialize data sets to achieve (a, k)-anonymity which is able to defend certain attribute linkage attacks. The data utility of the resultant anonymous data is heavily influenced by the choice of splitting attributes and values, while local recoding does not involve such factors. Our approach leverages clustering to accomplish local recoding because it is a natural and effective way to anonymize data sets at a cell level.

To preserve privacy against attribute linkage attacks, a variety of privacy models have been proposed for both categorical and numerical sensitive attributes. l-diversity and its variants [8] require each QI-group to include at least l well represented sensitive values. Note that l-diverse data sets are already l-anonymous. Some privacy models, such as (a, k)-anonymity, extend k-anonymity with the confidence bounding principle that requires the confidence of associating a quasi identifier to a sensitive value to be less than a user-specified threshold. In general, these integrated models are more flexible than l-diversity. Since the models above handle categorical attributes only, they fail to thwart proximity privacy breach in numerical sensitive attributes. As a result, several privacy models such as (k,e) anonymity, variance control, (E,m)-anonymity and t-closeness are put forth. (k,e)-anonymity requires that the difference of the

maximum and minimum sensitive values in a QI-group must be at least e, while the variance control principle demands that the variance of the sensitive values must be not less than a threshold. (E,m)-anonymity, a stronger model, requires that for any sensitive value in a QIgroup, at most 1=m of records can have sensitive values similar to the value, where "determines the similarity.

A stringent privacy model t closeness, which mainly combats data distribution skewness attacks by requiring the distribution of sensitive values in any QI-group should be close to the distribution of the entire data set, incorporates semantics through the kernel smoothing technique to mitigate the proximity breaches to a certain extent. T-closeness is applicable to both categorical and numerical attributes as it only demands a pre-defined distance matrix. But t-closeness is insufficient to protect against proximity attacks. To cope with both categorical and numerical attributes, Wang et al. proposed a general proximity privacy model, namely, (£,δ) - dissimilarity, where determines the similarity threshold, d controls the least number of dissimilar sensitive values for any sensitive value in a QI-group, and k means k-anonymity is integrated. The privacy model we proposed herein can be regarded as an extended form of $(£,\delta)^k$-dissimilarity. Nevertheless, our model differs from it in that multiple sensitive attributes are taken into account and categorical sensitive attributes have semantic proximity in terms of their taxonomy trees.

Scalability issues of anonymization over large-scale data sets have drawn the attention of research communities. LeFevre et al. addressed the scalability problem of multidimensional anonymization scheme via introducing scalable decision trees and sampling techniques. Iwuchukwu et al. proposed an R-tree index-based approach by building a spatial index over data sets, achieving high efficiency. Fung et al. [6] proposed a top-down specialization approach, improving the efficiency of sub-tree anonymization scheme by exploiting a data structure named Taxonomy Indexed PartitionS (TIPS). Our previous work [13] addressed the scalability problem of the subtree scheme in big data scenarios via leveraging MapReduce paradigm. The approaches above aim at either multidimensional scheme or sub-tree scheme, both of which are global recoding, thereby failing to work out for the local recoding scheme investigated herein.

3. Privacy Preserved Big Data Applications

Cloud Computing and big data receives enormous attention internationally due to various business-driven promises and expectations such as lower upfront IT costs, a faster time to market, and opportunities for creating value-add business. As the latest computing paradigm, cloud is characterized by delivering hardware and software resources as virtualized services by which users are free from the burden of acquiring the low level system administration details. Cloud computing promises a scalable infrastructure for processing big data applications such as the analysis of huge amount of medical data. Cloud providers including Amazon Web Services (AWS), Sales force. com, or Google App Engine, give users the options to deploy their application over a network of a nearly infinite resource pool. By leveraging Cloud services to host Web, big data applications can benefit from cloud advantages such as elasticity, pay-per-use, and abundance of resources with practically no capital investment and modest operating cost proportional.

Some big data centers or software services cannot be migrated into a public cloud due to some security and privacy issues. If a web based application covers some public cloud services,

private cloud services and local web services in a hybrid way, cross-cloud collaboration is an ambition for promoting complex web based applications in the form of dynamic alliance for value-add applications. It needs a unique distributed computing model in a network-aware business context. Cross-cloud service composition provides a concrete approach capable for large-scale big data processing. Existing analysis techniques for service composition often mandate every participant service provider to unveil the details of services for network-aware service composition, especially the QoS information of the services. Unfortunately, such an analysis is infeasible when a private cloud or a local host refuses to disclose all its service in detail for privacy or business reasons. In such a scenario, it is a challenge to integrate services from a private cloud or local host with public cloud services such as Amazon EC2 and SQS for building scalable and secure systems in the form of mashups. As the diversity of Cloud services is highly available today, the complexity of potential cross-cloud compositions requires new composition and aggregation models.

Cloud computing environment provides scalable infrastructure for big data applications. Cross clouds are formed with the private cloud data resources and public cloud service components. Cross cloud service composition provides a concrete approach capable for large scale big data processing. Private clouds refuse to disclose all details of their service transaction records. History record based Service optimization method (HireSome-II) is privacy aware cross cloud service composition method. QoS history records are used to estimate the cross cloud service composition plan. k-means algorithm is used as a data filtering tool to select representative history records. HireSome-II reduces the time complexity of cross cloud service composition plan for big data processing. The following drawbacks are identified from the existing system. The following drawbacks are identified from the existing system. Big data processing is not integrated with the system. Security and privacy for big data is not provided. Limited scalability in big data process. Mining operations are not integrated with the system.

4. Resource Scheduling and MapReduce

Businesses today are increasingly reliant on large-scale data analytics to make critical day-to-day business decisions. This shift towards data-driven decision making has fueled the development of MapReduce, a parallel programming model that has become synonymous with largescale, data-intensive computation. In MapReduce, a job is a collection of Map and Reduce tasks that can be scheduled concurrently on multiple machines, resulting in significant reduction in job running time. Many large companies, such as Google, Facebook, and Yahoo!, routinely use MapReduce to process large volumes of data on a daily basis. Consequently, the performance and efficiency of MapReduce frameworks have become critical to the success of today's Internet companies.

A central component to a MapReduce system is its job scheduler. Its role is to create a schedule of Map and Reduce tasks, spanning one or more jobs, that minimizes job completion time and maximizes resource utilization. A schedule with too many concurrently running tasks on a single machine will result in heavy resource contention and long job completion time. Conversely, a schedule with too few concurrently running tasks on a single machine will cause the machine to have poor resource utilization. The job scheduling problem becomes significantly easier to solve if we can assume that all map tasks have homogenous resource requirements in terms of CPU, memory, disk and network bandwidth. Indeed, current MapReduce systems, such

as Hadoop Map- Reduce Version 1:x, make this assumption to simplify the scheduling problem. These systems use a simple slot-based resource allocation scheme, where physical resources on each machine are captured by the number of identical slots that can be assigned to tasks. Unfortunately, in practice, run-time resource consumption varies from task to task and from job to job. Several recent studies have reported that production workloads often have diverse utilization profiles and performance requirements [3]. Failing to consider these job usage characteristics can potentially lead to inefficient job schedules with low resource utilization and long job execution time.

Motivated by this observation, several recent proposals, such as resource aware adaptive scheduling (RAS) [9] and Hadoop MapReduce Version 2 [7], have introduced resourceaware job schedulers to the MapReduce framework. These schedulers specify a fixed size for each task in terms of required resources, thus assuming the run-time resource consumption of the task is stable over its life time. This is not true for many MapReduce jobs. In particular, it has been reported that the execution of each MapReduce task can be divided into multiple phases of data transfer, processing and storage [12]. A phase is a sub-procedure in the task that has a distinct purpose and can be characterized by the uniform resource consumption over its duration. The phases involved in the same task can have different resource demand in terms of CPU, memory, disk and network usage. Scheduling tasks based on fixed resource requirements over their durations will often cause either excessive resource contention by scheduling too many simultaneous tasks on a machine, or low utilization by scheduling too few.

We present PRISM, a Phase and Resource Information-aware Scheduler for MapReduce clusters that performs resource-aware scheduling at the level of task phases. Specifically, we show that for most MapReduce applications, the run-time task resource consumption can vary significantly from phase to phase. By considering the resource demand at the phase level, it is possible for the scheduler to achieve higher degrees of parallelism while avoiding resource contention. We have developed a phase-level scheduling algorithm with the aim of achieving high job performance and resource utilization. Through experiments using a real MapReduce cluster running a wide-range of workloads, we show PRISM delivers up to 18 percent improvement in resource utilization while allowing jobs to complete up to 1:3 faster than current Hadoop schedulers. Even though PRISM is currently designed for Hadoop MapReduce, we believe our solution can be applied to Dryad and other parallel computing frameworks as well.

4.1 Hadoop MapReduce

MapReduce is a parallel computing model for largescale data-intensive computations. A MapReduce job consists of two types of tasks, namely map and reduce tasks. A map task takes as input a key-value block stored in the underlying distributed file system and runs a user-specified map function to generate intermediary key-value output. Subsequently, a reduce task is responsible for collecting and applying a user-specified reduce function on the collected key-value pairs to produce the final output. Currently, the most popular implementation of MapReduce is Apache Hadoop MapReduce [1]. A Hadoop cluster consists of a large number of commodity machines with one node serving as the master and the others acting as slaves. The master node runs a resource manager that is responsible for scheduling tasks on slave nodes. Each slave node runs a local node manager that is responsible for launching and allocating resources for each task. To do so, the task tracker launches a Java Virtual Machine (JVM) that

executes the corresponding map or reduce task. The original Hadoop MapReduce adopts a slot-based resource allocation scheme. The scheduler assigns tasks to each machine based on the number of available slots on that machine. The number of map slots and reduce slots determines respectively the maximum number of map tasks and reduce tasks that can be scheduled on the machine at a given time.

As a Hadoop cluster is usually a multi-user system, many users can simultaneously submit jobs to the cluster. The job scheduling is performed by the resource manager in the master node, which maintains a list of jobs in the system. Each slave node monitors the progress of each running task and available resources on the node, and periodically transmit a heartbeat message to convey this information to the master node. The resource scheduler will use the provided information to make scheduling decisions. Currently, Hadoop MapReduce supports several job schedulers such as the Capacity scheduler and Fair scheduler. These schedulers make job scheduling decisions at task level. They determine which task should be scheduled on which machine at any given time, based on the number of unoccupied slots on each machine. While this simple slot-based allocation scheme is simple and easy to implement, it does not take run time task resource consumption into consideration. As different tasks may have different resource requirements, this simple slotbased resource allocation scheme can lead to resource contention if the scheduler assigns multiple tasks that have high demand for a single resource. Motivated by this observation, Hadoop Yarn [7] enables resource-aware task scheduling in Hadoop MapReduce clusters. While still in alpha version, it offers the ability to specify the size of the task container in terms of CPU and memory usage.

4.2 MapReduce Job Phases

Current Hadoop job schedulers perform task-level scheduling, where tasks are considered as the finest granularity for scheduling. We examine the execution of each task, we can find that a task consists of multiple phases. A map task can be divided into two main phases: map and merge. The input of a Map- Reduce job is stored as data blocks in the Hadoop Distributed File System (HDFS) [4], where data blocks are stored across multiple slave nodes. In the map phase, a mapper fetches a input data block from the Hadoop Distributed File System and applies the user-defined map function on each record. The map function generates records that are serialized and collected into a buffer. When the buffer becomes full, the content of the buffer will be written to the local disk. The mapper executes a merge phase to group the output records based on the intermediary keys, and store the records in multiple files so that each file can be fetched a corresponding reducer. The execution of a reduce task can be divided into three phases: shuffle, sort, and reduce. The reducer fetches the output file from the local storage of each map task and then places it in a storage buffer that can be either in memory or on disk depending on the size of the content. The reducer also launches one or more threads to perform local merge sort in order to reduce the running time of the subsequent sort phase. Once all the map output records have been collected, the sort phase will perform one final sorting procedure to ensure all collected records are in order. Finally, in the reduce phase, records are processed according the user defined reduce function in the sorted order, and the output is written to the HDFS.

Different phases can have different resource consumption characteristics. For instance, the shuffle phase often consumes significant network I/O resources as it requires collecting outputs from all completed map tasks. In contrast, the map and reduce phases mainly process the

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)

International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

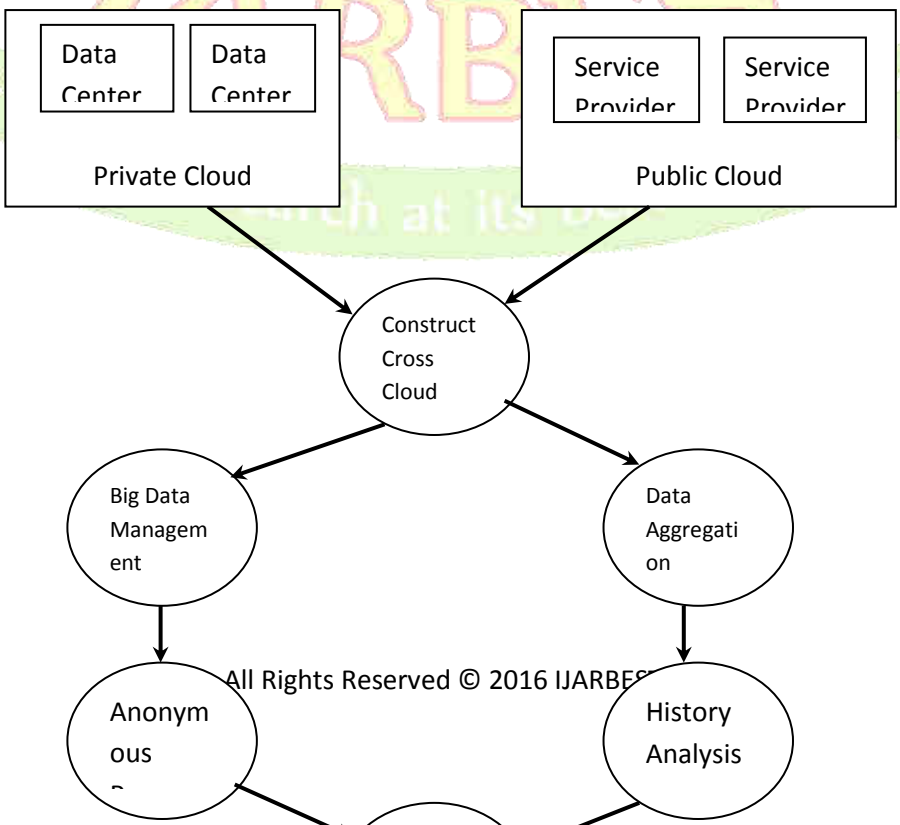
records on local machines, thus they typically demand greater CPU resources than network bandwidth. We provide empirical evidence to the run-time task resource consumption can change significantly across phase boundaries.

5. An Integrated Protection Scheme for Big Data

History record based Service optimization method (HireSome-II) is enhanced to process big data values. Security and privacy is provided for cross cloud service composition based big data processing environment. Privacy preserved map reduce methods are adapted to support high scalability. The HireSome-II scheme is upgraded to support mining operations on big data.

Security and privacy preserved big data processing is performed under the cross cloud environment. Big data classification is carried out with the support of map reduce mechanism. Service composition methods are used to assign resources. The system is divided into six major modules. They are Cross Cloud Construction, Big Data Management, History Analysis, Map Reduce Process, Service Composition and Big Data Classification.

Public and private clouds integrated in the cross cloud construction process. Big data management module is designed to provide big data for the cloud users. Resource sharing logs are analyzed under the history analysis. Task partition operations are performed under the map reduce process. Service provider selection is carried out service composition module. Classification process is carried out under the cross cloud environment. Private and public cloud resources are used in the cross cloud construction process. Big data values are provided under the data centers in private cloud environment. Service components are provided from public cloud environment. Public cloud services utilize the private cloud data values. Larger and complex data collections are referred as big data. Medical data values are represented in big data form. Anonymization techniques are used to protect sensitive attributes. Big data values are distributed with reference to the user request. Service provider manages the access details in the history files. User name, data name, quantity and requested time details are maintained under the data center. History data values are released with privacy protection. Data aggregation is applied on the history data values.



ISSN 2395-695X (Print)
ISSN 2395-695X (Online)

International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

Figure 5.1. Protection Scheme for Big Data

Map reduce techniques are applied to break the tasks. Map reduce operations are partitioned with security and privacy features. Redundancy and fault tolerance are controlled in the system. The data values are also summarized in the map reduce process. HireSome-II scheme is adapted for the service composition process. History records are analyzed with K-means clustering algorithm. Privacy preserved data communication is employed in the system. Public cloud service components are provided to the big data process. Medical data analysis is carried out on the cross cloud environment. Privacy preserved data classification is applied on the medical data values. Public cloud resources are allocated for the classification process. Bayesian algorithm is tuned to perform data classification on parallel and distributed environments.

6. Conclusion

Service composition methods are used to provide resources for big data process. History record based Service optimization method (HireSome-II) is used as privacy ensured service composition method. HireSome-II scheme is enhanced with privacy preserved big data process mechanism. Map reduce techniques are also integrated with the HireSome-II scheme to support high scalability. Security and privacy are provided for the big data and history data values under the cloud environment. Map reduce techniques reduces the computational complexity in big data processing. Data classification is performed on sensitive big data values with cloud resources. Efficient resource sharing is performed under cross cloud environment

References

[1] Hadoop MapReduce distribution [Online]. Available: http:// hadoop.apache.org, 2015.

[2] M. Zaharia, M. Chowdhury, T. Das, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for, in-memory cluster computing,” in Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation, 2012, p. 2.

[3] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in Proc. Eur. Conf. Comput. Syst., 2010.
[4] Hadoop Distributed File System. Available: hadoop. apache.org/docs/hdfs/current/, 2015.
[5] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," ACM Comput. Survey, vol. 42, no. 4, pp. 1–53, 2010.
[6] N. Mohammed, B. Fung and C. K. Lee, "Centralized and distributed anonymization for high-dimensional healthcare data," ACM Trans. Knowl. Discovery Data, vol. 4, no. 4, 2010.
[7] The Next Generation of Apache Hadoop MapReduce [Online]. Available: http://hadoop.apache.org/docs/current/ hadoop-yarn/hadoop-yarn-site/YARN.html, 2015.
[8] Xuyun Zhang, Wanchun Dou, Jian Pei, Surya Nepal, Chi Yang, Chang Liu and Jinjun Chen, "Proximity Aware Local-Recoding Anonymization with MapReduce for Scalable Big Data Privacy Preservation in Cloud", IEEE Transactions On Computers, Vol. 64, No. 8, August 2015
[9] J. Polo, C. Castillo, D. Carrera, M. Steinder and E. Ayguad_e, "Resource-aware adaptive scheduling for MapReduce clusters," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011.
[10] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation," J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.
[11] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu, "Achieving anonymity via clustering," ACM Trans. Algorithms, vol. 6, no. 3, 2010.
[12] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin and S. Babu, "Starfish: A self-tuning system for big data analytics," in Proc. Conf. Innovative Data Syst. Res., 2011.
[13] X. Zhang, C. Liu, S. Nepal, C. Yang, W. Dou and J. Chen, "A hybrid approach for scalable sub-tree anonymization over big data using Mapreduce on cloud," J. Comput. Syst. Sci., 2014.