Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.1091 – 1096

RESEARCH ARTICLE

Minimal Cost Data Sets Storage in the Cloud

Sanoop Jacob Thomas, Dr. P. Balakumar (Ph.D)
Department of Computer Science, Mahendra Institute of Technology, Mallasamudaram,
Tiruchengode, TamilNadu, India
sanoop0007@gmail.com, psbalakumar@gmail.com

Abstract: Scientists are able to deploy computation and data intensive applications using massive computation power and storage capacity of cloud computing systems. These applications can be deployed without infrastructure investment. Cloud can be used to store large application data sets. For cost-effectively storing large volume of generated data sets in clouds, development of storage strategies and bench marking approaches have done based on the pay-as-you-go model. But they are either impractical at run time or inadequately cost-effective for storage.
In this paper, a novel high cost-effective and practical storage strategy is proposed to achieve a minimum cost bench mark. Here in this proposed strategy, it can automatically decide if at run time or not the storing of generated data must be done or not. Local-optimization for the tradeoff between computation and storage is the primary objective of this strategy. Secondary objective is to take into consideration the users' preference on storage. In this paper we manage both original and generated data storage. Also we use data compression for the efficient cost effective data storage in Cloud.

Keywords—Data sets storage, computation-storage tradeoff, computation- and data-intensive applications, cloud computing

I. INTRODUCTION

NOWADAYS, scientific research increasingly relies on IT technologies, where large-scale and high-performance computing systems (e.g., clusters, grids, and supercomputers) are utilized by the communities of researchers to carry out their applications. Scientific applications are usually computation and data intensive, where the generated data sets are often terabytes or even petabytes in size. As reported by Szalay and Gray, science is in an exponential world and the amounts of scientific data will double every year over the next decade and future. Producing scientific data sets involves large number of computation intensive tasks, e.g., scientific workflows, hence taking a long time for execution. These generated data sets contain important intermediate or final results of the computation, and need to be stored as valuable resources. This is because: 1) data can be reused—scientists may need to re-analyze the results or apply new analyzes on the existing data sets; 2) data can be shared—for collaboration, the computation results are shared; hence, the data sets are used by scientists from different institutions . Storing valuable generated data sets can save their regeneration cost when they are reused, not to mention the delay caused by regeneration. However, the large size of the scientific data sets is a big challenge for their storage.

In recent years, cloud computing is emerging as the latest distributed computing paradigm, which provides redundant, inexpensive and scalable resources on demand to system requirements. Meanwhile, cloud computing adopts the pay-as-you-go model, where users are charged according to the usage of cloud services such as computing, storage, and network services like conventional utilities in everyday life (e.g., water, electricity, gas and telephony). Evidently, cloud computing offers a new way for deploying applications. As IaaS is a very popular way to deliver computing resources in the cloud , the heterogeneity of computing systems of one service provider can be well shielded by the virtualization technologies. Hence, users can deploy their applications in unified resources without any infrastructure investment, where excessive processing power and storage can be obtained from commercial cloud service providers. With the pay-as-you-go model, the total application cost in the cloud highly depends on the strategy of storing the application data sets, e.g., storing all the generated application data sets in the cloud may result in a high storage cost, because some data sets may be rarely used but large in size; in contrast, deleting all the generated data sets and regenerating them every time when needed may result in a high computation cost. A good strategy is to find a balance to selectively store appropriate data sets and regenerate the rest when needed; however, current approaches are not highly cost-effective. A minimum cost benchmarking approach for data sets storage has been developed, which can (theoretically) achieve the best tradeoff between computation and storage in the cloud; however, this approach is impractical for runtime storage strategy due to high computation complexity.

In this paper, toward achieving the minimum cost benchmark in a practical manner, we propose a novel local optimization-based runtime strategy for storing the generated application data sets in the cloud. We utilize a Cost Transitive Tournament Shortest Path (CTT-SP)-based algorithm, which was used for static on-demand minimum cost benchmarking of data sets storage in the cloud. We enhance the CTT-SP algorithm by incorporating users' (optional) preferences on storage that can offer users some flexibility. Based on the enhanced CTT-SP algorithm, we propose a runtime local-optimization-based strategy for storing the generated application data sets in the cloud. Theoretical analysis, general random simulations as well as specific case studies demonstrate that this strategy is highly cost-effective (i.e., close to or even the same as the minimum cost benchmark) with very practical computation complexity for runtime deployment.

This paper is a significantly extended version of our conference paper [1]. The extensions are from the following aspects:

1. Incorporation of Original data storage management,
2. Usage Of ETL tool,
3. Enhanced utilization of users' preference Parameters

II. PROPOSED SYSTEM

Toward achieving the minimum cost benchmark in a practical manner, we propose a novel local optimization-based runtime strategy for storing the generated application data sets in the cloud. We utilize a Cost Transitive Tournament Shortest Path (CTT-SP)-based algorithm which was used for static on-demand minimum cost benchmarking of data sets storage in the

Cloud. We enhance the CTT-SP algorithm by incorporating users (optional) preferences on storage that can offer users some flexibility. Based on the enhanced CTT-SP algorithm, we propose a runtime local-optimization-based strategy for storing the generated application data sets in the cloud. Theoretical analysis, general random simulations as well as specific case studies demonstrate that this strategy is highly cost-effective (i.e., close to or even the same as the minimum cost benchmark) with very practical computation complexity for runtime development. We also manage original application dataset storage using ETL tool and data compression which ensure efficient cost effective data storage.

A. *THE MODEL*

In general, there are two types of data stored in the cloud,
Original data and *generated* data:

1. Original data are the data uploaded by users, for example, in scientific applications, they are usually the raw data collected from the devices in the experiments. For these data, users need to decide whether they should be stored or deleted because they cannot be regenerated by the system once deleted.
2. Generated data are the data newly produced in the cloud while the applications run. They are the intermediate or final computation results of the applications, which can be reused in the future. For these data, their storage can be decided by the system because they can be regenerated if their provenance is known. Hence, our data sets storage strategy is applied to the generated data in the cloud that can automatically decide the storage status of generated data sets in applications.

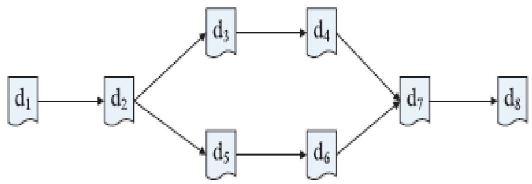

Fig. 1. A simple DDG.

Data Dependency Graph (DDG) is a directed acyclic graph (DAG), which is based on data provenance in scientific applications. All the data sets once generated in the cloud, whether stored or deleted, their references are recorded in DDG. In other words,it depicts the generation relationships of data sets, with which the deleted data sets can be regenerated from their nearest existing preceding data sets.

Cost = Computation + Storage;

where the total cost of the application data sets storage,
Cost, is the sum of Computation, which is the total cost of computation resources used to regenerate data sets, and
Storage, which is the total cost of storage resources used to store the data sets. To utilize the data sets storage cost model, we define the attributes for the data sets in DDG.
Briefly, for data set d_i, its attributes are denoted as: **<xi; yi;fi; vi; provSeti;CostRi>[5],** where:

- **xi** denotes the regeneration cost of data set di from its direct predecessors in the cloud.
- **yi** denotes the cost of storing data set di in the cloud per time unit.
- **fi** is a flag, which denotes the status of whether data set di is stored or deleted in the cloud.
- **vi** denotes the usage frequency, which indicates how often di is used.
- **provSeti** denotes the set of stored provenance that are needed when regenerating data set di. Hence, the regeneration cost of di is

$$genCost(d_i) = x_i + \sum_{\{k \mid d_j \in provSet_i \wedge d_j \to d_k \to d_i\}} x_k. \quad (1)$$

- **CostRi** is di's cost rate, which means the average cost per time unit of data set di in the cloud. The value of CostRi depends on the storage status of di, where

$$CostR_i = \begin{cases} y_i, & f_i = stored \\ genCost(d_i) * v_i, & f_i = deleted. \end{cases} \quad (2)$$

Hence, the total cost rate of storing a DDG is the sum of CostR of all the data sets in it, which is P di2DDG CostRi.

We further define the storage strategy of a DDG as S, where S _ DDG, which means storing the data sets in S in the cloud and deleting the rest. We denote the cost rate of storing a DDG with the storage strategy S as SCR, where

$$SCR = \left(\sum_{d_i \in DDG} CostR_i \right)_S . \qquad (3)$$

B. Enhanced CTT-SP Algorithm6 for Linear DDG Segment

The linear CTT-SP algorithm is enhanced to incorporate users' (optional) preferences that can represent user's preferences and provide users some flexibility in using the storage strategy. The two parameters are denoted as **T** and **λ.**

T is the parameter used to represent users' tolerance on data accessing delay. Users need to inform the cloud service provider about the data sets that they have requirements on their availabilities. For a data set di, which needs regeneration, Ti is the delay time that users can tolerant when they want to access it. Furthermore, T is also related to the requirements of applications. For example, some applications may have fixed time constraints, such as the weather forecast application. In this situation, for some particular data sets, the value for Ti can be set according to the starting time and finishing time (i.e., deadline) of the application. In a word, T is the time constraint of data sets' regeneration. In the storage strategy, the regeneration time of any deleted data set di cannot exceed its Ti. Especially, if Ti is smaller than the generation time of data set di itself (i.e., Ti < $xi=Price_{cpu}$, where $Price_{cpu}$ is the price of computation resources used to regenerate di in the cloud), then we have to store di, no matter how expensive di's storage cost is.

λ is the parameter used to adjust the storage strategy
when users have extra budget on top of the minimum cost benchmark to store more data sets for reducing the average data sets accessing time. Based on users' extra budget, we can calculate a proper value of **λ** [7], which is between 0 and 1. We multiply every data set di's storage cost rate (i.e., yi) by **λ**, and use it to compare with di's regeneration cost rate (i.e., genCost(di)*vi) for deciding its storage status. Hence, more data sets tend to be stored, and literally speaking, data sets will be deleted only when their storage cost rates are (1/ **λ**) times higher than their regeneration cost rates. For example, $\boldsymbol{\lambda}_i$ = 0:8 means that users are willing to store data sets with the storage cost up to 1.25 times higher than the regeneration cost.

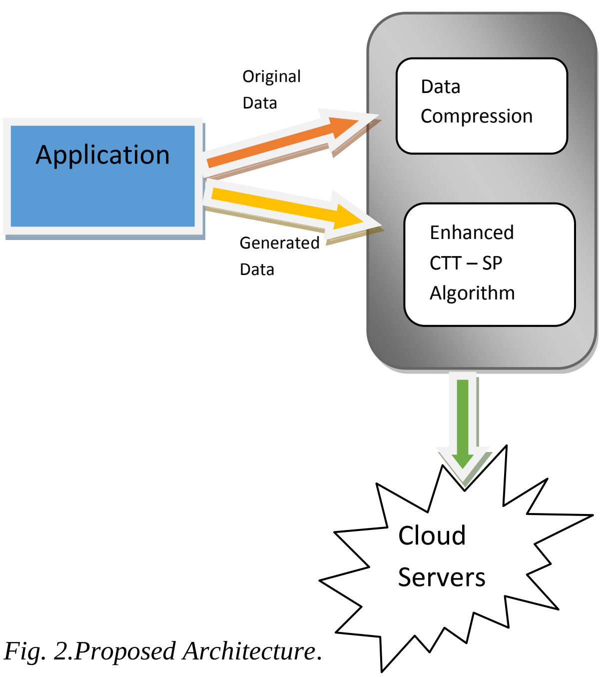

Fig. 2.Proposed Architecture.

ETL Tool: The usage of ETL tool allow the transformation of data into some other form which can save space, avoid redundancy and filter data for storage.

C. System Setup

Application data including both original and generated data need to send to an ETL tool which should be coded to handle data compression and CTT –SP algorithm. SQL Server 2014 supports the data compression feature to help reduce the size of the database. In addition to saving space, data compression can help improve performance of I/O intensive workloads because the data is stored in fewer pages and queries need to read fewer pages from disk. And ETL tool decide the data need to store in the cloud based on the CSS-SP algorithm. Figure 2 shows the architecture.

III. DISCUSSION

Our local-optimization-based storage strategy is close to the minimum cost benchmark, though it may not achieve the minimum cost storage benchmark. In real-world applications, a DDG often has a large number of data sets with complex dependencies. We could directly adopt the minimum cost benchmarking algorithm to derive the minimum cost storage strategy for the whole DDG. Even though the usage of ETL tool in cloud is expensive, it provide an efficient handler for managing the data and thereby reducing the data storage cost in cloud.

IV.CONCLUSION

In this paper, we have adopted a novel runtime local-optimization based storage strategy for both original and generated data sets storage in computation- and data intensive applications in the cloud. This was done with the enhanced linear CTT-SP algorithm by taking into the consideration of users' (optional) preferences with data compression. The usages of ETL tool allow the transformation of data into some other form which can save space, avoid redundancy and filter data for storage. Theoretical analysis, general random simulations, and specific case studies indicate that our strategy is very cost-effective by achieving close to or even the same as the minimum cost benchmark with highly practical runtime efficiency.

REFERENCES

[1] Dong Yuan, Yun Yang, "A Highly Practical Approach toward Achieving Minimum Data Sets Storage Cost in the Cloud".

[2] I. Adams, D.D.E. Long, E.L. Miller, S. Pasupathy, and M.W. Storer,"Maximizing Efficiency by Trading Storage for Computation,"Proc. Workshop Hot Topics in Cloud Computing, 2009.

[3] S. Agarwala, D. Jadav, and L.A. Bathen, "iCostale: Adaptive Cost Optimization for Storage Clouds," Proc. IEEE Int'l Conf. Cloud Computing, pp. 436-443, 2011.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, pp. 50-58, 2010.

[5] R. Bose and J. Frew, "Lineage Retrieval for Scientific Data Processing: A Survey," ACM Computing Surveys, vol. 37, pp. 1- 28, 2005.

[6] A. Burton and A. Treloar, "Publish My Data: A Composition of Services from ANDS and ARCS," Pro. IEEE Fifth Int'l Conf. e- Science, pp. 164-170, 2009.

[7] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the fifth Utility," Future Generation Computer Systems, vol. 25, pp. 599-616, 2009.

[8] R. Campbell, I. Gupta, M. Heath, S.Y. Ko, M. Kozuch, M. Kunze, T. Kwan, K. Lai, H.Y. Lee, M. Lyons, D. Milojicic, D. O'Hallaron, and Y.C. Soh, "Open CirrusTM Cloud Computing Testbed: Federated Data Centers for Open Source Systems and Services Research," Proc. Workshop Hot Topics in Cloud Computing, 2009.

[9] J. Chen and Y. Yang, "Temporal Dependency based Checkpoint Selection for Dynamic Verification of Temporal Constraints in Scientific Workflow Systems," ACM Trans. Software Eng. and Methodology, vol. 20, article 9, 2011.

[10] Y. Cui, H. Wang, and X. Cheng, "Channel Allocation in Wireless Data Center Networks," Proc. IEEE INFOCOM, pp. 1395-1403, 2011.

[11] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An Overview of Workflow System Features and Capabilities," Future Generation Computer Systems, vol. 25, pp. 528-540, 2009.

[12] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The Cost of Doing Science on the Cloud: The Montage Example," Proc.ACM/IEEE Conf. Supercomputing, 2008.

[13] I. Foster, J. Vockler, M. Wilde, and Z. Yong, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," Proc. 14th Int'l Conf. Scientific and Statistical Database Management, pp. 37-46, 2002.

[14] I. Foster, Z. Yong, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," Proc. Grid Computing Environments Workshop, 2008.

