International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 04 | Apr-2018 www.irjet.net p-ISSN: 2395-0072

COST EFFECTIVE WORKFLOW SCHEDULING IN BIGDATA

V.M. PAVITHRA[1], Dr. S.M. JAGATHEESAN[2]

[1]M.Phil Research Scholar, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, TamilNadu, India
[2]Associate Professor, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, TamilNadu, India

---

Abstract - *Computational science workflows have been successfully run on traditional High Performance Computing (HPC) systems like clusters and Grids for many years. Now a day, users are interested to execute their workflow applications in the Cloud to exploit the economic and technical benefits of this new rising technology. The deployment and management of workflows over the current existing heterogeneous and not yet interoperable Cloud providers, is still a challenging task for the workflow developers. The Pointer Gossip Content Addressable Network Montage Framework allows an automatic selection of the goal clouds, a uniform get entry to the clouds, and workflow data management with respect to user Service Level Agreement (SLA) Requirements. Consequently, a number of studies, focusing on different aspects, emerged in the literature. In this comparative review on workflow scheduling algorithm cloud environment is provide solution for the problems. Based on the analysis, the authors also highlight some research directions for future investigation. The previous results offer benefits to users by executing workflows with the expected performance and service quality at lowest cost.*

***Key Words*: Montage framework, Pointer gossip, Content addressable network, Scheduling.**

1. INTRODUCTION

Since 2007, the term cloud has become one of the most buzz words in IT industry. Lots of researchers try to define cloud computing from different application aspects, but there is not a consensus definition on it. Among the many definitions, three widely quoted as follows "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms and services are delivered on demand to external customers over Internet." As an academic representative, foster focuses on several technical features that differentiate cloud computing from other distributed computing paradigms. For example, computing entities are virtualized and delivered as services and these services are dynamically driven by economies of scale.

A style of computing where scalable and elastic IT capabilities are provided as a service to multiple external customers using internet technologies. Garter is an IT consulting company, so it examines qualities of cloud mostly from the point of view of industry. Functional characteristics are emphasized in this definition, such as whether cloud computing is scalable, elastic, service offering and Internet based.

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Compared with other two definitions, U.S. National Institute of Standards and Technology provides a relatively more objective and specific definition, which not only defines cloud concept overall, but also, specifies essential characteristics of cloud computing and delivery and deployment models.

2. REVIEW OF LITERATURE

A. Greenberg [1] has proposed Wide-area transfer of large data sets is still a large challenge despite the deployment of high-bandwidth networks with speeds attainment 100 Gbps. Most users fail to obtain even a fraction of hypothetical speeds promised by these networks. Effective usage of the obtainable network capacity has turn out to be increasingly important for wide-area data movement. They have developed a "data transfer scheduling and optimization system as a Cloud-hosted service", Stork Cloud, which will mitigate the large-scale end-to-end data group bottleneck by efficiently utilizing fundamental networks and effectively preparation and optimizing data transfer. In this paper, the authors present the initial design and prototype performance of Stork Cloud, and demonstrate its efficiency in large dataset transfer across geographically distant storage sites, data centres, and collaborating institutions.

A. Thakar [2] has proposed Today's continuously growing cloud infrastructures provide support for processing ever increasing amounts of scientific data. Cloud resources for computation and storage are spread among globally distributed data centres. Thus, to leverage the full computation power of the clouds, global data processing across multiple sites has to be fully enabled. However, managing data across geographically distributed data centres is not trivial as it involves high and variable latencies among sites which come at a high monetary cost. In this work, the author proposes a uniform data management system for scientific applications running across geographically distributed sites. This project solution is environment aware, as it monitors and models the global cloud infrastructure and offers predictable data handling

performance for transfer cost and time. In terms of efficiency, it provides the applications with the possibility to set tradeoffs between money and time and optimizes the transfer strategy accordingly. The system was validated on Microsoft's Azure cloud across the 6 EU and US data centres. The experiments they are conducted on hundreds of nodes using both synthetic benchmarks and the real life A-Brain application. The results show that the project system is able to model and predict they'll the cloud performance and to leverage this into efficient data dissemination. This project approach reduces the monetary costs and transfer time by up to 3 times.

B. Da Mota[3] has proposed this e-Science Central (e-SC) cloud data processing system and its application to a number e-Science projects. e-SC provides both Software and Platform as a Service (SaaS/PaaS) for scientific data management, analysis and collaboration. It is a portable system and can be deployed on both private (e.g. Eucalyptus) and public clouds (Amazon AWS and Microsoft Windows Azure). The SaaS application allows scientists to upload data, edit and run workflows and share results in the cloud using only a web browser. It is underpinned by a scalable cloud platform consisting of a set of components designed to support the needs of scientists. The platform is exposed to developers so that they can easily upload their own analysis services into the system and make these available to other users. A REST-based API is also provided so that external applications can leverage the platform's functionality, making it easier to build scalable, secure cloud based applications. This paper describes the design of e-SC, its API and its use in three different case studies: spectral data visualisation, medical data capture and analysis, and chemical property prediction.

B. E. A. Calder [4] has proposed Dryad is a general-purpose distributed execution engine for coarse-grain data-parallel applications. A Dryad application combines computational "vertices" with communication "channels" to form a dataflow graph. Dryad runs the application by executing the vertices of the graph on a set of available computers, communicating as appropriate through files, TCP pipes and shared-memory FIFOs. The vertices provided by the application developer are quite simple and are usually written as sequential programs with no thread creation or locking. Concurrency arises from Dryad scheduling vertices to run simultaneously on multiple computers, or on multiple CPU cores within a computer. The application can discover the size and placement of data at run time, and modify the graph as the computation progresses to make efficient use of the available resources. Dryad is designed to scale from powerful multi-core single computers, through small clusters of computers, to data centres with thousands of computers. The Dryad execution engine handles all the difficult problems of creating a large distributed, concurrent application: scheduling the use of computers and their CPUs, recovering from communication or computer failures and transporting data between vertices.

C. Guo [5] has proposed the widely discussed scientific data deluge creates not only a need to computationally scale an application from a local desktop or cluster to a supercomputer, but also the need to cope with variable data loads over time. Cloud computing offers a scalable, economic, on-demand model well matched to the evolving eScience needs. Yet cloud computing creates gaps that must be crossed to move science applications to the cloud. In this article, the authors proposed a Generic Worker framework to deploy and invoke science applications in the Cloud with minimal user effort and predictable, cost effective performance. Their framework is an evolution of Grid computing application factory pattern and addresses the distinct challenges posed by the Cloud such as efficient data transfers to and from the Cloud, and the transient nature of its VMs. The authors present an implementation of the Generic Worker for the Microsoft Azure Cloud and evaluate its use in a genome sequencing application pipeline. The results shows that the user overhead to port and run the application seamlessly across desktop and the cloud can be substantially reduced without significant performance penalties, while providing on-demand scalability.

Brad Calder[6] has proposed Windows Azure Storage (WAS) is a cloud storage system that provides customers the ability to store seemingly limitless amounts of data for any duration of time. WAS customers have access to their data from anywhere at any time and only pay for what they use and store. In WAS, data is stored durably using both local and geographic replication to facilitate disaster recovery. Currently, WAS storage comes in the form of Blobs (files), Tables (structured storage), and Queues (message delivery). The paper described the WAS architecture, global namespace, and data model, as they'll as its resource provisioning, load balancing, and replication systems.

3. METHODOLOGY

Their resource Scheduling approach is based on they find many risks in a PG-CAN on multiple clouds. Still, there are many practical and challenging issues for current multi-cloud environments. The author presents a Montage-based Pointer Gossip Content Addressable Network Montage Frame Work for running workflows in a multi-Cloud environment. The framework allows an automatic selection of the target Clouds, a uniform access to the Clouds, and workflow data management with respect to user Service Level Agreement (SLA) requirements. Following a simulation approach, they evaluated the framework with a real scientific workflow application in different deployment scenarios. The results show that the author Pointer Gossip Content Addressable Network Montage Framework offers benefits to users by executing workflows with the expected performance and service quality at the lowest cost [12].

Those issues include relatively limited cross-cloud network bandwidth and lacking of cloud standards among cloud providers. It relies on the assumption that all qualified nodes must satisfy Inequalities in existing system. All the re-

projection jobs can be added to a pool of tasks and performed by as many processors as are available, exploiting the parallelization inherent in the Montage architecture. The author shows how they can describe the Montage application in terms of an abstract workflow so that a planning tool such as Pegasus can derive an executable workflow that can be run in the Grid environment. The execution of the workflow is performed by the workflow manager DAG and the associated Scheduling graphs. To meet this requirement, the author designs a resource discovery protocol, namely Montage Approach, to find these qualified nodes. The author proposes pg-can, a workflow scheduling system in order to minimize the monetary cost of executing the workflows in IaaS clouds. The main components of pg-can are illustrated in Figure 3. When a user has specified the probabilistic deadline requirement for a workflow, WaaS providers schedule the workflow by choosing the cost-effective instance types for each task in the workflow. The overall functionality of the pg-can optimizations is to determine the suitable instance configuration for each task of a workflow so that the monetary cost is minimized whiles the probabilistic performance requirement is satisfied. The author formulates the optimization process as a search problem, and develops a two-step approach to find the solution efficiently. The instance configurations of the two steps are illustrated in Figure 3. The author first adopt an A⋆ -based instance configuration approach to select the on-demand instance type for each task of the workflow, in order to minimize the monetary cost while satisfying the probabilistic deadline guarantee. Second, starting from the on-demand instance configuration, the author adopt the hybrid instance configuration refinement to consider using hybrid of both on-demand and spot instances for executing tasks in order to further reduce cost. After the two optimization steps, the tasks of the workflow are scheduled to execute on the cloud according to their hybrid instance configuration. At runtime, they maintain a pool of spot instances and on-demand instances, organized in lists according to different instance types. Instance acquisition/release operations are performed in an auto-scaling manner. For the instances that do not have any task and are approaching multiples of full instance hours, they release them and remove them from the pool. The author schedule tasks to instances in the earliest-deadline-first manner. When a task with the deadline residual of zero requests an instance and the task are not consolidated to an existing instance in the pool, they acquire a new instance from the cloud provider, and add it into the pool. In their experiment, for example, Amazon EC2 poses the capacity limitation of 200 instances. If this cap is met, they cannot acquire new instances until some instances are released [9].

They choose Pointer Gossip Content Addressable in this system each node works as a duty node under PG-CAN is responsible for a unique multidimensional range zone randomly selected when it joins the overlay.

4. RESULTS AND DISCUSSIONS

The system have two sets of experiments: firstly calibrating the cloud dynamics from Amazon EC2 as the input of optimization system; secondly running scientific workflows on Amazon EC2 and a cloud simulator with the compared algorithms for evaluation. The author measure the performance of CPU, I/O and network for four frequently used instance types, namely m1.small, m1.medium, m1.large and m1.xlarge. The author find that CPU performance is rather stable, which is consistent with the previous studies. Thus, the experiments focus on the calibration for I/O and network performance. In particular, it repeats the performance measurement on each kind of instance for 10, 000 times (once every minute in 7 days). When an instance has been acquired for a full hour, it will be released and a new instance of the same type will be created to continue the measurement [13].

The measurement results are used to model the probabilistic distributions of I/O and network performance. The author measure both sequential and random I/O performance for local disks. The sequential I/O reads performance is measured with hdparm. The random I/O performance is measured by generating random I/O reads of 512 bytes each. Reads and writes have similar performance results, and they do not distinguish them in this study. The author measure the uploading and downloading bandwidth between different types of instances and Amazon S3. The bandwidth is measured from uploading and downloading a file to/from S3.

The author acquires the four measured types of instances from the dataset using the created AMI. The hourly costs of the on-demand instance for the two instance types are shown in Table.

Those four instances have also been used in the previous studies. As for the instance acquisition time (lag), in this experiment shows that each on demand instance acquisition costs 2 minutes and spot instance acquisition costs 7 minutes on average. This is consistent with the existing studies. The deadline of workflows is an important factor for the candidate space of determining the instance configuration. There are two deadline settings with particular interests: Dmin and Dmax, the expected execution time of all the tasks in the critical path of the workflow all on the m1.xlarge and m1.small instances, respectively. By default, they set the deadline to be Dmin+Dmax

The authors assume there are many workflows submitted by the users to the WaaS provider. In each experiment, they submit 100 jobs of the same workflow structure to the cloud. They assume the job arrival conforms to a Poisson distribution. The parameter λ in the Poisson distribution affects the chance for virtual machine reuse. By default, they set λ as 0.1. As for metrics, they study the average monetary cost and elapsed time for a workflow. All the metrics in the figure 4.1are normalized to those of Static.

Given the probabilistic deadline requirement, they run the compared algorithms multiple times on the cloud and record their monetary cost and execution time.

In this module the dynamic optimal proportional-share resource allocation method, which leverages the proportional share model? The key idea to redistribute available resources among running tasks dynamically, such that these tasks could use up the maximum capacity of each resource in a node, while each task's execution time can be further minimized in a fair way. DOPS consists of two main procedures:

1) Slice handler: It is activated periodically to equally scale the amount of resources allocated to tasks, such that each running task can acquire additional resources proportional to their demand along each resource dimension.

2)Event handler: It is responsible for resource redistribution upon the events of task arrival and completion.

RESULT

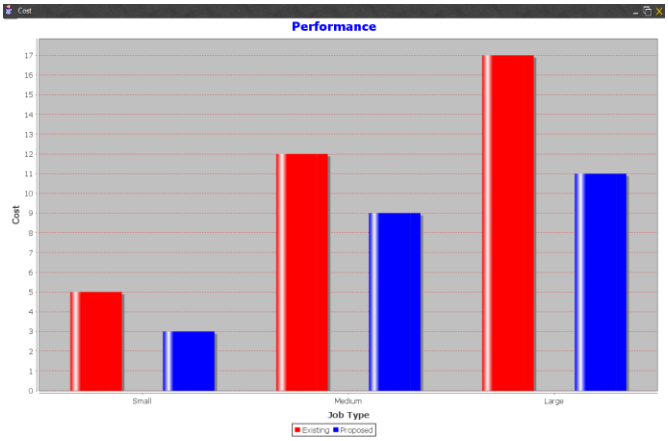

Figure 4.1

5. CONCLUSION AND FUTUREWORK

The proposed approach processing consists of not just one application but some applications combined to form a workflow to reach a certain goal. The existing approach does not work with large data difference and at different speed, but their work will focus applications' execution and resource needs will also vary at runtime. These are called dynamic workflows. One can say that they can just throw more and more resources during runtime [10].

Applied to proficiently schedule computation jobs among processing resources onto the cloud data centre's in a way to reduce implementation time by spreading the jobs on to available resources. The proposed work is to create power efficient clusters in cloud data centre's that shows that cluster creation that helps in decreasing the power consumption as compared to other available algorithm. Their explored the performance variation under different workloads and supply plans. Their also center a couple of data aware scheduling policies and evaluated them with positive results. The future work should focus for this problem of economic power dispatch to obtain a system that is fast and more robust.

REFERECES

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data centre networks," SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 68–73, Dec. 2008.

[2] A. Thakar and A. Szalay, "Migrating a (Large) Science Database to the Cloud", 1st Workshop on Scientific Cloud Computing, June 2010.

[3] B. Da Mota, R. Tudoran, A. Costan, G. Varoquaux, G. Brasche, P. Conrod, H. Lemaitre, T. Paus, M. Rietschel, V. Frouin, J.-B. Poline, G. Antoniu, and B. Thirion, "Generic machine learning pattern for neuroimaging-genetic studies in the cloud," Frontiers in Neuroinformatics, vol. 8, no. 31, 2014.

[4] B. E. A. Calder, "Windows azure storage: a highly available cloud storage service with strong consistency," in Proceedings of the Ttheynty-Third ACM Symposium on Operating Systems Principles, ser. SOSP '11, 2011, pp. 143–157.

[5] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Luz. Dcell: A scalable and fault-tolerant network structure for data centers. In SIGCOMM, 2008.

[6] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley, "Data centre networking with multipath tcp," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ser. Hotnets-IX, 2010, pp. 10:1–10:6.

[7] E. S. Ogasawara, J. Dias, V. Silva, F. S. Chirigati, D. de Oliveira, F. Porto, P. Valduriez, and M. Mattoso, "Chiron: a parallel engine for algebraic scientific workflows," Concurrency and Computation: Practice and Experience, vol. 25, no. 16, pp. 2327–2341, 2013. [Online]. Available: http://dx.doi.org/10.1002/cpe.3032.

[8] E. Yildirim, J. Kim, and T. Kosar, Optimizing the sample size for a cloud-hosted data scheduling service. In Proc. of CCSA Workshop (2012).

[9] G. Khanna, U. Catalyurek, T. Kurc, R. Kettimuthu, P. Sadayappan, and J. Saltz, "A dynamic scheduling approach for coordinated wide-area data transfers using gridftp," in Parallel and Distributed Processing, 2008. IPDPS 2008., 2008, pp. 1–12.

[10] H.Hiden, S. Woodman, P.Watson, and J.Caa, "Developing cloud applications using the e-science central platform." In Proceedings of Royal Society A, 2012.

[11] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[12] J. Yu, R. Buyya, and R. Kotagiri, (2008) Metaheuristics for Scheduling in Distributed Computing Environments. In Workflow Scheduling Algorithms for Grid Computing, pp. 173–214. Springer, Berlin, Germany.

[13] K. Jackson, L. Ramakrishnan, K. Runge, R. Thomas, "Seeking Supernovae in the Clouds: A Performance Study", 1st Workshop on Scientific Cloud Computing, June 2010.