Cost Effective Resource Allocation for Applications with Deadline Constraints Using CloudSim

G. Supriya, N. Subha Lakshmi, V. Vigneshkumar and C. Sagana

Abstract--- Cloud computing has emerged as a mainstream paradigm for hosting various types of applications by supporting easy-to-use computing services. They are particularly relevant for applications requiring large volumes of computing power exceeding the computational capacity However, the use of hybrid clouds introduces the challenge of how much and when public cloud resources should be added to the pool of resources and especially when it is necessary to support quality of service requirements of applications with deadline constraints. These resource provisioning decisions are far from trivial if scheduling involves data-intensive applications using voluminous amounts of data. Issues such as the impact of network latency, bandwidth constraints, and location of data must be taken into account in order to minimize the execution cost while meeting the deadline for such applications. In this paper, we propose a new resource provisioning algorithm to support the deadline requirements of data-intensive applications in hybrid cloud environments. To evaluate our proposed algorithm, we implement it in the CloudSim which proves our resource provisioning algorithm is able to more efficiently allocate resources in cost effective manner compared to existing methods.

I. INTRODUCTION

Data-intensive applications involving the analysis of large datasets have become increasingly important as many areas of science and business are facing thousand-fold increases in data volumes [1]. The explosive growth of data is mainly driven by the rapid expansion of the Internet, smart cities, social networks, e-commerce, and widespread usage of high-throughput instruments, sensor networks, Internet of Things devices, accelerators, and supercomputers. This expansion forms a voluminous amount of structured and unstructured data, known as big data that needs to be processed to be useful [1]. The ability to analyze and process such large quantities of data has become an important and challenging mission for many fields. Cloud computing [2] platforms are becoming one of the most preferred ways of hosting data-intensive applications. Challenges posed by big data can be overcome with the aid of cloud computing services offering the illusion of an infinite pool of highly reliable, scalable, and flexible computing, storage, and network resources. However, in many cases, data isavailable in local IT infrastructure with limited processing capacity. Therefore, it is not time or cost effective to transfer the whole dataset to clouds to be processed. To tackle this issue, the cloud bursting model can be used in which an application runs in a private infrastructure and bursts onto a public cloud when more resources are required. This model has found broad acceptance due to its benefits such as cost reduction and dealing with issues related to the location of sensitive data [3]. To achieve the vision of cloud bursting, hybrid cloud middleware is required to acquire and release resources from both local infrastructures and external cloud providers in a seamless fashion [4]. It is essential for such hybrid cloud middleware to make efficient decisions regarding the workloads that must be outsourced to the public cloud based

G. Supriya, Student, Department of CSE, Kongu Engineering College, Erode, Tamil Nadu.
N. Subha Lakshmi, Student, Department of CSE, Kongu Engineering College, Erode, Tamil Nadu.
V. Vigneshkumar, Student, Department of CSE, Kongu Engineering College, Erode, Tamil Nadu.
C. Sagana, Assistant Professor, Department of CSE, Kongu Engineering College, Erode, Tamil Nadu.

on the timing and number of externally provisioned resources to meet deadline constraints of applications. In such a setting, however, building a middleware that jointly minimizes cost and meets the deadline for applications is far from trivial [5]. There is a large body of literature aimed at cost and execution time minimization of running computational tasks in hybrid cloud environments. These studies mostly overlook aspects such as data locality, the impact of network bandwidth constraints, and data transfer time which significantly affect the time and cost performance of the scheduling. This is exacerbated for data intensive applications where the data transfer time to the external cloud is often comparable to the computational time. In this paper, one of our main goals is to take these aspects into consideration for scheduling and resource provisioning of deadline-driven data intensive applications in hybrid cloud environments.

II. PROBLEM DOMAIN

One of the main challenges for efficient scaling of applications is the location of the data relative to the available computational resources [6]. Co-locating data and computation is evidently ideal in terms of performance especially for data-intensive applications. However, this is not always feasible for various reasons. For example, data might be located in the storage nodes of the user's local organizational infrastructure (e.g., a cluster or desktop grid) with limited or overloaded computational resources and the user facing deadline constraints may prefer to leverage on-demand computing resources from a public cloud provider to reduce the execution time of the application. In the above particular scenario, it may not be ideal for the user to move the entire dataset to the cloud as the data transfer time, due to the data size and network bandwidth, might dominate over the performance gain resulting from utilizing external CPUs. Moving data to distant computational resources, in particular for big data and data intensive applications, to get access to more CPUs is often inefficient and can become the bottleneck in many cases.

III. CLOUDSIM AND RESOURCE ALLOCATION

CloudSim is a new, generalized, and extensible simulation framework that allows seamless modelling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. By using CloudSim, researchers and industry-based developers can test the performance of a newly developed application service in a controlled and easy to set-up environment.

A. Resource Allocation

Resource Allocation means utilizing and allocating scarce resources within the limit of cloud environment so as to meet the needs of the cloud application. Clouds contain the virtualization layer that acts as an execution, management, and hosting environment for application services. Hence, traditional application provisioning models that assign individual application elements to computing nodes do not accurately represent the computational abstraction, which is commonly associated with Cloud resources. For example, consider cloud host that has single processing core. There is a requirement of concurrently instantiating two VMs on that host. Although in practice VMs are contextually isolated, still they need to share the processing cores and system bus. Hence, the amount of hardware resources available to each VM is constrained by the total processing power and system bandwidth available within the host. This critical factor must be considered during the VM provisioning process, to avoid creation of a VM that demands more processing power than is available within the host. In order to allow simulation of different provisioning policies under varying levels of performance isolation, CloudSim supports VM provisioning at two levels: first, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM. At the VM level, the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine.

IV. Deadline Driven Data-aware Resource Provisioning Algorithm

A private cloud with a limited number of resources in the local infrastructure is available for execution of a data intensive Bag-of-Tasks application. Since the number of tasks that can be running concurrently on the private cloud is limited, to meet the deadline requirement of the application, extra resources from the public cloud need to be acquired to scale out available resources. The public cloud provider is able to full fill all requests and thus has an infinite number of resources available from the user's perspective. The network bandwidth available between the private and public cloud is limited and can impose a significant amount of data transfer time for each task running on the public cloud resource. The application's workload consists of a number of trivially parallel tasks, each requiring specific input data files located in the local infrastructure. The Data-aware algorithm takes into account the available bandwidth and the data size associated with each task and calculates the number of extra resources required to meet the deadline constraints of the application. The algorithm is executed when any of the following conditions are observed:

1. Task from the application is queued
2. Execution of a task completes.

The Data-aware algorithm checks if the currently available resources are sufficient for the completion of the application tasks within the given deadline based on estimation of the average runtime of tasks on the private resources. The algorithm first updates remaining time based on the left time to the deadline. Then it computes the number of tasks that can be completed on the private resources within the left time to the deadline. Then, it calculates the number of remaining tasks. These remaining tasks must be scheduled on the dynamic resources. Since the execution of tasks on resources from the public cloud requires transferring data from the local storage to the allocated VMs, the algorithm first calculates the total transfer time for the tasks that should be executed on the public cloud resources.

<u>Data-aware Provisioning Algorithm</u>

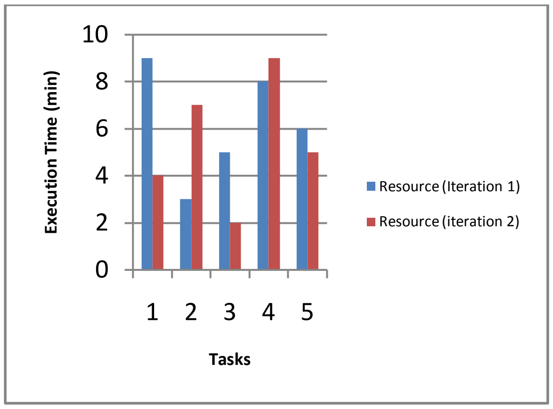
```
1. privateCores ← private cores available for the application;
2. avgTaskRuntime ← Average task runtime on a prviate core;
3. timeRemaining ← Time to application deadline;
4. totalTasks ← Total number of tasks in the application;
5. tasksCompeleted ← Total number of tasks compeleted so far;
6. startupTime ← Startup time of a resource (VM);
7. tasksInPrivate ← ⌊timeRemaining
avgTaskRuntime × privateCores⌋;
8. tasksRemaining = (totalTasks − tasksCompeleted − tasksInPrivate)+;
9. totalTransferTime ← tasksRemaining × taskInputDataSize
upBandwidth ;
10.  actualTimeRemaining ← (timeRemaining −startupTime−totalTransferTime)+;
11. avgTaskRuntimeOnPublic ← Average task runtime on public core;
12. provisionedCores ← Current daynamically prorvisioned cores;
13. totalExecutionTime ← tasksRemaining × avgTaskRuntimeOnPublic;
14. tasksPerCore ← ⌊actualTimeRemaining
avgTaskRuntimeOnPublic ⌋;
15. if tasksPerCore <1 then
16. if toGrow then
17. totalCoresRequired ← provisionedCores;
18. else
19. totalCoresRequired ← provisionedCores − 1
20.  end if
21. else
22. totalCoresRequired ← ⌈totalExecutionTime
tasksPerCore×avgTaskRuntimeOnPublic ⌉;
23. end if
24. extraResources ← ⌈totalCoresRequired−provisionedCores
numberofCoresPerResource ⌉;
```

The actual remaining time which can be effectively used for the execution of the tasks on dynamic resources is computed by subtracting the total transfer time and the start-up time of resources from the remaining time to the deadline. The average task runtime is only calculated based on the actual time tasks are being executed on the public cloud resources and does not include any associated data transfer time, i.e., it is the time period from when the task starts execution on the compute node (after all input data is available) up to the moment its execution is over. The time required to execute all remaining task on a single public CPU core is calculated. The actual time remaining is divided by the average runtime of tasks on public cloud resources, the number of tasks each CPU core on the public resources can execute is calculated. If the number of tasks can be executed on each core in the public resources is at least one, then the total number of required cores can be estimated by dividing total execution time by the result of tasks per core multiplied by the average runtime of the task. Otherwise, there is not enough time for allocating new resources and the algorithm sets the number of required CPU cores to that of the already provisioned cores in the pool. Extra resources that must be added to or removed from the pool by the ratio of extra required cores to the number of cores per resource.

V. PERFORMANCE METRICS

Execution Time

Prediction of execution time fast and accurately not only can help users to schedule jobs smarter, but also maximize the throughput and minimize the resource consumption of cloud platform. Execution time refers to the total execution time of the application. Two samples were taken for Resource 1 and the execution times were compared. It yields minimal execution time in both the samples.



Task Remaining

Task Remaining is the difference between total task, task completed and task in private. Total task evaluates the total number of task available. Task completed involves the number of task that has been completed. Task in private involves the number of task in the private cores.

Actual Time Remaining

Actual time remaining is the difference between the time remaining, start up time and total transfer time gives the actual time remaining.

Experimental Results

The preliminary experiment, estimates that the expected execution time of the application, which allows us to impose a deadline triggering the resource provisioning in other experiments. The key reason is that these algorithms rely on only a single variable for measuring average runtime of tasks to allocate dynamic resources that is significantly different in our scenario for private and public cloud resources. This difference is due to the dissimilarity of the data transfer time for the tasks executed on the private and public cloud resources which is dependent on the difference in the available bandwidth for each case. We conclude that our new algorithm is able to meet strict application deadlines by taking into account the start-up time of the VMs and data transfer time.

VI. CONCLUSION AND FUTURE WORK

This paper presents a provisioning algorithm for scheduling deadline-constrained data-intensive applications while taking into account aspects such as data transfer time, the location of data, and the network bandwidth. This work builds upon previously proposed provisioning algorithms for the CloudSim platform for developing and deploying scalable applications on the cloud. In this work, we propose a provisioning algorithm that computes the extra resources needed to complete application tasks within deadlines by considering aspects such as data locality, start up time of public cloud resources, network bandwidth, and data transfer time. We demonstrate that our proposed algorithm is able to meet strict deadlines for a sample data-intensive application while minimizing cost and the total number launched instances compared to other existing algorithms.

Contrary to other algorithms, the proposed algorithm measures the average runtime of tasks on public cloud resources as a separate variable and takes the data transfer time calculated based on the available bandwidth into account.

The future work is to extend our proposed algorithm for workflows with data dependencies. This is a complex problem and the major challenge in the design of such algorithms in addition to consideration of data transfer time and data locality is how to devise the task grouping and task assignment techniques that minimize inter cloud communications.

REFERENCES

[1] C.P. Chen and C.Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data", Inform. Sci.,Vol. 275, Pp. 314–347, 2014.

[2] 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 5th utility", Future Gener. Comput. Syst., Vol. 25, No. 6, Pp. 599–616, 2009.

[3] X. Xu and X. Zhao, "A framework for privacy-aware computing on hybrid cloudswith mixed-sensitivity data", Proceedings of the IEEE International Symposium on Big Data Security on Cloud, Pp. 1344–1349, 2015.

[4] R.N. Calheiros, C. Vecchiola, D. Karunamoorthy and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds", Future Gener. Comput. Syst., Vol. 28, No. 6, Pp. 861–870, 2012.

[5] R.V. den Bossche, K. Vanmechelen and J. Broeckhove, "Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds", Future Gener. Comput. Syst., Vol. 29, No. 4, Pp. 973–985, 2013.

[6] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared", Proceedings of Grid Computing Environments Workshop, Pp. 1–10, 2008.

[7] F.A. da Silva and H. Senger, "Scalability limits of bag-of-tasks applications running on hierarchical platforms", J. Parallel Distrib. Comput, Vol. 71, No. 6, Pp. 788–801, 2011.

[8] R.O. Sinnott, C. Bayliss, A. Bromage, G. Galang, G. Grazioli, P. Greenwood, A. Macaulay, L. Morandini, G. Nogoorani, M. Nino-Ruiz, M. Tomko, C. Pettit, M. Sarwar, R. Stimson, W. Voorsluys and I. Widjaja, "The Australia urban research gateway", Concurr. Comput.: Pract. Exper, Vol. 27, No.2, Pp. 358–375, 2015.

[9] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud", Journal of Network and Computer Applications, Vol. 82, Pp.93-113, 2017.