ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

SLA-aware and Cost-aware Provisioning and Scheduling of Cloud Resources across Multiple Data centres

Ayesha S

Student, Department of computer Science and Engineering, M.S. Ramaiah Institute of Technology (MSRIT), Bangalore, Karnataka, India[1]

Abstract: The tremendous growth in data has immensely impacted organizations. Their infrastructure and traditional data management systems are unable to handle Big Data. They have to either invest heavily on their infrastructure or move their Big Data analytics to Cloud where they can benefit from both on-demand scalability and contemporary data management techniques. However, to make Cloud hosted Big Data analytics available to wider range of enterprises, we have to carefully capture their preferences in terms of budget and service level objectives. Therefore, in this study we propose SLA driven resource provisioning and scheduling in multiple data centre environment. The user requests in terms of SLA (deadline and budget) are captured at an entry point from where user request user information is sent to cloud provider. The cloud provider receives SLA constraints and user's job details, checks all the data centres for availability of resources and decide the data centre at which the user application can be deployed without violating the SLA and budget constraints. Further, a pruned tree based scheduling algorithm is used to provision cloud resources and schedule the tasks.

Keywords- Big data, Cloud Computing, Service level Agreement, multiple data centres, deadline, budget, scheduling, pruned tree

1. INTRODUCTION

Big data gives new opportunities for enterprises although constitutes series of challenges. The key issues are how to analyse Big Data and convert it to information and knowledge that possess business value. However, this conversion is not economically viable for small to medium enterprises (SMEs) in a traditional infrastructure setting. The majority of SMEs are constantly looking for cutting edge technologies and solutions to accomplish objectives of the company more efficiently and at the minimum cost.

Cloud computing[1] has led to a paradigm shift where enterprises, rather than maintaining their own infrastructure, started to outsource their IT and computational needs to on-demand cloud providers. Clouds are typically large scale virtualized data centres hosting thousands of servers. While there are several advantages of these virtualized infrastructures such as on-demand scalability of resources, and pay-as-you-go model there are still issues which prevent their widespread adoption in clouds. In particular, for a commercial success of this computing paradigm, the cloud data centres need to provide a better and strict Quality of Service (QoS) guarantees. These guarantees which are documented in the form of Service Level Agreement (SLA) are crucial, since only then the customers can be confident in outsourcing their jobs to clouds.

For organizations to adopt cloud hosted big data analytics, we have to carefully consider their preferences in terms of budget and services level objectives through

provisioning and scheduling the cloud resources. There are many SLA-based Big Data computing and MapReduce scheduling studies in the Cloud context, however only few consider to provision Cloud resources dynamically. Most studies have resources pre-provisioned (static) on a private Cloud, which form a virtual cluster. We argue that Cloud resources should be provisioned dynamically and on-demand based on the application workload and the size of the data. This introduces new challenges, namely: a) how many and which type of cloud resources to provision; b) which data centre to select within a cloud provider for a given request with budget and deadline constraints.

Our major contributions are summarized as follows: 1) a model for SLA-based resource provisioning and tasks scheduling for Big data processing in multiple data centres cloud environment. 2) an SLA based cost minimization algorithm to provision cloud resources and schedule the tasks.

2. RELATED WORK

Cloud User leases resources from the Cloud Provider via an IaaS interface. In the IaaS context, the Cloud Provider will seek to satisfy SLAs it has agreed with Cloud Users regarding the provision of virtual infrastructure via its data centre resources. An SLA[2] is a formal agreement between the Cloud Provider and the Cloud User, defining in quantitative terms the functional and non-functional aspects of the service being offered. SLAs encompass aspects like service availability, service performance, security and privacy, data access, problem resolution, change management and dispute mediation. Service Level Objectives (SLOs) pertaining to availability (e.g., the service is available for 99.99 % of the time within a given

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

year) and performance (e.g., the maximum query response time is 10 ms) are directly related to resource management. Depending on the specifics of an SLA the satisfaction of a given SLO may be
viewed as a constraint or as an objective SLOs should be satisfied, to the degree possible given other constraints and objectives. The Cloud Provider may offer different service levels to its customers and may choose to prioritize access to resources to different customer groups depending on the nature of their SLAs.

Most existing schedulers and frameworks do not consider budget as a constraint although they help in efficiently running a MapReduce job in Cloud environments. In addition, the majority of the implementations are Hadoop-based and use Hadoop default scheduling algorithms to schedule jobs on Clouds. Nevertheless, Hadoop scheduling algorithms were designed for clusters of homogeneous machines, which is not applicable for Cloud heterogeneous resources.

Palden Lama, Xiaobo Zhou[3] propose and develop AROMA, a system that automates the allocation of heterogeneous Cloud resources and configuration of Hadoop parameters for achieving quality of service goals while minimizing the incurred cost. It addresses the significant challenge of provisioning ad-hoc jobs that have performance deadlines in Clouds through a novel two-phase machine learning and optimization framework. By this Customers may suffer from a lack of performance guarantee and increased cost of leasing the cloud resources.

Wei Zhang propose MIMP[4] Deadline and Interference Aware Scheduling of Hadoop Virtual Machines which consists of two schedulers: one in the virtualization layer designed to minimize interference on high priority interactive services, and one in the Hadoop framework that helps batch processing jobs meet their own performance deadlines. This approach uses performance models to match Hadoop tasks to the servers that will benefit them the most, and deadline-aware scheduling to effectively order incoming jobs. The combination of these schedulers allows data centre administrators to safely mix resource intensive Hadoop jobs with latency sensitive web applications, and still achieve predictable performance for both.

Min Li propose, CAM[5], a cloud platform that provides an innovative resource scheduler particularly designed for hosting MapReduce applications in the cloud. CAM reconciles both data and VM resource allocation with a variety of competing constraints, such as storage utilization, changing CPU load and network link capacities. CAM uses a flow-network based algorithm that is able to optimize MapReduce performance under the specified constraints.

Foued Jrad, present a multi-dimensional resource allocation scheme[6] to automate the deployment of data-intensive large scale applications in Multi-Cloud environments. The scheme applies a two level approach in which the target Clouds are matched with respect to the Service Level Agreement (SLA) requirements and user payment at first and then the application workloads are distributed to the selected Clouds using a data locality driven scheduling policy.

In our study, we considering SLA and cost as main factors such that cloud provider satisfy the user request and provision and schedule the cloud resources for Big data application.

3. ARCHITECTURE

The proposed architecture is depicted in Fig 1.
To pave the way for organizations to adopt Cloud-hosted Big Data analytics, we have to carefully consider their preferences in terms of budget and service level objectives.
The major components are :
User Request: Enterprises sending their application and SLA constraints.
Entry point: Capturing all the user preferences in terms of application length, SLA objectives (budget and deadline) and send this information to the cloud provider.
Cloud provider: Receives all the data from the user.
Admission control: in each data centre check whether it can satisfy the user request without violating SLA constraints.
VMscheduler: provision the cloud resources and schedule the task.

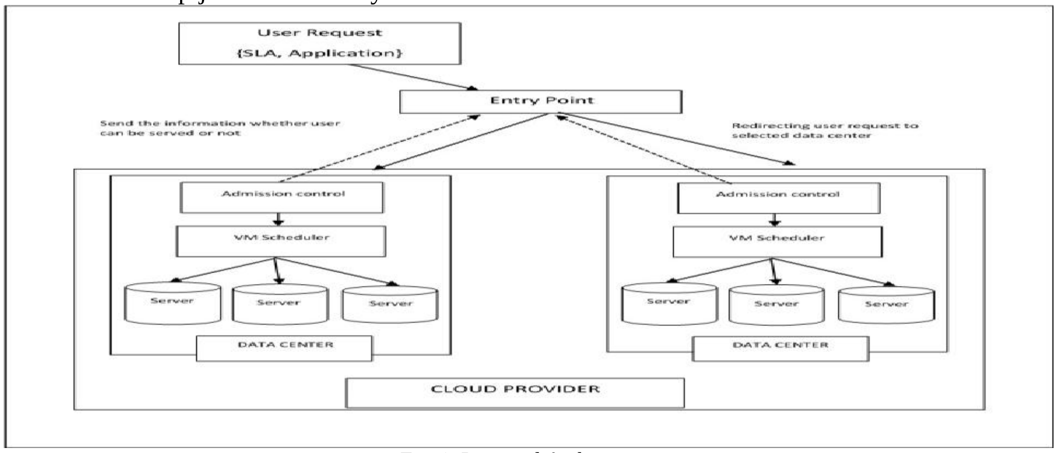

Fig. 1. Proposed Architecture

4. SYSTEM MODEL

4.1 User Request Model

User request (UR) model (shown in Equation 1) consists of: SLA objectives (SLO) and Job (J). Each SLO includes: Budget (B) and Deadline (DL).

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 5, May 2015

$UR = \{SLO, J\}$ (1)
$SLO = \{B,DL\}$ (2)

An application/job (J) (shown in Equation 3) consists of a D_{Source}, a set of tasks (Task). Each Task consists of Input Data Size (ID_{Size}), the required million CPU instructions (MI), and the size of the output data (OP_{Size}) . Each Task has the required instructions in million instructions (MI).

$J = \{ D_{Source}, [Task]\}$ (3)
$Task = \{IDSize,MI,OPSize\}$ (4)

4.2 Cloud provider Model

The Cloud provider model (shown in Equation 5) consist of: a set of data centers (D_{center}), a set of data sources (D_{Source}), a matrix (MT_{VM}) that shows throughputs between a virtual machine type and other types of all data centers in megabits per second (mbps), and a matrix (MTD_{Source}) that shows throughputs between a data source and a virtual machine type in megabits per second (mbps). Each D_{Source} has a cost for transferring the data from it (CTD_{Source}) per terabyte. Each data center has a set of virtual machine types (VM). Each virtual machine type has: cost of leasing (CL_{VM}) per hour, the cost of transferring the data from it(CT_{VM}) per terabyte, the performance of the virtual machine (MIPS) in million instructions per second.

$CP = \{ [D_{Source}],[D_{center}],[MT_{VM}],[MTD_{Source}]\}$ (5)
$D_{center} = \{[VM]\}$ (6)
$VM = \{LC_{VM},CT_{VM},MIPS\}$ (8)
$D_{Source} = \{CTD_{Source}\}$ (9)

The objective is to satisfy the SLA requirements of the user while minimizing the total cost. The total cost of running the job is denoted as $Total_C$. Given that total of n machines and m data sources are used, the $Total_C$ can be computed as shown in Equation 10 where LP is the leasing period for a virtual machine and $TotalD_{VM}$ and $TotalDD_{Source}$ are total data in terabyte transferred from a machine and a data source respectively. The total execution time for running the job is denoted as: Ex_{Time}, which is the total time of executing a task and transferring the data in and out from data centers.

$$Total_C = \sum^{n}_{i=1} LC_{VM\,i} * LP_{VM\,i} + CT_{VMi} * TotalD_{VMi} + \sum^{m}_{j=1} CTD_{Source\,j} * TotalDD_{Source\,j} \qquad (10)$$

the SLA requirements consist of Budget (B) and Deadline (DL). As a result, algorithms'objective is given as:

$$\text{Min } (Total_C) \text{ Subject to } Total_C < B \text{ and } Ex_{Time} < DL \qquad (11)$$

TABLE I
IMPORTANT NOTATIONS

D_{center}	Set of Data centers
D_{Source}	Set of data sources
MT_{VM}	A matrix that shows throughputs between a virtual machine type and other types of all data centers in megabits per second (mbps)
MTD_{Source}	A matrix that shows throughputs between a data source and a virtual machine type in megabits per second (mbps)
CTD_{Source}	A cost for transferring the data from Data source per terabyte
VM	virtual machine types
LC_{VM}	Cost of leasing VM per hour
CT_{VM}	Cost of transferring the data from VM per terabyte
$Total_C$	The total cost of running the job
$TotalD_{VM}$	Total data in terabyte transferred from a machine and a data Source
$TotalDD_{Source}$	Total data in terabyte transferred from a data Source
Ex_{Time}	Total execution time for running the job

5.ALGORITHM

The provisioning and scheduling problem is a multidimensional knapsack problem that was shown to be NP-complete. To tackle the problem, one may consider a greedy algorithm. However, it cannot be directly adopted as it is not capable of satisfying the budget constraint. In addition, it is important to emphasize that the optimization algorithms are required to both determine what is the best set of Cloud resources to provision and also how to schedule tasks on those resources.

5.1 Standard Trees

Fig 2 illustrates an example of a constructed Standard Tree for two tasks (Task#1 and Task#2) and two VM types (L for Large VM and XL for X.Large VM). The depth (levels) of the tree is the total number of tasks (Task#1 and Task#2 in Fig 2), while the breadth (branches) is the total number of tasks multiplied by the number of VM instances. Therefore, in each level of the tree, each node is a VM type that can be selected for a task execution. The number of branches of all nodes are the same, which is the number of tasks multiplied by the number of VM instances. The reason why the Standard Tree is constructed in this way is that we need to cover all of the possibilities for scheduling tasks. For example, in Fig 2 we have two tasks and two type of VMs. As a result, we have four branches under each node. This can be considered as a disadvantage in the Standard Tree as it grows exponentially as number of tasks and VM instances increases.Standard Tree is sorted by cost ascending from left to right, so it will start consolidating all tasks into the cheapest virtual machine, which is the most left leaf solution. If that solution does not satisfy the deadline constraint it schedule one of the tasks to the next cheapest virtual machine. However, if the budget is violated the traversing process will stop with no solution found. An example of a solution set/vector is the the third leaf node in Fig 2 (v ={L1,XL1}), which means that the first task willbe scheduled in a Large VM L1, and the second task in an X.Large VM XL1.

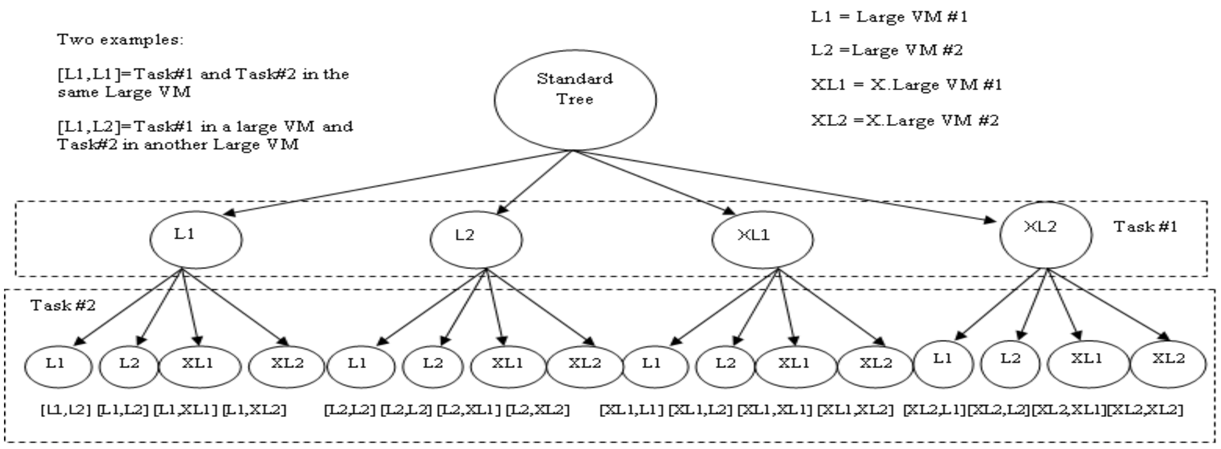
Two examples:
[L1,L1]=Task#1 and Task#2 in the same Large VM
[L1,L2]=Task#1 in a large VM and Task#2 in another Large VM
Standard Tree
L1 = Large VM #1
L2 =Large VM #2
XL1 = X.Large VM #1
XL2 =X.Large VM #2
Task #1
Task #2
[L1,L2] [L1,L2] [L1,XL1] [L1,XL2] [L2,L2] [L2,L2] [L2,XL1] [L2,XL2] [XL1,L1] [XL1,L2] [XL1,XL1] [XL1,XL2] [XL2,L1] [XL2,L2] [XL2,XL1] [XL2,XL2]

Fig. 2. An example of a Standard Tree with two tasks.

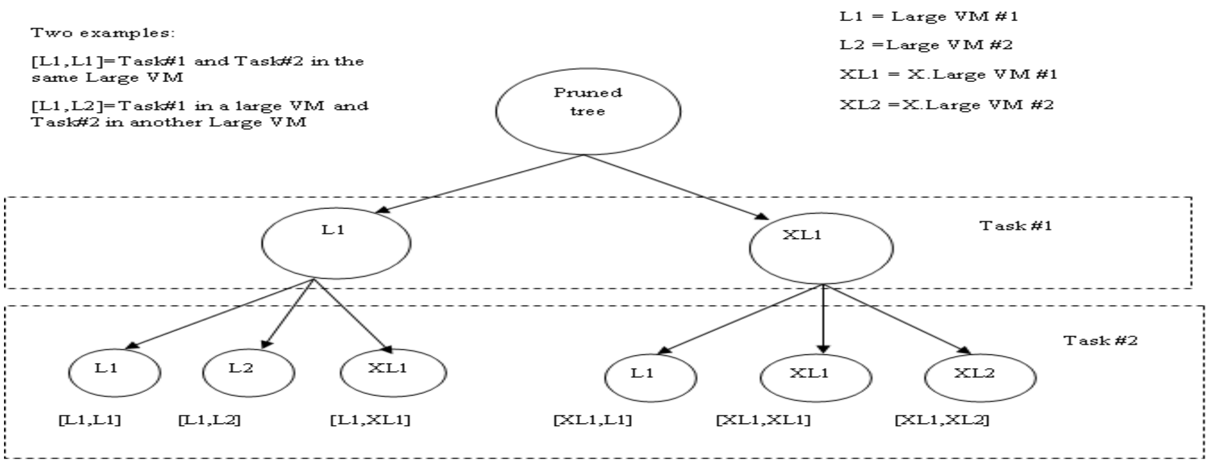
Two examples:
[L1,L1]=Task#1 and Task#2 in the same Large VM
[L1,L2]=Task#1 in a large VM and Task#2 in another Large VM
Pruned tree
L1 = Large VM #1
L2 =Large VM #2
XL1 = X.Large VM #1
XL2 =X.Large VM #2
Task #1
Task #2
[L1,L1] [L1,L2] [L1,XL1] [XL1,L1] [XL1,XL1] [XL1,XL2]

Fig. 3. An Example of pruned tree used algorithm with two task

5.2 Pruned Tree

Fig 3 illustrates an example of a constructed Pruned tree for two tasks (Task#1 and Task#2) and two types of VM (L for Large VM and XL for X.Large VM). We managed to reduce the size of the tree compared to the Standard Tree. Similar to the Standard Tree, the depth (levels) of the tree is the total number of tasks. However, the breadth in Pruned Tree is different than the Standard Tree. As shown in Fig 2 and 3, L2 is eliminated from children of the root node. The reason is that there is no difference in cost and execution time of scheduling the first task in L1 compared to L2. Similarly, XL2 (and generally for root children, all instances of each specific VM Type except one) is eliminated. The rest of the nodes are built from: 1) the set of nodes in the path from the root to the current node, and 2) an extra VM instance from each VM type, where the maximum number of VM instances to be added from each type is the number of tasks. As illustrated in Fig 3 node branches are not of the same root branches, not like Standard Tree branches. The path will be selected as an optimal solution once it does not violate the SLA constraints. In summary the solution space of Pruned Tree is considerably smaller in size compared to Standard tree, as out of VM instances with similar performance, we have only kept one and removed the others when it makes no difference in total cost and execution time. Traversing the Pruned Tree is similar to the Standard Tree, as nodes are sorted by cost ascending from left to right. Hence, it will start consolidating all tasks into the cheapest VM (i.e the most left leaf solution). If that solution does not satisfy the deadline SLA objective; it moves one task to the next cheapest virtual machine, and so on. However, if the budget SLA objective is violated at any time the traversing will stop and return no solution is found. An example of a solution set/vector is the fifth leaf node in Fig 3 (v = {XL1;XL1}) the first task and the second task will be scheduled in the same X.Large VM XL1.

5.3 Algorithm for Provisioning and Scheduling

Algorithm 1: Branch and bound algorithm on a Pruned Tree sorted by cost

```
Input: Request (R) = {SLO, J}
Output: Scheduling vector v

1  foreach vm from each type of VM do
2        vectors ← vectors+ Search({vm},null) ;
3 end
```

```
return Min(vectors); Search (currentBranch, path)
    foreach vm of currentBranch do
        newPath ← path + vm ;
        T← GetT(newPath) ;
        C ←GetC(newPath) ;
        if T < D & C < B then
            if is in leaf then
                return newPath ;
            else
                nextBranch ←newPath;
                Add to nextBranch one VM instance from each type ;
    return Search(nextBranch, newPath);
            end
        end
    end
end
```

Branch and Bound algorithm for a Pruned Tree : It uses a recursive function (Search) to find a solution. The Search function is called on each node except leaf nodes and its inputs are the current branch nodes (currentBranch) and the path solution vector. The core functionality of this recursive function is to iterate on each node of currentBranch and return path as an optimal solution if it is a leaf node and it does not violate the constraints, or call Search if it is not a leaf node and it does not violate the constraints. However, the node will simply be skipped if it violates any of constraints. To find a solution with the minimum cost, the Pruned Tree is sorted by cost ascending from left to right, so it will start consolidating all tasks into the cheapest virtual machine, which is the most left leaf solution. If that solution does not satisfy the objectives, it evaluates the next cheapest solution.

6 CONCLUSION AND FUTURE WORK

In this paper we discussed the problem of provisioning and scheduling Cloud resources for Big Data analytics. To pave the way for organizations to adopt Cloud-hosted Big Data analytics, we carefully capture the user preferences in terms of budget and service level objectives then considering these we proposed an efficient architecture and algorithm to provision resources and schedule tasks using Pruned tree such that the schedule does not violate SLA constraints.

For future work, proposed algorithm need to be implemented and simulation need to be performed using CloudSim which is an extensible simulation toolkit that enables modelling and simulation of cloud computing systems. And gathering the results by doing exhaustive experiments and comparing the results with existing algorithms.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST special publication, vol. 800, p. 145, 2011.
[2] Brendan Jennings Rolf Stadler, "Resource Management in Clouds: Survey and Research Challenges", Springer Science+Business Media New York 2014.
[3] Palden Lama, Xiaobo Zhou," AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud"
[4] Wei Zhang, Sundaresan Rajasekaran, Timothy Wood, Mingfa Zhu," MIMP: Deadline and Interference Aware Scheduling of Hadoop Virtual Machines"
[5] Min Li, Dinesh Subhraveti, Ali R. Butt, Aleksandr Khasymski, Prasenjit Sarkar,,"CAM: A Topology Aware Minimum Cost Flow Based Resource Manager for MapReduce Applications in the Cloud"
[6] Foued Jrad, JieTao, IvonaBrandic, AchimStreit, Karlsruhe," Multi-dimensional Resource Allocation for Data-intensive Large-scale Cloud Applications"