Task Allocation and Re-allocation for Big Data Applications in Cloud Computing Environments

Chapter · May 2020

DOI: 10.1007/978-981-15-3284-9_77

2 authors, including:

Akila D.
Saveetha College of Liberal Arts and Sciences
112 PUBLICATIONS **489** CITATIONS

SEE PROFILE

Task Allocation and Re-allocation for Big Data Applications in Cloud Computing Environments

P. Tamilarasi and D. Akila

Abstract Resource allocation for Big data streams in cloud systems involves selecting the appropriate cloud resources. Since incorrect resource allocation results in either under provisioning or over provisioning, accurate resource allocation becomes challenging in Big data applications. Hence, the objective of this work is to design an optimal solution for resource allocation for minimizing the network bandwidth and response delay. In this paper, a task allocation and re-allocation mechanism for Big data applications is designed. It consists of two important agents: RE-allocation Agent (REA) and Resource Agent (RA). The RA is responsible for mapping the user requirements to the available VMs. The REA monitors the resources and chooses the VMs for resource reconfiguration. Then, it dispatches an allocation or de-allocation request to RA, running in the physical system, based on the varying requirements of virtual machines. Experimental results show that the proposed TARA has less execution time and achieves better utilization of resources, when compared to existing tool.

Keywords Big data · Task allocation · TARA

1 Introduction

Big data consists of large volumes of data generated by various systems in Internet-based applications. Big data processing and analysis are helpful in designing applications for social media, business transactions, etc. Big data denotes the mechanisms involved in extracting, storing, distributing and analyzing huge datasets with various structures and maximum data rate [1]. Cloud environment provides suitable infrastructures for executing huge-sized Big data applications. But, Big data analysis in cloud environment raises various challenges and issues.

P. Tamilarasi (✉) · D. Akila
School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies, Chennai, India
e-mail: tamilmalu@yahoo.com

D. Akila
e-mail: akiindia@yahoo.com

S.-L. Peng et al. (eds.), *Intelligent Computing and Innovation on Data Science*,
Lecture Notes in Networks and Systems 118,
https://doi.org/10.1007/978-981-15-3284-9_77

Scheduling is defined as a set of policies to manage the flow of work which will be executed by computing resources. But, scheduling more tasks in multi-cloud environment becomes the most challenging issue in the current research [2].

Task scheduling algorithms need to provide high performance and efficient system throughput. Since incorrect resource allocation results in either under provisioning or over provisioning, accurate resource allocation becomes challenging in Big data applications. Since cloud resources are having different characteristics, it is difficult to allocate specific resource for each task [3].

The major challenges of task allocation and scheduling involve:

- Selecting the best cloud resources for each task based on their requirements
- Enhance the task completion time
- Reduce cost of execution (in terms of bandwidth and storage)
- Minimize allocation time
- Utilize the idle resources
- Improve resource utilization
- Increased energy efficiency
- Maintaining fairness.

Our research work aims to design a task allocation and re-allocation model which consists of various servers containing virtual machines (VMs).

In this paper, an optimal resource allocation algorithm to minimize the network bandwidth and allocation latency is designed.

2 Related Works

The system developed by Kaur et al. [4] determines various data characteristics which are expressed as Characteristics of Big data (CoBa). Then, it dynamically clusters the cloud resources by applying self-organized maps (SOM). One among these clusters is assigned to Big data streams depending on its CoBa. Whenever the CoBa of any stream varies, the allocated cloud cluster is also varied.

The task scheduling algorithm proposed by Abed et al. [5] performs Big data processing and storing in cloud environments, depending on the user requirements. To enhance the Big data cloud computing solution, they have used a multi-metric-based solution. The model constitutes multiple control nodes and compute nodes. It also consists of a load-balancing algorithm for task scheduling.

To address the problems of cloud-based Big data applications such as performance, cost and availability, a multi-objective optimization algorithm was developed by Dai et al. [6]. By analyzing the interactions between these metrics, their system has been designed and implemented in experimental test bed.

A joint optimization algorithm proposed by Vakilinia [7] aims to reduce the power consumed by servers and cost of VM migration satisfying the resource and bandwidth conditions. It also handled the server heterogeneity problem. In the re configuration phase, VMs of uncompleted tasks may be migrated and new tasks are allocated to VMs.

Surputheen et al. [8] have proposed a concurrent VM reconfiguration mechanism for Big data tool which is MapReduce on virtualized cloud environments. It adds cores to VMs to execute local tasks temporarily and adjust the computing efficiency of the VMs to contain the scheduled tasks unlike the traditional schemes which can lead to user-friendly configuration methods for cloud resources.

A deadline-aware flexible bandwidth allocation for big-data transfers (DaFBA) algorithm has been developed by Srinivasan et al. [9]. In this approach, the allocated bandwidth can be adaptively adjusted at any time. The scheduling algorithms apply batch processing and dynamic scheduling at each interval for each request. They maximize the acceptance rate and satisfy the deadline constraints.

3 MapReduce Tool

MapReduce is a framework which can be applicable in large parallel data analyses. In MapReduce model, the calculation considers a set of key/value pairs as input and output.

It contains two functions for computation: Map and Reduce. The Map function accepts a pair of input to create a set of temporary key/value pairs, whereas the Reduce function gets a temporary key/values corresponding to that key. It aggregates these value pairs to generate a tiny set of values.

MapReduce with Hadoop implementation employs Hadoop distributed file system (HDFS) which stores data and also the interim results. HDFS maps all the locally stored data to a single file system order enabling the data to be spread at the entire set of nodes [10].

When a task request is received for scheduling, the VMs should have enough processing slots for each task. If no such slot exists, the task is rescheduled to a far away node by transferring the corresponding data. The MapReduce architecture is shown in Fig. 1.

4 Proposed Solution

4.1 Overview

Our proposed scheme consists of two important agents: RE-allocation Agent (REA) and Resource Agent (RA). The REA dispatches de-allocation and allocation requests to the hypervisor based on the physical machine. The RA monitors resource and chooses VM residing at the cluster which demands resource reconfiguration. Then, it dispatches an allocation or de-allocation request to RA, running in the physical system.

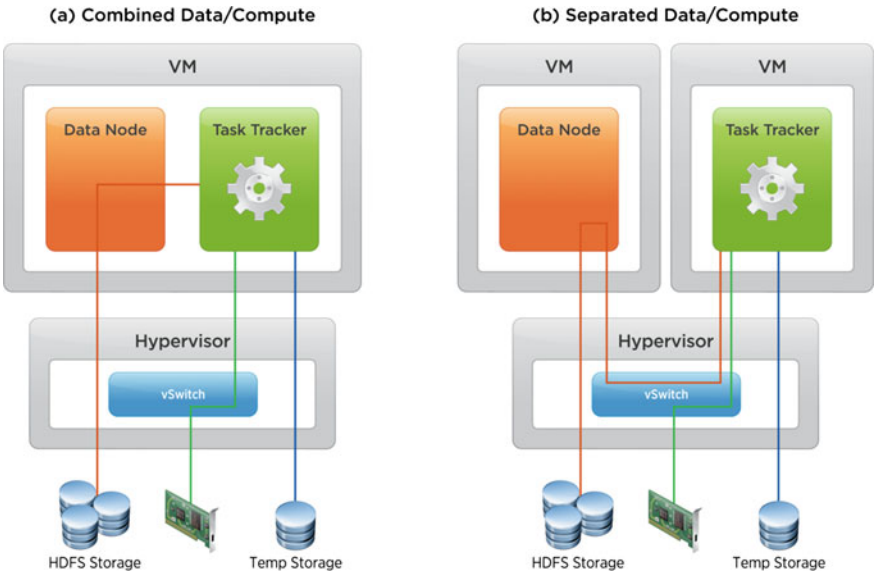

Fig. 1 MapReduce architecture on various VMs

4.2 Resource Allocation

4.2.1 VM Allocation

A VM allocation $\delta(m_1), \delta(m_2), \ldots, \delta(m_i)$ is defined as mapping of server to set a set of VMs that should satisfy the following two conditions:

1. Each VM should be allocated to at least one server, and no VM is allocated to more than one server

$$\cup_{m_j \in m} \Theta(m_i) = \Theta \tag{1}$$

$$\Theta(m_i) \cap (m_j) = \Phi, \forall\ 1 \leq y, i \neq j \tag{2}$$

2. For each server, the total resource requirements of its hosted VM do not exceed its available resources

$$\sum_{\theta j \in \Theta(mi)} w_i \leq \mathrm{CS}_j, \forall\ 1 \leq i \leq y \tag{3}$$

where W_i is the resource requirement of VM_i and CS_j is the available capacity of server.

4.2.2 Mapping of VMs to Tasks

Let $\{T = T_1, T_2, \ldots T_n\}$ be the set of tasks to be assigned.
For each task T_j, let BWR_j, D_j be the bandwidth requirement and deadline of the task.
Let TS_j be the size of each task T_j
Let $\{S_i = \{\text{VM}_{i1}, \text{VM}_{i2}, \ldots \text{VM}_{im}\}$ be the set of VMs on server S_i.
Let $\{C_{i1}, C_{i2}, \ldots C_{im}\}$ and $\{E_{i1}, E_{i2}, \ldots \text{E}_{im}\}$ be the capacities and expected delay of each VMs on S_i, respectively.
Let Time_{UE} be the time required to execute a task of unit size.

Algorithm-1: Resource allocation by RA

For each Task T_j

1. Submit T_j to RA
2. RA extracts BWR_j and D_j from T_j
3. For each VM_ik on server S_i, k=12...m
4. Estimate $\text{E}_\text{ik} = \text{TS}_\text{j} * \text{Time}_{\text{UE}}$
5. If ($\text{BWR}_\text{j} < \text{C}_\text{ik}$) and ($\text{E}_\text{ik} < \text{D}_\text{j}$)
6. Allocate T_j to VM_ik
7. End if
8. End For
9. If(T_j could not find any suitable VM)
10. Submit Tj to REA
11. End if
12. End For

In Algorithm-1, if a task T_j is submitted to RA with task number, task size, bandwidth required and required deadline), RA compares the task characteristics to each VM characteristics. If a VM with capacity higher than the required bandwidth and expected delay less than the deadline is found, it is selected and task will be assigned to it. If no such VM can be found, then the task request will be forwarded to the REA.

The REA handles task scheduling under load balancing. Hence, it will check the VMs on another server apart from S_i. If another idle VM can be found on any server, it can migrate to that VM such that the traffic load is reduced and number of resources is minimized. If all are engaged, then the task will be put in a waiting queue. These steps are repeated continuously.

4.3 Resource Reconfiguration

In order to process the varying demands on each VM, it may be dynamically reconfigured. When more number of physical resources are found, each VM is allocated with more number of virtual CPUs and memory space during VM execution.

The REA monitors the resources and chooses a VM which demands resource reconfiguration. If the resource requirement W of any VM exceeds the available capacity of the server CS, then it dispatches an allocation request (ALLOC) to RA. On the other hand, if the available capacity *C* of any VM is excess than its demanded resources, it dispatches a de-allocation request (DeALLOC) to the corresponding RA.

Upon obtaining requests from REA, RA reallocates the demanded virtual CPUs to the VM. RA requests the hypervisor of the physical server, to plug or to remove virtual CPUs for VMs present in the system. Algorithm-2 summarizes the steps involved in the resource reconfiguration process.

Algorithm-2 Resource Re-allocation

1. For each VM_{ik} on server S_i, k=12...m
2. REA monitors VM_{ik} at time t
3. If($W_{ik} > CS_i$) ,
4. REA send ALLOC to RA
5. Else if ($C_{ik} > W_{ik}$)
6. REA send DEALLOC to RA
7. End if
8. If(RA receives ALLOC)
9. RA demands hypervisor to hot-plug virtual CPUs
10. Else if (RA receives DEALLOC)
11. RA demands hypervisor to un-plug virtual CPUs
12. End if
13. End For

5 Experimental Results

5.1 Results

The workload with size 25 GB is input to Amazon Elastic Compute Cloud (EC2) at a velocity of 50, after every 5 min for the duration of 1 h. The proposed task allocation and re-allocation (TARA) is implemented on Amazon EC2 compute optimized c4 large instances. The performance of TARA is evaluated by comparing with the Apache Storm 0.92 tool using the execution time and resource utilization metrics.

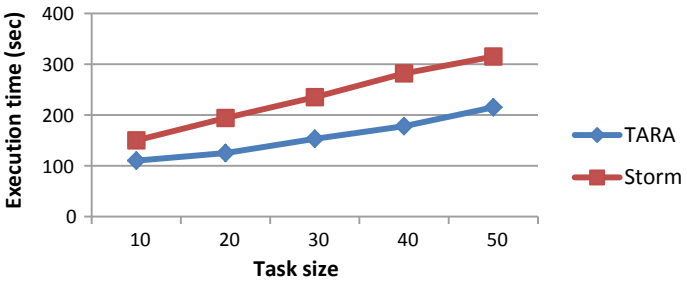

Fig. 2 Execution times for various task sizes

5.1.1 Execution Time

In this section, the execution time for different task sizes is measured and depicted in Fig. 2.

It can be observed from the figure that among all the sizes of tasks, the execution time is lowest when task size is low. Similarly, it becomes highest when the task size is high. The results show execution times for task are less in case of TARA when compared to Storm.

5.1.2 Resource Utilization

The results of resource utilization (%) for different sizes of task are measured and depicted in Fig. 3.

Figure 3 shows the comparison results of both the systems in terms of resource utilization. From the results, it can be observed that TARA achieves higher utilization when compared to Storm.

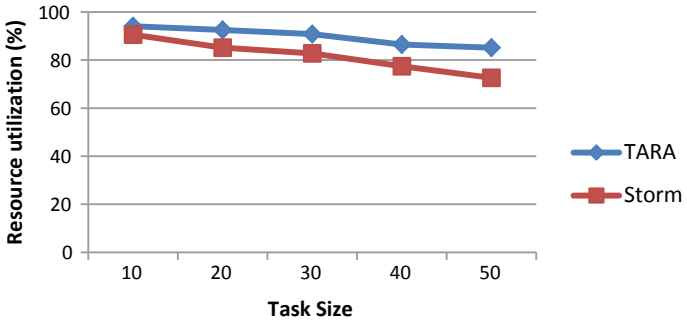

Fig. 3 Resource utilization for various task sizes

6 Conclusion

In this paper, a task allocation and re-allocation mechanism for Big data applications has been designed. It consists of two important agents: RE-allocation Agent (REA) and Resource Agent (RA). The RA is responsible for mapping the user requirements to the available VMs. The REA monitors the resources and chooses the VMs for resource reconfiguration. Then, it dispatches an allocation or de-allocation request to RA, running in the physical system, based on the varying requirements of virtual machines. The performance of TARA is evaluated by comparing with the Apache Storm 0.92 tool using the execution time and resource utilization metrics. Performance results have proved that TARA has less execution time and achieves better utilization of resources, when compared to existing tool.

References

1. Usama M, Liu M, Chen M (2017) Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs, Digital communications and networks. Elsevier, pp 260–273
2. Mishra SK, Satya Manikyam P, Sahoo B, Obaidat MS, Puthal D, Pratama M (2017) Response-aware scheduling of big data applications in cloud environments. Future Technologies Conference (FTC), Canada
3. Ebrahimi M, Mohan A, Lu S (2018) Scheduling Big data workflows in the cloud under deadline constraints. In: IEEE fourth international conference on Big data computing service and applications
4. Kaur N, Sood SK (2017) Dynamic resource allocation for big data streams based on data characteristics (5Vs). Int J Netw Manage 27(4)
5. Abed S, Shubair DS (2018) Enhancement of task scheduling technique of Big data cloud computing, In: International conference on advances in Big data, computing and data communication systems (icABCD), IEEE, South Africa
6. Dai W, Qiu L, Wu A, Qiu M (2016) Cloud infrastructure resource allocation for Big data applications. IEEE Trans Big Data 4(3):313–324
7. Vakilinia S (2018) Energy efficient temporal load aware resource allocation in cloud computing datacenters. J Cloud Comput Adv Syst Appl 7(2):2–24
8. Surputheen MM, Abdullah M (2017) Dynamic resource allocation scheme for map reduce tool in cloud environment. J Comput Eng 19(4)
9. Srinivasan SM, Truong-Huu T, Gurusamy M (2018) Deadline-aware scheduling and flexible bandwidth allocation for Big-data transfers. IEEE Access 6:74400–74415
10. Tamilarasi P, Akila D (2019) Ground water data analysis using data mining: a literature review. Int J Recent Technol Eng 7:2277–3878