TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/09 pp149–159
Volume 22, Number 2, April 2017

Load Feedback-Based Resource Scheduling and Dynamic Migration-Based Data Locality for Virtual Hadoop Clusters in OpenStack-Based Clouds

Dan Tao*, Zhaowen Lin, and Bingxu Wang

Abstract: With cloud computing technology becoming more mature, it is essential to combine the big data processing tool Hadoop with the Infrastructure as a Service (IaaS) cloud platform. In this study, we first propose a new Dynamic Hadoop Cluster on IaaS (DHCI) architecture, which includes four key modules: monitoring, scheduling, Virtual Machine (VM) management, and VM migration modules. The load of both physical hosts and VMs is collected by the monitoring module and can be used to design resource scheduling and data locality solutions. Second, we present a simple load feedback-based resource scheduling scheme. The resource allocation can be avoided on overburdened physical hosts or the strong scalability of virtual cluster can be achieved by fluctuating the number of VMs. To improve the flexibility, we adopt the separated deployment of the computation and storage VMs in the DHCI architecture, which negatively impacts the data locality. Third, we reuse the method of VM migration and propose a dynamic migration-based data locality scheme using parallel computing entropy. We migrate the computation nodes to different host(s) or rack(s) where the corresponding storage nodes are deployed to satisfy the requirement of data locality. We evaluate our solutions in a realistic scenario based on OpenStack. Substantial experimental results demonstrate the effectiveness of our solutions that contribute to balance the workload and performance improvement, even under heavy-loaded cloud system conditions.

Key words: Hadoop; resource scheduling; data locality; Infrastructure as a Service (Iaas); OpenStack

1 Introduction

Cloud computing is one of the hottest areas of research at home and abroad, which integrates large-scale computing, storage, and network resource via a network, and provides these resources for different users on demand[1]. As an open-source framework for distributed system architecture, Hadoop can achieve large-scale data computing and storage, and is usually deployed on physical cluster. There are some drawbacks in traditional Hadoop clusters. First, its deployment and configuration are tedious tasks. When Hadoop starts running, the realtime monitoring on Hadoop consumes plenty of manpower and financial resources. Second, the fluctuation of tasks causes imbalance of resource utilization. With the appearance of peaks in the tasks, resource bottlenecks may be encountered. In contrast, the troughs in the tasks will bring idle resource. Hadoop cannot realize dynamic resource allocation. Third, the utilization of high-performance computers in physical

• Dan Tao and Bingxu Wang are with School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, and Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing 210003, China. E-mail: dtao@bjtu.edu.cn.

• Zhaowen Lin is with Network and Information Center, Institute of Network Technology, Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, National Engineering Laboratory for Mobile Network Security, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: linzw@bupt.edu.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2016-10-01; revised: 2016-12-26; accepted: 2016-12-27

clusters is insufficient, particularly for computation and storage resource, resulting in severe resource wastage.

To solve the abovementioned problems, it is essential to deploy a Hadoop cluster on an OpenStack-based cloud as its service[2]. This study adopts OpenStack, which can provide an Infrastructure as a Service (IaaS) solution in the form of Virtual Machines (VMs). Sahara, an open-source project, is developed to rapidly deploy a Hadoop cluster in an OpenStack-based cloud environment. A virtual cluster, which can simplify cluster management, enables autonomic management of the underlying hardware, facilitating cost-effective workload consolidation and dynamic resource allocations for better throughput and energy efficiency. However, virtualization in such cloud platforms is known to cause performance overheads[3]. Understanding how to optimize the performance of a Hadoop cluster has attracted considerable attention. Researchers have accumulated a series of research achievements on resource scheduling and data locality in the related context.

Scheduling techniques for dynamic resource adjustment have been recently addressed. Sandholm and Lai[4] presented a dynamic priority parallel task scheduler for Hadoop. It allowed users to control their allocated capacity by adjusting their spending time. Sharma et al.[5] proposed a MapReduce resource Orchestrator (MROrchestrator) framework, which dynamically identified resource bottlenecks and resolved them through fine-grained, coordinated, and on-demand resource allocations. However, the abovementioned studies focused on a resource scheduling-based traditional Hadoop cluster. Lama and Zhou[6] studied automated resource allocation and configuration of the MapReduce environment in the cloud without considering the load of physical hosts. Zuo et al.[7] proposed a resource evaluation model based on entropy optimization and dynamic weighting. The entropy optimization filtered the resources that satisfied user QoS and system maximization by goal function, constraints of maximum entropy, and the entropy increase principle, which achieved optimal scheduling and satisfied user QoS. Liu et al.[8] presented an adaptive method aiming at spatio-temporal efficiency in a heterogeneous cloud environment. A prediction model based on an optimized kernel-based extreme learning machine algorithm was proposed for a quick forecast of job execution duration and space occupation, which consequently facilitates the process of task scheduling.

For data locality, to address the conflict between locality and fairness, Zaharia et al.[9] proposed a simple delay scheduling algorithm wherein a job waited for a limited amount of time for a scheduling opportunity on a node that has data on it. Experimental results showed that waiting can achieve both high fairness and high data locality. Jin et al.[10] proposed an availability-aware data placement strategy, and its basic idea was to dispatch data based on the availability of each node for reducing network traffic and improve data locality. Both works were studied on a traditional Hadoop cluster. Thaha et al.[11] presented a data location-aware virtual cluster provisioning strategy to identify the data location and provision the cluster near the storage. However, multiple tasks might be executed on a same physical host, which negatively impacted system performance.

Motivated by this, we propose load feedback-based resource scheduling and dynamic migration-based data locality solutions based on a novel Dynamic Hadoop Cluster on IaaS (DHCI) architecture. The resource utilization can be improved by the load balance of physical hosts and the flexible scalability of VMs. Moreover, based on the separated deployment of the computation and storage VMs, computation VMs can be quickly migrated to match their corresponding storage VMs in order to effectively guarantee data locality.

The remainder of this study is organized as follows. In Section 2, we introduce a DHCI architecture. Based on this architecture, load feedback-based resource scheduling and dynamic migration-based data locality solutions are explored in Section 3. In Section 4, we perform a comprehensive evaluation to validate our solutions. Finally, we conclude this study in Section 5.

2 DHCI Architecture

There exists a huge difference on Hadoop's running environment between a physical cluster and an IaaS cloud platform[12]. In the IaaS cloud environment, Hadoop is deployed on VMs provided by the cloud platform. In this case, the Hadoop cluster cannot sufficiently understand the resource usage of the underlying physical hosts, which will result in load imbalance and performance degradation. In addition, the scalability of the Hadoop cluster is not satisfactory. In contrast, the virtual Hadoop cluster on the cloud platform is more convenient for the flexible adjustment of cluster scaling.

Motivated by this, we integrate Hadoop onto an IaaS cloud platform and propose a new DHCI architecture. In the DHCI architecture illustrated in Fig. 1, we introduce four kernel modules besides the original packages of private cloud and Hadoop.

- *Monitoring Module*: Considering that different clusters in a virtual environment are isolated, Hadoop cannot obtain the load of physical hosts at all. A monitoring module is introduced to periodically monitor the load on the physical hosts as well as the VMs. The load information collected can be used to provide the basis for resource scheduling.
- *Scheduling Module*: It is responsible for two aspects: (1) periodically pushing the load information of the physical hosts to the scheduling node (e.g., ResourceManager) in Hadoop and (2) issuing the corresponding scalability strategy to the VM management module according to the load of the VM clusters.
- *VM Management Module*: It achieves dynamic scaling of the VMs by adding or deleting operations. This is an execution module which takes instructions from the scheduling module and interacts with the VMs on the IaaS platform.
- *VM Migration Module*: It is used to detect a task's data locality and execution process. Once this module finds (1) the execution progress of a task is slower than a given threshold and (2) its CN and SN do not meet the data locality, it will migrate this task to a suitable physical host by the storage of data duplication.

In summary, the DHCI architecture has two features: (1) joint load monitoring, and (2) flexible resource scheduling. The monitoring module monitors the physical and virtual resources with full awareness of the current system load conditions. This necessary data can be utilized to optimize subsequent resource scheduling. Through the scheduling and VM management modules, the resource utilization can be optimized according to the load balancing of the physical hosts and the flexible scalability of the VMs. Based on the idea of "mobile computing", the reuse of VMs migration, achieved by the VM migration module in the DHCI architecture, can also reduce bandwidth consumption and improve system performance.

3 Resource Scheduling and Data Locality

3.1 Load feedback based resource scheduling

The resource scheduling selects appropriate resources assigned to different tasks for execution[13]. Currently, most evaluation indicators are static or predictive physical performance items, such as the computing power of CPU, storage capacity, and network bandwidth[14]. However, in the dynamic environment of cloud computing, it is difficult for these indicators to reflect the actual service ability of the physical resource.

In our solution, the load of physical hosts can be described from two aspects: CPU utility rate and load average. Load average is a kind of performance

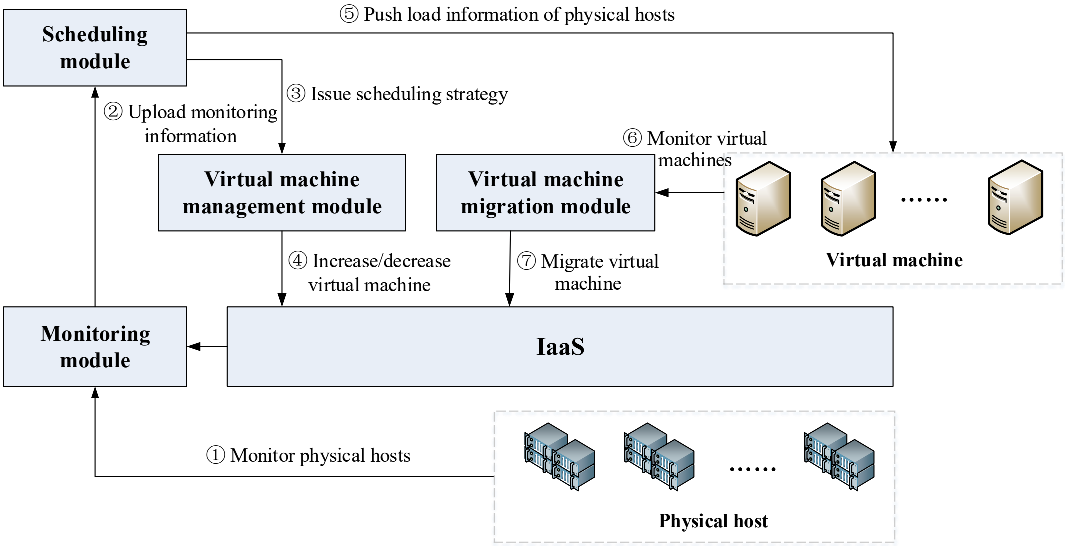

Scheduling module
⑤ Push load information of physical hosts
③ Issue scheduling strategy
② Upload monitoring information
⑥ Monitor virtual machines
Virtual machine management module
Virtual machine migration module
Virtual machine
④ Increase/decrease virtual machine
⑦ Migrate virtual machine
Monitoring module
IaaS
① Monitor physical hosts
Physical host
......
......

Fig. 1 DHCI architecture.

parameter (e.g., memory, disk, and network). This parameter denotes the average utilization rate of run queues. The higher the values of CPU utility rate and load average, the heavier the workload of a physical host. The VMs mentioned here run Linux OS; therefore, the performance of the system can be monitored every minute using the *Top* command in Linux. For the whole VM cluster, we adopt a unified script to collect the status of resource consumption.

In the DHCI architecture, for a physical host, its load information will be uploaded and fed back to the scheduling module periodically via the monitoring module. We adopt a single-level threshold method to compare the load information, and its workflow can be illustrated in Fig. 2. Once the load exceeds a preset threshold, the physical host is considered as stressed out, and the resource application using it will be canceled. Otherwise, the resource application will be supported.

One of the most significant advantages of integrating Hadoop onto the IaaS cloud platform is flexibility. In other words, the scale of the virtual cluster can dynamically adjust according to its real-time workload. Similarly, for the virtual Hadoop cluster comprising multiple VMs, the monitoring module in the DHCI architecture collects its load information and feeds it back to the scheduling module. A double-level threshold method is used to distinguish between the lowest load VM and the highest load VM, as shown in Fig. 3. If the load exceeds a ceiling value, the VM addition operation will be issued and a new VM will be created on the lowest load physical host. However, if it is below a floor value, the VM deletion operation will be issued and excessive VM(s) will be deleted on the highest load physical host.

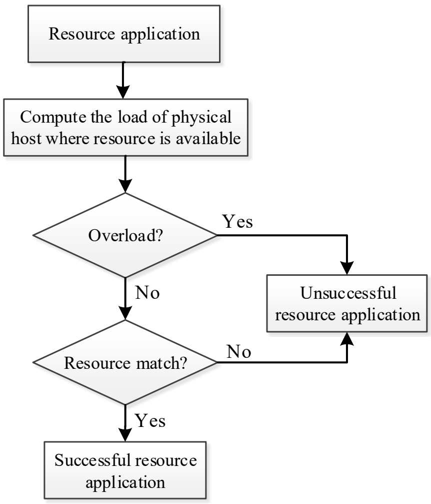

Resource application
Compute the load of physical host where resource is available
Overload?
Yes
No
Unsuccessful resource application
Resource match?
No
Yes
Successful resource application

Fig. 2 Single-level threshold resource scheduling algorithm.

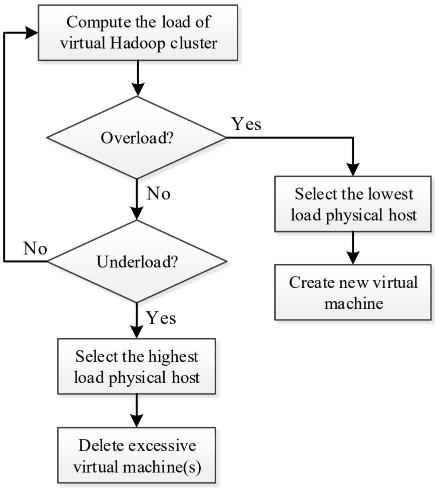

Compute the load of virtual Hadoop cluster
Overload?
Yes
No
Select the lowest load physical host
Create new virtual machine
No
Underload?
Yes
Select the highest load physical host
Delete excessive virtual machine(s)

Fig. 3 Double-level threshold resource scheduling algorithm.

3.2 Dynamic migration based data locality

Large-scale distributed systems aim at processing data as close as possible to the storage location to reduce data movement between the computer and storage facilities, which is typically known as data locality[12]. Data locality is a crucial factor impacting the performance of a virtual Hadoop system. In a traditional Hadoop cluster comprising physical hosts, computation VMs (used for task computation, denoted by CNs), and storage VMs (used for data storage, denoted by SNs) are combined into a single VM. The advantage is that CNs can directly obtain data from SNs while avoiding data transmission across a network. However, this deployment is no longer an effective method for a virtual Hadoop system. The scalability of a virtual Hadoop cluster can be achieved by dynamically adding or deleting VMs. The combination of CNs and SNs results in poor scalability. This means that once CNs are added or deleted, their corresponding SNs should be applied with the same operation. Moreover, in the process of VM migration, VMs functioning as CNs and SNs will cause massive data movement, thereby reducing the efficiency of VM migration.

Therefore, in the DHCI architecture, the separation of CNs and SNs is adopted to improve flexibility. In particular, they are deployed as respective VMs. In this manner, CNs can be migrated to a “suitable” place based on the idea of “mobile computing”. It is obvious that this deployment form offers several advantages over a centralized method[15]: (1) strong scalability, which allows for respective fluctuating numbers of CNs or SNs and (2) flexible migration, i.e., CNs can be migrated without considering any other SNs.

Compared to that of the traditional Hadoop cluster, data locality in the DHCI architecture can be classified into three categories[16], as illustrated in Fig. 4.

- Host data locality: CNs and SNs are deployed on the same host (e.g., VM1 and VM2 are on Host1).
- Rack data locality: CNs and SNs are deployed on the same rack but different hosts (e.g., VM1 and VM4 are on Rack1).
- Across-rack data locality: CNs and SNs are deployed on different racks (e.g., VM1 and VM10 are on Rack1 and Rack2, respectively).

Experimental results have shown that the speeds of task execution for meeting different types of data locality are significantly different under the same conditions[17]. In particular, the task completion time for meeting “rack data locality” and “across-rack data locality” approaches three and four times as long as that for meeting “host data locality”, respectively. Data transmission between co-located VMs is often as efficient as local data access mainly because inter-VM communication within a single host is optimized by the hypervisor[18]. Hence, we consider to improve “host data locality” in order to optimize the performance of the DHCI architecture. Considering that the separation of CNs and SNs influences data locality, we dynamically migrate the CNs to any host(s) or rack(s) where the corresponding SNs are deployed to guarantee data locality. During the migration process, a VM remains on and the program executed in this VM keeps track of the running state. Even if this VM is connected to a network, the network connection will not be affected. In fact, the cost of migrating the VM is considerably less than that of reading/writing operations among different VMs[16].

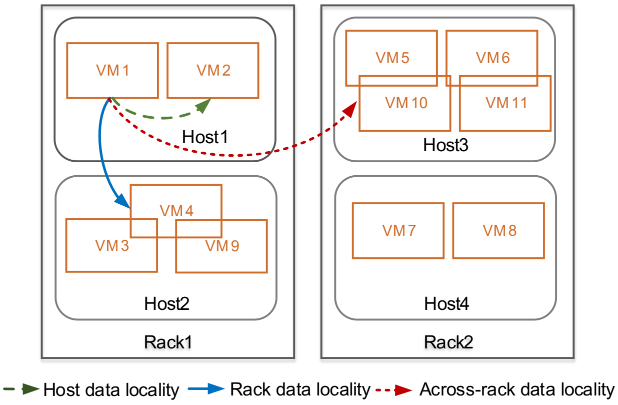


Fig. 4 Three types of data locality in a virtual Hadoop cluster.

First, the initial resource allocation should keep data locality. In the Hadoop YARN adopted, ContainerAllocator is responsible for communicating with Resource Manager and applying resources for tasks. Usually, there exist three backups for each task in HDFS. Considering the level difference of data locality, there will be multiple resource requests. VMs can be allocated resource, prioritized by “host data locality”, “rack data locality”, and “across-rack data locality”. A resource request for a task can be described as a tuple <Priority, Hostname, Capability, Containers>, where “Hostname” can represent the ID of the host or the rack.

Consider the case in Fig. 4 as an example wherein a task applies a resource in order from the host to the rack. If this task can acquire a resource from a certain host, the request will stop; otherwise, it will apply it one by one in the following manner:

<20,“Host1(VM1,VM2)”,“memory:2G”,“1”>
<20,“Host2(VM3,VM4,VM9)”,“memory:2G”,“1”>
<20,“Host3(VM5,VM6,VM10,VM11)”,“memory:2G”, “1”>
<20, “Rack1”,“memory:2G”,“1”>
<20, “Rack2”,“memory:2G”,“1”>.

Second, data locality should be optimized in the process of task execution. Hadoop monitors task execution and judges whether data locality is satisfied or not. If not, Hadoop continues to search whether there exist one or more hosts which can meet the requirement of data locality. Then, CN will migrate to the correct one.

The real-time dynamics and uncertainty of cloud resources make resource management and task scheduling challenging[19]. Parallel computing entropy is developed from Shannon information entropy, which has the characteristics of symmetry, nonnegativity, and scalability. Sun et al.[20] have proved that parallel computing entropy can be maximized if and only if the load is completely balanced in the homogeneous cluster or the grid environment. In this study, we extend the method of parallel computing entropy into the heterogeneous cluster or cloud computing environment. To accurately grasp the dynamic load and available information of resources, we propose a parallel computing entropy based VM migration solution. In particular, the load of the i-th physical host can be represented as L_i,

$$L_i = \sum_{j=1}^{n} \omega_{ij} s_j \tag{1}$$

where n denotes the total number of VMs on the physical host i. For the physical host i, ω_{ij} denotes the ratio of the number of VMs with the task type j to the total number of VMs and s_j denotes the computation of VM with the task type j.

The relative load K_i of physical host i, which is the ratio of the individual load to the total load of m physical hosts, can be represented as follows:

$$K_i = L_i / \sum_{i=1}^{m} L_i \tag{2}$$

Considering the difference among performance parameters (e.g., CPU, memory, and bandwidth) of each physical host, we express the processing capacity of the physical host as follows:

$$P_i = \{P_{\text{cpu}, i}, P_{\text{loadaverage}, i}\} \tag{3}$$

where $P_{\text{cpu}, i}$ and $P_{\text{loadaverage}, i}$ respectively denotes the CPU utility rate and the load average of the available physical host i. We quantify the above equation as follows:

$$P_{\text{total}} = \alpha P_{\text{cpu}, i} + \beta P_{\text{loadaverage}, i} \tag{4}$$

where α and β are constant coefficients and satisfies $\alpha + \beta = 1$.

Hence, the relative load K_i of the physical host i can be expressed as follows:

$$K_i = \frac{L_i / P_{\text{total}, i}}{\sum_{i=1}^{m} L_i / P_{\text{total}, i}} \tag{5}$$

Assume that there are m physical hosts in a cloud computing environment. At time t, the relative load of the physical host i is denoted as $K_i(t)$. Parallel computing entropy of the whole physical cluster at time t can be defined as follows:

$$H(t) = \sum_{i=1}^{m} K_i(t) \ln \frac{1}{K_i(t)} \tag{6}$$

During the migration process, the migration can be selected by the maximum entropy increment at each step, and thus make the execution time of all the tasks the shortest. Therefore, the increase of parallel computing entropy causes (1) the decrease in calculation amount of physical host(s) with the biggest load and (2) a more balanced load of other physical host, and thus a decrease in the total execution time of all the tasks. The goal of load balancing is to increase the parallel computing entropy as far as possible to reduce the total execution time of all the tasks.

The amount of computation for all the VMs on each physical host can be calculated using the sampling interval T, and the parallel computing entropy $H(t)$ can be calculated by Eq. (6). Once the value of $H(t)$ is less than the threshold of γ, the load balancing of the system is unsatisfactory to some extent, the VM migration process is needed.

To achieve a completed CN migration, three major issues ("2W1H") should be solved.

3.2.1 Which CN needs to be migrated?

The physical host needs to be migrated as it is the highest-loaded device. Similarly, the CN that needs to be migrated is the CN experiencing the heaviest-computation load. For a CN needs to satisfy the need to be migrated, it must satisfy two conditions: (1) it is the highest-computation CN, i.e., CN has the greatest amount of calculations and (2) CN and its corresponding SN are on a separate host or rack, as shown in Algorithm 1.

3.2.2 Where should a CN be migrated?

The destination host to which a CN is migrated should include its corresponding SNs store data replication. In a virtual Hadoop cluster, each task can be partitioned into several Map and Reduce tasks. Each Map task runs map functions processing one data block (128 MB by default in YARN). The data replication of each data block is three by default, and can be stored in different

Algorithm 1 Determine CN to be migrated

```
Input: the load $L_i$ of $m$ physical hosts.
Output: CN.
begin
    Calculate the average load of the system $L_{\text{avg}} = \sum_{i=1}^{m} L_i / m$;
    Calculate the load difference $\triangle L_i = L_i - L_{\text{avg}}$;
    Sort $\triangle L_i$ from the largest to the smallest, and store $q_1$, $q_2$, ..., $q_m$ into the queue $Q_{\text{ph}}$ ;
    Select the first element $q_1$ in the queue $Q_{\text{ph}}$, $q_1$ denotes the ID of physical host with the highest load;
    Calculate the corresponding computation amount $c_j$ of CN$_j$ on the physical host $q_1$, where $1 \leqslant j \leqslant s$;
    Sort $c_j$ from largest to smallest, and store CN$_1$, CN$_2$, ..., CN$_s$ into queue $Q_{\text{CN}}$;
    Select the first element CN$_1$ in queue $Q_{\text{CN}}$, CN$_1$ denotes the ID of CN with the heaviest-computation amount;
    return CN$_1$
end
```

hosts even racks. Here, we choose the least-loaded host which satisfies the requirement of data locality as the destination host with the lowest cost.

3.2.3 How should one migrate a CN?

The selected OpenStack cloud platform can support VM migration very quickly. The whole migration process involves three kinds of physical hosts: the source, destination, and control hosts. We mainly utilize the Python interface function in the Libvirt tool to migrate the VM, and data transmission can be realized in a tunneled way. As space is limited, the further description will not be given.

4 Emulation and Analysis

In this emulation, we choose OpenStack as the cloud platform and Hibench as the Hadoop performance testing tool. Hibench can provide a series of typical Hadoop benchmark test cases, which can be directly used to conduct a performance test. Here, three classic benchmark test cases, WordCount (counts the words in the input files), TeraSort (sorts the data generated by TeraGen), and Sort (sorts the data written by the random writer), are adopted to evaluate the performance of the proposed DHCI architecture. The hardware configuration for the testing environment is listed in Table 1.

4.1 Comparison of running time under the same load

First, we compare the running times using three classic schedulers (FIFO Scheduler, Fair Scheduler, and Capacity Scheduler) for the traditional Hadoop cluster and DHCI architecture. It should be noted that FIFO-DHCI, Fair-DHCI, and Capacity-DHCI are defined as the three schedulers used in the DHCI architecture. The emulation results in Fig. 5 show that the running time in the DHCI architecture is less than that in the traditional Hadoop cluster with the same workload (data volume is 2 GB). Using Fair Scheduler as an example, for the WordCount, TeraSort, and Sort cases, the running times in the DHCI architecture are decreased by 14%, 9%, and 8%, respectively. The proposed scheduling approach can outperform other traditional approaches mainly because in the DHCI architecture, the resource allocation can be avoided on overburdened physical hosts or the strong scalability of the virtual cluster can be achieved by fluctuating the number of VMs.

Table 1 Hardware configuration for the testing environment.

Parameter	Configuration
CPU type	4-core 2.4 GHz Intel(R) Xeon (R)
Memory	32 GB
Network card	Three 2 Gbps LANs
OS	Linux 14.04
VM images	Virtual CPU, 2 GB RAM, and 30 GB HDD

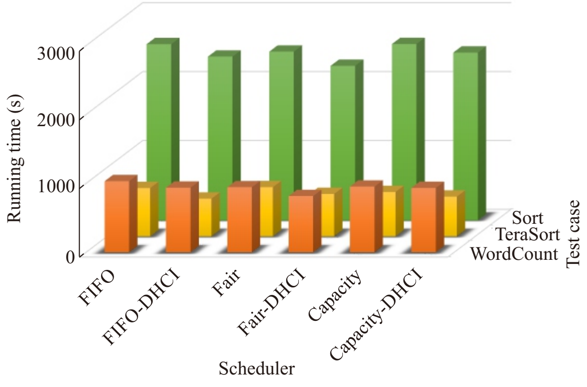


Fig. 5 Comparison of running time for the three schedulers under the two architectures.

Second, we evaluate the running time under two architectures with different data volumes (from 256 MB to 2048 MB), as illustrated in Fig. 6. As far as WordCount is concerned, the emulation results show that the performance of the DHCI architecture is superior to the traditional Hadoop cluster. Obviously, the more the data volume, the greater the benefits of the DHCI architecture be apparent.

4.2 Comparison on running time under the load pressure

Once the workload approaches the peak value or valley value, the DHCI architecture will process tasks by dynamically increasing or decreasing the number of virtual machines. In this way, compared to the

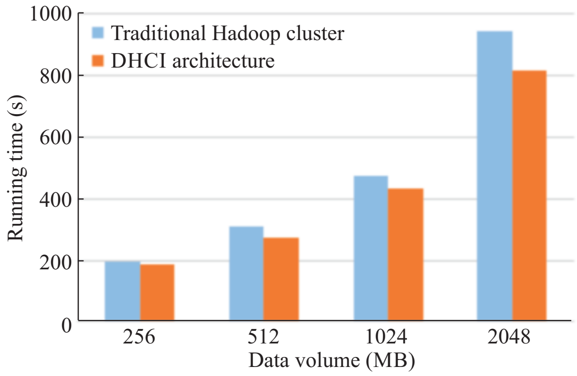


Fig. 6 Comparison of running time under two architectures with different data volumes.

traditional Hadoop cluster, the DHCI architecture can process the same workload with less running time by optimizing resource utilization. To test the operational condition of the Hadoop cluster under certain load pressure in a private cloud environment, when the workload on virtual cluster 1 reaches the maximum, a new virtual cluster 2 will be added, as shown in Fig. 7.

Supposing that the original workload of the task (WordCount, TeraSort, and Sort) run on virtual cluster 1 is 2 GB. The effect on the operational efficiency of the task on the two architectures with different scales of virtual clusters can be illustrated in Fig. 8. There is no doubt that the dynamic increase of virtual clusters allows for resource utilization as well as load balancing, and thus results in less running time. Under a specific workload, compared to the running time for the traditional Hadoop cluster, that for the traditional cluster under load pressure decreases to 67%, 71%, and 55% for WordCount, TeraSort, and Sort, respectively. This depicts that multiple tasks can be simultaneously operated on a virtual Hadoop cluster with relatively low cost. In the case of load pressure, compared with the running time for the traditional Hadoop cluster, that for the DHCI architecture decreases to 56%, 59%, and 52%, respectively. Under a specific workload, we compare the running time for the traditional Hadoop cluster with that of the DHCI architecture under load pressure; they are close to 2:1 (e.g., 913 vs. 507, 633 vs. 371, and 2450 vs. 1266, respectively). The results clearly show that with the same available computation resource of the cloud platform, the DHCI architecture can significantly reduce the running time and thus improve the operational efficiency of tasks.

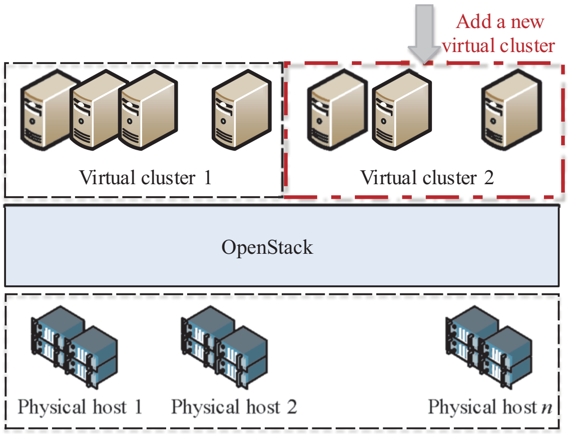


Fig. 7 Adding new virtual cluster(s) onto IaaS cloud platform for relieving load pressure.

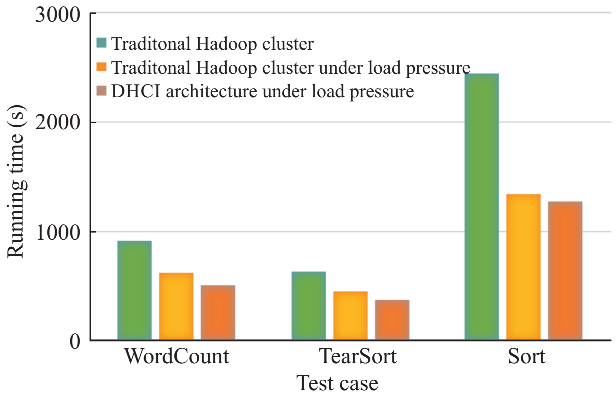


Fig. 8 Comparison of running time under two architectures with different scales of virtual clusters.

4.3 Comparison of CPU utility rate and load average between the two architectures

In this section, we evaluate the performance parameters of each physical host when the WordCount task is executed in the DHCI architecture.

Figure 9 shows the CPU utility rates of four physical hosts (Ph1, Ph2, Ph3, and Ph4) from the 1st minute to the 14th minute. In this experiment, we set 50% as a threshold. Once the CPU utility rate is greater than this threshold, we regard the workload of the physical host as excessive. In this case, the resource on this physical host will not be allocated for task(s) any more. From the curves in Fig. 9, we find that in the 3rd, 6th, 7th, 8th, and 10th minutes, the CPU utility rates of part of the physical hosts exceeded the preset threshold of 50%. For example, in the 3rd minute, both the CPU utility rates of Ph1 and Ph2 are 53%, which is greater than 50%. After 1 min, the CPU utility rates of Ph3 and Ph4 increase gradually to accomplish workload sharing. Correspondingly, their values of Ph1 and Ph2 decrease to 43% and 49%, respectively.

Figure 10 shows the load average of four physical hosts during a period of 14 min. This experiment sets

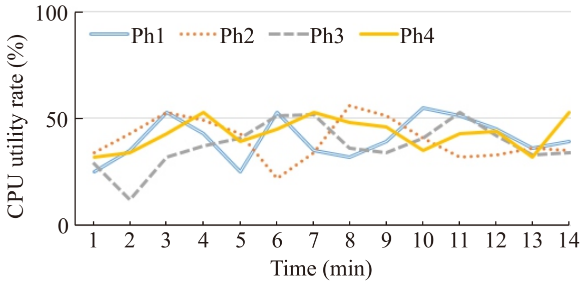


Fig. 9 CPU utility rates of multiple physical hosts in the DHCI architecture.

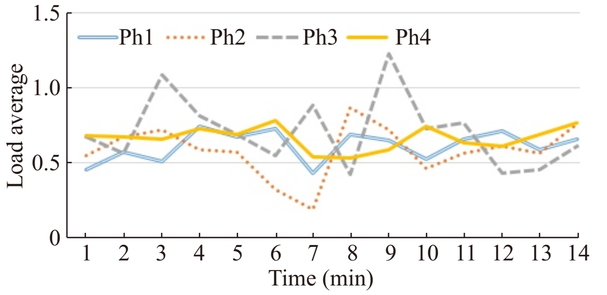


Fig. 10 The load averages of multiple physical hosts in the DHCI architecture.

70% as a threshold. Similarly, when the load average exceeds this threshold, the workload of the physical host is regarded as severe. This striking trend in Fig. 7 shows that when a physical host's workload rises and exceeds the preset threshold, other physical hosts in the same virtual cluster will take part in workload balancing and thus improve the efficiency.

Figures 9 and 10 give the performance parameters: CPU utility rate (CPU for short) as well as the load average (LA for short) of 4 physical hosts during a period of 14 minutes for two different architectures. We take their averages for each physical host respectively, as illustrated in Figs. 11 and 12. Then, we calculate their variances to reflect the fluctuations in workload of multiple physical hosts. The variance of CPU utility rate in the traditional Hadoop cluster and the DHCI architecture are 0.196 and 0.1, respectively. The efficiency of the cluster load balance in the DHCI architecture is superior to that in the traditional Hadoop cluster. From the perspective of load average, the similar conclusion can be drawn.

4.4 Test on data locality optimization

To verify the effectiveness of the data locality optimization strategy, we also use the benchmark test cases, namely WordCount, TeraSort, and Sort with 2 GB data volume. The data in Fig. 13 shows the difference in testing data from the DHCI architecture with and without data locality optimization, respectively. We can conclude that the time taken to execute these tasks with data locality optimization is less than that without data locality optimization while under the same data volume condition.

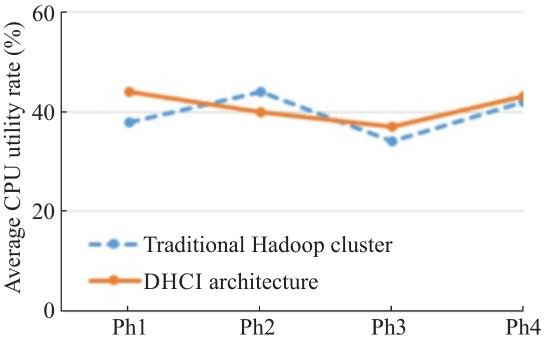


Fig. 11 Average CPU utility rate of each physical host in the two architectures.

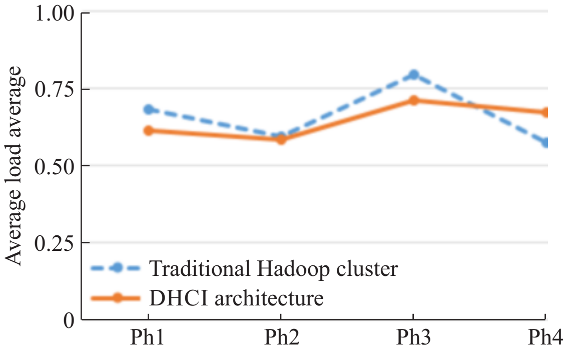


Fig. 12 Average load average of each physical host in the two architectures.

5 Conclusion

In this study, we designed a novel dynamic Hadoop cluster IaaS architecture by introducing the following four modules: monitoring, scheduling, VM management, and VM migration modules. In particular, we proposed resource scheduling and data locality solutions. We assessed the efficiency of our solutions on the aforementioned virtual Hadoop cluster. Convincing experimental results show that our solutions can effectively balance the load and improve system performance.

Acknowledgment

This work was supported by the Open Project Program

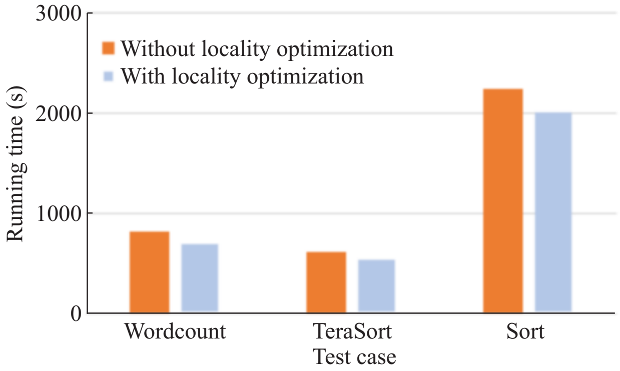


Fig. 13 Comparison of running time in the DHCI architecture with and without data locality optimization.

of Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks (No. WSNLBKF201503), the Fundamental Research Funds for the Central Universities (No. 2016JBM011), Fundamental Research Funds for the Central Universities (No. 2014ZD03-03), the Priority Academic Program Development of Jiangsu Higher Education Institutions, and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology.

References

[1] L. Z. Wang, J. Tao, H. Marten, A. Streit, S. U. Khan, J. Kolodziej, and D. Chen, MapReduce across distributed clusters for data-intensive applications, in *IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, Shanghai, China, 2012, pp. 2004–2011.

[2] M. Armbrust, A. Fox, R. Griffith, and M. Zaharia, A view of cloud computing, *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] H. Kang, Y. Chen, J. L. Wong, R. Sion, and J. Wu, Enhancement of Xen's scheduler for MapReduce workloads, in *Proceedings of the 20th International ACM Conference on High Performance Distributed Computing*, New York, NY, USA, 2011, pp. 251–262.

[4] T. Sandholm and K. Lai, Dynamic proportional share scheduling in Hadoop, in *Proceedings of the 15th International Conference on Job Scheduling Strategies for Parallel Processing*, Springer-Verlag, 2010, pp. 110–131.

[5] B. Sharma, R. Prabhakar, S. H. Lim, M. T. Kandemir, and C. R. Das, MROrchestrator: A fine-grained resource orchestration framework for MapReduce clusters, in *IEEE 5th International Conference on Cloud Computing*, Hawaii, HI, USA, 2012, pp. 1–8.

[6] P. Lama and X. Zhou, AROMA: Automated resource allocation and configuration of MapReduce environment in the cloud, in *Proceedings of the 9th International Conference on Autonomic Computing*, 2012.

[7] L. Y. Zuo, Z. B. Cao, and S. B. Dong, Virtual resource evaluation model based on entropy optimized and dynamic weighted in cloud computing, (in Chinese), *Journal of Software*, vol. 24, no. 8, pp. 1937–1946, 2013.

[8] Q. Liu, W. D. Cai, J. Shen, Z. J. Fu, X. D. Liu, and N. Linge, A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, *Security and Communication Networks*, vol. 9, no. 17, pp. 4002–4012, 2016.

[9] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy, S. Shenkeret, and I. Stoica, Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling, in *Proceedings of the 5th European Conference on Computer Systems*, ACM, 2010, pp. 265–278.

[10] H. Jin, X. Yang, X. H. Sun, and I. Raicu, ADAPT: Availability-aware MapReduce data placement for non-dedicated distributed computing, in *IEEE International Conference on Distributed Computing Systems*, Macau, China, 2012, pp. 516–525.

[11] A. F. Thaha, M. Singh, A. H. M. Amin, N. M. Ahmad, and S. Kannan, Hadoop in OpenStack: Data-location-aware cluster provisioning, in *IEEE the 4th World Congress on Information and Communication Technologies*, 2014, pp. 296–301.

[12] Z. Fadika and M. Govindaraju, DELMA: Dynamically elastic MapReduce framework for CPU-intensive applications, in *The 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011, pp. 454–463.

[13] Y. Kong, M. J. Zhang, and D. Y. Ye, A belief propagation-based method for task allocation in open and dynamic cloud environments, *Knowledge-based Systems*, vol. 115, pp. 123–132, 2016.

[14] C. L. Cheng, J. Li, and Y. Wang, An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing, *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 28–39, 2015.

[15] R. Q. Sun, J. Y. Yang, Z. Gao, and Z. Q. He, A virtual machine based task scheduling approach to improve data locality for virtualized Hadoop, in *IEEE/ACIS 13th International Conference on Computer and Information Science*, 2014, pp. 297–302.

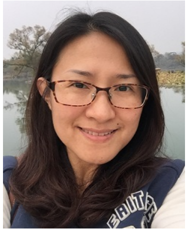
[16] R. Q. Sun, J. Yang, A. Gao, and Z. Q. He, A resource scheduling approach to improving data locality for virtualized Hadoop cluster, (in Chinese), *Journal of Computer Research and Development*, vol. 51, no. Suppl., pp. 189–198, 2014.

[17] X. P. Bu, J. Rao, and C. Z. Xu, Interference and locality-aware task scheduling for MapReduce applications in virtual clusters, in *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, New York, NY, USA, 2013, pp. 227–238.

[18] Q. Zhang, L. Liu, Y. Ren, K. Lee, Y. Z. Tang, X. Zhao, and Y. Zhou, Residency aware inter-VM communication in virtualized cloud: Performance measurement and analysis, in *Proc of the 6th IEEE International Conference on Cloud Computing*, 2013, pp. 204–211.

[19] Z. J. Fu, X. M. Sun, Q. Liu, L. Zhou, and J. G. Shu, Achieving efficient cloud search services multi-keyword eanked aearch over encrypted cloud data supporting parallel computing, *IEICE Transactions on Communications*, vol. E98-B, no. 1, pp. 190–200, 2015.

[20] H. Y. Sun, W. X. Xie, X. Yang, and K. Lu, A load balancing algorithm based OH parallel computing entropy in HPC, (in Chinese), *Journal of Shenzhen University Science and Engineering*, vol. 24, no. 1, pp. 64–68, 2007.



Dan Tao is currently a professor with School of Electronic and Information Engineering, Beijing Jiaotong University, China. She received the PhD degree from Beijing University of Posts and Telecommunications, China, in 2007. She was a visiting scholar in the Department of Computer Science, Illinois Institute of Technology, USA, during 2010-2011. She has published more than 50 papers in the academic journals and conferences. Her research interests include wireless networks and cloud computing.

Zhaowen Lin is currently an associate professor with Institute of Network Technology, Beijing University of Posts and Telecommunications, China. He received the PhD degree from Beijing University of Posts and Telecommunications, China, in 2008. His research interests include future Internet key technology and network security.

Bingxu Wang is currently a postgraduate student of School of Electronic and Information Engineering, Beijing Jiaotong University, China. He received the bachelor degree from Shandong University of Science and Technology, China, in 2013. His research interest is cloud computing.