A Virtual Machine Consolidation Framework for MapReduce Enabled Computing Clouds

Zhe Huang, Danny H.K. Tsang, James She
Department of Electronic & Computer Engineering
The Hong Kong University of Science and Technology
Email: {ecefelix, eetsang, eejames}@ust.hk

***Abstract*—In nowadays computing clouds, it is of the cloud providers' economic interests to correctly consolidate the workload of the virtual machines (VMs) into the suitable physical servers in the cloud data center in order to minimize the total maintenance cost. However, during the consolidation process, sufficient protection should be provided to the service level agreement (SLA) of the VMs. In this paper, the VM consolidation problem for MapReduce enabled computing clouds has been investigated. In the MapReduce enabled computing clouds, MapReduce jobs are carried out by homogeneous MapReduce VM instances that have identical hardware resource. Two resource allocation schemes with corresponding SLA constraints for the MapReduce VMs and the non-MapReduce VMs are proposed. Based on these schemes, the VM consolidation problem is modeled as an integer nonlinear optimization problem and an efficient algorithm has been proposed to locate its solutions. The results show that better VM consolidation performance can be achieved by colocating MapReduce instances together with non-MapReduce instances in the same set of physical servers.**

I. Introduction

The MapReduce framework [1] popularized by Google is emerging as one of the most powerful distributed data analysis tools that enable users to process a huge amount of data within a relatively short time period. The parallel computing structure of the MapReduce framework is able to significantly reduce the total runtime of the data analysis process when a large number of MapReduce nodes are used. Nowadays, computing clouds often offer users pay-as-you-go virtual machines (VMs) leasing schemes that allow users to easily rent out a large number of VMs with minimal cost. This makes a large scale computing cloud like Amazon EC2 [2] a perfect candidate for hosting the MapReduce applications. Several MapReduce frameworks specifically designed for cloud computing platforms have already been proposed [3] [4] [5].

In a MapReduce enabled computing cloud, a MapReduce cluster is set up using various VMs hosted in the cloud data center. We refer to the VMs that host the MapReduce nodes as the MapReduce instances and the VMs that host other cloud applications as the non-MapReduce instances. To minimize the maintenance cost of the cloud data center, VMs' workload is consolidated by correctly grouping compatible VMs together and assigning them to the suitable physical servers. However, consolidating the VM workload will heat up the competition for hardware resource among VMs. As pointed out in [6] and [4], the parallel computing nature of the MapReduce framework dictates that the performance of the MapReduce applications depends on the slowest MapReduce instances in the cloud data center and it is also very sensitive to the I/O bandwidth[1] (e.g., disk I/O, network bandwidth) competition among other colocated non-MapReduce instances. As a result, MapReduce instances should be hosted by homogeneous and isolated VMs that have dedicated I/O bandwidth reserved separately from other non-MapReduce VMs. However, it may not be beneficial for the cloud providers to create such isolated VMs since the reserved I/O bandwidth cannot be shared among other VMs. A new VM consolidation scheme specifically designed for MapReduce enabled computing clouds which consolidates VMs based on the characteristics of MapReduce framework is required.

In this paper, the VM consolidation problem for MapReduce enabled computing clouds is investigated. Resource allocation schemes and service level agreement (SLA) models specially designed for MapReduce instances and non-MapReduce instances are introduced. The SLA model for MapReduce instances reserve I/O bandwidth from physical servers and distribute it evenly among all the MapReduce instances. The servers' remaining I/O bandwidth will be shared among the non-MapReduce instances in a statically multiplexing manner. Based on the proposed SLA models, the VM consolidation problem for MapReduce enabled computing clouds is modeled as an integer nonlinear optimization problem. An efficient algorithm is proposed to obtain practical VM consolidation solutions in a prompt manner. The results presented in this paper demonstrate that significant improvement of the VM consolidation performance can be achieved by carefully configuring the MapReduce instances and colocating them with the non-MapReduce instances. The numerical results suggest that the proposed VM consolidation algorithm can achieve suboptimal VM consolidation performance with minimal algorithm complexity.

The rest of the paper is organized as follows. Section II describes the background and the requirements of the MapReduce framework for a scientific computing cloud. In Section III, SLA models for MapReduce and non-MapReduce instances and the VM consolidation framework are introduced. Section IV presents the numerical results for the proposed

[1] Although in this paper our discussion focuses on the I/O bandwidth of the physical servers, the proposed models and algorithm are generic and can be applied to any type of hardware resource such as CPU and memory resource in the physical servers.

solution. Finally, we conclude the paper in Section V.

II. Background

Among all the cloud based MapReduce frameworks, Hadoop [7] stands out distinctively as one of the most widely deployed MapReduce frameworks. Mainly developed by Yahoo, Hadoop is quickly adopted by Facebook and eventually used by Amazon to implement its own version of the MapReduce enabled computing cloud called ElasticMapReduce (EMR) [8]. In this paper, we focus our discussion on the MapReduce frameworks similar to Hadoop.

One of the most attractive features of the MapReduce framework is its capability of processing large amount of data in a rapid and parallel manner. The MapReduce framework divides a computational task into multiple mapping and reducing subtasks which are assigned to the mapper nodes and the reducer nodes in the MapReduce cluster individually. The MapReduce operation is terminated when all the subtasks are finished. As a result, the runtime of a MapReduce job greatly depends on the slowest bottleneck MapReduce instances which are sometimes referred to as stragglers. Several speculative execution strategies [6] have been proposed to minimize the performance impacts of the straggler problem.

In the design of the parallel computing structure of the MapReduce framework, the MapReduce task scheduler implicitly assumes that MapReduce nodes are homogeneous so that they receive the same amount of hardware resource. It is also assumed that roughly the same amount of computational workload is assigned to the mapper nodes and the reducer nodes. With these assumptions, the MapReduce subtasks processed by different MapReduce instances progress at roughly the same speed so that all the MapReduce instances finish their works at roughly the same time [6]. It has been pointed out that the performance of the MapReduce operation can be degraded significantly if the above homogeneous assumptions are violated [6]. These homogeneous assumptions can be very fragile for public computing clouds in which hardware resources are shared among colocated VMs. Competition for I/O bandwidth among VMs will cause performance fluctuation for the MapReduce instances.

Since the mapper and the reducer nodes in Hadoop use Hadoop distributed file system (HDFS) for data storage and exchange, the I/O bandwidth consumption of the MapReduce instances will remain at a constant level when they are busy. It is more preferable to reserve I/O bandwidth for these MapReduce instances in order to fulfill the homogeneous assumptions. Nowadays hardware virtualization technologies allow flexible resource allocation among VMs so that a portion of the hardware resources in a server (e.g., CPU cycles, memory and I/O bandwidth) can be reserved for different VMs [9] [10]. It is assumed that each physical servers in the cloud data center will reserve a portion of its I/O bandwidth for the MapReduce instances it hosts. The reserved I/O bandwidth is divided and assigned to these MapReduce instances evenly. The reserved I/O bandwidth will not be shared among any VM (not even among the MapReduce instances). The SLA of the MapReduce instances is considered to be well protected when all the MapReduce instances receive an I/O bandwidth reservation larger than a threshold so that the MapReduce job can be finished within the expected time period. The rest of the physical servers' I/O bandwidth will be shared among the non-MapReduce instances in a statistical multiplexing manner. The SLA of the non-MapReduce instances is said to be well protected if the resource outage probability (i.e., the probability that any non-MapReduce instance fails to obtain I/O channel because of I/O bandwidth outage) is smaller than a threshold. Similar SLA model has also been adopted in previous works [11].

Given the above SLA models for the MapReduce and the non-MapReduce instances, the VM consolidation problem is equivalent to the problem of minimizing the total maintenance cost of the cloud data center while hosting a given number of non-MapReduce instances and carrying out MapReduce jobs without violating their corresponding SLA constraints.

III. VM Workload Consolidation

A. Resource Allocation for MapReduce Instances

Due to the facts that non-MapReduce VM instances are heterogeneous and their workload cannot be adjusted by the cloud data center, a VM consolidation scheme that purely focuses on non-MapReduce VM instances may suffer from severe resource wastage. Such wastage happens when a server's hardware resource is under-utilized but no extra non-MapReduce instance can be further assigned to this server without violating the corresponding SLA constraints. Compared with non-MapReduce VM instances, MapReduce instances present several attractive properties like static I/O bandwidth consumption and fine granular scalability (i.e., the MapReduce jobs can be divided and carried out by arbitrary number of MapReduce VM instances.). As a result, a VM consolidation scheme which colocates MapReduce instances with non-MapReduce instances can effectively reduce the resource waste and hence improve the servers' hardware utilization rate and reduce the maintenance cost of the cloud data center.

To consolidate VMs optimally, resource allocation scheme for both MapReduce instances and non-MapReduce instances should be thoroughly studied. For a MapReduce job submitted to the computing cloud, the total I/O bandwidth requirement can be estimated as the total volume of the data generated during the job divided by the expected running time of the job. Let T denote this estimated total I/O bandwidth requirement. When L number of MapReduce instances are created to carry out the MapReduce job, each MapReduce instance is required to have $T/L+\theta$ amount of I/O bandwidth where θ is a constant level of I/O overhead introduced by the operational system (OS) running in the virtual machines. We refer to θ as the OS overhead. By changing L, one can adjust the minimum I/O bandwidth requirement of the MapReduce instances. Denote $\mathcal{S}_m$ to be the set of physical servers in a cloud data center that are turned on. To simplify the presentation, let $N = |\mathcal{S}_m|$. For each server i, let u_i represent its available I/O bandwidth. Because MapReduce

instances are colocated with the non-MapReduce instances, the I/O bandwidth of server i is partitioned into two parts with a partition coefficient π_i where $0 \leq \pi_i \leq 1$. $\pi_i u_i$ is the amount of I/O bandwidth reserved for MapReduce instances. Denote m_i to be the number of MapReduce instances hosted by server i. For the homogeneous MapReduce instances hosted by server i, the $\pi_i u_i$ amount of reserved I/O bandwidth is evenly distributed to these m_i number of MapReduce instances. Given the I/O bandwidth partition vector $\Pi = (\pi_1, ..., \pi_N)$ and the number of MapReduce instances L, the resource allocation problem for the MapReduce instances can be modeled as a max-min programming problem as follows.

$$\begin{aligned} \max_{m_i, i \in \mathcal{S}_m} \quad & min(\frac{\pi_1 u_1}{m_1}, ..., \frac{\pi_N u_N}{m_N}) \\ s.t. \quad & \sum_{i \in \mathcal{S}_m} m_i = L \\ & m_i \in \mathbb{Z}^+ \cup \{0\}, \forall i \in \mathcal{S}_m. \end{aligned}$$

$\mathbb{Z}^+$ in the above programming problem denotes the positive integer set. The optimal value returned by the above programming problem represents the minimum amount of I/O bandwidth that can be guaranteed for all the MapReduce instances. We refer to this optimal value as the effective I/O bandwidth. As a result, the SLA constraint for the MapReduce instances can be mathematically modeled as the condition that the effective I/O bandwidth must be larger than $T/L+\theta$. In the VM consolidation process, it is impractical to rely on solving the above nonlinear integer programming problem to obtain the effective I/O bandwidth. Defining the SLA constraint using a tight lower bound of the effective I/O bandwidth which can be obtained within $O(1)$ complexity can greatly simplify the later VM consolidation process. Denote r^* to be the effective I/O bandwidth, the following lemma and theorem provide such a bound.

Lemma 1. *There exists an optimal solution vector $M^* = (m_1^*, ..., m_N^*)$ of the MapReduce instances resource allocation problem so that $r^* \geq \pi_i u_i - m_i^* r^*, \forall i \in \mathcal{S}_m$.*

Proof: Let $M' = (m_1', ..., m_N')$ be one of the optimal solution vectors of the MapReduce instances resource allocation problem. Define $\mathcal{S}_1 = \{\forall i \in \mathcal{S}_m | \pi_i u_i - m_i' r^* = 0\}$ and $\mathcal{S}_2 = \{\forall i \in \mathcal{S}_m | \pi_i u_i - m_i' r^* \geq r^*\}$. Consider the modified solution vectors $(m_1', ..., m_i' - 1, ..., m_j' + 1, ..., m_N'), \forall i \in \mathcal{S}_1, \forall j \in \mathcal{S}_2$. Because r^* is the optimal value, the corresponding objective values achieved by these modified solution vectors are always smaller or equal to r^*. From the definition of $\mathcal{S}_2$, it is easy to show that $(\pi_j u_j)/(m_j' + 1) \geq r^*, \forall j \in \mathcal{S}_2$. It shows that $(\pi_j u_j)/(m_j' + 1)$ is not the minimum among other $(\pi_i u_i)/(m_i'), \forall i \in \mathcal{S}_m$. Such modification for $j \in \mathcal{S}_2$ does not alter the objective value achieved. Meanwhile, $(\pi_i u_i)/(m_i' - 1) > (\pi_i u_i)/(m_i') \geq r^*, \forall i \in \mathcal{S}_1$. It shows that the objective value achieved by the modified solution vectors remains r^* and the modified solution vectors are again optimal solution vectors. By repeating this solution vector modification procedure, one can remove elements in $\mathcal{S}_2$ one by one and create an optimal solution vector $M^* = (m_1^*, ..., m_N^*)$ so that $r^* \geq \pi_i u_i - m_i^* r^*, \forall i \in \mathcal{S}_m$. ■

Theorem 2. *The effective I/O bandwidth is lower bounded by $(\frac{L}{L+N})(\frac{\sum_{i \in \mathcal{S}_m} \pi_i u_i}{L})$.*

Proof: Let $M^* = (m_1^*, ..., m_N^*)$ denote the optimal solution vector in Lemma 1 with $r^* \geq \pi_i u_i - m_i^* r^*, \forall i \in \mathcal{S}_m$ and $\sum_{i \in \mathcal{S}_m} m_i^* = L$. It can be shown that

$$\begin{aligned} \frac{\sum_{i \in \mathcal{S}_m} \pi_i u_i}{L} - r^* &= \frac{\sum_{i \in \mathcal{S}_m} \pi_i u_i}{L} - \frac{r^* L}{L} \\ &= \frac{\sum_{i \in \mathcal{S}_m} (\pi_i u_i - m_i^* r^*)}{L} \\ &\leq \frac{N r^*}{L}. \end{aligned}$$

The above inequality produces a lower bound of r^* to be $(\frac{L}{L+N})(\frac{\sum_{i \in \mathcal{S}_m} \pi_i u_i}{L})$. ■

Based on Theorem 2, a simplified SLA constraint for MapReduce instances can be proposed as

$$(\frac{L}{L+N})(\frac{\sum_{i \in \mathcal{S}_m} \pi_i u_i}{L}) \geq \frac{T}{L} + \theta. \quad (1)$$

The above SLA constraint implicitly represents a physical condition that the total I/O bandwidth required by all the MapReduce instances (i.e., T) must be less than the total amount of I/O bandwidth reserved for all the MapReduce instance (i.e., $\sum_{i \in \mathcal{S}_m} \pi_i u_i$) deflated by a coefficient of $L/(L+N)$ minus $L\theta$ amount of OS overhead consumed by the L number of MapReduce instances. The deflation can be explained by the quantization error between the aggregated I/O bandwidth requirement of the MapReduce instances and the amount of I/O bandwidth reserved by the host servers. Since MapReduce instance workload is added to each physical server in a discrete manner, it is unavoidable that a small portion of the servers' I/O bandwidth cannot be utilized because no MapReduce instance has an I/O bandwidth requirement small enough to fit in the gap. The altered SLA constraint in 1 also represents the relationship between the aggregated I/O bandwidth required by the MapReduce instances and the I/O bandwidth reserved by the servers as a function of L. When L is large, VMs' OS overhead dominates and hardware resource utilization rate becomes low. When L becomes smaller, each MapReduce instance requires a larger amount of I/O bandwidth which makes the quantization error larger. The hardware resource utilization rate becomes low again. To balance the tradeoff between the OS overhead and the quantization error, the number of MapReduce instances created should be optimally determined so that the MapReduce instances can be maximally consolidated (i.e., achieving the highest hardware resource utilization rate).

B. Resource Multiplexing among Non-MapReduce Instances

Since the I/O bandwidth consumption behavior of the heterogeneous non-MapReduce instances is not static or synchronized, consolidation of non-MapReduce instances can

be achieved by allowing the non-MapReduce instances to consume I/O bandwidth in a statistical multiplexing manner. It is assumed that the heterogeneous non-MapReduce instances hosted by server i consume I/O bandwidth from $(1-\pi_i)u_i$ amount of I/O bandwidth according to arbitrary independent random distributions. Since nowadays the virtualization platforms in the computing clouds limit the maximum amount of hardware resource that can be accessed by each VM [9] [10], it can be assumed that the random I/O bandwidth consumption of the non-MapReduce instances is uniformly upper bounded and fulfills the Lindeberg's condition [12]. As a result, when a physical server is powerful and able to host a reasonably large number of non-MapReduce instances, the random distribution of the aggregated I/O bandwidth consumption from all the non-MapReduce instances hosted by this server approaches to a Gaussian distribution according to the central limit theorem. The experiments conducted in [13] demonstrate that a reasonably large number of VMs can be hosted simultaneously by a moderate server without severe performance impacts.

Because non-MapReduce instances share I/O bandwidth in a statistical multiplexing manner, the SLA constraint of the non-MapReduce instances requires that the probability of the non-MapReduce instances failing to acquire any I/O channel must be smaller than a threshold. We refer to this probability as the resource outage probability and we refer to the threshold as the maximum resource outage probability. The resource outage probability can be modeled as the tail probability of a standard normal distribution. Denote $\mathcal{Z}_i$ to be the set of non-MapReduce instances that physical server i hosts. Let μ_j and σ_j^2 be the mean and the corresponding variance of the I/O bandwidth consumption of non-MapReduce instance j. Let Δ be the maximum resource outage probability. The non-MapReduce instance SLA constraint for server $i \in \mathcal{S}_m$ can be modeled as the following Q-function inequality.

$$Q(\frac{(1-\pi_i)u_i - \sum_{j\in\mathcal{Z}_i}\mu_j}{\sqrt{\sum_{j\in\mathcal{Z}_i}\sigma_j^2}}) \leq \Delta. \tag{2}$$

The above SLA constraint model becomes less accurate when the physical servers are not powerful enough to host enough number of non-MapReduce instances. In this case, a non-parametric kernel density estimation model that estimates the resource outage probability based on the past history data of I/O bandwidth consumption is more preferable. However, a non-parametric kernel density estimation model will significantly complicate the problem and extra workload will be introduced to all the non-MapReduce instances since each of the instances needs to report its past I/O bandwidth consumption history to the kernel density estimator.

C. VM Consolidation

The VM consolidation process optimally consolidates MapReduce and non-MapReduce VM instances into a group of physical servers with minimum maintenance cost given the condition that the SLA constraints are not violated. The process mainly consists of three major tasks:

1) selecting appropriated physical servers
2) grouping and assigning non-MapReduce instances
3) determining the appropriated number of MapReduce instances and assigning them to the physical servers.

To minimize the maintenance cost of the cloud data center, servers that have high performance and low cost are preferred. To protect the SLA constraints, VM instances must be grouped compactly so that the aggregated I/O bandwidth consumption does not exceed the total I/O bandwidth provided by the physical servers. Since non-MapReduce instances are heterogeneous and consume I/O bandwidth in a statistical multiplexing manner, grouping non-MapReduce instances according to the concordance of the resource consumption behavior can effectively reduce the aggregated I/O bandwidth consumption. However, homogeneous MapReduce instances consume I/O bandwidth in a static manner. Grouping the MapReduce instances according to the concordance of the resource consumption behavior does not affect the aggregated I/O bandwidth consumption at any time. The consolidation process for the MapReduce instances is achieved by fine tuning the number of MapReduce instances created in order to find a good balance between the OS overhead (i.e., θ) and the quantization error so that the aggregated I/O bandwidth required is minimized.

Denote $\mathcal{Z}$ to be the set that contains all the non-MapReduce instances. Define $\mathcal{S}$ to be the server set which includes all the physical servers in the resource pool of the cloud data center. Denote c_i to be the maintenance cost of server i. Let d_i be the binary server selection variable where $d_i = 1$ indicates that physical server i is turned on to host VM instances. Note that the number of server used to host VMs can be calculated as $\sum_{i\in\mathcal{S}} d_i$. Let e_{ij} be the binary VM assignment variable where $e_{ij} = 1$ represents that non-MapReduce instance j is hosted by server i. By modifying the SLA constraints using the binary variables, the VM consolidation problem can be modeled as

$$\begin{aligned}
\min_{L,\pi_i,d_i,e_{ij},\forall i,j} \quad & \sum_{i\in\mathcal{S}} c_i d_i \\
s.t. \quad & (\frac{L}{L+\sum_{i\in\mathcal{S}} d_i})(\frac{\sum_{i\in\mathcal{S}}\pi_i u_i d_i}{L}) \geq \frac{T}{L} + \theta \\
& Q(\frac{(1-\pi_i)u_i d_i - \sum_{j\in\mathcal{Z}}\mu_j e_{ij}}{\sqrt{\sum_{j\in\mathcal{Z}}\sigma_j^2 e_{ij}}}) \leq \Delta, \forall i \in \mathcal{S} \\
& d_i \in \{0,1\}, \forall i \in \mathcal{S} \\
& e_{ij} \in \{0,1\}, \forall i \in \mathcal{S}, \forall j \in \mathcal{Z} \\
& \sum_{i\in\mathcal{S}} e_{ij} = 1, \forall j \in \mathcal{Z} \\
& 0 \leq \pi_i \leq 1, \forall i \in \mathcal{S}.
\end{aligned}$$

The above mixed Integer nonlinear programming problem is among the class of theoretically difficult problems (NP-complete). Obtaining the exact optimal solutions is not practical for a moderate size problem. Alternatively, heuristic algorithm that can achieve suboptimal performance in a short time is more preferable.

By analyzing the optimal solutions' structure, one can learn that the servers selected in the optimal solution mainly consist of a group of servers with low marginal cost (i.e., maintenance cost divided by the available I/O bandwidth, $c_i/u_i, \forall i \in \mathcal{S}$). Occasionally, a small number of servers with low I/O bandwidth and low maintenance cost are also selected. This happens when the majority of the VMs have already been maximally consolidated into servers with low marginal cost and the remanding VMs can fit into servers with low I/O bandwidth. Although utilizing servers with low I/O bandwidth and low maintenance cost can reduce the total maintenance cost of the cloud data center marginally, it also limits the possibility of further consolidating more VMs without switching to a new set of servers. As a result, an easier and more flexible server selection strategy focuses on selecting those servers with low marginal cost only.

Grouping and assigning VM instances depend on the corresponding SLA constraints of the VMs. For non-MapReduce instances hosted by server i, the corresponding SLA constraint can be rewritten as

$$(1-\pi_i)u_i d_i \geq \sum\nolimits_{j\in\mathcal{Z}} \mu_j e_{ij} + Q^{-1}(\Delta)\sqrt{\sum\nolimits_{j\in\mathcal{Z}} e_{ij}\sigma_j^2} \quad (3)$$

where $Q^{-1}(x)$ is the inverse Q-function. By summating up the above inequality for all the servers in $\mathcal{S}$, a relationship between the aggregated I/O bandwidth requirement of the non-MapReduce instances and the VMs' I/O bandwidth consumption behavior can be expressed as follows.

$$\sum_{i\in\mathcal{S}}(1-\pi_i)u_i d_i \geq \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{Z}}\mu_j e_{ij} + Q^{-1}(\Delta)\sum_{i\in\mathcal{S}}\sqrt{\sum_{j\in\mathcal{Z}}\sigma_j^2 e_{ij}}. \quad (4)$$

The aggregated I/O bandwidth required by all the non-MapReduce instances can be minimized if the right hand side of (4) is minimized. Note that $\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{Z}}\mu_j e_{ij} = \sum_{j\in\mathcal{Z}}\mu_j$ because of the $\sum_{i\in\mathcal{S}} e_{ij} = 1$ constraint in the VM consolidation problem. Also note that $\sum_{i\in\mathcal{S}}\sqrt{\sum_{j\in\mathcal{Z}}\sigma_j^2 e_{ij}}$ is simply a sum of l_2-norms. Since the summation of l_2-norms is a Schur-convex function [14], the right hand side of (4) is minimized when the non-MapReduce instances are grouped so that $\sum_{j\in\mathcal{Z}}\sigma_j^2 e_{ij}$ approaches to the same level for all the server $i \in \mathcal{S}_m$.

For the MapReduce instances, the corresponding SLA constraint can be rewritten as

$$\sum_{i\in\mathcal{S}}\pi_i u_i d_i \geq \frac{L+\sum_{i\in\mathcal{S}} d_i}{L}T + (L+\sum_{i\in\mathcal{S}} d_i)\theta. \quad (5)$$

Given the server selection variables $\{d_i, \forall i \in \mathcal{S}\}$, the right hand side of the above inequality represents the minimum aggregated I/O bandwidth required by the L number of MapReduce instances. To consolidate the workload of the MapReduce instances, L is carefully calculated so that the right hand side of the above inequality is minimized. The optimal value of L can be easily located using a simply numerical search algorithm in one dimension (e.g., linear search).

Based on the above observations, we propose an efficient VM consolidation algorithm as follows. First, the algorithm sorts the servers in $\mathcal{S}$ according to their marginal cost in an ascending order. The non-MapReduce instances in $\mathcal{Z}$ are also sorted according to their variance in an ascending order. Without loss of generality, it is assumed that $i < j$ indicates $\sigma_i^2 \leq \sigma_j^2$ and $c_i/u_i \leq c_j/u_j$. In each iteration of the VM consolidation algorithm, the first server in the current $\mathcal{S}$ is selected. The VM consolidation algorithm then repeatedly assigns a pair of non-MapReduce instances that are selected from the head and the tail of current $\mathcal{Z}$ (i.e., the non-MapReduce instances in the current $\mathcal{Z}$ that have the largest and the smallest variance values) to the selected server until the server cannot afford to host any more non-MapReduce instances. Whenever a pair of non-MapReduce instances are assigned to the server, they are removed from $\mathcal{Z}$. If the selected server does not have enough I/O bandwidth to support both non-MapReduce instances in the selected pair, only the non-MapReduce instance with larger variance is assigned to the server. If no more non-MapReduce instance can be hosted by the selected server, the server will be removed from $\mathcal{S}$ and a new server with the minimum marginal cost in the current $\mathcal{S}$ will be selected. Note that the remaining I/O bandwidth of the removed server can still be used to host MapReduce instances. The above process is able to effectively even out the value of $\sum_{j\in\mathcal{Z}}\sigma_j^2 e_{ij}$ among all the active servers so that the aggregated I/O bandwidth consumption of the non-MapReduce instances is minimized. Recall that $\mathcal{Z}_i$ is the set of non-MapReduce instances that physical server i hosts and $j \in \mathcal{Z}_i$ if and only if $e_{ij} = 1$. Also recall that $\mathcal{S}_m$ is the server set that contains all the servers used to host VMs. After all the non-MapReduce instances are successfully assigned to their corresponding servers, $\mathcal{S}_m$ contains all the servers that are used to host the non-MapReduce instances. The residual bandwidth for server $i \in \mathcal{S}_m$ is calculated as

$$v_i = u_i - \sum\nolimits_{j\in\mathcal{Z}_i}\mu_j + Q^{-1}(\Delta)\sqrt{\sum\nolimits_{j\in\mathcal{Z}_i}\sigma_j^2}. \quad (6)$$

At this point, a one dimensional linear search algorithm is carried out to determine the integer L value that minimize

$$\frac{L+|\mathcal{S}_m|}{L}T + (L+|\mathcal{S}_m|)\theta. \quad (7)$$

If the minimum value of (7) is larger than $\sum_{i\in\mathcal{S}_m} v_i$, an empty server j with lowest marginal cost in the current $\mathcal{S}$ is added to $\mathcal{S}_m$. The corresponding residual bandwidth of the newly added server is set to u_j. The algorithm repeatedly adds new servers into $\mathcal{S}_m$ until the corresponding minimum value of (7) is smaller than $\sum_{i\in\mathcal{S}_m} v_i$. Denote L^* to be the integer value that minimizes (7) when the minimum value of (7) is smaller than $\sum_{i\in\mathcal{S}_m} v_i$ for the first time. The minimum I/O bandwidth that each MapReduce instance require can be calculated as $T/L^* + \theta$. As a result, each server i in $\mathcal{S}_m$ will be assigned

with

$$\lfloor \frac{v_i}{T/L^* + \theta} \rfloor \tag{8}$$

number of MapReduce instances.

The proposed VM consolidation algorithm can be described using the following pseudocode.

Algorithm 1 VM Consolidation Algorithm

Sort servers according to c_i/u_i in an ascending order
Sort VMs in according to σ_j^2 in an ascending order
Select the first server with smallest c_i/u_i, index it to be i
while $|Z| \neq 0$ **do**
  Calculate $k = argmax(g : \sum_{j=1}^{g}(\mu_j + \mu_{|Z|+1-j}) + Q^{-1}(\Delta)(\sum_{j=1}^{g}(\sigma_j^2 + \sigma_{|Z|+1-j}^2))^{\frac{1}{2}} \leq u_i)$
  if $\sum_{j=1}^{k+1} \mu_j + Q^{-1}(\Delta)(\sum_{j=1}^{k+1} \sigma_j^2)^{\frac{1}{2}} + \sum_{j=|Z|+1-k}^{|Z|} \mu_j + Q^{-1}(\Delta)(\sum_{j=|Z|+1-k}^{|Z|} \sigma_j^2)^{\frac{1}{2}} \leq u_i)$ **then**
    Insert the first $k+1$ and the last k non-MapReduce instances in the current $\mathcal{Z}$ to $\mathcal{Z}_i$
    Remove these instances from $\mathcal{Z}$
  else
    Insert the first and the last k non-MapReduce instances in the current $\mathcal{Z}$ to $\mathcal{Z}_i$
    Remove these instances from $\mathcal{Z}$
  end if
  Insert server i into $\mathcal{S}_m$
  Remove server i from $\mathcal{S}$
  Select the first server in the current $\mathcal{S}$, index it as i again
end while
Update $v_i, \forall i \in \mathcal{S}_m$ according to (6)
Search for L that minimizes (7)
while The minimum value of (7) is larger than $\sum_{i \in \mathcal{S}_m} v_i$ **do**
  Insert the first server in $\mathcal{S}$ into $\mathcal{S}_m$, index it as i
  Set $v_i = u_i$ for the newly added server
  Search for L that minimizes (7) again
end while
Set L^* to be the last L that minimizes (7)
Assign $\lfloor \frac{v_i}{T/L^*+\theta} \rfloor$ number of MapReduce instances to server $i \in \mathcal{S}_m$

IV. Numerical Analysis

In the following experiments, the available I/O bandwidth for the heterogeneous physical servers and their corresponding maintenance cost are randomly selected from a uniform distribution over [50, 250]. The heterogeneous non-MapReduce instances are assigned with $\mu_j, \forall j \in \mathcal{Z}$ randomly selected from a uniform distribution over [0, 30] and corresponding $\sigma_j^2, \forall j \in \mathcal{Z}$ randomly selected from a uniform distribution over [0, 10]. Δ is set to 0.1 and θ is set to 2.

In the first experiment, the VM consolidation performance of the proposed algorithm is examined. To provide a performance benchmark, the VM consolidation problem presented in Section III-C is solved optimally using a commercial problem solver named Lingo. Due to Lingo's limited capability to solve large scale integer nonlinear programming problems, 15 trials of small scale problems that consist of 10 physical servers, 50 non-MapReduce instances are examined. It is assumed that a MapReduce job is submitted to the computing cloud and it requires in total 150 unit of I/O bandwidth (i.e., $T = 150$). Figure 1 shows the relative increase of the total maintenance cost obtained by our proposed algorithm compared with the total maintenance cost achieved by Lingo's solutions. On average, the total maintenance cost of the servers obtained by Lingo is 1.42% smaller than that obtained by the proposed VM consolidation algorithm. The results demonstrate that the proposed algorithm can achieve a VM consolidation performance very close to the optimal cases.

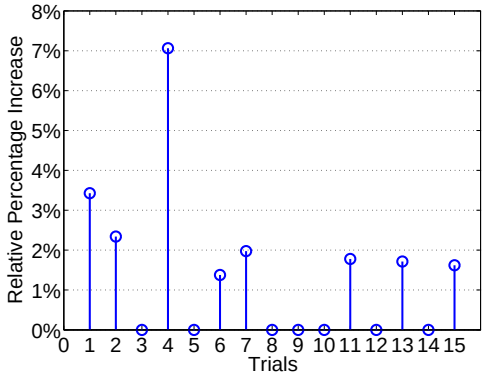

Fig. 1. Relative increase of the total maintenance cost

To demonstrate how colocating MapReduce instances with non-MapReduce instances together improves the VM consolidation performance, two naive VM consolidation schemes are implemented. In both naive schemes, MapReduce instances and non-MapReduce instances are hosted separately using different servers. Moreover, non-MapReduce instances are randomly grouped together. The first scheme allocates VM instances to servers with high I/O bandwidth in order to minimize the total number of physical servers used. We refer to this scheme as the resource-first scheme. The second scheme assigns VM instances to low marginal cost servers to reduce the total maintenance cost. We refer to this scheme as the marginal-cost-first scheme. To illustrate that the scale of the computing clouds also has great impacts to the VM consolidation performance, several experiments with computing clouds that have different scale are performed. To simply the presentation, a scale coefficient η is introduced to represent that there are in total 30η physical servers and 100η non-MapReduce instances in the computing clouds. It is assumed that the MapReduce job requires in total 500η amount of I/O bandwidth. The value of η is varied in the range of [1, 10]. For each η value, the experiment is repeated for 1000 times and the average total maintenance cost of the cloud data center is reported. Figure 2 and Figure 3 show the relative decrease of the total maintenance cost obtained by the proposed algorithm compared with the total maintenance cost achieved by the naive schemes. On average 29% of the total maintenance

cost can be saved using our proposed algorithm compared with the resource-first scheme and on average 7% of the total maintenance cost can be saved compared with the marginal-cost-first scheme. The noticeable VM consolidation performance improvement is expected because the proposed VM consolidation algorithm colocates the MapReduce instances with the non-MapReduce instances. The MapReduce instances work as gap fillers so that any improper consolidation decision for non-MapReduce instances (e.g., imbalanced VMs grouping and assignment) can be mended by correctly configuring and consolidating the MapReduce instances. Note that in both cases, the percentage of the improvement drops when the scale of the computing clouds becomes larger. This result can be explained by the fact that grouping a large number of VM instances randomly can already achieve acceptable resource multiplexing gain when the scale of the computing cloud is large.

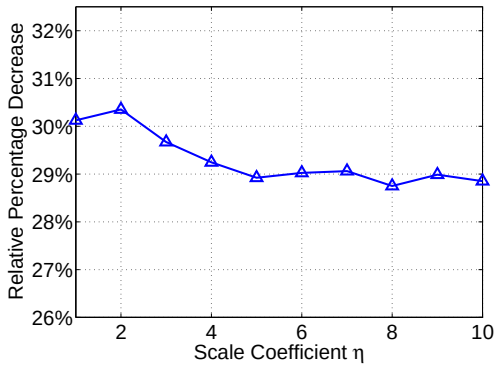

Fig. 2. Relative performance improvement compared to the resource-first scheme

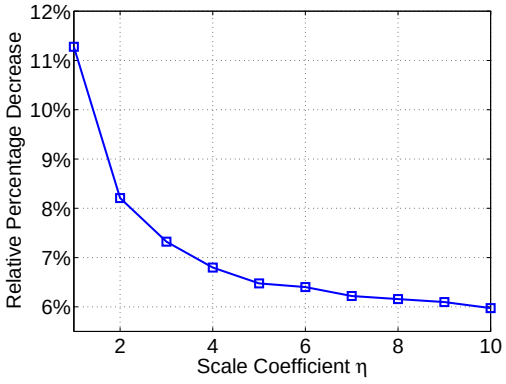

Fig. 3. Relative performance improvement compared to the marginal-cost-first scheme

The next experiment quantitatively examines the MapReduce instances' ability of improving the VM consolidation performance. It is assumed that 1000 non-MapReduce instances will be hosted by a cloud data center that consists of 300 physical servers with heterogeneous I/O bandwidth. The OS overhead value, θ, is varied in the range of [1, 10]. Meanwhile the total I/O bandwidth required by the MapReduce job, T, is varied from 5000 to 500 accordingly so that the total amount of I/O bandwidth consumed by all the MapReduce instances remains in a constant level. The experiments also include a non-colocated VM consolidation scheme. The non-colocated VM consolidation scheme follows the same VM consolidation procedure as presented in Algorithm 1 except that the MapReduce instances and the non-MapReduce instances are hosted by different servers (e.g., new empty servers are selected to host MapReduce instances.). Figure 4 shows the average total maintenance cost achieved for different θ values. As indicated in the figure, the total maintenance cost remains at a constant level when the MapReduce instances are separately hosted. However, a noticeable reduction of the total maintenance cost can be obtained when colocation of MapReduce instances is allowed. This result proves that colocating MapReduce instances can be very effective in improving the utilization of the I/O bandwidth for the physical servers in the cloud data center. It can also be observed that when θ increases, colocating MapReduce instances is less effective in reducing the total maintenance cost for the cloud data center. It is mainly because when the OS overhead increases, the proposed algorithm tends to reduce the number of MapReduce instances created (i.e., L becomes smaller in the process of minimizing (7)). In this case, each MapReduce instance requires a higher amount of I/O bandwidth and becomes more difficulty to be consolidated because of the larger quantization error.

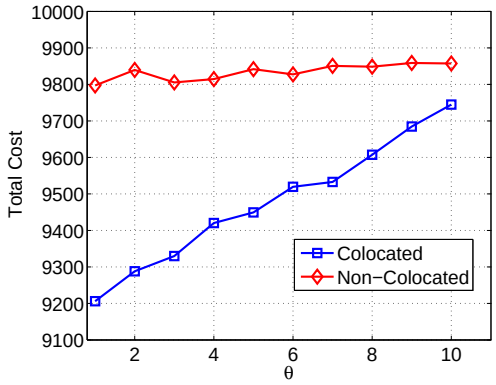

Fig. 4. Total maintenance cost of the cloud data center for different θ values

V. Conclusion and Future Works

In this paper, the VM consolidation problem for MapReduce enabled computing clouds is investigated. The VM consolidation problem tries to assign virtual machines including both MapReduce instances and non-MapReduce instances generated by other cloud applications to the correct set of physical servers in the cloud data center in order to minimize the cloud data center maintenance cost. The VM consolidation problem is modeled as an integer nonlinear optimization problem. An efficient algorithm is proposed to obtain the corresponding solutions. The numerical results show that the VM assignment solutions obtained by the proposed algorithm can efficiently

reduce the total maintenance cost of the cloud data center. The solutions obtained by the proposed algorithm also suggest that allowing MapReduce instances and non-MapReduce instances to be collocated on the same physical servers is economically beneficial to the cloud providers.

It is of our interest to apply the proposed models and algorithm to a scientific computing cloud in order to perform empirical studies that examine how well the proposed algorithm can consolidate VM workload in reality. In the future, we plan to construct a small scale private computing cloud that implements our proposed VM consolidation algorithm. Extensive empirical experiments will be carried out with different system parameters.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[2] Amazon Elastic Compute Cloud, Amazon.com, Inc. [Online]. Available: http://aws.amazon.com/ec2/

[3] S. Pallickara, J. Ekanayake, and G. Fox, "Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 31 2009.

[4] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Mapreduce in the clouds for science," *Cloud Computing Technology and Science, IEEE International Conference on*, vol. 0, pp. 565–572, 2010.

[5] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 222 –229.

[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 29–42. [Online]. Available: http://portal.acm.org/citation.cfm?id=1855741.1855744

[7] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Jun. 2009. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0596521979

[8] Amazon ElasticMapReduce, Amazon.com, Inc. [Online]. Available: http://aws.amazon.com/elasticmapreduce/

[9] *Xen Hypervisor 3.0 Users' Manual*, 2009, whitepaper, Citrix Systems. [Online]. Available: http://bits.xensource.com/Xen/docs/user.pdf

[10] *vSphere Resource Management Guide*, 2009, whitepaper, VMware Inc.

[11] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *ICAC '10: Proceeding of the 7th international conference on Autonomic computing*. New York, NY, USA: ACM, 2010, pp. 11–20.

[12] R. B. Ash and C. Doléans-Dade, *Probability and Measure Theory*, 2nd ed. Academic Press, 2000.

[13] *The True Cost of Virtual Server Solutions*, 2009, technical Report, Taneja Group. [Online]. Available: http://www.vmware.com/files/pdf/True_Cost_Virtual_Server_Solutions_Taneja_Mar09.pdf

[14] A. W. Marshall, I. Olkin, and B. C. Arnold, "Inequalities: Theory of majorization and its applications, second edition," *International Statistical Review*, vol. 79, no. 2, pp. 293–293, 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1751-5823.2011.00149_19.x