Divide the Task, Multiply the Outcome: Cooperative VM Consolidation

Mina Sedaghat[a], Francisco Hernández-Rodriguez[a], Erik Elmroth[a], Sarunas Girdzijauskas[b]
[a] *Dept. of Computing Science, Umeå University, Sweden*
[b] *KTH - Royal Institute of Technology, Sweden*
{mina, hernandf, elmroth}@cs.umu.se, sarunasg@kth.se

***Abstract*—Efficient resource utilization is one of the main concerns of cloud providers, as it has a direct impact on energy costs and thus their revenue. Virtual machine (VM) consolidation is one the common techniques, used by infrastructure providers to efficiently utilize their resources. However, when it comes to large-scale infrastructures, consolidation decisions become computationally complex, since VMs are multi-dimensional entities with changing demand and unknown lifetime, and users often overestimate their actual demand. These uncertainties urges the system to take consolidation decisions continuously in a real time manner.**

In this work, we investigate a decentralized approach for VM consolidation using Peer to Peer (P2P) principles. We investigate the opportunities offered by P2P systems, as scalable and robust management structures, to address VM consolidation concerns. We present a P2P consolidation protocol, considering the dimensionality of resources and dynamicity of the environment. The protocol benefits from concurrency and decentralization of control and it uses a dimension aware decision function for efficient consolidation. We evaluate the protocol through simulation of 100,000 physical machines and 200,000 VM requests. Results demonstrate the potentials and advantages of using a P2P structure to make resource management decisions in large scale data centers. They show that the P2P approach is feasible and scalable and produces resource utilization of 75% when the consolidation aim is 90%.

***Keywords*-Cloud computing; Peer to Peer; Gossip protocols; VM consolidation; Resource management;**

I. Introduction

Cloud providers are widely using the virtualization technologies for the efficient utilization of their available resources. Virtual Machines (VMs) are treated as blocks that can be put together on a Physical Machine (PM) or can be moved from one PM to the other to maintain a high resource utilization in a data center. Consolidation of multiple VMs on a single machine helps cloud providers to increase their resource utilization and decrease their power consumption, specially as users often overestimate their actual demand. It also helps opening up capacity to run services with special constraints or the ones that are larger to fit in a small fragmented resource.

However, the fact that VMs are serving a changing demand and they have unknown arrivals and lifetimes forces the system to continuously make the consolidation decisions and re-assign the resources in a real time manner. It should be noted that both VMs and PMs are multi-dimensional entities, i.e. they are defined in terms of CPU and memory capacity, and they have specific shapes. Thus, maintaing the efficiency of utilization when the entities are multi-dimensional and their shapes are changing over-time can be challenging.

Moreover, the applications and management decisions become more complex and larger in scale, such that the traditional centralized or hierarchical approaches cannot scale with the number of PMs in the data center. To support scalability, they either need to compromise the quality of the solution to the resource management problem, to keep the response time within an acceptable time frame or alternatively partition the infrastructure statically. The latter restricts the system from using the resources efficiently due to the lack of coordination between partitions. Centralized approaches also require a continuous fine grained monitoring data which is often huge and expensive to collect and process [1].

However, decentralized approaches allow the complex resource management decisions to be taken in collaboration, by a number of autonomous entities working toward a management objective. In such systems, the objective is achieved as the emergent behavior of a number of autonomous entities, acting as processing units, making management decisions within their own local scope. Besides, such systems are more robust since each autonomous entity operates independently from others' failures or departures.

In this paper, we investigate how resource management problem in cloud data centers can be formulated and addressed using a decentralized approach. We specifically tackle the VM consolidation problem, to achieve energy efficiency and increase resource utilization. Extending our resource management framework introduced in [2], we propose a P2P gossip-based protocol for VM placement and consolidation, considering the multi-dimensionality of the PMs and VMs. Each peer in the P2P framework, cooperates with its fellow peers, to improve its status with a new state of a greater value, thus moving toward a more efficient state. The peer uses a dimension aware decision function to quantify its state and determine the right actions. Besides, each peer only monitors its own resources and a small-sized metadata is exchanged only to the peer's immediate neighbors. We also perform an extensive study on feasibility and performance of the proposed approach when scale matters.

We show that the P2P approach is feasible and scalable

and it produces an almost-optimal VM placement for the experimented scale of 100,000 PMs and 200,000 VM requests when the load is dynamic. The system benefits from a high degree of concurrency and decentralization of control with no central bottleneck. This help the system to have a low computation time in making placement decisions for the mentioned scale. This advantage is essential for real time management of a dynamic environment such as cloud infrastructures. We also believe that delegating the complex decisions to the autonomous entities would eliminate cumbersome configuration settings that are required in the centralized approaches. It is an important advantage, since these configuration values, i.e, low utilization thresholds, offload trigger points or monitoring intervals are usually hard to devise, not applicable for all the data center's entities and they are sensitive to the load changes.

II. Consolidation problem

Assume that a data center consists of n PMs. Each PM has a CPU and memory capacity of C_{PM} and M_{PM}. The CPU and memory utilization of each PM i at time t is denoted as $c_i(t)$ and $c_i(t)$, respectively. It is assumed that all PMs are homogeneous, although the formulation can also be generalized for heterogeneous machines. The data center offers l VM types, where each VM has an expected computational and memory capacity of CPU_l and Mem_l. These VMs are categorized in 3 different types of *compute optimized*, *memory optimized* and *general purpose*, due to their usability scenario. It is also assumed that m VM placement request arrive to the system during the data center operation time. Each request j demands a specific VM type l and consumes a $CPUDemand_j(t)$, $MemDemand_j(t)$ at each timestep.

The goal is to achieve energy efficiency, $E(t)$, in the data center by minimizing the total power consumption, modeled as a function shown in Equation (1).

$$Minimize\ E(t) = \sum_{i=1}^{n_{active}} P_i(t) \tag{1}$$

subject to:

$$\sum_{j=1}^{m} Res_j(t) \leq n_{active} \times (\alpha \times Capacity_{PM}) \tag{2}$$

where $Res_j(t)$ is the utilized capacity by *VM request*$_j$ at time t, n_{active} is the number of non-idle PMs, α is a utilization factor, defined to avoid performance degradations caused by interferences among the consolidated VMs, and finally, $Capacity_{PM}$ is the capacity of the PM in terms of CPU and memory. For the sake of brevity, we generalize the notations of $Res_j(t)$ and $Capacity_{PM}$ to address the concept of *resource*. However, whenever these values are referred in the paper, we calculate the value for each resource, CPU or memory, independently with respect to their relevant values. Moreover, P_i(t) is the consumed power of PM_i, and it is a linear function of the fixed consumption for PM when it is in idle state and additional power usage proportional to PM's CPU utilization [3] and it is calculated by Equation (3).

$$P_i(t) = (P_{max} - P_{idle}) \times c_i(t) + P_{idle} \tag{3}$$

where P_{max} is the power consumption at maximum performance and P_{idle} is the PM's power consumption when idle.

One important point to be considered is that resources are multi-dimensional. Hence, for a mixed workload, consisting of both compute and memory optimized VMs, the placement algorithm should be dimension aware, meaning that the proportionality of resource usage (CPU vs. memory) in each PM should be considered while deciding on the placement [4], [5]. To best utilize the capacity of a physical machine along two dimensions (CPU/Memory), it is desirable that at each point of time, the resources being proportionally utilized in both dimensions. The unbalanced utilization of the resources in each physical machine will lead to wasting the space in one of the dimensions. Therefore, to ensure the efficient utilization, we also strive to minimize an imbalance factor, formulated as:

$$Minimize\ Imb_{dc}(t) = (\sum_{i=1}^{n} |c_i(t) - m_i(t)|)/n \tag{4}$$

where $Imb_{dc}(t)$ is the average of data center's imbalance at time t, n is the total number of PMs, both idle and non-idle, $c_i(t)$ and $m_i(t)$ are the relative values for CPU and memory utilization of PM_i at time t.

III. VM Placement through gossiping

We consider the VM placement problem as a distributed decision making process. We introduce a decentralized gossip-based protocol, where decisions are continuously being made between random pairs of cooperative agents, trying to improve a common value. The common value is defined as the total imbalance, $Imb(t)$, of each pair at the time of decision making and the goal is to reduce this imbalance by redistributing the VMs among them. The cooperative approach among two agents prevents the undesirable bounces of VMs and ensures a stable state, which is essential to avoid the redundant migrations.

A. General Architecture

A data center is a collection of PMs. In our design, each PM is considered as a peer in a P2P network. Peers are logically connected by an overlay network. The overlay is built and maintained by a peer sampling service, known as newscast [6]. Peer sampling service periodically provides each peer with a list of peers to be considered as neighbors. Each peer at each timestep only knows about k random neighbors, shaping its local view.

Each peer locally runs a resource agent responsible for monitoring its associated PM's state (CPU and memory consumption), communicating with the neighbors and processing the information received from the neighbors. Peers communicate in a gossip-based fashion via small messages that represent their state. The details about the general architecture is discussed in [2].

B. Design concerns

To investigate the most suitable solution, we review the common questions and enumerate some of the issues that should be addressed during the design.

1) **When should a consolidation be performed?** In common practices, the re-consolidation process is triggered by an event, i.e. when a PM detects a violation of an under or over-utilization threshold. However, event based algorithms, require configuring thresholds which is often tricky and complex and need a good understanding about the system and the changes in load. In such algorithms, setting the threshold too low, may lead to losing the chance of efficient consolidation of VMs, while setting it too high, may end up moving VMs constantly from one machine to the other.
   Thresholds may also be susceptible to changes since the system dynamics are changing over time. In addition to that, they are usually defined as absolute values, so a small deviation from the threshold would disqualify the PM to be triggered for re-consolidation. Hence, there would be cases, that the load of two machines can be accommodated in one, but since the load in none of the PMs fall below the threshold, this consolidation would never happen.
   The autonomy of the peers in a P2P structure allows the system to avoid the complexity of threshold setting. Using P2P structure, the peers involved in the decision process use their real time state to decide if they can benefit from a re-distribution of their VMs or not. If both peers realize that their load can be accommodated in one, within a reasonable cost of migration, then potentially a re-consolidation process can be triggered when the P2P interactions are converged. This way re-consolidation is not bound to the thresholds but it would be planned dynamically based on the peers' states. Hence, adopting P2P approach decreases the need for threshold setting and also increases the chance of triggering more efficient re-consolidations.
2) **How to re-consolidate?**
   The second question is how the re-consolidation should be performed? One strategy can be to incrementally migrate the VMs from a low-utilized machine to a PM with available capacity, with the hope of future release.
   The other option is to migrate VMs only when the complete release of the PM is guaranteed. Each of these two approaches have their own advantages and weaknesses that we will discuss in next sections.
3) **Consolidation consequences:** Different works have studied the impacts of consolidating multiple VMs on the same machine on the performance [4], [7]. The impacts are usually due to the interferences and contentions among VMs, such as cache contentions, conflicts at functional units of CPU, disk write buffer or disk scheduling conflicts [7]. However, consolidation of multiple VMs also increases the probability of overloading the PMs when the load of the VMs are changing overtime.
   The PM overloads can be handled in different ways, such as service differentiation [8], [9] or application brownouts [10], but they are usually handled through PM offloads via migration. Offload processes are often costly. The cost is due to the performance impacts or possible SLA violations and also the additional management process required to decide which VMs to be migrated to where. Based on the above-mentioned impacts, we can argue that the best strategy for VM placement and re-consolidation is the one that accommodates the demand on the fewest PMs while minimizing the over-consolidation consequences. These consequences can be listed as the longer overload time experienced by the system, possible VM rejections caused by inefficient use of resources, or even increase in power consumption.

C. Consolidation strategies

Common consolidation approaches can be enumerated as follows:

- **Incremental release using thresholds:** Using this strategy, the PM decides to migrate its VMs incrementally over time with the hope of full release after a while. The re-consolidation process is triggered when the utilization of the PM fall below the threshold. Setting the threshold in this approach is inevitable, since the number of migrations should be limited in some way.
- **One-shot merge:** The PM decides to migrate its VMs when another PM, with sufficient capacity can be found, to accommodate all its VMs. Since the decisions are based on the state of the PMs involved, there is no need for any configuration on thresholds.
- **One-shot merge + dimension aware re-distribution:** In both above-mentioned scenarios, it is assumed that accommodating VMs in fewer machines is the only factor affecting the efficient consolidation. The intuition is that the resources should either be fully utilized or be in the idle mode. However, they have ignored the fact that the requested resources have shapes and the way these shapes are put together would also affect the efficiency of utilization.
  As it is depicted in Figure 1, efficient distribution of VMs over the PMs can lead to the probability of better consolidation and thus lower power consumption. We define the efficient distribution of VMs as the one in which both CPU and memory in each PM are used in

a balanced proportion. We propose a P2P gossip-based protocol which takes into account both, the possibility of releasing a PM and re-distributing the VMs among the peers, to balance the consumption of both dimensions on the each PM.

D. Dimension aware consolidation protocol:

In this section, we introduce a gossip protocol for VM placement and consolidation. The protocol is an iterative algorithm, starting from an arbitrary initial VM placement. In this protocol, each pair of neighboring peers exchanges gossip messages. This state exchange triggers a local decision function that can possibly lead to a local state change in the associated peers. The decision function evaluates the states of the pair with respect to a common local objective and it proposes a new distribution of VMs between the associated peers that maximizes this objective considering the migration costs. In another word, the protocol continuously moves toward the optimum by iteratively applying a control operator to the peers' states and substitutes their states if the new state has a greater value. Each peer keeps track of the list of VMs assigned to them by the decision function and considers the new list as their updated state for future interactions. The protocol continues the process until it converges and no more VM is re-distributed. When the protocol is converged, a reconfiguration plan can be devised to migrate the VMs from their original location to their assigned peer. The pseudo-code of gossip interactions is illustrated in Algorithm 1.

Algorithm 1 Gossip Protocol

List of events Best deployment candidate (BestPeer) **procedure** ACTIVE THREAD **loop** wait Δ **for** Each neighbor k in the local view **do** Send *myState* Receive the state from neighbor k, $State_k$ *newState*=Decisionfunction() *myState*=Update(*newState*) **end for** **end loop** **end procedure** ——— **procedure** PASSIVE THREAD **loop** Receive $State_i$ from i Send *myState* to i *newState*=Decisionfunction() *myState*=Update(*newState*) **end loop** **end procedure**

1) Decision function:

The state exchange between the neighboring peers triggers a decision function. Using this function, the peers involved in the interaction, assesses if the re-distribution of their VMs can lead to a better consolidation. The function evaluates the possibility of re-consolidation for either of the following cases:

1) If the aggregated load of the pair, involved in the negotiation, can be accommodated in one peer, then the VMs would be deployed in one peer and the released peer is either set into the power saving mode or considered as free space to be used for other purposes in the future.
2) If not, the function assesses whether re-distribution of the VMs can lead to a more balance utilization of CPU and memory per PM, for both peers.

If any of the above-mentioned conditions are met, the function proposes a new re-distributed set of VMs for each peer, based on the following steps:

1) **Step 1: Calculate the Imbalance factor**

   The algorithm calculates the *imbalance* ratio for the possible permutations of VMs, using Equation (5). If re-distributing the VMs leads to the possible future state of (S'_1, S'_2) for $peer_1$ and $peer_2$ and imposes a CPU and memory consumption of (c_1, m_1) and (c_2, m_2) at time t', respectively, the *imbalance* ratio for this specific VM distribution is calculated as:

$$Imb(S'_1, S'_2) = \phi(S'_1, S'_2) \times (\sum_{i=1}^{2} |c_i - m_i|) \tag{5}$$

   Where $\phi(S'_1, S'_2)$=

$$\begin{cases} 0 & \text{if } (c_1 + c_2 \leq \alpha C_{PM}) \ \& \ (m_1 + m_2 \leq \alpha M_{PM}) \\ 1 & \text{if } (c_1 + c_2 > \alpha C_{PM}) \ \& \ (m_1 + m_2 > \alpha M_{PM}) \end{cases} \tag{6}$$

2) **Step 2: Calculate the migration costs**
   Since the migration of VMs is an expensive task, the algorithm considers the cost of migration when deciding on re-distributing the VMs. The cost of migration is defined as a function of migration time. The total migration time of a VM is calculated as:

$$t = t_i + t_c + t_s + t_r \tag{7}$$

   where t_i denotes the time for iteratively transferring the memory pages, t_c is the time for suspending the VM at source, t_s is the time for CPU transfer and t_r is the time for resuming the VM destination. In the above equation, t_c, t_s and t_r are rather short, however the iterative memory transfer is hard to predict and depends on the memory consumption of the VM.
3) **Step 3: Select the VM sets that maximize the *Gain***
   Finally, the algorithm selects a distribution set which leads to a better consolidation among the peers with minimum cost of migration. In the other word, the algorithm selects a distribution of VMs in which the peers can gain the most from transitioning from their current states (S_1, S_2) to a transferred state (S'_1, S'_2) with the minimum migration cost. The gain is quantified via Equation (8).

$$Gain() = \frac{Imb(S_1, S_2) - Imb(S'_1 - S'_2)}{Migration\ cost} \tag{8}$$

The general steps of the algorithm are illustrated in Algorithm 2. It should be noted that finding a VM distribution, defined by CPU and memory, is a 2D bin packing problem and the complexity of the search space grows with the number of VMs deployed in each peer.

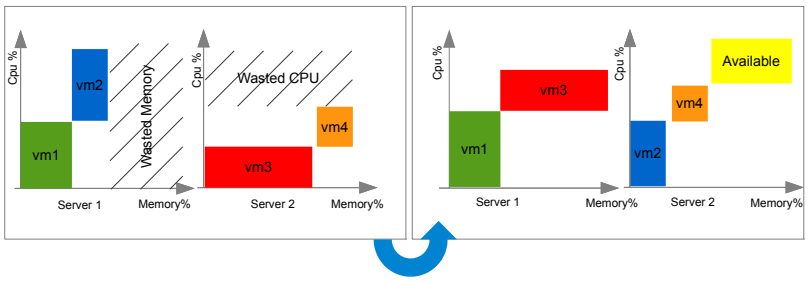


Figure 1: Redistribution of VMs can increase the balance between memory and CPU and improve the consolidation

Algorithm 2 Decision Function

Require: MyState [$c_{me}(t)$, $m_{me}(t)$, list of VM_{me}] , NeighborStates [$c_k(t)$, $m_k(t)$, list of VM_k]
  if $(c_{me} > \alpha \times C_{PM}) or (c_{me} > \alpha \times M_{PM})$ **then**
    Offload();
  else if $(c_{me}(t) + c_k(t) \leq \alpha \times C_{PM})$ & $(m_{me}(t) + m_k(t) \leq \alpha \times M_{PM}))$ **then**
    if $(m_{me}(t) < m_k(t))$ **then**
      Add the references of VMs on *me* to VM_k.
      Tag *me* as to-be-idled.
    else
      Add the references of VMs on k to VM_{me}.
      Tag k as to-be-idled.
    end if
  else if $((c_{me}(t) + c_k(t) \geq C_{PM})$ & $(m_{me}(t) + m_k(t) \geq M_{PM}))$ **then**
    Select a distribution of VMs that minimizes the total *Imbalance* for each PM in both peers with minimum memory transfer.
  end if

E. Reconfiguration Plan

In previous section, a mapping between the VMs and the PMs in the system is devised, aiming for efficiency of resource utilization and low power consumption. It has also taken into account the cost of migration in terms of the required memory transfer. However, these VMs are usually serving real time requests or in some cases they are associated with stateful data on the PMs. In these cases, the cost of migration is not just the cost of memory transfer. Therefore, a re-configuration plan should be devised to carefully consider the real time factors such as VMs states, the available network bandwidth and the durability of re-configurations.

F. Offload

As mentioned earlier, in an environment with changing demand, having overloaded PMs is inevitable. In this situations, the system should decide on how to offload the overloaded PM. When a PM becomes overloaded, it contacts its neighbors to find a suitable PM with sufficient capacity to offload its load. On the first round, the peer tries to select among the active neighbors and see if they can accommodate the extra load. In each interaction, the algorithm examines a subset of VMs on the overloaded PM in which it can be accommodated in the neighboring peer with minimum migration cost. The released capacity after migrating this set should be sufficient enough to offload the peer to fall below the overload bar. This subset is devised via an exhaustive search among the VMs currently deployed on the PM.

If none of the active neighbors have sufficient capacity for the offload, the peer activates one of its idle neighbors, if it has any, or it waits for the next cycle to receive a new set of random neighbors via the peer sampling service and repeats the above procedure.

IV. Evaluation

We evaluate the performance of three consolidation strategies, discussed in Section III, and we compare their performance with a random VM placement as the benchmark. The investigated strategies are:

1) Random VM placement
2) Incremental consolidation using thresholds
3) One-Shot Merge
4) One-Shot Merge + re-distribution w.r.t *Imbalance* ratio

A. Performance metrics

The performance of each strategy is evaluated with respect to the following metrics:

1) Data center power consumption
2) Number of active PMs
3) Imbalance rate: It is calculated as:

$$Imb_{dc}(t) = (\sum_{i=1}^{n} |c_i(t) - m_i(t)|)/n \qquad (9)$$

Algorithm 3 Offload

for Each neighbor k in the local view **do**
  if (k is active) & $c_k(t) < Overload\ bar$ & $(m_k(t) < Overload\ bar)$ **then**
    List all the possible subsets of myVMs as the possible sets to be transferred
  end if
  for Each VM subset **do**
    Calculate the total CPU and Memory demand of each VM subset as the CPU and memory to be transferred
    if $((C_k - c_k(t)) \geq transfer\ Cpu)$ & $((M_k - m_k(t)) \geq transfer\ Mem)$ & $(c_{me}(t) - transfer\ CPU < Overload\ bar)$ & $(m_{me}(t) - transfer\ Mem < Overload\ bar)$ & $(transfer\ Mem\ is\ minimum)$ **then**
      Select the subset
      overload-Resolved=true;
    end if
  end for
  if overload-Resolved==true **then**
    Break
  end if
end for
if subset!= null **then**
  Transfer the subset to the neighbor
end if
if subset= null **then**
  Activate one the idle PM
  Transfer the extra load
end if

where $Imb_{dc}(t)$ is the average of data center's imbalance at time t, n is the total number of PMs, both idle and non-idle, $c_i(t)$ and $m_i(t)$ is the CPU and memory utilization of PM_i at time t.

4) Average resource utilization: Resource utilization has a direct impact on power consumption. It is defined as the average utilization of non-idle PMs over each dimension, i.e. CPU and Memory. Hence, for each type of resource, the average resource utilization is calculated as:

$$U_{dc}(t) = \frac{\sum_{j=1}^{m} Res_j(t)}{n_{active} \times Capacity_{PM}} \quad (10)$$

Where m is the number of placement requests at time t, $Res_j(t)$ is the utilized capacity, either CPU or memory, by VM_j at time t, n_{active} is the total number of non-idle PMs, $Capacity_{PM}$ is the capacity of each PM.

5) Computation time: The computation time is the time it takes for the protocol to compute an efficient VM to PM mapping and it is measured in terms of number of cycles it takes for the protocol to converge.
6) Number of migrations
7) Overload time: The aggregated timestep that the system faces overload.
8) Number of rejected VM requests

B. Simulation setup

The evaluation is performed through simulation of a data center in PeerSim [11]. Our data center consists of 100,000 PMs, each has the capacity of 58 vCPU and 64 GB of memory. Our data center offers 6 VM types each fit to a specific use case, similar to Amazon EC2 use cases. The details on the VM characteristics is illustrated in Table I.

Category	VM name	vCPU	Memory	vCpu%	Mem%
Compute	c1.medium	15	7.44	13%	6%
Compute	c3.xlarge	30	14.88	27%	12%
Memory	m2.xlarge	6.5	17.1	6%	13%
Memory	m2.2xlarge	13	34.2	12%	27%
General	m3.medium	3.24	3.75	3%	3%
General	m3.large	6.5	7.5	6%	6%

Table I: VM types and their capacity details

The data center receives 200,000 VM requests during the simulation time. The requests types are uniformly distributed among *memory optimized*, *compute optimized* and *general purpose* VM types. They also intend to run a combination of batch jobs and stateless interactive applications. The interactive applications have long lifetimes, whole simulation run, and their CPU and memory demand is changing over time according to Equation (11), derived from an analysis on google traces introduced in [12]. The number of interactive applications is constant during the simulation run.

$$CPUDemand_j(t) = (\frac{CPU_l}{2})(1 + u_m sin(\frac{2\pi t}{86400} - 2\pi s_m) \quad (11)$$

$$MemDemand_j(t) = \beta \times CPUDemand_j(t) \quad (12)$$

where CPU_l is the expected maximum CPU demand of VM type l, u_m, $s_m \in [0,1]$ is selected uniformly at random. β is the CPU/memory capacity ratio of the VM type l, e.g. $\beta = 0.5$ for a compute optimized VM.

The second group of VMs are a set of batch jobs with the constant CPU and memory demand. The arrival rate of these requests follows a Poisson distribution with $\lambda = \frac{SimulationTime}{2}$, and they have a lifetime follows a

truncated power-law distribution with exponent 2, truncated to the length of the simulation run.

We set the utilization factor $\alpha = 0.9$, to ensures the tolerance of the system to performance degradations caused by resource contentions between neighboring VMs. We also assume a PM power consumption in the idle state is 175W and 250W when fully utilized. Each peer maintains a local view of k=20 neighbors which is being updated by the peer sampling service in each cycle. The simulation time is 1000 timesteps and the monitoring system samples the VM demands each 20th timestep. During the two consecutive monitoring, the system load is considered constant. All results presented are the average of 10 simulation runs with the same configuration.

V. Results and Discussions

A. Feasibility of P2P approach

In this section, we investigate the feasibility of having a P2P protocol to perform VM consolidation. The main concern when designing a resource manager for a large scale distributed system is the cost of the decision making, usually defined in terms of computation time and the bandwidth consumption. We investigate the cost of our P2P process, in terms of convergence cycles representing the computation time and network bandwidth consumed by gossip protocol during the decision process.

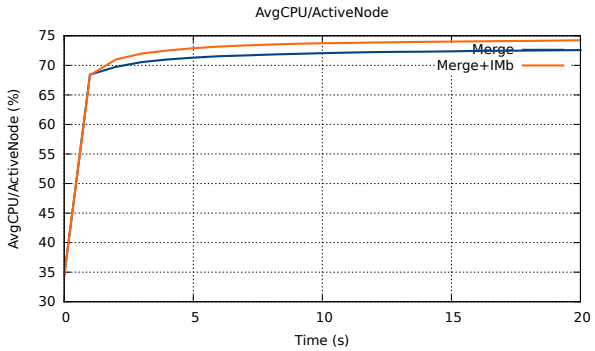
1) Convergence cycle: In the first experiment, we investigate the time required for the algorithm to reach a stable VM-PM mapping for a specific load. This time is defined in terms of the number of cycles it takes for the protocol to converge. To do this, we considered a time interval in the experiment in which the load is not changing. Hence, the number of the VMs and the total CPU and memory demand during this interval is completely constant.

Figure 2 shows the trend of the average CPU and memory utilization over the time interval with the constant load for the proposed P2P protocols with *Merge* and *Merge+Imb* strategies. The graph shows a fast convergence of the P2P protocols as it reaches a high utilization within 2 to 3 cycles, and it completely converges at 7th cycle, for the configuration specified for this experiment.

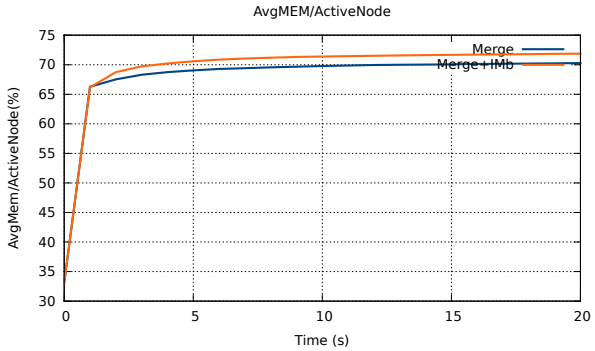
2) Bandwidth consumption: For each consolidation decision, the maximum bandwidth consumed by the gossip protocol, to reach a VM-PM mapping, is calculated by the following equation:

$$\begin{aligned} &Bandwidth\ consumption \\ &= Number\ of\ PMs \times convergence\ cycle \\ &\times Number\ of\ exchanged\ messages \\ &\times message\ size\ (Byte) \end{aligned} \tag{13}$$

The maximum bandwidth consumption required for a consolidation decision in a data center with 100,000 PMs is 160 MB =100,000 * 7 * 2 * 120 byte. However, it should be emphasized that this is the maximum bandwidth consumption since in each cycle, a number of peers are excluded from the peers who exchange messages. Messages contain metadata about the type, CPU and memory consumption of each VM, resides on each peer.



(a) Convergence on CPU utilization



(b) Convergence on memory utilization

Figure 2: Computation time in terms of convergence cycles

B. Decision strategy

Figure 4 shows a comparison on different performance metrics when each of the strategies are adopted. As it can be seen in Figures 3a, 3b and 3c, after the random placement, *Incremental* strategy on average has the lowest resource utilization, and the highest number of active PMs. It is because, by setting a threshold for triggering the re-consolidation process, some of the possible load consolidations are automatically ignored. Hence, it results in higher number of active PMs and higher power consumption.

The comparison between *Merge* and *Merge+Imb*, shown in Figures 3a, 3b, 3c and 3d, shows higher CPU and memory utilization and lower number of active PMs and lower power consumption for the latter. However, as it can be seen in Figure 3e, this result comes with the price of higher number of migrations. These extra migrations are performed to reduce the imbalance between CPU and memory utilization in each PM.

The better performance in *Merge+Imb* strategy is because

of accounting the dimensionality of the resources while re-distributing the VMs. The *Merge+Imb* strategy tries to reduce the imbalance between the CPU and memory utilization, depicted in Figure 4a. A balanced utilization of a two dimensional resource results in more efficient utilization of resources in both dimensions, thus lower rejections of VM requests, as shown in Figure 4b, and faster resolution of overloaded PMs, as shown in Figure 4c. By re-distributing the VMs to achieve a lower imbalance, we can have 28% lower rejections, 12.6 % faster offload, for a higher number of overloaded PMs and also 3.2 % lower power consumption for the cost of 25% more migration.

C. Impact of local view size (number of neighbors)

We investigate impact of the local view size on the protocols performance. In this experiment, we consider the data center with 50,000 PMs and 100,000 VM requests and a static load. We vary size of the local view to K=10, 20, 30, 50. Results, shown in Figure 5, indicate that the larger the size of the local view, the faster the protocol can converge. However, this impact become less and less significant for the sufficiently large local views.

VI. Related work

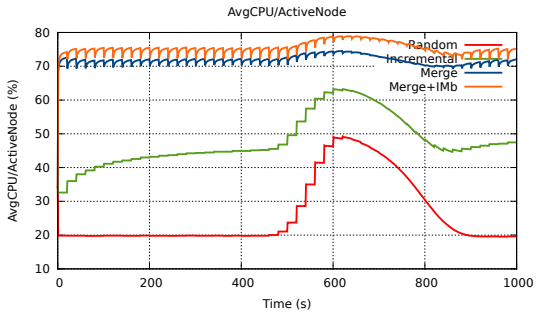
The problem of VM consolidation is discussed in different studies. The following is a brief outline:

Beloglazov and his colleagues in [13] used a *best-fit* heuristic for initial VM placements and further on, they migrate the VMs if a violation on one of the upper or lower utilization thresholds is occurred. However, they based their placement decisions on only one dimension, CPU consumption, and they also ignored the cases that changing the arrangements of VMs the PMs can lead to a more optimal utilization of resources. The algorithm is centralized and it is examined for a data center with 100 PMs.

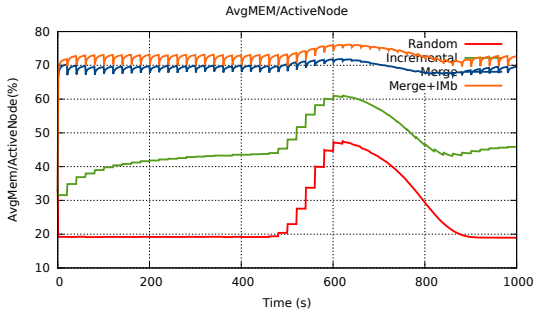
Svärd et al. [14] studied a set of heuristics to maintain the optimality of allocations via a set of actions, such as VM and PM suspend and resume, and also VM migration. These actions are triggered when either a PM crashes, or a VM arrives or exits. Their approach has a centralized design and supports up to 48 PMs. We, on the other hand, are interested in solving the problem for larger scale.

Marzolla et al. [15] presented V-Man, a decentralized algorithm, using gossip protocol, for VM consolidation. They modeled the PMs and VM requests as one dimensional entities, and they assumed that applications have constant load. Their algorithm is robust to PM and service failures. However, the model is a bit simplistic and does not cover the complexities of a multi-dimensional placement problem and the dynamic load. They also didn't consider the migration costs while deciding on which VM to migrate.

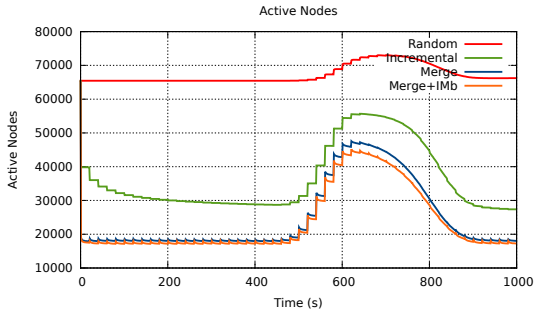
Wuhib in [16] proposed a resource allocation architecture and a gossip protocol to address a set of well known management objectives, such as fairness, balanced load,



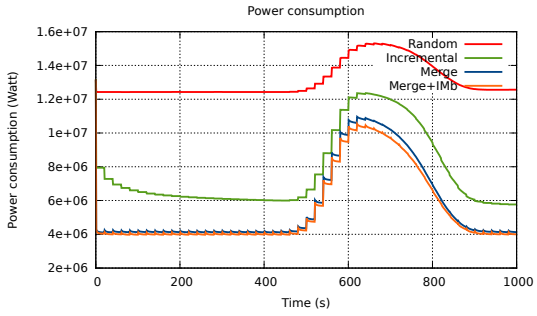
(a) Average CPU utilization



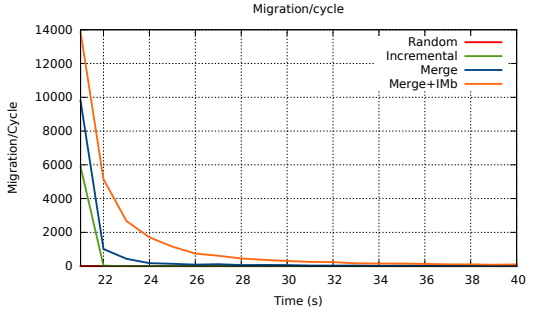
(b) Average memory utilization



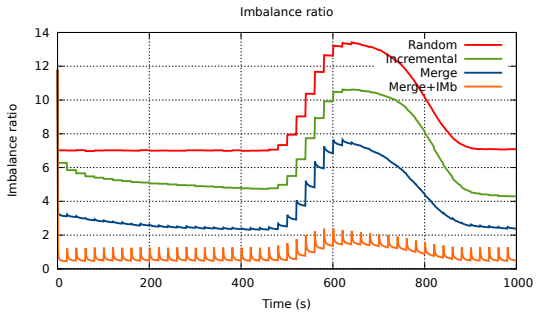
(c) Number of active PMs



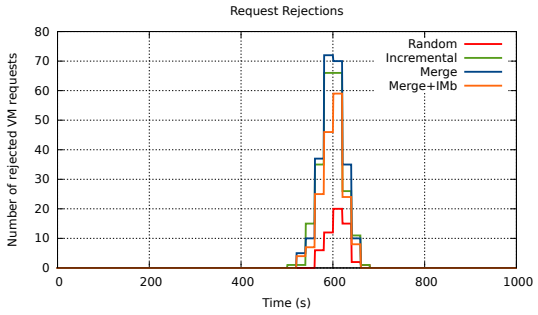
(d) Power consumption



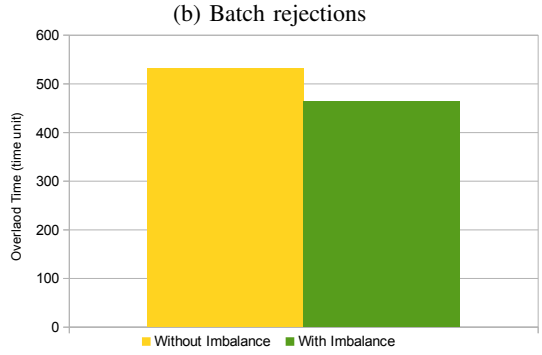
(e) Number of migrations during a constant load interval



(a) Imbalance rate

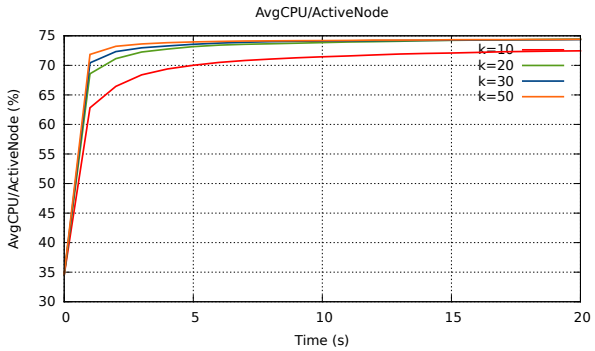


(b) Batch rejections

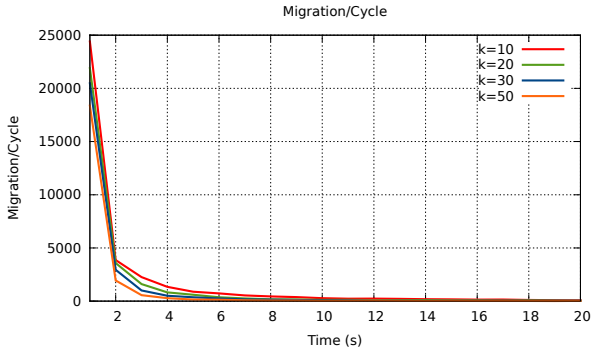


(c) Overload time

Figure 4: Comparison between different Decision strategies



(a) Impact of local view size on convergence (Ava CPU)



(b) Impact of local view size on the number of migrations

Figure 5: Impact of local view size

energy efficiency and service differentiation. In their model, machines are associated with CPU, memory, and network interface capacity. Their protocol attempts to either minimize the overload if the PM is overloaded, or to minimize the objective function under the constraint of live migration. The results shows that the protocol is effective and scales well, despite the fact that they did not consider the proportionality of resource usage per PM.

Mastroianni and his colleagues [17] proposed a probabilistic consolidation of VMs in a data center. The main idea is that a single PM is the one to decide whether they should accept or reject a VM. When a VM request arrives, the request is broadcasted by a coordinator to all the PMs and they respond the coordinator if they can accept the request. This decision is based on a Bernoulli trial, which ensures that the PMs tend to respond positively when they have intermediate utilization values for both CPU and memory. When the local decisions are made, a data center manager coordinates the decisions. Finally, the coordinator selects one of the respondents randomly. A PM also decides to migrate its VMs if it is too low utilized or high utilized, in the similar fashion.

A consolidation algorithm is proposed in [7], focusing on the trade-offs between energy consumption, performance and resource utilization. They argued that consolidation leads to performance degradations and longer execution times, therefore, to save the energy an optimal consolidation rate should be carefully determined. The intuition is similar to our approach on measuring *imbalance* rate for allocation and it is to ensure that both dimensions are equally used. However, they also discussed the problem for the one time consolidation without considering that the load of the environment is changing and the initial consolidation and also the optimal point is susceptible to change.

VII. Conclusion

In this paper, we investigated the potentials and advantages of P2P systems for making complex resource management decisions. We discussed the common design concerns regarding VM consolidation and gave a brief comparison among them. We formulated the consolidation problem in

a P2P fashion, to achieve scalability and support complex decisions in a short computational time. We also presented a P2P gossip-based protocol for multi-dimensional VM placement and consolidation, considering the changes in both VMs demand and infrastructure load.

Through extensive experiments, the results show that the P2P approach is feasible and scalable up to 100,000 PMs and 200,000 VM requests. It also produces, resource utilization of 75%, on both dimensions, CPU and memory, when the consolidation aim is 90%. This result is produced within a short computation time, less than 7 cycles, for the examined scale, which is essential to be responsive in a dynamic environment. Based on these results, we can argue that dividing management responsibilities to a set of identical autonomic elements allows the system to scale without compromising the complexity of the problem or quality of the solution, that is required to keep the response time within an acceptable time frame. Adopting a P2P approach, also eliminates cumbersome configuration settings that are required in the centralized approaches.

The observations also indicate that a balanced utilization of a two dimensional resource, in a mixed workload, results in more efficient utilization of resources in both dimensions. This leads to a lower rejections of future VM requests and faster resolution of overloaded PMs. The results shows 28% lower rejections, 12.6% faster offload, for a higher number of overloaded PMs and also 3.2% lower power consumption for the cost of 25% more migrations.

VIII. ACKNOWLEDGMENT

Financial support for this work is provided in part by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control and the Swedish Government's strategic research project eSSENCE.

REFERENCES

[1] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on openstack cloud," *Future Generation Computer Systems*, vol. 32, pp. 118–127, 2014.

[2] M. Sedaghat, F. Hernandez, and E. Elmroth, "Autonomic resource allocation for cloud data centers: A peer to peer approach.," in *The ACM Cloud and Autonomic Computing Conference (CAC'14), Accepted*, 2014.

[3] L. Minas and B. Ellison, "The problem of power consumption in servers," *Intel Corporation. Dr. Dobb's*, 2009.

[4] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder, "Validating heuristics for virtual machines consolidation," *Microsoft Research, MSRTR-2011-9*, 2011.

[5] S. He, L. Guo, M. Ghanem, and Y. Guo, "Improving resource utilisation in the cloud environment using multivariate probabilistic models," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 574–581, IEEE, 2012.

[6] M. Jelasity and M. Van Steen, "Large-scale newscast computing on the internet," tech. rep., Citeseer, 2002.

[7] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10, USENIX Association, 2008.

[8] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "Dynaqos: model-free self-tuning fuzzy control of virtualized resources for qos provisioning," in *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, pp. 1–9, IEEE, 2011.

[9] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, ACM, 2011.

[10] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: building more robust cloud applications.," in *ICSE*, pp. 700–711, 2014.

[11] A. Montresor and M. Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*, pp. 99–100, IEEE, 2009.

[12] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, ACM, 2012.

[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[14] P. Svärd, W. Li, E. Wadbro, J. Tordsson, and E. Elmroth, "Continuous datacenter consolidation," 2014.

[15] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pp. 1–6, IEEE, 2011.

[16] F. Wuhib, R. Yanggratoke, and R. Stadler, "Allocating compute and network resources under management objectives in large-scale clouds," *Journal of Network and Systems Management*, pp. 1–26, 2013.

[17] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.