Impact of Cloud Computing Virtualization Strategies on Workloads' Performance

Qingling Wang, Carlos A. Varela
Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA
http://wcl.cs.rpi.edu/
{*wangq9, cvarela*}*@cs.rpi.edu*

***Abstract*—Cloud computing brings significant benefits for service providers and users because of its characteristics: *e.g.*, on demand, pay for use, scalable computing. Virtualization management is a critical task to accomplish effective sharing of physical resources and scalability. Existing research focuses on live Virtual Machine (VM) migration as a workload consolidation strategy. However, the impact of other virtual network configuration strategies, such as optimizing total number of VMs for a given workload, the number of virtual CPUs (vCPUs) per VM, and the memory size of each VM has been less studied. This paper presents specific performance patterns on different workloads for various virtual network configuration strategies. For loosely coupled CPU-intensive workloads, on an 8-CPU machine, with memory size varying from 512MB to 4096MB and vCPUs ranging from 1 to 16 per VM; 1, 2, 4, 8 and 16VMs configurations have similar running time. The prerequisite of this conclusion is that all 8 physical processors are occupied by vCPUs. For tightly coupled CPU-intensive workloads, the total number of VMs, vCPUs per VM, and memory allocated per VM, become critical for performance. We obtained the best performance when the ratio of the total number of vCPUs to processors is 2. Doubling the memory size on each VM, for example from 1024MB to 2048MB, gave us at most 15% improvement of performance when the ratio of total vCPUs to physical processors is 2. This research will help private cloud administrators decide how to configure virtual resources for given workloads to optimize performance. It will also help public cloud providers know where to place VMs and when to consolidate workloads to be able to turn on/off Physical Machines (PMs), thereby saving energy and associated cost. Finally it helps cloud service users decide what kind of and how many VM instances to allocate for a given workload and a given budget.**

I. Introduction

Cloud computing is an emerging distributed computing paradigm that promises to offer cost-effective scalable on demand services to users, without the need for large up-front infrastructure investments [3]. Through the use of virtualization, cloud computing provides a back-end infrastructure that can quickly scale up and down depending on workload [2].

Cloud computing brings significant benefits for both service providers and service users. For service users, they pay the computing resources only on demand and without worrying about hardware, software maintenance or upgrade. For service providers, with VMs, they can shrink or expand the utilization of physical resources based on workloads' requirements. Therefore, it is possible for providers to make higher profit without affecting users' satisfaction.

Thus, it is worthy to investigate more deeply into adaptive virtual network management techniques. A high ratio of VMs to PMs enables sharing of resources yet guaranteeing isolation between workloads. In contrast to physical resources, VMs provide dynamic control on their system settings, *i.e.*, vCPUs, virtual memory. Mature virtualization infrastructures, like Xen and VMware, provide resource modulation on VMs. For example in Xen, you can statically set memory size, vCPU number for each VM in the configuration file when starting a VM; and you can also change these settings dynamically.

Although it is known that resource reallocation is possible in virtual infrastructure, no specific data shows how these settings impact workloads' performance. In previous preliminary work [11], we studied the impact of VM granularity (different VM/PM ratios) on two categories of workloads. Our performance evaluation indicated that, for tightly coupled CPU-intensive workloads, VM granularity indeed affects the performance significantly; by contrast, the impact of VM granularity on the performance of loosely coupled network intensive workloads is not critical. This paper goes beyond studying the impact of VM granularity on performance. We also consider the impact of different virtual network configurations, namely, the number of vCPUs per VM and the allocated virtual and physical memory. We also study a third category of workload: loosely coupled CPU-intensive workloads. In summary, we tackle the following problem: given a class of cloud computing workloads, what kind of virtual network configuration (*i.e.*, number of VMs, number of vCPUs per VM, memory size of each VM) optimizes the workload's performance and the utilization of physical resources?

Private cloud owners, service providers and service users will all benefit from our experiments. *Private cloud owners* have the ability to change virtual network configurations at any time. They will benefit by knowing how to configure and place VMs for limited physical resources, and when to consolidate VMs to be able to turn off servers to save energy without affecting performance. *Service providers* will benefit by knowing the proper time and way to consolidate VMs, thus to maximize profit but still keeping well-provisioned

Table I
CATEGORIES OF WORKLOADS

Workloads \ Resources	CPU-intensive	Network-intensive	
		Internal Communication	External Communication
Stars	√		
Heat	√	√	
Web System			√

VMs to guarantee service quality. Finally, *service users* will be able to analyze what kinds of (for example, small, or large? high-memory or high-CPU?) and how many VM instances maximize a workload's performance under a given budget.

This paper is organized as follows. Section II describes our system architecture. The experimental results are shown in Section III. Section IV presents the analysis of the results. Section V positions our research among related work. Finally, Section VI concludes the paper and discusses future work.

II. SYSTEM AND EXPERIMENTS SETUP

This study aims to find the relationship between virtual network configuration strategies and performance of categories of cloud computing workloads. We have built a testbed for experimentation and performance evaluation of strategies for virtual network configurations.

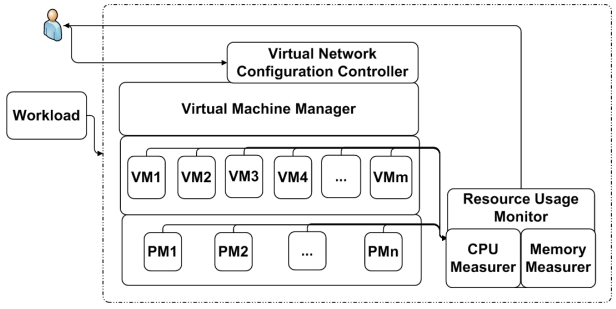


Figure 1. Benchmark architecture

A. Testbed Setup

In order to minimize the influence of other factors, for example bandwidth, and to concentrate on the relationship between the virtual and the physical layer, we deploy most of our experiments on one server machine. The server is equipped with a quad-processor, dual-core Opteron. The Opteron processors run at 2.2GHz, with 64KB L1 cache and 1MB L2 cache. The machine has 32GB RAM. The hypervisor is Xen-3.4.1 and the operating systems of the VM instances are Linux 2.6.18.8 and Linux 2.6.24-29.

B. Workloads

We study three different categories of workloads: the first one is *Stars Distance Computing*, a loosely coupled CPU-intensive workload; the second is a *Heat Distribution* problem, a tightly coupled CPU-intensive workload. They are both developed in SALSA [10], using actors [1] to perform distributed sub-tasks. Load balancing is taken into account in our experiments such that each actor is responsible for the same amount of work. We also give a brief analysis for a loosely coupled external network-intensive workload - a Web 2.0 system. Table I shows the corresponding categories of the three workloads.

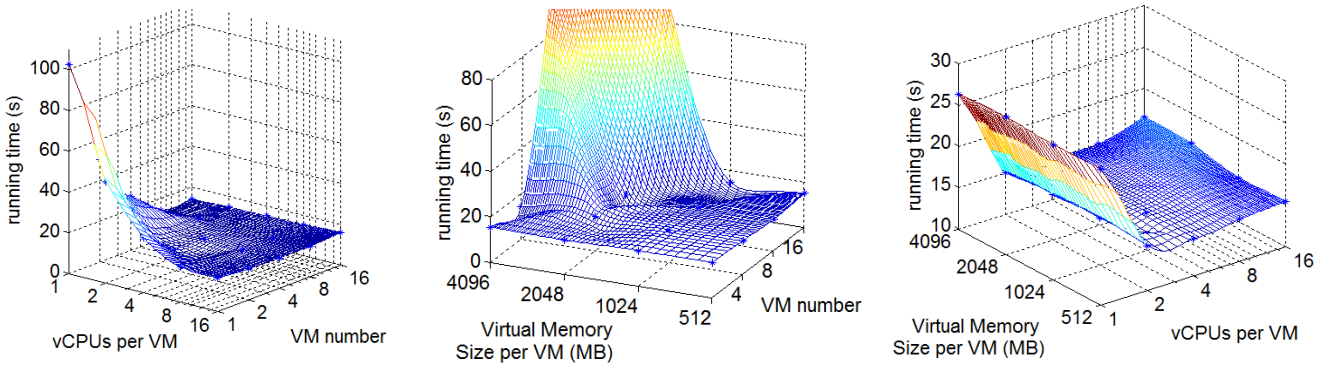
Stars Distance Computing: The goal of this problem is to analyze three-dimensional data from the Sloan Digital Sky Survey, in particular, stars from the Milky Way galaxy. The program computes the closest and farthest neighbor stars, which are the set of pairs of stars that minimize and maximize pairwise distance respectively; the ideal *hub* stars, which minimize the maximal distance to any other star; the ideal *jail* stars, which maximize the minimal distance to any other star and the ideal *capital* stars, which minimize the average distance to all other stars.

In this problem, each actor is responsible for computing a certain number of stars, it holds the immutable data it needs, and therefore the actors are independent from each other.

Heat Distribution: This problem simulates heat distribution on a two-dimensional surface. Suppose the surface is a square grid and it has some size (for example, 122 by 122). The initial temperature is 100 degrees for some coordinates (for example, $\{(y,0)|30 < y < 91\}$) and 20 degrees for all other points in the surface. The temperature for a grid element at a given time step is the average of the grid element's neighbors at the previous time step (boundary elements remain constant over time.) This workload simulates heat transfer for certain time steps to approximate equilibrium temperatures.

In this problem, each actor is in charge of specific sub-blocks of the grid and all the actors have to communicate with neighbors to get the temperatures of the boundary elements.

Web 2.0 System: The web system is built as a distributed system and serves to process continuous requests from clients. Since the server nodes in the system are independent from each other, the web system is a loosely coupled external network intensive workload.



(a) Performance change on different number of VMs and different number of vCPUs per VM (memory size of each VM is 512MB)

(b) Performance change on different number of VMs and different memory size per VM (number of vCPUs per VM is 4)

(c) Performance change on different number of vCPUs and different memory size per VM (number of VMs is 4)

Figure 2. Performance of *Stars* running on different virtual network configurations

C. Benchmark Architecture

Figure 1 shows the architecture of our performance evaluation system. We use Xen as our *virtual machine manager*. The *virtual network configuration controller* is responsible for dynamically controlling and refining the total number of VMs, vCPU number per VM, and memory size allocated to each VM. The *resource usage monitor*, running on both virtual and physical layers, periodically collects CPU and memory utilization information.

D. Experimental Design

The overall objective of our experiments is to quantify the slowdown or improvement in performance of a given workload when the virtual network configurations are changed in a fixed physical environment. We conduct overlapping experiments on the workloads with different virtual network configurations:

VMs and vCPUs: In this experiment, we set the memory size of each VM as a constant, change the total number of VMs from 1 to 16 and the number of vCPUs of each VM from 1 to 16. We collect the running time on every pair of VMs and vCPUs per VM.

VMs and virtual memory: Similar to the first experiment, we make the number of vCPUs of each VM constant, for example 4, change the total number of VMs from 1 to 16 and the memory size of each VM from 512MB to 4096MB. We collect the running time on every pair of VMs and memory size per VM.

vCPUs and virtual memory: Last, we make the total number of VMs in our experimental environment constant, for example 4, change the number of vCPUs from 1 to 16 and the memory size of each VM from 512MB to 4096MB. We collect the running time on every pair of vCPUs and memory size.

III. Experimental Results

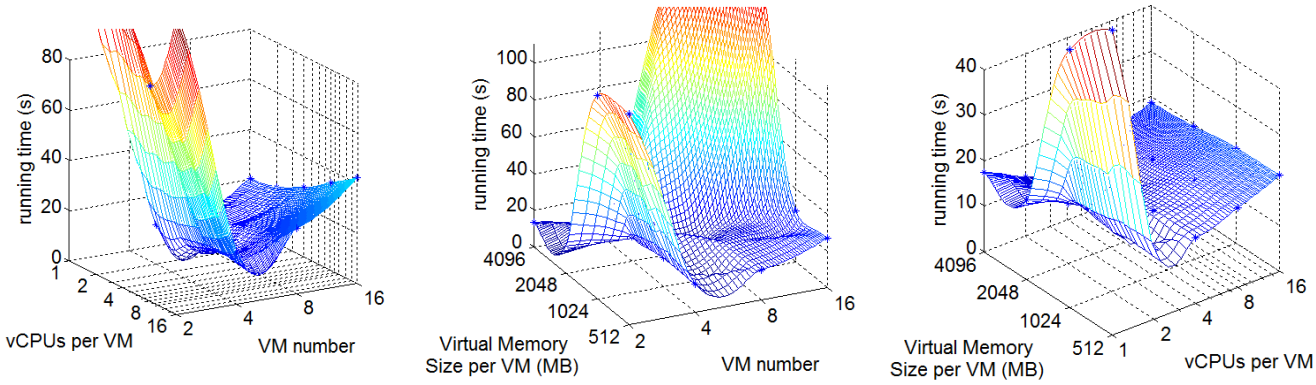
A. Loosely Coupled CPU-Intensive Workload

Figure 2(a) shows the performance change on different number of VMs with different number of vCPUs per VM. When the total number of vCPUs is less than the number of physical processors, either increasing the number of VMs or the number of vCPUs per VM will reduce the running time significantly; increasing the number of vCPUs per VM alone cannot bring down running time as quickly as increasing number of VMs.

After all the physical processors are occupied, increasing number of VMs or vCPUs per VM has no significant influence on performance. Instead, too many vCPUs (*e.g.*, the total number of vCPUs is greater than the number of actors) will eventually decrease the performance.

Performance on different number of VMs with different memory size of each VM is shown in Figure 2(b). In this experiment, we fix the number of vCPUs on each VM as 4. Figure 2(b) indicates that adding VM instances or allocating more virtual memory to each VM will not necessarily improve performance. On the contrary, it may raise VM configuration exceptions and cause a downtime if servers do not have enough physical memory to allocate. In our experiment, increasing memory becomes a performance problem when the total virtual memory gets to 16GB, which is only half the available physical memory (32GB.)

Figure 2(c) presents the impact of different number of vCPUs and virtual memory size per VM on performance. In this experiment, the number of VMs is fixed at 4. The results indicate that allocating larger memory to VMs will not necessarily improve performance; or conversely that smaller memory size of each VM will not worsen performance. Adding more vCPUs per VM after all physical CPUs are occupied is not better either. On the contrary, it will worsen performance, though not significantly.



(a) Performance change on different number of VMs and different number of vCPUs per VM (memory size of each VM is 1024MB)

(b) Performance change on different number of VMs and different memory size per VM (number of vCPUs per VM is 4)

(c) Performance change on different number of vCPUs and different memory size per VM (number of VMs is 4)

Figure 3. Performance of *Heat* running on different virtual network configurations

B. Tightly Coupled CPU-Intensive Workload

Figure 3(a) shows the performance change on different number of VMs and vCPUs. The number of VMs becomes critical in this workload.

A high ratio of actors to VMs is undesirable, since that means more actors would be assigned to the same VM, which causes more CPU and memory contention between actors. The intense CPU and memory competition will influence the computing speed of each actor and meanwhile slow the communication between each other. For example, in our experiment when we have 2 VMs, no matter how many vCPUs per VM we have, running time is always very high.

Too many VMs or too low a ratio of actors to VMs is also inadvisable. In such case, memory utilization is not optimal, since an actor's memory no longer fits in the lower levels of the memory hierarchy (L1 and L2.) Therefore, an optimum number of VMs exists to get the best performance. For vCPUs setting, it is also not trivial. Too few vCPUs per VM can not make full use of physical CPUs but too many vCPUs will add significant context switching overhead. In our experiment, when we set the total number of vCPUs to 2 times the number of processors, we obtain the best performance.

Figure 3(b) indicates the relationship between performance and the configuration of different number of VMs and memory size per VM is subtle. Higher memory size per VM is generally better except when the number of VMs grows. Memory becomes a bottleneck when the number of VMs is small. The reason is that less VMs means more actors on each VM and thus more memory is needed for a VM to save both raw data and intermediate results. The graph shows that 2 VMs with 512MB or 1024MB memory per VM configuration has a very bad performance, several times worse than other configurations; when allocating more memory to each VM (for example, 2048MB or larger), the running time decreases to some ideal point. However, always keeping a large memory for every VM is not reasonable. The reasons are as follows. First, as the number of VMs grows, memory size becomes not critical and the performance improvement caused by increasing memory size is inconsequential; for example, in our experiment when the number of VMs is greater than 2, performance improvement by doubling the memory size of each VM has been around 10%. Second, creating new VMs becomes unworkable due to lack of free physical memory, in this experiment, when we have 8 VMs, the largest memory we can allocate to each VM is 2048MB; while when having 16 VMs, the largest memory can be allocated is 1024MB. Once again, we can efficiently use only half the available physical memory (32GB.) Lastly, a VM with large memory takes more time doing migration.

The last experiment is about the impact on workloads' performance when given different number of vCPUs and different virtual memory size on each VM. We set the total number of VMs in the system as 4 and the results are shown in Figure 3(c). As mentioned before, running time becomes really large when the number of vCPUs per VM is low, especially when the memory size of the VM is also small. This is because all actors will fall into a vicious cycle due to low CPU utilization but high memory consumption. In such case, either larger memory size or more vCPUs per VM will make an improvement. Figure 3(c) also indicates that a special configuration exists to give the best performance. In this experiment, 4 vCPUs per VM give the best performance if given the same size of memory on VMs. More vCPUs will slightly bring down performance because of the less effective memory hierarchy utilization.

C. External Network Intensive Workload

Figure 4 shows the transaction rate of the *web system* under different number of concurrent clients. The figure

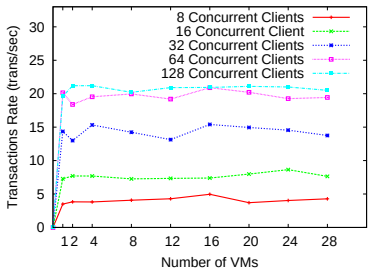
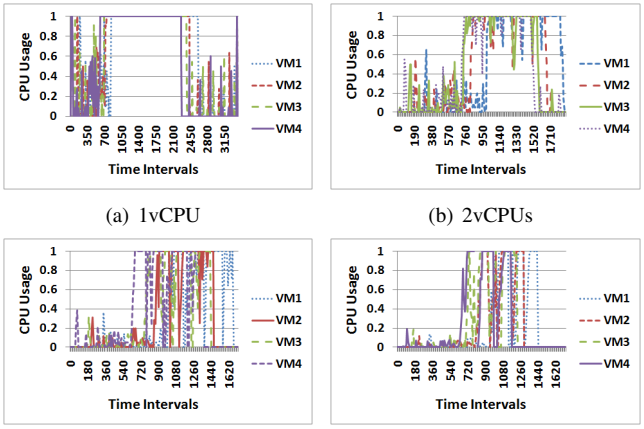


Figure 4. The impact of VM granularity on network-intensive workload's performance (results come from [11])

indicates that given a certain number of concurrent clients, the transaction rate fluctuates at a small extent when the number of VMs changes. VM granularity is not critical for the performance of loosely coupled network intensive workloads, which provides an opportunity for VM consolidation.

D. Resource Usage

In this section, we will discuss the CPU and memory consumption for the CPU-intensive workloads. We developed a sub-module, the *resource usage monitor,* which runs on both PMs and VMs, to collect CPU and memory statistics during the workloads' execution. These statistics were collected and measured from the `/proc` file system in 10ms intervals. All experiments were repeated five times and averaged. In our experiments, we collected the usage statistics on all kinds of configuration strategies. However, we cannot present all of them here due to limited space. We choose some typical settings which can sufficiently express the key inside points.



(a) 1vCPU (b) 2vCPUs

(c) 4vCPUs (d) 8vCPUs

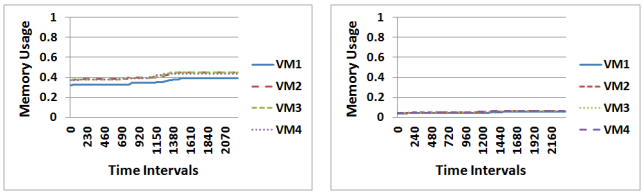
Figure 5. CPU usage of *Stars* on different number of vCPUs per VM

1) Resource Usage of Stars Workload: Figure 5 displays the measured CPU usage on 4 VMs (each has 1024MB memory) when having different number of vCPUs.

In Figure 5(a), CPU utilization stays at 100% for a long time and is relatively high during other time intervals. This demonstrates that the VMs have a heavy workload for low number of vCPUs configurations (*i.e.* 1vCPU per VM in our experiments). In Figure 5(b), Figure 5(c) and Figure 5(d), the vCPUs are not as busy as Figure 5(a).

Throughput is relatively low for high number of vCPUs per VM configurations; take Figure 5(d) for example, the CPU utilization is less than 20% during a long time interval and only get 100% at some time points; the throughput is lower when having 16 vCPUs per VM. Though we have low throughput for high number of vCPUs configurations, the work load for each vCPU becomes relatively light. That is why we get similar performance for different number of vCPUs configurations.

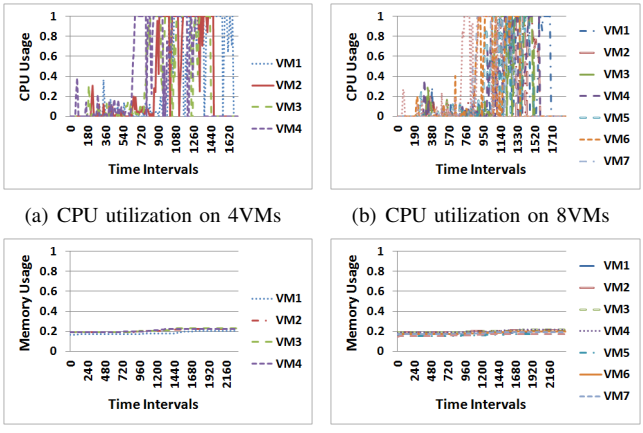
In Figure 6, we show the memory usage (by percent) on 4 VMs (2vCPUs per VM) when having different memory size.



(a) 512MB (b) 4096MB

Figure 6. Memory usage of *Stars* on different memory size per VM

Figure 6(a) has a relatively high memory utilization, greater than 40% for most of the time. The utilization in Figure 6(b), which has a much larger memory on every VM, falls into a very low level ($< 10\%$). Since the workload is loosely coupled, no more significant memory is needed during runtime once all data are loaded at the start. Increasing memory size for each VM thus will not bring improvement on performance. It is also a waste to allocate larger memories for VMs, which may preclude VM consolidation.



(a) CPU utilization on 4VMs (b) CPU utilization on 8VMs

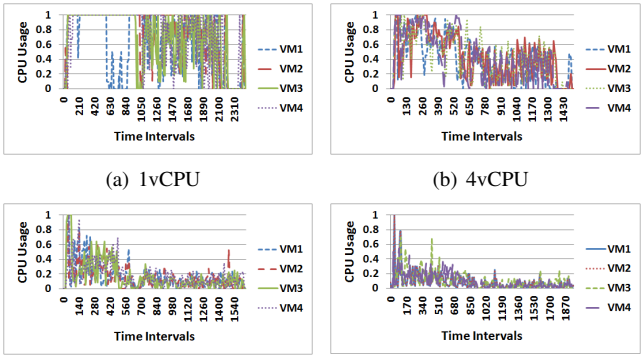
(c) Memory utilization on 4VMs (d) Memory utilization on 8VMs

Figure 7. CPU and memory usage of *Stars* on different number of VMs

We have discussed the resource usage on different number of vCPUs and different virtual memory size. Given fixed number of vCPUs and memory size per VM, we consider resource utilization for different number of VMs. Each VM is allocated 4vCPUs and 1024MB memory. Figure 7 shows the corresponding CPU, memory utilization on 4 VMs and 8 VMs respectively.

They have similar CPU and memory utilization and both are in a reasonable level. Therefore the 4VMs virtual network configuration is enough. Doubling the virtual resources (including CPU, memory) for loosely coupled CPU-intensive workloads will not necessarily improve performance.

2) Resource Usage of Heat Workload: Figure 8 displays the measured CPU usage on different number of vCPUs of 4VMs (each VM has 1024MB memory).



(a) 1vCPU (b) 4vCPU

(c) 8vCPU (d) 16vCPU

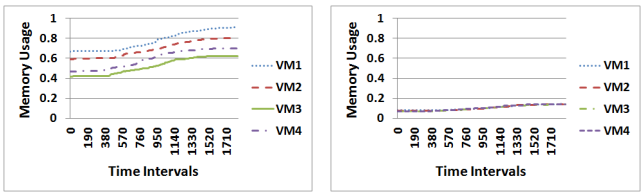
Figure 8. CPU usage of *Heat* on different number of vCPUs per VM

In Figure 8(a), we can see that CPU utilization is high ($>$ 90%) for almost all the time. The utilization is generally less than 30% in Figure 8(d). This demonstrates the VMs have a very high work load for low number of vCPUs settings (*i.e.* 1vCPU per VM in our experiments) , but have a low throughput for high number of vCPUs setting (16vCPUs per VM in our experiments). Figure 8(c), which has 8vCPUs per VM has better CPU utilization than Figure 8(d), but is still not efficient enough. Figure 8(b), the 4vCPUs per VM setting, is the best one of all. It has a reasonable utilization ($>$ 60%), not high as Figure 8(a) and also not low as 8vCPU or higher setting. 2vCPUs per VM setting is not shown here. It has similar CPU utilization with Figure 8(b).

In Figure 9, we show the memory usage (by percent) on different memory size of 4 VMs (2vCPUs per VM).

Figure 9(a) has a relatively high memory utilization, greater than 60% for most time; The utilization in Figure 9(b), which has a much larger memory on every VM, falls into a very low level ($<$ 20%). As mentioned before, larger memory is generally better. However it is a waste of resources if the memory is too large.

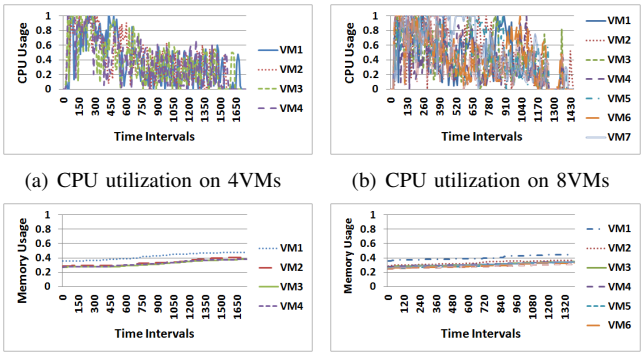
We have discussed the resource usage on different vCPU and virtual memory configurations. The resource utilization of different number of VMs (each VM has 2vCPUs and 1024MB memory) is shown in Figure 10.



(a) 512MB (b) 4096MB

Figure 9. Memory usage of *Heat* on different size of virtual memory per VM

First, memory is not a bottleneck in these two cases and they have similar memory utilization. The 8VMs configuration has a better CPU utilization. It is also a proven fact that it improves the performance about 15% by increasing VM number from 4 to 8 with the configuration of 2vCPUs and 1024MB per VM. Since we are using the same physical resources, it is better to double the virtual resources and get better performance.



(a) CPU utilization on 4VMs (b) CPU utilization on 8VMs

(c) Memory utilization on 4VMs (d) Memory utilization on 8VMs

Figure 10. CPU and memory usage of *Heat* on different number of VMs

IV. Analysis

Our experimental results show that for CPU-intensive workloads, the total number of vCPUs (*i.e.*,VMs*vCPUs/VM) has to be large enough to fully utilize all available physical processors. The total virtual memory of all VMs allocated to a physical machine, has to be less than the physical memory of the machine to avoid poor performance due to swapping.

For loosely coupled CPU-intensive workloads, the number of vCPUs and memory size of VMs have no significant impact on performance. The effect of VM granularity is also not critical. We get slightly better results by using a large number of VMs with a small number of vCPUs per VM versus using a few VMs with a large number of vCPUs per VM.

For tightly coupled CPU-intensive workloads, the virtual network configurations become critical for performance. First, if the number of VMs is small, there should be large memory allocated to each VM to ensure efficient inter-process communication (*e.g.*, in our experiments, if we have only 2 VMs, they should each be allocated at least 2GB of virtual memory). Second, the impact of VM granularity (number of VMs) on performance is significant (see Figure 3(a)). Last, given a number of VMs, too large a ratio of vCPUs to CPUs (for example greater than 2 in our experiments) will hinder performance (see Figure 3(c)).

For external network-intensive workloads, virtual network configuration is not critical for performance. It provides an opportunity to apply VM consolidation without sacrificing quality of service.

Service providers benefit by knowing where, when and how to place VMs: first, system administrators or middleware need to guarantee all physical processors will be occupied when placing or consolidating VMs; second, consolidation is only feasible when physical memory is enough. If the workload is external network-intensive, consolidation can be more effectively applied. Private cloud owners can configure virtual resources from the results for these two kinds of workloads: first, the ratio of total number of vCPUs to processors has to be greater than 1 to guarantee a reasonable performance; second, memory shrinking to consolidate VMs and save energy is feasible. For tightly coupled CPU-intensive workloads, private cloud owners need to balance between performance and different configuration strategies. Small number of VMs with small number of vCPUs and small size of memory of each VM should always be avoided. The number of VMs should be the first parameter to optimize in order to stay clear of the poor performance regions. Large memory is good, but if VM migration is required, the administrator or middleware should consider to shrink the memory size. Service users can learn what kind of instances they actually need for a given workload. For loosely coupled CPU-intensive and external network-intensive workloads, it is a waste of money to request large or high-memory or high-CPU VM instances. It is better to request more smaller VMs. For tightly coupled CPU-intensive workloads, a few large memory VM instances have comparable performance to more small memory ones (see Figure 3(b)), so the decision on the cheapest well-performing configuration depends on the public cloud providers' pricing for large/small VM instances.

To guide cloud administrators and middleware developers, the key to performance improvement of loosely coupled CPU-intensive workloads is the throughput of the CPUs, they should avoid the heavy CPU load situation (see Figure 5(a)). For tightly coupled workloads, the resource consumption data can help know the proper time to change virtual network configuration strategies: if CPU utilization is very high during some time interval (see Figure 8(a)), adding more vCPUs for each VM or adding more VMs should be considered; on the other hand, if CPU utilization is very low during a time interval (see Figure 8(d)), it is better to decrease the number of vCPUs or consider VM consolidation. Memory usage would help private cloud owners know the proper time for memory ballooning: if memory utilization is already high at the beginning (see Figure 9(a)), it is more likely we need to allocate a larger memory; during runtime, if the slope of memory usage is stiff and utilization is high, more memory should be allocated to VMs; if the slope is flat and utilization is relative low (see Figure 6(b) and Figure 9(b)), then memory shrinking is feasible. Service users can learn from the resource consumption by knowing that large VM instances may have very low throughput and therefore may be a waste of money.

V. Related Work

Virtualization management, which refers to the abstraction of logical resources away from their underlying physical resources in order to improve agility, flexibility, reduce costs and thus enhance business value [7], is a critical component to accomplish effective sharing of physical resources and scalability. Existing works on virtual network management mainly focus on VM migration [8], [5], [4]. Sapuntzakis et al. show that efficient capsule (the state of running computer across a network, including the state in its disks, memory, CPU registers, and I/O devices) migration can improve user mobility and system management [8]. Hansen et al. present two prototypes that allow application and the underlying operating system migration together [5]. Clark et al. introduce live migration and achieve impressive performance with minimal service downtime by carrying out the majority of migration while keeping OSes running [4]. The results of our experiments in this paper can help create more efficient virtual machine networks, thus help improve migration and adaptation capabilities.

Our results are important middleware level references for cloud service providers to decide where and how to place VMs, thus improving resources utilization. Much work has been done on workload level to improve resource utilization. Tirado et al. present a predictive data grouping scheme to scale up and down servers dynamically, aiming to improve resource utilization and implement load balance [9]. Their forecasting model is based on access history. It works for a mature portal web site, like the music portal LastFM mentioned in their paper. Wu et al propose resource allocation algorithms for SaaS (Software as a Service) providers to help them to minimize infrastructure cost and SLA (Service Level Agreement) violations [12]. The main method is to decide where and which type of VMs to be initiated. Marshall et.al. propose "a cloud infrastructure that combines on-demand allocation of resources with opportunistic provisioning of cycles from idle cloud nodes to other processes by backfill

VMs", targeting to improve the utilization of Infrastructure Clouds [6].

VI. Conclusion and Future Work

In this paper, we analyzed three different categories of workloads on various virtual network configuration strategies (*i.e.*, number of VMs, number of vCPUs per VM, memory size of each VM). To prevent poor performance regions, the total number of vCPUs has to be larger than the number of processors (*e.g.,* twice), and the total virtual memory allocated to VMs has to be less than physical memory (*e.g.*, half.) For tightly coupled CPU-intensive workloads, a key virtualization parameter for performance is the number of virtual machines: too many VMs will add significant inter-VM communication overhead and will limit the amount of virtual memory per VM; while too few VMs will require larger virtual memory per VM to obtain reasonable performance. Since the performance of a few large memory VM instances is comparable to more smaller memory ones, the decision of the best virtual network configuration lies on VM instances pricing by public cloud providers. For external network-intensive workloads, virtual machine granularity is not critical for performance.

In reality, the scale of the workloads submitted to a cloud computing environment is much larger than the benchmarks in our experiments, the difference of performance, resource consumption or cost on different virtual network configurations is significant. Therefore, it is important to know the impact of different virtual network configurations in a cloud environment for service users, service providers and private cloud owners. Service providers and private cloud owners can build a middleware to implement autonomous refining of virtual network configurations by monitoring the CPU and memory utilization.

Based on further experimental results, virtualization in our environment added 9.3% overhead in the loosely coupled CPU-intensive application, and 12.8% in the tightly coupled CPU-intensive application. Private cloud owners may choose not to use the virtual environment because of the overhead. However, considering the scalability benefits of cloud computing, it is still worthy to have a grid-cloud hybrid environment.

In next steps, we will design corresponding virtual network configuration refining policies to implement autonomous virtual network reconfiguration. Moreover, we will expand our testbed infrastructure to a larger scale environment, which includes more data-intensive applications and an interface to request VM instances from a public cloud.

Acknowledgments

We would like to thank Ping Wang and Wei Huang for their previous work and thank Shigeru Imai for his helpful comments. Funding for this research is provided in part by NSF CAREER CNS Award No. 0448407 and NSF MRI No. 0420703.

References

[1] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep.*, UCB/EECS-2009-28, Feb 2009.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[5] Hansen, J. Gorm, and E. Jul. Self-migration of operating systems. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, New York, NY, USA, 2004. ACM.

[6] P. Marshall, K. Keahey, and T. Freeman. Improving utilization of infrastructure clouds. In *CCGrid 2011, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Newport Beach, CA, USA, May 2011.

[7] B. Rimal, E. Choi, and I. Lumb. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pages 44 –51, Aug. 2009.

[8] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. *SIGOPS Oper. Syst. Rev.*, 36:377–390, December 2002.

[9] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero. Predictive data grouping and placement for cloud-based elastic server infrastructures. In *CCGrid 2011, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Newport Beach, CA, USA, May 2011.

[10] C. Varela and G. Agha. Programming dynamically reconfigurable open systems with SALSA. *SIGPLAN Not.*, 36:20–34, December 2001.

[11] P. Wang, W. Huang, and C. A. Varela. Impact of virtual machine granularity on cloud computing workloads performance. In *Workshop on Autonomic Computational Science (ACS'2010)*, Brussels, Belgium, October 2010.

[12] L. Wu, S. K. Garg, and R. Buyya. SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. In *CCGrid 2011, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Newport Beach, CA, USA, May 2011.