Sustainable Computing: Informatics and Systems 4 (2014) 1–9

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Performance bounded energy efficient virtual machine allocation in the global cloud

Patrick Raycroft[a], Ryan Jansen[a], Mateusz Jarus[b], Paul R. Brenner[a,*]

[a] University of Notre Dame Center for Research Computing, Notre Dame, IN, USA
[b] Applications Department Poznań Supercomputing and Networking Center, Poznań, Poland

ARTICLE INFO

Article history:
Received 8 February 2013
Received in revised form 26 June 2013
Accepted 1 July 2013

Keywords:
Cloud
Virtual machine
Dynamic allocation
Network performance

ABSTRACT

Reducing energy consumption is a critical step in lowering data center operating costs for various institutions. As such, with the growing popularity of cloud computing, it is necessary to examine various methods by which energy consumption in cloud environments can be reduced. We analyze the effects of global virtual machine allocation on energy consumption, using a variety of real-world policies and a realistic testing scenario. We found that by using an allocation policy designed to minimize energy, total energy consumption could be reduced by up to 14%, and total monetary energy costs could be reduced by up to 26%. Further, we have begun performance qualification of our energy cost driven allocation policies through network capability tests. Our results indicate that performance and IaaS provider implementation costs have a significant influence on selection of optimal virtual machine allocation policies.

1. Introduction

As adoption of virtualization services increases, cloud computing platforms are becoming increasingly popular. Demand for existing cloud infrastructures such as Amazon's Elastic Compute Cloud [1] is steadily rising [2], and the use of private compute clouds is becoming more popular among a variety of institutions [3].

While virtualization has in many cases facilitated IT infrastructure consolidation for individual organizations, expanding demand for IT services via cloud technology drives growth. As virtualization on various cloud platforms becomes more prevalent, the rising number of virtual machine requests in any given cloud necessitates a proportionally increasing number of physical host servers to fulfill them. As such, data center sizes are expanding, generating a growing concern over energy consumption and electricity costs among businesses and hosting providers alike.

The energy cost incurred in running a data center has been steadily rising for years. Data center energy costs in the United States accounted for nearly 1.5% of all electricity costs in 2006, reaching up to approximately $4.5 billion per year, and trend data estimates that this cost will jump to an annual cost of $7.6 billion for 2011 [4]. According to recent report by Koomey [5], the growth in electricity used by data centers worldwide was in fact lower than previously predicted. It was mainly affected by two factors–slowdown of growth in the installed base of servers because of virtualization and the financial crisis of 2008 with its associated economic inhibition. Nevertheless, the energy issue is still an important concern as the high density computing facilities keep expanding and, as a result, require more and more power.

With this in mind, it is worthwhile to attempt to minimize energy consumption through any means available. In this paper, we will examine various cloud allocation policies used to match virtual machines to physical hosts in a cloud environment. We will simulate each policy independently and analyze its effectiveness in a number of categories, with a focus on energy consumption.

Related work by Garg et al. [6] focuses on deploying high-performance computing (HPC) services across a cluster while minimizing carbon emissions, using a combination of minimization algorithms and CPU voltage scaling. Additionally, Kim et al. [7] has focused on developing a similar system that combines scheduling and CPU voltage scaling to achieve reduced energy consumption across a cluster. Such methods reduce energy costs, but their focus is on power consumption at the CPU level as opposed to the cluster level. More akin to our scheduling analysis is a system developed by Chase et al. [8] in which services bid on host machines and are scheduled to minimize energy costs, while properly allocating services to handle varying web loads. Unlike our research, their approach relies on an outside scheduling framework. Further, work by Mazzuco et al. [9] develops dynamic scheduling of servers local to one data center to maximize user experience while minimizing the energy costs of cloud providers. Their work differs from ours in

* Corresponding author at: 111 Information Technology Center, Notre Dame, IN 46556, USA. Tel.: +1 574 631 9286.
E-mail addresses: praycrof@nd.edu (P. Raycroft), ryan.alan.jansen@gmail.com (R. Jansen), jarus@man.poznan.pl (M. Jarus), pbrenne1@nd.edu, paul.raymond.brenner@gmail.com (P.R. Brenner).

that it addresses the scheduling policies of local physical servers as opposed to globally distributed VMs. In addition, Beloglazov et al. [10] focus on virtual machine reallocation by taking into account Quality of Service and cost minimization. However, their work concentrates solely on a single data center, contrasting our global methodology. Finally, work by Aikema et al. [11] takes a global migration approach, similar to our migration policies originally discussed by Jansen et al. [12]. However, we differ by considering multiple experimental network performance tests as well as energy cost optimization.

A different approach, focusing on optimizing the allocation process, is described by Srikantaiah et al. [13]. Their strategy involves modeling the cloud as a bin packing problem, with physical hosts as bins and virtual machines as objects to fit inside of them. Using this model, they attempt to consolidate the virtual machines to as few hosts as possible, in an effort to minimize overall energy usage. As we will see later on, this approach is akin to (although much more advanced than) the *Packing* allocation policy defined in our simulation.

Zheng et al. [14] created an optimal energy-aware load dispatching model to minimize the electricity and network costs for Online Service Providers. They selected end-to-end response time as the metric of performance, which consists of network delay and response time inside an Internet Data Center. Geographic distance was used as a rough measure of network latency. The round trip time for a request from user group to data center is a linear function of its distance. However, this approach is not accurate. As our experiments indicate, the latency and throughput of a network connection cannot be always calculated in this simple manner. Moreover, network bandwidth changes throughout the day, being affected by network congestion and load on servers. For this reason our simulation is based on real results from experiments measuring network efficiency between selected data centers.

The allocation policies presented in this paper are either already available on two popular cloud platforms (OpenNebula [15] and Eucalyptus [16]), or they are straightforward to implement using either platform's scheduling policy syntax. In this paper, we will attempt to analyze how various allocation policies affect energy consumption, as well as CPU load and overall energy costs, in a realistic environment based on dynamic website loads.

Of the seven allocation policies we tested, four are currently available by default in existing open-source cloud platforms. The four existing policies tested include Round Robin, Striping, Packing, and free-CPU-count-based Load Balancing. One of the remaining three, ratio-based Load Balancing is a variation on the original count-based load balancing, and the other two, the Watts per Core and Cost per Core policies, are experimental, intended to minimize overall data center energy consumption and energy costs respectively. These policies are described in depth by Jansen et al. [12] and are summarized in Section 2.

2. Simulation scenario

Via our simulation, we tested seven different cloud allocation policies: Round Robin, Striping, Packing, Load Balancing (free CPU count), Load Balancing (free CPU ratio), Watts per Core, and Cost per Core.

- *Round Robin*: This allocation policy iterates sequentially through available hosts. When a host is found that has sufficient resources, the VM is matched to the host. On the next iteration, the policy starts its iterations where it previously left off.
- *Striping*: This policy first discards all hosts that do not have sufficient available resources to host the machine. It then selects from the remaining hosts the one that is currently hosting the fewest number of VMs and matches the virtual machine to that host.
- *Packing*: The Packing policy is the opposite of Striping. After discarding all hosts similarly to Striping, it selects from the remaining hosts the one that is currently hosting the greatest number of VMs and matches the virtual machine to that host.
- *Load Balancing (free CPU count)*: Similarly to the other policies, the count-based Load Balancing policy first discards all hosts that do not have sufficient available resources. From the remaining hosts, it then selects the one with the greatest number of free CPU cores and matches the virtual machine to that host.
- *Load Balancing (free CPU ratio)*: This policy is similar to the count-based Load Balancing; however, it instead selects the host with the greatest ratio of free CPU cores to allocated CPU cores and matches the virtual machine to that host.
- *Watts per Core*: From the pool of hosts that have sufficient available resources, this policy selects the host that would result in using the least additional wattage per CPU core if chosen, based on each host's power supply, and matches the virtual machine to that host.
- *Cost per Core*: This policy is similar to the Watts per Core policy above; however, it instead selects the host that would result in using the least additional cost per CPU core if chosen, based on each host's power supply and electricity costs, and matches the virtual machine to that host.

Table 1
Physical host specifications.

Physical server	Cluster			
	US East server counts	US West server counts	Asia server counts	Europe server counts
server.A1	0	24	0	0
server.A2	0	8	0	0
server.B1	24	0	0	0
server.B2	16	0	0	0
server.C1	0	0	8	0
server.C2	0	0	8	0
server.D1	0	0	0	16
server.D2	0	0	0	16
server.D3	0	0	0	16

Our simulation scenario attempts to accurately simulate a large-scale website–the social media site Reddit.com. Reddit.com [17] shifted their entire infrastructure to Amazon EC2 virtual machine instances, and, as of February, 2011, the site serves up to 1 billion users monthly [18].

In the scenario, we define four clusters of physical hosts, each representing a geographical location around the world as well as an existing Amazon EC2 data center [19]. As mentioned above, the website is hosted entirely on a set of virtual machines, which will be distributed among the clusters as necessary to deal with dynamic server loads. The load structure was chosen specifically to imitate typical web server loads based on the time of day at the different geographical locations.

In this section, we will state the specification of each of our physical hosts, the requirements of each of our virtual machines, and the website load scheme used in our simulation.

2.1. Physical hosts

The physical hosts in our simulation are based off of commodity servers available from IBM. Server specifications are based off of the specifications and power requirements provided by IBM's Power Configurator tool [20]. The different servers used in our simulation are defined in Table 1.

Table 2
Cluster allocations.

Name	IBM model	Cores	Memory (GB)	Min power usage (W)	Max power usage (W)
server.A1	x3550 M3	8	16	200	410
server.A2	x3550 M3	12	32	210	430
server.B1	x3455	4	8	125	260
server.B2	x3455	8	16	180	375
server.C1	x3550 M2	4	8	170	280
server.C2	x3550 M2	8	16	250	410
server.D1	x3650 M2	4	8	180	300
server.D2	x3650 M2	8	16	260	430
server.D3	x3650 M3	12	32	230	470

Table 3
Virtual machine specifications.

Name	EC2 instance type	Cores	Memory (GB)	Size (GB)
vm.Application	c1.xlarge	8	7	1690
vm.Frontend	m1.large	2	7	850
vm.Database	m1.xlarge	4	15	1690

The physical hosts are divided up amongst four individual clusters. Each cluster contains a different number of one to three different types of servers.

Each of the four clusters represents a different geographical location. Those locations are defined as:

- US East–represents a data center in Ashburn, Virginia in the United States.
- US West–represents a data center in Los Angeles, California in the United States.
- Asia–represents a data center in Singapore.
- Europe–represents a data center in Dublin, Ireland.

Each cluster hosts a varying number of physical host servers. Table 2 defines the server allocation of the clusters.

2.2. Virtual machines

For the simulation, 108 virtual machines, representing the EC2 instance makeup of Reddit.com, were used for testing. Each virtual machine is based off of an existing EC2 instance type, with similar core, memory, and size requirements. The virtual machines and their specifications are shown in Table 3.

Each cluster has a number of virtual machines of each type that are meant to handle the base load at any given time. As such, these virtual machines are persistent and will always be allocated on their respective clusters. Alternatively, there are some virtual machines that are meant to help handle additional load at any given cluster. As such, these virtual machines are allocated based on each cluster's load each hour. Table 4 shows the virtual machine allocation across the clusters.

Table 4
Virtual machine allocations.

Virtual machine	Cluster				
	US East VM counts	US West VM counts	Asia VM counts	Europe VM counts	Load VM counts
vm.Application	8	8	2	6	20
vm.Frontend	5	5	1	3	10
vm.Database	8	8	2	6	16

2.3. Dynamic load

The simulation emulates dynamic website loads coming from the different clusters at different times of the day. Over the course of 24 h, the load for each cluster is determined as a percentage of the total load on all clusters. The load percentage $Load_c$ for each cluster is calculated as the ratio between the number of users accessing the website from the cluster's region $Users_c$ to the total number of users accessing the site across all regions $Users_T$. This calculation is represented by the following equation:

$$Load_c = Users_c / Users_T$$

Once the load percentage for each cluster is determined, the simulation uses each load percentage to assign a proportional number of virtual machines to each of the clusters. Each cluster has a designated number of persistent hosts that always remain in their respective clusters. These are meant to handle hosting of the website in each region without any additional load. The regional VM counts in Table 4 represent these persistent virtual machines.

The remaining virtual machines that are not persistent, or the "load-handling" virtual machines, are transferred from cluster to cluster as the load changes, each cluster receiving a number of load-handling virtual machines proportional to its load percentage. For example, if, at a single iteration, the US East cluster had a load percentage of 25% and the US West cluster had a load percentage of 75%, then 25% of the load-handling virtual machines would be assigned to the US East cluster, and 75% would be assigned to the US West cluster. Note that these load handling virtual machines would be assigned to the cluster in addition to its persistent virtual machines. If a cluster does not have the resources to host all of its assigned load-handling virtual machines, they are assigned to other clusters instead. The Load VM counts in Table 4 represent the types and quantities of the load-handling virtual machines.

The load is simulated as changing on an hourly basis, over a 24 h period. It assumes that users are most active from 6 PM to 3 AM in their local time zone. Additionally, we are assuming that the majority of users visiting the site are American, that there are a smaller number of Europeans users, and that there is a very small population of Asian users. The allocated cluster servers shown in Table 1 reflect such a setup. Fig. 1 shows the load percentage of each cluster over a 24 h period.

2.4. Simulation process

The simulation process is relatively straightforward, but we will outline it here. At each iteration over a 24 h period, we perform the following steps:

- Allocate load-handling virtual machines: First, we used the current load percentages for each cluster to divide up the load-handling virtual machines.
- For each cluster, match its load handling machines to its hosts using a scheduling policy: Next, we use the scheduling policy to match each cluster's set of load-handling virtual machines.
- For virtual machines that could not be matched, match each to all of the remaining hosts using the scheduling policy: For each virtual machine that could not be matched to its preferred cluster, we match it against all of the physical hosts, on any cluster.

During the simulation, virtual machine allocations are logged and later analyzed to determine the overall performance of the scheduling policy that was used.

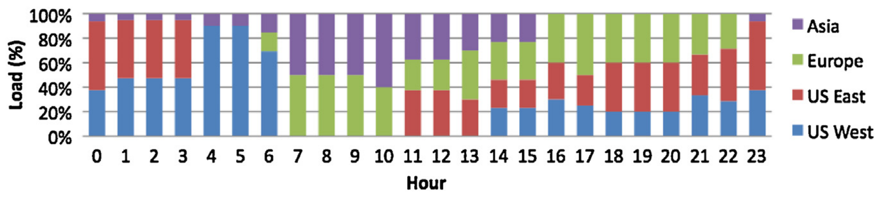

Fig. 1. Luster load percentages per hour [12].

2.5. Assumptions

A number of assumptions are made in our simulation to both simplify the simulation process and mimic a real world scenario. These assumptions are as follows.

2.5.1. Non-idle hosts use a fraction of their maximum power usage

Hosts that are hosting a number of virtual machines are assumed to use a fraction of their maximum power. This fraction is calculated using the minimum and maximum power usage values for the host (P_{min} and P_{max} respectively), found in Table 2, as well as the ratio of allocated CPU cores $Cores_{alloc}$ to total CPU cores $Cores_{total}$ of the host in question. At any given time, the current power consumption P is calculated using:

$$P = P_{min} + (P_{max} - P_{min})^{*} \left(Cores_{alloc}/Cores_{total}\right)$$

2.5.2. Idle hosts use no power

We also assume that idle machines are in a low-power state of some sort (hibernating, off, etc.), and, as such, we evaluate their power consumption to 0.0 W.

2.5.3. When allocating load-handling virtual machines, local machines are always allocated first

When load-handling virtual machines are allocated, each cluster chooses machines already on its cluster before remote virtual machines.

For example, if during the first iteration, the US West Cluster required 2 vm.Application machines, they would be allocated to it as needed. If during the next iteration, the cluster needed 3 vm.Application machines, it would choose the 2 that it is already hosting first, and then proceed to retrieve 1 more from a remote location.

2.5.4. Each cluster has its own shared SAN

Each cluster has shared storage, and can migrate machines between hosts instantaneously.

3. Results and analysis

In this section, we will examine the results of our simulation, which are shown in Table 5. In the tables, energy consumption is measured by the average total kilowatts used per hour across all hosts. Cost is the average total cost per hour, based on our estimated server costs specified in part B of this section. And finally, CPU load is the average hourly ratio of allocated cores total cores among active hosts.

Table 5
Simulation results.

Scheduling policy	kW/hr	Cost/hr (USD)	CPU Load (%)
Round Robin	30.33	$6.17	72.45%
Striping	32.47	$6.84	67.19%
Packing	30.01	$5.68	76.33%
Load balance (Count)	31.27	$6.51	70.25%
Load balance (Ratio)	32.43	$6.84	67.13%
Watts per core	27.89	$5.03	79.51%
Cost per core	28.04	$5.05	79.05%

3.1. Energy

Energy consumption is measured in average kilowatts per hour over the 24 h simulation period.

Unsurprisingly, the Watts per Core policy did the best from an energy-saving standpoint, achieving an average energy usage of approximately 27.9 kW per hour. Cost per core did well also, which is expected considering the direct correlation between energy and cost.

Packing and Round Robin did reasonably well, averaging approximately 30.0 and 30.3 kW per hour, respectively. The Load Balancing and Striping policies did significantly worse, each averaging between 31.2 and 32.4 kW per hour. Fig. 2 shows the hourly energy consumption of each policy.

Based on our assumption that inactive hosts can be placed in a low power state and require no energy, packing the hosts should provide the best energy efficiency, as such a strategy minimizes the number of hosts that are powered on. With this in mind, we can see that the Watts per Core and Cost per Core policies are actually optimizations of the Packing policy. Whenever the Packing policy fills up a host machine, it arbitrarily chooses a new one. The Watts per Core and Cost per Core policies will always choose the most energy-efficient or cost-efficient host to begin packing next, and, in this way, they perform better than basic packing.

Load balancing and striping are expected to perform the worst in this category, as they both involve starting up as many hosts as necessary, and, in turn, use more energy. The Load Balancing policy performs better than the Striping policy, because it adds more virtual machines to larger hosts, whereas the Striping policy will instead try to naively use as many hosts as possible.

The Round Robin policy performed, in the middle, worse than any of the packing-based policies and better than striping or load balancing. Fig. 2 shows the hourly consumption of each policy.

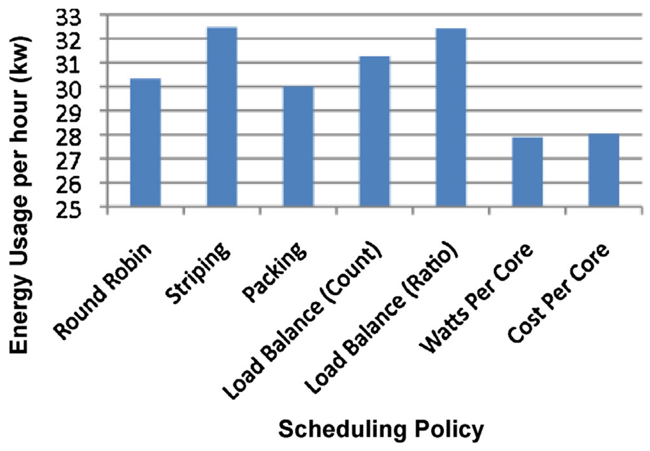

Fig. 2. Average energy consumption per hour vs. scheduling policy.

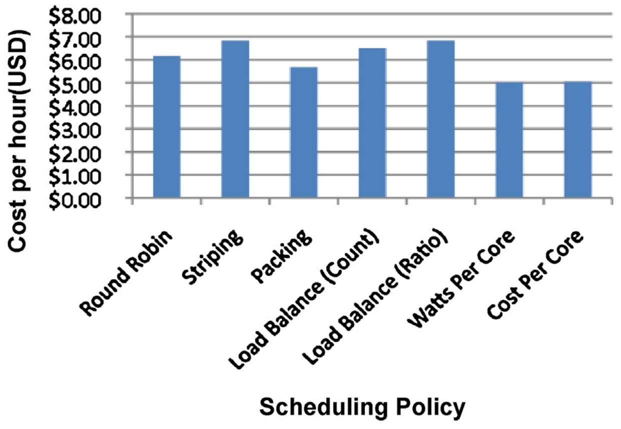

Fig. 3. Average cost per hour vs. scheduling policy.

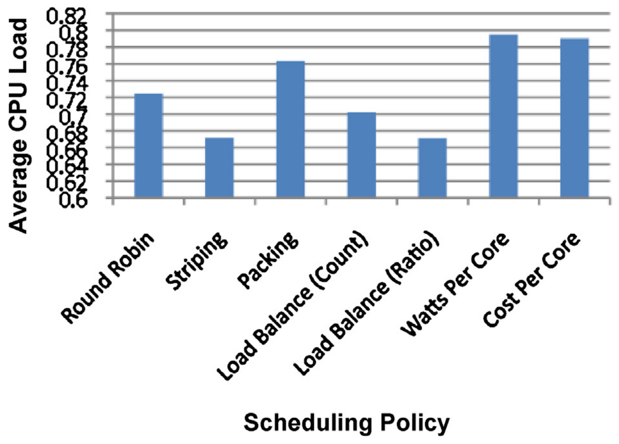

Fig. 4. Average CPU load vs. scheduling policy.

3.2. Cost

Cost is the average cost (in USD) per hour to run all active hosts across the cluster.

The cost of electricity for each cluster is based on actual industrial electricity costs for each geographic location. United States electricity costs are based on reports from the U.S. Energy Information Administration [21], European costs are based off of reports from Europe's Energy Portal [22], and Asian costs are based off of a report from the Senate of the Philippines [23]. The cost per kilowatt-hour for each cluster is defined as:

- US East–$0.1105
- US West–$0.0674
- Asia–$0.1058
- Europe–$0.1602

Additionally, each cluster has an associated *Power Usage Effectiveness* (PUE) value. This value is essentially a measure of how efficient a given data center is at using its electricity, as is determined by the following equation [24]:

PUE = Total Facility Power/IT Equipment Power

The PUE values for each cluster are defined as follows:

- US East–1.2
- US West–1.4
- Asia–1.8
- Europe–1.6

For each cluster, we calculate the true cost per kilowatt-hour by multiplying the cluster's cost by its PUE value.

Based on these true costs, our simulation determined the cost of hosting virtual machines for each allocation policy. As expected, the costs are directly related to energy consumption. Striping is the worst, at a price of $6.84 per hour, while the Watts per Core and Cost per Core policies are the best, at $5.03 and $5.05 per hour, respectively. Fig. 3 shows the hourly costs of each policy.

3.3. CPU load

We define CPU load as the ratio of a host's allocated cores to its total cores. The averages only take into account hosts that are actively hosting virtual machines.

The Load Balancing policies and the Striping policy perform best in this category, each achieving an average CPU load between approximately 65% and 70%. Striping and ratio-based Load Balancing should always result in a very similar allocation, as they will allocate virtual machines to the same hosts until every host in a cluster is already active. At that point, they will begin to choose different hosts, which accounts for the slight deviation between the two policies.

The count-based Load Balancing policy, on the other hand, will choose larger hosts with more free CPU cores, allowing it to better pack virtual machines on to its selected hosts. As such, it sees better energy efficiency, but poorer load-balancing performance than ratio-based Load Balancing.

Unsurprisingly, because the Packing, Watts per Core, and Cost per Core algorithms all minimize active hosts, the average load between those active hosts is significantly higher.

Again, the Round Robin policy performed in the middle of the pack, doing better than any of the packing policies, but worse than the load balancing policies. Fig. 4 shows the average CPU load for each policy.

4. Performance considerations

In order to provide a more robust analysis of VM migration, we evaluate the impact of moving VMs across different global regions. In this section, we discuss the network performance between Amazon EC2 instances and the impact that these results have on VM migration.

4.1. Migration overhead

Because our main simulation assumes a dynamic website load at any given time, we move virtual machines around to various clusters to properly handle changing cluster demands. In any real-world scenario, storage transfer costs must be taken into consideration. There are two ways to store data in a virtual environment.

First, using *instance storage*, data is stored in the virtual machine instances themselves. Using this approach would require that each time a virtual machine is transferred, all of its accompanying data is transferred as well, incurring the penalties (time, cost, etc.) to do so.

Second, using *shared storage*, data is stored in some sort of shared filesystem between the hosts, such as a distributed file system or a SAN. Using this method, instead of transferring virtual machines to handle load, a website could merely shut down existing virtual machines and start new ones in the appropriate clusters. Because the data is not stored on the machines themselves, the virtual machines do not need to be transferred and can be run as independent processes interacting with the data at any location. While such an approach eliminates transfer costs, it also necessitates a large shared storage pool with access from all physical hosts.

Table 6
Specifications of EC2 instance types.

EC2 instance type	Memory	EC2 compute units	I/O performance
t1.micro	613 MB	Up to 2	Low
m1.small	1.7 GB	1	Moderate
m1.large	7.5 GB	4	High

Our original simulation assumed that each cluster used shared storage between its hosts, but not between all of the hosts. That is, a host could transfer virtual machines to another host in its cluster at no penalty, but transferring a virtual machine to a different cluster would incur various transfer costs. Additionally, we assumed that each virtual machine *had* to transfer to a cluster in order to serve users in that cluster, and, as the load changed, virtual machine placement strictly adhered to the needs of each cluster. As such, we moved virtual machines when necessary, but we tried to minimally transfer between clusters.

4.2. Network considerations

A real-world scenario needs to take into account the network performance in order to estimate the transfer times of virtual machines. We decided to analyze the real bandwidth between selected Amazon data centers. Below are presented the datacenters examined in our experiments–we performed bidirectional tests to analyze the connections between:

- Virginia, US
- California, US
- Ireland
- Singapore
- Tokyo, Japan
- Sao Paolo, Brazil

To accurately measure the bandwidth and latency between different data centers, we installed one virtual machine on each cluster using Amazon Elastic Compute Cloud. We decided to choose the default VM, the t1.micro (“micro”) instance type [25], to test every connection between each cluster.

To measure bandwidth results for different EC2 instance types, we also installed an m1.small (“small”) and an m1.large (“large”) instance on both the Virginia and California clusters. Specifications for the three instance types used are shown in Table 6.

The input/output performance of the three different instance types varies upon size. While not affecting latency results, this specification does affect the bandwidth levels of the different instances. In order to demonstrate how these I/O performance specifications affect bandwidth results, we conducted a 24-hour test to measure bandwidth levels between micro, small, and large instances in Virginia and California. The results can be seen in Figs. 5 and 6.

As can be seen, the variation and standard deviation of bandwidth results increase significantly with larger VM sizes. Thus, the micro instance type provides a suitable baseline of small variability on which we center the subsequent analysis.

Despite bandwidth results varying between instance types, instance size does not appear to have an effect on network latency; round-trip times were reasonably uniform across different VM sizes (see Section 4.2.2).

4.2.1. Bandwidth

An Iperf application was used in measurements, one host serving as a server and the second one as a client. A connection between hosts in different data centers was established every 30 min for 5 days. Each connection lasted for 10 s, during which data was transferred at the maximum rate over 5 two-second intervals. Using

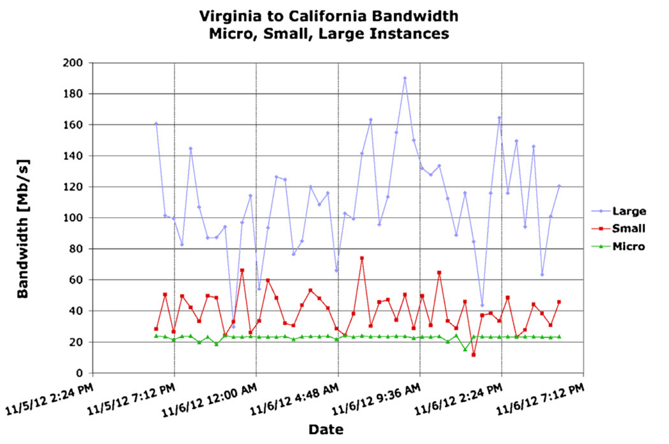

Fig. 5. Bandwidth using micro, small, and large instances: Virginia to California (in Mb/s).

this approach, each connection yielded 5 different samples specifying current throughput at the time. Of these 5 samples, the first was always discarded to take into account the Slow Start algorithm used by TCP. Because TCP gradually increases the amount of data sent over time to assess the maximum bandwidth of a connection, it could not be guaranteed that the first sample in each connection was sending the maximum amount of data possible at the time. As such, to calculate the throughput at any given time, we used the remaining 4 samples to compute an average. For comparison, we collected data without discarding first samples to check how it affected the results–values were 6% to 17% lower, depending on selected data centers.

Fig. 7 shows bandwidth results between micro instances on clusters located in Virginia and the other five data center locations listed above (data is transferred outbound from Virginia). Despite a very small number of large deviations, the majority of bandwidth tests between Virginia and each location appeared stable.

Fig. 8 presents bandwidth as well as latency results between data centers in Singapore and Sao Paolo. Although this connection represents the lowest bandwidth and highest latency averages, the results demonstrate a rather stable connection between these two locations.

Table 7 shows the average bandwidth of each connection examined using the micro instances. It is important to notice that the bandwidth is not proportional to geographical distance. Although the distance from Ashburn in Virginia to Dublin in Ireland is more than 40% higher than to Palo Alto in California, the average bandwidth drops only by 15% from the Virginia-California connection to

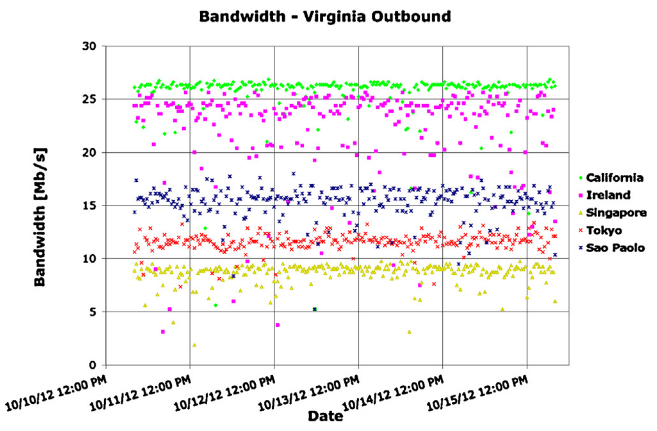

Fig. 6. Bandwidth using micro instances: Virginia Outbound (in Mb/s).

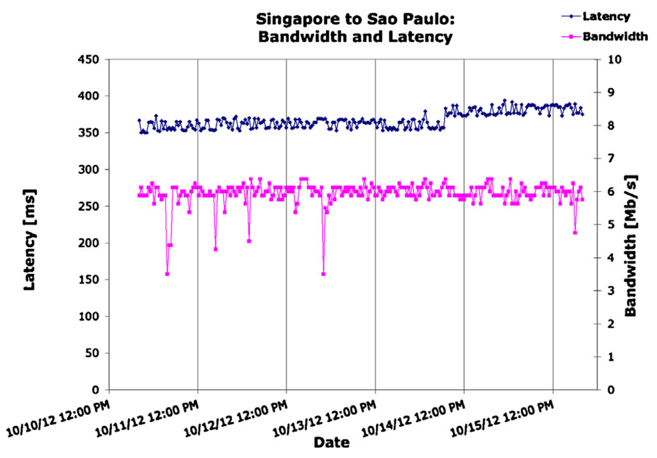

Fig. 7. Connection between Singapore and Sao Paolo.

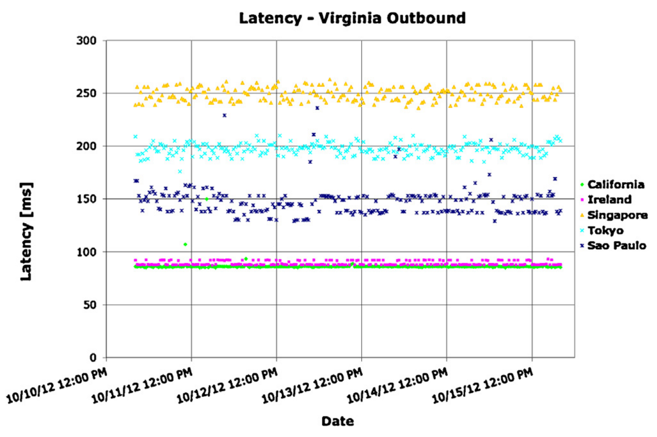

Fig. 8. Latency: Virginia Outbound in (ms).

Virginia to Ireland. This trend is also visible in the next two results. The bandwidth between Virginia and Singapore is more than 65% lower than between Virginia and Tokyo (Table 7), but the difference in distance is only 25%.

Our experiments showed, using micro instances, that the direction of data flow does not have a significant impact on bandwidth between micro instances. The highest difference in bandwidth averages of experiments examining connections of opposite direction is between Singapore and Sao Paolo; this difference, however, is still less than 10%. Most other differences remain at or less than 5%.

However, results from the Virginia–California tests using small and large instances indicate that the direction of data flow does begin to have a significant effect on bandwidth as instance size grows. Using small instances instead of micro instances, the average bandwidth from Virginia to California is nearly 45% higher than that from California to Virginia.

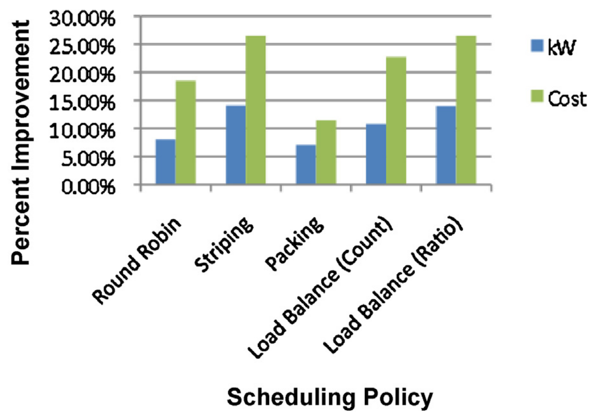

Fig. 9. Energy and cost improvement of Watts per Core.

4.2.2. *Latency*

A ping application was used to calculate the latency between the six data centers. Similar to the bandwidth experiments, a connection was created between each of the data centers every 30 min for 5 days, and the round trip time was recorded.

Fig. 9 presents latency data collected between micro instances on clusters in Virginia and the other five data center locations in the same period of time. As expected, based on the bandwidth results, the average round-trip time between Virginia and California was the lowest of all connections at 86.4 ms. Additionally, connections between the Virginia and California data center were also the most stable, with very few recorded round trip times strongly deviating (+/– 5 s) from the average.

An average round-trip time from Virginia to Ireland is only about 3.4% higher than to California–89.3 ms. Note that the increase in round-trip time is extremely low relative to the increase in geographical distance between these data centers. Table 8 presents the averages of the latency experiments between all 6 data centers.

Despite having the highest average round-trip time of all connections, Fig. 8 demonstrates that latency between Singapore and Sao Paolo remains stable over time.

Unlike bandwidth, latency does not appear to be affected by the EC2 instance type. There were no significant differences between the results obtained from the experiments on micro, small, and large instances.

4.2.3. *Network conclusions*

From both the bandwidth and latency experiments, we concluded that network performance between the Amazon data centers was not temporally variant on timescales of a day. Due to the volatility of the bandwidth between larger instance types, however, bandwidth performance is time-dependent on timescales less than one hour. This indicates that virtual machines can be moved to the optimal data center location irrespective of the time-of-day with the caveat that there exists an apparently random high deviation in performance measures on the temporal order of an hour.

Table 7
Average bandwidth results in (Mb/s).

	Destination						
		VA	CA	IRE	SIN	TOK	SP
Source	Virginia		25.6	22.3	8.64	11.6	15.3
	California	25.4		13.8	12.1	18.6	11.6
	Ireland	23.6	13.8		6.17	7.96	10.7
	Singapore	8.46	11.9	6.03		24.9	5.94
	Tokyo	11.2	18.2	7.68	24.4		7.37
	Sao Paulo	14.4	10.4	9.82	5.44	7.22	

Table 8
Average latency results in (ms).

	Destination						
		VA	CA	IRE	SIN	TOK	SP
Source	Virginia		86.4	89.3	249	197	147
	California	86.5		159	184	128	199
	Ireland	90.1	163		343	289	219
	Singapore	249	184	343		89.2	367
	Tokyo	197	128	287	89.8		289
	Sao Paulo	147	199	219	367	289	

Table 9
Power ratings and average bandwidth by data center location.

Data center location	Power rating (Power costs) × (PUE)	Average outgoing bandwidth (Mb/s)
California	.0944	16.3
Virginia	.1326	16.69
Singapore	.1904	11.45
Ireland	.2563	12.45

The results also indicate that the localities of the data centers do affect network performance. With variations by as much as 300% of both latency and bandwidth results, network differences are significant. Thus, network performance must be taken into account when considering migrating a VM to a different data center location.

Even with the higher bandwidth of the larger instance types, moving infrastructure on the scale of terabytes and petabytes between data center locations is not practical. However, even with a 10 Mb/s worst-case bandwidth, moving a 1 GB virtual machine would only take roughly 2 h. This shows that moving small virtual machines between different data centers across the world could be practical. (Note: The study of alternative network paths–such as re-routing the connection from Sao Paulo to Singapore to route through Virginia–is an interesting optimization opportunity beyond the scope of this paper but is planned for future work).

With latency averages as high as 367 ms, user experience must also be taken into account when considering moving a VM. On sites where users often download large files or where browser latency could cause significant issues, moving the host virtual machines to data centers across the world to which network performance is low would be highly unfeasible.

Finally, corporations, when considering our proposed migrations, must also be concerned about their internal network contention and saturation. If an institution owns or leases physical bandwidth between any two locations, that bandwidth is finite, as is the cost of maintaining it. On any such connection, there are competing demands for bandwidth, such as customer data ingest, customer's inner services communication, corporation's IaaS communications, and possibly our proposed VM migrations. Corporations therefore must prioritize bandwidth usage and determine how much of their network they can allocate to VM migrations.

4.3. Network effects

Large, international hosting providers must consider not only energy consumption and cost efficiency of their clouds but also user experience and bit rate transmissions of their global applications. As such, global VM migration policies are directly affected by network performance considerations, as well as by IaaS providers' network costs and internal network saturation.

When determining where to migrate VMs globally, providers should take into account energy, costs, and performance metrics. In Table 9, we focus on data center power ratings–a metric derived from energy costs and PUE scores–and network performance values. (Note: Data centers located in Tokyo and Sao Paulo were excluded from this table because adequate power rating data could not be found). As one can see, high network performance will not always be the location with the lowest operating costs. In this example of the network that supports Amazon's EC2 instances, Virginia maintains the highest outbound bandwidth while California possesses the best power rating value. Thus, institutions must be able optimize energy usage and operating costs while still taking into account network performance and how it affects their applications.

4.4. Overall

Overall, it appears that the best policy, in terms of minimizing energy costs, is the Watts per Core policy. It is the most energy-efficient of all the scheduling policies in both simulation scenarios, and its cost efficiency was similar to the Cost per Core policy in the original scenario, and slightly better than it in the ideal simulation.

In our original simulation, the Watts per Core policy achieved overall energy consumption 7.1% to 14.1% better than any of the Round Robin, Striping, Packing, or Load Balancing policies. Additionally, it achieved a cost improvement of 11.4% to 26.5% over any of the aforementioned policies. Fig. 10 shows the energy and cost improvement of the Watts per Core policy over the other policies (not including Cost per Core).

Out of the remaining four policies, the Striping and Load Balancing policies do the most poorly in conserving energy but excel at minimizing CPU load, showing up to a 15.6% improvement in CPU load over the other policies.

Additionally, it should be noted that the Round Robin policy offers a middle ground, consistently showing performance in between that of the best and worst categories for all of the performance categories.

5. Conclusion

By choosing a more energy efficient allocation policy, energy consumption on cloud platforms can potentially be reduced by approximately 7% to 14%, lowering overall energy costs by anywhere from 11% to 26%. Such an improvement comes at the cost, however, of increased CPU load.

The effects displayed by the various cloud allocation policies remained fairly constant in both our realistic and ideal scenarios, implying that choosing an appropriate allocation policy will have lasting benefits even as networks and cloud technologies continue to improve in the future.

At greater level of detail, our work to capture current performance metrics relative to an existing global cloud network provides insights into correlations (or lack thereof) between site's operational costs and network bound VM migrations. These insights reveal opportunities to adjust allocation algorithms further with network performance qualifications to avoid performance penalties when minimizing operational costs. This can be seen in the example of California versus Ireland: California maintains a low power rating paired with high bandwidth values, while Ireland has a much larger power rating with lower bandwidth performance.

It is also worth noting that the policies presented here can be quickly and trivially implemented in existing cloud platforms, without the need for external systems, providing cloud administrators with a convenient and simple way of improving energy efficiency and lowering energy costs across their data centers.

5.1. Future work

While our simulation outlines the basic effects of various virtual machine allocation policies, there are a number of improvements that could be made to make it more applicable to a real world scenario. Currently, the simulation makes a number of assumptions that might not hold true in an actual data center.

First, the simulation is centered on a single application, reddit.com. A real data center would host multiple applications, each of which having a unique configuration of virtual machines as well as unique cluster load at different times of the day. To account for this, the simulation could be run against a larger, more diverse set of virtual machines, and it could simulate different loads across different subsets of those virtual machines.

Second, the simulation does not consider communication between virtual machines across regions. In reality, this incurs a large performance cost and may not be desirable. For example, in our scenario, an Application VM might need to talk to a Database VM, but if they are located in separate regions, this could drastically degrade the performance of the application as a whole. In our simulation, we make the assumption that each cluster has its own shared storage. In such a case, VMs in one region are not expected to frequently access VMs in another region.

That said, to combat this problem in practice, applications usually specify that certain resources or VMs need be co-located, depending on the specific performance needs of the application. The simulation could be improved to allow for such co-location specifications, ensuring that certain VMs will be grouped in the same region.

Third, the simulation currently moves VMs between regions as necessary. In reality, it is difficult to do this–depending on the parameters of the connection between these data centers. On the other hand, as we noted in Section 4.2.3, moving small virtual machines is practical. We can also imagine a situation where some dedicated links with higher bandwidth are used to transfer VMs, in such case the transfer times would be lower.

Fourth and finally, our simulation does not take into account Quality of Service (QoS) agreements. In practice, data centers usually need to maintain Service Level Agreements (SLA) for virtual machines, typically expressed as CPU or response time guarantees. Lowering the operating costs by reducing the energy consumption without increasing the number of SLA violations requires consideration of some tradeoffs.

It is possible, however, to at least minimize SLA violations by slightly modifying our algorithms. Our simulation could be improved to track and measure historical data and then forecast future demand. For example, in the case of Reddit, it is possible to estimate the load percentage of each cluster by taking into account the average number of users throughout the world and then observing that users are most active in the evening and night hours. The migration of VMs could then be scheduled for early morning hours to minimize SLA impacts. Future work will involve exploring solutions like this one, as well as specifying SLA guarantees in the simulation itself.

Acknowledgments

The authors would like to recognize Professors D. Thain and D. Go for their insights and helpful conversations. Partial support for this work was provided by J. Nabrzyski and the Notre Dame Center for Research Computing, as well as a Department of Energy grant for the Northwest Indiana Computational Grid. The authors would also like to thank Ariel Oleksiak and his team at the Poznan Supercomputing and Networking Center for their helpful insights and discussions.

References

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/
[2] R. Miller. Strong growth for Amazon EC2 in Ireland. Data Center Knowledge. http://www.datacenterknowledge.com, December 2010.
[3] US Environmental Protection Agency, Report on Congress on Server Data Center Energy Efficiency, August 2007.
[4] D. Linthicum. Why cloud adoption is driving hardware growth, not slowing it. Info World. http://www.infoworld.com, August 2010.
[5] J. Koomey.Growth in data center electricity use 2005 to 2010. Analytics Press. http://www.analyticspress.com/datacenters.html, August 2011.
[6] S. Garg, C. Yeo, A. Anandasivam, R. Buyya, Energy-efficient scheduling of HPC application in cloud computing environments, Distributed, Parallel, and Cluster Computing (2009).
[7] K. Kim, A. Beloglazov, R. Buyya, Power-aware provisioning of cloud resources for real-time services, in: Proceedings of Middleware for Grids, Clouds, and e-Science (MGC) 2009, Urbana-Champaign, IL, 30 November–1 December 2009, 2009.
[8] J. Chase, Managing energy and server resources in hosting centers, in: Proceedings of 8th ACM Symposium on Operating Systems Principles, Banff, Canada, 2001.
[9] M. Mazzucco, D. Dyachuk, Optimizing cloud providers revenues via energy efficient server allocation, Sustainable Computing: Informatics and Systems (2011).
[10] A. Beloglazov, R. Buyya, Energy efficient allocation of virtual machines in cloud data centers, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing (CCGrid), 2010.
[11] D. Aikema, A. Mirtchovski, C. Kiddle, R. Simmonds, Green cloud VM migration: power use analysis, in: 2012 International Green Computing Conference, IGCC 2012, San Jose, CA, USA, 2012.
[12] R. Jansen, P.R. Brenner, Energy efficient virtual machine allocation in the cloud, in: International Green Computing Conference (IGCC), 2011.
[13] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: HotPower'08 Proceedings of the 2008 Workshop on Power Aware Computing and Systems, San Diego, CA, 2008.
[14] X. Zheng, Y. Cai, Energy-aware load dispatching in geographically located internet data centers, in: Sustainable Computing: Informatics and Systems, 2011.
[15] Eucalyptus. http://www.eucalyptus.com/
[16] OpenNebula. http://opennebula.org/
[17] Reddit http://reddit.com/
[18] M. Schiraldi, Reddit: billions served, Reddit Blog (2011).
[19] R. Miller. Where Amazon's data centers are located. Data Center Knowledge. http://www.datacenterknowledge.com, November 2008.
[20] IBM System X and Blade Center Power Configurator. http://www-03.ibm.com/systems/bladecenter/resources/powerconfig.html
[21] U.S. Energy Information Administration. http://www.eia.doe.gov
[22] Europe's Energy Portal. http://www.energy.eu/
[23] Senate of the Phillipines, Electric power at a glance, Senate Economic Planning Office (2005).
[24] C. Brady, A. Rawson, J. Pflueger, T. Cader, Green grid data center power efficiency metrics: PUE and DCIE, Microsoft, AMD, Dell, Spraycool (2008).
[25] Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types/