Utilization-aware Allocation for Multi-Tenant Datacenters

Yang Gao[*][†], Yibo Xue[†][‡] and Jun Li[†][‡]
[*]Department of Automation, Tsinghua University, Beijing, China
[†]Research Institute of Information Technology, Tsinghua University, Beijing, China
[‡]Tsinghua National Lab for Information Science and Technology, Beijing, China
{gaoyang11}@mails.tsinghua.edu.cn, {yiboxue, junl}@tsinghua.edu.cn

Abstract—**Cloud datacenters need efficient resource management that is able to orchestrate bulks of different resources. In current public cloud datacenters, there is a mismatch between the properties of tenant requests and resource utilization. Multiple resource types in datacenters make the situation even more complex. Intelligent resource allocation methods are demanded to improve the overall utilization of multiple resources. This paper presents Multiple Resource Utilization (MRU), a novel resource allocation method for multi-tenant datacenters. MRU allocates multiple types of datacenter resources according to the diverse and dynamic requests from tenants, and simultaneously maximize resource utilization of datacenter and guarantee the resource availability and performance for tenants. Experiments demonstrates that MRU significantly improves the utilization of multiple resources and thus save the cost.**

Index Terms—**multiple resource, data center, allocation, utilization**

I. Introduction

Datacenters play an important role in Cloud Computing. More and more companies are migrating their system and platform into datacenters. Large enterprises often build up their private datacenters to assure the service to be provided under control. Small businesses or startups usually rent remote resources from public datacenters to benefit from the low cost and flexible performance of the resource sharing platform.

As there are numerous resources in datacenters, resource allocation becomes a very tough job. The tenants and datacenter operators have different requirements for the resource allocation job. For the tenants, the most important is to ensure the allocation fairness, hence they will be able to deploy application or service with guaranteed quality. For datacenter operators, they are more likely to care about the utilization of resources to save capital and operation expenses of the facility. Fortunately, the development of virtualization technology helps a lot for this situation. For fairness, the parameters of VMs are easy to be set up by the virtualization hypervisor and the basic performance that it indicates will be ensured undoubtedly. For utilization, virtualization makes a fine-grained use of resources. Several VMs share one server and each VM can be allocated to tenants so that resources are shared and the utilizations will be improved.

Though virtualization technology benefits a lot for datacenter resource sharing [1]–[3], resource allocation is still a complex issue. In a large enterprise that owns a private datacenter, people from various departments sharing the whole datacenter together and the datacenters need to finish the tasks efficiently. Because tenants here do not need to pay for the resources they applied for, the fairness property is as important as the utilization property. So resource allocation in private datacenters should make a trade-off between them. And it is also necessary to provide strategy-proofness so that tenants cannot manipulate their resource demands to get a better performance by cheating.

However, for public datacenters, *e.g.,* Amazon EC2 [4], resource allocation focuses on utilization property because of its virtual machine rental service model. The tenants of a public datacenter will pay for the amount of resources that they reserve, therefore they have no interest to provide false requests to administrators about their demands, and the resource allocation does not need to preserve fairness property. Datacenter operators only expect a high utilization of resources because of less power consumption for them.

Another fact from the Amazon EC2 is that its operational load is around 70%. This indicates that datacenters are not always running at full load. When a datacenter is not at full load, the allocation should occupy lesser servers to provide a full-cycle utilization optimization, rather than spread out evenly across all servers. For instance, to handle 1000 requests in 200 servers, when the system accepts 500 requests, it is better to have 100 servers occupied rather than 200 servers occupied. The allocation method should provide a high utilization of resources at any load so that more tenant requests are likely to be accepted which leads to a larger business values for the datacenters.

The difficulty of public datacenter resource allocation also comes from the objective mismatch between tenants and datacenter operators. As tenants, they may have different amounts of subscription for resources because of their diverse applications. For example, a tenant that provide personal website may subscribe only little resources per VMs because the tenant does not expect the website will attract heavy network traffic. But the tenants who run public information platforms or forums may demand much more resources or even a load balance service to solve the possible large visiting traffic. Also, the resource types that different applications required are not always identical. Online game servers may ask for more memories capacity, while searching and distributed computing will need more CPUs. The attractions of public datacenters for tenants are that they only need to pay for the resources as much as they use. This brings about the result of various request types with different resources demands. On the other hand, datacenter commonly operates on large number of servers with identical configuration to leverage on economy of scale in purchasing and simplify hardware maintenance tasks. Therefore, facing various requests that come from the tenants, an efficient multiple resource allocation method is highly desired to maximize overall resource utilization of each physical server and each server cluster.

The common solution to resolve this conflict is to generate a series of templates with several resources reservation choices. The datacenter operators can simply provide VMs with (CPU: 2 cores, Mem: 4 GB), VMs with (CPU: 8 cores, Mem: 16 GB) and so on. Tenants will have to choose from these templates to satisfy their requirement. This solution benefits the datacenter operators on resource allocation for they can divide the servers into certain pieces. But it is not fair or desirable for tenants because only tenants themselves know the type and capacity of resources they need and most of tenants prefer to pay for the amount of resources that they actually require. To satisfy both the tenants and the datacenter operators, a dynamic resource allocation method is expected. Tenants can generate their requests independently under this dynamic method while datacenter operators also benefit from it for they can achieve a higher resource utilization to save their cost.

The request diversity also limits the developing of resource allocation method. Sometimes the tenant's requests do not match the resources provided. For example, there are two types of tenant requests. One tenant doing search requires more CPUs but less memories than average cases (CPU-heavy) and another tenant serving game requires more memories but less CPUs (Memory-heavy). In a certain period, the rate of CPU-heavy requests and Memory-heavy requests may vary dramatically and cannot meet the resources mix provided by datacenter. Because it has no reason to reject the tenant's requests on the premise of having enough resources, once the requests are accepted, there will be a waste of resources. So the goal of resource allocation is to achieve a self-adapting solution for this kind of demands.

Based on the above observations in public datacenters, we present multiple resource utilization (MRU), a novel resource allocation method to improve the utilization of multiple resource types. The virtual machine rental service model in today's public datacenters is that tenants apply for VMs with different resource requirements and the administrator allocates them on the available servers. Choosing the right server to place a typical VM in order to optimize the utilization is the goal. With the MRU allocation method, the resource waste is reduced and the utilization is improved. This means VMs will be centralized so that the charge for power is decreased. And datacenters operators can make more money with more requests accepted. Moreover, MRU is a dynamic and self-adapting allocation method. Tenants can generate their requests based on their demands independently. MRU will solve this situation by adding a new limit to the allocation condition which can help balance the usage of different resource types. Also, such improvement can help to self-adapt the requests to the servers so that the high resource utilization is achieved. This means both the tenant request variety and the unpredictable request diversity situation can be handled by MRU.

This paper is organized as follow. Section 2 is the existing resource allocation methods for both public and private datacenters. Section 3 is the design and discussion of the MRU allocation method. In Section 4, MRU is compared with Domain Resource Fairness (DRF) and Best-Fit method to illustrate its advantage in utilization. And Section 5 is the conclusion of the work.

II. Related Work

There are several previous works on resource allocation problem for both private and public datacenters. Shieh et al. [5] proposed a network bandwidth allocation scheme named Seawall. This scheme uses a weighted additive increase, multiplicative decrease (AIMD) based adaptation logic to dynamically allocation the resource. Although Seawall can be applied in both private and public datacenters, it is designed to provide the interconnection network capacity division instead of solving the resource allocation problem under the virtual machine rental service model. Besides, with an AIMD-based adaptation logic and a feedback mechanism, the scheme cannot provide a strict performance guarantee for the tenants.

Domain Resource Fairness (DRF) [6]–[8] is a model and also an available solution for the private datacenters, which makes trade-offs between the fairness and utilization properties. The Domain Resource of a tenant request is identified to be the resource type which is the maximum share that the user has been allocated of any resource. Different from the intuitive max-min fairness method that maximizes the minimum allocation received by a tenant in the datacenter, DRF considers about multiple resource types and makes sure that tenants with different domain resources share the datacenter resources fairly. As the fairness property is achieved based on the volume of domain resources, the utilization of the resources has also been improved.

Ballani et al. [9] and Xie et al. [10] proposed two different solutions for resource allocation of public datacenters. The former uses a greedy algorithm to select nodes that satisfy the network bandwidth requirement on the communication path. And the latter generates models based on the VM's different network bandwidth requirements in the lifecycle of it and dynamically allocates network bandwidth. The two existing works both improve the utilization of the servers but only concentrating on a single resource, *e.g.,* network bandwidth. Gurusamy et al. [11] also focused on the same situation. It presented a three-phase mechanism to find the set of servers for requested VMs in order to reduce the bandwidth on shared links so that a datacenter can host more cohabiting tenants. Our previous work SEAL [12] is an allocation method for single resource in multi-tenant datacenters that combines two classical allocation algorithms, Best-Fit and Next-Fit, to provide an agile and efficient allocation.

III. MRU Allocation Method

In MRU design, domain resource type is used to identify different requests, and it adapts the requests to the allocation condition. When a request is adapted and the VMs associated to the request is successfully started on a chosen server, this is called an acceptance. But if all the servers are traversed and no suitable server is found to realize the request, it will be rejected. This is called a rejection.

A. Method Description

The virtual machine rental service model is a request/reply model. A request consists of a series of resource subscription including the volume of different resource types as $\{R_1 : r_1, R_2 : r_2, \ldots, R_n : r_n\}$. Assuming R_1 stands for CPUs and R_2 stands for memories, a request, (CPU: 2 core, Memory: 4 GB), can be shown as $\{R_1 : 2, R_2 : 4\}$. Correspondingly, supplied resources on the servers can be indicated as $\{RS_1 : rs_1, RS_2 : rs_2, \ldots, RS_n : rs_n\}$. The values of supplied resources are the available capacity and will be updated every time an allocation instance finishes.

The allocation of MRU is determined by request identification and allocation condition. Request identification aims to classify tenant requests into several types. Similar to DRF, we use Domain Resource to identify the tenant request which is used to adapt the allocation condition so that the right server will be found out. The conception of Domain Resource of DRF method has been declared in Section 2. Each request will have only one domain resource and is identified with it. With request identification, MRU can make a balance of the usage of different resource types in one server so to achieve the high utilization. Allocation condition is used for the acceptance and rejection of identified requests with servers. Allocation condition consists of adaption condition and additional condition. Only when these two conditions are both satisfied will the request be accepted by the server. The adaption condition is for the performance guarantee because one basic requirement of virtual machine rental service model is to guarantee the tenant's performance. This condition is necessary for all the allocation schemes including the basic one without any concern about utilization or other properties. The additional condition is used for the utilization improvement. As there may be several servers that can satisfy the adaption condition, the allocation of requests can be finer grained. If the requests with different domain resources can co-exist on one server, the total utilization will be improved.

Upon the description above, the procedure of the MRU allocation method is shown in Fig.1. In the beginning of the procedure the domain resource of the request is identified. Then every available server will be checked for the adaption. If the request can satisfy both the adaption condition and the additional condition within a server, the request is accepted and resources on the server are allocated to the request. Otherwise, the request is rejected. When an allocation instance finishes, an update of supplied resources for the server is necessary, although not shown in the procedure figure.

```
Require:   $n$ resource types & $m$ servers
Ensure:    Allocation for request $r$
1.  Identify the domain resource of $r$
2.  For round $i$ form 1 to $n$
3.      For server $j$ from 1 to $m$
4.          Get the $i$th resource type from $j$'s
            domain resource table
5.          Compare $r$'s every resource subscriptions
            Rr to $j$'s remaining resources Rsj
6.          If all Rr <= Rsj
7.              If resource type $i$ same to $r$'s
                domain resource
8.                  Return accept
9.   Return reject
```

Fig. 1. The procedure of MRU allocation method

B. Request Identification

The tenant requests are diverse, so request identification is needed before adapting. For example, a request with (CPU: 2 cores, Memory: 4 GB) is surely different with a request with (CPU: 6 cores, Memory: 2 GB). If a request with subscriptions $\{R_1 : r_1, R_2 : r_2, \ldots, R_n : r_n\}$ demands resource R_i more than any other resources, its domain resource dr is identified to be R_i. The question is how to compare different resource types to get the most demanding resource. In fact, different resource types have their own quantization unit like CPU with "cores" or Memory with "GB", etc. Resources need to be normalized with a set of standard values first to make volume comparison of different resource types.

Obviously there are various choices for the standard values. It can be the initial resource capacity, the remaining resource capacity after the last allocation, or even a static value that is given by the administrators. Different choices of standard value may give the same request different identifications, which finally influences the allocation consequence. As a consequence, the choices depend on administrator expectations. In this paper, the initial capacity of each resource type is selected to be the standard value. This is based on the objective of resource utilization maximization, so whether a request is CPU-heavy or Memory-heavy also depends on the server configuration in datacenters.

C. Allocation Condition

Allocation Condition consists of adaption condition and additional condition. Based on the supposition of performance guarantee for tenants, the basic adaption condition of MRU allocation method is described as: Given supplied resources of a server, if the resource requirements of a request satisfy $\{R_1 \le RS_1, R_2 \le RS_2, \ldots, R_n \le RS_n\}$, the request is a feasible request. Once a request becomes feasible, it will be accepted in any cases. Since there may be multiple servers that are eligible, different choices will have a different utilization performance.

The building up of additional condition is similar to request identification. Each server associates with one table that records the domain resource list of the server. This table is generated for the additional condition and will be updated every time an allocation instance finishes. Every element in the list is given by *er*:

$$er = \{i \in [1, min(k_v, n)], s.t. max(\frac{RS_i}{STANDARD\_V_i})\}$$

Each time a resource type is chosen, it will be recorded on the list and the left resources will be used for the same calculation till no resources left. k_v means the number of resources left and $STANDARD\_V_i$ is the standard value used for the value normalization of different resource types. The expression above is to find out the resource type that has the maximum domain resource share. Similarly, there are numerous choices for the standard values and it can be static or dynamic. The static standard values also can be simply set to the initial resource capacity. But it is not a good choice because the resource requirements of different tenants are changing all the time and the arrival time of tenant's requests is unpredictable. Therefore, a set of dynamic standard values is more suitable into this situation. Using a sliding window of *m* requests, the standard values can be calculated as the average resource subscriptions of the *m* requests. These values are called the historical statistic values and can offer heuristic information so that the allocation can be more flexible and follow demand trend in the long run. Also the window width "m" should ideally reflect the pattern of request sequence, and also selected based on the scalability of datacenter to achieve a better allocation performance.

Additional condition provides the allocation optimization. No matter with tenant request variety or unpredictable request diversity situation, additional condition can help to balance the resources usage in case of leaving too many resources unused when one resource uses up. This indicates that the additional condition in MRU can adds the dynamically adjustability to the tenant requests and provide self-adapting to the unpredictable request diversity. Both these two benefits lead to the utilization improvement.

D. Method Discussion

It has been known that the most efficient allocation method for single objective is the Best-Fit method [13]. But this method is not suitable in the multiple resource allocation according to our experiments. Best-Fit searches the resource pools in a certain order and once a satisfied pool is found the resources will be allocated. In this multi-dimensional situation, it means that if the adaption condition is satisfied the request is accepted and allocated on that server. Obviously, this is thoughtless of resource utilization because the usage of different resource types is not considered so that the consequence of resource usage is unpredictable. Different from Best-Fit, MRU allocation method built up domain resource tables for servers so that the resource usage can be balanced to improve the utilization of overall resources. In fact, thinking about one resource type, the tables grade servers into several groups with priority. The adapting procedure of MRU is actually an adapting from the highest priority servers to the lowest priority servers and in every priority it still can be regarded as a best-fit action. Extremely, when the remaining domain resource type number decreases to 1, since the domain resource of tenants and the domain resource of servers are the same, MRU regresses to be the Best-Fit method.

When building up the allocation condition, a historical statistic value is used to be the normalization standard value. This is because the mismatch between the subscription of tenant requests and resource provided by the datacenter operators. In consideration of cost, it is common for public datacenter to have large group of uniform servers. But each tenant's demand for multiple resources is unique and variable. The resource subscriptions of a tenant may be vary significantly in different time of a year, let alone the request diversity phenomenon. The historical statistic can handle this situation in certain degree. The so-called normalization of the server's resources with historical statistic standard values restricts the ratio of the resources allocated to be around the average ratio of resources of the request so that the server can be used efficiently. With request identification, varieties of tenant requests are classified into several types, *e.g.,* CPU-heavy or Memory-heavy. Requests with different domain resource types co-existing on the servers help balance the resource usage. Historical statistic values here provide the standard that how the balance is defined. Actually, compared to the historical statistic, when more CPUs are used on the server, a Memory-heavy request is more likely to be accepted by the server unless it is nowhere to be allocated.

It is known that sometimes a oversubscribed allocation is used in the shared resources pool and with a appropriate oversubscription value both the resources utilization and the demands of tenants can be achieved. But in the problem description above the problem is simplified to be a non-oversubscription allocation problem with performance guarantee property. This simplification is very helpful for the problem modeling. Besides, the oversubscription allocation situation can also be realized in this model by setting the server's virtual resources usage according to the oversubscription value and the user's actual resources usage. This situation is significant for network resources and storage resources which are not the focus of this paper.

In this paper, most of the situations or examples are about two resource types. But MRU allocation method is designed to solve the allocation problem of more than two resource types. Future on MRU can introduce more resource types, including network bandwidth, to explore its advantages in more complicate datacenter resource allocation scenarios.

IV. Evaluation

MRU is evaluated in four aspects. First, we set up a general simulation and show the superiority of MRU on resource utilization directly. Second, MRU is validated to deal with request diversity. Then, the status of servers in the whole

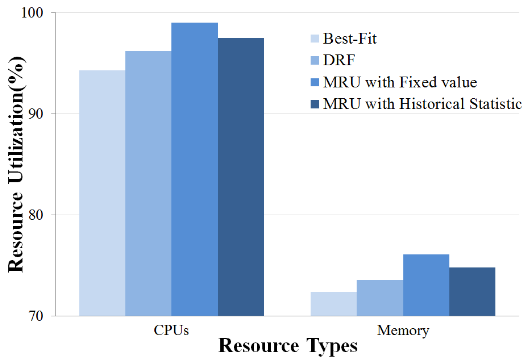

Fig. 2. The resource utilization of Best-Fit, DRF, MRU with Fixed standard value and MRU with Historical Statistic standard value when servers are under full load

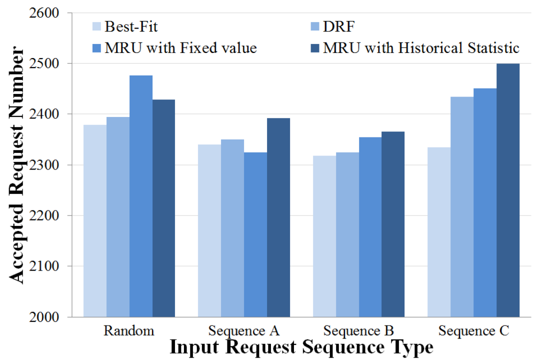

Fig. 3. The allocation results of Best-Fit, DRF, MRU with Fixed standard value and MRU with Historical Statistic standard value with four types of request sequence

process of allocation is laid out to illustrate MRU's energy-saving property. Finally, the impact of sliding window is analyzed. Two resource types, which are CPUs and memories, are considered in the evaluations, and the simulation is at the scale of 100-500 servers and 1,000-3,000 requests. Two different requests types are employed: the CPU-heavy request and the Memory-heavy request.

A. Utilization

In this experiment we use a random request sequence with the two request types evenly mixed. Under the full load of requests, we calculate the average resources usage of all servers to show the overall utilization status. Fig.2 shows the experiment result. Faced with a multiple resources situation, Best-Fit cannot achieve the best utilization but performance the worst which proves that the Best-Fit algorithm is not suitable to the multiple resources allocation situation. In contrast, MRU is better than the other two methods. Since the input requests are irregular, the statistic has little effect to the optimal allocation. Using a static value (such as the resource limit value of the server) as the standard value is better than historical statistic value because the average resources demand is predicable in this condition. This experiment shows the benefit of MRU's dynamic property.

The high utilization that MRU achieves can result in the saving of the cost. As for a certain number of servers, the higher resource utilization means the larger number of requests that can be accepted. Thus, the datacenter operators can make more money by the extra accepted requests. Because MRU is based on the tenant performance guarantee property, the tenant's benefit can be hardly hurt.

B. Request Diversity

We generated four request sequences with the two types of requests for request diversity handling test. Sequence A has more CPU-heavy requests and few Memory-heavy requests and Sequence B is contrary. Sequence C takes the first half of Sequence A and the second half of Sequence B and make up a hybrid one. In Fig.3, the three sequences are applied and each one has a length of about 1,000 requests. The results show

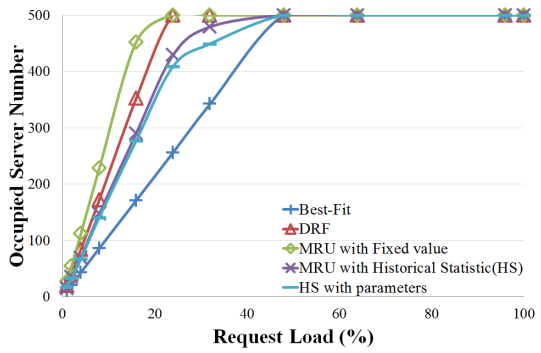

Fig. 4. The occupied server number growth of Best-Fit, DRF and MRU with various standard value choices

the benefit of historical statistic standard value and the limits or instability of the fixed standard value. Tenant requests are expected to have non-uniform distribution characteristics in practice, and thus MRU with historical statistic standard value can be self-adapting for the diverse request sequence.

C. Energy-Saving

To save energy, it is better to occupy less number of servers for the same number of requests. In this experiment, the occupied server number is measured at different request load. The request sequence is Sequence C in Section *IV.B*. Fig.4 is the occupied server number growth of five allocation methods. MRU with fixed standard value has the worst performance and it is even worse than the DRF method. The Best-Fit method is no doubt the best performer in energy-saving but with a high request rejection rate, which is not shown in the figure. With historical statistic standard value, MRU can achieve a better performance than DRF with a low request rejection rate. In addition, when we combine the static value (the resource limit value of the server) with the historical statistic standard value, MRU can achieve a better performance than the original with little reduction of accepted request number. This again indicates that the choice of standard value determines the performance of the allocation in many aspects.

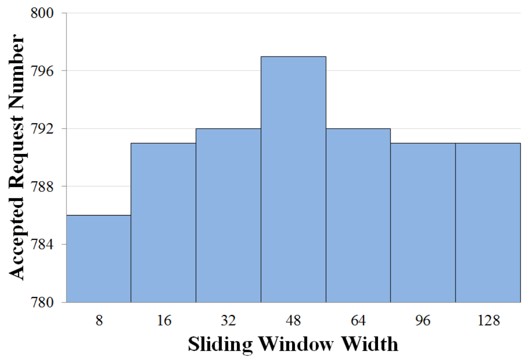

Fig. 5. The influence of historical statistic sliding window width on the allocation results

D. Sliding Window

Last but not the least, the impact of sliding window for statistic is evaluated. The experiment scale is 1,000 requests with 100 servers, and the experiment result in Fig.5 shows that the sliding window size only has limited impact to the allocation result. As shown in the figure, the accepted request number of the best choice is averaging about 1.4% more than that of the worst choice. Although the impact of sliding window size is small, it is still an important factor which can influence the allocation. First, the allocation scale can be large and varies [5]. When the scale is too large or too small, the impact can be obvious than the normal scale. Second, there is a calculation of historical statistic value where the sliding window size may affect the efficiency of the method.

V. Conclusion

In this paper, a novel allocation method for multiple resource types in datacenters is proposed. Namely MRU, the method is designed with intention to solve the problem of the mismatch between tenant requirements and the resource availability. It dynamically allocates resources for unpredictable requests of multiple tenants, and still achieves high overall utilization. Simulation shows that not only MRU can improve the utilization of multiple resources on server, but also save the energy at any request load. This benefits the datacenter operators a lot for that they can save the cost and maximize the usage of their resources provided.

Future work can progress in two directions. One is to extend the method and evaluation to more than two resource types. Multiple resource types may lead to a more rigid allocation constrains and increase the cost of utilization improvement, hence further improvement of MRU may be needed to maximize utilization. The other is to analyze the existing request sequence and find out the pattern of the input to find more accurate standard values, such that a more detailed and targeted solution can be given to the particular case.

References

[1] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," *Proc. HotNets (October 2009)*, 2009.

[2] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "Netlord: a scalable multi-tenant network architecture for virtualized datacenters," *SIGCOMM-Computer Communication Review*, vol. 41, no. 4, p. 62, 2011.

[3] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[4] Amazon ec2. [Online]. Available: http://aws.amazon.com/ec2/

[5] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association, 2011, pp. 23–23.

[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association, 2011, pp. 22–22.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *USENIX NSDI*, 2011.

[8] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queueing for packet processing," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 1–12.

[9] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," Technical Report MSR-TR-2011-72, Microsoft Research, Tech. Rep., 2011.

[10] D. Xie, N. Ding, Y. Hu, and R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 199–210, 2012.

[11] M. Gurusamy, T. N. Le, and D. M. Divakaran, "An integrated resource allocation scheme for multi-tenant data-center," in *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*. IEEE, 2012, pp. 496–504.

[12] Y. Gao, L. Li, J. Jiang, B. Yang, Y. Xue, and J. Li, "Seal: Hybrid resource distribution for multi-tenant data centers."

[13] P. Wilson, M. Johnstone, M. Neely, and D. Boles, "Dynamic storage allocation: A survey and critical review," *Memory Management*, pp. 1–116, 1995.