Peer-to-Peer Bargaining in Container-Based Datacenters

Yuan Feng, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*yfeng, bli*}*@eecg.toronto.edu*

Bo Li
Department of Computer Science
Hong Kong University of Science and Technology
bli@cs.ust.hk

Abstract—**In container-based datacenters, failure-prone components are sealed in pre-packaged shipping containers, and component failures over time reduce the availability of resources. From the perspective of services, application instances can usually be migrated across the boundary of containers as virtual machines (VMs). In such an environment, it would be sometimes beneficial to migrate application instances to take better advantage of available resources. In this paper, we first point out that application placement strategies that are singularly focused on the efficiency of resource usage may not be beneficial, especially when resources are over-provisioned in container-based data centers. We believe that application instances should be placed based on the *Buffet principle*, using resources in an aggressive fashion. Once failures occur, application instances can be migrated across containers, by allowing containers to bargain with each other in a peer-to-peer fashion, treating application instances with different resource requirements as "commodities" in a Nash bargaining game. We show that the ratio of utilizing resources can improved by establishing such local Nash bargaining games, where two containers bargain and trade VMs with each other to increase their own utilities. With simulation, we verify the effectiveness of the proposed bargaining games.**

I. Introduction

One of the design objectives of container-based (modular) datacenters is the management of complexity and cost of deployment. Still, a basic configuration of the Sun MD S20 modular datacenter, for example, is quoted at $\$559,000$ [1], which represents only a fraction of the total cost of ownership. As a consequence, it is important to achieve a high level of resource utilization, when it comes to computational, bandwidth, and storage resources. Unfortunately, utilization in operational datacenters can be remarkably low, even only 10% [2].

Recent success in virtualization technologies provides possible solutions to improve resource utilization. Such technologies encourage datacenters be transformed from existing rigid IT configurations to a more agile infrastructure [3]. Server virtualization techniques, such as VMware and Xen, facilitate *application instances* to be packaged inside *virtual machines* (VMs). These VMs can then be migrated from one server to another without any downtime to the application running within. In addition to virtualizing computational and bandwidth resources, storage virtualization techniques, such as the IBM SAN Volume Controller and Sun ZFS, are able to virtualize storage space into virtual disks (Vdisks), which can be easily migrated from one storage subsystem to another.

As live migration of application and storage instances become feasible by using virtualization, existing work (*e.g.*, Singh *et al.* [4]) has proposed centralized algorithms to manage a load-balanced cluster of servers by migrating VMs across server boundaries. By migrating VMs off *overloaded* servers or storage nodes, called "hotspots," to under-utilized servers, performance degradation or premature failures due to overload can be avoided or mitigated. Such centralized load management and VM migration is triggered by overload situations in hotspots, and migration is performed across the boundary of individual servers.

In this paper, we argue that such fine-grained centralized micromanagement to alleviate hotspots represents a step towards the right direction, but is still too microscopic to be realistic in modern *container-based*, or *modular*, datacenters. In container-based datacenters, we believe that computational, bandwidth and storage resources are abundantly available, but component failures over time are the norm, rather than the exception [5]. This is due to the fact that, once such server-packed shipping containers are sealed and operational, it is very difficult to repair or replace their components individually. Though the reliability and maintainability of resources are theoretically described by MTBF or MTTF, failures in different resource dimensions in distinct containers may follow their own degradation distributions.

In this paper, we advocate the application of the *Buffet principle* [6] when it comes to launching application instances to utilize abundant resources in container-based datacenters. The *Buffet principle* stipulates that, rather than carefully optimizing resource usage for efficiency, one can launch as many application instances as needed to utilize *all* available resources in an aggressive manner, as long as the marginal benefit outweighs the marginal costs. While the Buffet principle fits well in the unique characteristics of container-based datacenters, are we able to handle a reduction of resource availability over time due to component failures, if nearly *all* resources are being utilized?

Live migration of VMs comes to our rescue again, but at a much coarser granularity. We argue that VMs should be migrated across the boundary of *containers*, rather than servers, and only when it is necessary to do so due to a lack of resources (caused by failures). In this paper, we propose that containers should be treated as *peers*, and *bargain* with each other in a *peer-to-peer* fashion in a local trading market, with VMs traded as "commodities." Such bargaining games should only be triggered when there exists a resource deficiency, and should terminate when the utilization ratio of resources is relatively balanced. Such a design is simple and realistic to be implemented, and takes place in an autonomic and self-organized manner, precisely as any other bargaining trading markets.

The remainder of this paper is organized as follows. In Sec. II, we briefly introduce the Buffet principle and the VM placement strategy. After showing the benefits of redistributing VMs across different containers, Sec. III describes the decentralized VM migration algorithm based on the Nash Bargaining Solution. In Sec. IV, we show the effectiveness of the proposed VM migration algorithm. We conclude this paper in Sec. V.

II. Applying the Buffet Principle

Modern applications hosted by container-based datacenters are highly diverse in their resource requirements. For example, video streaming encoders requires substantial CPU computational resources; while one-click online hosting services have a huge appetite for storage space. Generally, to ensure efficiency, the number of application instances, in virtual machines (VMs), is determined within the context of the application. For example, the default number of replicas in Google File System is 3 [7], while alternative email services may require as many as 15 replicas.

After the number of application instances is determined, each of these instances is then deployed according to certain load balancing considerations. The design objective of these strategies is the *efficiency* of utilizing resources, which attempt to find the "sweet spot" of operation, such that the performance per unit of resource consumed is maximized.

However, a singular focus on efficiency may lead to the under-utilization of resources. Since application demands vary quickly over time, it is difficult to predict such demands accurately. To guarantee service quality during peak hours, container-based datacenters routinely over-provision resource availability. If resources provisioned in a datacenter has the ability to support 5 instances of 100 different applications, yet only 3 instances of these applications are deployed within, the strategy unnecessarily wastes resources even when they are readily available.

We believe that this is a classic example where the *Buffet principle* [6] should be applied. The Buffet principle argues that resources should be utilized as long as the marginal cost is lower than the marginal benefit. In the context of designing a strategy to determine how many application instances should be placed in each container, we may simply let each container accommodate as many application instances as it can to saturate nearly all of its available resources, with respect to either bandwidth, CPU, or storage space. The Buffet principle is a natural fit in this context for the following reasons:

▷ *Low marginal costs.* Once provisioned during design, resources in a container would be wasted if not utilized. This implies that the *marginal cost* of using such resources is exceedingly low. Though performance gains beyond the sweet spot may not be substantial, the marginal benefits — such as improved resilience to failures if more replicas are maintained — may still outweigh the marginal costs. With a container costing millions to acquire and hundreds of kilowatts of power to keep up, leaving resources unused may not rational.

▷ *Simple design.* Based on the Buffet principle, there is no need to design elaborate schemes to determine the number of application instances (as VMs) to be deployed in each container. The maximum number of VMs is deployed to utilize all available resources.

▷ *Enhanced resilience.* With the service demand fluctuating quickly and the network changing dynamically, the "sweet spot" is hard to be determined accurately. When experiencing a bursty increase of requests within one application, containers who hold this application currently may be overloaded, leading to performance degradation. Instead of setting a stringent limit on resource usage, a greedy consumption of resources may improve the resilience of the entire datacenter to unanticipated circumstances.

III. The VM Migration Algorithm

In this section, we propose a distributed VM migration algorithm based on Nash Bargaining Solution to alleviate resource under-utilization incurred by failures. To minimize the transmission overhead by migrating, the algorithm responds "lazily" and "locally."

A. The Benefits of VM Migration

As component failures start to occur over time, and as availability of resources in one dimension (*e.g.*, bandwidth) may decrease substantially, it is often the case that such reduced availability makes it harder to fully utilize resources in other dimensions (*e.g.*, storage and CPU). For example, with BCube [8], the failure of an aggregator in one container will cause a decrease of its upload bandwidth; then, the reduced bandwidth may degrade the access to this container's storage and CPU computing resources.

Taking full advantage of the virtualization technology, we may *migrate* VMs to an alternative container, to achieve a higher level of resource utilization in the system. The following example illustrates potential benefits by such migrating.

Consider a datacenter which has two containers and two applications. Each of the applications is provided by a corresponding VM: VM1 and VM2, respectively. The available resources of containers at the beginning and resources required to handle one request in a VM are summarized in Table I.

TABLE I
Available Resources for Each Container and Required Resources for Each VM

Resources	Container 1/2	VM1	VM2
CPU (MIPS)	6	3	1
Storage Space (GB)	6	3	3
Bandwidth (Mbps)	6	1	3

Initially VMs are placed as shown in Fig. 1 (a), according to the application placement strategy we discussed in Sec. II. A container's resource utilization ratio is defined as the geometric mean of its utilization ratios in three resource dimensions, storage space, bandwidth and CPU. In this datacenter, the average resource utilization ratio is about 76%.

Then as time goes by, Container 1 loses some computing resources, which has only 3 MIPS left, and Container 2 loses some bandwidth, with 3 Mbps left. If no VM migration is involved, requests for Application 1 and 2 directed to the same container can not be satisfied at the same time, which leads to the average resource utilization ratio decreasing to around 44%. Nevertheless, with VM migration, we can shift VM2 in Container 1 and VM1 in Container 2, as shown in Fig. 1 (b). In such scenario, the VM, which requires more resource in one dimension, is fit into the container that holds more available resource in that dimension. Then, requests directed to the same container for both Application 1 and 2 can be satisfied simultaneously, so that the average resource utilization ratio can be increased to 87%.

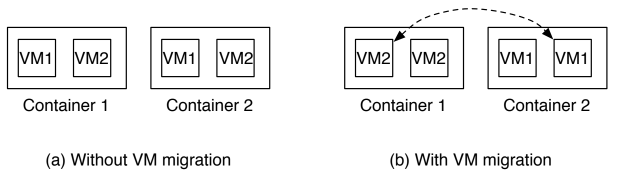

Fig. 1. The different application fit with and without VM migration.

From this example, we can see that by allowing VM migration across different containers, the datacenter is not only able to gain higher resource utilization with limited system-wide resources, but also gives a more satisfactory performance as it can handle more requests at the same time.

B. The VM Migration Algorithm Based on Nash Bargaining Solution

1) System Model and Algorithm Trigger: Before introducing the VM migration algorithm, we first present the system model. Let $\mathcal{N}$ denote the set of containers in a datacenter. For every container $i \in \mathcal{N}$, it is associated with a current available storage space of $C_i(t)$, in Gigabytes; a bandwidth of $U_i(t)$, in Mbps; and a CPU computing capability of $P_i(t)$, capturing the amount of processing power it has, in MIPS. Let the set of application instances provided by the datacenter be denoted by $\mathcal{M}$. For any $k \in \mathcal{M}$, VM$_k$ requires a certain amount of storage space s_k, bandwidth of r_k, and computing resource of cl_k to handle one request. $I_i^k(t)$ is a binary variable indicating if VM$_k$ is provided by container i or not at time t. And $D_i^k(t)$ represents the number of requests for VM$_k$ directed to container i at time t.

Instead of responding to the failures eagerly or regularly, our VM migration algorithm responses *lazily*, *i.e.*, it is only operated when the imbalance of resource utilization ratios in different dimensions alters over a certain threshold. To be precise, at time t, the utilization ratios in storage, bandwidth, and computing resources of each container i can be represented as follows:

$$\begin{aligned} r_i^s(t) &= \frac{\sum_{k\in\mathcal{M}} I_i^k(t) s_k D_i^k(t)}{C_i(t)} \leq 1 \\ r_i^b(t) &= \frac{\sum_{k\in\mathcal{M}} I_i^k(t) r_k D_i^k(t)}{U_i(t)} \leq 1 \\ r_i^c(t) &= \frac{\sum_{k\in\mathcal{M}} I_i^k(t) cl_k D_i^k(t)}{P_i(t)} \leq 1, \end{aligned}$$

where inequalities are the *resource constraints* that resources utilized by all application instances stored on one container can not exceeds its available resources. Let $\sigma_i^r(t)$ be the standard deviation of $r_i^s(t)$, $r_i^b(t)$ and $r_i^c(t)$. When a container i's $\sigma_i^r(t)$ exceeds the pre-defined threshold $\sigma_{threshold}$, this container will "trigger" the VM migration algorithm at time t.

2) The Relaxed Nash Bargaining Solution: To be practical and auto-managed, the VM migration algorithm is done in a peer-to-peer fashion. We envision the existence of a bargaining trading market. VMs are considered as "commodities" in this market. Peers, containers who own the commodities, evaluate the utility of each commodity in the market, then try to bargain with each other and exchange their commodities to increase their own utilities.

Our VM migration algorithm is based on the *Nash Bargaining Solution*, which is a Pareto efficient solution to a two-player bargaining game. In such a game, two individuals have the opportunity to collaborate for mutual benefit in more than one way. By assuming that I) two individuals are highly rational, II) each can accurately compare its desire for various things, III) they are equal in bargaining skill, and IV) each has full knowledge of the tastes and preferences of the other, Nash proposed a solution which should satisfy certain axioms [9]. Let u be the utility function for Player 1, and v the utility function for Player 2. Under these conditions, rational agents will choose what is known as the Nash bargaining solution. Namely, they will seek to maximize $|u(x) - u(d)|$ and $|v(y) - v(d)|$, where $u(d)$ and $v(d)$ are the *status quo* utilities (*i.e.*, the utility obtained if one decides not to bargain with the other player).

In the datacenter trading market, there are multiple containers, *i.e.,* possible players. The trigger peer will choose the corresponding player according to the following **Player Selection Principle**: *when a container triggers the VM migration algorithm, it first checks out the dimension in which its resource utilization ratio is the highest; and then chooses the container with the lowest resource utilization ratio in this dimension to bargain with.*

The rationale behind is when $\sigma_i^r(t) > \sigma_{threshold}$, dimension with the highest r_i becomes the "bottleneck" of fully utilizing resources in other dimensions. Application instances require high resource usage in the bottleneck dimension should be moved out to achieve a more balanced resource utilization in all dimensions. Reasonably, the ideal destination of these VMs should be a container with relatively sufficient available resources in the bottleneck dimension.

The Nash Bargaining Solution tries to find the optimal ownerships of the two players' commodities, so that both players' utilities can be maximized. Since a player's utility here is the sum of all self-evaluated VMs' utilities stored in this container, getting the optimal solution may require extensive VM transmissions, which causes heavy transmission overhead. Seeing that the objective here is to find a feasible solution, we relax the Pareto optimality property of Nash Bargaining Solution, making the solution practical and cost effective in datacenters.

Specifically, both players evaluate each VM according to their own information. For player i, the utility of VM$_k$ is computed as:

$$V_i^k(t) = \omega_i^s(t)\frac{s_k D_i^k(t)}{C_i(t)} + \omega_i^b(t)\frac{r_k D_i^k(t)}{U_i(t)} + \omega_i^c(t)\frac{cl_k D_i^k(t)}{P_i(t)}. \quad (1)$$

where $\omega_i^s(t)$, $\omega_i^b(t)$ and $\omega_i^c(t)$ are the weights given to resources in different dimensions according to the player i's current resource usage states, and they are constrained by $\omega_i^s(t) + \omega_i^b(t) + \omega_i^c(t) = 1$. As we discussed before, resource usage in the bottleneck dimension should be given less value. For simplicity, we define the weights to be inversely proportional to the resource utilization ratio in the corresponding dimension, as:

$$\omega_i^s(t) = \frac{\frac{1}{r_i^s(t)}}{sum_{\frac{1}{r}}(t)}, \omega_i^b(t) = \frac{\frac{1}{r_i^b(t)}}{sum_{\frac{1}{r}}(t)}, \omega_i^c(t) = \frac{\frac{1}{r_i^c(t)}}{sum_{\frac{1}{r}}(t)},$$

where $sum_{\frac{1}{r}}(t) = \frac{1}{r_i^s(t)} + \frac{1}{r_i^b(t)} + \frac{1}{r_i^c(t)}$.

The bargain process is based on the players' own evaluation of VMs. Whenever comes a "win-win" situation within resource constraints, *i.e.* the exchange of commodities leads to an increase of both players' utilities:

$$\begin{aligned} u(i) - u(d) &= \sum_{k\in\mathcal{M}_i'} V_i^k(t) - \sum_{k\in\mathcal{M}_i} V_i^k(t) > 0 \text{ AND} \\ v(j) - v(d) &= \sum_{k\in\mathcal{M}_j'} V_j^k(t) - \sum_{k\in\mathcal{M}_j} V_j^k(t) > 0, \end{aligned}$$

the trade is done. $\mathcal{M}_i$ and $\mathcal{M}_j$ are the owned VM sets of player i and j before commodities exchange, and $\mathcal{M}_i'$ and $\mathcal{M}_j'$ are the owned VM sets of player i and j after commodities exchange. Once the standard deviation σ^r of the trigger peer falls below the threshold, the market is closed, which means the VM migration algorithm is terminated. The VM migration algorithm is summarized as **Algorithm 1**.

3) Implementation Issues: Towards a practical implementation, we briefly discuss the implementation concerns here.

▷ *Lazy Response:* Though migrating VMs have potential benefits to improve resource utilization ratio, it does not come without substantial upfront costs of bandwidth. An example orchestration of live VM and storage migration on the testbed through HARMONY shows the transaction throughput drops around 11.9% during VM migration [4]. Application performance may be affected severely by these live migrations, which requires avoiding the migration activities as much as possible. As a consequence, our VM migration algorithm responses lazily, *i.e.*, it only starts when triggered and terminates immediately when the utilization ratio is tolerable.

Algorithm 1 *The VM Migration Algorithm.*

```
1: Wait until a container $i$'s $\sigma_i^r(t) > \sigma_{threshold}$. Container $i$ is the trigger peer.
2: According to the Player Selection Principle, the trigger peer chooses another player $j$ in the Bargaining game.
3: while $\sigma_i^r(t) > \sigma_{threshold}$ do
4:     Player $i$ and $j$ compute the utilities of VM$_k, \forall k \in \mathcal{M}$ according to Eqn. (1), and bargain with each other.
5:     if $u(i) - u(d) > 0$ and $v(j) - v(d) > 0$ when commodities exchange of $k_1$ and $k_2$ happens, then
6:         Player $i$ and $j$ trade with commodities $k_1$ and $k_2$.
7:     end if
8: end while
```

▷ *Prioritization to Traffic:* To further reduce the affection of migration traffic to the productive work, prioritization can be adopted. The VM migration traffic gets lower priority. They can be deferring to other tasks, *e.g.*, background transfer of TCP Nice or be restricted to opportunistic usage, *i.e.*, VM traffic occurs only when the resource is idle [6].

▷ *Algorithm Termination:* In reality, it is possible that not a single win-win situation happens between two players. It is also possible that through several exchanging between two players, the standard deviation of resource utilization ratios of the trigger peer is still over the threshold. To ensure the stability, a limit on the number of bargain round can be added to the algorithm. When the number of bargain round is achieved, the algorithm is forced to be terminated. Then the trigger peer chooses the second proper player according to the Player Selection Principle in the next execution period.

IV. Experimental Evaluation

We dedicate this section to investigations of how the proposed VM migration algorithm performs in practical scenarios. The results validate that our VM migration algorithm increases resource utilization successfully in container-based datacenters.

The evaluation of the proposed VM migration algorithm is based on its implementation in an event-driven simulator using C++. We simulate in a system with 20 containers and 100 VMs. Every container has the same 2000 storage, 2000 Mbps bandwidth and 2000 MIPS CPU computing resource capacities. For every container, resource in each dimension fail at times according to a Poisson distribution, the mean of which follows a normal distribution of $N(30, 10)$ among containers. The amount of resource failures of each container at every time follows a uniform distribution of $U(1, 50)$. Different resources required by individual VMs are generated according to a normal distribution of $N(25, 10)$, with restriction of range between 1 and 100. We set $\sigma_{threshold}$, the threshold of standard deviation of resource utilization ratios to trigger the VM migration algorithm, to be 0.1.

Since our objective is to increase resource utilization ratio, the main performance metric in this simulation is the standard deviation of resource utilization ratios, which reflects the

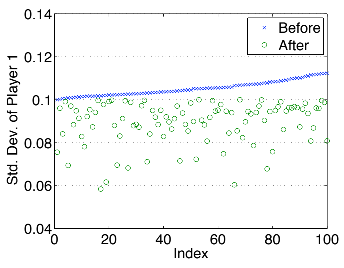

Fig. 2. Standard deviation of resource utilization ratios of Player 1 (trigger peer).

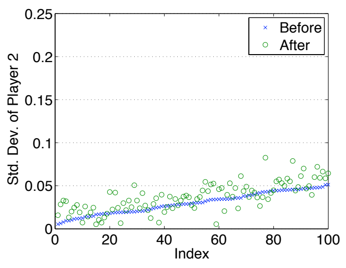

Fig. 3. Standard deviation of resource utilization ratios of Player 2.

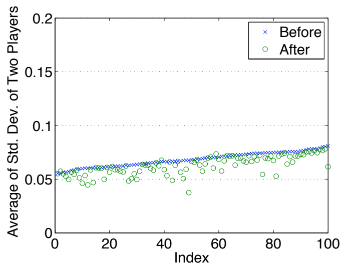

Fig. 4. The average of standard deviation of resource utilization ratios of two players.

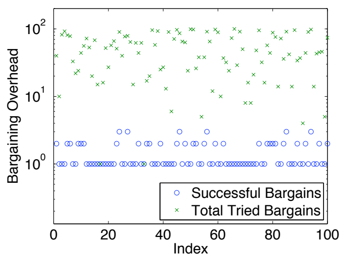

Fig. 5. The communication and transmission overhead of bargain.

balanced level of resource usage in different dimensions. Besides, we also show the bargaining overhead from the implementation point of view. We collect 100 samples by running the VM migration algorithm.

A. *Overall Performance of the VM Migration Algorithm*

First, we would like to show the overall performance of the VM migration algorithm. From Fig. 2, we can observe that the standard deviation of resource utilization ratios of the trigger peer is successfully decreased below the threshold after running the algorithm. This shows the effectiveness of the VM migration algorithm to lessen the pressure of resource under-utilization put by imbalance usage among different dimensions.

It is critical to point out that there are some cases that the standard deviation of Player 2 appears higher than without running the VM migration algorithm, which is indicated in Fig. 3. The reason is that by improving the trigger peer's performance, sometimes it is required to sacrifice the other container's performance a little bit, so that the overall performance is improved. As shown in Fig. 4, the average of the standard deviation of resource utilization ratios is improved evidently.

B. *Bargaining Overhead*

To investigate the communication and transmission overhead incurred by bargain, we show how many tried bargaining rounds and actual trades are needed to lower the resource utilization ratios of all the containers to a tolerable balanced situation. As shown in Fig. 5, in most cases the algorithm terminates within dozens of bargain rounds, which implies the communication overhead is acceptable. It also reveals that in 80% of all the samples only 1 trade is needed. While sometimes it may require $2-3$ successful trades between the two players, at most this number will not exceed 5, which suggests that the transmission overhead is pretty satisfying.

V. Concluding Remarks and Further Work

Our focus in this paper is to fully utilize resources in container-based datacenters via VM migration. Before describing the VM migration algorithm, we propose a new application placement strategy based on Buffet principle, which advocates to use the resources aggressively. In contrast to having a centralized controller, we present the VM migration algorithm in a peer-to-peer fashion regulated by bargaining behaviors between containers. Relying on the inherent self-organized manner of bargaining games, the proposed VM migration algorithm is effective, with low management complexity. Resource utilization ratio can be increased locally by bargaining behaviors between two peers to increase their own utilities. As shown by experiment results, the proposed VM migration algorithm enjoys substantial improvement with respect to the resource utilization ratio.

We believe that this work represents the first step towards raising resource utilization ratios in container-based datacenters. There are many other interesting topics worth discussing. For example, what is the best routing maps for VMs to "maximize" resource utilization ratios; how much is the performance gap between our bargaining solution and the optimal one, and so on. For the proposed VM migration algorithm, adopting trading strategies in some multiplayer bargaining games rather than the two-player bargaining game might obtain higher resource utilization. We defer these investigations to our future work.

References

[1] [Online]. Available: http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci1306761,00.html

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[3] M. Korupolu, A. Singh, and B. Bamba, "Coupled Placement in Modern Data Centers," in *Proc. IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, 2009, pp. 1–12.

[4] A. Singh, M. Korupolu, and D. Mohapatra, "Server-Storage Virtualization: Integration and Load Balancing in Data Centers," in *Proc. of ACM/IEEE Conference on Supercomputing (SC)*, 2008, pp. 1–12.

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 2009.

[6] R. R. Ratul Mahajan, Jitendra Padhye and B. Zill, "Eat All You Can in an All-You-Can-Eat Buffet: A Case for Aggressive Resource usage," in *Proc. 7th ACM Workshop on Hot Topics in Networks (Hotnets)*, 2008.

[7] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.

[9] J. Nash, "The Bargaining Problem," *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.