Network-aware Virtual Machine Consolidation for Large Data Centers

by

Kakadia Dharmesh, Nandish Kopri, Vasudeva Varma

in

The 3rd International Workshop on Network-aware Data Management

Denver, Colorado.

Report No: IIIT/TR/2013/-1

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2013

Network-aware Virtual Machine Consolidation for Large Data Centers

Dharmesh Kakadia
International Institute of
Information Technology
Hyderabad, India
dharmesh.kakadia
@research.iiit.ac.in

Nandish Kopri
Unisys Corp.
Bangalore, India
nandish.kopri
@in.unisys.com

Vasudeva Varma
International Institute of
Information Technology
Hyderabad, India
vv@iiit.ac.in

ABSTRACT

Resource management in modern data centers has become a challenging task due to the tremendous growth of data centers. In large virtual data centers, performance of applications is highly dependent on the communication bandwidth available among virtual machines. Traditional algorithms either do not consider network I/O details of the applications or are computationally intensive. We address the problem of identifying the virtual machine clusters based on the network traffic and placing them intelligently in order to improve the application performance and optimize the network usage in large data center. We propose a greedy consolidation algorithm that ensures the number of migrations is small and the placement decisions are fast, which makes it practical for large data centers. We evaluated our approach on real world traces from private and academic data centers, using simulation and compared the existing algorithms on various parameters like scheduling time, performance improvement and number of migrations. We observed a ∼70% savings of the interconnect bandwidth and overall ∼60% improvements in the applications performances. Also, these improvements were produced within a fraction of scheduling time and number of migrations.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Network operating systems; D.4 [**Operating Systems**]: Scheduling

General Terms

Algorithms

Keywords

network-aware placement, virtual machine scheduling

NDM'13 November 17, 2013, Denver CO, USA

ACM 978-1-4503-2522-6/13/11 ...$15.00.
http://dx.doi.org/10.1145/2534695.2534702 .

1. INTRODUCTION

Modern data centers are extremely large, complex and host applications with a wide range of communication requirements that make resource management a challenging task. In a cloud environment where pay-as-you-go model and on-demand computing are encouraged, VMs from the same applications can be placed across data center and network performance among VMs cannot be guaranteed. This can adversely affect performance of applications, particularly with large east-west traffic and projects that aim to build cluster computing systems on cloud. In large virtual dynamic environment where clusters are scaled up and down frequently, performance of such modern distributed application can face performance issues that are acknowledged by studies [18], showing wide variations in available bandwidth between VMs.

Using current placement algorithms, which do not take application network requirements and dynamism into account, VMs from the same cluster/application can be deployed in a non-optimal way and network can be a serious performance bottleneck. The improper VM placement not only affects I/O intensive applications but also to other non-I/O intensive applications that will see large I/O waiting time due to shared interconnect and oversubscription in network equipment common in the data centers. This network unaware placement also results into the internal network bandwidth being wasted. Thus internal network saturation not only harms the data-intensive applications but also all the other applications running in the data center.

In this paper, we propose a VM placement algorithm whose primary objective is to consolidate VMs using network awareness. Our specific contributions are,

- VMCluster formation algorithm : to cluster the VMs, based on their traffic exchange patterns.
- VMCluster placement algorithm : consolidation algorithm for placing the VMClusters such that, the application performance increases and the internal datacenter traffic is localized as much as possible.

We use network traffic history to group VMs in virtual clusters and use greedy algorithm to consolidate VMs so as to localize traffic from the same group, which results in better application performance and saving in internal bandwidth. The pervious approaches have either been too computationally intensive or application specific or have considered only CPU or suggest too many migrations, limiting their applicability in practice.

The reminder of the paper is organized as follows. We summarize related work in section 2, section 3 describes our system model of the data center. Section 4 explains our proposed algorithms and section 5 reports our experimental results. Finally, we conclude the paper with our findings in section 6.

2. RELATED WORK

Resource allocation and management in data centers has received a lot of attention in recent years due to the emergence of cloud computing. This has resulted in a significant research effort from the community and much advancements in this area.

Researchers have realized the limitations of current data center network architectures and have spawned parallel efforts to redesign data center architectures and topologies to accommodate requirements of the modern data center traffic. PortLand [11] and VL2 [6] are three-tier architectures with Clos [4] topologies. BCube [7] is a multi-level architecture in which the servers forward traffic on behalf of other servers and they are considered to be first class citizens of the network architecture. We approach the problem from the other end, making better use of the available interconnect bandwidth. Our goal is to come up with the VM placement algorithm that places the high-communicating VMs near each other, to help scaling the interconnect.

Most of the proposed algorithms for VM placement and consolidation focus on computation resource utilization. Considering the computing resource as utility, the VM placement problem can be modeled as various constraint satisfaction problems [12] [15]. Kusic et al.[9] have used Limited Lookahead Control (LLC) to solve the continuous consolidation problem as a sequential optimization. They require application specific learning. The complexity of the proposed model is high and can take even 30 minutes to make scheduling decision for even a very small number of nodes.

A wide range of heuristics based algorithms has been proposed for VM placement. Beloglazov et al. [2] proposed heuristic based algorithms for VM consolidation to minimize the energy utilization. Tang et al. proposed a task scheduling algorithm which is thermal and energy-aware [16]. Affinity-aware migration was proposed in [14] to minimize the communication overhead. They use a bartering based algorithm to negotiate VM relocations based on the local information. This algorithm can take longer to execute because of the negotiation. Use of local information can result in non-optimal placement of VMs. In [8] the authors consider demand, availability and communication constraints and transformed the VM placement problem into a graph coloring problem. They also show that each of these constrained subproblems are NP-hard.

There are some prior approaches to the VM placement problem, which model it as some form of optimization problem [10][8]. Tantawi et al. [17] consider the problem of placing virtual machine clusters on to physical machines in data center. They use importance sampling based approach to solve the placement problem. They note that the implementation of this approach is inefficient and relies on a sampling process to incorporate communication needs and other constraints of requests within the placement algorithm. In [19] authors consider the problem of virtual machine placement in virtualized environments with multiple objectives. They use a genetic algorithm with fuzzy evaluation to search the solution space and combine the constraints. While the above approaches can accommodate the network requirements, they are computationally intensive and difficult to scale for real world data centers. Also, they do not consider the performance benefits of a consolidation and can lead to excessive migration for very little performance improvement that can degrade overall performance.

There has been some work to localize the network traffic using application-awareness. While these approaches are useful in specialized applications such as Hadoop, their applicability is limited to particular frameworks. Also if these nodes are running in virtualized environments, which is increasingly the case, the framework will not be effective in localizing the traffic because of no visibility into virtual infrastructure. Modern data centers run a variety of applications with a variety of network requirements and it may not be possible for the provider to know the traffic patterns beforehand. Our work does not assume any application-awareness and thus is applicable to all data center workloads. Also, as these frameworks have very specialized traffic patterns, our algorithm will benefit them automatically.

To summarize, previous work has focused on optimizing resources like CPU or energy or thermal awareness. The proposed approaches for network-aware consolidation has limited applicability in real data centers as they require intensive computations and do not consider the benefits against the migrations.

3. SYSTEM MODEL

We consider typical virtualized data center model consisting of a set of physical machines (PMs) connected through hierarchical network topology. Applications can have several tasks, distributed across many VMs and can scale up and down resulting in dynamic environment where VMs are dynamically spawned and terminated and their traffic pattern changes over the time.

3.1 VMCluster

We introduce the concept of VMCluster to represent a group of VMs that has large communication cost over time period T. Our approach does not require knowledge about application running inside VM and uses metrics readily available in hypervisor and network devices. This is important for large data center operators such as public cloud providers, which has no knowledge about customer applications running inside VMs and their communication patterns. Membership of VMs to VMClusters is dynamic and only property that needs to hold is data exchange between the VMs for T time period.

3.2 Communication Cost Model

For identifying the VMClusters , we define the following as our communication cost,

$$c_{ij} = AccessRate_{ij} \times Delay_{ij} \tag{1}$$

Where $AccessRate_{ij}$ is rate of data exchange between VM_i and VM_j and $Delay_{ij}$ is the communication delay between them. We used $AccessRate$ as We measured $Delay$ as ping delay between corresponding VMs. c_{ij} is maintained over time period T in moving window fashion and mean is taken as the value.

The intuition behind defining cost as Eq.1 is to ensure that it captures not only network delay but also how frequent

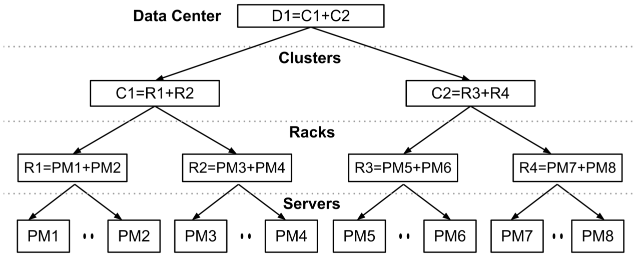

Figure 1: Example cost tree for a data center with 2 clusters each containing 2 racks

data communication is between VMs. This is important, as we do not want to optimize placement of VMs which are very far but exchange very little traffic and similarly for VMs which have a very small delay, placement optimization of which will not result in significant performance benefits.

3.3 Cost Tree

We model data center as hierarchical tree structure called *cost tree*. The root node in the cost tree represents the data center and leaves represent physical servers (PM). The levels below represent clusters, server racks, etc., in superset relation as shown in Figure 1. The value of a node represents communication cost of the traffic flowing through the node. The value at parent node is sum of values of its child nodes. The leaf nodes have values representing the access cost to that server. Note that tree representation is logical and not tied to any specific network topology.

We note that with the advent of the Software Defined Network (SDN) based networking in large data centers, the collection of network information is possible providing a global view of network. Note that SDN provides one way of collecting network information and we do not require a data center to be operating SDN. We only require traffic information to be available.

4. PROPOSED ALGORITHMS

We propose algorithms to form the VMClusters, selection of VM from VMCluster for migration and placement algorithm for consolidating using cost tree.

4.1 VMCluster Formulation

Traffic information between VMs is maintained in the form of $AccessMatrix$,

$$AccessMatrix_{n\times n} = \begin{bmatrix} 0 & c_{12} & c_{13} & \cdots & c_{1n} \\ c_{21} & 0 & c_{23} & \cdots & c_{2n} \\ c_{31} & c_{32} & 0 & \cdots & c_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \cdots & 0 \end{bmatrix} \quad (2)$$

Each element c_{ij} in the access matrix represents communication cost between VM_i and VM_j. It may seem that $AccessMatrix$ size ($n \times n$) is prohibitively large, which is not the case. Note that we only collect c_{ij} for VMs which exchange traffic between them, making $AccessMatrix$ extremely sparse and disjoint and it is maintained in Compressed Sparse Row (CSR) format which makes it very practical. Further optimization is possible by maintaining only top-K candidates. We assume communication cost within VM is negligible and costs are symmetric ($\forall i, j : c_{ij} = c_{ji}$). The following is the VMCluster formulation algorithm, used to identify virtual clusters.

Algorithm 1 VMCluster Formation Algorithm

```
for each row $A_i \in$ AccessMatrix do
    if maxElement($A_i$) > (1 + $opt\_threshold$) * $avg\_comm\_cost$ then
        form a new $VMCluster_i$ from non-zero elements of $A_i$
    end if
end for
```

Each row in $AccessMatrix$ represents communication cost incurred by the VM. The algorithm considers maximum of theses costs (max element of the row) and compares it with *avg_comm_cost*. *avg_comm_cost* is average communication cost (calculated as average of $AccessMatrix$ elements c_{ij}) of data center that we use as baseline to decide which communication costs are considered significant. *opt_threshold* $\in [0, 1]$, is a parameter which controls the number of VMClusters to be considered by the algorithm. If the *opt_threshold* is near one, all the VMClusters with above average communication cost will be considered and as the value grows the number of VMClusters will be decreasing.

4.2 Consolidation Algorithm

The consolidation algorithm is responsible for placement of VMClusters to localize the traffic. There are two main steps of the consolidation algorithm.

VM Selection Choosing which VM to migrate is very important as it can affect the optimization achieved and the migration time. We want to choose the VM in such a way that the chosen VM has large communication cost to the rest of the VMs or in other words communication cost-wise farthest from the rest of the VMs in its cluster. The average distance from each VM to all other VMs in VMCluster is calculated. The VM with largest average distance is chosen which ensures that largest communication cost optimized at each step.

$$VMtoMigrate_i = \arg\max_{VM_i} \sum_{j=1}^{|VMCluster_j|} c_{ij} \quad (3)$$

Placement Algorithm The placement decision for migration destination is hierarchical using cost tree. Each level in the cost tree represents a decision about the placement. While selecting destination for the VM, the cost tree is traversed bottom-up candidates for the migration destination using following recursive definition ($CandidateSet$ for leaf level is $\emptyset$) are tried at each level,

$$\begin{aligned} &CandidateSet_i(VMCluster_j) = \{c \mid where \\ &\text{c and } VMCluster_j \text{ has a common ancestor at level } i \text{ in cost-tree}\} \\ &- CandidateSet_{i+1}(VMCluster_j) \end{aligned} \quad (4)$$

For example, in Figure 1, if we have a

$$VMCluster = \{PM3, PM4, PM6, PM8\}$$

Algorithm 2 The Consolidation Algorithm

```
for $VMCluster_j \in$ VMClusters do
    $VMtoMigrate$ according to Eq. 3
    for $i$ from leaf to root do
        $CandidateSet_i(VMCluster_j - VMtoMigrate)$ according to eq. 4
        for $PM \in candidateSet_i$ do
            if UtilAfterMigration(PM,$VMtoMigrate$) <overload_threshold AND PerfGain(PM,$VMtoMigrate$) $>$ significance_threshold then
                migrate VM to PM
                continue to next VMCluster
            end if
        end for
    end for
end for
```

then the $CandidateSets$ will be
$CandidateSet_3(VMCluster) = \{PM5, PM7\}$
$CandidateSet_2(VMCluster) = \{PM1, PM3\}$
Here the members of the $CandidateSet_3$ will be tried first in the order of their cost. If $c5$ and $c7$ do not have sufficient resources for the incoming VM, members of $CandidateSet_2$ are tried next, and these will be continued towards root.

Before choosing migration destination, we consider resource utilization on candidate PM after migration. To determine if the candidate PM has sufficient resources (compute and memory) available by calculating utilization of PM if the VM is migrated, function $UtilAfterMigration$ is used. The resource availability decision of PM is controlled by parameter *overload_threshold*. If PM is estimated to have required resources, it is chosen as migration destination. Otherwise other candidates in the *CandidateSet* are tried. If VM cannot be allocated to any of the PM in $CandidateSet_i$, PMs in $CandidateSet_{i-1}$ is tried. This continues recursively towards root. Candidates within same *CandidateSet* are considered in the order of their cost, the higher cost node is tried first. The algorithm also estimates the benefit that will be achieved by migrating VM to suggested PM before actual migration, in terms of percentage improvement in communication cost after migration as follows,

$$PerfGain = \sum_{j=1}^{|VMCluster|} \frac{c_{ij} - c'_{ij}}{c_{ij}} \tag{5}$$

Where c_{ij} is the original communication cost and c'_{ij} is the cost calculated after migration. c'_{ij} calculation is done using equation 1, but with the delay updated to reflect the migration. *significance_threshold* $\in [0, 1]$ controls the aggressiveness of the algorithm. If $PerformanceImprovement$ is above administrator defined *significance_threshold*, then the migration is carried out, otherwise no migration is performed In both the cases, next VMCluster is tried to for performance improvement.

5. EXPERIMENTAL EVALUATION

In this section we provide experimental results to illustrate performance of our algorithm.We compared our approach to traditional placement approaches like Vespa[15] and previous network-aware algorithm like Piao's approach[13]. We extended NetworkCloudSim[5] to support SDN functionalities and used Floodlight [1] as our SDN controller. Counters available in Floodlight were used to collect the required network information and T is set to 3s. The server properties were not dictated in the trace and we assumed them to have typical data center server specification (HP ProLiant ML110 G5 (1 x [Xeon 3075 2660 MHz, 2 cores]), 4GB) connected through 1G using HP ProCurve switches in three-tier. We used traces from three real world data centers, two from universities (uni1, uni2) and one from private data center (prv1), to simulate the traffic to evaluate our approach [3]. The statistics of the traces used for simulation are shown in Table 1.

Property	Uni1	Uni2	Prv1
Number of Short non-I/O-intensive jobs	513	3637	3152
Number of Short I/O-intensive jobs	223	1834	1798
Number of Medium non-I/O-intensive jobs	135	628	173
Number of Medium I/O-intensive jobs	186	864	231
Number of Long non-I/O-intensive jobs	112	319	59
Number of Long I/O-intensive jobs	160	418	358
Number of Servers	500	1093	1088
Number of Devices	22	36	96
Over Subscription	2:1	47:1	8:3
Traffic report delay	10 μs		
Trace duration	12 hours over multiple days		

Table 1: Trace statistics

Appendix A shows the summary of the results achieved across all the three traces. We now discuss detailed results and various properties of algorithms.

5.1 Performance Improvement

We have categorized the jobs into 6 different types based on their I/O requirement and running time for finer evaluation of performance on various classes of jobs. We measured percentage improvement in runtime of jobs of various classes of jobs. As shown in Figure 2, I/O intensive jobs are benefited the most but short jobs also have significant improvements. It is important to note that although short jobs have generally low n/w requirements, number of such jobs are typically much larger, making them important for overall improvement of performance.

5.2 Traffic Localization

The goal of consolidation algorithm is to localize traffic for better interconnect scaling and improving application performance. The traffic localization can be measured by the traffic reduction in the higher-level network switches and increase in the lower-level network switches. The effect of consolidation on the core and Top of Rack (ToR) switch traffics in all the tree trace scenarios is depicted in the figure 3. In the figure, the traffic after consolidation is shown for switches are compared for our approach and Piao's approach (Vespa isn't n/w aware). As seen in graph, $\sim$ 60% increase

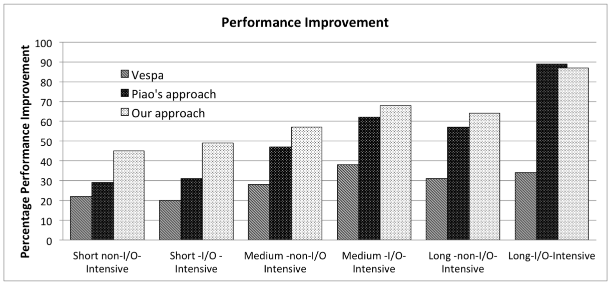

Figure 2: Performance Improvement for various class of traffics

in ToR traffic and ∼ 70% reduction in core traffic confirms effective localization using our approach, while Piao's approach resulted in nearly ∼ 30% increase and ∼ 37% decrease correspondingly. This result also confirms that the network aware placement can mitigate network scalability issues.

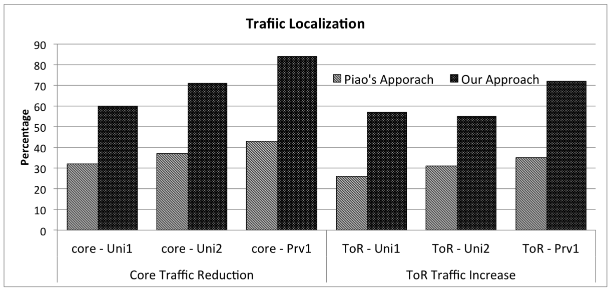

Figure 3: Localization for core and ToR traffic

5.3 Complexity

Complexity of scheduling is an important measure in practical settings. We report decision making time, stability (variance) and number of migration for all the three approaches on various measures in Table 2. Our algorithm performed significantly better than both of the approaches. Also the worst-case time and variance in scheduling are important factors for building practical systems and affect interdependent components. Our approach took 558 ms to in worst case and only 60 ms variance across traces that shows the stability of our approach. Vespa on the other hand, took 973 ms in worst case, with 130 ms variance. Piao's approach is clearly showing the effect of not keeping time complexity in mind and in worst case took more than 1.3 secs. The advantage of complexity is thus confirmed.

Figure 4 compares number of migrations and variance as it can affect application performance and overall stability. Small variance in time shows the predictable behavior of our algorithm compared to other two algorithms. The lack of consideration for number of migration is clearly visible in both other algorithms and our approach required much lesser number of migrations. This also confirms that large number of migrations is not required for significant performance improvement, opposite to what is suggested by most optimization based approaches.

Measure	Trace	Vespa	Piao's approach	Our approach
Avg. scheduling Time (ms)	Uni1	504.64	677.66	217.36
	Uni2	784.24	1197.54	376.84
	Prv1	718.33	1076.33	324.66
Worst-case scheduling Time (ms)	Uni1	846	1087	502
	Uni2	973	1316	558
	Prv1	894	1278	539
Variance in scheduling Time	Uni1	179.64	146.76	70.68
	Uni2	234.24	246.64	98.74
	Prv1	214.33	216.66	89.99
Number of Migrations	Uni1	154	213	56
	Uni2	547	1145	441
	Prv1	423	597	96

Table 2: Complexity – Time, Variance and Migrations

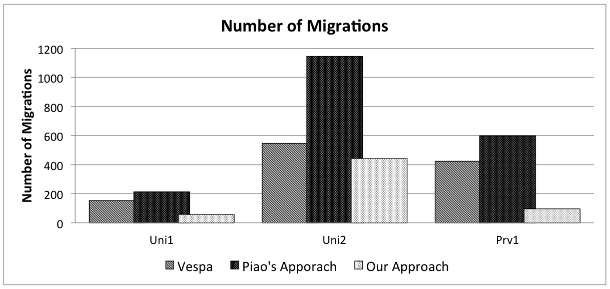

Figure 4: Number of Migrations by different approaches

5.4 Dependence on Parameters

Figure 5 shows how the performance improvement depends on the parameter *significance_threshold*. As noted earlier the parameter controls the aggressiveness of the algorithm. As seen in the graph, the initial slow decrease in the performance (10% decrease when *significance_threshold* varies from 0.05 to 0.40 in both university traces and 20% in the private trace) indicates that the benefits of the algorithm saturates around 0.60–0.70. Also much higher values of *significance_threshold* will force the algorithm to consider most of the possible improvements as insignificant and very little performance improvements can be realized.

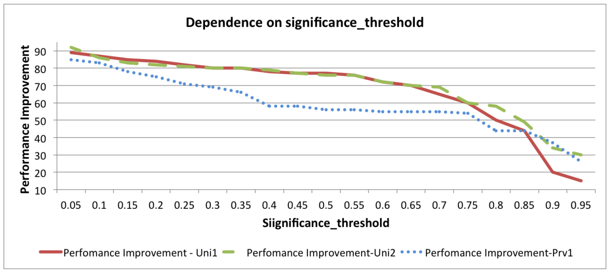

Figure 5: Dependence of performance improvement on *significance_threshold*

The number of VMClusters to be considered by the algorithm is controlled by *opt_threshold*. Figure 6 shows how the variation in number of clusters formed with various values of *opt_threshold* for all three traces. The step-wise decrement in VMClusters shows that performance of our algorithm is stable against small variations in the *opt_threshold*, but number of VMClusters decrease with increase the in *opt_threshold*. Number of VMClusters starts saturating around value 0.6-0.7 and further increase in *opt_threshold* does not affect VMClusters due to the inherent traffic patterns of the workload.

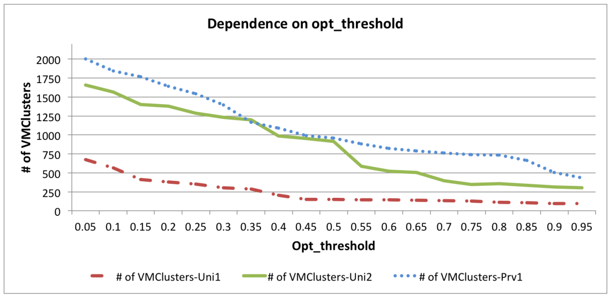

Figure 6: Dependence of number of VMClusters on *opt_threshold*

From both of the graphs, it is clear that the algorithm is stable against small variation in the parameters, but allows the operator making the tradeoffs between performance improvements and the complexity of the placement.

6. CONCLUSION

In this paper, we proposed algorithms to identify virtual clusters and consolidating them in order to improve application performance and localize traffic. We analyzed performance improvement, number of migrations and scheduling time of our algorithm and existing approaches on real world traces, with variety of jobs. Our scheduling decisions achieved better performance and were an order of magnitude faster and also required a fraction of migrations compared to others, making it practical for large data centers. Also we believe that the identified VMClusters using VMFormation algorithm can be useful outside of the scheduling decision too, for example in automatically replicating or migrating an entire service, hosted on group of VMs and would like to explore it further.

7. REFERENCES

[1] Floodlight openflow controller. http://floodlight.openflowhub.org.

[2] A. Beloglazov and R. Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 577–578, Washington, DC, USA, 2010. IEEE Computer Society.

[3] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 267–280, New York, NY, USA, 2010. ACM.

[4] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2003.

[5] S. K. Garg and R. Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. In *Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing*, UCC '11, pages 105–113, Washington, DC, USA, 2011. IEEE Computer Society.

[6] 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.*, 39(4):51–62, Aug. 2009.

[7] 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. *SIGCOMM Comput. Commun. Rev.*, 39(4):63–74, Aug. 2009.

[8] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible. Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 72–79. IEEE, 2011.

[9] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society.

[10] X. Meng, V. Pappas, and L. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, Mar. 2010.

[11] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 39–50, New York, NY, USA, 2009. ACM.

[12] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud. Autonomic virtual resource management for service hosting platforms. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[13] J. Piao and J. Yan. A network-aware virtual machine placement and migration approach in cloud computing. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 87–92. IEEE, 2010.

[14] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra. Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 228 –237, sept. 2010.

[15] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 331–340, New York, NY, USA, 2007. ACM.

[16] Q. Tang, S. Gupta, and G. Varsamopoulos. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *Parallel and Distributed Systems, IEEE Transactions on*, 19(11):1458–1472, 2008.

[17] A. N. Tantawi. A scalable algorithm for placement of virtual clusters in large data centers. *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 0:3–10, 2012.

[18] G. Wang and T. S. E. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1163–1171, Piscataway, NJ, USA, 2010. IEEE Press.

[19] J. Xu and J. A. B. Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 179–188, Washington, DC, USA, 2010. IEEE Computer Society.

APPENDIX

A. SUMMARY OF RESULTS

Following table summaries the performance improvements achieved by all three algorithms.

Measure	Job type	Trace	Vespa	Piao's approach	Our approach
Percentage Improvement in Avg. Completion Time	Short non-I/O-Intensive	Uni1	15	31	41
		Uni2	26	36	51
		Prv1	23	33	45
	Short I/O-Intensive	Uni1	17	56	43
		Uni2	23	50	59
		Prv1	25	58	55
	Medium-non-I/O-Intensive	Uni1	28	47	58
		Uni2	31	52	62
		Prv1	27	45	68
	Medium-I/O-Intensive	Uni1	32	59	69
		Uni2	37	69	76
		Prv1	29	74	73
	Long-non-I/O-Intensive	Uni1	28	65	68
		Uni2	29	67	81
		Prv1	31	77	79
	Long-I/O-Intensive	Uni1	25	89	83
		Uni2	26	81	86
		Prv1	32	85	88
Variance in Completion Time Improvement	Short non-I/O-Intensive	Uni1	24	25	21
		Uni2	31	37	25
		Prv1	29	35	19
	Short I/O-Intensive	Uni1	27	24	18
		Uni2	36	25	24
		Prv1	37	24	20
	Medium-non-I/O-Intensive	Uni1	26	38	21
		Uni2	34	39	26
		Prv1	29	29	23
	Medium-I/O-Intensive	Uni1	27	15	18
		Uni2	29	27	25
		Prv1	22	23	28
	Long-non-I/O-Intensive	Uni1	26	20	21
		Uni2	33	31	22
		Prv1	19	26	24
	Long-I/O-Intensive	Uni1	23	22	15
		Uni2	31	24	19
		Prv1	28	23	18