2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing

Communication-Aware and Energy-Efficient Scheduling for Parallel Applications in Virtualized Data Centers

Ibrahim Takouna*, Roberto Rojas-Cessa†, Kai Sachs‡, and Christoph Meinel*

*Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
† New Jersey Institute of Technology, Newark,USA
‡ SAP AG, Walldorf, Germany

***Abstract*—In this paper, we propose Peer VMs Aggregation (PVA) to enable dynamic discovery of communication patterns and reschedule VMs based on the determined communication patterns using VM migration. In the implementation, we consider that communication delays occur at the server (i.e., memory-bus) and at the data center network. To evaluate our approach, we modeled a network and a memory subsystem on CloudSim simulator. We then used NAS Parallel Benchmarks, which consists of six different applications as parallel applications. We thoroughly evaluated our proposed approach measuring several assessment metrics including VMs placement, performance degradation, and network utilization of each link. The results of the simulation show that our proposed approach significantly reduces the total amount of traffic in the network where it reduces the average of the network's utilization by 25%.**

***Keywords*-communication-aware; parallel application; energy-aware; virtualization; simulation**

I. INTRODUCTION

The usage of virtualization technology has become common in most modern data centers exploiting its advantages, which include server consolidation to reduce energy consumption, live migration, and agile resource provisioning. Currently, virtualized platforms are used not only for web or commercial transaction applications but also for High Performance Computing (HPC) applications that are mostly run on grids.

HPC applications consist of multiple parallel jobs communicating with each other either via shared memory or shared network. Their communication method depends on the used programming model (i.e., OpenMP and Message Passing Interface (MPI)). Hence, the performance of these applications highly depends on the shared memory bus and shared network. Unlike communication via shared memory, exchanging data among the parallel jobs through a shared network causes a major performance bottleneck [2][3]. Much research has been aimed to achieve energy-efficient Virtual Machines (VMs) placement. Nevertheless, the proposed approaches mainly consider a server's resource including the CPU [4], memory [5], and disk I/O [6].

Unfortunately, scheduling parallel applications based on server resource (e.g., CPU or memory) can be inefficient, causing a very significant degradation in the performance of the parallel applications. Walker compared the performance of NPB MPI using Amazon EC2 and the National Center for Supercomputing Applications (NCSA) [7]. The result showed that the programs CG, FT, IS, IU, and MG had greater than 200% performance degradation. We conjecture this to the random placement of these applications.

Recent research in network-aware VM placement has considered an optimal initial placement of VMs [9]-[10]. However, as the demand of bandwidth and the communication pattern of communicative VMs are known only during the run time, these VMs should be reactively rescheduled to realize an efficient placement. To this end, we propose communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers. Our proposed approach dynamically determines the bandwidth demand and the communication pattern of communicative VMs at run time. Then, it efficiently re-allocates the communicative VMs using migration whenever an inefficient placement is discovered.

The results of our simulations show that our proposed approach significantly reduces the total amount of traffic in the network. They also show a higher VMs performance compared to that of CPU-based placement scheme since the memory latency is very small compared to network latency. Our proposed approach efficiently utilizes the bandwidth of the network and memory-bus of the server where it reduces the average of the network's utilization by 25%. This reduction in traffic can achieve energy savings of around 60% by reducing the number of active switches.

II. METHODOLOGY

Our goal is to achieve an efficient scheduling for communicative VMs and minimize energy consumption by servers and network components. Usually, the approaches that are not communication-aware can cause a random placement of the VMs. This might improve the resource utilization at the server level but worsen the network bandwidth utilization at the switch level and the performance of these VMs. Our approach is to aggregate the communicative VMs exchanging traffic between each others to be placed in the same server. This assists in localizing the traffic and minimizing the communication delay among communicative VMs. To this end, we developed the Peer VMs Aggregation (PVA) approach, which is presented in algorithm 1. Our approach requires VMs to be involved with the Migration Manager (MM) to determine the communication patterns among VMs.

Our approach is built based on the following issues. The jobs in virtualized environments are executed on VMs. A VM can easily determine the VMs that communicate with it, but the VM is not capable of determining the communication pattern of the whole application. Determining the communication pattern is the role of the Migration Manager. Furthermore, we consider that the VMs communicate with one another through a shared network. However, when VMs are scheduled on the same server, they can communicate through a shared memory. Importantly, the servers always have enough free resource capacity for using migration (e.g., 10%-20% of CPU).

Algorithm 1: Peer VMs Aggregation (PVA)

```
1 Initialization
    Each VM$_i$ has communication with other peer VMs
    sending a request with its peer VMs for Migration
    Manager (MM), VmsRequests.add(VM$_i$, peer_VMs)
  While (VmsRequests is not empty) do
2   MM Sorts VMs in descending order based on the
      the number of in/out traffic flows from the VM's
      server to its peer VMs' servers
3   MM Selects the first ranked VM to be migrated
      to the destination server ($Server_2$) of the peer VMs
4   MM Checks the suitability of the destination server
      If is suitable
        MM Migrates the VM from the source ($Server_1$)
          into destination ($Server_2$)
      Else
5         MM Migrates a VM from the source ($Server_1$)
            to the destination ($Server_2$) whereas this VM
            is not one of the VM's peers and is suitable
            to be hosted by the source ($Server_1$)
6         MM Migrates the VM from the
            source ($Server_1$) to destination ($Server_2$)
```

Now, we discuss our approach. The communicative VMs initially send requests to the MM. The request for migration contains a list of peer VMs of the requesting VM. For example, these VMs, VM$_{1\_0}$, VM$_{1\_1}$, VM$_{1\_2}$, and VM$_{1\_3}$, belong to an application where VM$_{i\_j}$ indicates VM j of application i. Thus, the request of VM$_{1\_3}$ contains VM$_{1\_0}$, VM$_{1\_1}$, and VM$_{1\_2}$. Similarly, the request of VM$_{2\_3}$, which belongs to another application, contains VM$_{2\_0}$, VM$_{2\_1}$, and VM$_{2\_2}$. The MM is responsible for four procedures: Sort, Select, Check, and Migrate. Importantly, MM performs these procedures iteratively on servers until it the algorithm converges. The convergence of the algorithm means that all VMs belong to the same application are scheduled on the same server. We next illustrate each procedure.

- **Sort:** MM sorts the VMs, in descending order, based on a number of the in/out traffic flows, after compiling these requests to discover the communication patterns and determining the current placement of these VMs on the servers. As VMs have no knowledge of whether they are hosted on the same server, MM determines that and ignores the requests of VMs scheduled on the same server.
- **Select:** MM selects the top ranked VM to be migrated to the destination server that contains its peer VM. For instance, the selected VM (i.e., VM $_{1\_3}$, which is hosted in the server $Server_1$) has very mutual communication with other VMs (i.e., VM$_{1\_0}$, VM$_{1\_1}$, and VM$_{1\_2}$, which are hosted $Server_2$). Thus, the selected destination server is $Server_2$.
- **Check:** MM checks the suitability of the destination server in terms of CPU, memory, and network, as the resource demand of a VM is become known at the run time.
- **Migrate:** MM directly migrates the selected VM If the destination server is suitable. Otherwise, MM tries to migrate a VM (i.e., VM $_{2\_3}$) from the destination server ($Server_2$) to make room for the selected VM (VM $_{1\_3}$) to be placed in the same server ($Server_2$) of its peer VMs. However, the selected VM (i.e., VM $_{2\_3}$) should be also suitable to be migrated to $Server_2$. In this example, VM $_{1\_3}$ is migrated from ($Server_1$) to ($Server_2$) and VM $_{2\_3}$ is migrated from Server$_1$ to Server$_2$, simultaneously. However, if there is no VM to be moved from the destination server, MM can place the selected VM on a server shared the same edge switch with the server of its peer VMs.

III. Simulation

To evaluate our approach, we implemented the network topology of the simulated data center using CloudSim, as depicted in Figure 1, including three levels of networks. The bandwidth of the link between the edge and the server is 10Gbps. The bandwidth of the link increases as the network level rises toward the core network.

The NAS Parallel Benchmarks (NPB) suite includes a small set of benchmarks with different programming models (e.g., Serial, OpenMP, and MPI) designed to help evaluate the performance of parallel supercomputers. The benchmarks are divided into two groups: kernels benchmarks, consisting of MG, CG, FT, and IS, and pseudo-applications, including LU, BT, and SP. More detail can be found in [15].

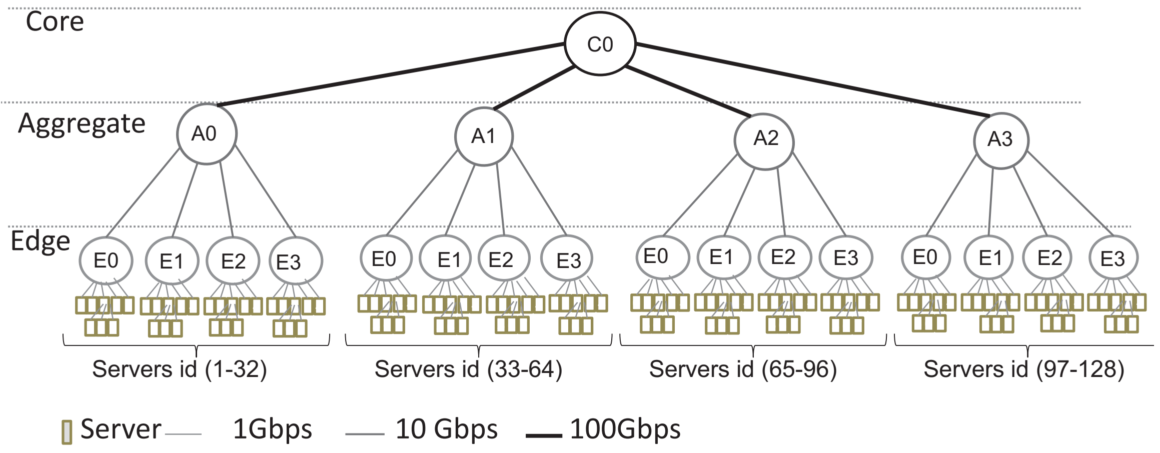
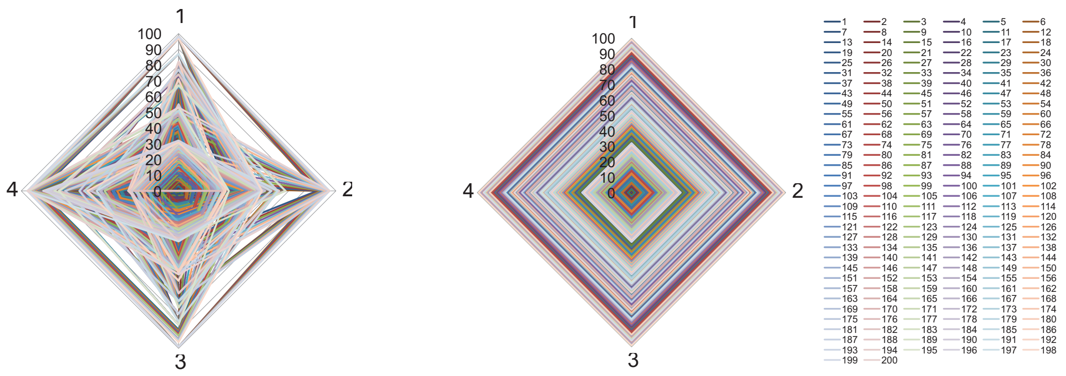


Figure 1: The simulated data center



(a) CPU based approach (b) Our approach (PVA)

Figure 2: The place of VMs into servers: CPU based vs. our approach placement

IV. Experiment and Evaluation

A. Experimental setup

To evaluate our approach, we generated 200 different parallel applications of NPB benchmark, inducing BT, SP, IS, CG, LU, and MG. The parallel application generated four jobs whereas each was executed in a VM. Each job required 1000 MIPS. The ideal execution time of the jobs is 1000 seconds. We also simulated the communication pattern of these applications. The total number of servers in our simulation was 128 servers. The simulated server model is HP ProLiant DL380 G7 (3.07 GHz, Intel X5675 processor with 6 Cores) and its memory-bus bandwidth is 40 GB/s. We considered that the total capacity of the server was 9000 MIPS and each server could host at maximum 8 VMs to keep CPU resource for migration overhead. Seeking simplicity, we assumed that all jobs were similar in their start time and length. We simulated the memory demand of these jobs. Then, we simulated the placement of the 800 VMs (i.e., jobs) based on CPU utilization random fashion. This is considered as the initial placement of the VMs where the CPU based approach is unaware of communication pattern among VMs. We simulated our proposed approach, which can discover the patterns and online rearrange the places of the VMs using migration.

B. Evaluation

We evaluated the efficiency of our approach in comparison to the CPU based placement using VMs placement, network utilization, and performance degradation.

1) VMs placement: Here, Figure 2 shows the distribution of VMs on the servers comparing our approach and CPU based placement approach. In Figure 2, the x-axis represents the VM index of a specific application, and the y-axis represents the server index, which hosts the VM. Figure 2-(b) clearly shows a uniform shape. This means that our approach aggregates all the VMs belongs to an application into the same server. Contrarily, CPU based placement shows a random placement of VMs due to the lack of awareness about the communication among VMs. Thus, our approach, PVA, is capable of achieving a perfect placement after determining the communication pattern of communicative VMs.

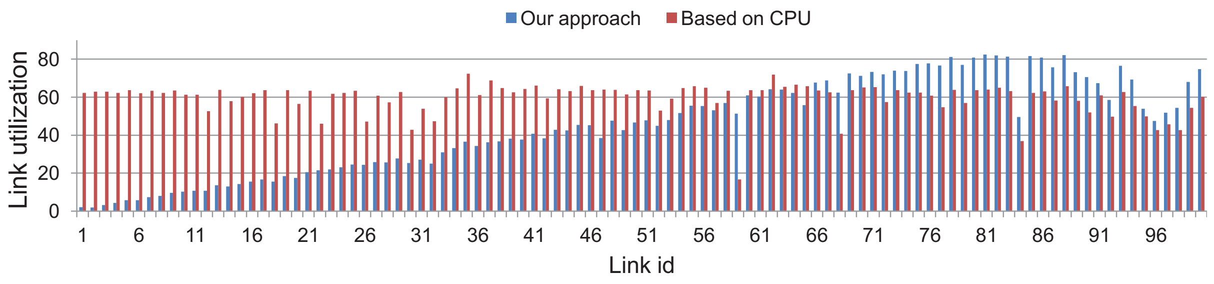


Figure 3: The average utilization of the links of the hosts

2) Network utilization: After illustrating the VMs placement on the servers, we also calculated the average traffic for each link in the network during the transition state of the simulation time. The transition state is the network state between the beginning of the simulation and the convergence of PVA algorithm. Importantly, once PVA algorithm converges, the traffic exchange among VMs becomes zero. This is because PVA algorithm is able to aggregate all VMs of a certain application into one server, as show in Figure 2. Figures 3 and 4 depict the average network utilization of each link in the network. Figure 3 depicts the average network utilization of the links between the servers and the edge switches where as the x-axis represents the link index, which corresponds to the sever index. As we mentioned earlier in Section II, our algorithm only migrates two VMs between two server at the same time starting from server $id = 1$, which is connected with the edge switch E0 in the right branch of the network (i.e., C0-A0-E0), and ending at server $id = 100$. This explains why the average utilization in the links looks like a ramp.

On the other hand, Figure 4 shows the average network utilization of the main links, which connect the switches together. For instance, the link C0-A0 represents the link between the core switch with $id = 0$ and the aggregate switch with $id = 0$ as shown in Figure 1, which depicts the data center network architecture. Importantly, the average utilization of the link (C0-A0) significantly decreases using our approach compared to the CPU based placement approach. This is because the link (C0-A0) aggregates the traffic of the servers (1 to 32) as shown in Figure 1 and our algorithm quickly rearranges the VMs in these servers after some time of the simulation. However, the average utilization of the link (C0-A2) is almost the same using both approaches because our algorithm converges when the VMs placed into servers (65 to 96) almost finish executing their jobs. Our approach reduces the average utilization of the network by 25%. It outperforms the CPU based placement in terms of reducing the network utilization. As our approach moves the communication between VMs from shared network to shared memory by aggregating the communicative VMs into the same server.

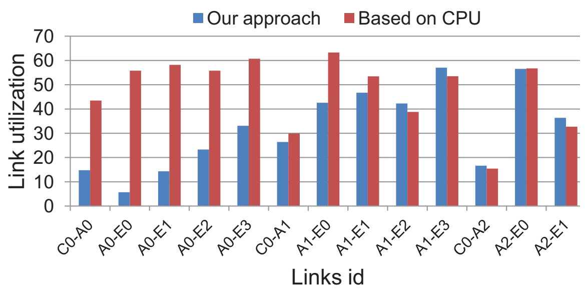


Figure 4: The average utilization of the links between switches

3) Performance degradation: To illustrate the performance degradation that caused by the CPU based placement approach. The performance degradation is computed and compared to the ideal execution time of the jobs. For instance, our approach shows that 18% of the VMs suffer degradation while the CPU based placement shows that 20% of the VMs suffers degradation. Importantly, our approach has no VM with more than 30% performance degradation. This proves that our proposed approach outperforms the CPU based placement, which causes random VMs placement.

V. Related Work

Recently, the problem of VM placement has been extended to include other important aspects of the data center infrastructure such as network traffic and storage facilities. In particular, VMs deployed in cloud infrastructures typically show a network traffic dependency which can be better accommodated by consolidating communicative VMs in the same physical machine [17]. Bansal et al. [9] proposed the online minimum congestion mapping problem. They considered placing a workload consisting of network traffic demands and processing demands on an underlying network graph substrate while trying to minimize the utilization over all links and nodes.

Zhang et al. [10] addressed VM placement to minimize the total traffic in a data center. They used heuristic algorithm based on clustering to consolidate VMs with high inter traffic on the same host. Fan et al. [12] discuss topology-aware deployment for scientific applications in cloud, and map the communication topology of a parallel application to the VM physical topology. Ferreto et al. [13] proposed consolidation approach using linear programming for minimizing the migration problem in virtualized data centers. Gupta et al.[11] address application-aware consolidation of VM instances to physical hosts. They implemented an HPC-aware scheduler on top of OpenStack Compute and also incorporate it in CloudSim simulator.

Meng et al. [18] formulated Linear Programming Problem for traffic-aware VM placement in a data center to optimize network traffic while minimizing the cost of VM migration. Their approach approximately optimizes the placement of VMs on servers based on mutual traffic patterns of VMs. Based on analysis of a run-time feedback from data center's switches and links, Kliazovich et al. [19] presented a data center scheduling methodology combining energy efficiency and network awareness. Their approach achieved the balance between job performance and traffic demand, and energy consumed by the data center. Cruz and Park [16] proposed a solution for communication-sensitive load balancing. Their solution uses run-time communication patterns among processes to balance the load. In [8], the authors attempted to simultaneously balance two different kinds of I/O load, namely, communication and disk I/O.

VI. Conclusions and Future work

Communication-aware placement is very necessary to achieve energy saving and keep application performance. Thus, we presented the Peer VMs Aggregation algorithm for communication-aware placement of parallel application. Our approach aggregates communicative VMs into a server using migration. To evaluate our approach, we implemented a detailed network infrastructure into CloudSim simulator.

The results of the simulation showed that our proposed approach significantly reduced the total traffic in the network. Furthermore, they showed better VMs performance compared to CPU based placement. By using our proposed approach, we efficiently utilize the bandwidth of the network and memory-bus. Furthermore, we achieved better application performance.

As future work, we plan to apply our approach using different number of VMs for each application, which was fixed to four VMs in this paper. Furthermore, we will compare our approach with other scheduling algorithms that concern awareness of communication and topology.

References

[1] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," *Proc. of the ACM Int'l Conf. on Supercomputing*, New York, NY, USA, pp. 125-134, 2006.

[2] W.-C. Feng, J. (Gus) Hurwitz, H. Newman, S. Ravot, R.L. Cottrell, O. Martin, F. Coccetti, C. Jin, X. (David) Wei, and S. Low, Optimizing 10-Gigabit Ethernet for Networks of Workstations,Clusters, and Grids: A Case Study, Proc. 2003 ACM/IEEE Conf.Supercomputing (SC 03), p. 50, 2003.

[3] L. Schaelicke and A.L. Davis, "Design Trade Offs for User-Level I/O Architectures, IEEE Trans. Computers, vol. 55, no. 8, pp. 962-973, Aug. 2006.

[4] M. Harchol-Balter and A.B. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing," ACM Trans. Computer Systems, vol. 15, no. 3, pp. 253-285, 1997.

[5] A. Acharya and S. Setia, "Availability and Utility of Idle Memory in Workstation Clusters," Proc. ACM SIGMETRICS, pp. 35-46, 1999.

[6] X. Qin, "Design and Analysis of a Load Balancing Strategy in Data Grids," Future Generation Computer Systems, vol. 23, no. 1, pp. 132-137, 2007.

[7] E. Walker "Benchmarking Amazon EC2 for high-performance scientific computing." Usenix Login 33.5 (2008): pp. 18-23.

[8] X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, "Communication-Aware Load Balancing for Parallel Applications on Clusters, *IEEE Trans. on Computers*, vol. 59, no. 1, pp. 4252, Jan. 2010.

[9] N. Bansal, K. Lee, V. Nagarajan, and M.Zafer,"Minimum congestion mapping in a cloud," In: *ACM PODC* , pp. 267-276, 2011.

[10] B. Zhang, Z. Qian, W. Huang, X. Li, and S. Lu, "Minimizing Communication Traffic in Data Centers with Power-Aware VM Placement," 2012, pp. 280285.

[11] A. Gupta, L. Kale, D. Milojicic, P. Faraboschi, and S. Balle,"HPC-Aware VM Placement in Infrastructure Clouds," in IEEE Intl. Conf. on Cloud Engineering IC2E, 2013, vol. 13.

[12] P. Fan, Z. Chen, J. Wang, Z. Zheng, and M. R. Lyu, "Topology-Aware Deployment of Scientific Applications in Cloud Computing, 2012 IEEE Fifth International Conference on Cloud Computing pp. 319326, 2012.

[13] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," Future Generation Computer Systems, vol. 27, no. 8, pp. 10271034, 2011.

[14] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, Vol. 24 Issue 13, pp. 1397-1420 , 2011.

[15] I. Takouna, W. Dawoud, and C. Meinel, "Analysis and Simulation of HPC Applications in Virtualized Data Centers," In Proc. of *IEEE Int'l Conf. GreenCom*, pp. 498-507, 20-23 Nov. 2012.

[16] J. Cruz and K. Park, "Towards Communication-Sensitive Load Balancing," Proc. 21st *Intl Conf. Dis. Comp. Sys.*,pp. 731-734, Apr. 2001.

[17] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," In *INFOCOM*, 2011 Proc. IEEE, pp. 71-75, 2011.

[18] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," In: *IEEE INFOCOM* (2010)

[19] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster Computing*, vol. 16, no. 1, pp. 65-75, Sep. 2011.