Efficient Virtual Machine Scheduling-policy for Virtualized Heterogeneous Multicore Systems

Ibrahim Takouna, Wesam Dawoud, and Christoph Meinel
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{ibrahim.takouna, wesam.dawoud, christoph.meinel}@hpi.uni-potsdam.de

Abstract— *Heterogeneous multicore processors could be the future trend of processors' industry due to their performance-power efficiency. In the operating systems domain, A heterogeneity-aware scheduler assigns a thread or an application to an appropriate core to realize this efficiency. Using virtualization technologies enables resource consolidation and achieves effective utilization of resources. Nevertheless, Hypervisors' scheduling-policy is based on the round robin algorithm to ensure fairness among VMs. Emerging heterogeneous system and virtualization increases power savings and enhances resources utilization. This combination needs a new scheduler, which schedules each VM to an appropriate core based on its characteristics. In this paper, we present sources of delay in virtualized environment that could degrade performance of VMs. Then, we investigate the sensitivity of a VM performance to changes in clock frequency. A new scheduling policy was implemented to alleviate sources of delay and to be aware of system's heterogeneity. We emulated our heterogeneous testing environment using DVFS, and we compared the results to default scheduling policy in Hypervisor's scheduler. Nevertheless, the results show performance improvements for VMs that run either CPU-intensive or I/O-intensive applications. Finally, the measured power savings of our heterogeneous testing environment reach up to 25%.*

Keywords: Virtualization, Heterogeneous, Scheduling, Hypervisor, Management.

1. Introduction

Heterogeneous multicore processors could be a common architecture of future multicore processors due to their performance per watt compared to homogeneous processors [1,2,3]. A single processor will contain hundreds of cores that vary in some micro-architecture features such as clock frequency, cache size, area, and others [4], but these cores exploit the same instruction-set architecture. A single chip might have several complex cores and many simple cores. The simple cores are characterized as low-speed clock frequency and low power consumption while fast cores are equipped with high-performance features such as high-speed clock frequency and high power consumption. Consequently, their potential to achieve different levels of performance that meet applications heterogeneity has prompted researchers in the operating systems domain to implement heterogeneous aware schedulers [5,6,7].

Nevertheless, current Hypervisors' schedulers such as Xen [8] do not support heterogeneous multicore processors, but this issue has been recently tackled in [9]. Authors in [9] have implemented An Asymmetry-Aware Scheduler for Hypervisors (AASH). Using AASH scheduler achieves a good performance improvement for CPU-intensive applications, but this improvement comes with performance degradation for memory and I/O intensive applications. A hypervisor scheduler is considered efficient if it assigns a virtual CPU (vCPU) to run on the appropriate cores based on the application characteristics in terms of CPU-intensive, Memory-intensive, or I/O-intensive. Further, the scheduler must have knowledge of the physical processors' architecture and their characteristics such as cores' clock frequency. By this knowledge, VMs with CPU-intensive applications should be assigned to complex fast cores to be executed faster. Generally, scientific applications are CPU-intensive, multithreaded, and fewer CPU stalls due to infrequent memory accesses or I/O operations. On the other hand, I/O-intensive could be assigned to simple slow cores without losing significant performance and achieving the power savings.

In this paper, we used NAS Parallel Benchmarks [10] as CPU-intensive application and netperf benchmark [11] as I/O-intensive application. We denoted performance sensitivity to CPU clock frequency as "performance-frequency sensitivity" and performance dependency on Domain-0 as "performance-Domain-0 dependency". Our scheduling-policy based on these two categories: "performance-frequency sensitivity" and "performance-Domain-0 dependency" to assign a vCPU to the appropriate core. Consequently, the results showed good performance improvements for VMs with CPU-intensive applications and for VMs with I/O-intensive applications as well. Further, in some experiments, the average combined performance gain reaches up to 70%. Eliminating sources of delay is the foundation of these improvements. Finally, our heterogeneous experimental environment achieves 25% of power savings. The power savings are gained from this architecture, which runs on two cores with high frequency and other two cores with low frequency.

The key contributions are as follows:

- We present and classify sources of delay that might lead to performance degradation of the whole system such as inter-process commutation and scheduling delay.

- Then, we illustrate performance of NPB benchmark performance-frequency sensitivity for both OpenMP and SERIAL versions of NPB benchmark. Similarly, we investigate sensitivity I/O-intensive applications to clock frequency and their dependency on Domain-0 using netperf benchmark with TCP and UDP streams options.
- We present our modified scheduling-policy that applied to the default credit scheduler of Xen Hypervisor.
- Finally, we discuss the results of experiments showing performance comparison between the default and the modified scheduling-policy.

The rest of paper is organized as follows. The next section discusses sources of delay in virtualized environment and motivates our work. Section 3 presents details about experimental platform and the benchmarks that were used. In Section 4, we discuss sensitivity of VMs performance to CPU clock frequency. Performance evaluation is presented in Section 5. The related work is described in Section 6. Finally, conclusions is presented in Section 7.

2. Background and Motivation

2.1 Background

In virtualized servers, virtual CPUs (vCPUs) of a virtual machine (VM) usually experience scheduling delay due to their competition on physical CPUs (pCPUs) with other vCPUs of the co-hosted VMs including the privileged domain of the Hypervisor (i.e., Domain-0). We discuss sources of delay that impede VMs to achieve the best performance. Sources of delay effect on Network I/O was discussed in [12], meanwhile inter-process communication delay was experienced in [13] due to threads synchronization. We point out these sources as follows.

a) Delay influences network I/O VMs performance:

1) The network communication between two VMs is a type of I/O that mainly depends on Domain-0 because a VM does not have privileges to access the physical NIC. Nevertheless, more details could be found in [12].
   - Sending a packet from a VM to another VM in the same host might experience delay due to scheduling Domain-0. The delay is the time period between a VM (a sender) copying a packet into the Domain-0's transmission-I/O-ring and Domain0 being scheduled next to notify another VM (a recipient) in the same host by setting up an event channel notification.
   - Sending/Receiving a packet from a VM to another VM into another host: The delay is the time period between a VM (a sender) copying a packet into the Domain-0's transmission-I/O-ring and Domain0 being scheduled next to send it via the physical NIC of host and the time period between receiving a packet in the physical NIC of the server hosting the recipient VM and Domain-0 being scheduled next to set up an event channel notification for the recipient VM.
2) Delay related to sender VMs scheduling which is the waiting period of a VM to be scheduled for copying a packet into the Domain-0's transmission-I/O-ring.
3) Delay related to recipient VMs scheduling which is the duration between when Domain0 sets up an event channel notification for the recipient VM and when the recipient being scheduled next to read the packet from Domain-0's transmission-I/O-ring.

b) Delay influences CPU-intensive VMs performance:

1) To achieve better I/O latency, the Xen Credit scheduler prioritizes vCPU I/O-intensive. When a vCPU is blocked waiting for I/O it will not consume credits; when it wakes, it enters the BOOST state and may immediately preempt running vCPU. Generally, vCPU (CPU-intensive) has less credit than vCPU (I/O-intensive). The frequent preemption of vCPU degrades performance especially for cache sensitive applications because some pCPU cycles go for cache-warming.
2) Delay related to inter-process communication comes from asynchronous assignment for vCPU. Xen credit scheduler assigns vCPU asynchronously to satisfy the fairness among vCPUs in the host. However, asynchronous scheduling decreases performance of a multithreaded application that needs synchronization among its threads.

After introducing sources of delay, we give an overview of Xen's credit scheduler [8]. The scheduler gives each vCPU 300 credits for a 30ms accounting period. A 100 credits is subtracted from vCPU each tick. A tick equals 10ms. The scheduler selects the next vCPU to run on pCPU after prioritizing vCPUs with either OVER or UNDER according to their remaining credits. The selection is preformed when the current running vCPU finishes its time-slice or its status becomes idle or blocked.

2.2 Motivation

Increasing number of cores in a single chip has become the mainstream industry to avoid vertical scaling of CPU frequency. A single chip expected to contain hundreds of cores that could be heterogeneous either by design or due to variability and possibility of defects with time [14]. Nevertheless, Heterogeneous multicore promises to achieve 60% in power saving compared to homogeneous [3]. Similarly, using virtualization technologies realizes efficient power savings by hosting multiple virtual machines on a single physical server. In this paper, we combine heterogeneous processors with virtualization technology to demonstrate the potentials of this combination in achieving power savings and maintaining applications performance at the acceptable level. Current Hypervisors' schedulers based on the round robin scheduling algorithm ensure fairness share of physical cores among VMs, but these schedulers were tailored for homogeneous cores, so using them in heterogeneous environment causes large performance losses. However, the advantages of this combination could not be achieved without dynamic classification for VMs and schedulers aware of VMs classification and processors' heterogeneity. This paper sheds light on some of these advantages to motivate research in this area.

3. Testing Environment and Benchmarks

3.1 Experimental Platform

Our experimental platform is a Dell OPTIPLEX 980 server with an Intel quad-core processor frequency range 2.79 - 1.2GHz, 8MB shared L3 Intel Smart Cache, and VID-Voltage rang 0.6500V-1.4000V. The server is equipped with 8GB memory. We emulated a heterogeneous processor according to expected frequency ranges in future heterogeneous systems[4], so we set two cores with high clock frequency F_F= 2.79GHz and other two cores with low clock frequency F_L=1.33GHz. We considered each core as a physical CPU (pCPU). Our experimental virtualized environment based on Ubunutu-10-32bit-Xen 4.1. Ubuntu operating system was used for para-virtualized unprivileged domains. The number of VMs and vCPUs was changed according to experiments purpose, and each experiment in this paper was repeated at least five times and the average of these readings was considered.

3.2 Benchmarks

The NAS Parallel Benchmarks [10] (NPB) was designed to evaluate the performance of HPC systems. NPB consists of a set of programs that differ in dataset size. The dataset size increases according to the chosen class (i.e., S, W, and A-D) during compilation. Further, NPB suite comes in a variety of versions: SERIAL, OpenMP, MPI, and Java. The SERIAL and OpenMP versions were used in our experiments and compiled with class C dataset which is the second largest dataset after class D. Authors of [15] studied NPB characteristics and provided performance analysis for MPI version. Generally, the NPB programs show a high CPU utilization, and this indicates that those programs are computation-intensive and infrequently blocked for communication or I/O operation. When a program runs with four threads, the communication patterns in these programs are as follows. BT and SP exhibit a mesh communication pattern, but BT includes a number of I/O operations. CG shows a one dimensional nearest neighbor chain pattern. LU and EP show a ring and negligible communication pattern respectively. Finally, CG and LU are communication intensive and their message size is large compared to the other programs, but LU is a synchronous-sensitive among its threads[13]. We denote OpenMP version of NPB by NPB-OMP that encloses CPU-intensive parallel programs, and the SERIAL version by NPB-SER that includes CPU-intensive single thread programs. Furthermore, we refer to individual program in NPB suite using this notion EP-OMP which means EP program of OpenMP version, or EP-SER which means EP program of the SERIAL version.

Netpref [11] is a network benchmark with a variety of options. We used netperf with TCP_STREAM option to measure TCP channel bandwidth and with UDP_STREAM to measure UDP channel bandwidth. In this paper, we refer to them with netperf-TCP and netperf-UDP respectively.

4. VMs SENSITIVITY ANALYSIS

In this section, we analyzed sensitivity of VMs' performance to changes in CPU clock frequency for VMs that run CPU-intensive and I/O-intensive applications. Then, we illustrated dependency of VMs' on Domian-0 for VMs with I/O-intensive applications.

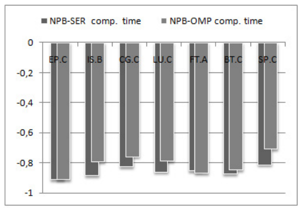
4.1 VMs with NBP Sensitivity Analysis

To analyze VMs performance-frequency sensitivity, we used NBP-SER and NPB-OMP benchmarks as CPU-intensive programs. In this experiment, we pinned vCPUs of Domain-0 to cores (0,1) and vCPUs of VMs were pinned to the another two cores (2,3) to avoid Domain-0's influence on the VMs; in other words, to prevent Domain-0 from being queued with the VMs in the same queue. First, the experiment was run while the cores (2,3) were set to run with high frequency F_F=2.79GHz as fast cores. Then, it was run again after changing frequency settings of the cores (2,3) to low frequency F_S=1.33GHz as slow cores. Finally, we used the price elasticity of demand economics formula to determine program's completion time and throughput sensitivity of clock frequency. We considered T the completion time and Th the throughput as the demand, and F clock frequency as the price. The sensitivity was determined using the formula from [18]. $E_{T,F}$ is the completion time sensitivity of clock frequency, and $E_{Th,F}$ is throughput sensitivity of clock frequency.

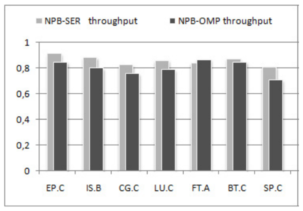
$$E_{T,F} = \frac{T_F - T_S}{F_F - F_S} * \frac{F_F + F_S}{T_F + T_S} \tag{1}$$

$$E_{Th,F} = \frac{Th_F - Th_S}{F_F - F_S} * \frac{F_F + F_S}{Th_F + Th_S} \tag{2}$$

Due to the inverse relationship between CPU frequency and completion time, $E_{T,F}$ values are negative, so completion time increases as CPU frequency decreases and



(a)



(b)

Fig. 1: Performance-frequency sensitivity for NPB-OMP and NPB-SER versions run on a VM with two vCPUs. (a)sensitivity of completion time to frequency, and (b)sensitivity of throughput to frequency.

vice versa. On the other hand, $E_{Th,F}$ values are positive because of the direct relationship between CPU frequency and throughput. Program speedup depends on program characteristics, but it does not have a liner relationship with CPU frequency. However, CPU-intensive programs might have a semi-liner relation with frequency because of either infrequent memory accesses or I/O operations. Figure 1 shows NPB-OMP and NPB-SER benchmarks performance-frequency sensitivity (i.e., completion time and throughput). In NPB benchmark, each program has a different memory access behavior and various inter-process communication patterns. These characteristics determine sensitivity of a program to frequency changes. For example, the completion time of EP-OMP and EP-SER programs had the same and the highest sensitivity. This high sensitivity due to the negligible inter-process communication in the multithreading EP-OMP program, and none inter-process communication in the single thread EP-SER program. Furthermore, EP-OMP is seldom memory access compared with CG-OMP and LU-OMP. Generally, NPB-SER programs sensitivity to frequency changes was higher than NBP-OMP due to the sequential execution of instructions in NPB-SER and inter-process communication patterns or I/O operations in some of NBP-OMP programs such as CG and BT respectively. On the other hand, NBP-OMP programs with intensive inter-process communication were less sensitive to frequency such as CG-OMP and LU-OMP. FT, a mixed type program, almost had the same sensitivity in NPB-SER and NPB-OMP. Unlike LU-OMP, BT-OMP includes a number of I/O operations that do not need synchronization among its threads.

4.2 VMs with I/O Sensitivity Analysis

We analyzed sensitivity of VMs performance with I/O-intensive to CPU frequency. Then, as I/O operations depend on Domain-0, we tested VMs performance-Domain-0 dependency.

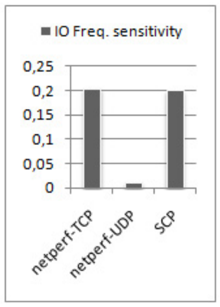
4.2.1 CPU Frequency Sensitivity

In this experiment, we ran netperf with TCP-STREAM and UDP-STRAEM options to test I/O performance-frequency sensitivity using formula 2. The setting of this experiment was the same setting when we tested VM with NBP sensitivity. As shown in figure 2-(a), TCP test is more sensitive to core frequency than UDP due to the nature of TCP-packet; UDP does neither message fragmentation nor reassembly. Further, the aggregate costs of non-data touching overheads consume majority of the total software processing time. The non-data touching overheads come from as network buffer manipulation, protocol-specific processing, operating system functions, data structure manipulations (other than network buffers), and error checking[16]. To validate our test, we used SCP application TCP-based to transfer a 500MB file between two VMs and we found the same results obtained using netperf-TCP.

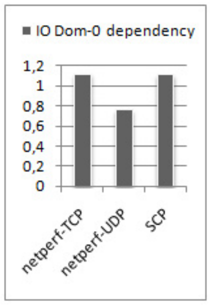
4.2.2 VMs with I/O Domain-0 Dependency

In this experiment, we ran netperf benchmark with TCP-STREAM and UDP-STRAEM options to test I/O performance-Domain-0 dependency. For this end, we reversed the scenario of VM performance-frequency sensitivity, so the cores (2,3) settings were not changed but were set to high frequency F_F=2.79GHz where VMs were pinned in cores (2,3). On the other hand, The cores (0,1) were set to high frequency F_F=2.79 GHz where Domain-0 was pinned. Then, we ran it again while the frequency of cores (0,1) is low F_S=1.33GHz. Finally, we computed the performance-Domain-0 dependency using formula (2). The result of this experiment is shown in figure 2-(b). It illustrates that both netperf-TCP and netperf-UDP depend on Domain-0 for commutation between to VMs, but netperf-TCP depends on Domain-0 more than netperf-UDP.

The conclusion is that applications based on TCP protocol are frequency sensitive and they are Domain-0 dependant as depicted in figure 2-(a) and figure 2-(b) respectively.



(a)



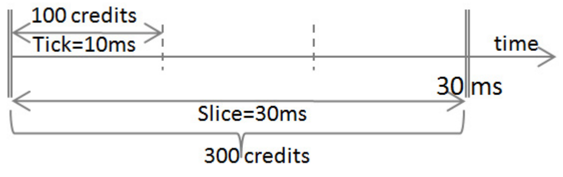
(b)

Fig. 2: Performance-frequency sensitivity and Domain-0 dependency for NPB-OMP and NPB-SER versions run on a VM with two vCPUs. (a) performance-frequency sensitivity, and (b) performance-Domain-0 dependency.

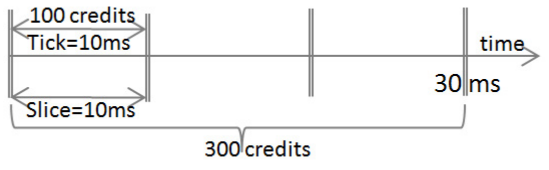
5. Performance Evaluations

In this section, we evaluated our improved scheduling-policy with the following rules:

- The weight of VM is proportional to the number of vCPUs.
- CPU-intensive vCPU should not be queued with I/O-intensive vCPU. Furthermore, CPU-intensive vCPU should be placed in the fast pCPU's queue meanwhile I/O-intensive vCPU in the slow pCPU's queue.
- A virtual machine with CPU-intensive application and a single vCPU should be placed in fast pCPU's queue to accelerate the sequential execution.
- The time-slice for the fast pCPU's queue is 30ms and time-slice for slow cores is 10ms as show in figure 3. We chose the value 10ms for the short slice as one tick to avoid high context switching and to keep consistent credit accounting.
- The cores settings for the experiments were that the fast cores (0,1) ran on frequency F_F=2.79GHz and the slow cores (2,3) ran on frequency F_S=1.33GHz.



(a)

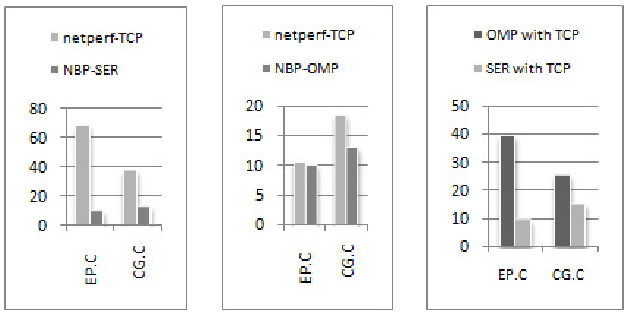


(b)

Fig. 3: Scheduling time-slice modifications.(a) time-slice = 30ms for fast cores, and (b) time-slice = 10ms for slow cores. The accounting period of vCPU is 30ms for both fast and slow cores.

5.1 I/O and CPU-intensive Isolation

In this experiment, we created three VMs one with two vCPUs while each of the other two VMs has one vCPUs. We ran netperf on the two VMs with one vCPU for testing TCP and UDP bandwidth channels between them. The VM with two vCPUs used to run NPB-SER and NBP-OMP programs. We ran the three VMs with our new scheduling-policy. First, we used EP and CG programs in NPB-SER with netperf, then EP and CG of NPB-OMP were used. We pinned the VMs with I/O to the slow cores (2,3) and the VM with CPU-intensive was pinned to the fast cores (0,1). Performance improvement for both I/O and CPU-intensive VMs compared to the default scheduler is illustrated in figure 4. Figure 4-(c) shows that the performance gain of CG.C is better than EP.C. Indeed, EP.C has negligible inter-process communication compared to CG.C which has also memory accesses. On the other hand, netperf-TCP throughput when co-hosted with VM that ran NBP-SER is better than when co-hosted with VM that ran NBP-OMP. As seen in figure 2-(b), netperf depends on Domain-0 and NPB-SER is a single thread test that gave Domain-0 chance to be scheduled in fast cores and improve I/O operations for netperf-TCP. The aggregate average gain is depicted in figure 4-(c). Obviously, isolating CPU-intensive vCPUs from I/O-intensive vCPUs was the main reason for performance improvement. Using isolation eliminated the sources of delay that affect CPU-intensive vCPUs performance.

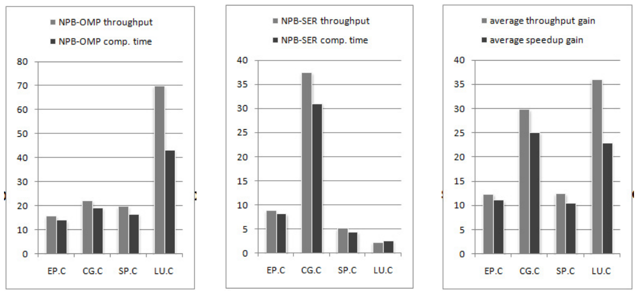


(a) (b) (c)

Fig. 4: I/O and CPU-intensive Isolation Performance improvements; netperf-TCP run on a VM with one vCPU,and NPB-OMP run on a VM with two vCPUs. (a) Throughput gain for NPB-OMP and netperf-TCP benchmark, (b) throughput gain for NPB-OMP and netperf-TCP benchmark, and (c) the average improvement of the overall system.

5.2 VMs with sensitive Inter-process Comm.

In this experiment, we tested the performance gain for inter-process communication intensive such as CG and LU of NPB-OMP version. The performance of NPB-OMP benchmark in VM is near to the performance in physical server as long as the vCPUs are less than pCPUs, and LU-OMP is the most sensitive program to communication delay [13]. For testing inter-process communication intensive program performance improvement, we created one VM with one vCPU and another VM with four vCPU. Nevertheless, we had five vCPUs in addition to four vCPUs for Domain-0. The performance gain is illustrated in figure 5 where figure 5-(a) shows Throughput gain and completion time speedup for NPB-OMP while figure 5-(b)illustrates Throughput gain and completion time speedup for NPB-SER. Figure 5-(c) shows the average aggregated performance gain for NPB programs with two versions. Nevertheless, LU-OMP gained about 70% performance improvement. This improvement due to changing the time-slice of the slow pCPUs' to 10ms which increases scheduling frequency. Increasing scheduling frequency gave chance for inter-process communication and synchronization. Further, decreasing time-slice decreases holding time when vCPU status "busy blocking" holds pCPU [17]. A lot of "busy blocking" wastes pCPU cycles and degrades the overall system performance.



(a) (b) (c)

Fig. 5: CPU-intensive with inter-process communication intensive performance improvements:NPB-OMP run on a VM with four vCPUs and NPB-SER run on a VM with one vCPU. (a) Throughput gain and completion time speedup for NPB-OMP,(b) throughput gain and completion time for NPB-SER, and (c) the average improvement of the overall system.

5.3 Estimated Power Savings

As our physical platform is homogeneous, we have used dynamic voltage and frequency scaling (DVFS) mechanism to emulate a Heterogeneous environment. Nevertheless, we calculate the expected power saving in our experimental environment using formula (3). Formula (3) shows that the power consumption has a direct proportional to F which is the clock frequency and the square of Voltage (V).

$$P = C * F * V_{VID}^2 \tag{3}$$

Consequently, homogeneous architecture (HO) will consume P_{HO} with four cores ran on high frequency F_F=2.79GHz and V_{VID}=1.4V, while heterogeneous architecture (HE) will consume P_{HE} with two cores with high frequency F_F=2.97GHz with V_{VID}=1.4V and the other two ran with low frequency F_S=1.33GHz with V_{VID}=0.65V. The theoretically power savings gain is 45% calculated using (1-P_{HE}/P_{HO})*100 formula, but the measured values of power savings reach up to 25%.

6. Related Work

Most of the works related to heterogeneous processors have been done in operating systems domain. Our work overlaps with two categories: (i) Heterogeneous scheduling awareness for OS, (ii) Heterogeneous scheduling awareness for Hypervisors. First, the algorithms for heterogeneous awareness in operating systems field can be described as follows. The algorithm presented in [6] assigns the best threads (i.e., threads with high computation demands) to run on fast cores; however, selecting the best threads via continuous monitoring of performance based on instructions per cycle (IPC). Furthermore, continuous monitoring for threads before getting the best thread to the fast core might take long time and consume more resources. This algorithm could be modified for Hypervisors if we consider a virtual machine as a long lived thread. By determining the architectural properties of an application, the algorithm in [16] find the best threads to be assigned to fast cores. However, we used the same methodology to classify virtual machines that proposed in [16]. The algorithm proposed in [19] boost the sequential phases of parallel applications by executing them on fast cores. In our work, the scheduler assigns CPU-intensive VMs with single vCPU to fast cores. In [20], the scheduler places more threads on fast cores than slow cores, where the core load is proportional to its frequency speed. Unfortunately, this technique is not suitable for Hypervisors because virtual machines will experience cache contentions that degrade their performance [21].

Second, heterogeneous scheduling awareness for Hypervisors according to the recent work was presented in [9]. These were not much research done in this domain;however, paper [9] was implemented An Asymmetry-Aware Scheduler for Hypervisors which is a scheduler aware to heterogeneity of multicore processor. Using AASH scheduler achieves a good performance improvement for CPU-intensive VMs, but this improvement came with performance degradation for memory and I/O intensive VMs. Our idea of shortening the time-slice for the slow cores similar to dynamic switching frequency scheduling policy proposed in [20], but it was proposed for homogeneous environment and for pinned virtual machines. Authors in [20] suggested to set the time-slice to one millisecond for some CPU-Intensive VMs, but according to the comments in Xen's source code [22], a 1ms is the given delay for pCPU to build its cache for vCPU between vCPU migrations. So, one millisecond time-slice is very expensive for CPU-intensive VMs due to frequent cache-warming that means more pCPU cycles losses.

7. Conclusions

In our scheduling-policy, we invested the recommendations that were proposed for operation systems schedulers. The policy is suitable for virtualized environments that co-hosted heterogeneous type of VMs. We presented scheduling-policy that aware of virtual machines and physical host heterogeneity to realize the promises of Heterogeneous multicore processor in virtualized environments. Analyzing VM characteristics is the most significant stage to place a VM at the suitable cores that keep its performance acceptable. Furthermore, elimination of delay sources that impede performance gaining could bring good performance improvements. Our results proved that heterogeneous mutlicore systems could add more advantages in terms of power savings when combined with virtualiziation technologies. Nevertheless, the average combined performance improvements gain for both CPU-intensive and I/O-intensive VMs reached up to 70% in some experiments compared with default scheduling-policy of Hypervisor's scheduler. Furthermore, the power savings achieved in our experimental environment almost realize the promises in [3], where 25% of power savings have been achieved compared with homogeneous architecture.

References

[1] K. Asanovic, R. Bodik, B. Catanzaro, J. Gebis, and P. Husbands, "The Landscape of Parallel Computing Research: A View From Berkeley," UC Berkeley Technical Report UCB/EECS-2006-183, 2006.

[2] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguade, "Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors," *IEEE Computer Architecture Letters 5(1):4*, 2006.

[3] R. Kumar, K. I. Farkas, and N. Jouppi et al, "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," *In Proc. of MICRO 36*, 2003.

[4] S. Borkar, "Thousand Core Chips-A Technology Perspective," *in Proc. of the DAC*, 2007.

[5] R. Kumar, Dean M. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas, "Single-ISA Heterogeneous Multicore Architectures for Multithreaded Workload Performance," *in Proc. of the 31st Annual International Symposium on Computer Architecture*, 2004.

[6] R. Kumar, D. M. Tullsen, and P. Ranganathan et al, "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," *in Proc. of ISCA*, 2004.

[7] M. Becchi and P. Crowley, "Dynamic Thread Assignment on Heterogeneous Multiprocessor Architectures," *in Proc. of the Conference on Computing Frontiers*, 2006.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *in Proc. SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[9] V. Kazempour, A Kamali, and A. Fedorova, "AASH: an asymmetry-aware scheduler for Hypervisors,"*in Proc. of the 6th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, 2010.

[10] R. V. der Wijngaart, "NAS Parallel Benchmarks v. 2.4", NAS Technical Report NAS-02-007, October 2002.

[11] R Jones, "NetPerf:a Network performance benchmark," http://www.netperf.org.

[12] S. Govindan, A. R. Nath, A. Das, B. Urgaonkar, and A. Sivasubramaniam, "Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms," *in Proc. of the 3rd international conference on Virtual execution environments*, pp. 126-136, 2007.

[13] C. Xu, Y. Bai, and C. Luo, "Performance Evaluation of Parallel Programming in Virtual Machine Environment," *In Proc. of Sixth IFIP International Conference on Network and Parallel Computing*, pp. 140-147, 2009.

[14] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *in IEEE Micro*, 5(6):10-16, 2005.

[15] J. Subhlok, S. Venkataramaiah, and A. Singh, "Characterizing NAS benchmark performance on shared heterogeneousnetworks," *in Proc. of 11th International Heterogeneous Computing Workshop*, 2002.

[16] J. Kay and J. Pasquale, "The Importance of Non-Data Touching Processing Overheads in TCP/IP," *In Proc. of ACM SIGCOMM*, 1993.

[17] H. Chen, H. Jin, K. Hu, and J. Huang, "Dynamic Switching-Frequency Scaling: Scheduling pinned Domains in Xen VMM," *in Proc. of 39th International Conference on Parallel Processing*,pp. 287-296, 2010

[18] D. Shelepov and A. Fedorova, "Scheduling on Heterogeneous Multicore Processors Using Architectural Signatures," *in Proc. of the Workshop on the Interaction between Operating Systems and Computer Architecture*, in conjunction with the 35th International Symposium on Computer Architecture (Beijing, China, June 21-25, 2008). WIOSCA '08.

[19] J. Saez, M. Prieto, A. Fedorova, and S. Blagodurov, "A Comprehensive Scheduler for Asymmetric Multicore Processors," *in Proc. of the 5th ACM European Conference on Computer Systems (EuroSys) 2010*, 2010.

[20] T. Li, D. Baumberger, and D. A. Koufaty et al, "Efficient Operating System Scheduling for Performance-Asymmetric Multi-Core Architectures,"*In Proc. of SC '07,* pp. 1-11, 2007.

[21] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling Virtual Machine Performance: Challenges and Approaches," Intel Corporation, 2009.

[22] http://lxr.xensource.com/lxr/source.