International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 1, February 2016, pp. 205~211
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i1.9341

A Prototype of Online Dynamic Scaling Scheduler for Real-Time Task based on Virtual Machine

Junho Seo, Kyong Hoon Kim
Department of Informatics, Gyeongsang National University, Korea

Article Info

Article history:

Received Aug 20, 2015
Revised Nov 9, 2015
Accepted Nov 28, 2015

Keyword:

Dynamic frequency scaling
Power-aware
Real-time scheduler
Virtualization

ABSTRACT

In this paper, we provide an architecture of power-aware scheduler for real-time virtual machine system using dynamic frequency scaling mechanism This architecture provides that how to manage real-time resource and how to control processor frequency. In addition, we propose two scheduling schemes that utilize slack time to adjust processor frequency without violating tasks deadline. Based on the provided architecture, we implement a virtualization framework with online power-aware scheduler using RT-Xen real-time hypervisor and Litmus-RT real-time OS. Our implementation manages entire of real-time resource and processor frequency depending on system policy. For trasferring real-time requirements, we implement an interface using hypercall mechanism. To evaluate provided system, we analyze performance evaluation in various aspects.

Corresponding Author:

Kyong Hoon Kim,
Department of Informatics,
Gyeongsang National University,
Jinjudae-ro 501, Jinju, Gyeongsangnamdo, Republic of Korea.
khkim@gnu.ac.kr

1. INTRODUCTION

As computing technology is grown, the power consumption has become an important issue in embedded system. As the computing performance is increased, power consumption is also increased. This problem induces increasing management cost such as electricity charge and cooling cost.

Dynamic Voltage Frequency Scaling (DVFS) technique is most commonly used technique for reducing the power consumption in computing systems. It allows to control processor frequency and voltage dynamically. Generally, when the processor frequency or voltage is decresed, the power consumption is also decresed. The DVFS-supported processor can easily achieve a reduced power consumption whenever the maximum frequency is not required by simply scaling down the operating frequency of the processor. However, decreasing frequency of processor induces that tasks spend more time to complete.

The virtual machine technology can execute different software platforms in a single system, which has been commonly used as a basic platform in various application area to support resource sharing, partitioning, and etc. This technology includes reallocating entire resource by virtualized hardware resource. In [1], they provide a strategy to adopt server virtualization for using system resource efficiently. Thus it can be used for a solution of efficient scheduling in complex real-time systems.

In real-time systems, a task has real-time requirement such as deadline and period. The task should be operated to satisfy real-time requirement. To guarantee working of task, the scheduler manages all of tasks by scheduling algorithm. Thus, to study resource sharing algorithm is important issue in real-time systems.

Many recent studies have been provided for real-time systems based on virtual machine. RT-Xen [2] is a kind of real-time hyperviosr based on Xen [3]. The original Xen hypervisor scheduling algorithm is Credit-based CPU scheduling, where each virtual machine is allocated CPU time in proportion to the credit. In [2] [4], they modified the Credit-based scheduler to support real-time requirement.

In this paper, we provide a prototype of online power-aware scheduler for real-time task based on hypervisor. The proposed system can reduce power consumption using DVFS mechanism while guaranteeing real-time requirements

2. RELATED WORKS

Xen hypervisor is kind of virtual machine framework to support various scheduling algorithm such as credit, earliest deadline first, and arinc 653. In [4], they expanded the Xen hypervisor scheduler to support real-time requirement for soft real-time applications. They implemented the load balancer to minimize the latency of real-time applications by specifying their required laxity values. Xi, et al. [2] proposed RT-Xen to support real-time scheduling algorithm such as Earliest Deadline First (EDF) and Rate Monotonic (RM) in Xen hypervisor.

The Dynamic Voltage Frequency Scaling (DVFS) mechanism is a kind of solution which scales the processor frequency and voltage dynamically for reducing dynamic power consumption. But this mechanism has trade-off between power consumption and execution time. When the frequency is decresed, task needs more time to execute. This problme is fatal to real-time system. Increasing execution time induces by violation of deadline.

To solve reducing power consumption on real-time system, much research has proposed scheduling algorithm using DVFS technique for periodic tasks [5]-[7]. In [6], they provided a algorithm that selects optimal processor frequency ratio. In [7], they provided a method that how to change processor frequency at running time of each task. For reducing power consumption, they utilize slack time which is remaining time of real-time resource.

In [5], they defined the power-aware schemes for periodic real-time tasks in various levels, such as system level, component level and task level. In system level, whole system are executed on same frequency. On the other hand, in component level and task level, the components which like virtual machines or tasks are executed on different frequency depending on requirements of each component or task. Each level has own scheduler and power-aware module. The scheduler operates own scheduling algorithm using given resource from upper-layer scheduler and provides resource to under-layer components. The power-aware module manages power-aware information and controls the processor frequency depending on system policy.

3. SYSTEM MODEL

The proposed architecture for online power-aware scheduler is shown in Figure 1. In hypervisor, we define a Discrete Frequency Control (DFC) module which manages real-time requirement of tasks in guest OS and controls processor frequency depending on real-time requirement whenever the real-time requirement is changed. The DFC module controls CPU frequency to reduce power consumption and modifies real-time tasks requirement for guarantee real-time requirement. And we add hypercalls which transfer real-time requirement from guest VMs. In Guest VM, we implement DFC API to collect real-time requirements. The DFC API is using a hypercall interface to transfer the requirements to DFC module in hypervisor.

In VM, the DFC API manages real-time requirement which is sum of real-time requirement in each task. The real-time requirement defines by $T(\Pi, \Theta)$ where Π is period of task and Θ is execution time of task. We consider that deadline is same to period. If a given job is failed to run until a period time, it is out of deadline. The DFC API includes DFC hypercall interface which is invoked to inform the change of task status when the task is created or terminated.

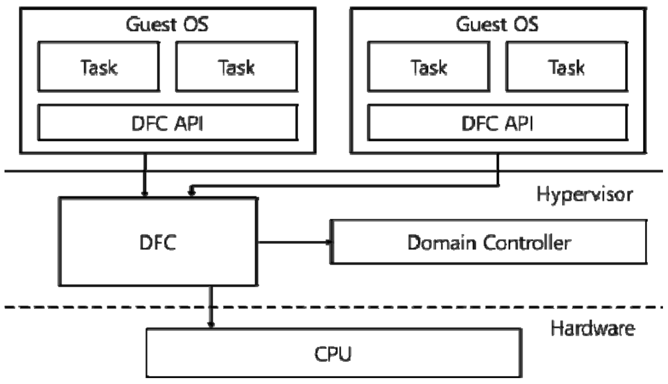

Figure 1. The architecture of proposed system

To reduce power consumption, we define two policies based on system level and component level principles.

3.1. System Static Frequency (SSF)

The SSF policy picks the minimum speed level among which all the virtual machines are scheduled. Thus, the system frequency is defined as follows:

$$F_{SSF} = min_{f_i \in F} \left\{ f_i \middle| \sum_{vm_j \in VM} \frac{\Theta_j}{\Pi_j} \cdot \frac{f_{max}}{f_i} \leq 1 \right\}$$

where i is number of speed level, j is number of virtual machine, f is level of frequency scaling, Θ is execution time of task, and Π is period of task. f_{max} is maximum frequency of processor, it is depending on system processor.

The SSF policy runs at same frequency for all virtual machines. When the tasks are created or terminated, the scheduler re-calculates optimal frequency. This policy has a little overhead less than Component Static Frequency (CSF) method which is explained below because it is invoked once per one variation of task. However, the SSF policy has any wasteful power consumption margin. To keep a same frequency is not satisfied to fully reduce power consumption for all of the virtual machine.

3.2. Component Static Frequency (CSF)

The CSF policy allows that each virtual machine to run on its minimum speed level. Each virtual machine frequency is defined by:

$$F_{CSF} = min_{f_{i,j} \in F} \left\{ f_{i,j} \middle| \sum_{vm_j \in VM} \frac{\Theta_j}{\Pi_j} \cdot \frac{f_{max}}{f_{i,j}} \leq 1 \right\}$$

In the CSF policy, each virtual machine runs at different frequency. The CSF policy can cover the wasteful power consumption margin in SSF because it uses different speed level. However, it has more overhead than SSF because it should change CPU frequency when the virtual machines are switched.

After the changing frequency, execution times of tasks are changed so that to avoid deadline violation. The DFC module changes execution times of VMs and the DFC API changes execution time of tasks. Generally, execution time of task is in inverse proportion to processor frequency. Thus, execution time is changed by ratio of decreasing process frequency, where ratio is defined by

$$e_i' = e_i \cdot f_{max} \ / \ f_i$$

where e_i' is changed execution time and e_i is a given execution time of task. Changing execution time is part of policy, thus this function is invoked when the frequency is changed.

We implement the adaptive power-aware scheduler based on rtpartition scheduler in RT-Xen hypervisor. The rt partition is real-time scheduler in RT-Xen based on EDF algorithm. We modify the scheduler to support power-aware function. We also use a Litmus-RT [8] real-time linux OS using GSN-EDF algorithm for power-aware VM.

To support the proposed architecture, we add the DFC module, DFC hypercall in RT-Xen and DFC API in VM. The DFC module manages power-aware information of all VMs in the system. When VMs provide their real-time requirement, the module decides optimal frequency of VMs according as policy and modifies execution time of VMs. When the frequency is selected, changing frequency function is invoked before conext switch.

The DFC hypercalls consist of three hypercalls: *dfc_init, dfc_op,* and *dfc_renew*. The *dfc_init* is initialization function that prepares for supporting DFC. At the end of function, the hypercall obtains list of frequency level by return value. It can also use cleanup function that frees the VM specific DFC memory. This hypercall is invoked when the VM is booting and shuting down. The *dfc_op* transfers real-time requirements which contain list of utilization at each frequency levels. This requirement lists are managed by DFC module and DFC module picks to compare these lists. To transfer data at one time, we devide 64bit argument to 8 blocks of 8bit argument. Each block has ranged between 0 and 255 unsigned integer, but we just use from 0 to 100 because it contains utilization that shows percentage. Figure 2 shows that architeture of argument. The *dfc_renew* is to check the system changes such as frequency and VM execution time. When a task is created or terminated, the frequency and VM execution time is changed, this hypercall informs the changes. Than the DFC API catches this information and updates execution time of tasks. This hypercall is invoked periodically.

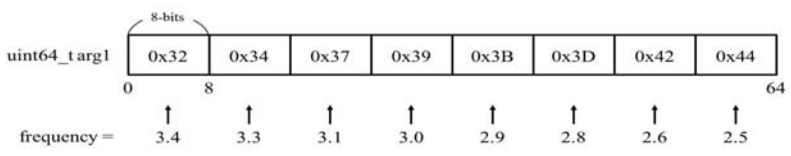

Figure 2. A structure of dfc_op argument

In VM, we add DFC API to manage real-time requirements of tasks in kernel. This API is provided real-time requirements and generates list of utilization at each frequency when the real-time task is created or terminated. To check system changes, we modify scheduler in VM kernel to invoke *dfc_renew* hypercall periodically. When the system changes are updated, the DFC API updates execution time of every real-time task.

For the simplicity, we restrict some argument. To calculate scheduling operation rapidly and transfer real-time requirement simply, the period in hypervisor scheduler is fixed by 100ms. Second, we represent the real-time requirement of hypercall interface as utilization. In CSF policy, we modify to pick frequency quickly using heuristic approach.

4. EXPERIMENTS

In order to simulate periodic real-time tasks, we use the π calculation program that calculates the π to a given number of digits after the decimal point and a simple sum program that adds a random number in double loop. A periodic task application is generated so that it executes a π calculation program or simple sum program every period.

To evaluate power consumption, we define two scenarios for each policy. In each scenario, we define two virtual machines. In first scenario, each virtual machine has two real-time tasks: The VM_1 has task set T_1*(7,2)* and T_2*(5,1)* where the first one is period and the second one is execution time. The T_1 executes 3 times in a single VM execution and T_2 executes 4 times. The VM_2 has task set T_1*(5,1)* and T_2*(6,1)*. The T_1 executes 2 times and T_2 executes a single time. Each task executes the simple sum program. In second scenario, each virtual machine has three real-time tasks that executes a PI calculation program: The VM_1 has task set T_1*(3, 0.2)*, T_2*(5, 0.5)* and T_3*(7, 1)*. The T_1 executes 6 times in a single VM execution, while T_2 executes 4 times and T_3 executes 3 times. The VM_2 has task set T_1*(2, 0.2)*, T_2*(4, 0.5)* and T_3*(6, 0.8)*. The T_1 executes 6 times, T_2 executes 4 times and T_3 executes 3 times. All of tasks are terminated when the tasks are done and re-created before the next period is started.

To compare our policy, we use power model [9] which the power consumption E is defined by

$$E = \alpha \cdot t/S \cdot S^3 = \alpha \cdot t/S^2$$

where α is a coefficient of a system, t is execution time, and S is processor frequency level. In this paper, we assume that $\alpha = 1$ for the sake of simplicity.

The figure 3 and figure 4 show that result in each scenario using power model. In the figures, power consumption value means calculating value by power model.

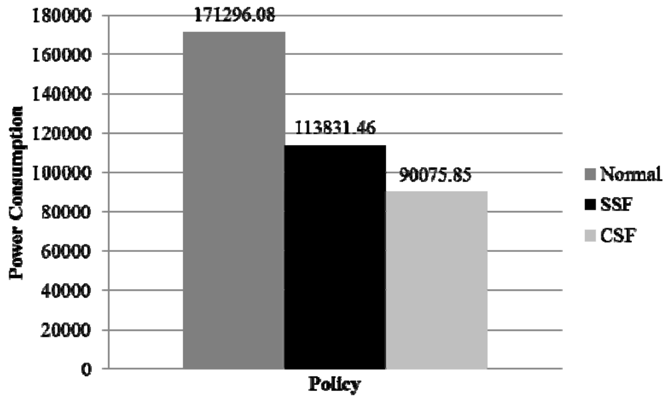

Figure 3. Power consumption of scenario 1

In Figure 3, it shows the results of power consumption using power model. The proposed policies spent less power than normal scheduling policy. Especially, the CSF spent the least power. It shows that decreasing frequency is more efficient to reduce power consumption even if total execution time is increased.

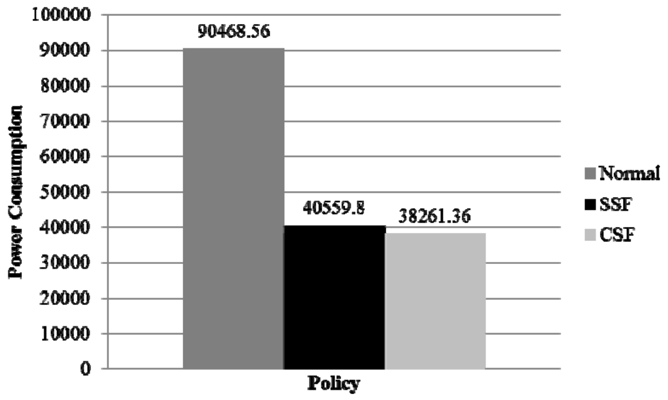

Figure 4. Power consumption of scenario 2

As shown on Figure 4, power consumption is more decreased by that variations are bigger. However, the power consumption of proposed policies are almost same. because it cannot reduce energy further under the frequency provided by system. Although it can reduce much more power consumption, it just runs on the minimum frequency provided by system.

In addition, we evaluate the overhead of power-aware control at each part. The control time means opertation time of scheduling and managing in DFC. The DFC API request time is that how long it takes to invoke hypercall. The changing frequency time is the required time that controls CPU frequency. Generally, real-time task has more than a millisecond execution time. Table 1 shows the result of overhead measurement. We analyzed that the proposed system has any influence on the running of real-time tasks.

Table 1. The result of overhead measurement

	Average	Maximum
Control time	4μs	7μs
DFC API request time	4μs	7μs
Changing frequency time	2μs	4μs

5. CONCLUSION

In this paper, we proposed an online power-aware scheduler for real-time hypervisor. The proposed system provides how to reduce processor frequency based on real-time requirement of tasks, how to transmit real-time requirement of tasks into hypervisor and how to guarantee execution time so that to avoid deadline violation.

In SSF policy, all the virtual machines are executed on same frequency. On the other hand, CSF policy, all the virtual machines are executed on different frequency depending on virtual machine's resource demand. To compare two policies, we expect that the CSF policy is better to reduce power consumption than SSF policy.

Throughout the experiments, we analyzed the power consumption and overhead. Especially, CSF policy spent less power than SSF policy. According to power model, power consumption is increased in proportion to execution time. However, power consumption is increased in proportion to square of frequency. It means, reducing frequency is much more profitable. Thus, the CSF policy is more profitable because CSF uses resource that unused remaining execution time in SSF.

Based on the proposed system, we will improve the system further. For example, we will add more system policies such as task level static. In addition, we plan to develop various power-aware scheduling algorithms such as rate monotonic on proposed system and support multi-core mechanism that operates power-aware scheduling at each core.

REFERENCES

[1] J.C. Song et al, "A Strategy for Adopting Server Virtualization in the Public Sector: NIPA Computer Center", *Journal of Information and Communication Convergence Engineering,* vol. 10-1, p.61-65, 2012.

[2] S. Xi et al, "*RT-Xen: Towards Real-time Hypervisor Scheduling in Xen*", in Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on. IEEE, p. 39-48, 2011.

[3] Xen Hypervisor, http://www.xenproject.org/

[4] M. Lee et al, "*Supporting soft real-time tasks in the Xen hypervisor*", in Proceeding of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, p. 97-108, 2010.

[5] G.M. Tchamgoue et al, "*Dynamic Voltage Scaling for Power-Aware Hierarchical Real-Time Scheduling Framework*", in Proceeding of 15th IEEE International Conference on Computational Science and Engineering, p. 540-547, 2012.

[6] Y. Shin and K. Choi, "*Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems*", in Proceeding of 36th Design Automation Conference, p. 134-139, 1999.

[7] P. Pillai and K.G. Shin, "*Real-Time Dynamic Voltage Scaling for Low Power Embedded Operating Systems*", in Proceeding of 18th ACM Symposium on Operating System Principles, p. 89-102, 2001.

[8] J. Calandrino et al. "*LITMUS RT: A Testbed for Empirically Comparing Real-Time Multiprocessor Schedulers*", in Proceedings of the 27th IEEE Real-Time Systems Symposium, p. 111-123, 2006.

[9] L. Niu and G. Quan, "*Reducing both Dynamic and Leakage Energy Consumption for Hard Real-Time Systems*", in Proceeding of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, p. 140-148, 2004.

BIOGRAPHIES OF AUTHORS

Junho Seo is currently a Ph.D. student at the Department of Informatics in Gyeongsang National University, Jinju-si, South Korea, where he also received his Bachelor of Science and Master of science degrees in 2013 and 2015, respectively. His research interests include virtualization, real-time systems, and power-aware computing.

Kyong Hoon Kim received his B.S., M.S., and Ph.D. degrees in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, 2005, respectively. Since 2007, he has been an assistant professor at the Department of Informatics, Gyeongsang National University, Jinju, South Korea. From 2005 to 2007, he was a post-doctoral research fellow at CLOUDS lab in the Department of Computer Science and Software Engineering, the University of Melbourne, Australia. His research interests include real-time systems, avionics, Grid and Cloud computing, and security.