A Survey of Methods for Improving GPU Energy Efficiency in Cloud Computing

Prof. Avinash Sharma[1] , Anchal Pathak [2]

[1](Associate Professor) ,[2](PG Scholar)

Department of CSE MITS, Bhopal, India

Abstract:In Cloud Computing, Recent years have witnessed a phenomenal growth in the computational capabilities and applications of GPUs. However, this trend has also led to dramatic increase in their power consumption. This paper surveys research works on analyzing and improving energy efficiency of GPUs. It also provides a classification of these techniques on the basis of their main research idea. Further, it attempts to synthesize research works which compare energy efficiency of GPUs with other computing systems e.g. FPGAs and CPUs. The aim of this survey is to provide researchers with knowledge of state-of-the-art in GPU power management and motivate them to architect highly energy-efficient GPUs of tomorrow. Dynamic Voltage Frequency Scaling (DVFS) techniques are used to improve energy efficiency of GPUs. Literature survey and thorough analysis of various schemes on DVFS techniques during the last decade are presented in this paper. Detailed analysis of the schemes is included with respect to comparison of various DVFS techniques over the years. To endow with knowledge of various power management techniques that utilize DVFS during the last decade is the main objective of this paper. During the study, we find that DVFS not only work solely but also in coordination with other power optimization techniques like load balancing and task mapping where performance and energy efficiency are affected by varying the platform and benchmark. Thorough analysis of various schemes on DVFS techniques is presented in this paper such that further research in the field of DVFS can be enhanced.

Index Terms: GPU, DVFS, Power Consumption.

I. INTRODUCTION

Cloud environment is latest scenario in IT industry. It indicates a computer model where users are provided with computing resources. These services include three parts like as Software as a Service, Platform as a Service and Infrastructure as a Service. Figure 1 shows the relationship of these services.

IaaS locates in bottom scale of cloud systems and it provides virtualized possessions such as storage, bandwidth and memory etc. PaaS provides a higher level of IaaS to create a cloud securely programmable. SaaS is a software delivery model [1]. As the importance of cloud computing is growing bigger and bigger, there are many researches are beginning. It is important to simulate the presentation of cloud system. However, there are numerous factors of a cloud infrastructure such as a hardware, software and services. Therefore, it is hard to quantify the presentation of cloud system.

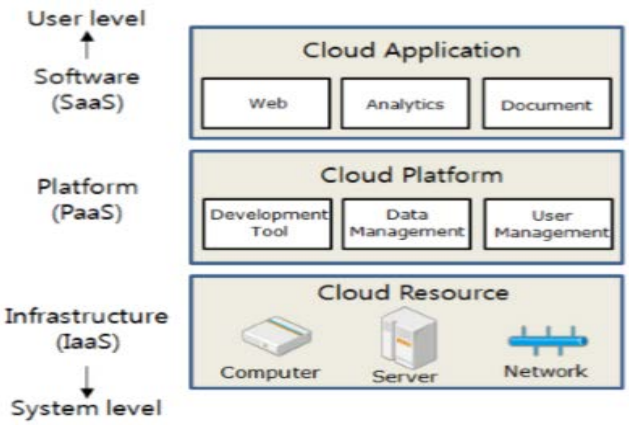

Figure 1: Services in cloud computing

As we enter into the post-petascale era, the requirements of data processing and computation are growing exponentially. To meet this requirement, researchers have moved from serial execution platforms to high-performance computing (HPC) platforms, such as multi-core processors, FPGAs and GPUs etc. GPUs, in particular, have been widely used for HPC applications due to their extremely high computational powers, and a large fraction of supercomputers in Top 500 list use GPU to achieve unprecedented computational power in cloud. Thus, GPUs have become integral part of today's mainstream computing systems.

II. RELATED WORK:

2.1. GPU Terminology and Sources of Power Consumption

Recently, several researchers have proposed models and tools for measurement and estimation of GPU power consumption [Hong and Kim 2010;Ramani et al. 2007; Nagasaka et al. 2010; Sheaffer et al. 2005a; Zhang et al. 2011; Jiao et al. 2010; Zhang et al. 2011; Chen et al. 2011; Suda and Ren 2009; Enos et al. 2010; Wang and Ranganathan 2011; Ren 2011; Ren et al. 2012; Luo and Suda 2011; Pool et al. 2010; Stolz et al. 2010; Li et al. 2011; Wang and Chen 2012; Collange et al. 2009; Wang et al. 2010; Vialle et al. 2011; Kasichayanula et al. 2012]. These models provide insights into the working of GPUs and relative contribution of different components in the total power consumption. In what follows, we briefly review the GPU architecture, terminology and sources of

power consumption, as relevant for this paper and refer the reader to above mentioned works for more details.

A GPU has several streaming multiprocessors, each of which has multiple cores. For example, NVIDIA GeForce GTX 590 has dual GPUs; where each GPU has 16 streaming multiprocessors (SMs); each of these SMs have 32 cores; for a total of 512 cores in each GPU and 1024 cores in the overall GTX 590 graphics card [GeForce GTX 590 2013]. The cores of a typical GPU are composed of ALUs, thread-schedulers, load/store units, scratchpad memory, register file and caches etc. A GPU is designed for stream or throughput computing, which has little data reuse and hence, a GPU has much smaller sized cache (for example 16KB L1 and 256KB L2 [Wong et al. 2010]) than a typical CPU. The GPU is used as a co-processor with a CPU and in such cases, GPU is referred to as the 'device' and the CPU as the 'host'. A GPU has its own device memory of a few GBs (gigabytes), and it is connected to the host through a PCI-Express (PCIe) bus. A GPU is programmed as a sequence of kernels. The code is executed in groups of 32 threads, called a warp. CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) are widely-used interfaces for programming GPUs.

The power consumption of GPU can be divided into two parts, namely leakage power and dynamic power. The dynamic power is a function of operating temperature and circuit technology. Leakage power is consumed when GPU is powered, even if there are no runtime activities. The dynamic power arises from switching of transistors and is determined by the runtime activities. Different components such as SMs and memories (e.g local, global, shared) etc. contribute to this power consumption.

2.2. Need for Improving Energy Efficiency of GPUs

GPU power management is extremely important for the following reasons.

2.2.1. Addressing Inefficient Resource Usage: To meet the worst-case performance requirements, the chip designers need to over-provision the computing resources of GPUs; however, on average, the utilization of these resources remains low. Also, in several applications, memory bandwidth of GPUs acts as a performance-bottleneck [Hong and Kim 2010; Daga et al. 2011; Cebrian et al. 2012; Spafford et al. 2012], due to which the cores are not fully utilized which leads to energy inefficiency. Further, unlike massively parallel applications, regular parallel applications do not scale well beyond a certain number of cores and hence, a large amount of power is wasted in idle cores or in synchronization. Finally, GPUs are increasingly being used in cloud infrastructure and data centers [Amazon EC2 2013], which experience highly varying usage patterns. Thus, dynamic power management techniques can offset these sources of inefficiencies by using runtime adaption.

2.2.2. Ensuring Reliability: Large power consumption has significant effect on the reliability of the computing systems. A 15 degree Celsius rise in temperature increases the component failure rates by up to a factor of two [Anderson et al. 2003]. The device failures may lead to system malfunction and as GPUs become increasingly employed in supercomputers and business services, system malfunction may have serious economic impact. For example, the service cost of merely one hour of downtime in brokerage operations and credit card authorization can be $6,450,000 and $2,600,000, respectively [Feng 2003]. Thus, since the performance-requirements grow at much faster pace than the effectiveness of cooling solutions, power management techniques are extremely important to ensure longevity and reliability.

2.2.3. Providing Economic Gains: For every watt of power dissipated in the computing equipment, an additional 0.5 to 1W of power is consumed by the cooling system itself [Patel et al. 2003], and with increasing ratio of cooling power to computing power, compaction of devices is inhibited, which results in increased operation costs. Due to these trends, in recent years, the energy cost of high-performance computing clusters has been estimated to contribute more than the hardware acquisition cost of IT equipment itself [Bianchini and Rajamony 2004; Mittal 2012].

2.2.4. Enabling Performance Scaling: The power challenges are expected to present most severe obstacle to performance scaling and it has been shown that thermal and leakage power constraints may disallow simultaneously using all the cores of a massively parallel processor [Esmaeilzadeh et al. 2013]. Large power consumption may necessitate complex cooling solutions (e.g. liquid cooling) which may increase chip complexity and offset the benefits of performance boost obtained by using GPUs.

III. TECHNIQUES FOR IMPROVING GPU ENERGY EFFICIENCY IN CLOUD COMPUTING

In this section, we discuss techniques for improving GPU energy efficiency.

3.1. Overview

For the purpose of this study, we classify the techniques into the following categories.

(1) DVFS (dynamic voltage/frequency scaling) based techniques

(2) CPU-GPU workload division based techniques

(3) Architectural techniques for saving energy in specific GPU components, such as caches

(4) Techniques which exploit workload-variation to dynamically allocate resources

(5) Application-specific and programming-level techniques for power analysis and management

We now discuss these techniques in detail. As seen through the above classification, several techniques can be classified in more than one groups. For sake of clarity, we discuss them in one group only.

3.2. DVFS Based Techniques

Dynamic voltage and frequency scaling (DVFS) is a well-known power-management technique which works by dynamically adjusting the clock frequency of a processor to allow a corresponding reduction in the supply voltage to achieve power saving [Rabaey et al. 2002]. By intelligently reducing the frequency, the voltage at which the circuit needs to be operated for stable operation can also be reduced, leading to power saving. However, since the reduction in frequency also harms the performance, the scaling of voltage/frequency needs to be carefully performed. Also note that in some of the works discussed below, the frequency scaling is actually applied to CPU; however, we still include these works since the power saving is achieved in the entire system and power management of CPU is done while taking into account the properties of GPU.

Nam et al. [2007] propose a low-power GPU for hand-held devices. The proposed GPU uses logarithmic arithmetic to optimize area and power consumption. The use of logarithmic arithmetic leads to some computation error, however, due to the small screen of the hand-held devices, the error can be tolerated. They divide the chip into three power domains, viz. vertex shadier, rendering engine and RISC processor, and DVFS is individually applied to each of the three domains. The power management unit decides the supply voltage and frequency of each domain based on its workload for saving power while maintaining the desired performance level.

3.3. CPU-GPU Work Division to to Improve Energy Efficiency

Researchers have shown that different ratios of work-division between CPUs and GPUs may lead to different performance and energy efficiency levels [Ma et al. 2012; Luk et al. 2009]. Based on this observation, several techniques have been proposed which dynamically choose between CPU and GPU as a platform of execution of a kernel based on the expected energy efficiency on those platforms.

3.4. Saving Energy in GPU components

Several techniques make architecture-level changes to GPUs to optimize the energy spent in individual components of the GPU. These techniques utilize the specific usage pattern of GPU components to make runtime adaptation for saving energy. Gebhart et al. [2011] present a technique for saving energy in core datapath of GPU. Since GPUs employ a large number of threads, storing the register context of these threads requires a large amount of on-chip storage. Also, the thread-scheduler in GPU needs to select a thread to execute from a large number of threads. For these reasons, accessing large register files and scheduling among a large number of threads consumes substantial amount of energy. To address this, Gebhart et al. present two improvements. First, a small storage structure is added to register files which acts like a cache and captures the working set of registers to reduce energy consumption. Second, the threads are logically divided into two types, namely, active threads (which are currently issuing instructions or waiting on relatively short latency operations), and pending threads (which are waiting on long memory latencies). Thus, in any cycle, the scheduler needs to consider only the active threads which are much smaller in number. This leads to significant energy savings.

3.5. Dynamic Resource Allocation Based Techniques

It is well-known that there exists large intra-application and inter-application variation in the resource requirements of different applications. In fact, several real-world applications rarely utilize all the computational capabilities of GPU. Thus, significant amount of energy saving can be achieved by dynamically adapting the components which exhibit low utilization levels.

Hong and Kim [2010] propose an integrated power and performance prediction system to save energy in GPUs. For a given GPU kernel, their method predicts both performance and power; and then uses these predictions to choose the optimal number of cores which can lead to highest performance per watt value. Based on this, only desired number of cores can be activated, while the remaining cores can be turned-off using power-gating. Note that power-gating is a circuit-level scheme to remove leakage by shutting-off the supply voltage to unused circuits.

3.6. Application-specific and programming-level techniques

It has been observed that source-code level transformations and application-specific optimizations can significantly improve the resource-utilization, performance and energy efficiency of GPUs. Thus, by performing manually or automatically optimizing GPU implementation and addressing performance bottlenecks, large energy savings can be obtained. Wang et al. [2010] propose a method for

saving energy in GPU using kernel-fusion. Kernel fusion combines the computation of two kernels into a single thread. Thus, it leads to balancing the demand of hardware resources, which improves utilization of resources and thus, improves the energy efficiency. The authors formulate the task of kernel-fusion as a dynamic programming problem, which can be solved using conventional tools.

IV. TOOLS RELATED TO GPU ENERGY EFFICIENCY IN CLOUD:

There are various cloud computing tool can be used for implement scheduling task.

A. CLOUDMIGXPRESS:

CloudMIG Xpress addresses those types of challenges and supports method provide for the evaluation and preparation phases to move around software techniques to PaaS or IaaS-based clouds scenario. It supplies from a rationally model and is make to provide research in cloud immigration. The basic characteristics are as follows:

- Extract code prototypes from jdk-based software
- Reproduce many cloud deployment options
- Compare the trade-offs
- Evaluate future values, response times, and SLA violations
- Model the current technique deployment
- Create artificial workload profiles
- Model cloud scenarios with the help of cloud profiles
- Model cloud atmosphere constraints
- Perform a static analysis to detect cloud violations
- Compare the suitability of different cloud profiles
- Graph-based visualization of searched cloud violations

B. CLOUDSIM:

CloudSim is an extensible simulation model that provides prototyping and imitation of Cloud computing technique and application provisioning atmosphere. The CloudSim simulator provides both system and activities modeling of clouds mechanism like as information centers, virtual machines and resource provisioning rules. It experiments generic application provisioning methods that can be elaborated with simplicity and limited attempt. Currently, it provides prototyping and simulation of cloud atmosphere including of both unit and inter-networked cloud system. Moreover, it shows typical interfaces for experimenting rules and provisioning approaches for allocation of virtual machines belongs to inter-networked cloud systems. Many researchers from organizations like as HP laboratory in US are using CloudSim in their examination on cloud supply provisioning and energy well-organized organization of information center possessions. The convenience of CloudSim is introduced by a case study consisting dynamic condition of application services in the mixed federated clouds atmosphere. The conclusions of this case study prove that the cloud computing scenario efficiently increases the application QoS requirements under swinging supply and service insist patterns.

C. ICANCLOUD:

Basically iCanCloud is a simulation place aimed to prototype and simulates cloud computing approaches, which is objected to those programmers who deal nearly with those types of systems. The main objective of iCanCloud is to assume the trade-offs between cost and effective performance of a given set of applications performed in a specific hardware and then support to programmers useful data about such values. Therefore, iCanCloud can be used by a wide range of programmers and users, from general active users to developers of more distributed applications. The most desirable characteristics of the iCanCloud simulation place consists the following:

- Both existing and non-existing cloud architectures can be prototyped and simulated.
- A more flexible cloud hypervisor function supports an easy technique for integrating and testing both new and previous cloud brokering rules.
- Custom VMs can be used to fast simulate uni-core/multi-core systems.
- iCanCloud supports a wide area of configurations for repository systems which consist prototypes for local storage systems, isolated storage systems like NFS and parallel repository systems like parallel systems and RAID systems.

Some other cloud computing tool is as follows:

(1) SIMCLOUD:

(2) REALCLOUDSIM

(3) SIMCLOUD:

(4) VIMCLOUD:

(5) APACHE-ANT:

V. FUTURE RESEARCH TRENDS AND CONCLUSION

We believe that in the near future, the challenges of GPU power consumption would need to be simultaneously addressed at different levels at the same time. At the chip-design level, researchers are aiming to develop energy-

efficient throughput cores and memory design to exploit instruction-level, data-level and fine-grained task-level parallelism. At the architecture level, CPU and GPU need to be efficiently integrated on the same chip with unified memory architecture [Foley et al. 2012; Yuffe et al. 2011]. This will address the memory bandwidth bottleneck and also avoid the replicated chip infrastructure and the need of managing separate memory spaces. At the programming level, per-application tuning is inevitable to achieve a fine balance between demands of the application and the resources of the GPU. Finally, at the system level, policies for intelligent scheduling and work-division between CPU and GPU are required, so that their individual competencies are integrated and they complement each other.

The 3D die stacking holds the promise of mitigating memory bandwidth bottleneck in GPUs, as it enables use of shorter, high-bandwidth and power-efficient global interconnect and provides denser form factor. 3D stacking also enables integration of heterogeneous technologies, which allows use of non-volatile memory (NVM), such as phase change RAM (PCM) and spin transfer torque RAM (STT-RAM) in the design of GPU memory [Mittal 2013]. NVMs consume negligible leakage power and provide higher density than SRAM and DRAM, however, their write latency and energy are significantly higher than those of SRAM and DRAM. It is expected that leveraging the benefits of 3D stacking and NVM would be a major step in improving the energy efficiency of GPUs and it would require novel solutions at device, architecture and system level.

REFERENCES

[1] S. Jeschke, D. Cline, and P. Wonka, “A GPU Laplacian solver for diffusion curves and Poisson image editing,” in ACM Transactions on Graphics (TOG), vol. 28, no. 5. ACM, 2009, p. 116.

[2] G. Barozzi, C. Bussi, and M. Corticelli, “A fast cartesian scheme for unsteady heat diffusion on irregular domains,” Numerical Heat Transfer, Part B: Fundamentals, vol. 46, no. 1, pp. 59–77, 2004.

[3] P. Bailey, J. Myre, S. Walsh, D. Lilja, and M. Saar, “Accelerating lattice Boltzmann fluid flow simulations using graphics processors,” in Parallel Processing, 2009. ICPP ’09. International Conference on, Sept 2009,pp. 550–557.

[4] R. Abdelkhalek, H. Calandra, O. Coulaud, J. Roman, and G. Latu, “Fast seismic modeling and reverse time migration on a GPU cluster,” in High Performance Computing Simulation, 2009. HPCS ’09. International Conference on, June 2009, pp. 36–43.

[5] E. Elsen, P. LeGresley, and E. Darve, “Large calculation of the flow over a hypersonic vehicle using a GPU,” Journal of Computational Physics, vol. 227, no. 24, pp. 10 148 – 10 161, 2008.

[6] S. Venkatasubramanian and R. W. Vuduc, “Tuned and wildly asynchronous stencil kernels for hybrid CPU/GPU systems,” in Proceedings of the 23rd International Conference on Supercomputing, ser. ICS ’09. New York, NY, USA: ACM, 2009, pp. 244–255.

[7] M. Griebel and P. Zaspel, “A multi-GPU accelerated solver for the three-dimensional two-phase incompressible navier-stokes equations,” Computer Science - Research and Development, vol. 25, no. 1-2, pp. 65–73, 2010.

[8] S. Donath, C. Feichtinger, T. Pohl, J. G¨otz, and U. R¨ude, “A parallel free surface lattice Boltzmann method for large-scale applications,” Parallel Computational Fluid Dynamics: Recent Advances and Future Directions, p. 318, 2010.

[9] A. Danalis, K.-Y. Kim, L. Pollock, and M. Swany, “Transformations to parallel codes for communication-computation overlap,” in Proceedings of the 2005 ACM/IEEE conference on Supercomputing. IEEE Computer Society, 2005, p. 58.

[10] N. Maruyama, T. Nomura, K. Sato, and S. Matsuoka, “Physis: an implicitly parallel programming model for stencil computations on large-scale GPU-accelerated supercomputers,” in High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. IEEE, 2011, pp. 1–12.

[11] M. Sourouri, S. B. Baden, and X. Cai, “Panda: A compiler framework for concurrent cpu + gpu execution of 3d stencil computations on gpu-accelerated supercomputers,” International Journal of Parallel Programming, pp. 1–19, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10766-016-0454-1