Noname manuscript No.
(will be inserted by the editor)

Energy-aware Design Space Exploration for GPGPUs

Pascal Libuschewski · Dominic Siedhoff · Frank Weichert

The final publication is available at http://link.springer.com/article/10.1007%2Fs00450-013-0237-5

Abstract This work presents a novel approach for automatically determining the most power- or energy-efficient Graphics Processing Units (GPUs) with respect to given parallel computation problems.

Keywords Simulation · Deployment of mechanisms · Design space exploration · GPGPU · Green computing

1 Introduction

In the last years Graphics Processing Units (GPUs) have become increasingly important in the field of high performance computing (HPC). This is due to the growing possibilities of general-purpose computing on graphics processing units (GPGPU). With GPUs getting more and more powerful and the number of transistors in GPUs exponentially rising in accordance with Moore's Law [10], reducing their energy consumption becomes increasingly crucial [6,9].

If a GPU for a given GPGPU task is to be selected, it would be beneficial to know ahead, which type of GPU is the most power- or energy-efficient for the task to be solved. This knowledge is not only advantageous in high performance computing, but also in conserving energy for mobile computing and for green computing.

In this paper a novel approach is presented, which automatically determines the most power- or energy-efficient GPU for a given GPGPU task. It can optimize the power- and energy-consumption of OpenCL (Open Computing Language) [5] and CUDA (Compute Unified Device Architecture) [11] code with respect to diversely configurable simulated GPUs. The main purpose of the presented approach is to identify the most power- or energy-efficient GPU for a given task by means of simulation, that is without the need to switch hardware. However, it is not restricted to the examination of existing GPUs: It can be used to conduct a design space exploration for future GPUs and to automatically find optimal hardware parameters like the number of streaming multiprocessors, core clock and DRAM clock. Furthermore, it can be generalized to objective functions other than power or energy, for example to identify the fastest GPU for a given task.

2 Related Work

The state of the art in the context of energy-aware computing can be divided into two main methodologies, differing in their approach to measuring energy consumption: On the one hand, energy consumption can be mea-

P. Libuschewski
Technische Universität Dortmund
Lehrstuhl Informatik VII
D-44221 Dortmund
Otto-Hahn-Str. 16
Tel.: +49-231-7556125
Fax: +49-231-7556321
E-mail: pascal.libuschewski@tu-dortmund.de

D. Siedhoff
Technische Universität Dortmund
Lehrstuhl Informatik VII
D-44221 Dortmund
Otto-Hahn-Str. 16
Tel.: +49-231-7556125
Fax: +49-231-7556321
E-mail: dominic.siedhoff@tu-dortmund.de

F. Weichert
Technische Universität Dortmund
Lehrstuhl Informatik VII
D-44221 Dortmund
Otto-Hahn-Str. 16
Tel.: +49-231-7556122
Fax: +49-231-7556321
E-mail: frank.weichert@tu-dortmund.de

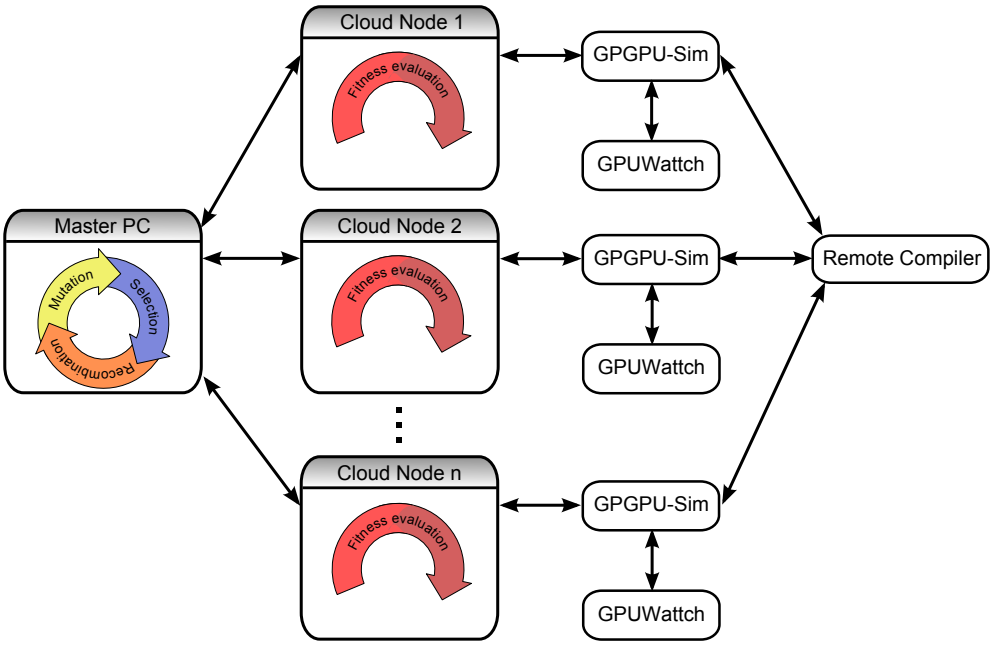

Fig. 1 Scheme of the evolutionary process

sured using hardware, restricting this type of approach to existing GPUs. On the other hand, hardware may be simulated, while an energy-model provides estimated energy consumption. A survey on both methodologies can be found in [1]. Cebrian et al. [3] use hardware to determine power consumption. This gives accurate results but constrains the number of GPUs that can be examined. In [2] the GPGPU-Sim simulator is used to examine GPUs with different design parameters, but this was not done in the context of energy consumption.

A comparison of the energy consumption of a GPU in contrast to a CPU-only setting has been done for example in [8] and [13]. The latter paper concludes that the use of GPUs can result in less energy consumption for some problems and that there is "a huge potential of research in the field of energy-aware high performance computing with GPUs".

In [12] a framework to estimate the power consumption of non-existing GPUs with respect to a finite set of typical GPU access patterns has been presented. In contrast to the cited papers, the presented work explores the GPU design space with respect to a given software. The design space encompasses but is not limited to existing GPUs. It is traversed by minimizing power- or energy-consumption as the objective function. To this end, a simulator provides estimates of these quantities. The final result is a vector containing the hardware parameters of most efficient platform for the task at hand. The task is defined in terms of its OpenCL [5] or CUDA [11] code.

The paper is structured as follows: Section 3 presents the methods for design space exploration of GPUs. Section 4 describes the experimental setup and the results. Finally, sections 5 and 6 provide discussion and future work.

3 Design Space Exploration of GPUs

This section describes a novel method for power- or energy-aware design space exploration of GPUs with respect to given GPGPU tasks. For a selectable software S, the best GPU hardware is to be found using an evolutionary algorithm to solve the following optimization problem O:

$$O = \underbrace{\text{minimize}}_{(p_1 \dots p_n) \in P^n} f(S, (p_1 \dots p_n)). \tag{1}$$

Here f is a function estimating the average power- or energy-consumption for the given GPGPU task S, when S is executed on a hardware platform with parameters as given in the parameter vector $(p_1 \dots p_n)$. Hence the parameter vector describes the examined GPU platform: Each p_i describes one hardware feature, e.g. the number of cores or the clock speed. P^n is the overall configuration set, i.e. it represents the set of all possible configurations, defined by all possible combinations of parameters p_i:

$$P^n = \bigcup_{a=1}^{m} \mathbf{p}_a \text{ with } \mathbf{p}_a = (p_1 \dots p_n), \tag{2}$$

Table 1 Examined architectures

Architecture	SMs	Core Clock Speed (MHz)	DRAM Clock Speed (MHz)	Number of Registers	|Search Space|
GF-108	1-2	700-810	1600-1800	16k-32k	1296
GF-106	3-4	590-790	1800-4000	16k-32k	21168
GF-104	6-7	650-675	3400-3600	32k-64k	350
GF-100	11-15	610-780	3200-4000	32k-64k	28800

where all p_i are valid values for the respective parameter.

As the size m of the overall configuration set P^n grows multiplicatively with in the number of examined parameter values and because only reasonable GPUs should be examined, P^n is confined and divided into different classes, C_j^n. Each class C_j^n may correspond to a single GPU or to a set of GPUs or to an entire GPU architecture. For example a class may consist of GPUs with four to six streaming multiprocessor cores and a RAM size of 512 to 2048 megabytes. A class C_j^n is defined by a minimum possible value $p_{i,j}^{\min}$, a maximum possible value $p_{i,j}^{\max}$ and a sample spacing $p_{i,j}^{\text{samp}}$ for each of the n parameters p_i in equation (2). Hence within a class C_j^n, valid parameters $p_{i,j}$ fulfill the following condition:

$$p_{i,j} = p_{i,j}^{\min} + k \cdot p_{i,j}^{\text{samp}} \leq p_{i,j}^{\max}, k \in \mathbb{N}_0 \tag{3}$$

In other words: A single parameter $p_{i,j}$ is valid in class C_j^n, if its value is between the corresponding class limits $p_{i,j}^{\min}$ and $p_{i,j}^{\max}$ and its distance to $p_{i,j}^{\min}$ is a positive integer multiple of the sample spacing $p_{i,j}^{\text{samp}}$. If all parameters p_i in a parameter vector $\mathbf{p}_a$ fulfill the condition in equation (3), the parameter vector is valid and belongs to class C_j^n. Finally, the confined configuration space C^n in which the search is carried out is defined as the union of all k classes C_j^n:

$$C^n = \bigcup_{j=1}^{k} C_j^n. \tag{4}$$

This limitation of the search space to a set of classes is very flexible and can easily be adapted to the research needs. For example only existing GPUs without modification of the clock speeds can be examined, or the search space can be extended to also examine GPUs that do not yet exist.

To summarize the process: For a given software S an optimization problem is solved to find the most power- or energy-efficient parameter vector $\mathbf{p}_e \in C^n$. This vector consists of n parameters which represent one GPU in the confined configuration space C^n. The confinement is obtained from the overall configuration space P^n by limiting the latter to k different classes C_j^n of sampled parameter intervals. With the confinement, the search space can be chosen according to the research goal. Finally, the objective function f is optimized with respect to parameters $(p_1 \dots p_n)$, which involves evaluating f by means of estimating the power- or energy-consumption of the program through simulation.

4 Results

To verify that the proposed method generalizes to different types of problems, the evaluation has been conducted for a set of benchmark programs and a set of different GPU architectures. The experimental setup is described in section 4.1, followed by the evaluation in section 4.2.

4.1 Experimental Setup

For the calculation of the objective function f in equation (1), an extended version of the research simulator GPGPU-Sim [2] is used to simulate the GPUs with cycle accuracy. In combination with the power-model GPUWattch [6], which is based on McPAT [7], the power consumed by the GPU can be estimated. However, other methods than GPGPU-Sim and GPUWattch can also be integrated into the presented framework. Furthermore, objective functions other than power can be used, e.g. energy consumption or the number of cycles.

To solve the optimization problem O in equation (1), a heavily extended version of ECJ (Java-based Evolutionary Computation Research System) [**?**] is used. The extensions to ECJ encompass limiting the search space, thus preventing non-reasonable GPUs to be evaluated and distributing the workload among heterogeneous PCs within a compute cloud. The limitation to classes was realized by mapping the parameter vectors $\mathbf{p}_a$ of equation (2) to the nearest neighbor in the set C^n of equation (4).

For evaluation purposes, a subset of the OpenCL programs in the benchmark set Rodinia [4] was used as the parameter S in the equation (1). A physics simulation named HotSpot, two common data mining algorithms (k-Means and k-Nearest Neighbors (k-NN)) and the bioinformatics algorithm by Needleman-Wunsch were selected. The selection was made to demonstrate the

Table 2 Evaluation of the average power consumption

Benchmark	Avg. Power Consumption	SMs	Core Clock	DRAM Clock	Registers	Architecture
HotSpot	11.5 Watt	1	700 MHz	1800 MHz	18k	GF-108
k-Means	7.5 Watt	3	610 MHz	1800 MHz	32k	GF-106
k-Nearest Neighbors	13.4 Watt	1	700 MHz	1600 MHz	32k	GF-106
Needleman-Wunsch	41.3 Watt	12	690 MHz	3200 MHz	32k	GF-100

generality of the method, there is no claim for completeness. For each algorithm the proposed method was used to optimize the estimated average power consumption. The resulting estimates are normalized to the run of a single OpenCL kernel program.

In figure 1 the overall process can be seen. One master PC is used to breed new individuals in an evolutionary process. A variable number of heterogeneous computer systems constitute a compute cloud to evaluate the fitness of the individuals. The fitness evaluations are independent of each other, and each one uses GPGPU-Sim and GPUWattch for estimating the consumed power of one individual in the population. One remote computer is used to provide the necessary PTX (Parallel Thread Execution) input for GPGPU-Sim, thus ensuring optimization of the same PTX code on every node, even in the presence of differing driver versions among the nodes. If all individuals in one generation have been evaluated on the cloud nodes, the master PC recombines and mutates the individuals and makes a selection to form the next generation of individuals.

For the master PC a PC with four Intel(R) Xeon(R) E5-2690 (each with 8 cores) and 64 GB RAM operating on Windows 7 was used. This machine was running the evolution and was also used to contribute 32 nodes to the compute cloud for the fitness evaluation. This was done by running an Ubuntu Linux operating system in a virtual machine.

To show that the system can utilize a number of heterogeneous systems, an Intel(R) Core(TM) Q9550 running Windows 7 and an Intel(R) Atom(TM) D510 with Ubuntu Linux were used to contribute six more nodes to the cloud. As the Intel Atom PC has a GPU driver capable of compiling OpenCL (Open Computing Language) code to PTX (Parallel Thread Execution) code, it was used as the remote compiler for all the other nodes in the cloud.

Summing up the experimental setup, one 32 core PC was used as the master, while at the same time providing 32 nodes in the compute cloud. A second PC added 4 more nodes to the cloud and a third PC delivers 2 more nodes, while simultaneously serving as a remote OpenCL compiler. It is not necessary for the master PC to also provide nodes to the cloud but as the breeding and evolution of the individuals are sequential steps, this can be done without losing performance.

4.2 Evaluation

To validate the presented method, the search space was chosen to consist of four different significant parameters p_1 to p_4 in equation (2), as can be seen in Table 1. The chosen parameters were the number of streaming multiprocessors (p_1), the core (p_2) and DRAM clock speed (p_3) and the number of registers (p_4). The streaming multiprocessor cores (SMs) form a group of many single compute units. Each compute unit can host a lightweight GPU thread and threads within one SM can share one fast local memory which can not be accessed from another SM. The more SMs there are in a GPU, the more thread groups can be run in parallel. The core clock speed and the DRAM clock speed define how fast the computations can be done, respectively how fast the access to the GPU main memory is. The number of registers can be a limiting factor for the evaluated program S if it uses more than 32k of registers. Other parameters like the cache size or the bus width can also be taken into account but were not considered in this preliminary study.

Four different NVidia architectures were used to define the classes C_1^4–C_4^4 in equation (4). The architectures are GF-100, GF-104, GF-106 and GF-108, spanning a wide variety of GPUs. Their major difference is the number of streaming multiprocessor cores (SMs). Each architecture can be uniquely identified by the number of SMs. The core clock speed is typically higher on the GPUs with fewer SMs but this is not always the case. The same applies for the DRAM clock speed. The number of registers is 16384 to 32768 for the GF-106 and GF-108 and 32768 to 65536 for the GF-104 and GF-100.

The sample spacing $p_{i,j}^{\text{samp}}$ in equation (3) was chosen to be 10 MHz for the core speed. The DDR DRAM speed was sampled with a 40 MHz spacing, and the number of registers was sampled at spacing 2048. This results in an overall search space of 51614 possible GPUs as can be seen in the last column of Table 1.

Table 3 Simulated power consumption of fastest GPU compared to the optimized results

Benchmark	Avg. Power Cons. GTX 480	Avg. Power Cons. Proposed	Power Saved
HotSpot	49.7 Watt	11.5 Watt	77%
k-Means	26.0 Watt	7.5 Watt	71%
k-Nearest Neighbors	91.6 Watt	13.4 Watt	85%
Needleman-Wunsch	50.1 Watt	41.3 Watt	18%

The evaluation results for the different benchmark programs are shown in Table 2. For every benchmark, the table shows the average power consumption in Watt, the number of used streaming multiprocessor cores, the core and DRAM clock speed in MHz and the number of used registers.

For the HotSpot benchmark, a GF-108 architecture was found to be optimal, resulting in an average power consumption of 11.5 Watt. The architecture uses only one streaming multiprocessor and runs at 700 MHz core clock speed and 1800 MHz DRAM clock speed. With the k-Means benchmark, a GF-106 architecture results in optimized power consumption of 7.5 Watt with three streaming multiprocessor cores. For k-Nearest Neighbors, again a GF-106 architecture was identified with an average power consumption of 13.4 Watt. The Needleman-Wunsch algorithm performed most power-efficiently on the GF-100 architecture, using 41.3 Watt with twelve cores.

As a result, the overall variation of cores is from one to twelve. The core clock speed does not vary strongly, ranging from 610 MHz up to 700 MHz. The DRAM clock speed has more variation, spanning from 1600 MHz up to 3200 MHz. The number of registers does not seem to have much impact on power consumption for the given programs as it varied in a wide range during simulations, without affecting the result.

Table 3 shows a comparison of the simulated consumed power for the GTX 480 against the proposed method. The HotSpot benchmark consumes 23 percent of the power if it is run on a GF-108 architecture, instead of a GTX 480 GPU. The k-Means algorithm is most efficient on a GF-106 architecture, with power consumption reduced to 29 percent. The optimized parameters correspond to a GTS 450 GPU. The k-Nearest Neighbors algorithm consumes 15 percent of the power in comparison to running it on a GTX 480. For the Needleman-Wunsch algorithm a GF-100 architecture performs best, with a GTX 465 or GTX 470 as the most power-efficient existing GPU. In this case, power consumption was reduced to 82 percent.

5 Discussion

In the face of increasing costs for energy and the need for green solutions it has been shown that the proposed framework can be used to identify power- or energy-efficient GPUs for various given problems. The search space has been reasonably confined to reduce the number of evaluations of the objective function in evolutionary optimization. Depending on the research goal, the reduced search space can be used to search for existing or non-existing GPUs.

The attained power-savings vary from 18 percent up to 85 percent when comparing the proposed method to using a GTX 480 GPU. It should be made clear that the power savings are to be taken as a proof of concept for the methodology. The validity of these results depends on the validity of the underlying power-model, which has been evaluated in [7] for CPUs and in [6] for GPUs. As the presented method does not depend on one specific power-model, the model can be replaced. Using a performance- or energy-model instead of a power-model is also possible and results in the most time- or energy-efficient GPU instead of the most power-efficient one. To determine the fastest GPU, the number of cycles provided by GPGPU-Sim can be used as the objective. For the most energy-efficient GPU, the average power consumption calculated by GPUWattch has to be multiplied by the number of cycles and normalized to one second depending on the known clock speed of the evaluated GPU.

6 Future Work

The presented approach is the basis for further research: If the most efficient results are wanted, GPUs should not be optimized for fixed code, but the software parameters should be taken into account simultaneously. This points towards hardware/software co-design and an optimization process that makes use of this potential. One way to do this, is to include the size of the work groups that are formed by cooperating threads on the GPU as a further parameter to be optimized. This size has a major impact on speed and energy consumption because it influences the utilization of the streaming multiprocessors, the local memory and the speed in

which data can be transferred from and to the DRAM. It thus has to be chosen carefully to match the underlying GPU architecture.

Another vital point to be considered is the runtime of the parallel programs, especially if deadlines have to be met, like in real-time applications. Further research can take runtime into account as either a constraint to be met, or as a further objective function in minimization, naturally leading to multi-objective optimization. Another aspect to consider is finding the optimal GPU for a set of n different problems to be executed on the same high performance system.

Acknowledgements Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project B2. URL: `http://sfb876.tu-dortmund.de`

References

1. Ahmad, I., Ranka, S.: Handbook of Energy-Aware and Green Computing - Two Volume Set, 1st edn. Chapman & Hall/CRC (2012)
2. Bakhoda, A., Yuan, G., Fung, W., Wong, H., Aamodt, T.: Analyzing cuda workloads using a detailed gpu simulator. In: Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on, pp. 163–174 (2009). DOI 10.1109/ISPASS.2009.4919648
3. Cebrian, J., Guerrero, G., Garcia, J.: Energy efficiency analysis of gpus. In: Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International, pp. 1014–1022 (2012). DOI 10.1109/IPDPSW.2012.124
4. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J., Lee, S.H., Skadron, K.: Rodinia: A benchmark suite for heterogeneous computing. In: Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on, pp. 44–54 (2009). DOI 10.1109/IISWC.2009.5306797
5. Khronos Group: OpenCL Specification (2013). URL: `http://www.khronos.org/registry/cl/`
6. Leng, J., Hetherington, T., ElTantawy, A., Gilani, S., Kim, N.S., Aamodt, T.M., Reddi, V.J.: Gpuwattch: Enabling energy optimizations in gpgpus. International Symposium on Computer Architecture (2013)
7. Li, S., Ahn, J.H., Strong, R., Brockman, J., Tullsen, D., Jouppi, N.: Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In: Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on, pp. 469–480 (2009)
8. Libuschewski, P., Siedhoff, D., Timm, C., Gelenberg, A., Weichert, F.: Fuzzy-enhanced, real-time capable detection of biological viruses using a portable biosensor. In: Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSIGNALS) (2013). Publication
9. Luke, S.: The ECJ Owner's Manual (2013)
10. McIntosh-Smith, S., Wilson, T., Ibarra, A.A., Crisp, J., Sessions, R.B.: Benchmarking energy efficiency, power costs and carbon emissions on heterogeneous systems. The Computer Journal **55**, 192–205 (2012)
11. Moore, G.: Cramming more components onto integrated circuits. Proceedings of the IEEE **86**(1), 82–85 (1998). DOI 10.1109/JPROC.1998.658762
12. NVIDIA Corporation: CUDA Architecture (2013). URL: `http://www.nvidia.com/object/cuda_home_new.html`
13. Ramani, K., Ibrahim, A., Shimizu, D.: Powerred: A flexible modeling framework for power efficiency exploration in gpus
14. Rofouei, M., Stathopoulos, T., Ryffel, S., Kaiser, W., Sarrafzadeh, M.: Energy-aware high performance computing with graphic processing units. In: In HotPower08: Proc. of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower) 2008 (2008)