Acta Technica *62* No. 1A/2017, 361–370

A parallel algorithm based on CPU/GPU dynamic coordination control and scheduling[1]

Hanyang Jiang[2,3], Danhua Wang[4]

Abstract. CPU/GPU dynamic coordination technology is gradually applied to the development of parallel algorithms, but the current algorithm has some shortcomings in terms of efficiency and power consumption. Therefore, it is necessary to further study the parallel algorithm based on dynamic coordination control and scheduling. Based on the power cap technology, this paper sets up the device frequency and the single computing node of the synchronization system by studying the synchronization system of the computer, controlling DVFS and mapping mode. In order to avoid power failure and load imbalance, this paper has developed a new empirical model of performance and synchronization system, these models make the most of the power of the system to reduce the power consumption to a predetermined level, Good settings can be done.

Key words. Parallel algorithm, CPU, GPU, dynamic coordination.

1. Introduction

The central processing unit is a call to the electronic circuit inside the computer, and shows the logic input and output commands as compared with the basic arithmetic education, and executes the computer program [1]. The computer industry uses this term at least since 1960, where the central processing unit traditionally refers to the processor and, more specifically, the basic elements of the processing unit and the control unit, the computer and the external components, such as the main memory area and the circuit need to be separated. The graphics processing unit is a dedicated electronic circuit that is designed to be quickly manipulated, capable of changing memory-created images, and a display device designed to speed

[1]Funding for research projects of The Education Department of Hunan Province (16C0227) and the Science and Technology Plan Project of Hunan Province (2016TP1020)

[2]Hengyang Normal University, Hengyang, 421002, China

[3]Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang, 421002, China

[4]Hengyang Normal University, Hengyang, 421002, China

up the output buffer [2]. GPU is an embedded system that can be used for mobile phones, personal computers, workstations and game consoles. Modern GPUs are very effective in computer graphics operations and image processing, and their highly parallel architectures enable large blocks of data to be processed in parallel, which is more efficient than using a common algorithm. In the personal computer, GPU can be installed on the video card, or can also be installed on the motherboard, CPU can also be integrated CPU.

On June 30, 1945, before the invention of ENIAC, the mathematician John von Neumann published the first draft of the report, named EDVAC [3]. This is the outline of the computer stored program, in August 1949, EDVAC design eventually completed, you can perform some types of instructions. EDVAC's written program will be stored in high-speed computer memory, rather than in the physical line of the computer. This overcomes the shortcomings of ENIAC, which is a serious flaw in reconfiguring the computer to perform new tasks and takes a lot of time and effort. Improved cache and bypass can maintain the shared memory capacity, to achieve the parallel operation of the thread.

Manchester ran the first program in the computer in 1949, the CPU can be customized to design, play a huge role, and sometimes can be used as part of a single computer [4]. However, the CPU custom design of this method of specific application space is limited, general-purpose processor production quality has been greatly developed. The era of standardized discrete transistors and minicomputers has begun, with the popularity and rapid development of ICs. The chip allows for more complex CPU designs that can be manufactured for nanometer tolerances. Standardized miniaturization of the CPU increases the use of digital devices in modern life, far beyond the application settings, resulting in a lot of specific computers.

2. Materials and methods

In this article, we present an effective power capping technique and a computer synchronization system based on DVFS coordinated mapping technology [5]. Complex performance and system energy consumption is because the frequency of the CPU is different, the frequency of the GPU conforms to the mapping relationship. In exploring the operating space of the parameters, we need to proactively establish a system with limited power that needs to be executed before mapping DVFS settings and tasks. In order to realize the mapping function of orientation and system setting, the performance task is proposed and the new empirical model is identified to verify the technical practicability of heterogeneous system. The fewer configuration files used, these models allow us to get the best parameters for a given power series constraint.

2.1. Power management technology

The capping power is the core technology of the computer power management system. Remove the power pulses of the components and the computer, keep the given nodes, and constrain the computer power consumption [6]. At the level of the

component, there has been an upper limit on the power system and CPU memory that have been proposed. At different levels, limiting the power of personal computing nodes is an effective means of achieving system computing power enhancement, large-scale calculation is very important [7]. Through technology-based power capping technology, we are able to optimize the application's running capacity, which is running on the Internet server's available power capacity.

We use the optimized compiler to mark the database we provide and use it for evaluation and calculation. For the development of mixed applications, we use the parallel code in the benchmark suite, combined with the Rodinia and BLAS libraries [8]. Using the GPU's computing functions and the CPU-side parallel computing code. The hybrid algorithm uses the CPU to provide the results for the user. In the hybrid performance, the data elements only need to be executed when a given task is needed to change the mapping of the GPU. In the experiment, we assume that we can change the percentage of CPU activity and reduce the GPU running memory. It is shown in Fig. 1.

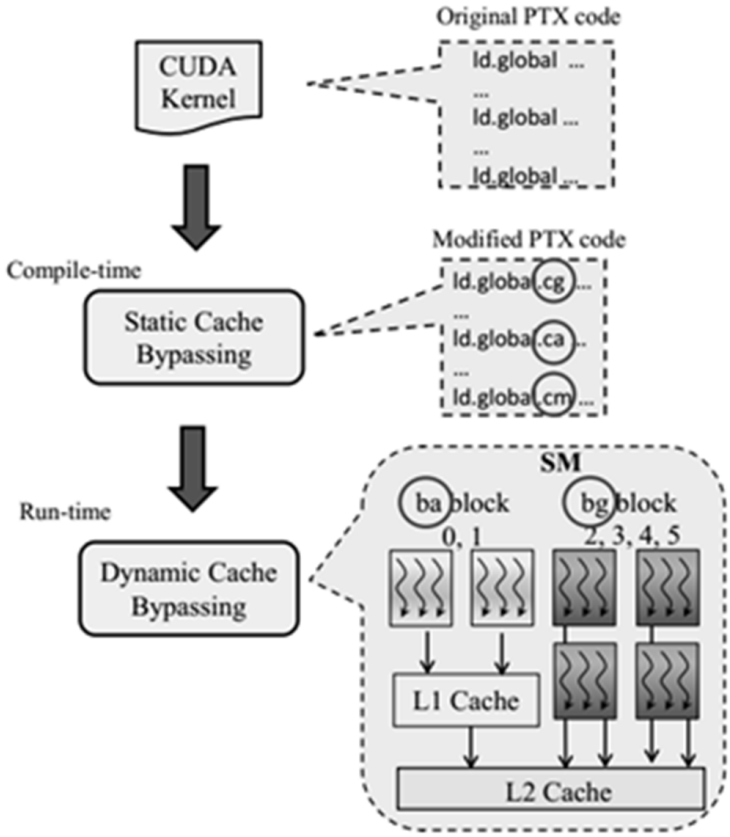

Fig. 1. Static bypassing classifies global loads into three categories using tags ca, cg, and cm

2.2. GPU heterogeneous systems

In this paper, we consider using the limit technique to calculate the power of each node, assuming the structure of the machine [9]. The structure includes a CPU and a GPU. CPU and GPU commercial production has been able to support dynamic voltage and frequency scaling technology, through the management of DVFS, the computer's power consumption can be integrated in the measurement device to install a dedicated hardware counter to read. Provided by the business software interface. Previous research reports indicate that the power displayed by components such as motherboards and hard drives can produce relatively small fluctuations under different workloads. Since CPUs and GPUs are the primary source of nodes, dynamically calculating power fluctuations will be considered the goal of power management.

Estimation of the execution time of the first I kernel K_i hybrid computation, where rcpu and rgpu are not zero, we make the following two assumptions:

- The actual execution time on CPU or GPU is proportional to the number of mapping tasks,

- Global obstacles need to be executed after each parallel kernel; we can estimate the execution time of the I kernel with the following equation

$$K_i = \max\left\{K_i^{\mathrm{c}}(f_{\mathrm{cpu}}, f_{\mathrm{gpu}}) \times \frac{r_{\mathrm{cpu}}}{100}, K_i^{\mathrm{g}}(f_{\mathrm{cpu}}, f_{\mathrm{gpu}}) \times \frac{r_{\mathrm{gpu}}}{100}\right\}. \tag{1}$$

The cache not only reduces access latency and power consumption, it also means that a particular hardware area consists of different parts [10]. For example, the computer's architecture uses a unified cache and shared project memory. The uniform design increases the size of the cache, meaning that the size of the shared memory can be reduced. However, this may affect performance, and the reduction in shared memory volume will reduce the number of active threads [11]. In contrast, the improved cache can bypass the maintenance of shared memory capacity, enhance the parallel performance of the thread. It is shown in Fig. 2.

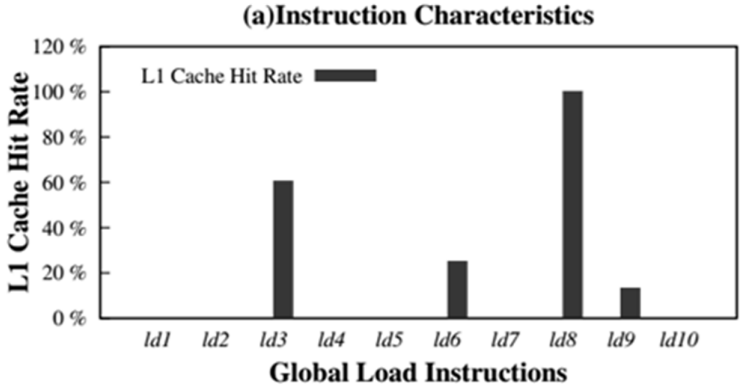

Fig. 2. Motivation for coordinated cache bypassing using SPV application

2.3. Parallel computing

Parallel computing of two CPUs and GPUs can mix data and improve the efficiency and performance of the energy of the synchronous compute nodes [12]. Recently, with the advent of portable GPU programming environments, such as OpenCL and OpenACC, many researchers have proposed a variety of compilation and execution systems to take full advantage of the data parallel processing capabilities. We evaluated the selected applications and BLAS libraries for the Rodinia benchmarks. When the compiler or database is running, connect with the CPU's capabilities. Therefore, we must manually change the mapping mechanism of heterogeneous tasks using a variety of compilers.

In this example, the CPU activity percentage is 0.3 and the GPU's task completion is 0.7. Since the parallel data task can be divided into unit tasks associated with the various elements of the data, we have to control the percentage of activity for each device to provide computing power. Previous studies have shown that the system can adjust the compiler's ability to map load to load. Depending on the computing power of the computer, depending on the percentage of detailed tasks, the use of GPU to achieve relative acceleration is applicable to different applications. It is shown in Fig. 3.

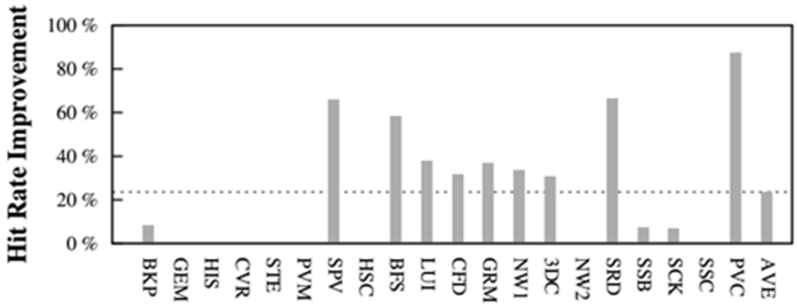

Fig. 3. L1 Cache Hit Rate Improvement (16 kB)

3. Results

The focus is on coordinating DVFS to achieve heterogeneous system task mapping [13]. To do this, under the existing adaptive synthesis method, the program is mapped to the feedback controller. In this method, a large number of parameters need to be adjusted in operation to explore the wider parameter space. Under the current architecture of the CPU / GPU, an additional capacity between the data memory and the memory is required to accommodate the task-mapped environment, which results in significant performance degradation [14]. In order to avoid this problem, we use the positive method to study the characteristics of the parameter space. Typical GPU application behavior does not perform substantially, using iterative algorithms to change the operation of many GPU applications. These facts

tell us that there is a desire to use a positive approach to coordinate task mapping between CPU/GPU and synchronization systems.

3.1. Power pack

In the proposed power capping technique, it is necessary to determine the operating frequency of the CPU and GPU, according to the percentage mapped to the CPU and GPU in the active program. To summarize the power cap technology, let's save the configuration file information for the application running on the CPU and the GPU in order for us to change the set parameters before executing the application. The information includes the empirical model and the timing of the forecast execution, and utilizes the maximum power mix of the CPU and GPU implementation. The model allows all possible settings to perform a detailed analysis of all possible settings for the mapping parameters. Predict the best parametric model, we run the application to find that the system can overcome the power constraints, and because the parameter prediction can eliminate the power dissipation error, which is shown in Fig. 4.

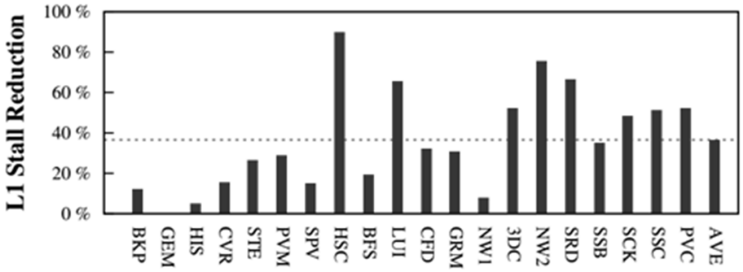

Fig. 4. Reduction on pipeline stall caused by L1 cache congestion (16 kB)

While this is rare, the research experience with this model is sufficient because we can use the existing technical force to adjust the frequency of the CPU at runtime to prevent unauthorized behavior. The response controller is responsible for adjusting the parameters. In this case, adjusting the frequency of the CPU alone does not result in large performance degradation, since the value is close to the chase high limit of the energy supply, and the adjustment of the frequency is within the specified range. We use the Linux cpufreq command to change the CPU frequency and use the nvidiasmi command to change the frequency of the GPU.

3.2. Experimental module

The power configuration information includes the number of compute nodes for the CPU and GPU. We also collect the parallel execution time of each kernel's execution function. We analyzed the execution time of the standard, compared with the execution time of the GPU at the highest frequency. Therefore, the implementation of the model can be applied to the data analysis, the use of different databases are different. When you install the application, we can get the analysis information of

the system. The system must support measurement of node energy consumption and allow the energy consumption of the CPU and GPU to be calculated, which is shown in Fig. 5.

However, this ignores the data caching capabilities of the analysis system and may ignore the good data area. Instead, by changing the location of the bypass cache data, make sure that the useful data is left in the global view. In addition, we can bypass the co-ordination of the work required to achieve data cache coordination. Recently, Professor Chen proposed an adaptive cache management technology, by protecting the limited distance within the wire to achieve cache protection to improve the performance of the cache. Similar to our findings, they have proven that the cache can effectively alleviate the transmission pressure and reduce the difficulty of purely linear execution of the program. Instead, our data is coordinated around the cache by identifying the static nature of the bypass, coordinating the local area with the overloaded area, and dynamically adjusting the local cache in the thread block.

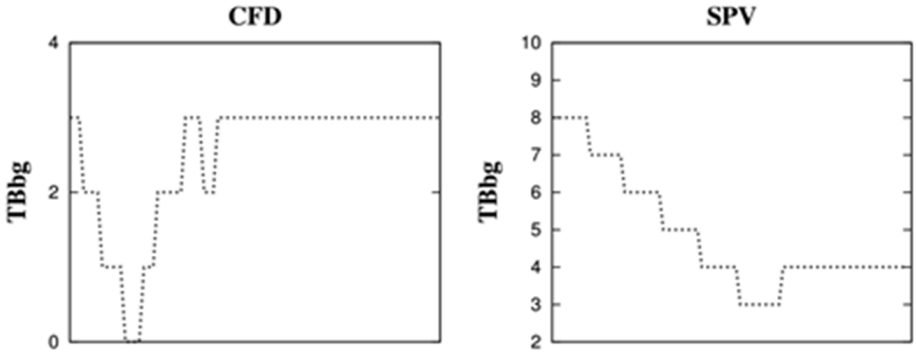

Fig. 5. TB_{bg} modulation

3.3. Execution time algorithm

By estimating the execution time of the kernel and combining the hybrid computation method, we make the following two assumptions: The actual execution time of the computer is proportional to the number of mapping tasks, and global errors need to be resolved after executing each parallel kernel The In this case, the faster equipment must wait for the other device to run. Using the two hypothetical information, we can estimate the time of the implementation of the kernel, in line with the following equation, the analysis of the collected information, through our study of empirical models and parameters, we set up the parameters to achieve the best parameter settings, Each power value constrains the best parameter in the storage list, so you need to create a table.

In addition, it takes a lot of time to build the data table, using the exhaustive search algorithm to get the best parameter settings, because we can quickly estimate the maximum execution time and estimate the actual power consumption of any parameter model. Before executing the application, get the optimal set of parameters at run time by looking up the table. However, the larger cache does not solve all

the problems. The application cache does not help improve the performance of the cache and the success rate of the calculation. But it is useful to improve the cache, because it can help reduce the stall of the pipeline.

3.4. Analysis and discussion

The GPU has been widely used to accelerate the speed of generic applications. However, making full use of GPU performance is not a simple matter. The biggest obstacle is to achieve performance optimization. GPU modeling and performance optimization, through multi-tasking and data conversion modeling, to achieve optimal parameter control, the use of the most advanced technology and memory layout program to design the chip. In all the optimization techniques, the memory system becomes more and more common, the optimization program access mode can change the irregular memory, and the application ported to the GPU. In the experiment, we used a single node and multi-core CPU to handle the problem. To get more information about the experiment, we measured the energy consumption associated with the total number of machines.

The CPU and GPU power measurement and analysis, we use the software to read the relevant hardware devices on the counter, get energy consumption information. For the bypass memory cleanup cache, for data congestion and limited cache resources, the effective way is to reduce the high speed of the pressure. Both static and dynamic methods can be used on general purpose processors. CPU technology is mainly used as a caching experimental model, cache hit rate does not always exist prediction model, including large-scale parallel processing, memory resource dispersion problem. Recent research explores the problem of GPU bypass caching. From the perspective of the cache bypass technology analysis of the data access mode and load the memory, shortening the compile time.

4. Conclusion

In order to improve the efficiency of the algorithm and reduce the power consumption of the system, this paper uses the power cap technology to study the synchronization system of the computer, control the DVFS and mapping mode, set the frequency of the synchronization system equipment and the single Computational nodes, and finally an effective power capping parallel algorithm is proposed to reduce the power consumption of the system to a predetermined level. We run five data parallel applications on a single machine, and the experimental results show that our proposed power cap algorithm is ideal for performance. At the same time, in order to avoid power failure and load imbalance, we develop new mathematical models to improve the performance of the synchronization system, which allows us to make the best settings. The parallel algorithm proposed in this paper greatly reduces the power consumption of the computer system, and has a wide application prospect. However, there are some shortcomings in the running speed and accuracy of the algorithm, the future will continue to improve the speed and accuracy of the algorithm.

References

[1] W. Shen, L. Sun, D. Wei, W. Xu, H. Wang, X. Zhu: *A hybrid parallel algorithm for computer simulation of electrocardiogram based on a CPU-GPU cluster.* Proc. IEEE/ACIS, International Conference on Computer and Information Science, 16–20 June 2013, Niigata, Japan, IEEE Conference Publications (2013), 167–171.

[2] F. Yang, T. N. Shi, H. Chu, K. Wang: *The design and implementation of parallel algorithm accelerator based on CPU-GPU collaborative computing environment.* Advanced Materials Research *529* (2012), Chapter. 4, 408–412.

[3] T. H. Chang, M. Alizadeh, A. Scaglione: *Real-time power balancing via decentralized coordinated home energy Scheduling.* IEEE Transactions on Smart Grid *4* (2013), No. 3, 1490–1504.

[4] G. J. Lu, Y. Yan, J. Y. Zhao: *Motion coordination control system with dual CPU based on ARM.* Journal of Mechanical & Electrical Engineering (2012), No. 08, 691 to 697.

[5] Y. Ma, L. Wang, A. Y. Zomaya, D. Chen,R. Ranjan: *Task-tree based large-scale mosaicking for massive remote sensed imageries with dynamic DAG Scheduling.* IEEE Transactions on Parallel and Distributed Systems *25* (2014), No. 8, 2126–2137.

[6] B. Chen, Y. Xu, J. Yang, H. Jiang: *A new parallel method of Smith-Waterman algorithm on a heterogeneous platform.* Proc. International Conference on Algorithms and Architectures for Parallel Processing, 21–23 May 2010, Busan, South Korea, Algorithms and Architectures for Parallel Processing, (LNCS) *6081* (2010) 79–90.

[7] K. Li, J. Liu, L. Wan, S. Yin, K. Li: *A cost-optimal parallel algorithm for the 0-1 knapsack problem and its performance on multicore CPU and GPU implementations.* Parallel Computing *43* (2015), 27–42.

[8] T. H. Bai, Y. G. Li, L. Y. Chen, Y. L. Wang: *Parallel optimization of geometric correction algorithm based on CPU-GPU hybrid architecture.* Applied Mechanics and Materials *543–547* (2014), No.Chapter. 6, 2804–2808.

[9] S. N. M. Shah, M. N. b. Zakaria, N. Haron, A. K. B. Mahmood, K. Naono: *Design and evaluation of agent based prioritized dynamic round Robin Scheduling algorithm on computational grids.* AASRI Procedia *1* (2012), 531–543.

[10] J. Li, B. Guo, Y. Shen, B. Li, J. Wang, Y. Huang, Q. Li: *GPU-memory coordinated energy saving approach based on extreme learning machine.* Proc. IEEE International Conference on High Performance Computing and Communications, IEEE International Symposium on Cyberspace Safety and Security, IEEE International Conference on Embedded Software and Systems, 24–26 August 2015, New York, NY, USA (2015), 827–830.

[11] L. Bukata, P. Šůcha, Z. Hanzálek: *Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA platform.* Journal of Parallel and Distributed Computing *77* (2015), 58–68.

[12] H. Guan, J. Yao, Z. Qi, R. Wang: *Energy-efficient SLA guarantees for virtualized GPU in cloud gaming.* IEEE Transactions on Parallel and Distributed Systems *26* (2015), No. 9, 2434–2443.

[13] K. Wang, Y. Huai, R. Lee, F. Wang, X. Zhang, J. H. Saltz: *Accelerating pathology image data cross-comparison on CPU-GPU hybrid systems.* Proceedings VLDB Endowment *5* (2011), No. 11, 1543–1554.

[14] M. Zouari, C. Diop, E. Exposito: *Multilevel and coordinated self-management in autonomic systems based on service bus.* Journal of Universal Computerence Science *20* (2014), No. 3, 431–460.

Received May 22, 2017