Injecting parallel computing into Architecture and Organization course

Han Wan, Xiaopeng Gao, Xiang Long
School of Computer Science and Engineering
Beihang University, Beijing, China
{wanhan, gxp, long}@buaa.edu.cn

Yi Li
Technology Engineering Group
Tencent Inc., Beijing, China
sincereli@tencent.com

ABSTRACT
The rise of multi- and many-core processing has certainly led new urgency to teaching parallel computing. This paper argues that the Architecture and Organization (AR) course is a natural place to introduce the parallelism. The main portion of the course expansion focused on general purpose computing on graphic processing unit (GPU) using CUDA. Through examples and labs, students learned the basics of GPU architectures, parallel computing along with optimization techniques to tuning the performance. All the teaching activities followed the regulations: understand the relationship between the system's elements in the height of system investigated view; using historical perspective to study the evolution of key architecture technologies; and exploration of quantitative characterization of the program's performance based on the balance of the design principles. A parallel cache simulator based on GPU was presented to conclude the course, which was found to be a good example to integrate the architecture research and parallel computing.

BACKGROUND

Course Orientation

Computer Architecture and Organization course presents the involution about the architecture; this course also does important impact on the system design related hardware and software courses. It provides support for the subjects such as software theory and computer application, which usually includes the following content- basic principles of computer design, analysis and design methods, performance evaluation, trends and new enabling technologies [1]. With the development of multi-core and many-core technology, this has brought the comprehensive change into the software and computing technology. The teaching changes are divided into two types: one is setting up new architecture course, whereas the other is adding the new framework into existing knowledge of architecture curriculum.

There are many other universities had introduced the parallel knowledge based on the CUDA environment [2][3]. Our course opened orienting the graduate student from all the majors. The necessary structural knowledge is presented to help students in understanding the basic structure and latest achievements of the system. We injected the parallel computing into AR course since 2009, and chosen CUDA as the study platform since its similarity to the C language. After five years' practice, we had adjusted the content based on students' feedback each round. In conjunction with theory of teaching curriculum, lab practices are requested to assist the understanding the parallel content.

Simulation Methodology

We thought it was important for our student to learn how to evaluate the system performance under the benchmark. Furthermore, student from the architecture major should learn how to do the architecture research. The simulation methodology is the most widely used method to do the architecture research, especially in finding the optimal architecture configuration for an application specific embedded system processor. Different architectures are evaluated and compared in order to find out which is the best. In our course, cache simulator [4] is introduced, including trace-driven simulation [5] from address reference traces of realistic workloads. And the backward of traditional sequential simulators is analyzed. Using binary instrumentation tool Pin [6] as the alternative to collect the traces is elaborated. Furthermore, the parallelism in the simulation is analyzed and the implementation of the parallel simulation on the GPU is discussed with the student.

By the end of the course, participants will understand the most important architectural performance considerations to developing GPGPU applications, as well as they can do the architecture research using simulation methodology.

ARCHITECTURE RESEARCH BASED ON GPU

We introduced a parallel cache simulator based on GPU, including decompose the task, implement parallelism using CUDA and the performance tuning. On the other hand, because expansion of the design space, the length of cache traces may run to hundreds of millions of references in traditional trace-driven simulation. Using Pin as the trace generator can characterize memory system behavior of application during its running time without using tracing (as shown in Fig. 1).

Using Pin to Generate Address Trace

Application runs on the top of Pin, then Pin provides the address traces to the cache simulation, and the cache simulation is parallel executed on the GPU. The traces include the instruction type, memory access type, memory access address and access amount. All the information should be transferred to the cache simulator, which is implemented on the GPU.

Parallelisms in the Cache Simulation

During the cache simulation, work including: fetch address from the trace; break it into tag, block number, and block offset; search blocks in the corresponding set; update the set status and metrics. We can exploit the set-parallelism: searching and updating process can be done in different sets parallel. Furthermore, the searching process in certain set can be paralleled. In a coarse granularity, multi-configuration simulation can be parallelized. That means the simulator can generate metrics for caches with different configurations within a single pass simulation.

Implement the Cache Simulator on GPU

- Memory Model

Since the trace data is large, we store it in the global memory; the information about cache set (cache lines, tag, status and metrics) are stored in the shared memory for high access speed.

- Single-level cache simulation

Each thread block is dedicated to the simulation of one cache set, while each thread within the thread block is dedicated to the simulation under different degrees of associativity.

- Multi-level cache simulation

When processes the reference, the access type and address are computed. Then we can tell the access is data access or instruction access, and the simulation of both types are in the similar way. First to find out whether hit or not, then update the cache set and statistic. The memory address that miss in the L1 cache and dirty cache line (needs to be wrote back to L2 cache) are stored, in order to form the memory reference trace for L2 cache access. So the whole simulation is divided into two parts: do the L1 cache simulation, generate the memory reference for L2, and then do the simulation for L2 cache.

- Parallel Improvement

To improve the simulator's performance, the process of trace generation and consume processes can be executed in pipeline. This parallelism focuses on the task. On the other hand, the simulation can use another timing partitioning parallel algorithm as mentioned in [7]. This work delighted that sometimes we need other directions to look at the problem.

LEARN FROM TEACHING

Historical Perspective to Study the Evolution

We first elaborated the architecture about CPU and GPU's evolution. When we introduced the GPU architecture, students need to know the control logic and memory model differences between CPU and GPU. This helped them in understanding the reason about the huge gap in the throughput of peak floating computation between CPU and GPU.

The introduction about the GPU's evolution from fixed function graphics pipeline was also necessary. When they can understand the graphics processor inheritance, the advantages and disadvantages of the current major computing model can be understood, especially help them to comprehend the modern GPU design philosophy – "massive parallel threads, a relatively small cache, and increased memory bandwidth". This also helped in grasping the future GPU development trend.

Balance Design based on Quantitative Characterization

We used the quantitative characterization to introduce the influence of different resources allocation on the performance.

For example, the increase usage of the register for each thread may result in the decrease number of thread block parallel execution, which always called "performance cliff".

When tuning the performance of Matrix multiplication, each thread can compute two elements in the result matrix instead of just compute one element. This programming improvement may decrease the access number to the global memory and increase the independent instructions in the prefetching ploy. But on the other hand, this improvement may use more registers and shared memory, which may all lead to performance cliff.

To sum up, students need to do balance design from the system-investigated view using quantitative characterization.

Students' Feedback in the Survey

The first part questions related to the course structure, nearly 77% students clearly recognized the structure and found it helpful to approach the subject. The second part questions focus on the lab sections. Most students gave positive feedback about the practical work done in the lab sessions. They liked the lab difficulty from the easier to the more advanced. And they announced that the hands-on programming based on the skeleton programs was welcomed. The lab sessions can help most students to learn and understand the content presented in class. The third part questions are about the architecture research session. Most students denoted that this session helped them lots in knowing the common architecture research methodology with tool, such as simulation methodology and Pin tool. The overall evaluation of the course turned out positive, but the flood of information still overwhelmed about 20% students.

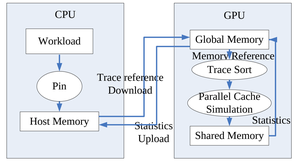

Fig.1:Parallel Cache Simulation Workflow

CONCLUSION

The computer architecture courses for undergraduate often focus on 'what the world is', our AR course for graduate paid more attention to elaborate 'why the world is like this'. Peter J.Denning has pointed out the great principles of computing in [8], which can be grouped into seven categories: computation, communication, coordination, recollection, automation, evaluation and design. For our parallel computing topic, we determined the principles that must be brought out in our course:

- The basic architecture of GPU - the parallel computing systems are built from processing core, with the memory hierarchy to store the information (computation, recollection).
- The communication in the GPU - how thread exchange information and coordination between CPU and GPU (communication).
- Predict the performance of the system based on the hardware resources' usage (evaluation).
- Decompose systems into sequential and parallel parts (design).

With the advent of parallel computing, CS departments must face the question of how to integrate the parallel knowledge units into their curricula. In this paper, we have argued that AR is a natural place to introduce parallelism. In our practice, classroom sessions, assignments and hands-on labs are used. Lastly, we have elaborated the combined architecture research and parallel computing work - cache simulator based on GPU. This is an effective supplement to our teaching philosophy: balance design based on quantitative characterization.

ACKNOWLEDGMENTS

This work is funded by China Scholarship Council (No. 201406025114) and Beihang University Boutique Graduate Course Building Project.

REFERENCES

1. John L. Hennessy, and David A. Patterson, Computer Architecture: A Quantitative Approach. ISBN-13: 978-0123838728. Morgan Kaufmann; 5th edition.
2. Engineering Tool IV - Introduction to GPU Programming [EB/OL], http://www.cse-lab.ethz.ch/index.php/teaching/42-teaching/classes/576-etvgpufall2013.
3. CUDA University Courses [EB/OL], http://www.nvidia.cn/object/cuda_university_courses_cn_old.html.
4. WAN H, LONG X, GAO X P, and Li Y, "GPU Accelerating for Rapid Multi-core Cache Simulation," IPDPS2011. Anchorage, AL, United states: IEEE Computer Society, 2011, pp. 1387-1396.
5. R. A. and T.N., "Trace-driven Memory Simulation: A survey," ACM Computing surveys, Vol.29, 1997.
6. V. Reddi, A. M. Settle, D. A. Connors and R. S. Cohn, "Pin: A Binary Instrumentation Tool for Computer Architecture Research and Education," In Proceedings of the Workshop on Computer Architecture Education, Article 22, June 2004.
7. Tobias Kiesling, "Using Approximation with Time-parallel Simulation," Simulation. Volume 81, Issue 4. Pages: 255-266. Year of Publication: 2005.
8. Peter J.Denning, "Great Principles of Computing, " Communication of ACM, Nov. 2003, 46(11), 15-20.