Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 5030 – 5034

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineeringand Information Science

BHS: an novel Scheduling strategy on modern processors

Lihan Ju*, Xingsheng Tang, Yuan Wen and Tianzhou Chen

College of Computer Science
Zhejiang University
Hangzhou, Zhejiang, 310027, P.R.China
{lhju,tang1986,wenyuan,tzchen}@zju.edu.cn

Abstract

In order to design a faster CPU, it is becoming more and more complex on the CPU architecture. But many-core is incompatible with the current programming mode designed for single-core CPU. This paper proposes a Block level Hardware-based Scheduling (BHS) on many-core architecture. The two main features are: First, design and implement a block-based hardware scheduler to reduce the overhead of threads, and to get a faster communication between processing units; second, it is very applicable to small and scalable cores on many-core architecture that is tightly coupled in the cores group, loosely coupled between groups. And a variety of parallel techniques would effectively exploit.

"Keywords: Scheduler; parallel; many-core; hardware; thread; partition"

* Corresponding author. Tel.:+8613757168142
E-mail address: lhju@zju.edu.cn

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.08.935

1. Introduction

1.1. Multi-core and many-core

These years many advanced technologies such as out-of-order execution, pipeline, branch predication and speculation have been used in CPU design for high-performance. Out-of-order is that CPU distributed instructions to the corresponding processing unit not in original order of instructions in program strictly. The purpose of using Out-Of-Order is to mine instruction-level parallelism and to cover some delay caused by access memory. So, the Out-of-Order technology can speed up the CPU but it has also some disadvantages such as high hardware complexity, hardware irregularity and great power consumption.[1]

Because the significant impact of branch failure to the pipeline performance, modern CPU would add a large branch predictor. Branch predictor uses the information existed in program to judge whether jump or not. The modern typical branch predictor has the hit rate over 90%.

Speculative threading is one of the hot areas in modern architecture research. The procedure of speculative threading is: first, some branch segments in program which had not yet been executed is executed in parallel ahead, then, according to judgment instruction, control dependence and so on, the system would have a decision to roll back or submit results of these program segments.[2]

1.2. Parallel Technology

The first one is the direct loop unrolling. But the method is available to only a few sections in program because very strict requirements needed. The prerequisite of using direct loop unrolling is that there is no other dependency among loops except Loop control parameter. The second is software pipeline[3]. Software pipeline is to exploit parallel in loops and has better adaptability than direct loop unrolling. This method executes loops in pipeline so it allows some dependency existing. The second is speculative threading. CPU executes all of branches in parallel before judged, and then rolls back or submits the results after one branch is chosen. This method needs transaction buffer to support.[4]

In fact, compiling technology needs the support of the architecture to treat with the speculative failure, speculative cache and so on. In a word, for a significant improvement of performance by exploiting parallelism, the co-processing of compiler and architecture should be necessary.

1.3. The optimized of architecture

Because the existence of Many-core and Multi-core, instruction-level parallelism can not have a good performance only by the support of architecture. The architecture is required to support some special technologies to have a nice comprehensive performance. As well as other technologies, for optimizing, the main idea is to have parallel threads as much as possible.

The rest of the paper is organized as follows: Section II summarizes the related works and motivation. Section III presents the theoretical basic of BHS of this work. Section IV describes the proposed micro-architecture of BHS mechanism. Section V and Section VI show our implementation details and experimental results.

2. Related work and motivation

One of the typical technologies is Open MP. There are two disadvantages. One is the limitation of granularity. The parallel unit should be large enough to counteract the costs of controlling threads. The consequence is the increase of system load. The other is the parallel styles are limited by Open MP which can not use the recent research results such as software pipeline, speculative thread and so on. [5]

Many-core is a heterogeneous architecture on chip, and there are different units such as GPU, memory controller and so on. Add other units to the architecture is convenience because heterogeneous. So the feature makes the design of this research possible.

3. Micro-Architecture

3.1. Parallel paragraphing

Figure 1 shows a program which has been divided into units as ES. This partition is according to the parallel feature with the cooperation of hardware architecture. Each ES is belong to a certain parallel feature, and for each different parallel features, system should use different execution models by identifying the control information in ES.

The control information can show the parallel feature of this ES for executing in different execution model. Parallel feature is including basic parallel block, loop unfolding, software pipeline and speculative execution.

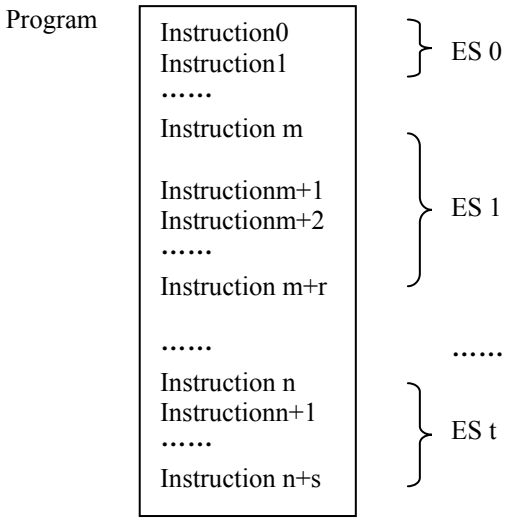

Figure 1. ES in program

3.2. Execution flow control

A program can be partitioned to many ES, and each ES has a parallel model. There needs different control structure for different parallel model. A program can be partitioned to many ES and each ES uses one parallel model. The Control Information for ES (CIES) that described control information in ES the can be used to decide the parallel model for each ES.

3.3. The design of control information-CIES

CIES described the information for executing. It is the necessary information for executing. It gives the basis of parallel feature. The construction of CIES is showed as follow:

Execution area: This is simple but the core of CIES, it gives the indication of control mode by the number of the mode. This area is just like the OP-code in instruction.

Branch address table: This area includes the branch address and the aim address of the branch. BHS is told this for speculating the branch thread in parallel.

Inferior data: The information for executing core includes the starting address, memory information and so on.

4. Implementation

Figure 2 shows the micro-architecture. That can be called a tile. A tile includes a distributer, PC controller and several simple executing cores.

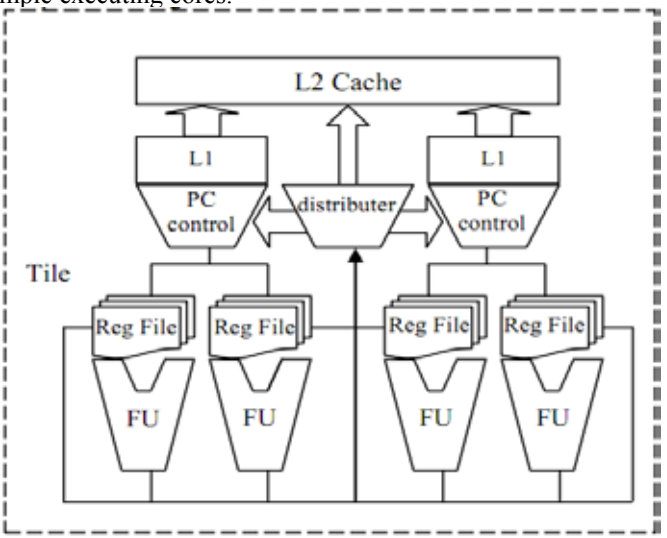

Figure 2. Micro-architecture

Executing core has a traditional PC-register. But the PC-register can be modified by the distributer which makes the PC address jumping to another block. The modification from the distributer has the highest privilege class. That means if a jump instruction and the distributer make a modification at the same time, the modification from the distributer should be done first to make the integrality of parallel section out of interruption.

The aim of the distributer is to coordinate the cores in a tile. The efficiency of distributer is according to the CIES. One function of the distributer based on CIES is to modify the PC-register for scheduling the cores. Another is to transfer register. In the process of executing parallel codes, the cores have to keep synchronous among cores. So, the distributer should transfer the data according to CIES to keep synchronous.

5. Experiment and analysis

The costs in OpenMP are caused by insufficient parallelism exploitation, inefficient expression of parallelism and lack of optimization for architecture. Figure 3 shows the relationship between efficiency and grain. When the number of loops is numerous, the speed-up ratio is same in every parallel method. But when the number of loops is few, the hardware execution stream has the better performance. That because in less loops, the extra costs of parallel thread is significant compare with it is in numerous loops. So, the more extra costs of threads, the more enervation of performance caused by this reason. In OpenMP, there is extra costs for thread library. This extra cost is about 6%.

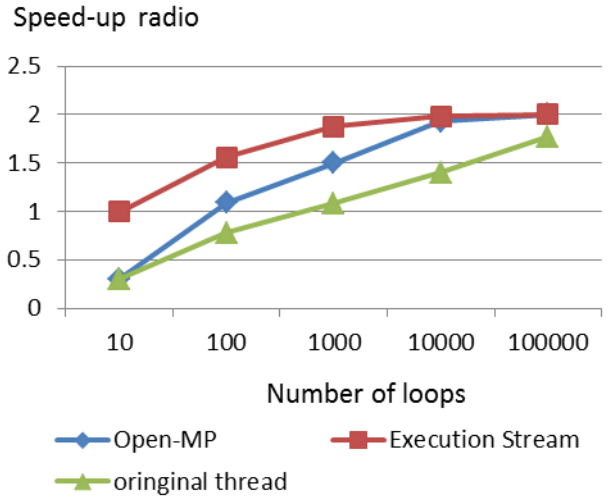

Figure 3: Relationship between efficiency and loop number

6. Acknowledge:

Supported by the National Natural Science Foundation of China under Grant No. 61070001, the Special Funds for Key Program of the China No. 2011ZX0302-004-002, the Key Science Foundation of Zhejiang Province under Grand No. 2010C11048, the Research Foundation of Education Bureau of Zhejiang Province under Grant No. Y200803333 and Y200909683, the State Key Laboratory of High-end Server & Storage Technology(No. 2009HSSA10), National Key Laboratory of Science and Technology on Avionics System Integration.

Conclusion

This paper proposes BHS. BHS can reduce the costs of thread and accelerate the communication among processing units by a compiler support hardware scheduler; the other is that the BHS has excellent extendibility for many-core architecture because of loose-coupled among tiles and tight-coupled in tile.

References

[1] M. Kyrman, *et al.*, "Cherry-MP: Correctly Integrating Checkpointed Early Resource Recycling in Chip Multiprocessors," presented at the Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture, Barcelona, Spain, 2005.

[2] T. Ohsawa, *et al.*, "Pinot: Speculative Multi-threading Processor Architecture Exploiting Parallelism over a Wide Range of Granularities," presented at the Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture, Barcelona, Spain, 2005.

[3] C. Jesshope, *et al.*, "The implementation of an SVP many-core processor and the evaluation of its memory architecture," *SIGARCH Comput. Archit. News,* vol. 37, pp. 38-45, 2009.

[4] W. Huang, *et al.*, "Many-core design from a thermal perspective," presented at the Proceedings of the 45th annual Design Automation Conference, Anaheim, California, 2008.

[5] P. Lindberg, "Basic OpenMP Threading Overhead," *http://cache-www.intel.com/cd/00/00/31/64/ 316421_316421.pdf.*