Adaptive Write-Update and Write-Invalidate Cache Coherence Protocols for Producer-Consumer Sharing

Bangjie Liu
School of Computer Science
Carnegie Mellon University
bangjiel@andrew.cmu.edu

Hao Li
School of Computer Science
Carnegie Mellon University
haol2@andrew.cmu.edu

1 Project Webpage

https://lihao98722.github.io/15740

2 Description

Cache coherence protocol has great influence on performance of shared memory multicore systems. Remote misses and bus traffics become the bottleneck of shared memory applications with high-performance demand. For this project, we focus on improving the performance of producer-consumer applications. The most popular cache coherence protocol used in modern multiprocessor architecture is directory-based write-invalidate protocol, which is inefficient for producer-consumer sharing due to extensive invalidation traffics and expensive remote misses.

In this project, we propose an adaptive cache coherence protocol optimized for producer-consumer sharing that reduces remote misses and communication traffics required to maintain coherence. It adaptively switches from write-invalidate to write-update on detecting producer-consumer sharing pattern, thereby converting remote misses to local misses and eliminating unnecessary invalidation traffics. Specifically, we will implement adaptive cache coherence as proposed protocol and directory-based write-invalidate as baseline protocol, and use cache simulator to compare latency, cache misses and bus traffics between proposed protocol and baseline protocol on the same runtime tasks in multicore systems.

3 Goals

100% goal: Implement proposed adaptive cache coherence protocols and producer-consumer sharing pattern detector. Evaluate and analyze the performance on producer-consumer applications. We expect to achieve a performance speed-up over the baseline.

75% goal: Implement and use cache simulators and testing tools (stack trace, logger, etc.) to fully evaluate and analyze the performance of directory-based cache-invalidate protocol on typical multi-threaded producer-consumer applications.

125% goal: Equip the cache simulator with functionalities that can synchronize application threads so as to observe more accurate cache access events (without the effect of pintool).

4 Logistics

03/13 - 03/19: Bangjie and Hao will fully verify the proposed approach in theory and blueprint the high-level design of cache simulators, testing tools, etc.

03/20 - 03/26: Hao will implement a highly customizable cache simulator based on pintool. And Bangjie will implement a baseline protocol (either cache-invalidate or cache-update).

03/27 - 04/02: Bangjie and Hao will integrate the cache simulator and baseline protocol. And testing tools such as stack trace, logger, etc, will be implemented if needed.

04/03 - 04/09: Bangjie and Hao will fully evaluate the baseline performance on typical producer-consumer applications and prepare for the mid-milestone presentation. We expect to finish 75% goal at the end of this period.

04/10 - 04/16: Hao will work on a detector that can identify producer-consumer sharing workloads. And Bangjie will be working on implementing another protocol in addition to baseline protocol.

04/17 - 04/23: At this point, we expect to have some issues. So we plan to use this week to address them and refactor our codebase.

04/24 - 04/30: Bangjie and Hao will integrate all components we have so far. We expect to get producer-consumer applications running on our system.

05/01 - 05/07: Bangjie and Hao will thoroughly evaluate the system performance on producer-consumer applications. We expect to achieve a reasonable speedup over the baseline.

05/18 - 05/14: Project summary and final presentation. And an evaluation of the performance on general applications will be done if we still have time.

5 Milestone

By Tuesday, April 11th, we expect to have an integrated system consist of a highly customizable cache simulator, a baseline protocol, and several testing tools if needed. And a thorough evaluation on its performance on typical producer-consumer applications can also be expected.

6 Literature Search

[1] Cheng, Liqun, John B. Carter, and Donglai Dai. "An adaptive cache coherence protocol optimized for producer-consumer sharing."High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on. IEEE, 2007.

[2] Danny. "The Disruptor - Lock-free Publishing." Codeaholics.org. http://blog.codeaholics.org/2011/the-disruptor-lock-free-publishing/, 28 June 2011. Web. 10 Mar. 2017.

7 Resources Needed

[1] Pin: https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

[2] Murphi: http://seclab.stanford.edu/pcl/mc/mc.html

[3] Stack trace: implement a stack trace tool which dynamically traces the sequences of multithread calls.

8 Getting Started

We have read multiple papers and articles on the joint research field of cache coherence and producer-consumer sharing pattern in multicore systems, and had two meetings with Prof. Beckmann discussing about the feasibility of research approach. We now have a clear idea of the current direction we should dig into: adapting write-update and write-invalidate cache coherence protocols according to sharing patterns. However, we dont have a very good estimation of the project complexity and it is possible that the project scheme may be adjusted in the future due to unexpected difficulty. We will get started by researching on the sequential consistency criterions of alternating two protocols.