15618 Project Proposal: Design and Implementation Of A Cache Hierarchy-Aware Task Scheduling On Multicore Architectures

URL

https://github.com/calvinfornialin/15618-Final-Project

Summary

- This project aims to design and implement a cache hierarchy-aware task scheduling system for multicore architectures. The goal is to optimize task execution on parallel systems by leveraging knowledge of the cache hierarchy to reduce cache misses and improve cache utilization and overall system performance.

Background

- Many modern applications, particularly those in scientific computing, data analytics, and machine learning, are compute-intensive and can benefit significantly from parallel execution. As learned in lectures, optimal performance on multicore systems requires careful consideration of how tasks are scheduled among multiple cores or processors, several crucial factors that we should consider are data locality, workload balance, and synchronization/communication time. However, the static or dynamic scheduling approaches introduced in lectures have neglected the cache hierarchy factor. So in this project we will propose a novel scheduling approach that takes not only data locality and workload balance but also cache hierarchy into account and focus on a compute-intensive component that involves frequent data accesses, such as matrix multiplication, where parallelism can be exploited through task partitioning and scheduling based on cache topology and behavior.

- Taking the task of matrix multiplication as an example, it can benefit from parallelism due to the nature that each element in the output matrix can be calculated independently from the corresponding rows and columns in the two input matrices. While this implies obvious parallelism, performing calculations in the straightforward way will actually cause latencies for large matrices if the data involved in the calculations are not arranged appropriately among the cache hierarchy. We would like to partition the task not only taking into account workload balance but also the cache hierarchy to greatly reduce latencies due to data access.
- The proposed scheduling scheme must be able to adapt to various underlying cache topologies and maintain the performance:

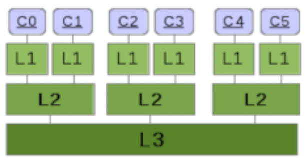

Fig. 1. Intel Dunnington

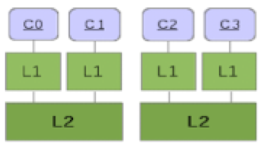

Fig. 2. Intel "Hapertown"

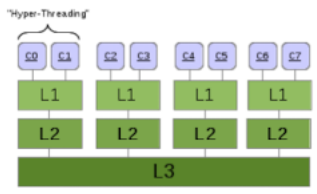

Fig. 3. Intel "Nehalem"

Challenges

- Correctly reflecting what we learned about task scheduling from lecture in the actual implementation requires firm understanding of crucial metrics and tradeoffs between different scheduling approaches.
- Although tasks such as matrix multiplication may seem to be easy to parallelize at first sight, but the challenge arise when dealing with large matrices. How to decrease the latencies due to cache misses based on the cache hierarchy while reducing the dependencies between calculations and maintaining workload balance will be a huge challenge. Proposing algorithms that can adapt to various different tasks will be even more challenging.
- Understanding each CADSS modules and how they are interconnected and interact with each other.

Workload Constraints:

- **Dependencies**: Some tasks might depend on the completion of others, which requires careful scheduling to avoid deadlocks and minimize synchronization/communication times.
- Considering the task of matrix multiplication, cache misses on each processor can potentially be decreased by dividing the calculation for each entry in the output matrix into subtasks instead of letting one single processor handle the whole calculation of the entry. However, this will introduce some dependencies, where multiple processors may be updating the same entry at the same time, causing latencies in updating the value at the entry. This level of dependency will be determined by how fine-grained the calculations are divided into.

System Constraints:

- Mapping tasks effectively to a multicore architecture considering the data locality and workload balance is already challenging, considering cache hierarchy and core interconnect bandwidth is even more difficult.
- Just like the Barnes-Hut algorithm and the ocean simulation introduced in lectures, performance of the cache (i.e. miss rate, memory access pattern) might vary a lot. So our approach should be in somewhat a higher level to adapt general cases and avoid overfitting to a specific case, which is to make the common case faster.

Resources

Machine used:

- We plan to develop the scheduling approach on local machines, and then run tests and benchmarks on the GHC and PSC machines. One reason of using both the GHC and PSC machine is because we want study both the effect of number of cores on the scalability and different cache topologies on the performance.

Starting framework:

- CADSS: We'll start implementing the computing system from scratch building upon the modules exposed by CADSS.

Reference paper:

- Nader Khammassi, Jean-Christophe Le Lann, “Design and Implementation of a Cache Hierarchy-Aware Task Scheduling for Parallel Loops on Multicore Architectures”

Goals and Deliverables

Plan to Achieve

- Implement a cache hierarchy-aware task scheduler that considers cache hierarchy for task execution.
- Demonstrate improved cache hit rates and reduced execution times for selected compute-intensive applications compared to several standard schedulers like static, dynamic scheduler, and task-stealing and achieve at least 10% of improvement.
- Gain performance improvement on several simple and common cases.

Hope to Achieve

- If ahead of schedule, we’d like to explore factors beside cache hierarchy that might also affect task scheduling performance metrics and include that in our scheduler.
- If ahead of schedule, we’d like to explore how to improve the task scheduling performance metrics of our scheduler on more test cases and situations.
- If ahead of schedule, we’d like to extend the scheduler to dynamically adapt task distribution on-the-fly based on real-time cache usage and observed behavior.

Demo

- An interactive demonstration showcasing the performance improvements in task execution times and cache utilization, compared to non-optimized scheduling. Speedup graphs and cache hit/miss statistics will be presented.

Analysis Goals

- There are several variables that we want to know how they can impact the performance of task scheduling:
    - Understand how task granularity influence task performance metrics.

    - Understand how different cache hierarchy levels influence task scheduling performance metrics and to identify the optimal task granularity for maximizing cache utilization.
    - Understand how CPU core number influences task scheduling performance metrics and to identify the optimal task granularity for maximizing cache utilization.

Platform Choice

- We will be doing most of our development and testing locally and on the GHC machines. As for the scalability study, we will be using PSC machine. Using these systems will ensure that our proposed solutions can adapt to various different cache hierarchies and scale to different numbers of processors.
- We will be using CADSS as out starting code base to simulate each component in the computer system.
- We will use C++ as our developing language.

Schedule

Week Number	Checkpoint
1	Study CADSS's basic modules code and how the simulation platform functions
2	Design and revise the algorithm of cache hierarchy-aware scheduler
3	Implementation of the scheduler
4	Perform analysis and gather data among several schedulers
5	Work on report and extending scheduler to be dynamically adjustable