Thread Scheduling for Chip-Multiprocessors: Milestone Report

Ravishankar Krishnaswamy (ravishan@cs.cmu.edu)
Harsha vardhan Simhadri (harshas@cs.cmu.edu)

November 17, 2009

1 Review of Goal

In this project, we plan to evaluate different policies for scheduling different threads of a parallel program on the different cores with the objective of minimizing runtime. Though we have still not built the scheduler, the idea is to use some information about how much memory footprint each thread would need (we expect that such information would be available to the scheduler via the programmer) and suitably place the threads. The schedulers we plan to implement are:

- Plain random work stealing.
- Work stealing based on per-cache queue, with processors attempting stealing from queues closest to it first.
- Dynamic job allocation based on working set size with out work stealing.

We plan to use easily available algorithmic code written in OpenMP or Cilk++ as our workloads. We intend to explore how these schedulers perform on different interconnects such as meshes, rings and fat trees.

2 Major Changes

There are no major changes to our project plan. While the initial idea was to use CMP-$im as a memory simulator for trace extracted from PIN, we had to change plans because of non-availability of this tool. We have looked at a number of tools and decided that the Ruby and Opal components of the GEMS simulator are most suitable for our purpose. These tools allows us to feed instructions to a configurable processor and memory model in an interactive way so that the scheduler knows when and where to put an instruction.

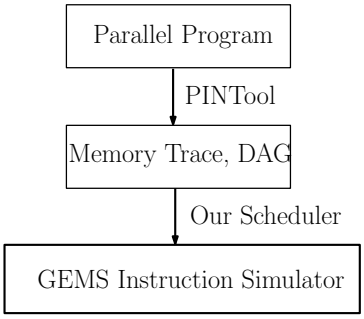

Figure 1: High Level Design

3 Status of Project

We are currently figuring out how to pick and use modules from GEMS in order to feed it instruction traces of the form (instr, memory, proc) bypassing the outer SIMICS simulation layer built into GEMS. This is so that we can control the thread scheduling outside Gems and feed into the simulators' processors, only the instructions and memory accesses. We have also written a pintool which will instrument a parallel program (like one written using OpenMP), and obtain a thread wise memory trace, and also form a Directed Acyclic Thread Graph, which indicates the spawning structure. The idea for our scheduler is to use some information about how the DAG looks (using a small amount of look-ahead) and make good placement decisions. Such look aheads will only use information about space usage of a thread which the user could have provided otherwise (or the system could have estimated on the fly).

4 Meeting Milestone

Since understanding the right simulator to use took longer than we expected it to, we are a little bit behind our proposed schedule. However, once these outer modules are completed and tested, we hope that placing a scheduler in the middle should be done without much difficulty (few hundred lines of code). Since we know what schedulers we want to program, this part is not a major bottleneck.

5 Surprises

Before we began this project, we assumed that finding a simulator which would let us alter its scheduler would be a simple task. It was very surprising to find out information on the contrary! Most simulators either fail to give timing information (they give only cache stats) or assume an OS level scheduler, which would require us to simulate an OS, and then run our program inside the OS,

while altering the scheduler of the OS. Since this seems to be a round-about method, we have decided to break open the Gems simulator and feed in instructions and processor ids to run them on directly. Our current problem is that GEMS does not have any documentation, not even comments in the code. We are trying to understand the interface by trial and error.

6 Revised Schedule

By this week end, we plan to have the layers ready and tested — the trace generator, and the interface to the processor and memory simulator. The interface would be the same irrespective of memory or processor model which are easily configured in the GEMS tool. In the last week, we plan to try out our scheduler and compare performances for different workloads.