Cache Invalidation Pattern in Shared Memory Kernel

Prakalp Srivastava

November 10, 2009

under the guidance of
Prof. Mainak Chaudhuri
Multiprocessors

In 1989, Angel L. DeCedama, “... sequential computers are approaching a fundamental physical limit on their potential computational power. Such a limit is the speed of light.”

In 2005, Intel President Paul Otellini, “... We are dedicating all of our future product development to multicore designs. We believe this is the key inflection point for the industry”
Cache Coherence

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Cache (CPU A)</th>
<th>Cache (CPU B)</th>
<th>Memory(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>A reads X</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B reads X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>A writes 0 in X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Coherence Protocols**

- Broadcast based snooping: Every cache that has a copy of the data from a block of physical memory also has a copy of the sharing status of the block. The caches are all accessible via some broadcast medium, and all cache controllers monitor or snoop on the medium.

- Directory based: The sharing status of a block of physical memory is just kept in one location, called the directory.
Cache Coherence

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Cache (CPU A)</th>
<th>Cache (CPU B)</th>
<th>Memory(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>A reads X</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B reads X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>A writes 0 in X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Coherence Protocols

- Broadcast based snooping: Every cache that has a copy of the data from a block of physical memory also has a copy of the sharing status of the block. The caches are all accessible via some broadcast medium, and all cache controllers monitor or snoop on the medium.
- Directory based: The sharing status of a block of physical memory is just kept in one location, called the directory.
Cache Coherence

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Cache (CPU A)</th>
<th>Cache (CPU B)</th>
<th>Memory(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>A reads X</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B reads X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>A writes 0 in X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Coherence Protocols
  - Broadcast based snooping: Every cache that has a copy of the data from a block of physical memory also has a copy of the sharing status of the block. The caches are all accessible via some broadcast medium, and all cache controllers monitor or snoop on the medium.
  - Directory based: The sharing status of a block of physical memory is just kept in one location, called the directory.
scalability

- None of the above two schemes scale well to hundreds of thousands of processors.
- Possible solution can be a cluster-based directory protocol, where each cluster of processors maintains coherence within the cluster using a directory, but does not maintain any coherence across clusters.
- The success of such a protocol depends on the invalidation pattern parallel programs exhibit.
None of the above two schemes scale well to hundreds of thousands of processors.

Possible solution can be a cluster-based directory protocol, where each cluster of processors maintains coherence within the cluster using a directory, but does not maintain any coherence across clusters.

The success of such a protocol depends on the invalidation pattern parallel programs exhibit.
Scalability

- None of the above two schemes scale well to hundreds of thousands of processors.
- Possible solution can be a cluster-based directory protocol, where each cluster of processors maintains coherence within the cluster using a directory, but does not maintain any coherence across clusters.
- The success of such a protocol depends on the invalidation pattern parallel programs exhibit.
PARSEC Benchmark Suite

- **Princeton Application Repository for Shared-Memory Computers**
- Benchmark suite for chip-multiprocessors composed of multithreaded programs.
- The suite provides a kernel named “canneal”, in which we observe the invalidation pattern.
PARSEC Benchmark Suite

- Princeton Application Repository for Shared-Memory Computers
- Benchmark suite for chip-multiprocessors composed of multithreaded programs.
- The suite provides a kernel named “canneal”, in which we observe the invalidation pattern.
PARSEC Benchmark Suite

- Princeton Application Repository for Shared-Memory Computers
- Benchmark suite for chip-multiprocessors composed of multithreaded programs.
- The suite provides a kernel named “canneal”, in which we observe the invalidation pattern.
Canneal Overview

- Optimization is one of the most common types of problems.
- Place and route is a difficult Electronic Design Automation (EDA) challenge.
- Transistor counts continue to increase at an exponential rate.
- Minimizes the routing cost of a chip design through simulated annealing.
- Input is a synthetic netlist, which lists the fanin and fanout for each element.
Canneal Algorithm: Simulated Annealing

1. Swap netlist elements to minimize routing cost
2. Accept disadvantageous swaps with decreasing probability to escape local minimums and converge
3. Take any two elements $a$ and $b$ at random and calculate the change in the routing cost if we swap their locations. Let us call this change $\delta$.
4. If $\delta < 0$ then swap the locations of the two elements. We call this a *good move*. However, if $\delta > 0$, which means that by performing the swap the routing cost will increase, perform the swap with probability $e^{-\delta/T}$. We call this a *bad move*.
Canneal Ganularity

- Canneal implements fine-grain parallelism.
- The number of swaps to be performed per temperature-step are divided among the threads.
Pin: A binary Instrumentation Tool

- What is instrumentation?
- A technique that inserts extra code into a program to collect runtime information

```
counter++;  # Inserted code
sub $0xff, %edx
counter++;  # Original code
cmp %esi, %edx
counter++;  # Inserted code
jle <L1>
counter++;  # Original code
mov $0x1, %edi
```
Pin: A binary Instrumentation Tool

Instrumentation Approaches

- Source Instrumentation
  - Instrument Source programs

- Binary Instrumentation
  - Instrument executables directly

Advantages for binary instrumentation

- Language independent
- Machine-level view
- Instrument legacy/proprietary software
Analysis

- Use instrumentation to generate trace.
- We track the state of all cache blocks.
- Count the invalidations at each shared write.
- Count the sharers of a memory block.
Data Collected

- Analyzed the invalidation pattern for “canneal”
- Input set was a netlist of 200,000 elements
- Assumed 32 byte block size.
- On csecourses3 machine for 2, 4, 8 and 16 threads.
Cache Invalidation Pattern in Shared Memory Kernel

My work

Shared writes = 0.3%
Avg. Invalidations per shared write = 1.03

Shared writes = 0.41%
Avg. Invalidations per shared write = 1.09

Shared writes = 0.82%
Avg. Invalidations per shared write = 1.14

Shared writes = 0.88%
Avg. Invalidations per shared write = 1.15
Cache Invalidation Pattern in Shared Memory Kernel

My work

Figure 2

Shared writes = 0.3%

Shared writes = 0.41%

Shared writes = 0.82%

Shared writes = 0.88%
Cache Invalidation Pattern in Shared Memory Kernel

My work

Reads - 3 threads
200,000 elements

Number of sharing threads

Average Sharers per cache block = 1.51

Reads - 5 threads (%)
200,000 elements

Number of sharing threads

Average Sharers per cache block = 1.72

Reads - 9 threads (%)
200,000 elements

Number of sharing threads

Average Sharers per cache block = 1.93

Reads - 17 threads (%)
200,000 elements

Number of sharing threads

Average Sharers per cache block = 2.18
Future Work

- Next step would be to look into other kernels of PARSEC
- Also we would try to relate the invalidation pattern to high level data structures, like locks and barriers
Thank You

Questions?