Scal-Tool: Pinpointing Scalability Bottlenecks

Yan Solihin, Vinh Lam, Josep Torrellas
Department of Computer Science
University of Illinois at Urbana Champaign
solihin, lam, torrella @cs.uiuc.edu
http://iacoma.cs.uiuc.edu
Problems in Programming DSMs

• Parallelizing compilers do not produce highly-scalable codes
• Programmers still need to hand-tune their code
• Performance tools are:
  – Cumbersome to use
  – Require large space
  – Unable to capture some high level info
Goals of Study

• Develop a model that pinpoints scalability bottlenecks:
  – Insufficient L2 cache size
  – Synchronization
  – Load imbalance

• The model should use few resources and be fast

<table>
<thead>
<tr>
<th>Proc Size</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>…</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>So</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>…</td>
<td>X</td>
</tr>
<tr>
<td>So/2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>So/4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
Testing Platform

- SGI Origin 2000:
  - 32 - 250 MHz R10000 processors
  - 32 KB L1 I-Cache + 32 KB L1 D-Cache
  - 4 MB L2 Cache
  - Hardware event counters in the processor
- Parallelism model: MP + PCF
- Applications: t3dheat (Los Alamos), swim (SPEC95), hydro2d (SPEC95)
Basic Formulae

• 3 types of instructions:
  – Instr that access L2 cache (fraction = $h_2$)
  – Instr that access main memory (fraction = $h_m$)
  – the rest (fraction = $1-h_2-h_m$)

$$cpi = (1-h_2-h_m)\ cpi_0 + h_2\ t_2 + h_m\ t_m$$

• $cpi$, $h_2$, $h_m$ are measured directly with performance counters.

• $cpi_0$, $t_2$, $t_m$ have to be estimated
Parameter Estimation: $cpi0$

- **Initial estimate** = $cpi0^*$
  obtained by measuring cpi of a small data set size

- **adjust the initial estimate:**
  
  $cpi0 = (1 - h2 - hm) cpi0^* - h2 t^2 - hm tm$

- **assume cpi0 constant for all applications**
Parameter Estimation: $t_2$ and $tm$

- Assume $t_2$ constant, $tm = f(n)$ for an application
- Solve linear equation system:
  \[
  cpi^1 = (1 - h2^1 - hm^1) \cdot cpi0 + h2^1 \cdot t2 + hm^1 \cdot tm
  \]
  \[
  cpi^2 = (1 - h2^2 - hm^2) \cdot cpi0 + h2^2 \cdot t2 + hm^2 \cdot tm
  \]
  \[
  \vdots
  \]
  \[
  cpi^n = (1 - h2^n - hm^n) \cdot cpi0 + h2^n \cdot t2 + hm^n \cdot tm
  \]
- Data points are obtained by varying data set size
## Bottlenecks

<table>
<thead>
<tr>
<th>Bottlenecks</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <em>Insufficient L2 cache size</em></td>
<td>Conflict misses</td>
</tr>
<tr>
<td>2. <em>MP Effects</em></td>
<td></td>
</tr>
<tr>
<td>- Synchronization</td>
<td>Coherence Misses + extra instructions</td>
</tr>
<tr>
<td>- Load imbalance</td>
<td>Extra instructions</td>
</tr>
</tbody>
</table>

WSSM '99
Execution Time

Number of processors

Execution time

- Base (Origin2000)
- Removing insufficient L2 bottleneck
- Removing mp effects
Execution Time

cpi(s,n)*inst

cpi_{\infty}(s,n)*inst

cpi_{\infty 1}(s,n)*(1-fract_{sync}-fract_{spin})*inst

(cpi_{sync} fract_{sync} + cpi_{spin} fract_{spin})*inst

Number of processors

Execution time

0 0.2 0.4 0.6 0.8 1.0 1.2

0 5 10 15 20 25 30 35

WSSM '99 10
Cpi and L2 Hit Rate

- where

\[ h2(s, n) = (1 - L1hitr(s, n)) m(s, n) L2hitr(s, n) \]

\[ hm(s, n) = (1 - L1hitr(s, n)) m(s, n) (1 - L2hitr(s, n)) \]

\[ m(s, n) = \frac{load + store}{inst} \]

- Thus

\[ cpi(s, n) = f(L2hitr(s, n), tm(n)) \]
Estimating $cpi_\infty(s,n)$

$cpi(s,n) = f(L2hitr(s,n), tm(n))$

- $cpi_\infty(s,n)$ needs $L2hitr_\infty(s,n)$
Estimating $\text{cpi}_{\infty 1}(s,n)$

\[ \text{cpi}(s,n) = f(L2\text{hitr}(s,n), tm(n)) \]

- $\text{cpi}_{\infty 1}(s,n)$ needs $L2\text{hitr}_{\infty 1}(s,n)$
Estimating $frac_{syn}, frac_{spin}$

$$cpi_{\infty}(s,n) = cpi_{\infty 1}(s,n) \left(1 - frac_{syn} - frac_{spin}\right) + cpi_{syn} * frac_{syn} + cpi_{spin} * frac_{spin}$$

- $cpi_{syn}, cpi_{spin}$: measured kernel
- $frac_{syn}, frac_{spin}$: unknown, can be measured