Application Analysis Using Memory Pressure

Kartik Sudeep (kartik@us.ibm.com)
Ahmed Gheith (ahmedg@us.ibm.com)
IBM Research
Austin Research Lab

June 2005
Agenda

- Motivation
- Related work
- Metrics
- Analysis and Results
- Conclusion
Checkpoint

- Motivation
  - Related Work
  - Metrics
  - Analysis and Results
  - Conclusion
Motivation

- “Memory Wall”
  - Memory references - the most important performance bottleneck
- Need to analyze the memory access behavior of applications
  - Characterize application’s memory reference pattern
  - Help architects evaluate micro-architectural features most suited to the application and investigate new design approaches
  - Help software designers extract the best performance for the application given the knowledge of the underlying system
Checkpoint

Motivation

Related Work

Metrics

Analysis and Results

Conclusion
Related Work

- Traditional methods of analysis
  - Mostly rely on cache misses
  - Can be misleading because all cache misses are not created equal
    - Impact of a cache miss is a function of the micro-architecture and the structure of the application itself
    - Some cache misses propagate their penalties down to the instructions that depend on the data, others can have their latencies masked by the processor pipeline
      - Examples: ATOM, DynInst, Performance Monitoring Counters
  - Instrumentation of the application to collect statistics or traces
    - Execution is perturbed rendering some results useless
Checkpoint

Motivation

Related Work

Metrics

Analysis and Results

Conclusion
The Metrics

- **Memory pressure**
  - Quantifies the demand an application places on the memory subsystem
  - Amount of slack between loading values from memory and using them
  - An extension of the load-to-use latency used in some studies

- **Metrics that are tracked for every executed instruction**
  - Value-computation-to-use delay
  - Condition-resolution-to-use delay
  - Address-computation-to-use delay
  - Value-load-to-use delay
The Metrics – what they mean

- **Value-computation-to-use delay (L)**
  - How far back were the all the values needed by this instruction computed and available on-chip
- **Condition-resolution-to-use delay (AL)**
  - How far back were all conditions resolved for all the values needed by this instruction
- **Address-computation-to-use delay (AV)**
  - How far back were addresses computed for all the values needed by this instruction
- **Value-load-to-use delay (V)**
  - How far back were the values needed by this instruction loaded from memory
- Obviously, \( V \leq AV \leq AL \leq L \)
### The Metrics - example

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>L</th>
<th>AL</th>
<th>AV</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>lwz r0, 52(r31)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1004</td>
<td>cmpwi cr7, r0, 99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1008</td>
<td>ble cr7, 0x1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100c</td>
<td>b 0x1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1100</td>
<td>lwz r0, 52(r31)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The metrics – example dissected

Compute the value X that will be stored in r31

Load the value X into an available register

Use the value X to compute other values

Resolve conditions to flow down to PC 0x1000

Load the value into r31

0x1000: lwz r0, 52(r31)

<table>
<thead>
<tr>
<th>L</th>
<th>AL</th>
<th>AV</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>m₁</td>
<td>n₁</td>
<td>p₁</td>
<td>q₁</td>
</tr>
</tbody>
</table>
The metrics – example dissected (contd.)

- Compute the value that will be stored in 52(r31)
- Load the value into an available register
- Use the value to compute other values
- Resolve conditions to flow down to PC 0x1004
  - 0x1000: lwz r0, 52(r31)
  - 0x1004: cmpwi cr7, r0, 99

Dependency table:

<table>
<thead>
<tr>
<th>L</th>
<th>AL</th>
<th>AV</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2</td>
<td>n2</td>
<td>p1+1</td>
<td>1</td>
</tr>
</tbody>
</table>
The metrics – example dissected (contd.)

Compute the PC (the only value that the execution of unconditional branch depends on)

Load the value into the PC register

Use the value to compute other values

0x1000 : lwz r0, 52(r31)
0x1004: cmpwi cr7, r0, 99
0x1008: ble cr7, 0x1010

0x100c: b 0x1100

New condition resolved and dependant on r0

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>AL</th>
<th>AV</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>m3</td>
<td>n3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

n3
m3
The metrics – example dissected (contd.)

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>L</th>
<th>AL</th>
<th>AV</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>lwz r0, 52(r31)</td>
<td>m₁</td>
<td>n₁</td>
<td>p₁</td>
<td>q₁</td>
</tr>
<tr>
<td>0x1004</td>
<td>cmpwi cr7, r0, 99</td>
<td>m₂</td>
<td>n₂</td>
<td>p₁+1</td>
<td>1</td>
</tr>
<tr>
<td>0x1008</td>
<td>ble cr7, 0x1010</td>
<td>m₂+1</td>
<td>n₂+1</td>
<td>p₁+2</td>
<td>2</td>
</tr>
<tr>
<td>0x100c</td>
<td>b 0x1100</td>
<td>m₃</td>
<td>n₃</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0x1100</td>
<td>lwz r0, 52(r31)</td>
<td>m₁+4</td>
<td>n₁+4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
The Metrics – how they are collected

- **E-mambo**
  - Native PowerPC execution-driven simulator
  - Maintains a working state of the PowerPC register set
  - Interprets the instructions and executes them on the simulator modifying registers as needed
  - Application does not need to be instrumented
  - Models memory hierarchy to collect the pressure metrics and other locality statistics
  - Generates memory/instruction traces
  - Easily extensible
Checkpoint

Motivation

Related Work

Metrics

Analysis and Results

Conclusion
Matrix Multiplication

- Iterative
  - Computes address for row element in matrix A, loads it into an FPR
    - Good use of cache due to temporal/spatial locality
  - Computes address for column element in matrix B, loads it into an FPR
    - Bad use of cache due to low spatial locality
  - Multiplies the two FPRs

```
mullw r4, r10, r28
add r11, r7, r10
addi r12, r10, 1
extsw r9, r11
extsw r10, r12
rldicr r5,r9, 2, 61
lfsx f5,r5, r27
cmpw r10, r31
add r3, r4, r8
extsw r29, r3
rldicr r0, r29, 2, 61
lfsx f4, r30, r0
fmadds f12, f5, f4, f12
```
Iterative Matrix Multiplication – No cache
Iterative Matrix Multiplication – 32K, 2W L1D
Recursive Matrix Multiplication – No cache

Latency

Instructions
Recursive Matrix Multiplication – 32K, 2W L1D
HYCOM – heat flux calculation – No cache

Latency

Instructions
HYCOM – heat flux calculation – 64K, 8W L1D
LBMHD – No cache
LBMHD – 32K, 2W L1D

Latency

Instructions
RF-CTH (erdrv) – No cache
RF-CTH (erdrv) – 64K, 8W L1D
Checkpoint

Motivation

Related Work

Metrics

Analysis and Results

Conclusion
Conclusion

- New metrics for quantifying the pressure exerted by an application on memory
  - Helps understand the behavior of applications and evaluate new architectural modifications
- Results of caching benefits for scientific applications

Future Work
- We know that caching helps...but what about other architectural techniques like Processor-In-Memory type of architectures?
This work is part of the PERCS project and was supported under DARPA HPCS Phase II

For more information: kartik@us.ibm.com
Backup slides