Positional Adaptation of Processors: Application to Energy Reduction

Michael Huang
University of Rochester

Jose Renau, Josep Torrellas
University of Illinois
Motivation

• Adaptness is increasingly important
  – Wider variety of applications
  – More complex applications
  ⇒ Behavior (demand) changes within/across apps
• …and more common
  – Different architectural components will be adaptable
Motivation (cont)

- Controlling adaptation has to be precise

Example: Applying Filter Cache

![Graph showing energy reduction over time.](image)
Contribution: Positional Adaptation

- Adaptation tied to code position (address)
  - Testing the effects of adaptations
  - Applying adaptations

- Advantages
  - More accurate in assessing effects
  - More effective adaptation: global scheduling

⇒ More effective than conventional temporal solution
Outline

• Mechanisms of positional adaptation
• Different implementations
• Evaluation/Discussion
• Related work
• Conclusions
Conventional Adaptation: *Temporal*

- A simple algorithm – DEETM’
  - Huang *et. al.* [MICRO’00]

- Improvement: variable interval
  - Balasubramonian [MICRO’00]: Phase change detection
  - Dhodapkar [ISCA’02]: Unnecessary tuning inhibition

⇒ Testing and application (of adaptation) tied to time
Problems with Temporal Adaptation

- Testing on different code sections
- Applying chosen adaptation on yet a different section
Proposed Adaptation: *Positional*

- Testing and application tied to code address
  - Less variation: more accurate testing
  - Enables *global* adaptation control

![Diagram showing IPC and code time comparison between code A and EPI with time percentages: 12%, 33%, 7%, 8%]
Implementing Positional Adaptation

- When to adapt: instrumentation (e.g. major subroutines)
- What to adapt: decision

<table>
<thead>
<tr>
<th>WHEN</th>
<th>Instrumentation</th>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Static</td>
<td>Decision Making</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Static</td>
<td>SISD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>SIDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>DIDD</td>
<td></td>
</tr>
</tbody>
</table>
SISD: Deciding *When & What* Statically
SISD: Deciding \textit{When} \& \textit{What} Statically

- No Adapt.
- Adapt. 1
- Adapt. 2
SISD: Deciding *When & What* Statically

**Off-line profiling of $\Delta D$, $\Delta E$**
- Per adaptation technique \{ a pair \}
- Per subroutine

\[ \Delta E = E_{\text{orig}} - E_{\text{activated}} \]
\[ \Delta D = D_{\text{activated}} - D_{\text{orig}} \]

<table>
<thead>
<tr>
<th></th>
<th>Adapt$_1$</th>
<th>Adapt$_2$</th>
<th>...</th>
<th>Adapt$_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub$_1$</td>
<td>$(\Delta D_{i,j}, \Delta E_{i,j})$</td>
<td>...</td>
<td>$(\Delta D_{1,3}, \Delta E_{1,3})$</td>
<td></td>
</tr>
<tr>
<td>sub$_2$</td>
<td>$(\Delta D_{2,1}, \Delta E_{2,1})$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>sub$_n$</td>
<td>$(\Delta D_{n,2}, \Delta E_{n,2})$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
E-D Tradeoff Differs Greatly across Pairs

\[
\begin{align*}
(\Delta D_{1,3}, \Delta E_{1,3})
\quad & 15% \\
(\Delta D_{2,1}, \Delta E_{2,1})
\quad & 12% \\
(\Delta D_{n,3}, \Delta E_{n,3})
\quad & 9% \\
(\Delta D_{3,m}, \Delta E_{3,m})
\quad & 6% \\
& 3%
\end{align*}
\]

- Never Apply
- Always Apply

Cumulative Energy Savings

0% 10% 20% 30% 40%
Speedup slowdown
Cumulative Slowdown

Positional Adaptation of Processors: Application to Energy Reduction – ISCA'03
Trends for Applications

Positional Adaptation of Processors: Application to Energy Reduction – ISCA’03
SIDD: Deciding *What* Dynamically

![Decision Making Diagram](chart)

- **WHAT**
  - Decision Making
    - Static
    - Dynamic

<table>
<thead>
<tr>
<th>WHEN</th>
<th>Instrumentation Static</th>
<th>Dynamic Static</th>
<th>Dynamic Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SISD</td>
<td>SIDD</td>
<td>DIDD</td>
</tr>
</tbody>
</table>
SIDD: Deciding *What* Dynamically

- Measure $\Delta D$, $\Delta E$ for the first few invocations

<table>
<thead>
<tr>
<th></th>
<th>Adapt$_1$</th>
<th>Adapt$_2$</th>
<th>Adapt$_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$j$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub$_1$</td>
<td>$\Delta D_{i,j}$, $\Delta E_{i,j}$</td>
<td>....</td>
<td></td>
</tr>
<tr>
<td>sub$_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{sub}_n$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DIDD: Deciding *When* & *What* Dynamically

<table>
<thead>
<tr>
<th>WHEN</th>
<th>Instrumentation</th>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Making</td>
<td>Static</td>
<td>SISD</td>
<td>SIDD</td>
</tr>
<tr>
<td>Dynamic</td>
<td>DIDD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*WHAT*

SISD

SIDD

DIDD
DIDD: Deciding *When & What* Dynamically

Use hardware to
- Identify major subroutines
- Facilitate efficient profiling

- **Call Stack Pointer**
- **Call Stack**
- **Call Cache**

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Mask</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Tradeoffs for Different Positional Schemes

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Pros</th>
<th>Cons</th>
<th>Target market</th>
</tr>
</thead>
</table>
| SISD    | • Global info | • Profiling | • Embedded system  
|         |      |      | • Specialized server |
| SIDD    | • Less profiling | • Limited global info  
|         |      | • Runtime overhead | • General purpose |
| DIDD    | • No profiling | • As SIDD + Hardware support | • Unavailable offline profile: e.g. dynamically generated binary |
Experimental Setup

- Processor: 6-issue OOO
- Mem System: 32KB L1 + 512KB L2 + 2 Rambus
- Variety of applications
  - SPEC-Int: bzip2, crafty, gzip, mcf, parser
  - SPEC-FP: apsi, hydro2d
  - Multimedia: mp3enc, mp3dec

- Energy largely based on Wattch [Brooks’00]
Adaptive Low-Power Techniques

• Adaptations (framework not limited to these):
  – Instruction filter cache
  – D-L1 phased cache
  – Disable slave FU cluster

• Save energy at a performance cost
Schemes Compared

• Ideal: best our low-power techniques can do
• Positional
  – SISD
  – SIDD
  – DIDD
• Temporal
  – DEETM’ (interval: 1,10, and 100μs): [Huang et. al.]
  – Rochester: [Balasubramonian et. al.]
  – Rochester’: enhancement from [Dhodapkar et. al.]
Positional Schemes More Effective

Positional Adaptation of Processors: Application to Energy Reduction – ISCA'03
Key Observations

• Positional schemes are more effective than temporal ones
  – Especially under small slack

• SISD (static *when & what*) is close to ideal
  ⇒ Code behavior in major subroutines appears largely homogenous (to our low-power techniques)
Accuracy in the Testing Period

- Behavior fluctuation across intervals w/o adaptations

⇒ Lower fluctuation for positional: more accurate testing
Related Work

- Temporal schemes (many)
  - [Balasubramonian’00] [Bahar’01] [Folegnani’01] …
- Schemes with some positional flavor
  - Adaptation for multimedia [Hughes’01][Sasanka’02]
  - Working set signature [Dhodapkar’02]
  - Hot spot-based reconfiguration [Iyer’01]
Conclusions and Future Work

• Positional more effective than temporal schemes
  – Different invocations of same code usually:
    • Have similar behavior
    • React similarly to the same adaptation

• Code behavior in subroutines appears largely homogenous (to our low-power techniques)

• Positional adaptation not limited to low-power:
  – Adaptation for high-performance
Positional Adaptation of Processors: Application to Energy Reduction

Michael Huang
University of Rochester
http://www.ece.rochester.edu/~mihuang

Jose Renau, Josep Torrellas
University of Illinois
http://iacoma.cs.uiuc.edu
Reduced \textit{ref} vs. \textit{ref}

High variation not due to reduced input set
Influence of Input Sets Small

Input-induced variation less than invocation to invocation variation
Impact Variation in the Steady-State

- Subroutine: homogenous among invocations, heterogeneous across subroutines
- More sophisticated temporal schemes better identify behavior change
Architectural Support for DIDD

Architecture support to
– Identify top subroutines
– Facilitate efficient profiling

Accounted Time
Accounted Energy
Overflow Counter
Call Stack Pointer

Return Address
Rprof

Call Table
ID | mask | D
---|------|---

Call Stack

Call Cache
ID | mask | D
---|------|---

On-chip | In-Memory
Phased Cache [Hasegawa’95]

- Sequentially access tag and data
  - Longer latency, lower E consumption