DVFS PERFORMANCE PREDICTION FOR MANAGED MULTITHREADED APPLICATIONS

Shoaib Akram, Jennifer B. Sartor, Lieven Eeckhout
Ghent University, Belgium
Shoaib.Akram@elis.UGent.be
Sample at all DVFS states 😞

Estimate performance 😊
Managed Multithreaded Applications

Garbage Collection Service

Heterogeneity

Synchronization

Store Bursts
Background

Base Frequency

- \( t_{\text{base}} \) sum of
  - Scaling (S)
  - Non-Scaling (NS)

Target Frequency

- \( r = \) Base/Target
- \( S \rightarrow S \times r \)
- \( \text{NS} \rightarrow \) No change
- \( t_{\text{target}} = (S \times r) + \text{NS} \)

- Not simple
- OOO+MLP
State of the Art

• CRIT estimates non-scaling by
  – Measuring critical path through loads
  – Ignoring store operations

Multithreaded CRIT (M+CRIT)

Base Frequency $\rightarrow$ Target Frequency $2X$

Use CRIT to identify each thread’s non-scaling

High error for multithreaded Java!
Sources of Inaccuracy in M+CRIT

Scaling or non-scaling?
Sources of Inaccuracy in M+CRIT

Scaling or non-scaling?
Our Contribution

Scaling or non-scaling?

A New DVFS Performance Predictor
Our Contribution

DEP + BURST

A New DVFS Performance Predictor
Example: Inter-thread Dependences

```
while (cond0)
{
    ...
}
Acquire(lock)
    crit_sec() ...
Release(lock)
...
```

```
while (cond1)
{
    ...
}
Acquire(lock)
    crit_sec() ...
Release(lock)
...
```

1. T0
2. T1
3. wake
4. wait

- Intercept synchronization activity
- Reconstruct execution at target frequency
Identifying Synchronization Epochs

Base Frequency ➔ Target Frequency

Epoch #1
Epoch #2
Epoch #3

T0 T1

crit_sec() loop

wait() wake()
Identifying Synchronizaton Epochs

Base Frequency  \rightarrow  Target Frequency

Epoch # 1

Epoch # 2

Epoch # 3

time
Identifying Synchronization Epochs

Base Frequency $\rightarrow$ Target Frequency

<table>
<thead>
<tr>
<th>Epoch #1</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch #2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Epoch #3</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

$\text{time}$

$= 30 \text{ units}$
Reconstruction at Target Frequency

Base Frequency → Target Frequency

Epoch #1

Epoch #2

Epoch #3

CRIT
Reconstruction at Target Frequency

Base Frequency

<table>
<thead>
<tr>
<th>Epoch</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Target Frequency

<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Longest running in an epoch = 17 units

- Zero book-keeping
- Not accurate
Reconstruction at Target Frequency

Base Frequency $\rightarrow$ Target Frequency

<table>
<thead>
<tr>
<th>Epoch #1</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch #2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Epoch #3</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

$2\times$

<table>
<thead>
<tr>
<th>#1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2</td>
<td>5</td>
</tr>
<tr>
<td>#3</td>
<td>5</td>
</tr>
</tbody>
</table>

Critical thread across epochs = 15 units

+ Accurate
- Book-keeping

= 30 units
DEP: Summary

Sync Activity

Decompose

- Sync Epochs
- Perf Counters

Reconstruct

Epochs @ Tgt.

Aggregate

Predicted Total Time
Our Contribution

DEP+BURST

A New DVFS Performance Predictor
Our Contribution

DEP+BURST

A New DVFS Performance Predictor
Store Bursts

• Reasons
  – Zero initialization
  – Copying collectors

• Modeling Steps
  – Track how long the store queue is full
  – Add to the non-scaling component
Methodology

- Jikes RVM 3.1.2
- Production collector (Immix)
- # GC threads = 2
- 2x min. heap

- 4 cores, 1.0 GHz → 4.0 GHz
- 3-level cache hierarchy
- LLC fixed to 1.5 GHz
- DVFS settings for 22 nm Haswell

- Seven multithreaded benchmarks
- Four application threads
Accuray

Baseline Frequency = 1.0 GHz

- M+CRIT
- M+CRIT+BURST
- DEP+BURST

% average absolute error

2.0 GHz: 27%
3.0 GHz: 13%
4.0 GHz: 6%
Energy Manager

tolerable_performance_degradation

4 GHz  New Freq1  New Freq2

Quantum 5 ms
Energy Savings

- Performance Degradation
- Energy Reduction

% Performance Degradation

- xalan
- pmd
- pmd.scale
- lusearch
- avg-mem
- lusearch.fix
- avrora
- sunflow
- avg-comp

Memory Intensive

Compute Intensive
Conclusions

- **DEP+BURST**: First predictor that accounts for
  - Application and service threads
  - Synchronization → inter-thread dependencies
  - Store bursts
- High accuracy
  - Less than 10% estimation error for seven Java benchmarks.
- Negligible hardware cost
  - One extra performance counter
  - Minor book-keeping across epochs
- Demonstrated energy savings
  - 20% avg. for a 10% slowdown (mem-intensive Java apps.)
Thank You!

Shoaib.Akram@elis.UGent.be

DVFS PERFORMANCE PREDICTION FOR MANAGED MULTITHREADED APPLICATIONS