Ithemal: Accurate, Portable and Fast Basic Block Throughput Estimation using Deep Neural Networks

Charith Mendis
Alex Renda
Saman Amarasinghe
Michael Carbin
Compilers need to search through code Sequences

High-level code

Optimizing Compiler

lea r14, [rbx-0x40]
lea rdx, [rbp+0x38]
cmp rdi, rax
Compilers need to search through code sequences.

40 Cycles

Optimizing Compiler

How many cycles does it take to run?
Basic Block Throughput

40 Cycles

Intel SKYLAKE

High-level code

lea r14, [rbx-0x40]
lea rdx, [rbp+0x38]
cmp rdi, rax
Compilers need to search through code sequences.

High-level code

Optimizing Compiler

Code 1
lea r14, [rbx-0x40]
lea rdx, [rbp+0x38]
lea rbx, [rbx-0x40]
cmp rdi, rax

Code 2
lea r14, [rbx-0x40]
sub rbp, 0x60
cmp rdi, rax

Code n
lea r14, [rbx-0x40]
mov rbp, rbx
cmp rdi, rax

Intel SKYLAKE

40 Cycles
44 Cycles
36 Cycles
Compilers need to search through code sequences. High-level code is optimized by the optimizing compiler, resulting in different code sequences (Code 1, Code 2, Code n). The diagram shows the process with examples of code and corresponding cycle counts (40 cycles, 44 cycles, 36 cycles). The optimized code is marked as 'Ground Truth' and the less optimized as 'Slow'.
Compilers need to search through code sequences.

High-level code → Optimizing Compiler

Code 1
- lea r14, [rbx-0x40]
- lea rdx, [rbp+0x38]
- cmp rdi, rax

Code 2
- lea r14, [rbx-0x40]
- sub rbp, 0x60
- cmp rdi, rax

Code n
- lea r14, [rbx-0x40]
- mov rbp, rbx
- cmp rdi, rax

Analytical Model

40 Cycles → Fast
44 Cycles
36 Cycles
Analytical models are inaccurate

- out-of-order
- pipelined
- super-scalar
- bypassed
- stateful components
- complicated and inaccurate manuals
- opaque implementations (vendor specific)

$\approx \sim ~20\%$ error
Analytical models are inaccurate

prediction problem is highly non-linear
Motivating Example - Zero Idioms

\[ \text{vxorps} \ xmm0, xmm0, xmm0 \]

Throughput: 1 clock cycle

\[ \text{vxorps} \ xmm1, xmm2, xmm3 \]

Special Case Throughput: 0.33 clock cycles
Motivating Example - Zero Idioms

llvm-mca

• Part of LLVM compiler infrastructure
• Uses industry standard compiler (LLVM) scheduling models
• e.g., more than >230 commits spread over 2 years for x86 Haswell Scheduling model

IACA

• Intel Architecture Code Analyzer
• Developed in-house at Intel

vxorps xmm0, xmm0, xmm0

100 iterations

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured</td>
<td>32</td>
</tr>
<tr>
<td>llvm-mca</td>
<td>100</td>
</tr>
<tr>
<td>IACA</td>
<td>24</td>
</tr>
</tbody>
</table>
Analytical models are not portable
Basic Block Throughput Estimation

```
mov ebx, [ecx]
add ebx, ecx
```

**Execute in a microprocessor**
- **Slow**
- **Ground Truth**

**Hand-written tools**
- LLVM-MCA, IACA
  - Fast
  - ~20% error
  - Not portable; manual effort needed

**Data-driven prediction**
- "Ithemal"
  - Fast
  - ~8% error
  - Portable; only need to retrain
Motivating Example - Zero Idioms

Ilvm-mca

- Part of LLVM compiler infrastructure
- Uses industry standard compiler (LLVM) scheduling models
- e.g., more than >230 commits spread over 2 years for x86 Haswell Scheduling model

IACA

- Intel Architecture Code Analyzer
- Developed in-house at Intel

Ithemal

- Data-driven model

vxorps xmm0, xmm0, xmm0

100 iterations

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured</td>
<td>32</td>
</tr>
<tr>
<td>Ilvm-mca</td>
<td>100</td>
</tr>
<tr>
<td>IACA</td>
<td>24</td>
</tr>
<tr>
<td>Ithemal</td>
<td>35</td>
</tr>
</tbody>
</table>
Ithemal model architecture

Hierarchical Embeddings

Token embeddings

Token Layer

\[ V_{\text{mov}} \quad V_{\langle S \rangle} \quad V_{\text{CONST}} \quad V_{\langle D \rangle} \quad V_{\text{ecx}} \quad V_{\langle E \rangle} \quad V_{\text{add}} \quad V_{\langle S \rangle} \quad V_{\text{ebx}} \quad V_{\text{ecx}} \quad V_{\langle D \rangle} \quad V_{\text{ebx}} \quad V_{\langle E \rangle} \]

Token Embedding Lookup Table

Canonicalization

\[ \text{mov} \quad \text{ecx}, \quad 0x02 \quad \text{add} \quad \text{ebx}, \quad \text{ecx} \]
Ithemal model architecture

Hierarchical Embeddings

Prediction Layer

Instruction Layer

Token Layer

Token Embedding Lookup Table

Canonicalization

Throughput Prediction 87.35

$V_{mov}$ $V_{<S>}$ $V_{<D>}$ $V_{<E>}$ $V_{add}$ $V_{<S>}$ $V_{ebx}$ $V_{<D>}$ $V_{ebx}$ $V_{<E>}$

$h_{mov}$ $h_{<S>}$ $h_{<D>}$ $h_{<E>}$ $h_{<S>}$ $h_{<D>}$ $h_{<E>}$

mov ecx, 0x02

add ebx, ecx

mov ecx, 0x02

add ebx, ecx
Ithemal halves the error rate

Average Prediction Error

- Ilvm-mca
- IACA
- Ithemal

Ivy Bridge
Ithemal halves the error rate across multiple microarchitectures

Average Prediction Error

- Ivy Bridge: 0.181
- Haswell: 0.209
- Skylake: 0.239

-llvm-mca
-IACA
-Ithemal

Ivy Bridge
Haswell
Skylake
Conclusion and Future Work

- Potential to replace or augment traditional systems with data-driven counterparts
- Can Ithemal be made more robust?
- Continuous improvement in compiler optimization - cost model guided learnt optimizations
Download and use!

- Dataset
- Over 1 Million-timed basic blocks
- Code: https://github.com/psg-mit/Ithemal
- Try live demo: http://3.18.198.23/predict

Come visit our poster

Today (Jun 11th Tuesday) from 06:30 to 09:00 PM at Pacific Ballroom #241