Criticality Aware Tiered Cache Hierarchy (CATCH)

Anant Nori*, Jayesh Gaur*, Siddharth Rai#, Sreenivas Subramoney*, Hong Wang*

* Microarchitecture Research Lab, Intel
# Indian Institute of Technology Kanpur, India
Popular Three Level Cache Hierarchy
Popular Three Level Cache Hierarchy

- Cache capacity $\leftrightarrow$ Access latency
- Target low *average* latency

<table>
<thead>
<tr>
<th>Level</th>
<th>Capacity</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 (Pvt.)</td>
<td>5 cyc</td>
<td></td>
</tr>
<tr>
<td>L2 (Pvt)</td>
<td>15 cyc</td>
<td></td>
</tr>
<tr>
<td>LLC (Shared)</td>
<td>40 cyc</td>
<td></td>
</tr>
</tbody>
</table>

Skylake-like Server

- 1.375MB/core
- 5.5MB (4 core)

Exclusive Data Code
- 32 KB
- 1MB
Popular Three Level Cache Hierarchy

- Cache capacity ↔ Access latency
  - Target low *average* latency
- Large distributed LLC, high latency
  - Lower L2 latency important
Popular Three Level Cache Hierarchy

- Cache capacity ↔ Access latency
- Target low *average* latency
- Large distributed LLC, high latency
- Lower L2 latency important
Popular Three Level Cache Hierarchy

- Cache capacity ↔ Access latency
- Target low average latency
- Large distributed LLC, high latency
- Lower L2 latency important
- Trend to larger L2 sizes
Popular Three Level Cache Hierarchy

- Cache capacity ↔ Access latency
- Target low *average* latency
- Large distributed LLC, high latency
- Lower L2 latency important

Is a large L2 the most efficient design choice?
Large L2 caches

- Inclusive LLC → Exclusive LLC
Large L2 caches

- Inclusive LLC → Exclusive LLC
- Lower effective on-die cache per core
Large L2 caches

- Inclusive LLC → Exclusive LLC
- Lower effective on-die cache per core
- Large LLC better for multiple threads with disparate cache footprints
Large L2 caches

- Inclusive LLC → Exclusive LLC
- Lower effective on-die cache per core
- Large LLC better for multiple threads with disparate cache footprints
- Area for Snoop-filter/Coherence-directory
Large L2 caches

- Inclusive LLC → Exclusive LLC
- Lower effective on-die cache per core
- Large LLC better for multiple threads with disparate cache footprints
- Area for Snoop-filter/Coherence-directory

Despite area and power overheads, *average* latency reduction (performance) drives large L2
Loads and Program Criticality

Program execution expressed in a Data Dependency Graph (Fields et. al.)
Program execution expressed in a Data Dependency Graph (Fields et. al.)

- Execution time governed by “Critical Path”
Program execution expressed in a Data Dependency Graph (Fields et. al.)

- Execution time governed by "Critical Path"
Loads and Program Criticality

Program execution expressed in a Data Dependency Graph (Fields et. al.)
- Execution time governed by “Critical Path”
Program execution expressed in a Data Dependency Graph (Fields et al.)

- Execution time governed by “Critical Path”

CRITICAL L2 HIT LOAD
~9% performance loss if L2 HIT → LLC HIT
Program execution expressed in a Data Dependency Graph (Fields et. al.)

- Execution time governed by “Critical Path”

CRITICAL L2 HIT LOAD
~9% performance loss if L2 HIT → LLC HIT

NON-CRITICAL L2 HIT LOAD
Program execution expressed in a Data Dependency Graph (Fields et. al.)
- Execution time governed by “Critical Path”

CRITICAL L2 HIT LOAD
~9% performance loss if L2 HIT → LLC HIT

NON-CRITICAL L2 HIT LOAD
No performance impact if L2 HIT → LLC HIT
Program execution expressed in a Data Dependency Graph (Fields et. al.)

- Execution time governed by “Critical Path”

Only *critical* load L2 hits matter to performance.
Cache Hierarchy and Program Criticality

**Oracle study**

- Track critical load PCs
- Increase latencies of targeted load PCs
Cache Hierarchy and Program Criticality

<table>
<thead>
<tr>
<th></th>
<th>Perf. Impact – All loads</th>
<th>Perf. Impact – NonCritical loads</th>
<th>% loads converted to higher latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL L1 hits to L2 lat.</td>
<td>-16.1%</td>
<td>-4.9%</td>
<td>49.1%</td>
</tr>
<tr>
<td>ALL L2 hits to LLC lat.</td>
<td>-7.8%</td>
<td>39.6%</td>
<td>-0.8%</td>
</tr>
</tbody>
</table>

Perf. Impact – All loads
Perf. Impact – NonCritical loads
% loads converted to higher latency

-16% -14% -12% -10% -8% -6% -4% -2% 0% 2% 4% 6% 8% 10% 12% 14% 16% 18%
Cache Hierarchy and Program Criticality

L2 cache most amenable to criticality optimizations
Criticality Aware Tiered Cache Hierarchy (CATCH)

A. Track *critical* load PCs
Criticality Aware Tiered Cache Hierarchy (CATCH)

A. Track *critical* load PCs
   - Served from non-L1 on-die caches
Criticality Aware Tiered Cache Hierarchy (CATCH)

A. Track critical load PCs
   • Served from non-L1 on-die caches

B. Prefetch critical loads into L1
   • Accelerate the critical path
Criticality Aware Tiered Cache Hierarchy (CATCH)

A. Track critical load PCs
   • Served from non-L1 on-die caches

B. Prefetch critical loads into L1
   • Accelerate the critical path
Criticality Aware Tiered Cache Hierarchy (CATCH)

A. Track critical load PCs
   • Served from non-L1 on-die caches

B. Prefetch critical loads into L1
   • Accelerate the critical path

L2 can become redundant in a criticality aware cache hierarchy
CATCH Configuration Options

**BASELINE**

- **Level1 (L1) Private**
  - Data: 32 KB

- **Level2 (L2) Private**
  - 1MB

- **Level3 (L3) Shared**
  - 5.5MB Exclusive

**CATCH Hardware**

- **A. Track critical load PCs**
- **B. Prefetch critical loads in L1**
CATCH Configuration Options

Level1 (L1)
- Private

Level2 (L2)
- Private

Level3 (L3)
- Shared

BASELINE
- Data: 32 KB
- Code: 1MB

CATCH Hardware
A. Track critical load PCs
- Data: 32 KB
- Code: 1MB

Three-Level CATCH
- Data: 32 KB
- Code: 1MB

A. Track critical load PCs
B. Prefetch critical loads in L1

Accelerates critical path
CATCH Configuration Options

**BASELINE**
- Level1 (L1) Private
- Level2 (L2) Private
- Level3 (L3) Shared

**CATCH Hardware**
- A. Track critical load PCs
- B. Prefetch critical loads in L1

**Three-Level CATCH**
- Accelerates critical path

**Two-Level CATCH (NoL2)**
- Accelerates critical path
  + Area saving
CATCH Configuration Options

Level1 (L1) Private

- PRIVATE

Level2 (L2) Private

- PRIVATE

Level3 (L3) Shared

- 5.5MB Exclusive

BASELINE

- 1MB

CATCH Hardware

- Data 32 KB
- Code

A. Track critical load PCs

B. Prefetch critical loads in L1

Three-Level CATCH

- Data 32 KB
- Code

- 5.5MB Exclusive

Accelerates critical path

Two-Level CATCH (NoL2)

- Data 32 KB
- Code

- 5.5MB Inclusive

Accelerates critical path

Two-Level CATCH (NoL2, IsoArea)

- Data 32 KB
- Code

- 1MB

+ Area saving

Accelerates critical path
CATCH Configuration Options

<table>
<thead>
<tr>
<th>Level1 (L1) Private</th>
<th>Level2 (L2) Private</th>
<th>Level3 (L3) Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 32 Code 1MB</td>
<td>Data 32 Code 1MB</td>
<td>5.5MB Exclusive</td>
</tr>
</tbody>
</table>

BASELINE

CATCH Hardware

A. Track critical load PCs

B. Prefetch critical loads in L1

Three-Level CATCH

Accelerates critical path

Two-Level CATCH (NoL2)

Accelerates critical path + Area saving

Two-Level CATCH (NoL2, IsoArea)

Accelerates critical path + Power saving

5.5MB Exclusive

9.5MB Inclusive

5.5MB Inclusive

Intel
A) Criticality Detection in Hardware

Instructions being executed (ROB)

Instructions being allocated
A) Criticality Detection in Hardware

Buffer execution DDG (Fields et. al.) on instruction retire
A) Criticality Detection in Hardware

Buffer execution DDG (Fields et. al.) on instruction retire

Enumerate critical path every 2x ROB instruction retires
A) Criticality Detection in Hardware

Buffer execution DDG (Fields et. al.) on instruction retire

Enumerate critical path every 2x ROB instruction retires

32 entry
Critical Load PC Table
A) Criticality Detection in Hardware

Buffer execution DDG (Fields et. al.) on instruction retire

Enumerate critical path every 2x ROB instruction retires

**Optimizations:** Area of DDG, Fast enumeration of critical path
A) Criticality Detection in Hardware

Buffer execution DDG (Fields et. al.) on instruction retire

Enumerate critical path every 2x ROB instruction retires

**Optimizations:** Area of DDG, Fast enumeration of critical path

- Uses 3KB of storage. Details in the paper
B) Timeliness Aware, Criticality Triggered (TACT) Prefetchers

*Critical load PC* prefetchers optimized for *inter-cache* prefetching into Data L1
B) Timeliness Aware, Criticality Triggered (TACT) Prefetchers

*Critical load PC* prefetchers optimized for *inter-cache* prefetching into Data L1

**Data Prefetchers**

- Identify “Trigger” load PCs for “Target” critical load PCs
B) Timeliness Aware, Criticality Triggered (TACT) Prefetchers

Critical load PC prefetchers optimized for inter-cache prefetching into Data L1

Data Prefetchers

- Identify “Trigger” load PCs for “Target” critical load PCs

Code “Run-Ahead” Prefetcher

- Cover LLC latency instead of L2 (for when the L2 is removed)
TACT: Data Prefetchers
TACT: Data Prefetchers

“Cross” Prefetcher:
“Cross” Prefetcher:

- Trigger load PC address @ constant delta from Target/Critical PC
TACT: Data Prefetchers

“Cross” Prefetcher:

- Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:

\[ a, a+\delta, a+2\delta, \ldots, a+16\delta, a+17\delta, a+18\delta \]
“Cross” Prefetcher:
- Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:
- Upto deep prefetch distance 16
TACT: Data Prefetchers

“Cross” Prefetcher:
• Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:
• Upto deep prefetch distance 16

“Feeder” Prefetcher: Address of Target/Critical load = $M \cdot$ Data of Feeder load + $C$

Addr. Trigger $T$ to Addr. Target $C = \Delta$

$a, a+\delta, a+2\delta, \ldots, a+16\delta, a+17\delta, a+18\delta$
TACT: Data Prefetchers

“Cross” Prefetcher:
- Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:
- Upto deep prefetch distance 16

“Feeder” Prefetcher: Address of Target/Critical load = \( M \) * Data of Feeder load + \( C \)
TACT: Data Prefetchers

“Cross” Prefetcher:
- Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:
- Upto deep prefetch distance 16

“Feeder” Prefetcher: Address of Target/Critical load = $M \times$ Data of Feeder load + $C$

SELF “Deep” Address Prefetch of Feeder F

Feeder Prefetch Data to Prefetch Address of Target D
TACT: Data Prefetchers

“Cross” Prefetcher:
• Trigger load PC address @ constant delta from Target/Critical PC

“Self” Deep Prefetcher:
• Upto deep prefetch distance 16

“Feeder” Prefetcher: Address of Target/Critical load = $M \times$ Data of Feeder load + $C$

Implementation details in the paper
TACT: Code “Run-Ahead” Prefetcher
TACT: Code “Run-Ahead” Prefetcher

On front-end stall:

- Use NIP logic (Branch Prediction, BTB)
- Speculatively prefetch code lines
Evaluation: Configuration

- 4 x86 cores @ 3.2GHz, 4-wide, 224 ROB entries
- 32 KB, 8-way Data and Code L1
- PC-based stride prefetcher, multi-stream prefetchers
- Dual channel DDR4-2400 main memory
Evaluation : Configuration

• 4 x86 cores @ 3.2GHz, 4-wide, 224 ROB entries
  • 32 KB, 8-way Data and Code L1
  • PC-based stride prefetcher, multi-stream prefetchers
• Dual channel DDR4-2400 main memory
• Two baseline L2/LLC configurations
  • Large L2 (1MB), Exclusive LLC (5.5MB, 1.375 MB/core)
  • Small L2 (256KB), Inclusive LLC (8MB)
Evaluation : Configuration

• 4 x86 cores @ 3.2GHz, 4-wide, 224 ROB entries
  • 32 KB, 8-way Data and Code L1
  • PC-based stride prefetcher, multi-stream prefetchers
• Dual channel DDR4-2400 main memory
• Two baseline L2/LLC configurations
  • Large L2 (1MB), Exclusive LLC (5.5MB, 1.375 MB/core)
  • Small L2 (256KB), Inclusive LLC (8MB)
• 70 diverse ST workloads – SPEC-06, HPC, Server, Client
  • 60 four-way multi-programmed workloads
Large 1MB L2, Exclusive 1.375 MB LLC per Core

ST GeoMean Performance Impact

[Graph showing performance impact with various configurations: NoL2, NoL2 + CATCH, NoL2 + 9.5MB LLC + CATCH, and CATCH.]
Large 1MB L2, Exclusive 1.375 MB LLC per Core

ST GeoMean Performance Impact

CATCH accelerates the critical path and enables designs with better performance and area/power tradeoffs.
Large 1MB L2, Exclusive 1.375 MB LLC per Core
ST Per-Workload Performance Impact

TACT prefetchers recover loss from no L2 in majority of workloads
Large 1MB L2, Exclusive 1.375 MB LLC per Core

ST Per-Workload Performance Impact

TACT prefetchers recover loss from no L2 in majority of workloads
Research on optimizations to improve remaining outliers
Power Analysis:
Iso-Area NoL2+9.5MB LLC + CATCH
Power Analysis:
Iso-Area NoL2+9.5MB LLC + CATCH

• Removal of L2
  • Reduced cache activity
  • Increased interconnect traffic
Power Analysis:
Iso-Area NoL2+9.5MB LLC + CATCH

- Removal of L2
  - Reduced cache activity
  - Increased interconnect traffic
- Increased LLC (+L2 from all cores)
  - Reduced DRAM traffic
Power Analysis:
Iso-Area NoL2+9.5MB LLC + CATCH

• Removal of L2
  • Reduced cache activity
  • Increased interconnect traffic
• Increased LLC (+L2 from all cores)
  • Reduced DRAM traffic

Overall impact
• ~11% energy savings
• With 7.2% performance gains
Power Analysis: Iso-Area NoL2+9.5MB LLC + CATCH

- Removal of L2
  - Reduced cache activity
  - **Increased interconnect traffic**
- Increased LLC (+L2 from all cores)
  - Reduced DRAM traffic

**Overall impact**
- ~11% energy savings
- With 7.2% performance gains

For large, power hungry (mesh) interconnects, power impact too high
Power Analysis:
Iso-Area NoL2+9.5MB LLC + CATCH

- Removal of L2
  - Reduced cache activity
- Increased interconnect traffic
- Increased LLC (+L2 from all cores)
  - Reduced DRAM traffic

**Overall impact**
- ~11% energy savings
- With 7.2% performance gains

For large, power hungry (mesh) interconnects, power impact too high
⇒ Small L2 to absorb L1 evictions preferable
Summary
Summary

- Fundamental re-look at each level of a three level cache hierarchy
Summary

- Fundamental re-look at each level of a three level cache hierarchy
  - L2 highly amenable to criticality optimizations
  - Trend towards large L2 not the most efficient design choice
Summary

- Fundamental re-look at each level of a three level cache hierarchy
  - L2 highly amenable to criticality optimizations
  - Trend towards large L2 not the most efficient design choice
- CATCH introduces:
  - Dynamic tracking of critical loads.
  - Optimized inter-cache prefetch into L1
Summary

• Fundamental re-look at each level of a three level cache hierarchy
  ▪ L2 highly amenable to criticality optimizations
  ▪ Trend towards large L2 not the most efficient design choice
• CATCH introduces:
  • Dynamic tracking of critical loads.
  • Optimized inter-cache prefetch into L1
• CATCH enables radical new processor designs
  • Efficient area/power/performance tradeoffs

<table>
<thead>
<tr>
<th></th>
<th>Three-level CATCH</th>
<th>Two-level CATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.4% perf. gain</td>
<td>Iso-area 7.2% perf. gain + 11% energy saving</td>
</tr>
<tr>
<td></td>
<td>+ 30% area saving</td>
<td>4.2% perf. gain + 30% area saving</td>
</tr>
</tbody>
</table>
THANK YOU