BASE-VICTIM COMPRESSION : AN OPPORTUNISTIC CACHE COMPRESSION ARCHITECTURE

Jayesh Gaur*, Alaa Alameldeen**, Sreenivas Subramoney*

* Microarchitecture Research Lab (MRL), Intel India
**Memory Architecture Lab (MAL), Intel Oregon

ISCA 2016
Seoul, Korea
Motivation

• Memory continues to be the bottleneck for modern CPU’s
• Larger Last Level Cache (LLC) capacity improves performance and power
  • Higher hit rates $\rightarrow$ Better Performance
  • Fewer off-chip DRAM accesses $\rightarrow$ Lower power, energy
• However this comes at a cost: Area and Leakage
• Cache compression is an attractive option
  • Increased Capacity at lower area

But Compression can interfere with Replacement Policies
We present a new compression architecture to address this
How Cache Compression Works?

Prior works have tried to change SRAM layout to allow compression. Changing a dense, timing sensitive SRAM layout is difficult.
Agenda

• Practical Architecture for Compressed Cache
• Interaction between Compression and Replacement Policies
• Base-Victim Proposal
• Results
  • Performance
  • Power
• Conclusions
Creating a Compressed LLC

- Tags per Set are doubled
- Exactly two tags are associated with each way
  - Tag hit to data fetch is optimized
- Only 64B of data for every two tags
  - Data corresponding to the tags is compressed

Data-0-size + Data-1-size ≤ 64B
Performance gain from compression

Average 12% Loss! Hit Rates lower in general
Why did larger LLC capacity lower performance?
Issues with Compressed Cache

Partner line victimization

- Replacement policy is broken because of size limitations
- Performs poorer than baseline with many negative outliers

Allocating into LRU way will victimize the partner MRU line!

We need to increase capacity but also preserve the replacement policy!
Opportunistic Victim Cache

- Extra capacity (Tag-1 in each way) logically belongs to a Victim Cache
  - Tag 0 victims are cached in Tag 1 space
- Base replacement policy strictly maintained in Tag 0
  - Guarantees baseline cache hit behavior, performance
- Victim cache is always clean
  - Partner line victimization is easy
Compressed LLC : Miss

Read miss allocates in Tag0 LRU

If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache

- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache
Compressed LLC: Miss

Read miss allocates in Tag0 LRU

If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache

- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache
Compressed LLC : Miss

Read miss allocates in Tag0 LRU

If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache

- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache

<table>
<thead>
<tr>
<th>&lt;Tag, Size&gt;</th>
<th>Ways</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Baseline Cache
- D,24
- C,8
- A,48
- Z,48

Victim Cache
- F,32
- E,8
- X,16
- Y,32
- B,24

Incoming Request

Compressed LLC : Miss
Compressed LLC : Miss

Read miss allocates in Tag0 LRU
If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache
- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache
Read miss allocates in Tag0 LRU
If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache
- In baseline it was never there
- Cannot be poorer than baseline
If victim created insert into Tag1 victim cache
Read miss allocates in Tag0 LRU

If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache

- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache
Compressed LLC : Miss

Read miss allocates in Tag0 LRU
If size exceeds what is available in Tag0, victimize partner line in Tag1 victim cache
- In baseline it was never there
- Cannot be poorer than baseline

If victim created insert into Tag1 victim cache

Victim cache is always clean → Dirty lines written to memory
Return data to core after decompression
If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical

- Needs more analysis
If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical
- Needs more analysis
Compressed LLC: Hit in Victim $\$

If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical
- Needs more analysis
Compressed LLC: Hit in Victim $\$

If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical

- Needs more analysis
If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical
- Needs more analysis
If hit from Tag1 victim cache then move data to Tag0

- Behave as if read miss served from memory
- But latency is just lookup of LLC
- Gives performance

Management of Victim Cache is critical
- Needs more analysis
Configuration

- x86 Core running at 4GHz
- 2MB, 16 way Inclusive LLC per core
- Not Recently Used (NRU) replacement
- DDR3-1600 15-15-15-34
- Base Delta Immediate (BDI) Compression
- Decompression Latency of 2 cycles

On an average each cache-line gets compressed to 55% of its size

**Doubling Tags should get most of the gains**
Results: IPC Gain

8% area addition gives performance equal to 50% increased area
Good gains across various categories of workloads
Results: Correlation with hit rate improvement

Hit Rate $\geq$ Baseline hit rate. No negative outliers
Memory traffic reduces by average 16%
Effect of Baseline Replacement Policy

Good gains with various state of the art replacement policies
Increases capacity while retaining benefits of good replacement!
Energy Savings

Power saved in DRAM compensates for increased power in LLC

Overall 6.5% energy savings
Conclusions

• Cache Compression increases capacity with low area impact
  • But compression interferes with replacement policies
• We propose Base-Victim compression
  • Opportunistic Victim cache created by compression
  • Preserves gains from replacement policies
  • No costly SRAM layout changes
    • All changes in the cache controller
• ~50% increase in capacity with 8% area addition
THANKS