Cache Coherence Scaling on Manycore Systems

Oliver Türk
oliver.tuerk@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg

17.01.2017
Motivation

Mainstream cache implementation today

Scaling
  The Naive Way
  Inexact Tracking of Sharers
  Cache Hierarchy

References
Processor Scaling

- Complexity will increase

Pollack's Rule - problematic

Solution: Split processor into cores

More cores - also problematic
Processor Scaling

- Complexity will increase
- Pollacks Rule - problematic
Processor Scaling

- Complexity will increase
- Pollack's Rule - problematic
- Solution: Split processor into cores
Processor Scaling

- Complexity will increase
- Pollacks Rule - problematic
- Solution: Split processor into cores
- More cores - also problematic
Pollack’s Rule Inverted

![Graph showing Pollack's Rule Inverted]

- X-axis: Core count
- Y-axis: Speedup

The graph illustrates the relationship between core count and speedup for Pollack’s Rule Inverted.
Motivation

Mainstream cache implementation today

Scaling
  The Naive Way
  Inexact Tracking of Sharers
  Cache Hierarchy

References
Mainstream cache implementation today

Intel i7 6950X processor with 25Mb L3 cache (1)

- hardware coherent
Mainstream cache implementation today

Intel i7 6950X processor with 25Mb L3 cache (1)

- hardware coherent
- inclusive
Mainstream cache implementation today

- Hardware coherent
- Inclusive
- Last level shared

Intel i7 6950X processor with 25Mb L3 cache (1)
Mainstream cache implementation today

- hardware coherent
- inclusive
- last level shared
- size of last level greater than aggregate below

Intel i7 6950X processor with 25Mb L3 cache (1)
Mainstream cache implementation today

- hardware coherent
- inclusive
- last level shared
- size of last level greater than aggregate below
- exact tracking of sharers with 1 bit for each core

Intel i7 6950X processor with 25Mb L3 cache (1)
Mainstream cache implementation today

Cache implementation on Intel i7 CPU (2)
Motivation

Mainstream cache implementation today

Scaling

The Naive Way
Inexact Tracking of Sharers
Cache Hierarchy

References
Today we have typically..

- 64 Byte per cache line
- 4 cores
- 1 Bit per core for tracking
- overhead below 5%
Scaling - the naive way

Let's assume..

- 64 Byte per cache line
- 1024 cores
- 1 Bit per core for tracking
Scaling - the naive way

Let's assume..

- 64 Byte per cache line
- 1024 cores
- 1 Bit per core for tracking

This results in..

- 1024 bit (64 Byte) for tracking
- overhead of ???
Scaling - the naive way

Lets assume..

- 64 Byte per cache line
- 1024 cores
- 1 Bit per core for tracking

This results in..

- 1024 bit (64 Byte) for tracking
- overhead of 50%

not acceptable!
Outline

Motivation

Mainstream cache implementation today

Scaling
  The Naive Way
  Inexact Tracking of Sharers
  Cache Hierarchy

References
Scaling - inexact tracking of sharers

Let's assume..

- 64 Byte per cache line
- 1024 cores
- 32 Bits for tracking in total
Scaling - inexact tracking of sharers

Let's assume..

- 64 Byte per cache line
- 1024 cores
- 32 Bits for tracking in total

This results in..

- overhead of 6.25%

and is acceptable.
Scaling - inexact tracking of sharers

How it works..

- inclusive shared cache
Scaling - inexact tracking of sharers

How it works..

- inclusive shared cache
- single level shared cache
Scaling - inexact tracking of sharers

**How it works..**
- inclusive shared cache
- single level shared cache
- exact tracking for up to 32 sharers
Scaling - inexact tracking of sharers

How it works..

- inclusive shared cache
- single level shared cache
- exact tracking for up to 32 sharers
- inexact tracking for more sharers
How it works..

- inclusive shared cache
- single level shared cache
- exact tracking for up to 32 sharers
- inexact tracking for more sharers

High traffic with more than 32 sharers (threads)
Scaling - inexact tracking of sharers

Traffic with inexact tracking of sharers (3)
Outline

Motivation

Mainstream cache implementation today

Scaling
  The Naive Way
  Inexact Tracking of Sharers
  Cache Hierarchy

References
Advantages..

- tracking of subsequent clusters only
Advantages..

- tracking of subsequent clusters only
- access to memory stays transparent
Advantages:

- tracking of subsequent clusters only
- access to memory stays transparent
- cache coherent
Advantages:
- tracking of subsequent clusters only
- access to memory stays transparent
- cache coherent
- software backwards compatibility
Advantages..

- tracking of subsequent clusters only
- access to memory stays transparent
- cache coherent
- software backwards compatibility
- invariable traffic
Advantages:

- tracking of subsequent clusters only
- access to memory stays transparent
- cache coherent
- software backwards compatibility
- invariable traffic

Disadvantages:
Scaling - cache hierarchy

Advantages..

▶ tracking of subsequent clusters only
▶ access to memory stays transparent
▶ cache coherent
▶ software backwards compatibility
▶ invariable traffic

Disadvantages..

▶ higher latency between distant cores
Traffic with exact tracking of sharers (3)
Scaling - cache hierarchy

Figure 3. Storage overhead in shared caches.

Storage Overhead for multiple hierarchy levels (3)
Scaling - cache hierarchy: single-level shared

Shared and private caches with 2 levels
Scaling - cache hierarchy: two-level shared

Shared Cache (L3)

- tracking bits
- state
- tag
- block data

~1 bit per clust
~2 bits
~64 bits
~512 bits

Cluster 0
Cluster 1
Cluster n

Shared and private caches with 3 levels

Oliver Türk
oliver.tuerk@fau.de

Cache Coherence Scaling on Manycore Systems
17.01.2017 23 / 28
Scaling - cache hierarchy: two-level shared

Shared and private caches with 3 levels
Mainstream processor: i7 6700K..

- 4 cores
- one-level hierarchy
- 256 KiB per L2
- 8 times of aggregate
- \(1.75 \times 10^9\) transistors
Scaling - cache memory in transistors

**Let's assume..**

- 1024 cores
- *two-level* hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

...This results in...

- 32 cores per cluster
- 32 cluster
- L2 with 256 KiB \(\times 32\) cores \(\times 8 = 64\) MiB
- L3 with 64 MiB \(\times 32\) clusters \(\times 8 = 16\) GiB

824 × 10^9 transistors
Scaling - cache memory in transistors

**Lets assume..**

- 1024 cores
- *two-level* hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

**This results in..**
Scaling - cache memory in transistors

Let's assume..
- 1024 cores
- two-level hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

This results in..
- 32 cores per cluster
Scaling - cache memory in transistors

Let's assume..

- 1024 cores
- two-level hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

This results in..

- 32 cores per cluster
- 32 cluster
Scaling - cache memory in transistors

Let's assume..

- 1024 cores
- two-level hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

This results in..

- 32 cores per cluster
- 32 cluster
- L2 with \(256 \text{KiB} \times 32 \text{cores} \times 8 = 64 \text{MiB}\)
Scaling - cache memory in transistors

Let's assume..

- 1024 cores
- *two-level* hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

This results in..

- 32 cores per cluster
- 32 cluster
- L2 with 256 KiB * 32 cores * 8 = 64 MiB
- L3 with 64 MiB * 32 clusters * 8 = 16 GiB
Scaling - cache memory in transistors

Let's assume..

- 1024 cores
- two-level hierarchy
- 6 transistors per flip-flop
- 256 KiB per L1
- 8 times of aggregate

This results in..

- 32 cores per cluster
- 32 cluster
- L2 with $256\text{KiB} \times 32\text{cores} \times 8 = 64\text{ MiB}$
- L3 with $64\text{MiB} \times 32\text{clusters} \times 8 = 16\text{ GiB}$
- $824 \times 10^9$ transistors
Figure 4. Likelihood a shared cache miss triggers a recall.

Associativity VS capacity (3)
I. Cutress, *Intel i7 6950X DIE*,
http://www.anandtech.com/show/10337/the-intel-
broadwell-e-review-core-i7-6950x-6900k-6850k-and-
6800k-tested-up-to-10-cores, [Online, accessed

J. Treibig, G. Hager, presented at the International
Conference on Parallel Processing and Applied Mathematics,
pp. 615–624.

M. M. K. Martin, M. D. Hill, D. J. Sorin, *Commun. ACM* 55,
78–89, ISSN: 0001-0782 (July 2012).