Cache Coherence Scaling on Manycore Systems

Oliver Türk
oliver.tuerk@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg

17.01.2017

Talbe Of Contents

Processor Scaling

- Complexity will increase

Processor Scaling

- Complexity will increase
- Pollacks Rule - problematic

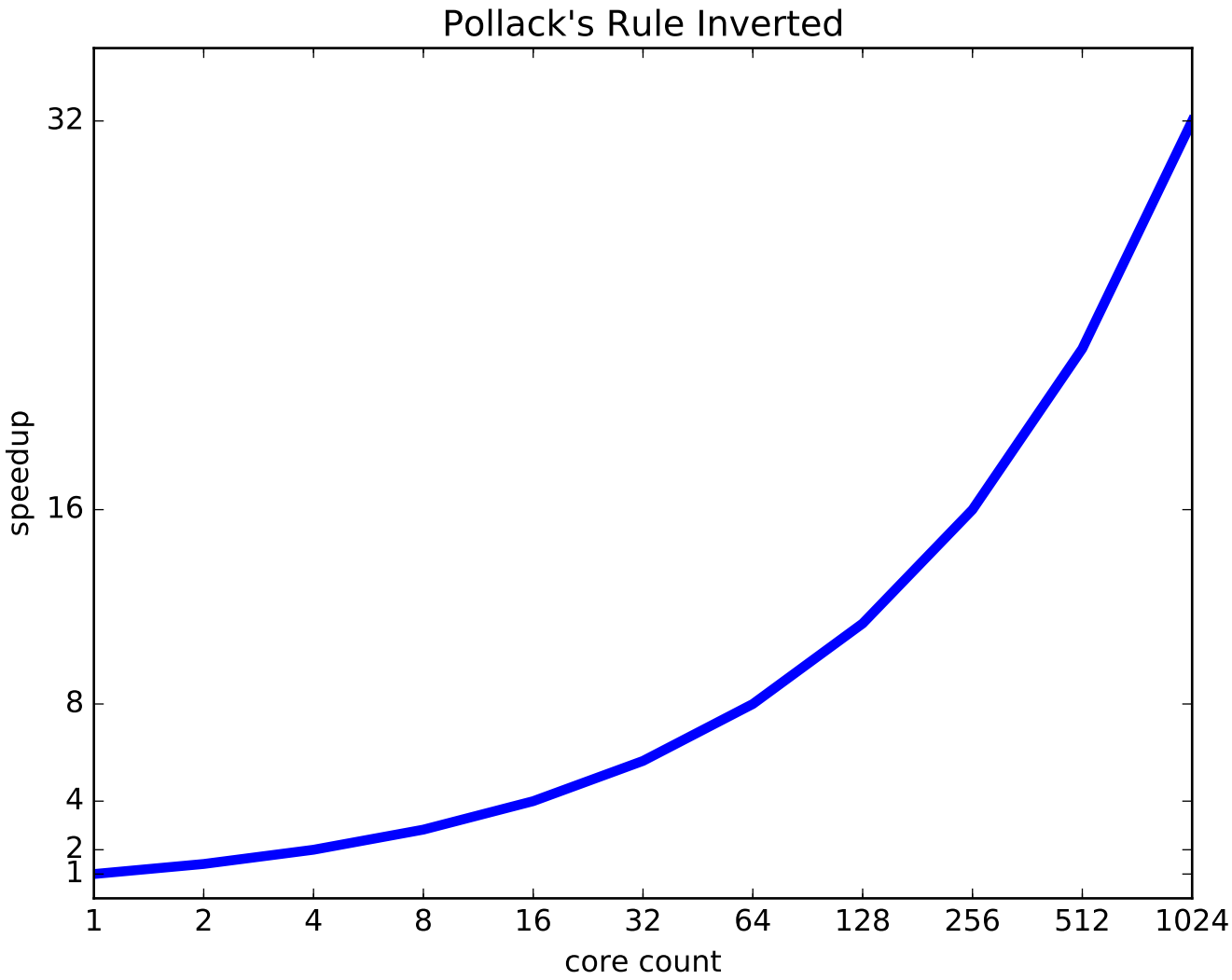
Processor Scaling

- Complexity will increase
- Pollacks Rule - problematic
- Solution: Split processor into cores

Processor Scaling

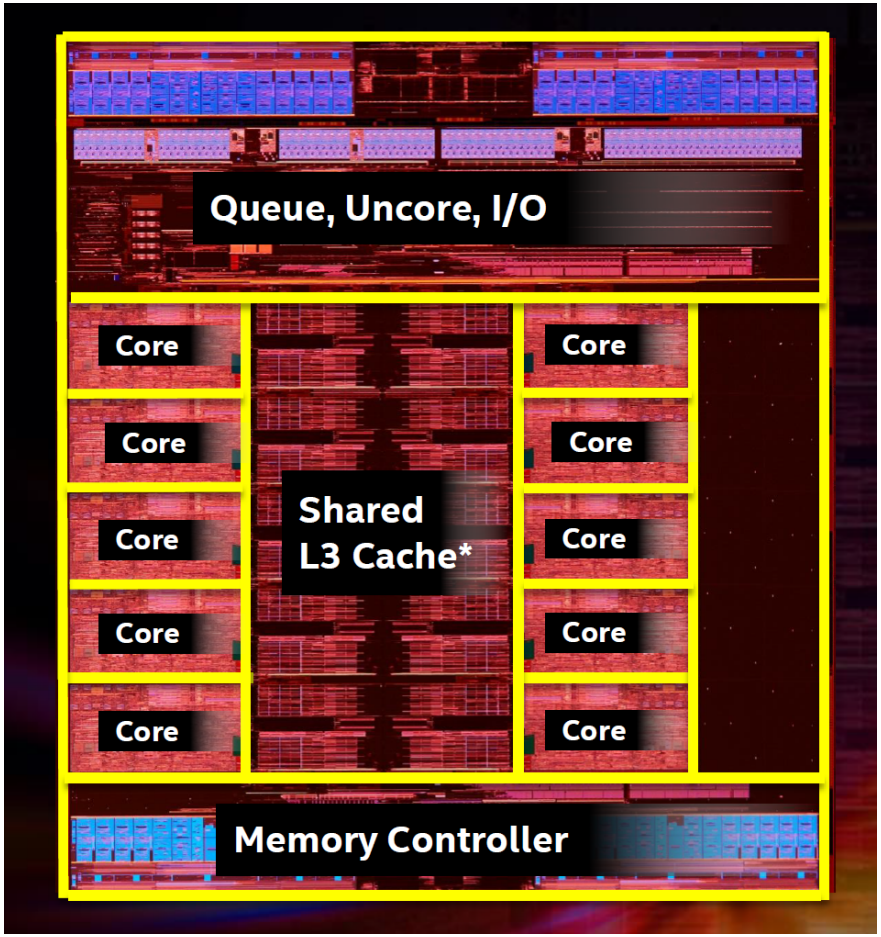
- Complexity will increase
- Pollacks Rule - problematic
- Solution: Split processor into cores
- More cores - also problematic

Pollack's Rule Inverted



Outline

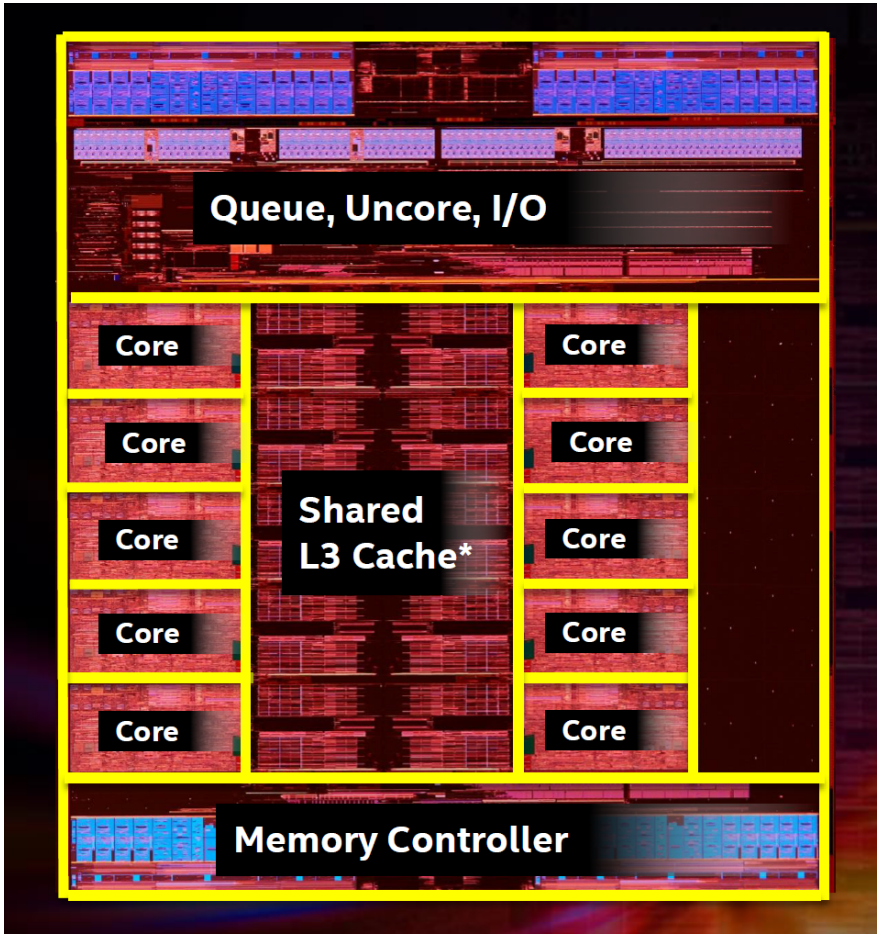
Mainstream cache implementation today



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

- **hardware coherent**

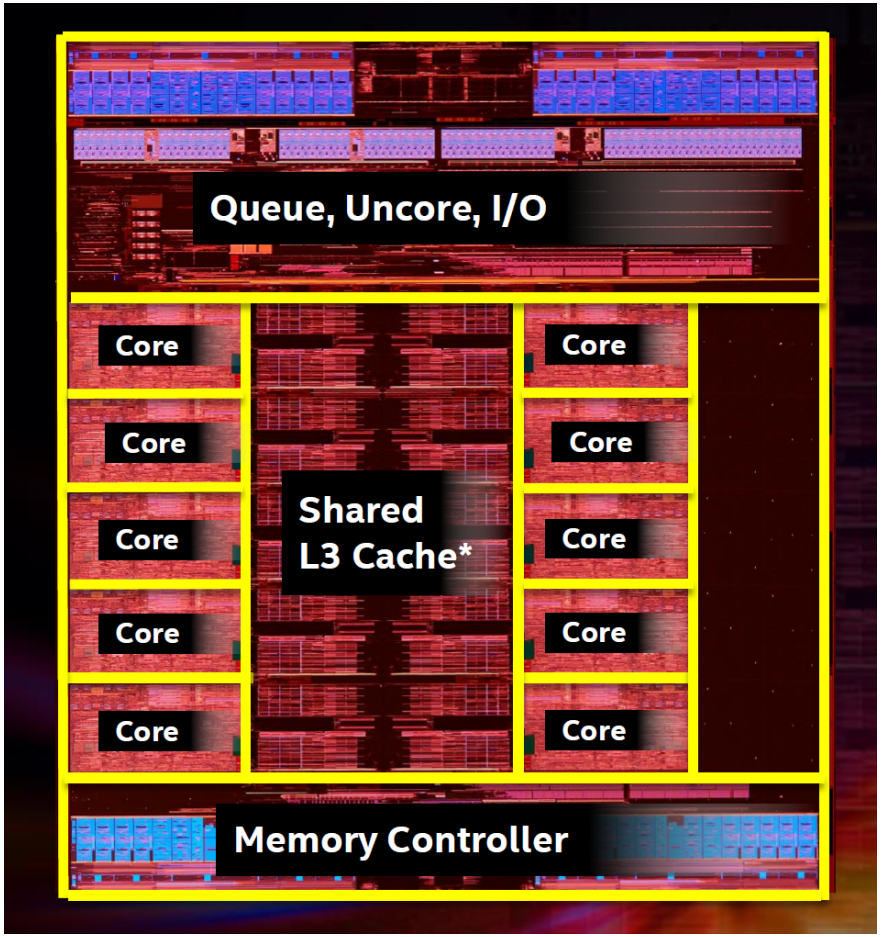
Mainstream cache implementation today



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

- **hardware coherent**
- inclusive

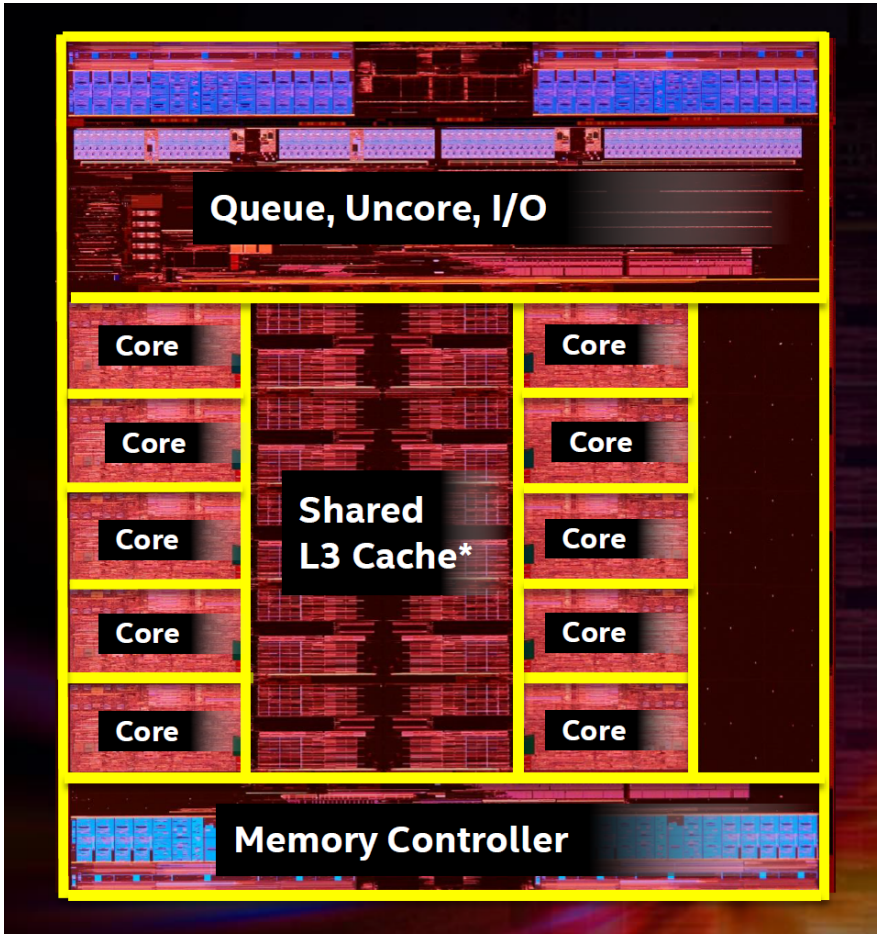
Mainstream cache implementation today



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

- **hardware coherent**
- inclusive
- last level shared

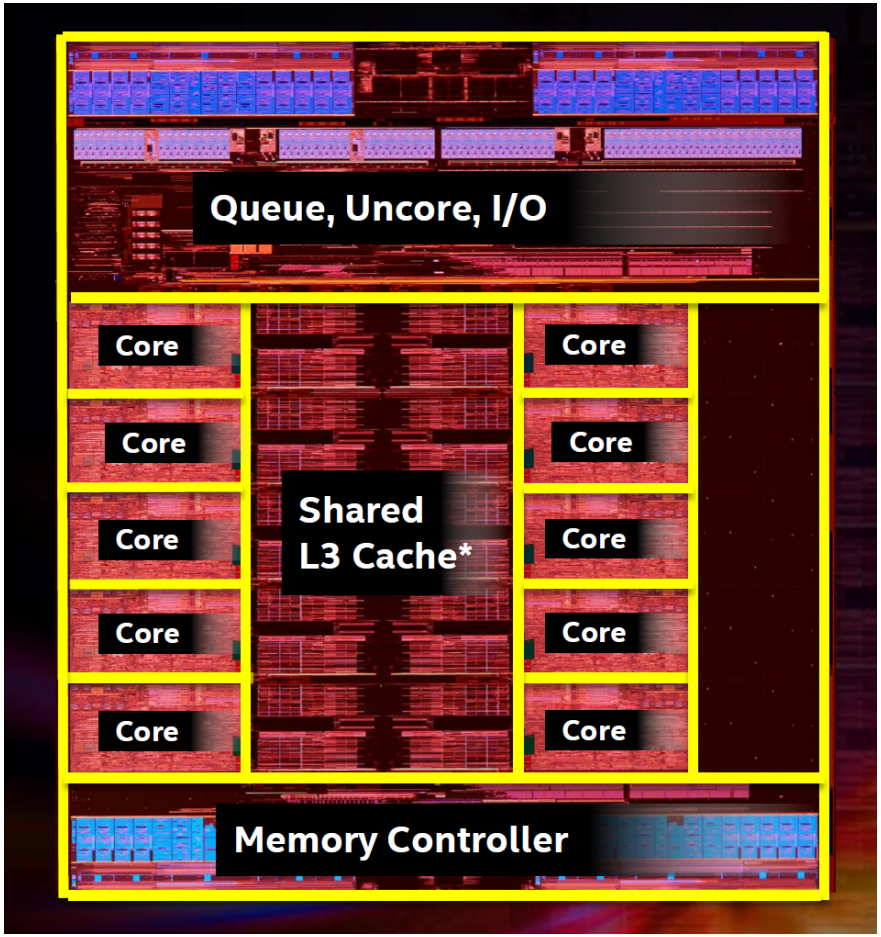
Mainstream cache implementation today



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

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

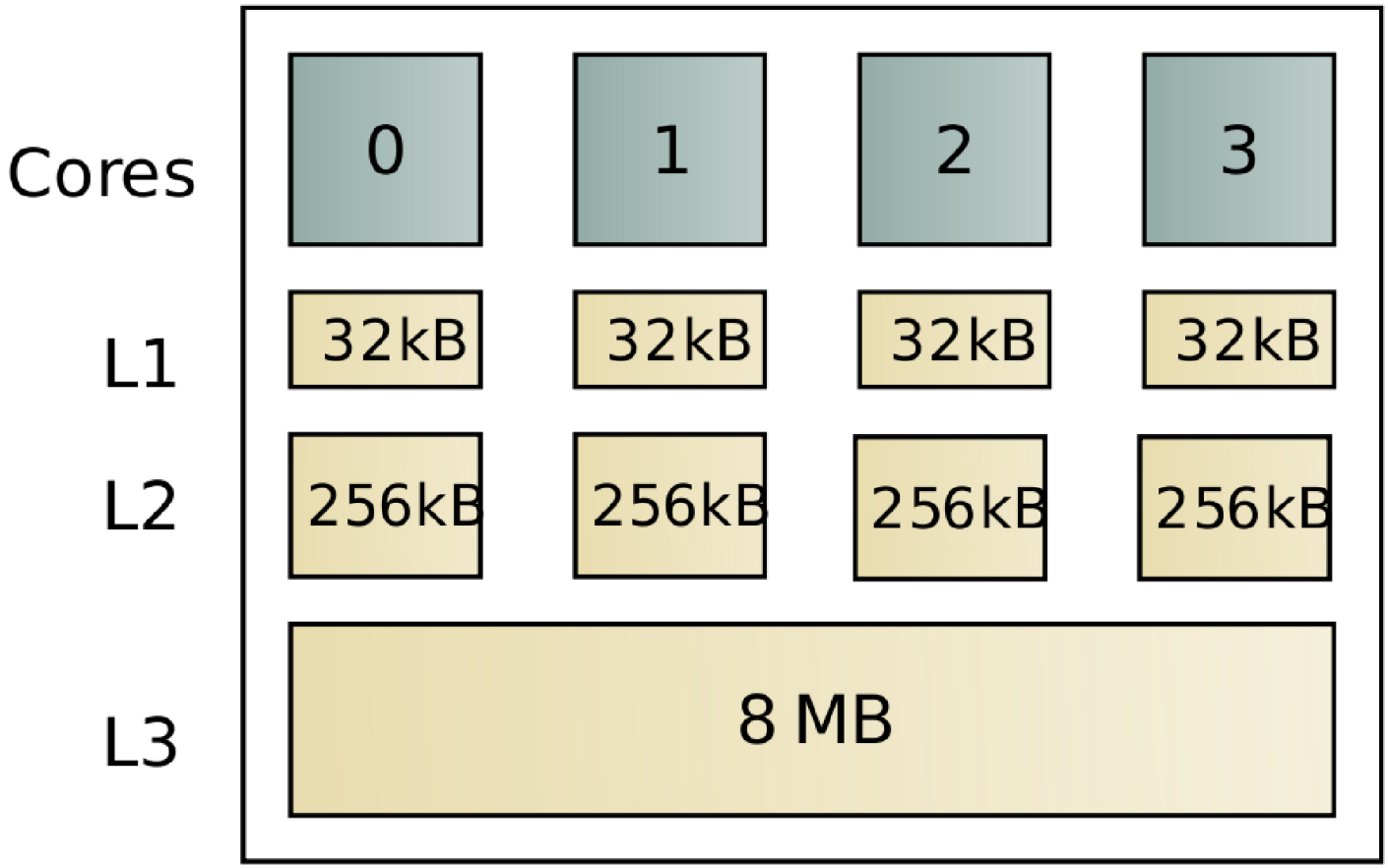
Mainstream cache implementation today



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

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

Mainstream cache implementation today



Cache implementation on Intel i7 CPU (*2*)

Outline

Scaling - cache memory

Today we have typcially..

- *64*Byte per cache line
- *4* cores
- **1** Bit per core for tracking
- overhead below **5%**

Scaling - the naive way

Lets assume..

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

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 ???

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

Scaling - inexact tracking of sharers

Lets assume..

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

Scaling - inexact tracking of sharers

Lets 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

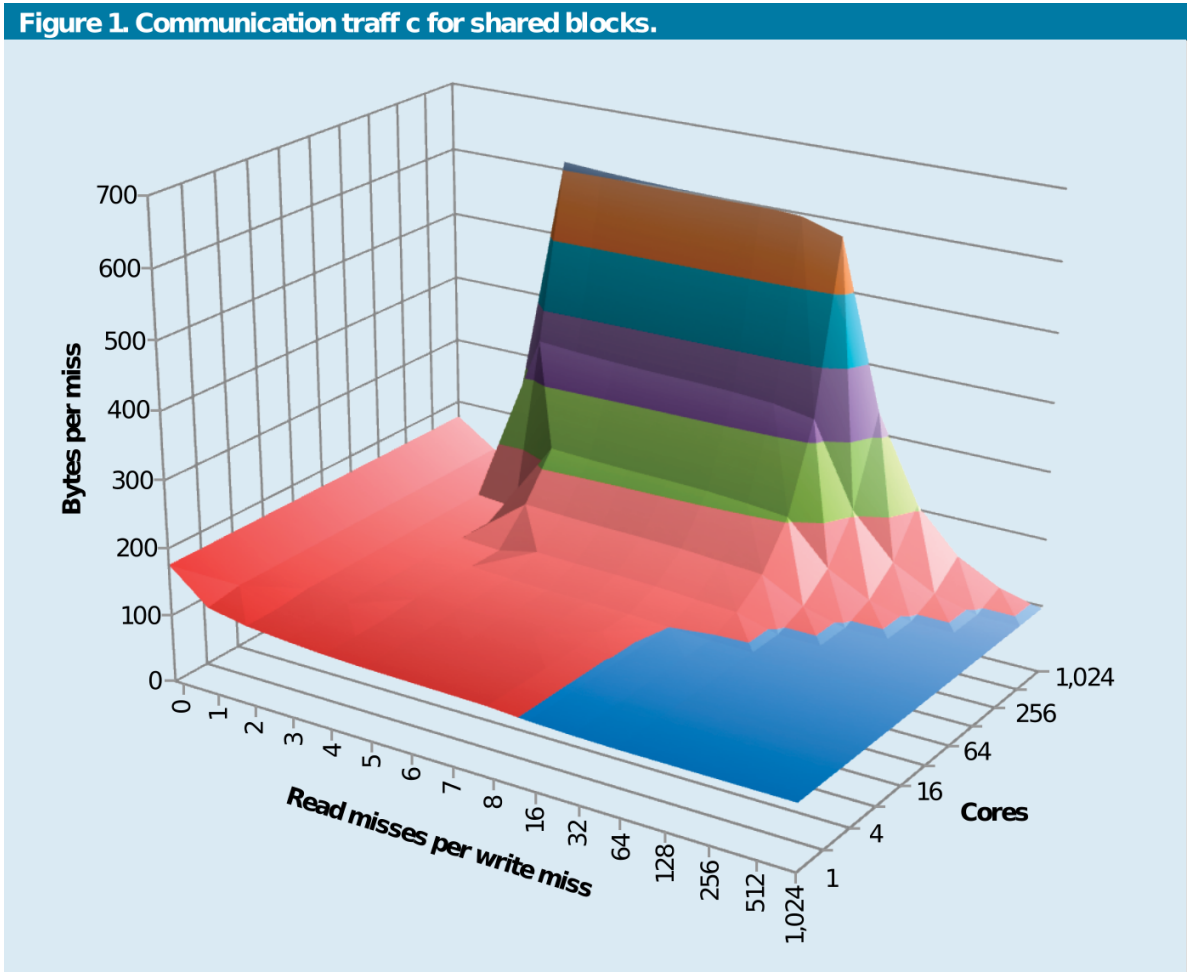
Scaling - inexact tracking of sharers

How it works..

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

High traffic with more than 32 sharers (threads)

Scaling - inexact tracking of sharers



Traffic with inexact tracking of sharers (*3*)

Outline

Scaling - cache hierarchy

Advantages..

- tracking of subsequent clusters only

Scaling - cache hierarchy

Advantages..

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

Scaling - cache hierarchy

Advantages..

- tracking of subsequent clusters only
- access to memory stays transparent
- cache coherent

Scaling - cache hierarchy

Advantages..

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

Scaling - cache hierarchy

Advantages..

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

Scaling - cache hierarchy

Advantages..

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

Disadvantages..

Scaling - cache hierarchy

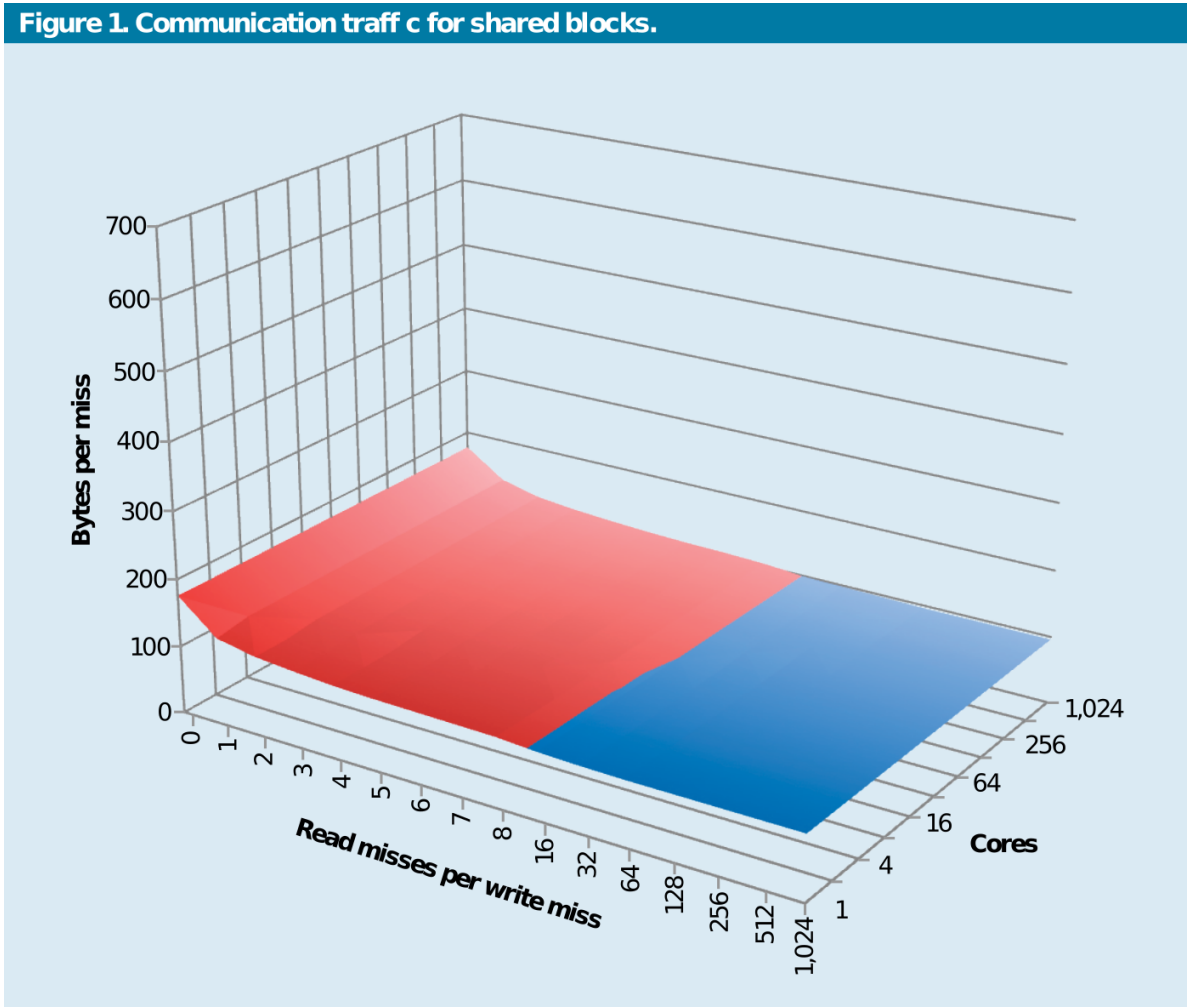
Advantages..

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

Disadvantages..

- higher latency between distant cores

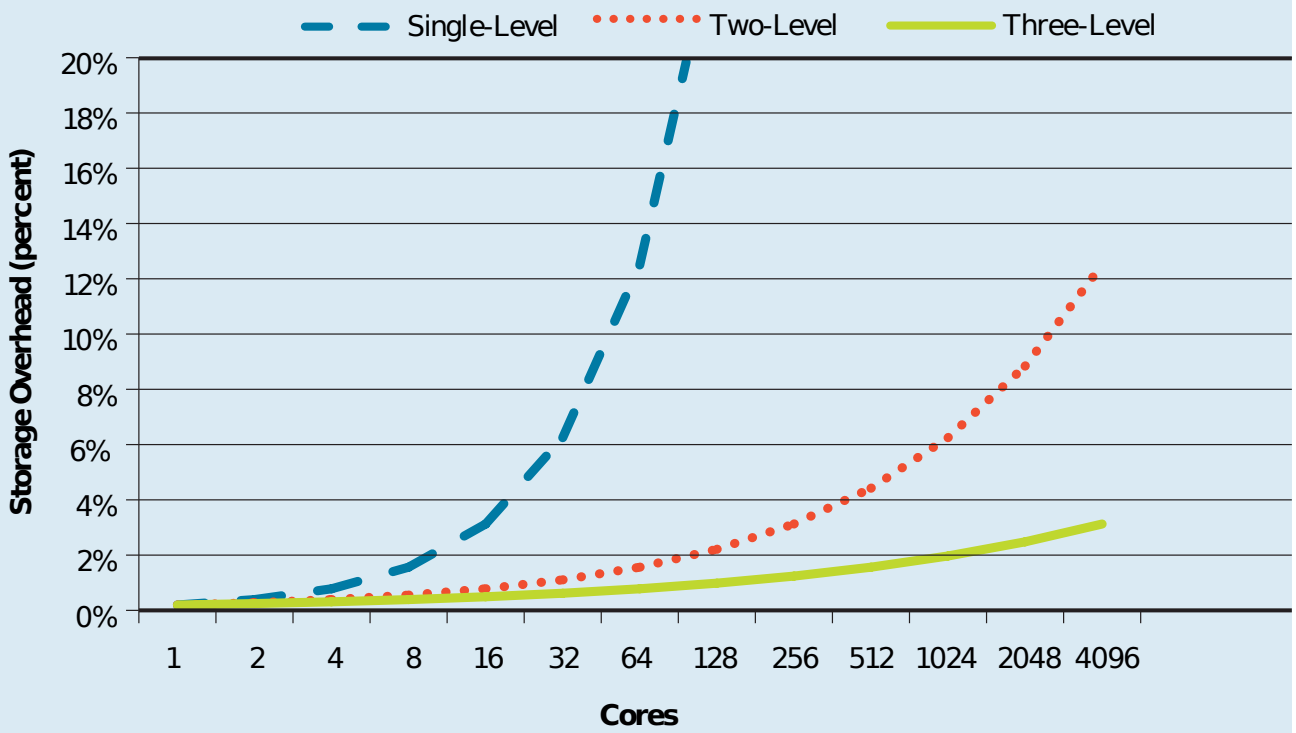
Scaling - cache hiearchy



Traffic with exact tracking of sharers (*3*)

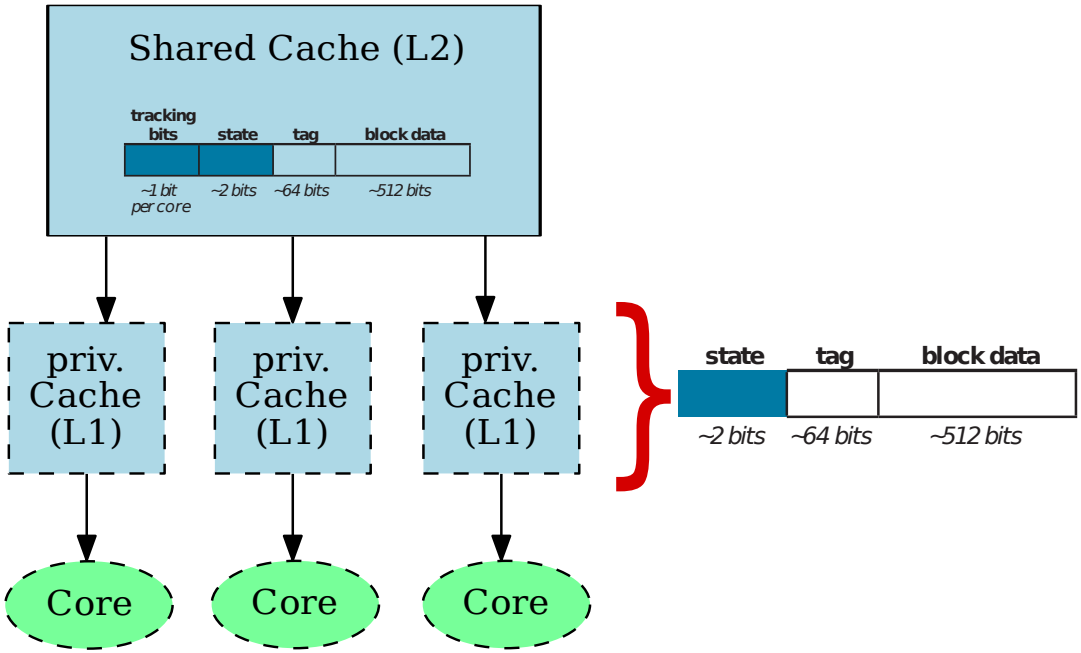
Scaling - cache hierarchy

Figure 3. storage overhead in shared caches.



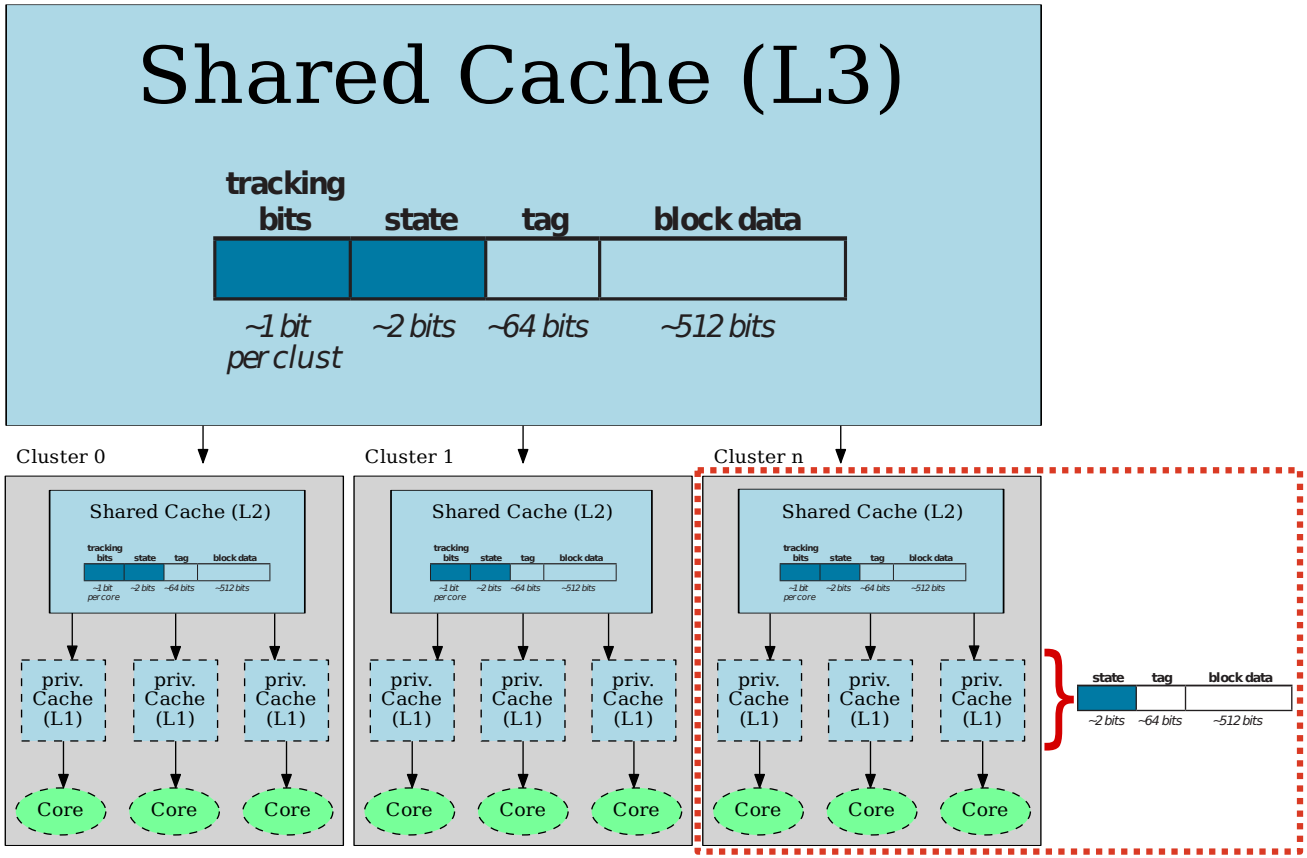
Storage Overhead for multiple hierarchy levels (*3*)

Scaling - cache hierarchy: single-level shared



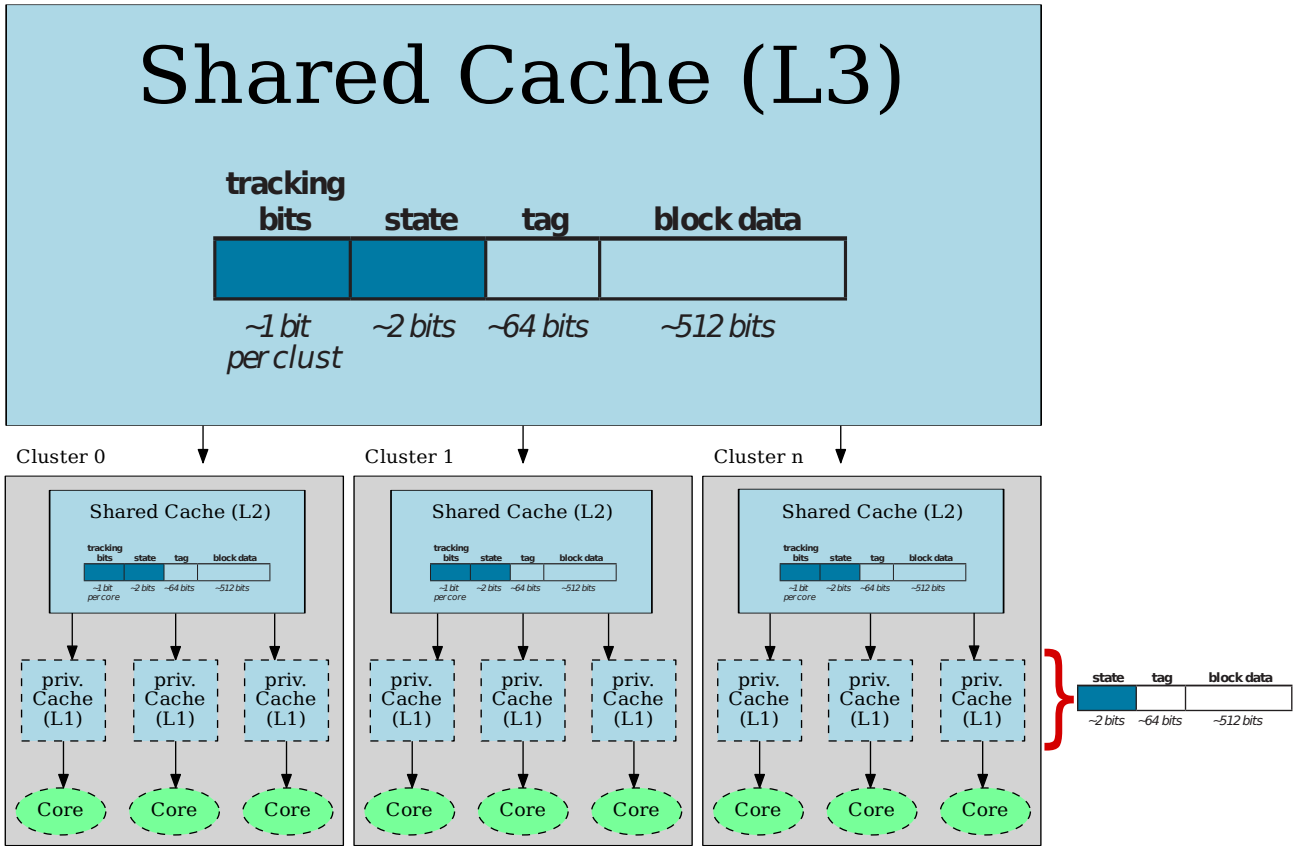
Shared and private caches with 2 levels

Scaling - cache hierarchy: two-level shared



Shared and private caches with 3 levels

Scaling - cache hierarchy: two-level shared



Shared and private caches with 3 levels

Scaling - cache memory in transistors

Mainstream processor:
i7 6700K..

- *4* cores
- *one-level* hierarchy
- *256 KiB* per L2
- *8* times of aggregate
- $\mathbf{1.75} * 10^9$ 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

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

Lets 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

Lets 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

Lets 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**

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..

- **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

Lets assume..

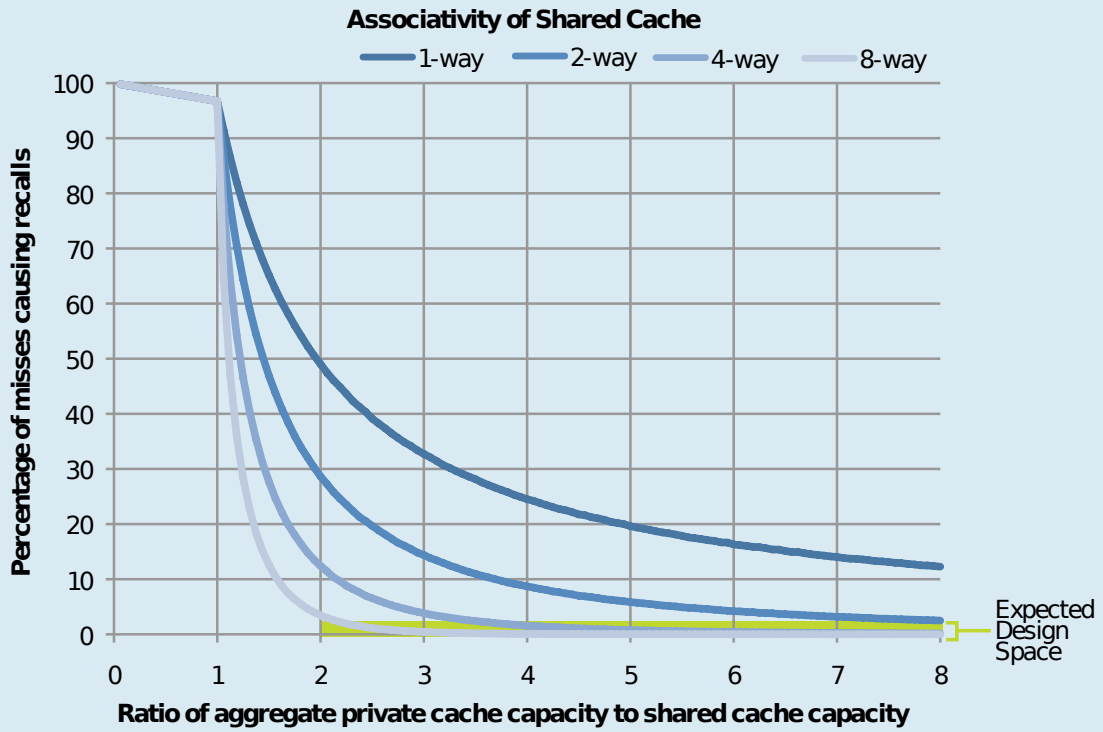
- *1024* cores
- *two-level* hierarchy
- *6* transistors per flip-flop
- *256 KiB* per L1
- *8* times of aggregate

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**
- $\mathbf{824} * 10^9$ transistors

Scaling - cache memory

Figure 4. Likelihood a shared cache miss triggers a recall.



Associativity VS capacity (*3*)

References


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 15-01-2017], 2016.


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).