Undersubscribed Threading on Clustered Cache Architectures

Wim Heirman[1,2] Trevor E. Carlson[1] Kenzo Van Craeynest[1]
Ibrahim Hur[2] Aamer Jaleel[2] Lieven Eeckhout[1]

[1] Ghent University
[2] Intel Corporation

HPCA 2014, Orlando, FL

CONTEXT

- Many-core processor with 10s-100s of cores
  - E.g. Intel Xeon Phi, Tilera, GPGPU
- Running scalable, data-parallel workloads
  - SPEC OMP, NAS Parallel Benchmarks, …
- Processor design @ fixed area/power budget
  - Spend on cores, caches?
  - Cache topology?

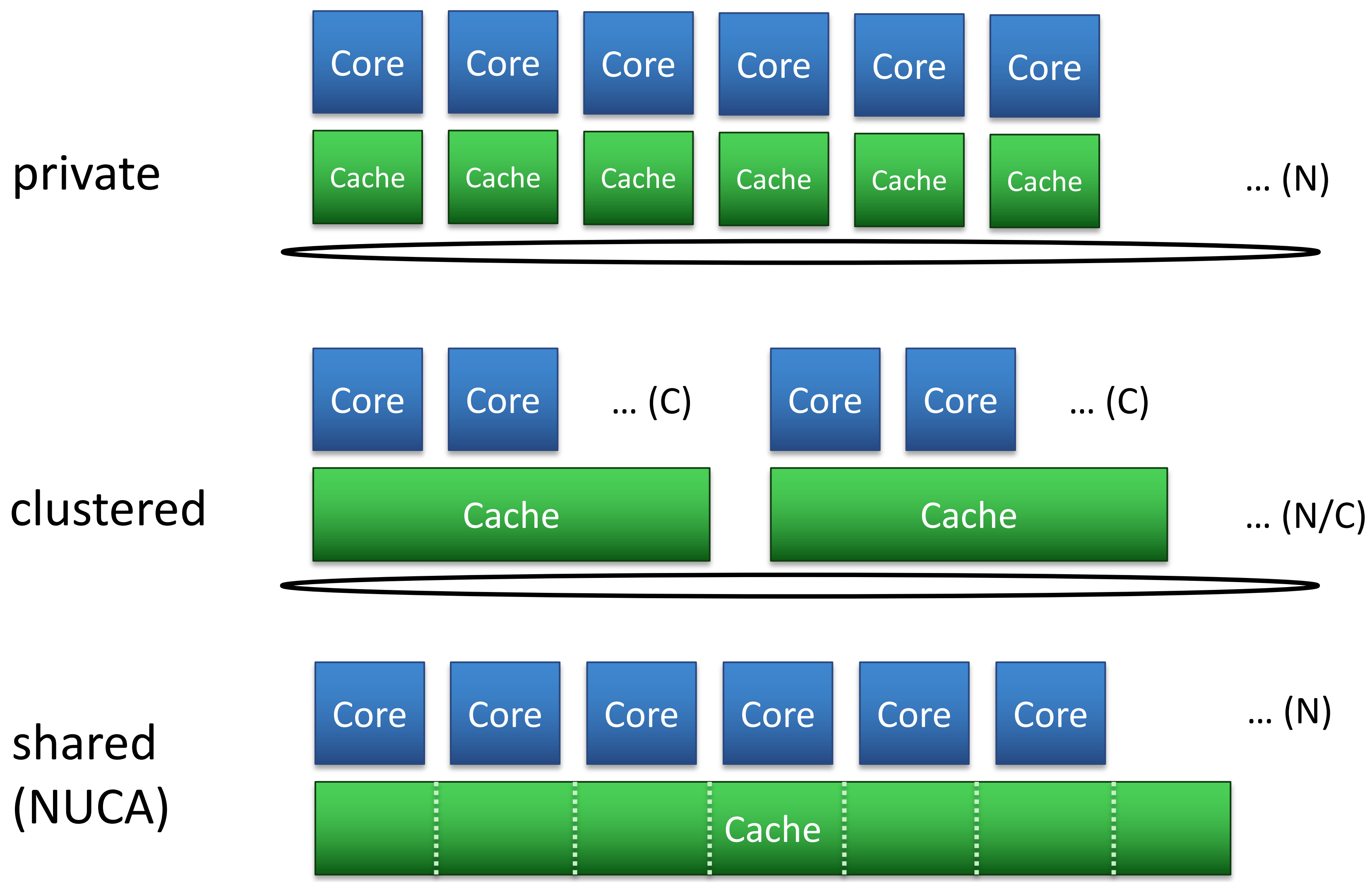
Overview

- Cache topology:
  - Why clustered caches?
  - Why undersubscription?

- Dynamic undersubscription
  - CRUST algorithms for automatic adaptation

- CRUST and future many-core design

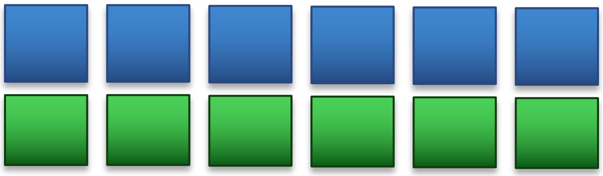
Many-core Cache Architectures



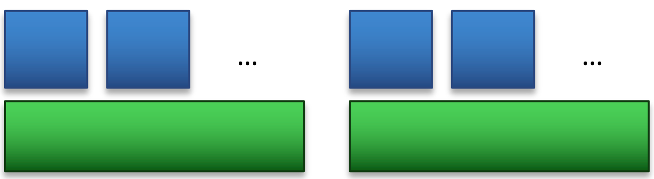
Many-core Cache Architectures

	hit latency	sharing
private	++	✗
clustered	+	+
shared	✗	++

private

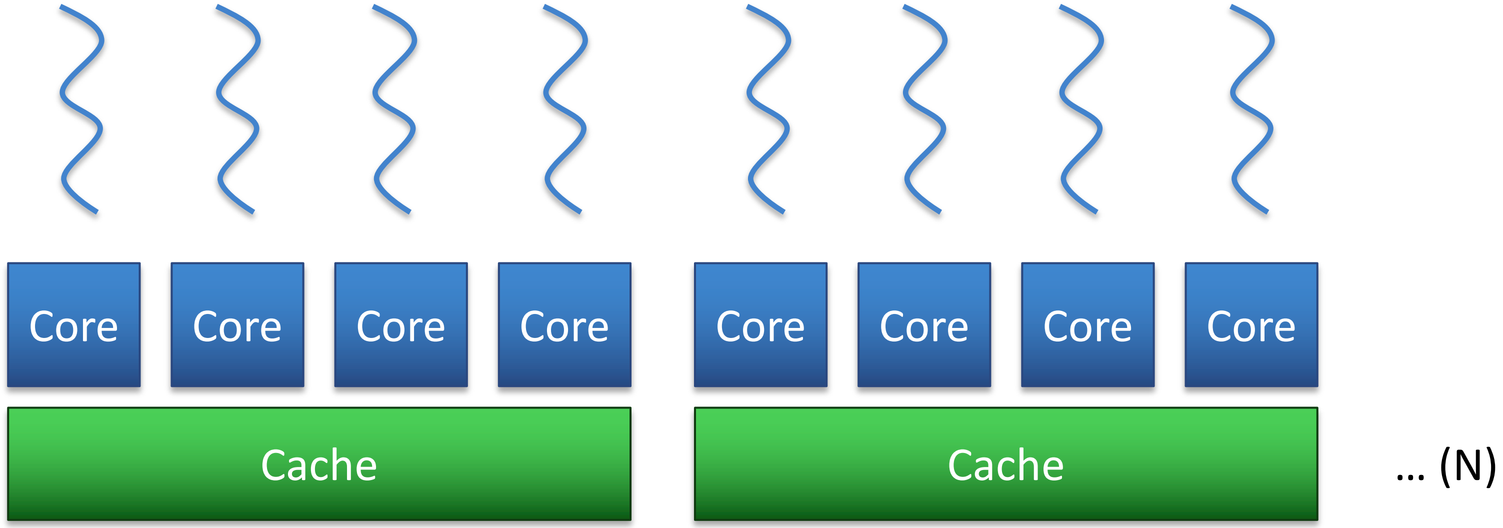


clustered



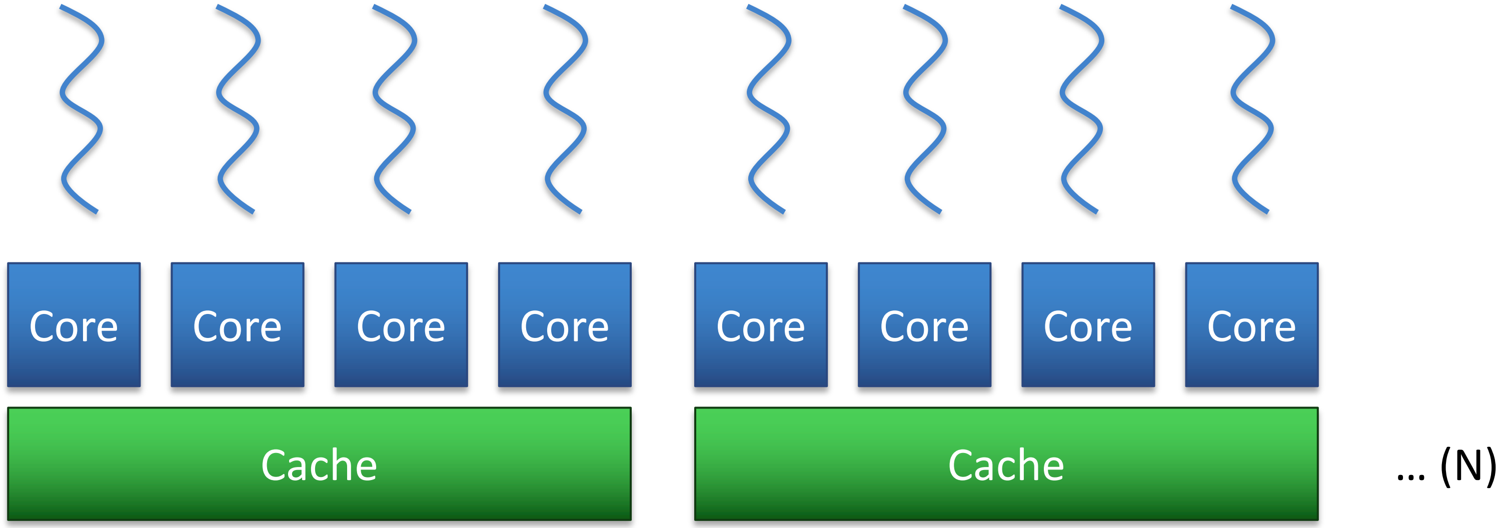
shared

UNDERSUBSCRIBING FOR CACHE CAPACITY



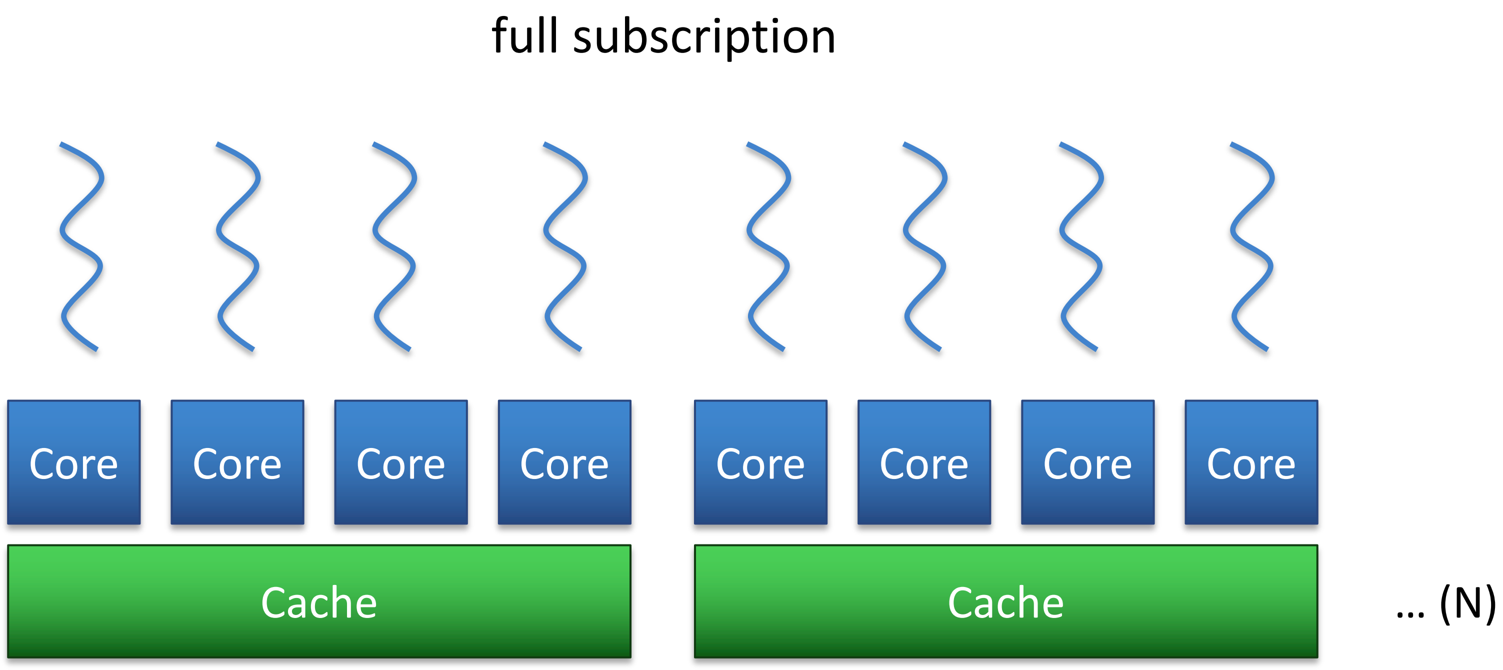
UNDERSUBSCRIBING FOR CACHE CAPACITY

- Less than C active cores/threads per cluster



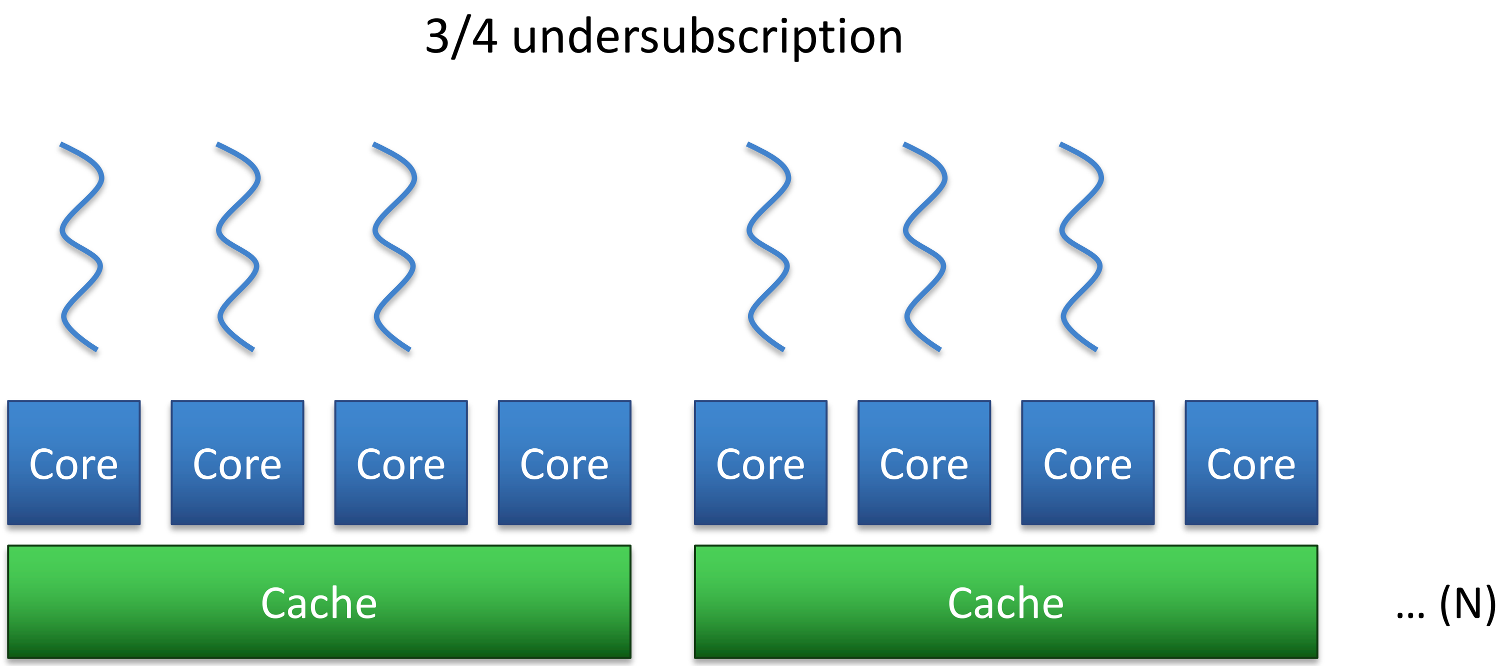
Undersubscribing for Cache Capacity

- Less than C active cores/threads per cluster



UNDERSUBSCRIBING FOR CACHE CAPACITY

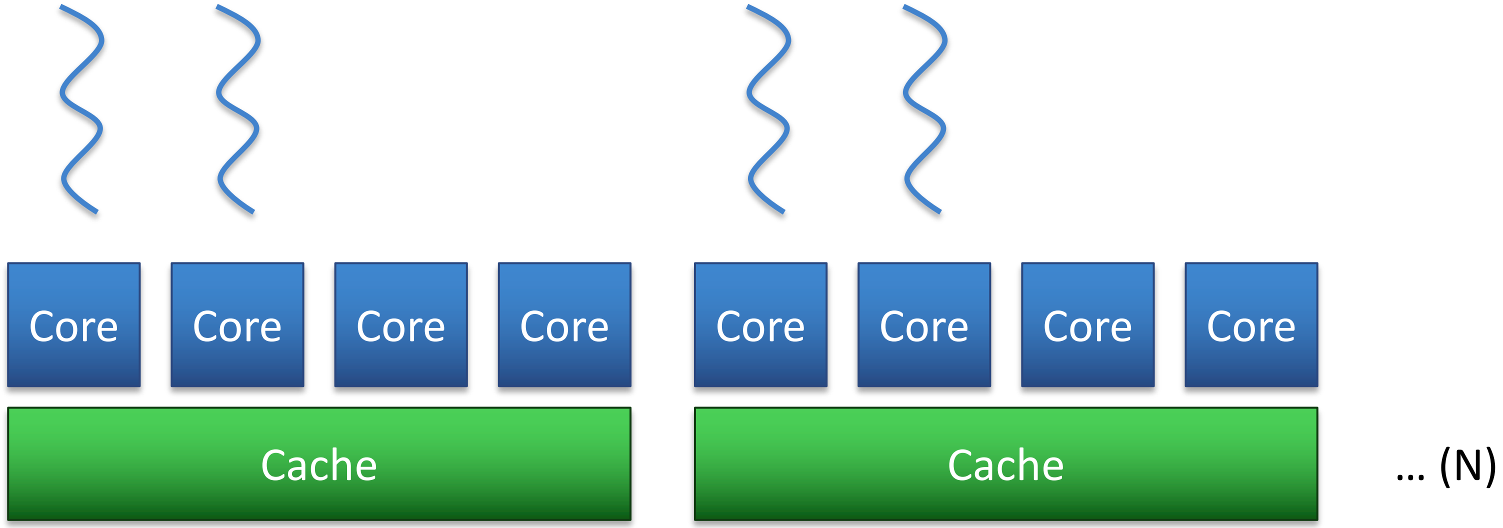
- Less than C active cores/threads per cluster



Undersubscribing for Cache Capacity

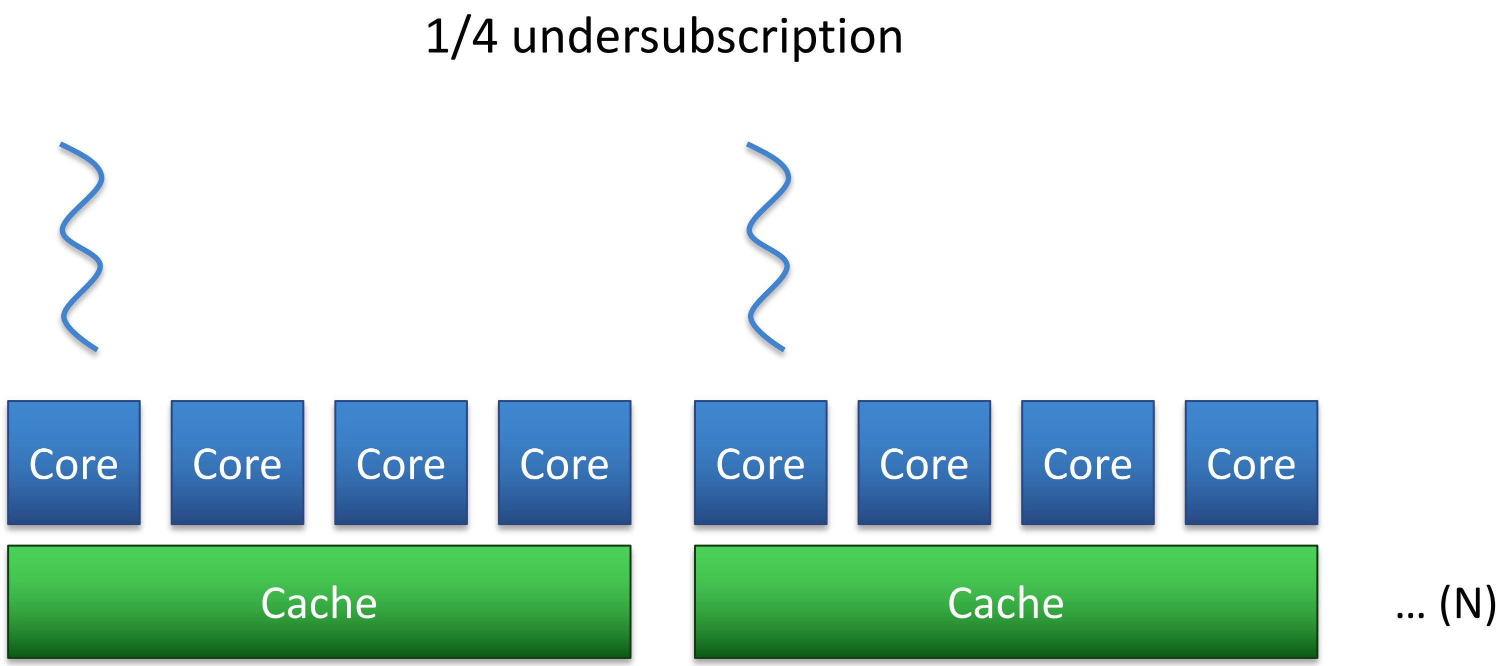
- Less than C active cores/threads per cluster

2/4 undersubscription



UNDERSUBSCRIBING FOR CACHE CAPACITY

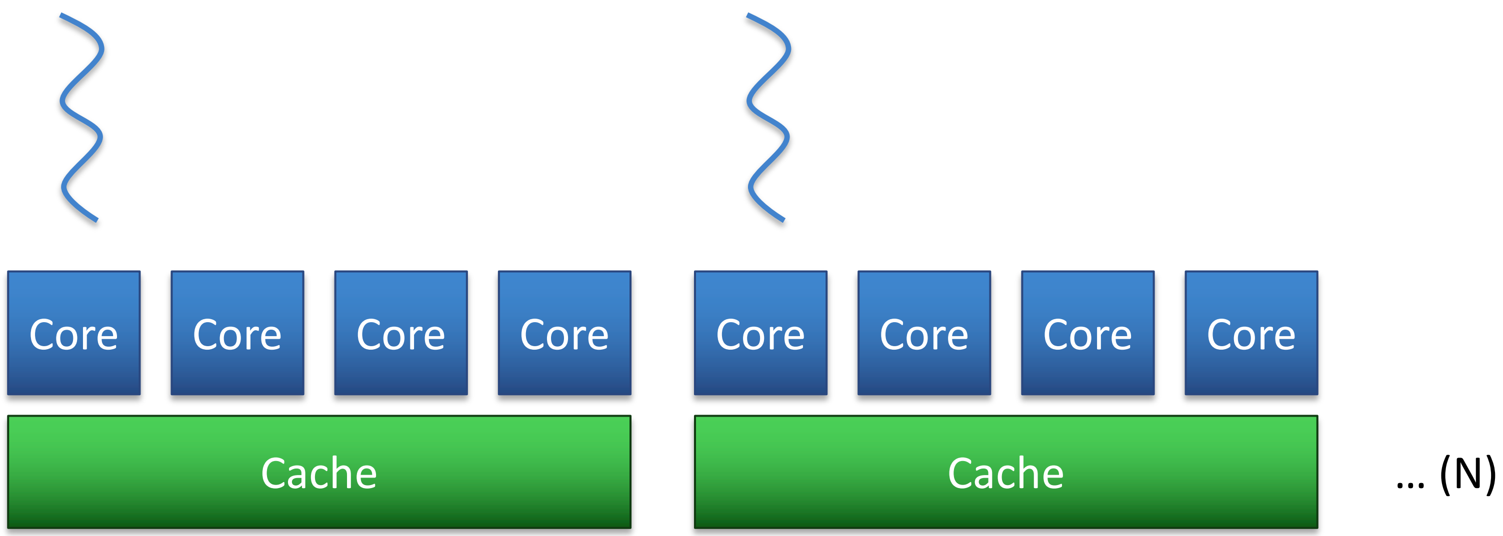
- Less than C active cores/threads per cluster



Undersubscribing for Cache Capacity

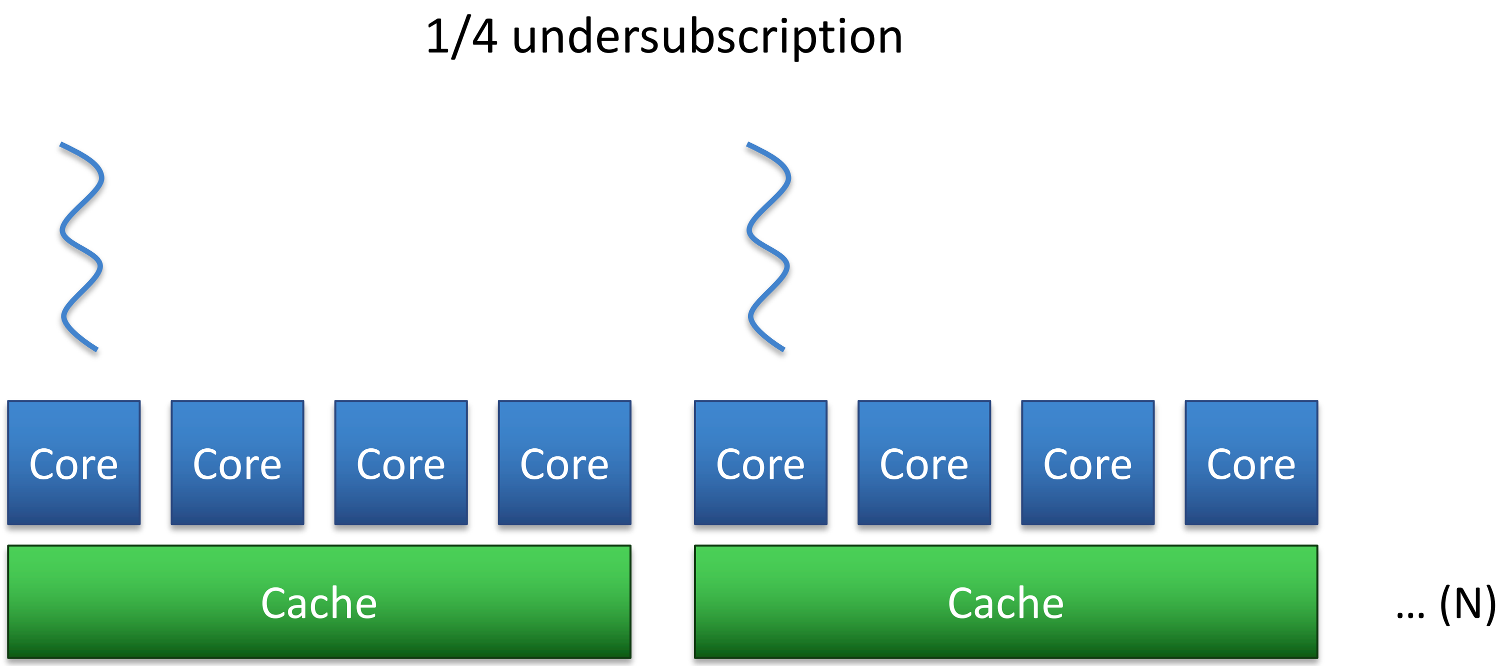
- Less than C active cores/threads per cluster
- When working set does not fit in cache

1/4 undersubscription

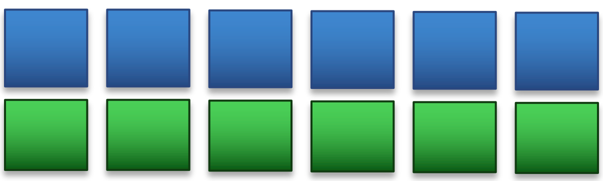
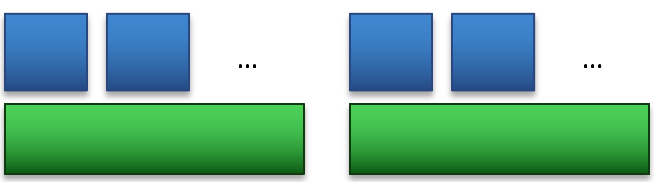


UNDERSUBSCRIBING FOR CACHE CAPACITY

- Less than C active cores/threads per cluster
- When working set does not fit in cache
- *Keep all cache capacity accessible*



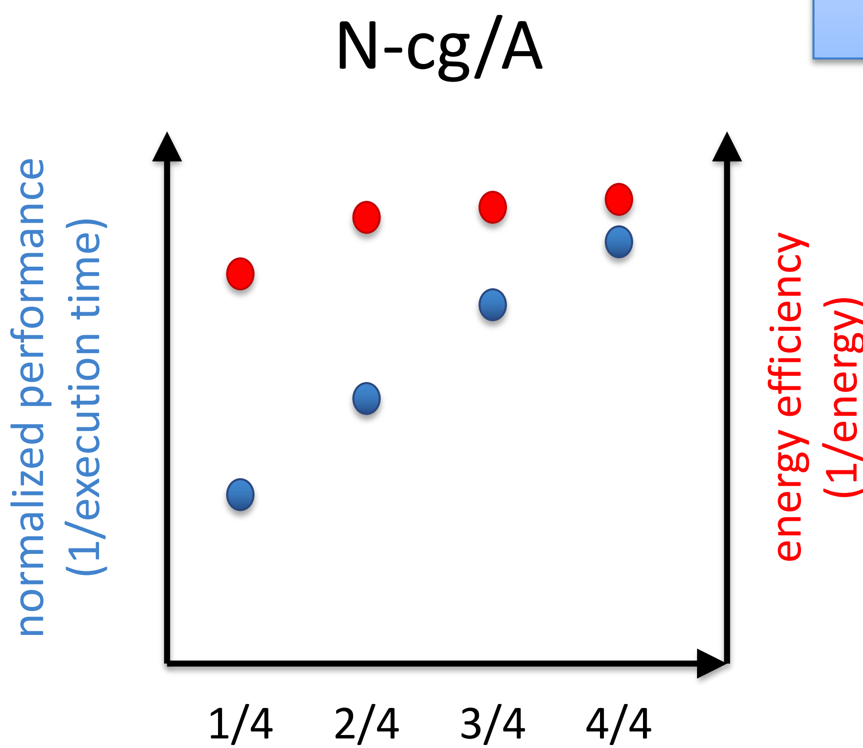
Many-core Cache Architectures

	hit latency	sharing	undersubscription	
private 	++	✗	✗	(1:1)
clustered 	+	+	+	(1:C)
shared 	✗✗	++	+	(1:N)

Performance & Energy: Working set vs. Cache size

Baseline architecture:

- 128 cores
- private L1
- clustered L2
  1M shared per 4 cores



Performance & Energy: Working set vs. Cache size

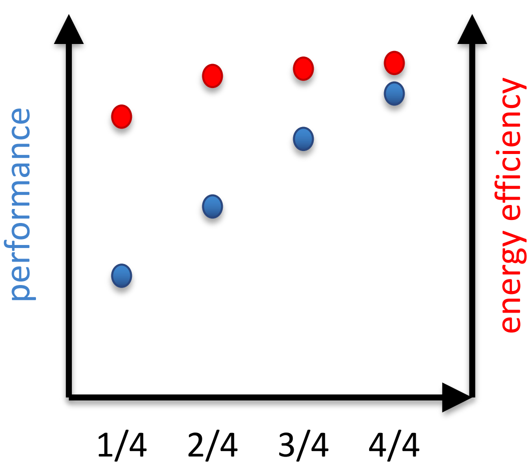
Compute bound:
use all cores for
highest performance

Bandwidth bound:
disable cores for better
energy efficiency

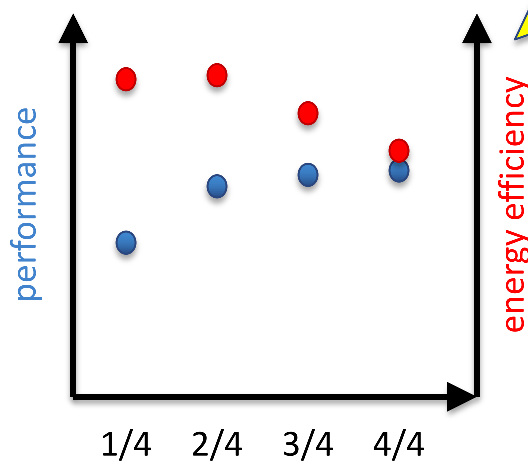
Capacity bound:
reduce thread count
to optimize hit rate

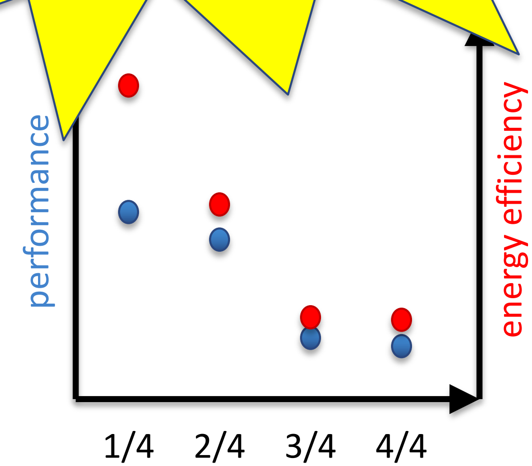
1/4 undersubscription:
3.5x performance
80% energy savings

N-cg/A



N-cg/C

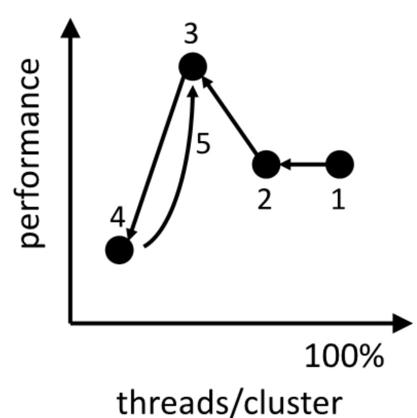




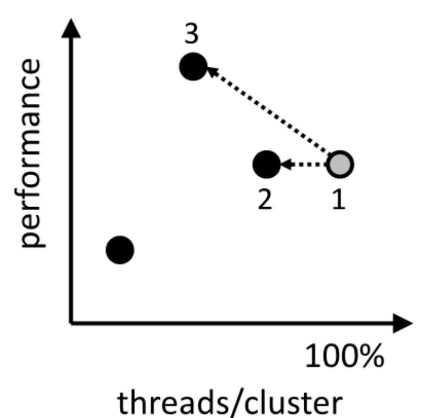
CLUSTER-AWARE UNDERSUBSCRIBED SCHEDULING OF THREADS (CRUST)

- Dynamic undersubscription
- Integrated into the OpenMP runtime library
  - Adapt to each *#pragma omp* individually
- Optimize for performance first, save energy when possible
  - Compute bound: full subscription
  - Bandwidth bound: no* performance degradation (* <5% vs. full)
  - Capacity bound: highest performance
- Two CRUST heuristics (*descend* and *predict*) for on-line adaptation

CRUST-descend

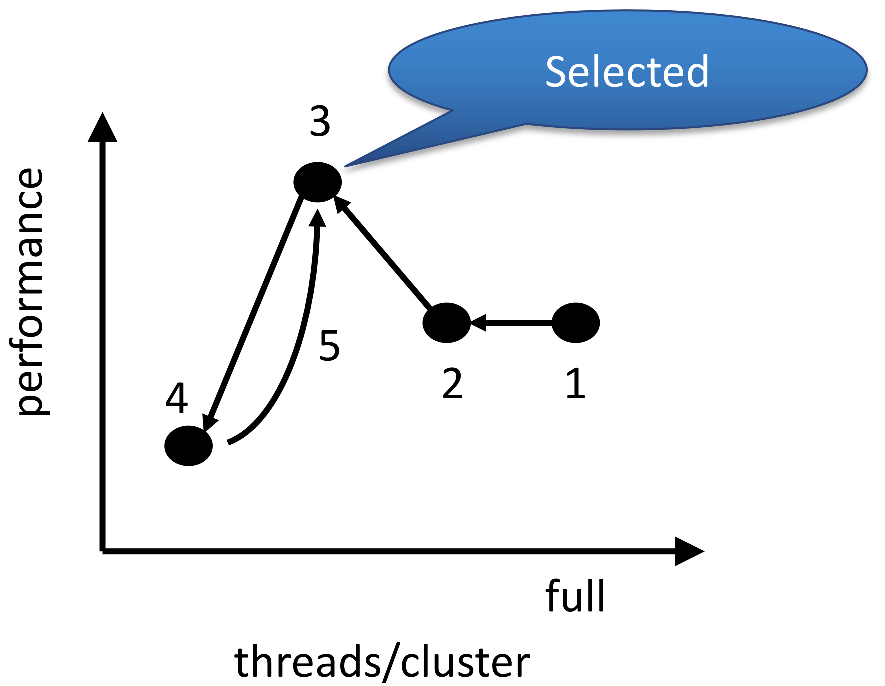


CRUST-predict



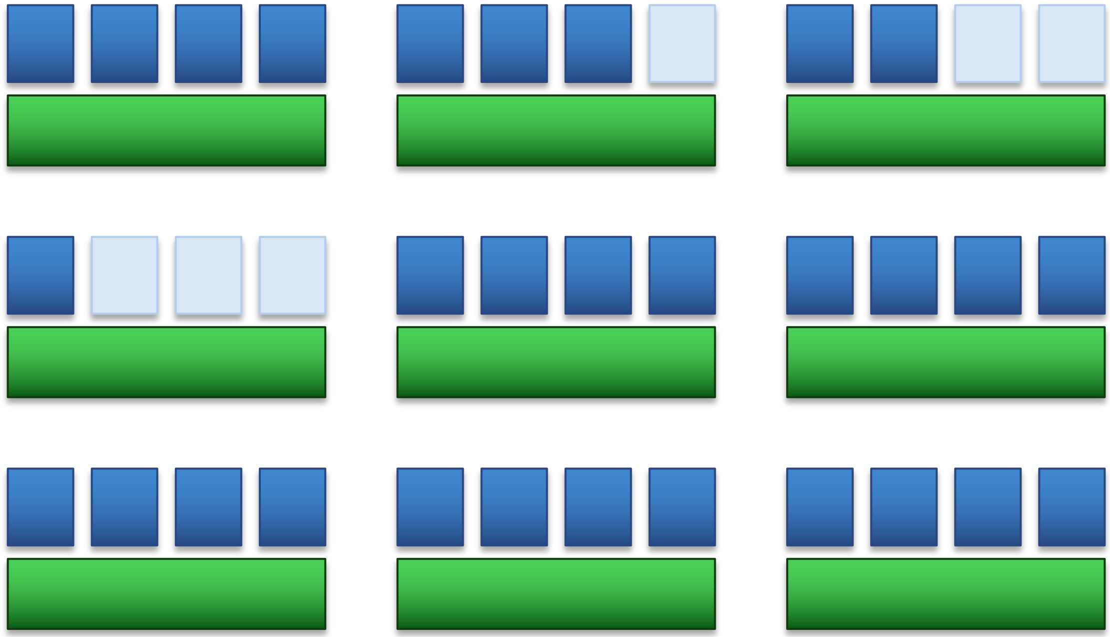
CRUST-DESCEND

- Start with full subscription
- Reduce thread count while performance increases



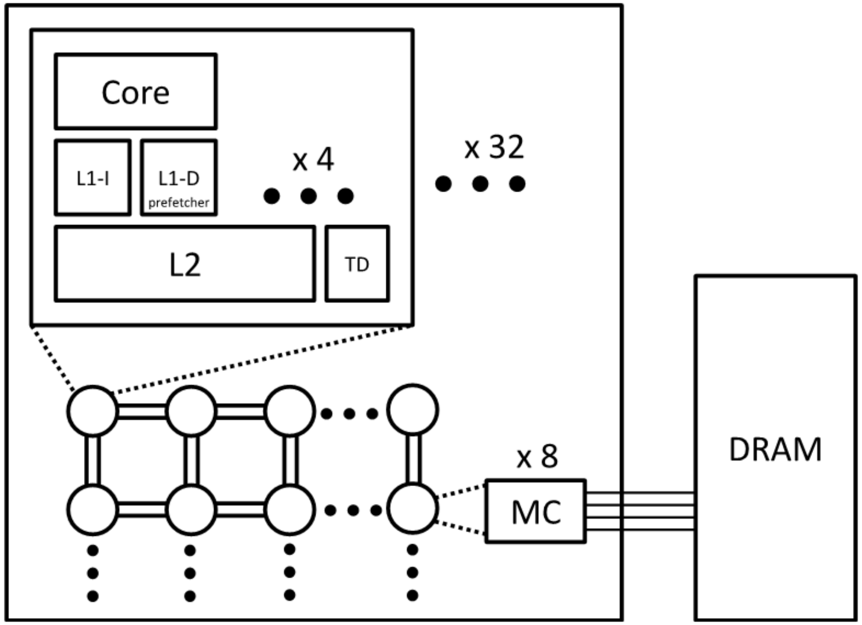
CRUST-PREDICT

- Reduce number of steps required by being smarter
- Start with heterogeneous undersubscription
  - Measure LLC miss rate for each thread/cluster option
  - Predict performance of each option using PIE-like model
- Select best predicted option



METHODOLOGY

- Generic many-core architecture
  - 128 cores, 2-issue OOO @1GHz
  - 2x 32 KB private L1 I+D
  - L1-D stride prefetcher
  - **1 MB shared L2 per 4 cores**
  - 2-D mesh NoC
  - 64 GB/s total DRAM bandwidth

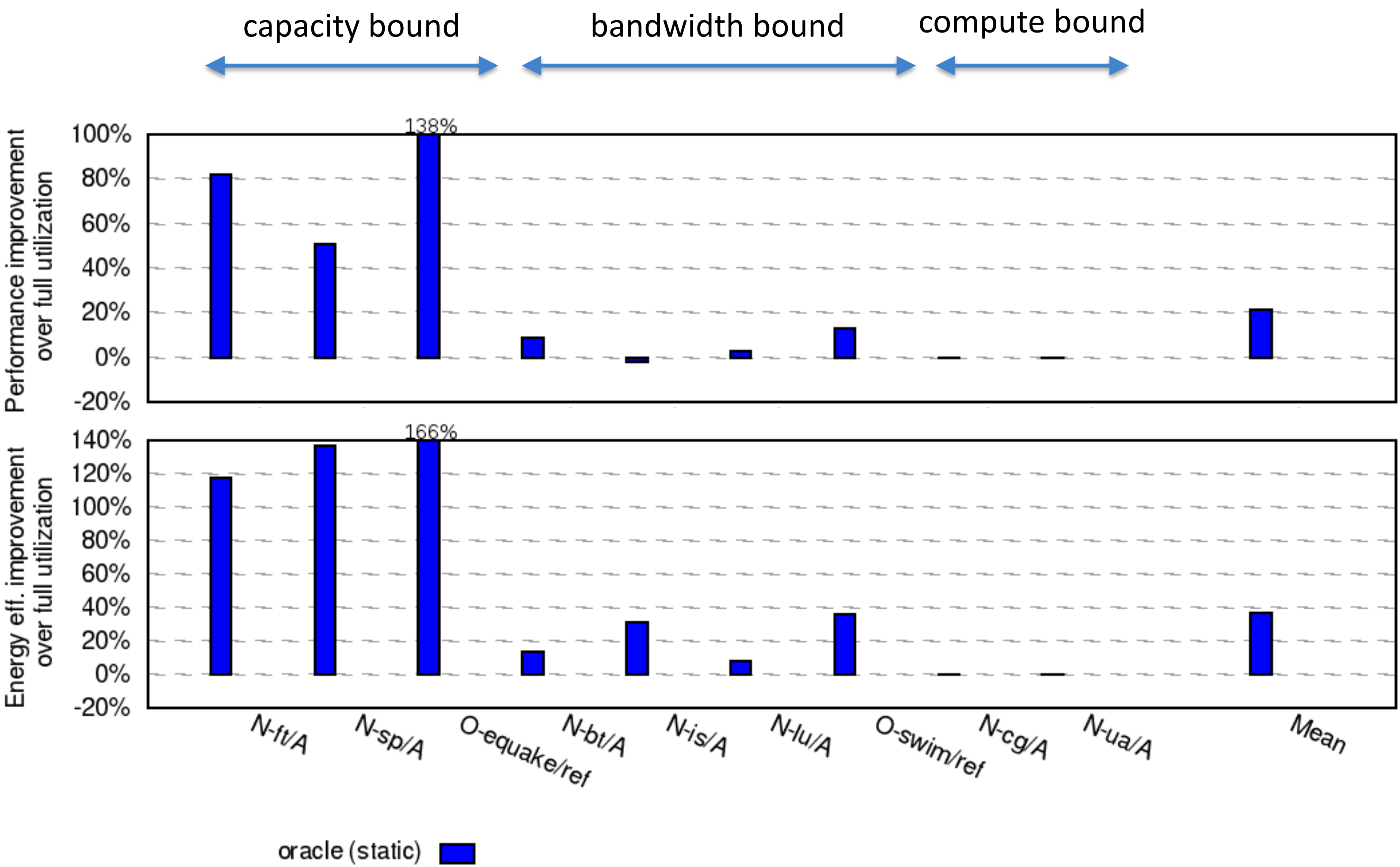


- Sniper simulator, McPAT for power

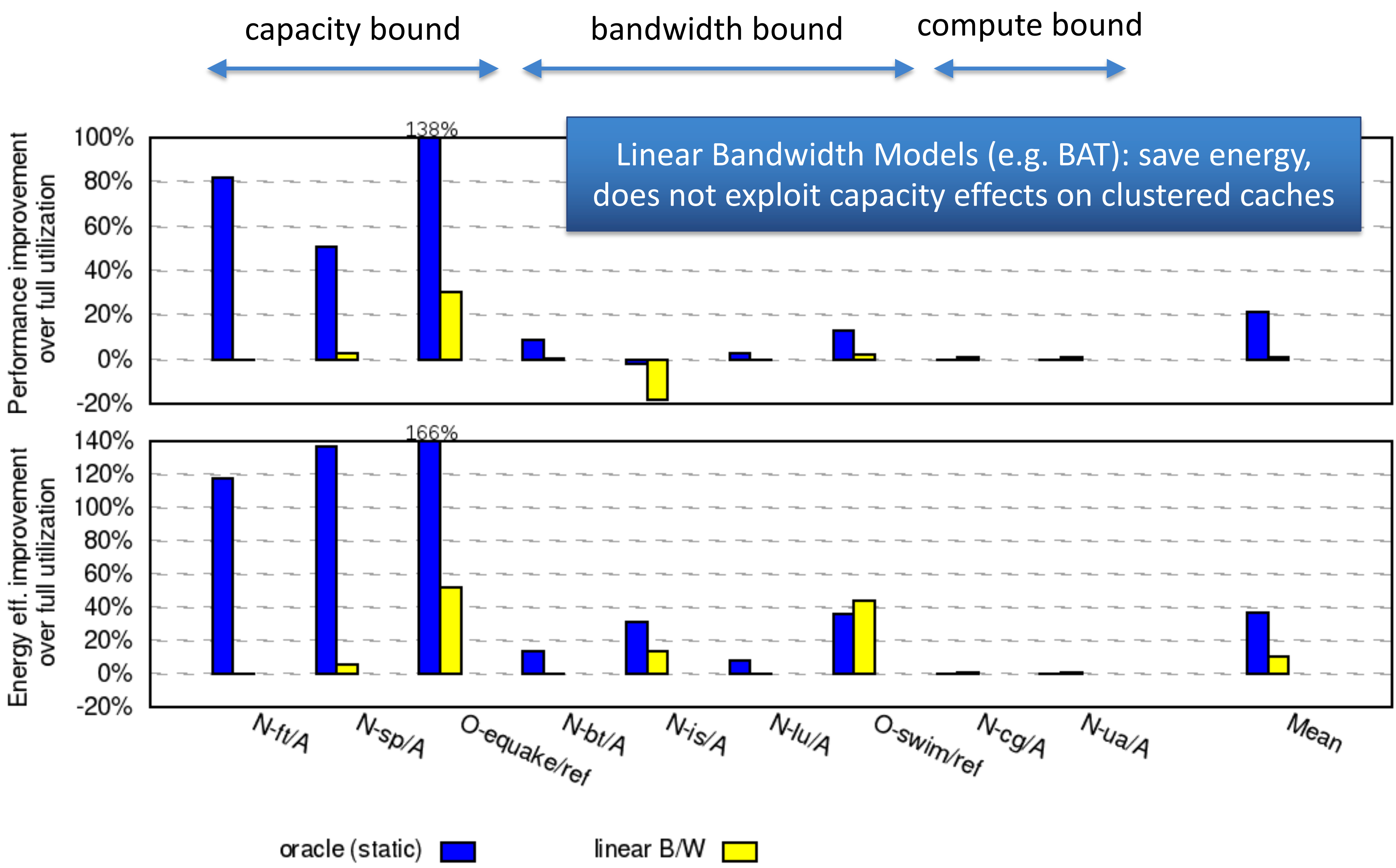
snipersim.org

- SPEC OMP and NAS parallel benchmarks
  - Reduced iteration counts from ref, class A inputs

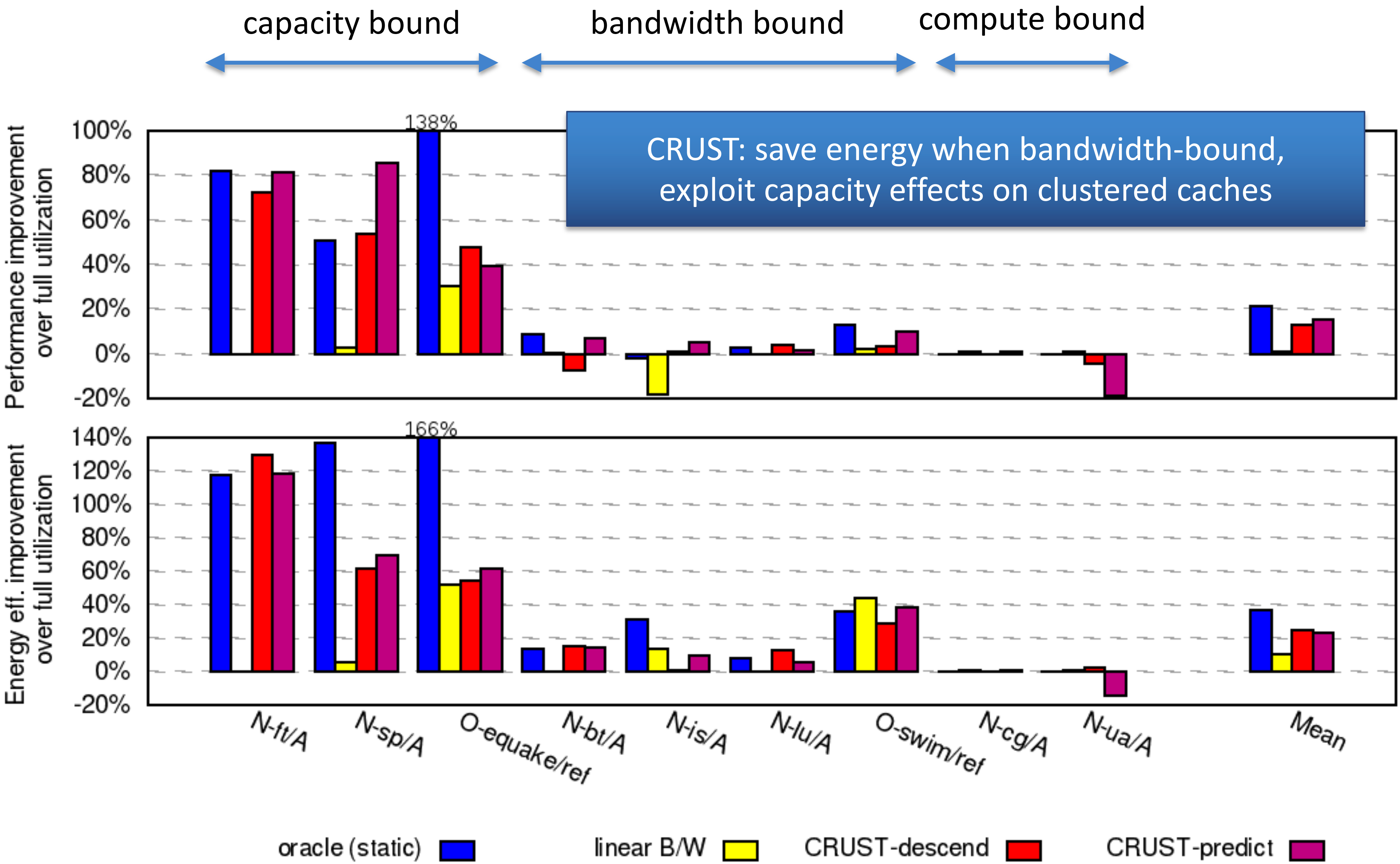
RESULTS: ORACLE (STATIC)



RESULTS: LINEAR BANDWIDTH MODELS



RESULTS: CRUST



CRUST: save energy when bandwidth-bound, exploit capacity effects on clustered caches

UNDERSUBSCRIPTION VS. FUTURE DESIGNS

- Finite chip area, spent on cores or caches
  - *Increasing max. compute* vs. *keeping cores fed with data*
- Undersubscription can adapt workload behavior to the architecture

Does this allow us to build a higher-performance design?

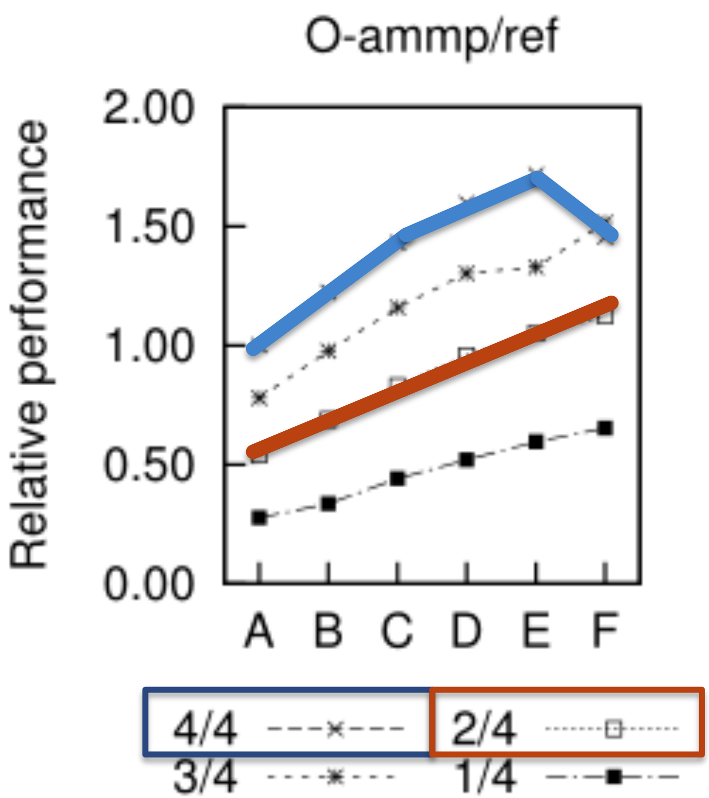
- Sweep core vs. cache area ratio for 14-nm design
  - Fixed 600 mm^2 area, core = 1.5 mm^2, L2 cache = 3 mm^2/MB
  - Clustered L2 shared by 4 cores, latency ~ log2(size)
  - 1 GB @ 512 GB/s on-package, 64 GB/s off-package

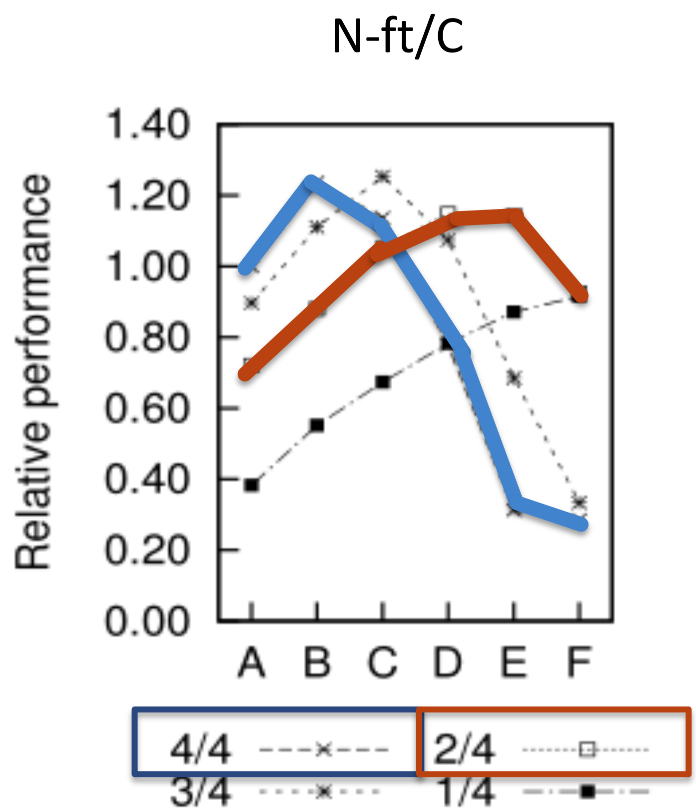
Variant	A	B	C	D	E	F
Cores	96	128	160	192	224	256
L2 size (MB/core)	1.5	1.0	0.8	0.5	0.4	0.3
Core area	25%	33%	40%	50%	58%	64%

UNDERSUBSCRIPTION FOR FUTURE DESIGNS

Compute bound: linear relation between active cores and performance

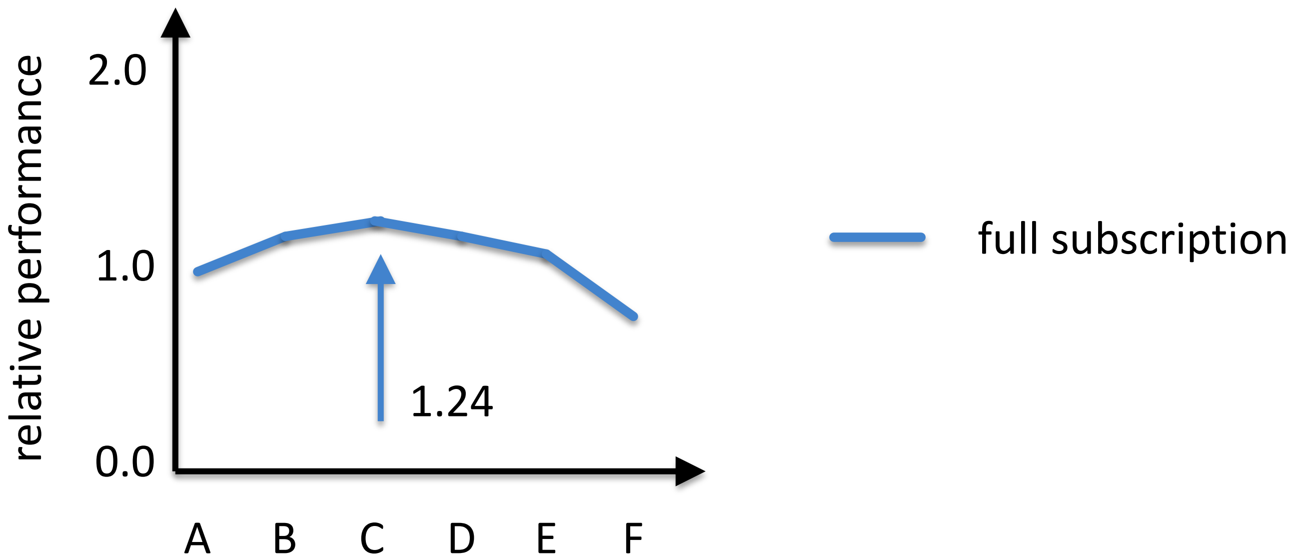
Capacity bound: reduce thread count until combined working set fits available cache





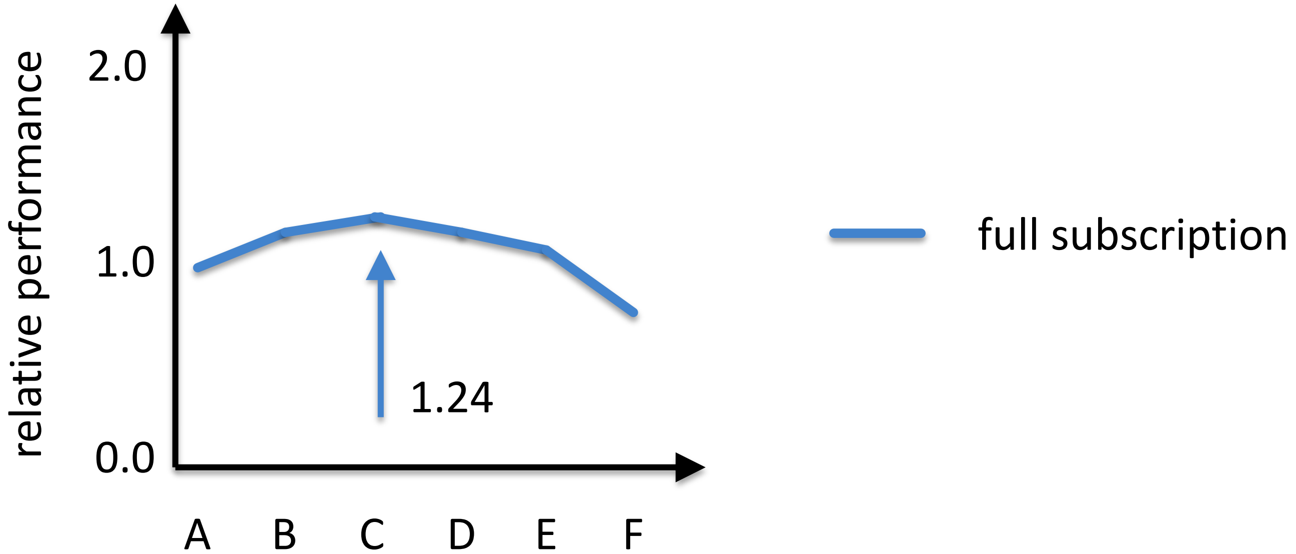
UNDERSUBSCRIPTION FOR FUTURE DESIGNS

- Build *one* design with best average performance



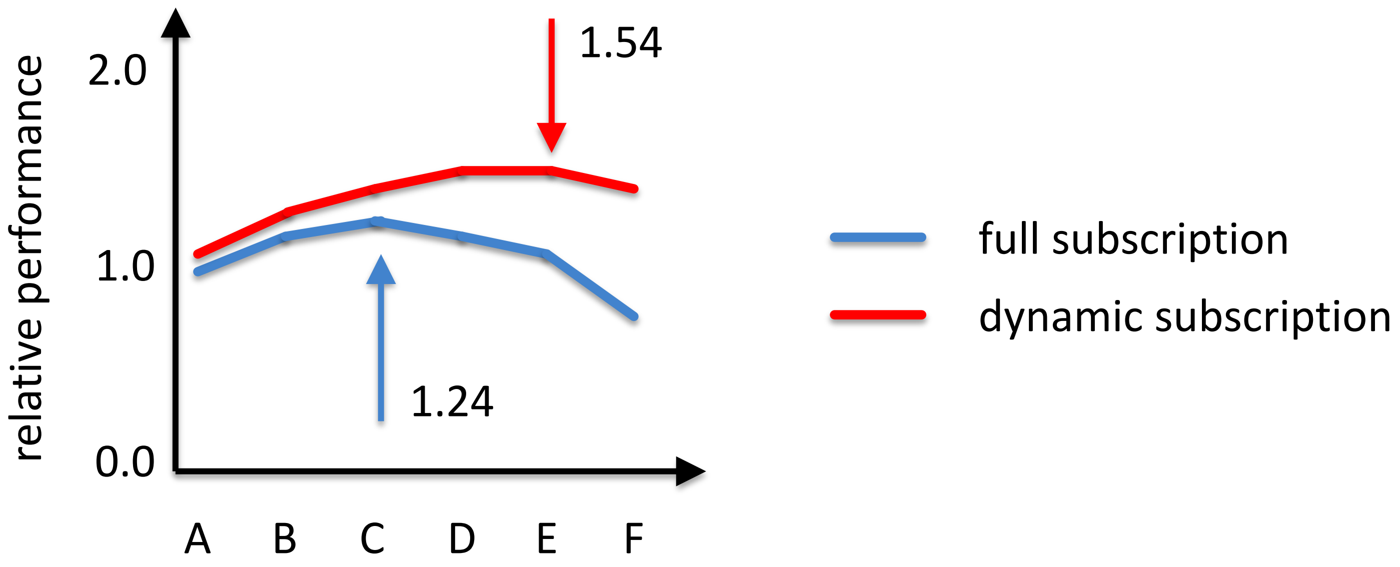
UNDERSUBSCRIPTION FOR FUTURE DESIGNS

- Build *one* design with best average performance
- Full subscription:
  - conservative option C has highest average performance



UNDERSUBSCRIPTION FOR FUTURE DESIGNS

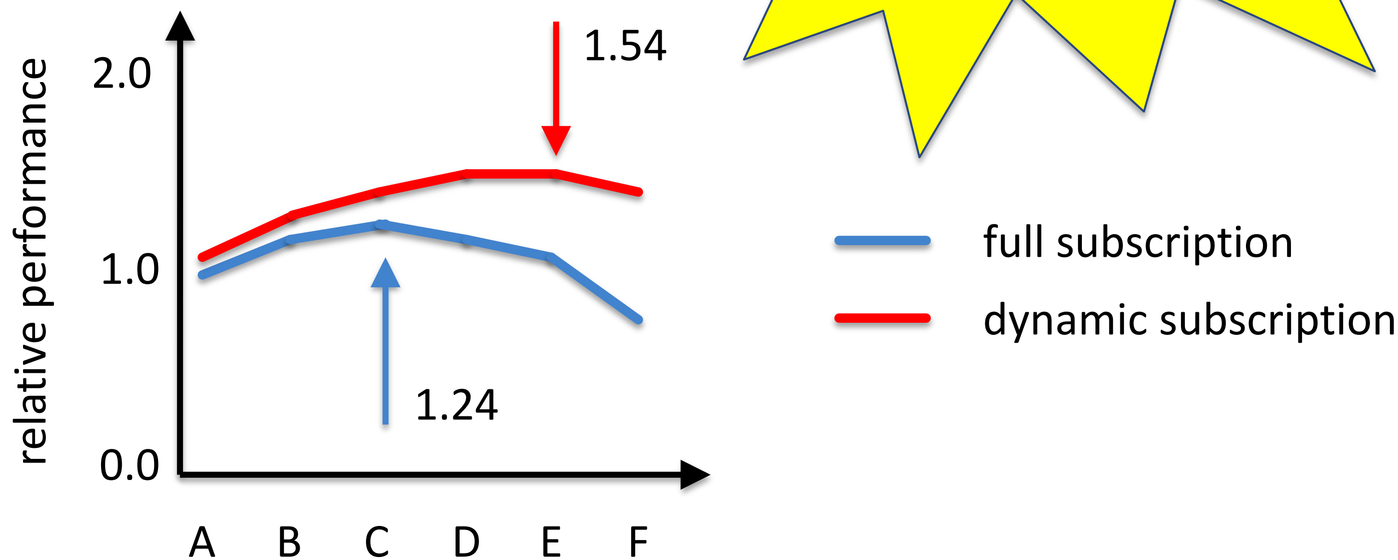
- Build *one* design with best average performance
- Full subscription:
  - conservative option C has highest average performance
- Dynamic undersubscription: prefer more cores
  - higher max. performance for compute-bound benchmarks
  - use undersubscription to accomodate capacity-bound workloads



UNDERSUBSCRIPTION FOR FUTURE DESIGNS

- Build *one* design with best average performance
- Full subscription:
  - conservative option C has highe… t average pe… formanc…
- Dynamic undersubscription: prefer …
  - higher max. performance for comp…
  - use undersubscription to accom… ca… …ds

E vs. C: 40% more cores,
15% higher performance



CONCLUSIONS

- Use clustered caches for future many-core designs
  - Balance hit rate and hit latency
  - Exploit sharing to avoid duplication
  - Allow for undersubscription (use all cache, not all cores)
- CRUST for dynamic undersubscription
  - Adapt thread count per OpenMP parallel section
  - Performance and energy improvements of up to 50%
- Take undersubscription usage model into account when designing future many-core processors
  - CRUST-aware design: 40% more cores,
    15% higher performance