Programming Techniques for Supercomputers: Performance Issues on Modern Multicore Architectures

Resource Scalability

Cache Coherence

Topology and Pinning

Dynamic Clock Speeds

Prof. Dr. G. Wellein[a,b] , Dr. G. Hager[a]
[a] Erlangen National High Performance Computing Center (NHR@FAU)
[b] Department für Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg, Sommersemester 2024

Performance Issues on Modern Multicore Architectures

Resource Scalability

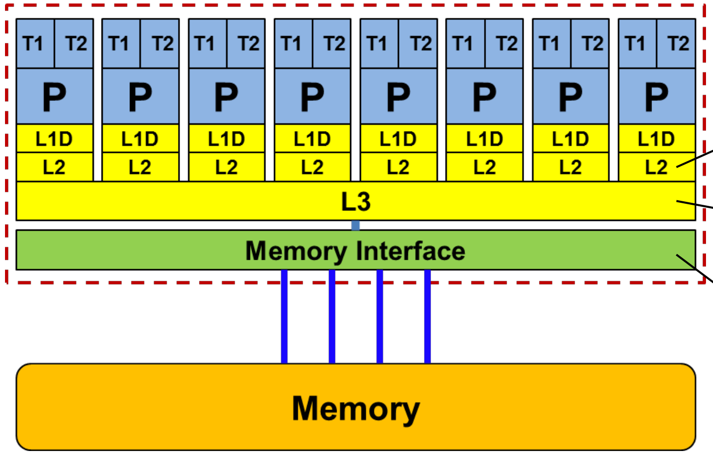
Cache Coherence

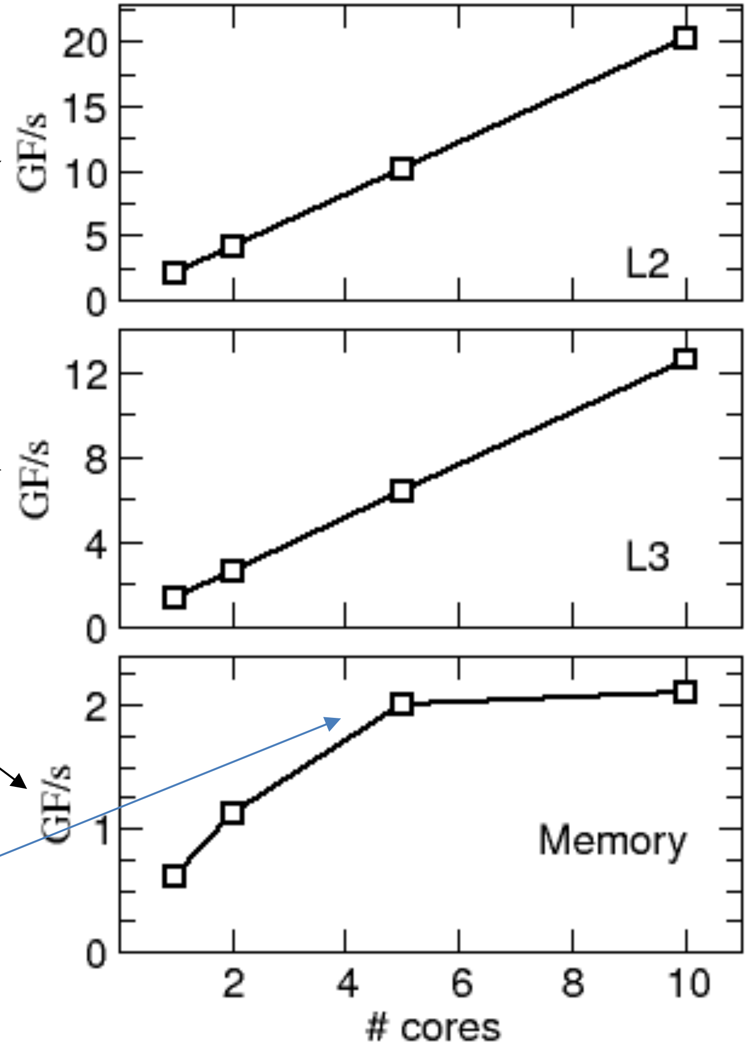
Topology and Pinning

Dynamic Clock Speeds

Scalable data paths on multicores?!

Run a copy of vector triads on each core
Intel Xeon E5-2660 v2: 10 cores@2.2 GHz





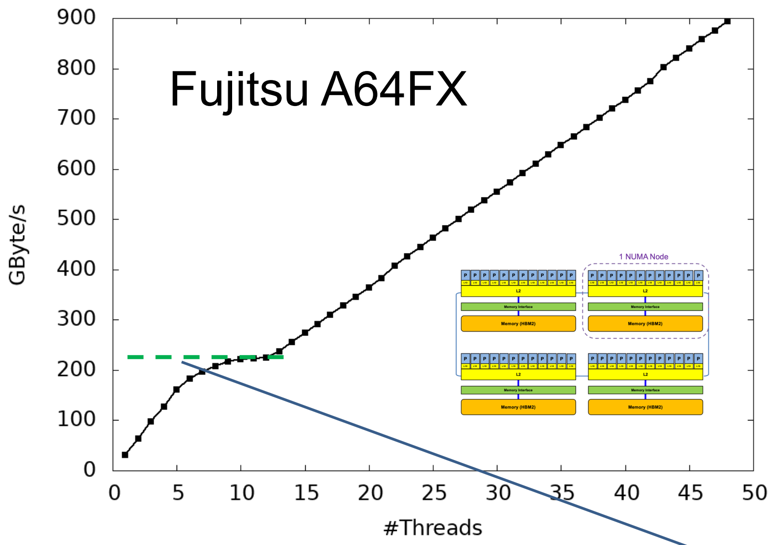
"Parallel" resources:

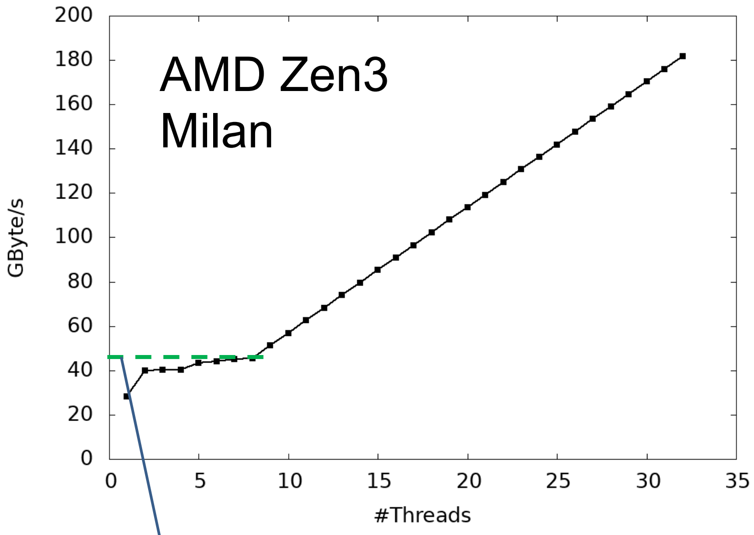
- Dedicated / core-local L2 cache scales linearly

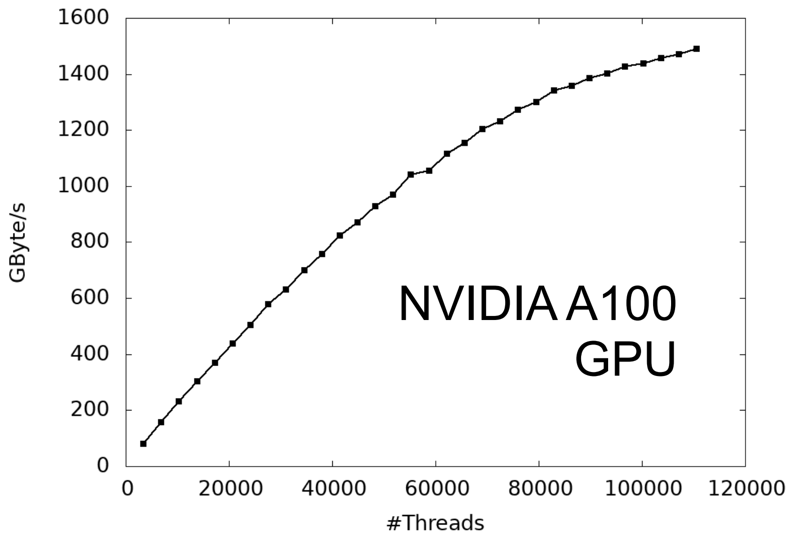
"Shared" resources:

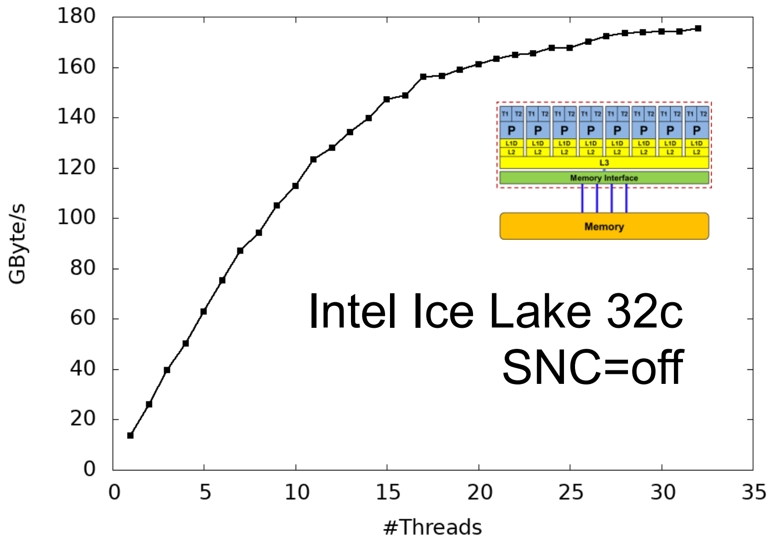
- Shared L3 caches scales linearly
- **Shared memory interface saturates**

Memory bandwidth saturation (read-only)



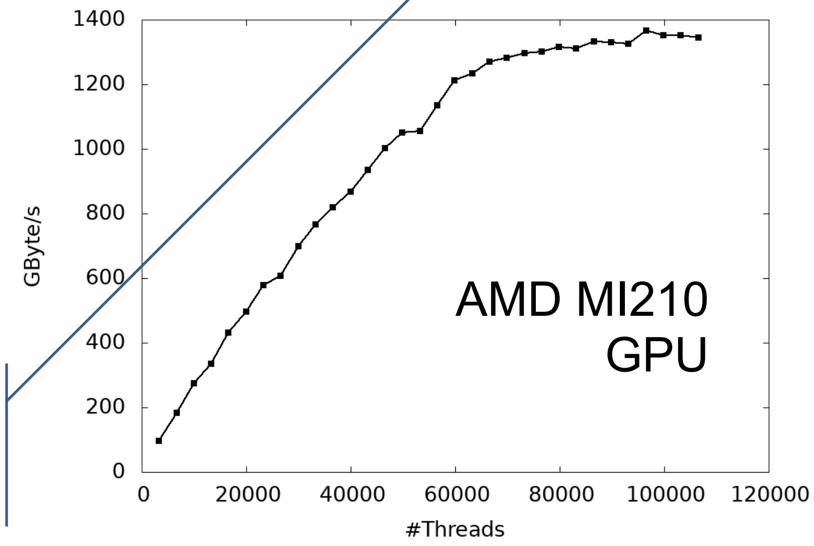






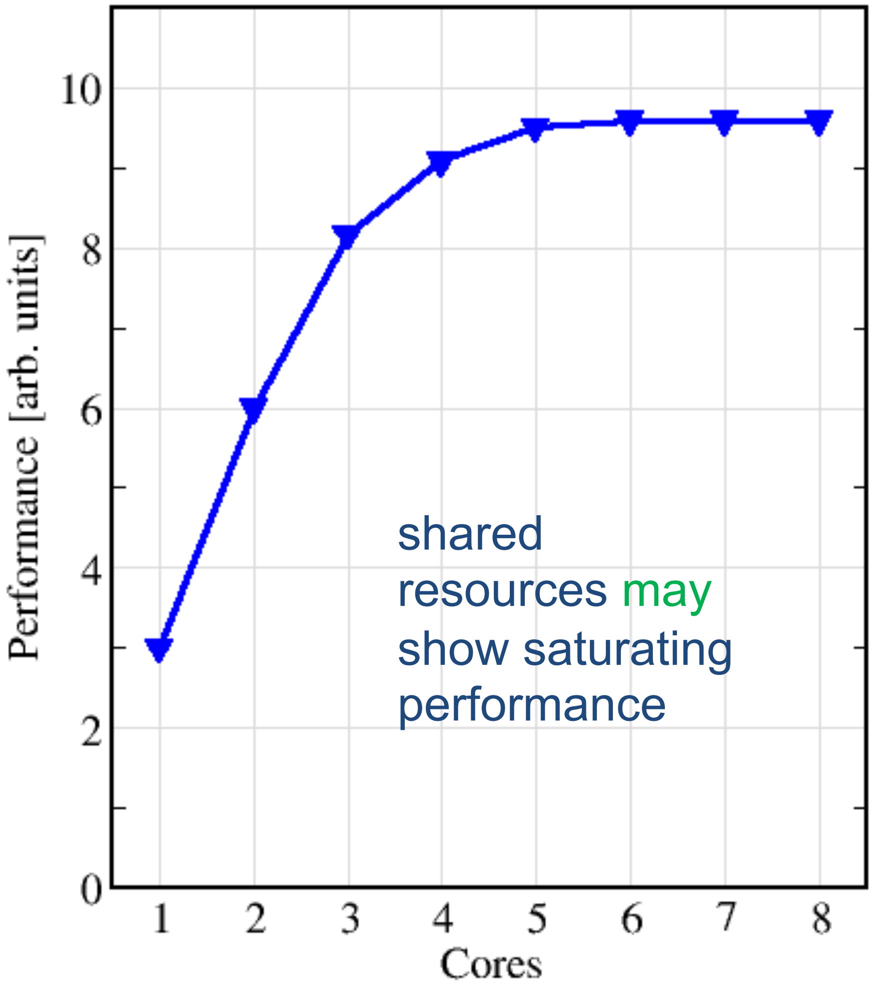
Bandwidth saturation on 1st ccNUMA domain

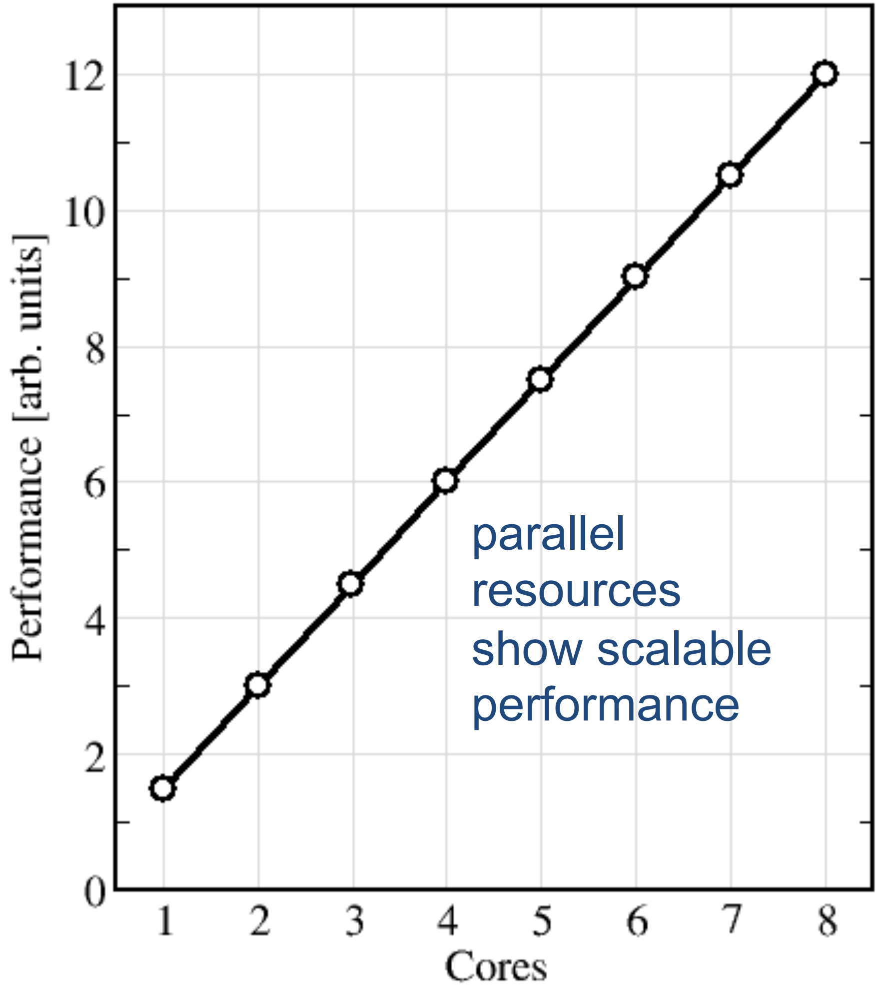
Massive thread parallelism needed on GPUs to saturate



Parallel/shared resources: Scalable/saturating behavior

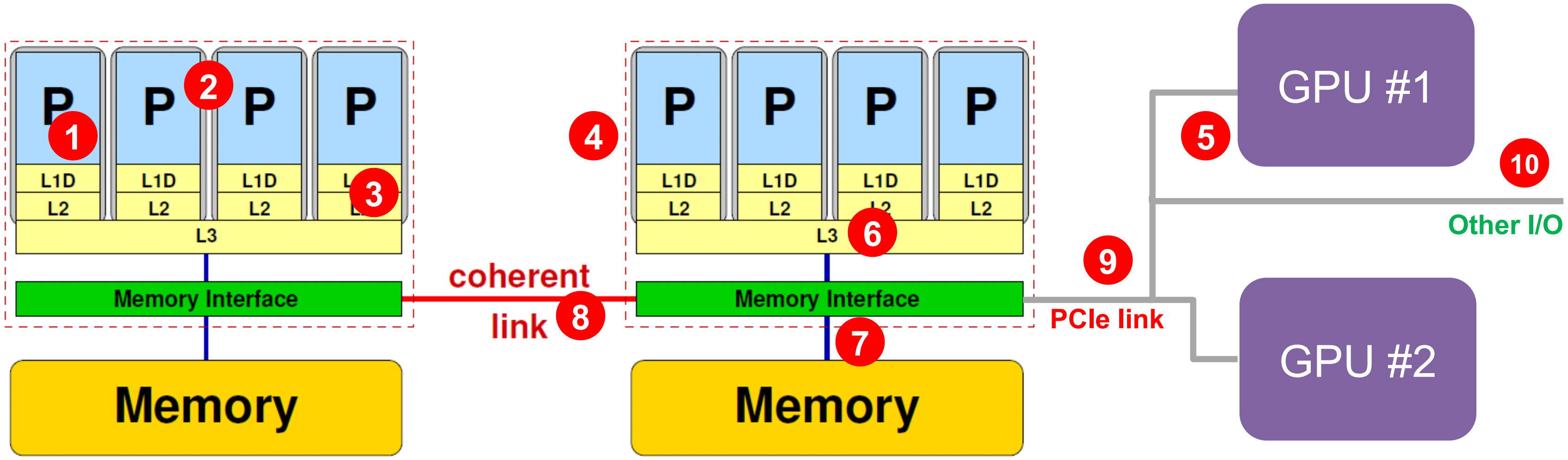
- Clearly distinguish between “saturating” and “scalable” performance on the chip level





Compute nodes: Parallel and shared resources

Parallel and shared resources within a shared-memory node



Parallel resources:

- Execution/SIMD units 1
- Cores 2
- Inner cache levels 3
- Sockets / ccNUMA domains 4
- Multiple accelerators 5

Shared resources:

- Outer cache level per socket 6
- Memory bus per socket 7
- Intersocket link 8
- PCIe bus(es) 9
- Other I/O resources 10

How does hardware scalability impact your parallel code?

Performance Issues on Modern Multicore Architectures

Resource Scalability

Cache Coherence

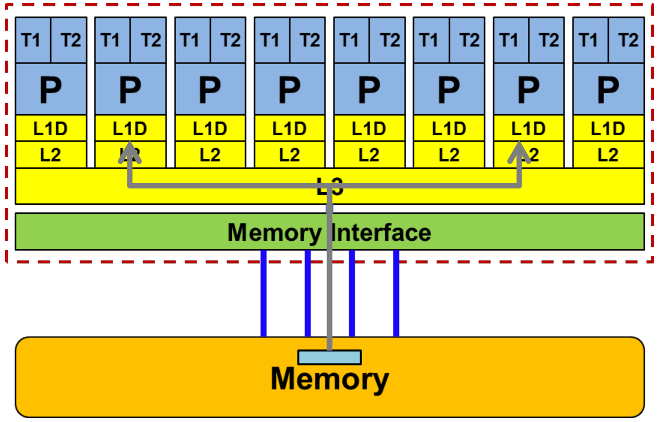
Topology and Pinning

Dynamic Clock Speeds

Shared-Memory parallel computers: cache coherence

- Cache coherence in shared-memory multi-core/-processor architecture

  - Copies of same cache line may reside in different caches (Example: If 2 cores load same CL to L1 there are 5 copies in various caches)



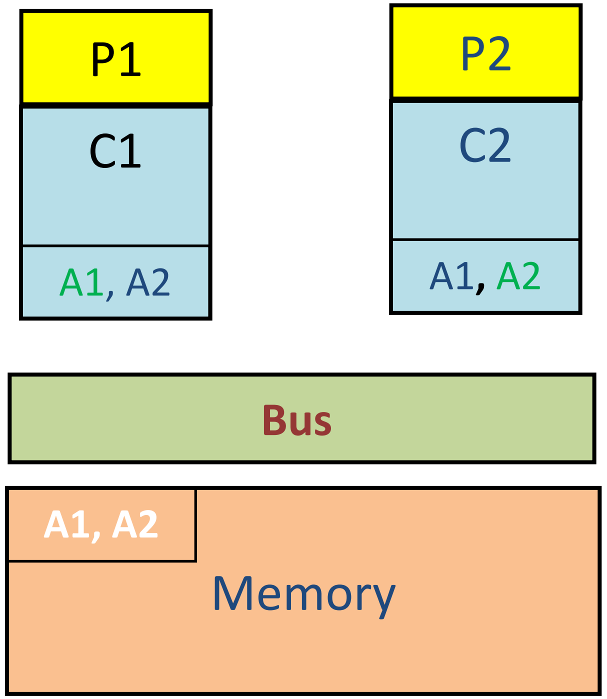
  - If one core updates data (usually in its L1), other copies become inconsistent/outdated

  - Consistency of cache line copies is ensured by cache coherence protocols

  - Cache coherence protocols do not alleviate correct parallel programming for shared-memory architectures!

Parallel computers: Shared Memory: Cache coherency

- Data in cache is only a copy of data in memory
  - Multiple copies of same data on multiprocessor systems
  - Cache coherence protocol/hardware ensure consistent data view
  - Without cache coherence: shared cache lines can become clobbered:
    (Cache line size = 2 WORD; [A1,A2] are in a single CL)



P1	P2
`Load A1`	`Load A2`
`Write A1=0`	`Write A2=0`

Correct Code!

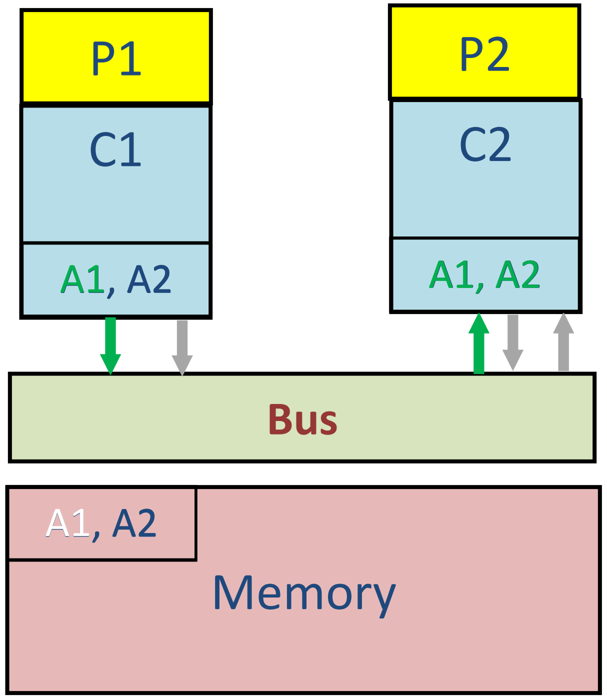
Write-back to memory leads to incoherent data:

A1, A2 | A1, A2 | A1, A2

C1 & C2 entry can not be merged to: A1, A2

Parallel computers: Shared Memory: Cache coherency

- Cache coherence protocol must keep track of cache line status



P1

`Load A1`

`Write A1=0:`

1. Request exclusive access to CL
2. Invalidate CL in C2
3. Modify A1 in C1

P2

`Load A2`

`Write A2=0:`

1. Request exclusive CL access
2. CL write back+ Invalidate
3. Load CL to C2
4. Modify A2 in C2

t

C2 is exclusive owner of CL ←

Parallel computers: Shared Memory: Cache coherency

- Widespread cache coherence protocol: MESI protocol

- A cache line can have four different states:
  - Modified: Cache line has been modified in this cache, and it resides in no other cache. Cache line needs to be evicted to ensure memory consistency
  - Exclusive: Cache line has been read from main memory but not (yet) modified. There are no (valid) copies in other caches
  - Shared: Cache line has been read from memory but not modified. There may be valid copies in other caches
  - Invalid: This cache line does not reflect any sensible data. Usually this happens if the cache line was in **S** state and another processor request exclusive ownership

Parallel computers: Shared Memory: Cache coherency

- Cache coherence can cause substantial overhead
  - may reduce available bandwidth
- Different implementations
  - Snoop: On modifying a CL, a CPU must broadcast its address to the whole system
  - Directory, “snoop filter”: Chipset (“network”) keeps track of which CLs are where and filters coherence traffic
- Directory-based ccNUMA can reduce pain of additional coherence traffic

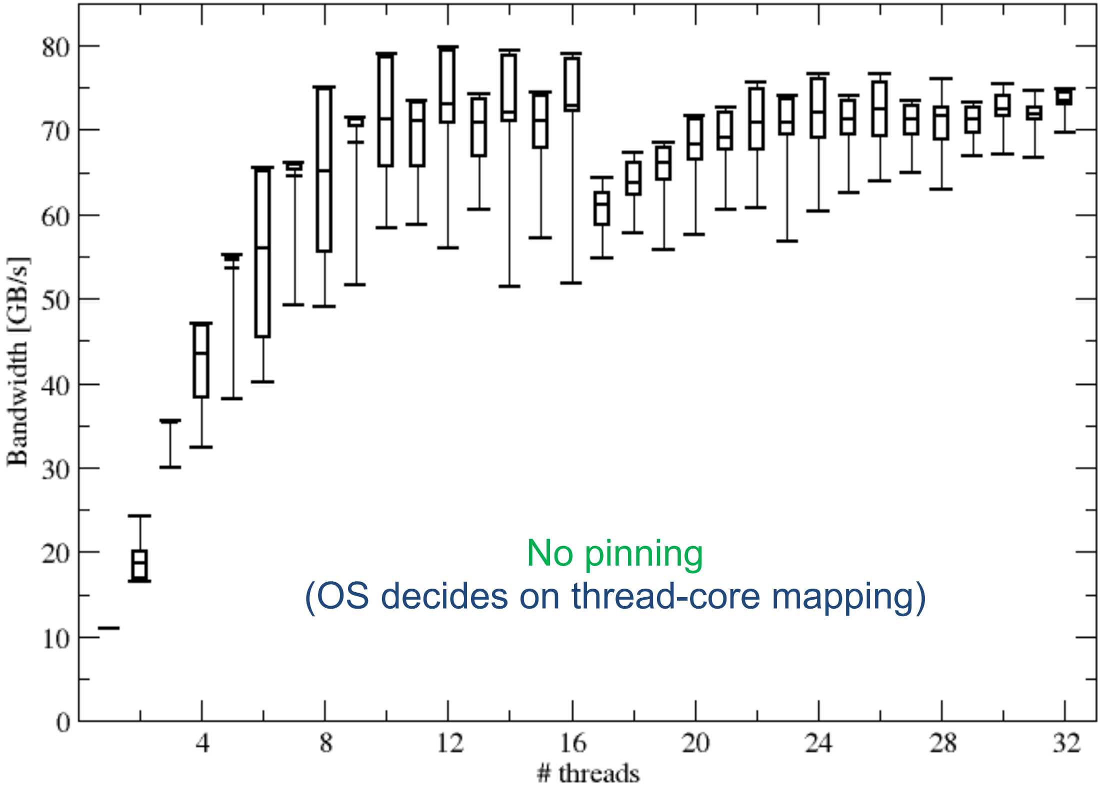
But always take care:

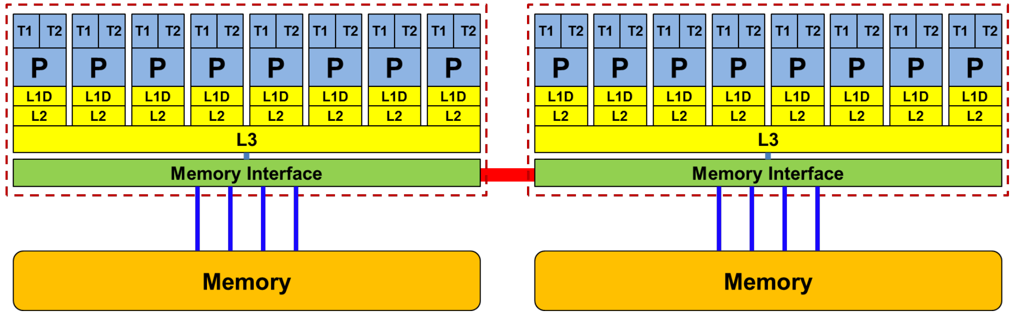
Multiple processors should never write frequently to the same cache line (“false sharing”)!

Performance Issues on Modern Multicore Architectures

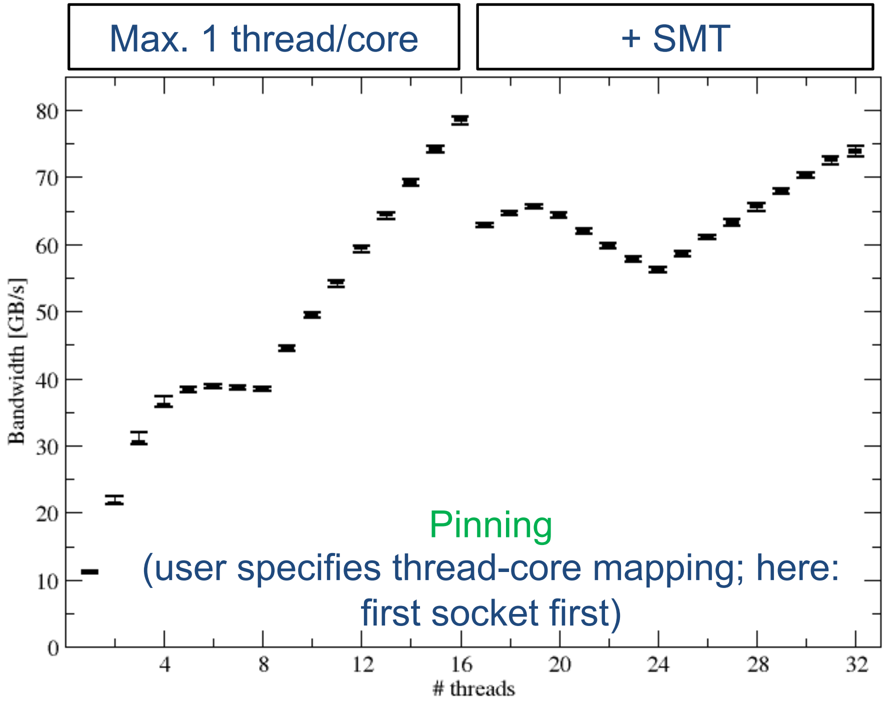
Resource Scalability
Cache Coherence
Topology and Pinning
Dynamic Clock Speeds

Performance on Multicores: Anarchy vs. thread pinning





2 x 8-core processor (+SMT)



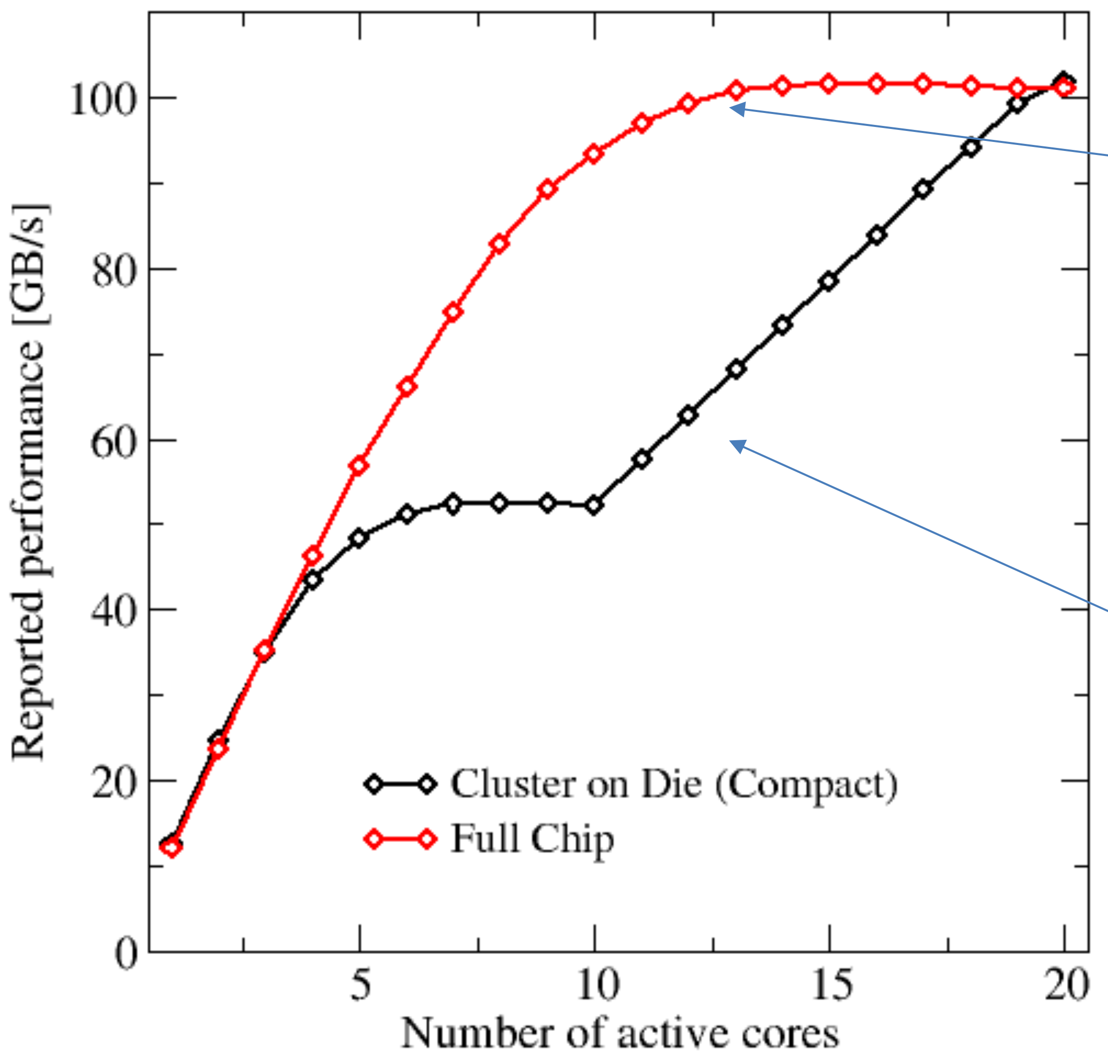
Experiment:

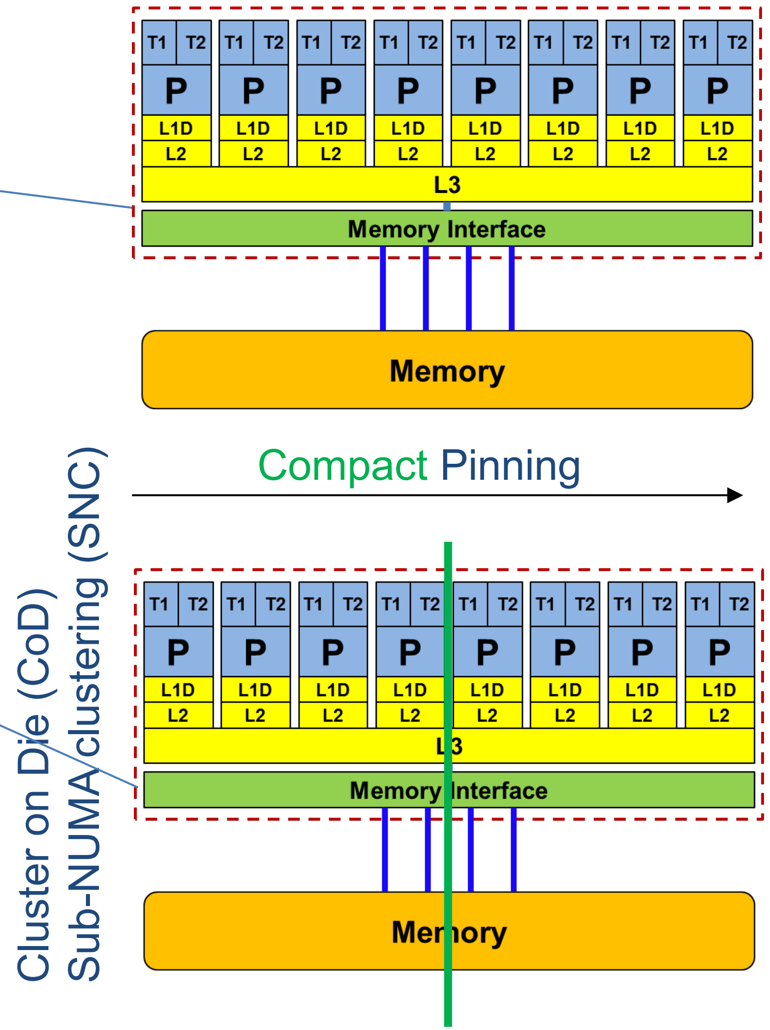
Run STREAM benchmark 100 times for each thread count

→ High performance variation without pinning

Performance on Multicores: Scalability, CoD/SNC, Pinning

- Performance scalability of STREAM triads on 20 core chip





CoD /SNC + Scattered pinning: see full chip

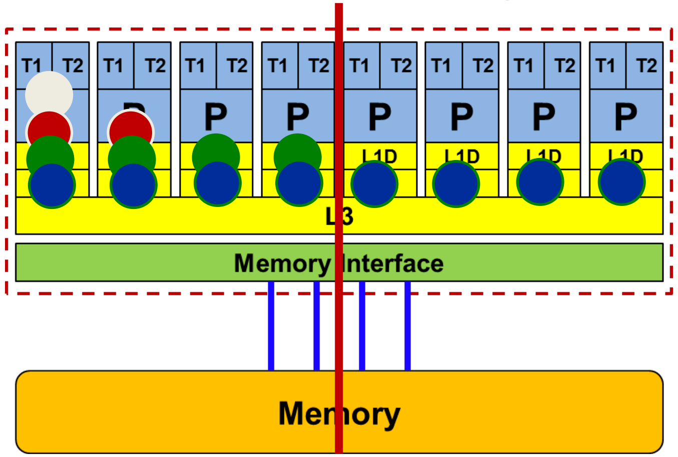
Controlling topology / pinning

- Highly OS-dependent system calls
  - But available on all systems

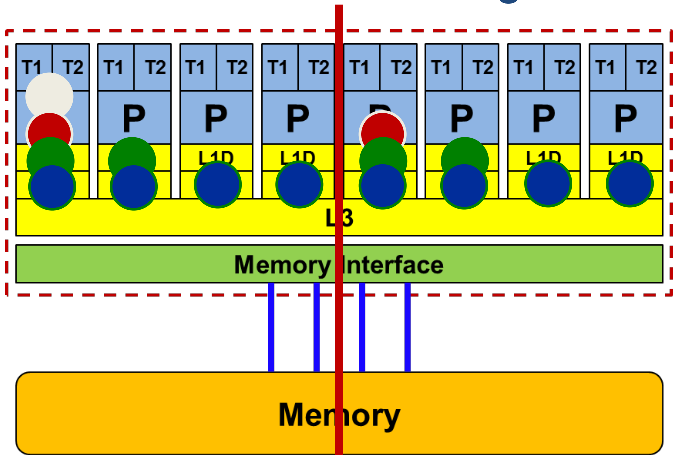
    Linux: `sched_setaffinity()`

    Windows: `SetThreadAffinityMask()`
- Support for “semi-automatic” pinning in some environments
  - All modern compilers with OpenMP support
  - Generic Linux: **`taskset`**, **`numactl`**, **`likwid-pin`** (see tutorial)
  - OpenMP 4.0

Compact Pinning



Scattered Pinning



Performance Issues on Modern Multicore Architectures

Resource Scalability

Cache Coherence

Topology and Pinning

Dynamic Clock Speeds

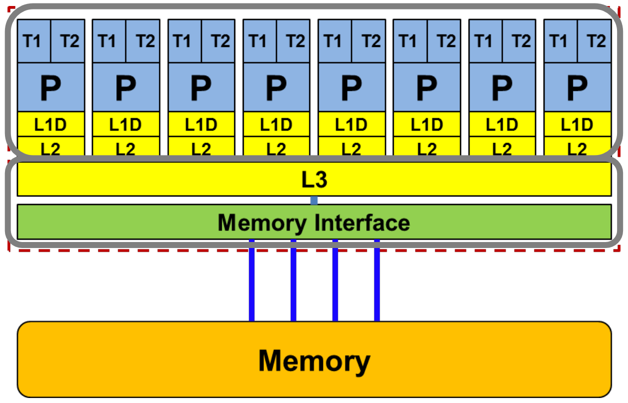
Dynamic Clock Speeds: Basics

- Modern multicore processors have a maximum power budget (TDP)

  Thermal Design Power
- Power consumption of given chip depends on:
  - Actual workload (#cores, SIMD Units active, clock speed,…)
  - Chip production quality or environmental conditions (e.g. temperature)
- Dark silicon: Parts of the chip run at (substantially) lower clock speed

- Turbo Mode: Processor decides dynamically on clock speed:
  Increasing ressource utilization / temperature → decreasing clock

- Multiple clock speed domains may be possible, e.g. for Intel
  - Core Clock (Core + L1/L2 cache)
  - Uncore Clock (L3 + Memory controller)



Dynamic Clock Speeds: Frequencies

- Frequency range for each multicore processor series (e.g. for Intel Xeon E5-2697 v4: 1.2 GHz,…,3.6 GHz)
- Two clock speed limits if using all cores of a modern multicore processor
  - CPU base frequency (a.k.a. nominal frequency): Minimum guaranteed clock speed if all cores are active (e.g. 2.3 GHz)
  - CPU all core turbo: Maximum supported clock speed if all cores are active (e.g. 2.8 GHz)
  - These clock speeds may may be different for the SIMD instruction set (SSE, AVX, AVX-512) used (e.g. 2.0 GHz / 2.7 GHz base / all core turbo for AVX code).
- Lower core counts: Clock speeds may stay within frequency range

Dynamic Clock Speeds: Overview

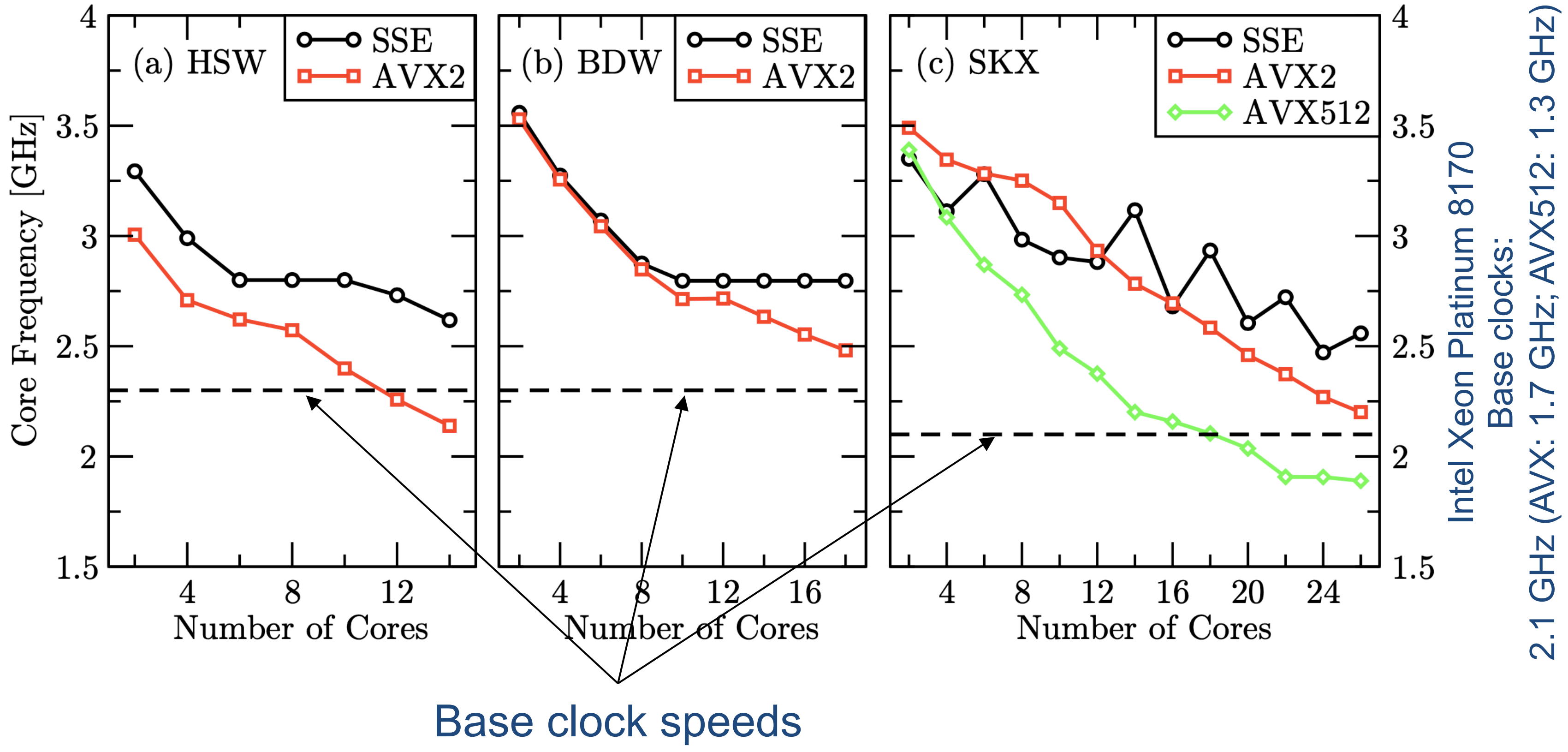
- Clock speeds when using all cores may dynamically vary by 20% - 30%
- Lower clock speeds for AVX (wide) SIMD units
- Using few (one) cores may boost clock speed by up to 50%!

Microarchitecture	Sandy Bridge-EP	Ivy Bridge-EP	Haswell-EP	Broadwell-EP	Zen	Power 8
Manufacturer	Intel	Intel	Intel	Intel	AMD	IBM
Chip model	Xeon E5-2680	Xeon E5-2690 v2	Xeon E5-2695 v3	Xeon E5-2697 v4	Epyc 7451	—
Release date	Q1/2012	Q3/2013	Q3/2014	Q1/2016	Q4/2017	Q2/2014
Cores/threads	8/16	10/20	14/28	18/36	24/48	10/80
Latest SIMD ext.	AVX	AVX	AVX2, FMA3	AVX2, FMA3	AVX, FMA3	VSX
CPU freq. range	1.2–3.5 GHz	1.2–3.6 GHz	1.2-3.3 GHz	1.2–3.6 GHz	1.2–3.6 GHz	2.1–3.5 GHz
Base freq.	2.7 GHz	3.0 GHz	2.3 GHz	2.3 GHz	2.3 GHz	2.9 GHz
AVX base freq.	—	—	1.9 GHz	2.0 GHz	—	—
All core turbo	3.1 GHz	3.3 GHz	2.8 GHz	2.8 GHz	3.2 GHz	3.5 GHz
AVX all core turbo	—	—	2.6 GHz	2.7 GHz	—	—
Uncore freq. range	—	—	1.2-3.0 GHz	1.2–2.8 GHz	—	—

J. Hofmann, „A First-Principles Approach to Performance, Power, and Energy Models for Contemporary Multi- and Many-Core Processors“, Dissertation, FAU, 2019

Dynamic Clock Speeds: Impact of cores / SIMD

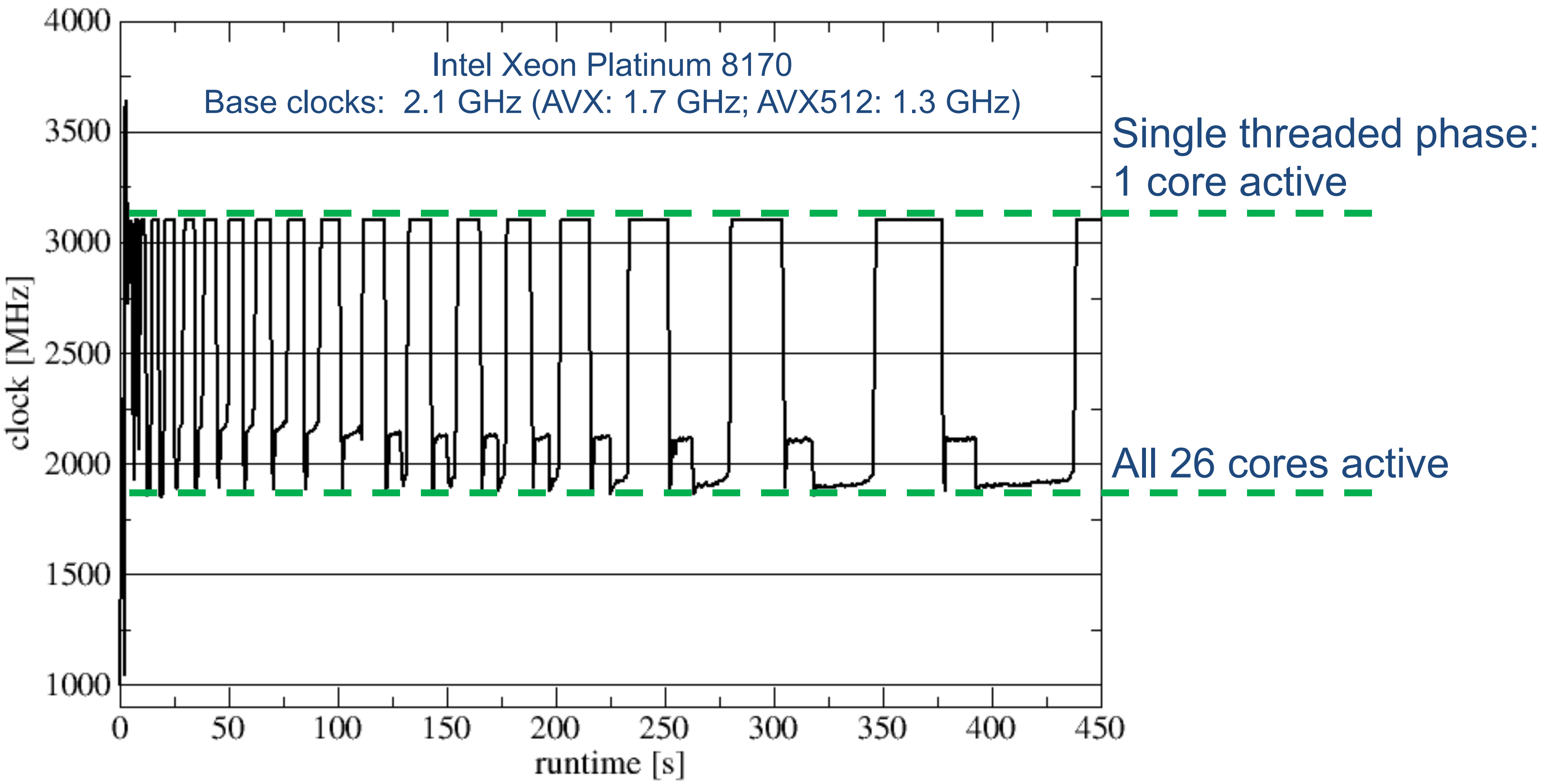
- Running LINPACK on one chip (Intel mkl implementation)
- Processor adapts clock speeds dynamically to ressource utilization (cores, SIMD widths)
- Base clock speeds are lower bounds



Dynamic Clock Speed: Dynamic Adaption

Running multithreaded LINPACK (Intel mkl version; AVX512) on one Intel Skylake

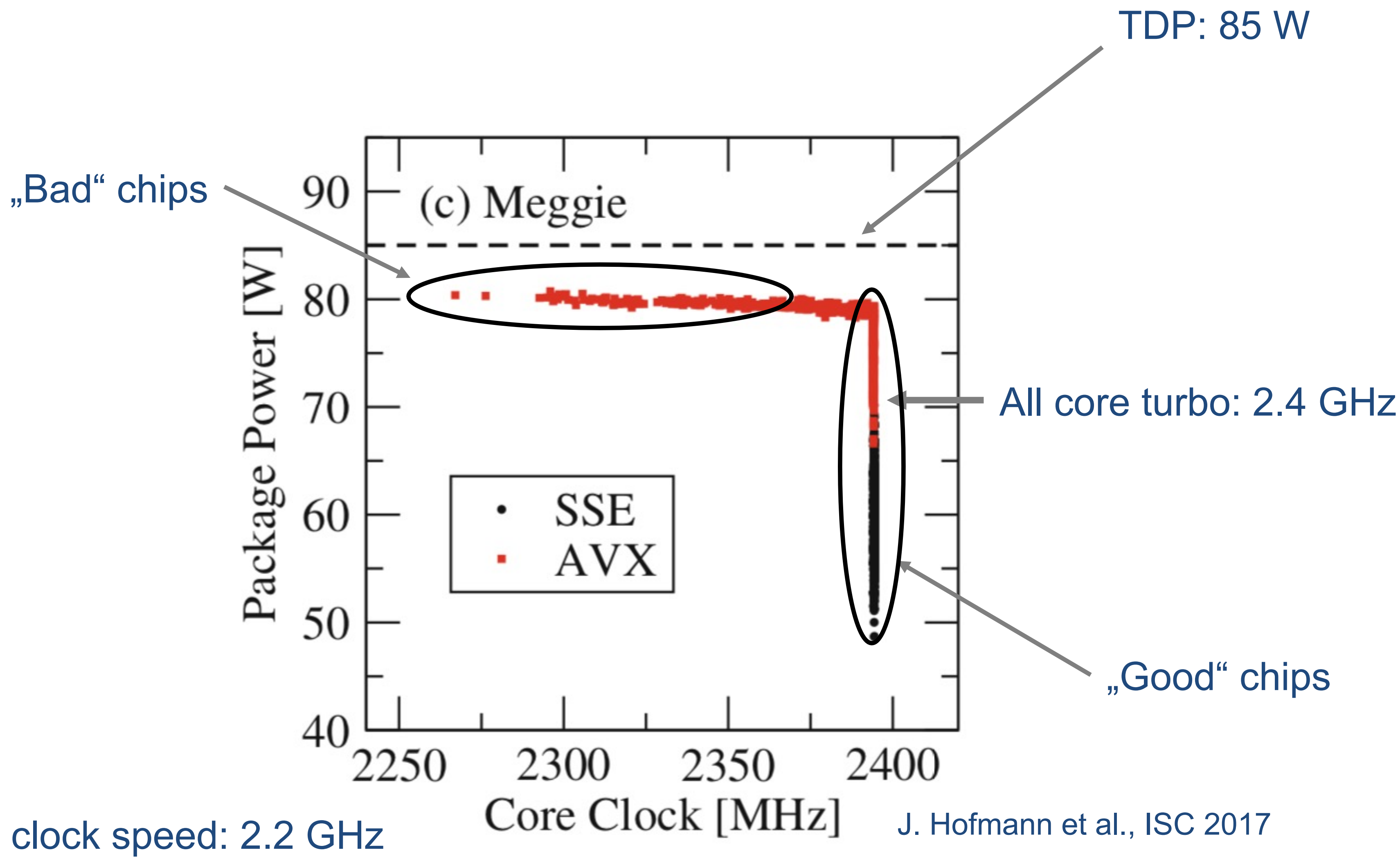
Monitoring clock speeds over time



At execution time the processor dynamically overclocks to always give you the full TDP envelope!

Dynamic Clock Speed: Chip Quality

- LINPACK: Power consumption vs. dynamic clock speed (1456 Intel Xeon E5-2630v4 chips)



Base clock speed: 2.2 GHz

J. Hofmann et al., ISC 2017

Dynamic Clock Speeds: Summary

- Turbo Mode may speed up your application execution
- Turbo Mode may introduce (strong) performance fluctuations: Chip quality, environment temperature,…
- Performance measurements should be done with fixed clock speed (e.g. using likwid) to CPU base frequency (default in PTfS)
- Information about clock speeds:
  - **`likwid-setFrequencies`**
  - https://en.wikichip.org/wiki/WikiChip
  - https://ark.intel.com/content/www/de/de/ark.html#@Processors

Lecture plan until July

- **3.6.2024: Lecture (Topologies & Clock Speeds)**
- **4.6.2024: Lecture (OpenMP)**
- **5.6.2024: Lecture (OpenMP)**

- **10.6.2024: No Lecture**
- **11.6.2024: Lecture (GPU – Sebastian Kuckuk)**
- 12.6.2024: **Lecture (GPU – Sebastian Kuckuk)**

- **17.6.2024: Lecture (Roofline)**
- **18.6.2024: Lecture (Roofline)**
- 19.6.2024: No Lecture

- **24.6.2024: Lecture (Roofline - Case Studies)**
- **25.6.2024: Lecture (Roofline - Case Studies)**
- 26.6.2024: No Lecture