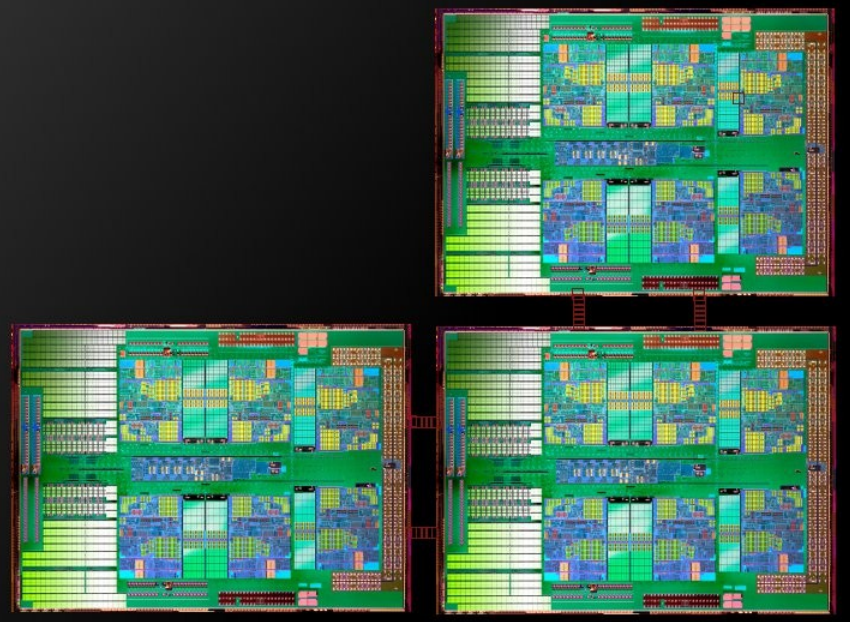
Why CPU Topology Matters

2010

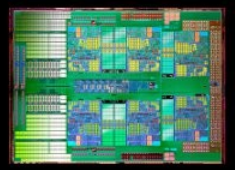
Andreas Herrmann <andreas.herrmann3@amd.com>

2010-03-13

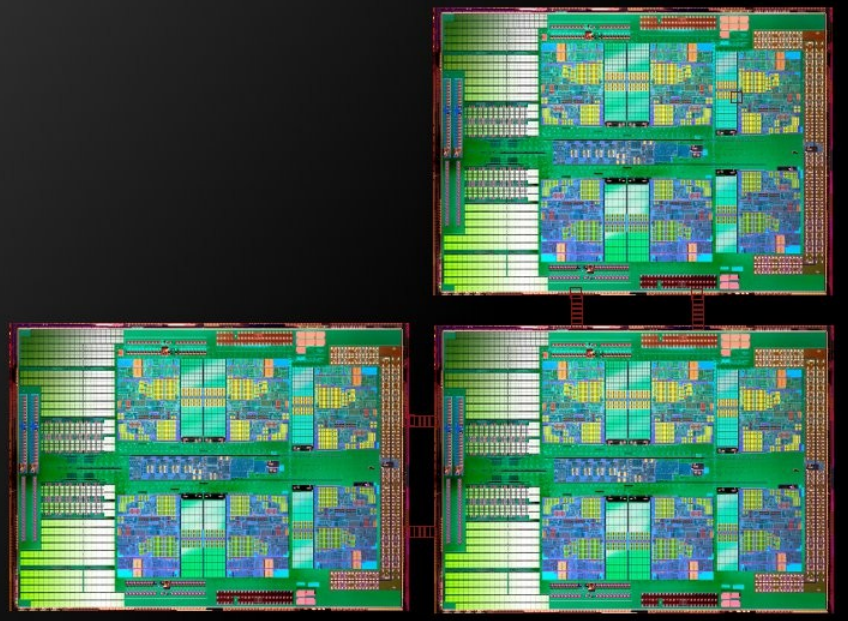


Contents

- Defining Topology
- Topology Examples for x86 CPUs
- Why does it matter?
- Topology Information Wanted
- In-Kernel Usage
- User space
- References



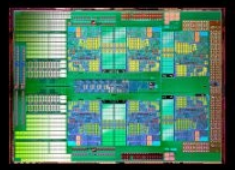
Defining Topology



Defining Topology

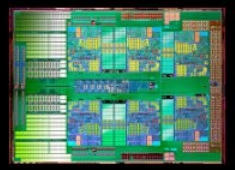
We can define different types of topology:

- Cache Topology
  - group CPUs by cache level hierarchy
- Package/Processor Topology
  - group CPUs by package hierarchy
- NUMA Topology
  - group CPUs by memory affinity
- “Power Topology”
  - group CPUs by frequency/voltage dependencies

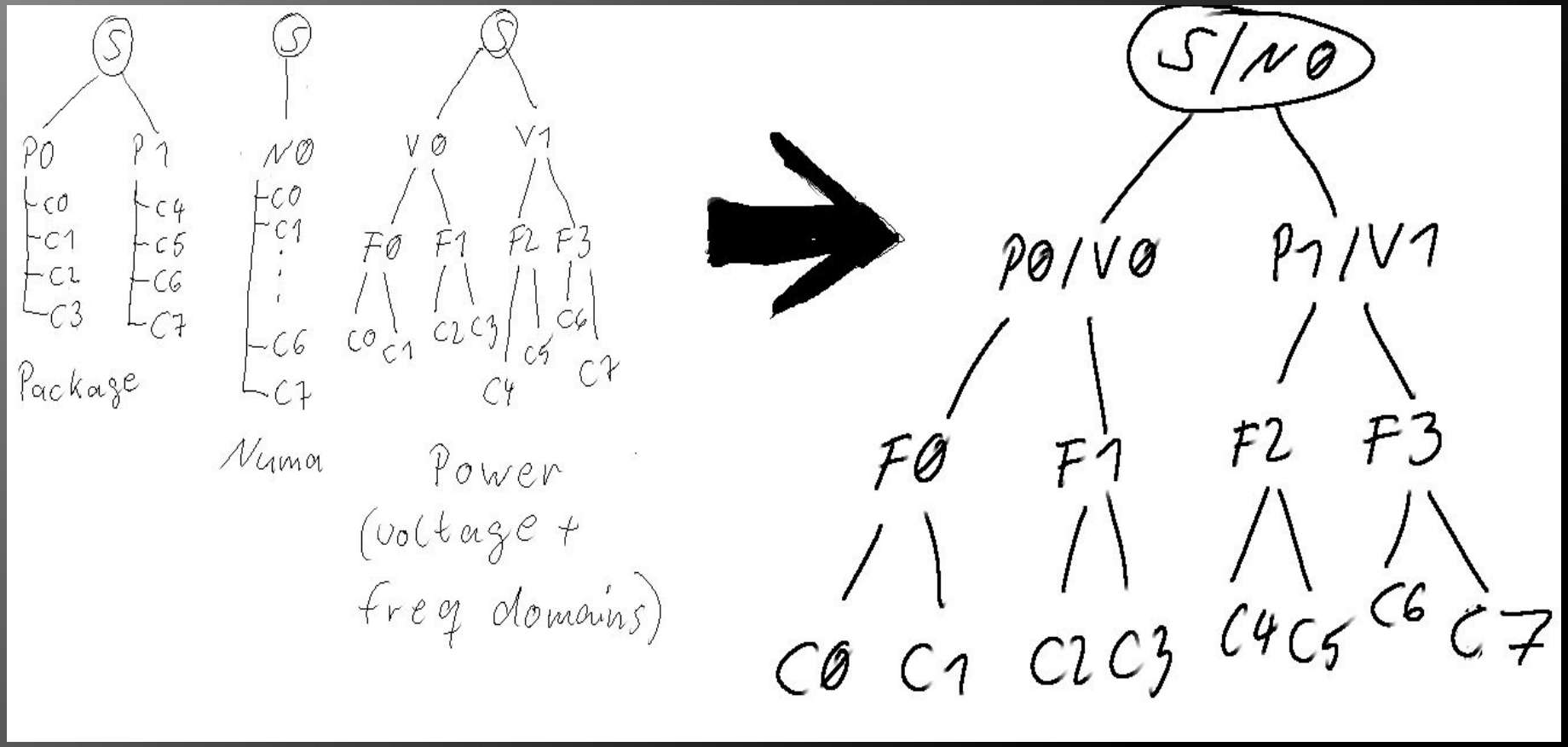


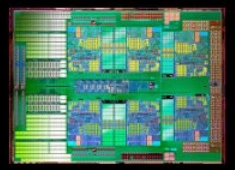
Defining Topology (cont'd)

- The hierarchical nature of these topologies is best described with a tree.
- In practice one tree is sufficient to reflect all topology information. (see example on next slide)
- There are other ways to visualize topology information, e.g. output created by `lstopo` (provided with `hwloc`)

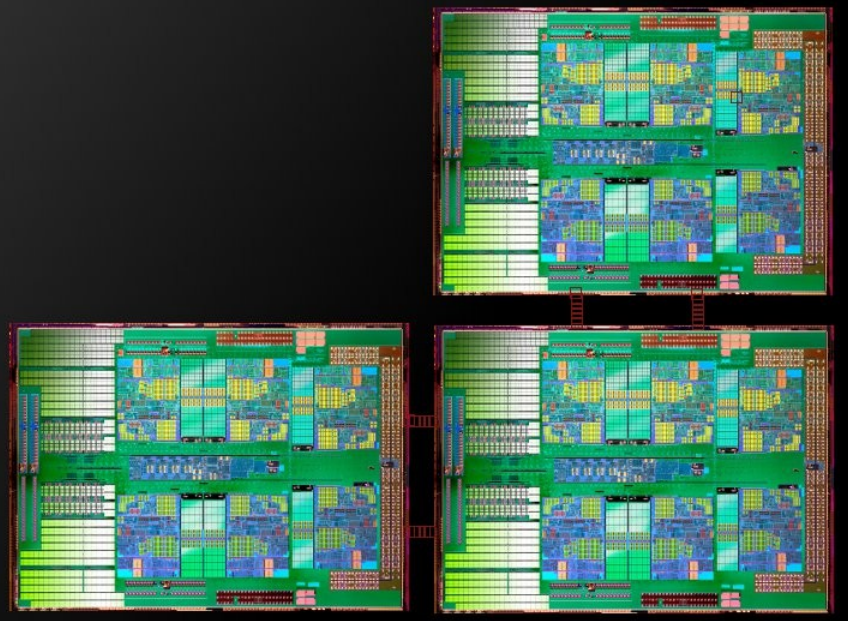


Defining Topology - Example

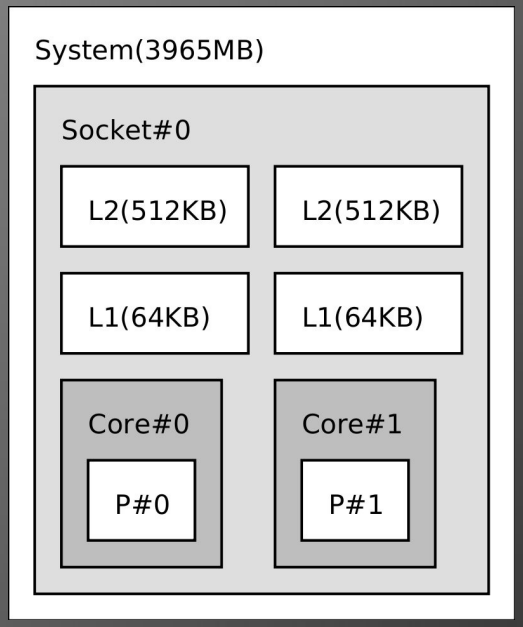


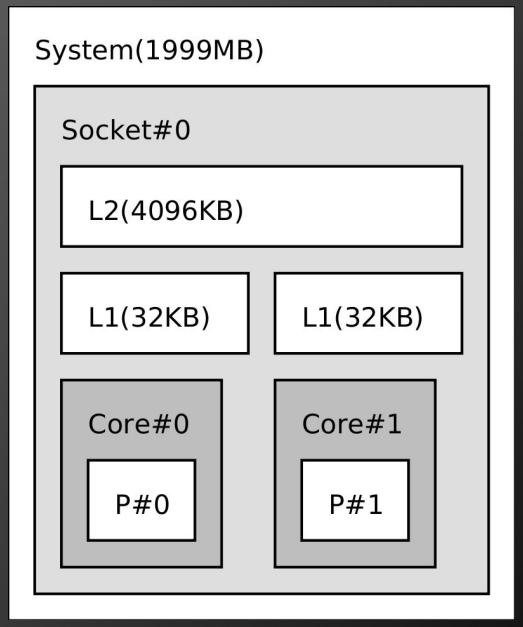


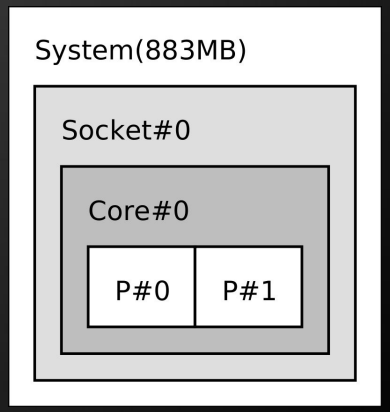
Topology Examples for x86 CPUs

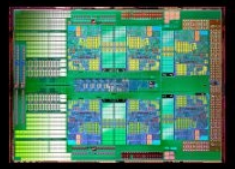


Topology Examples – AMD K8, Intel Core 2, Intel Pentium 4

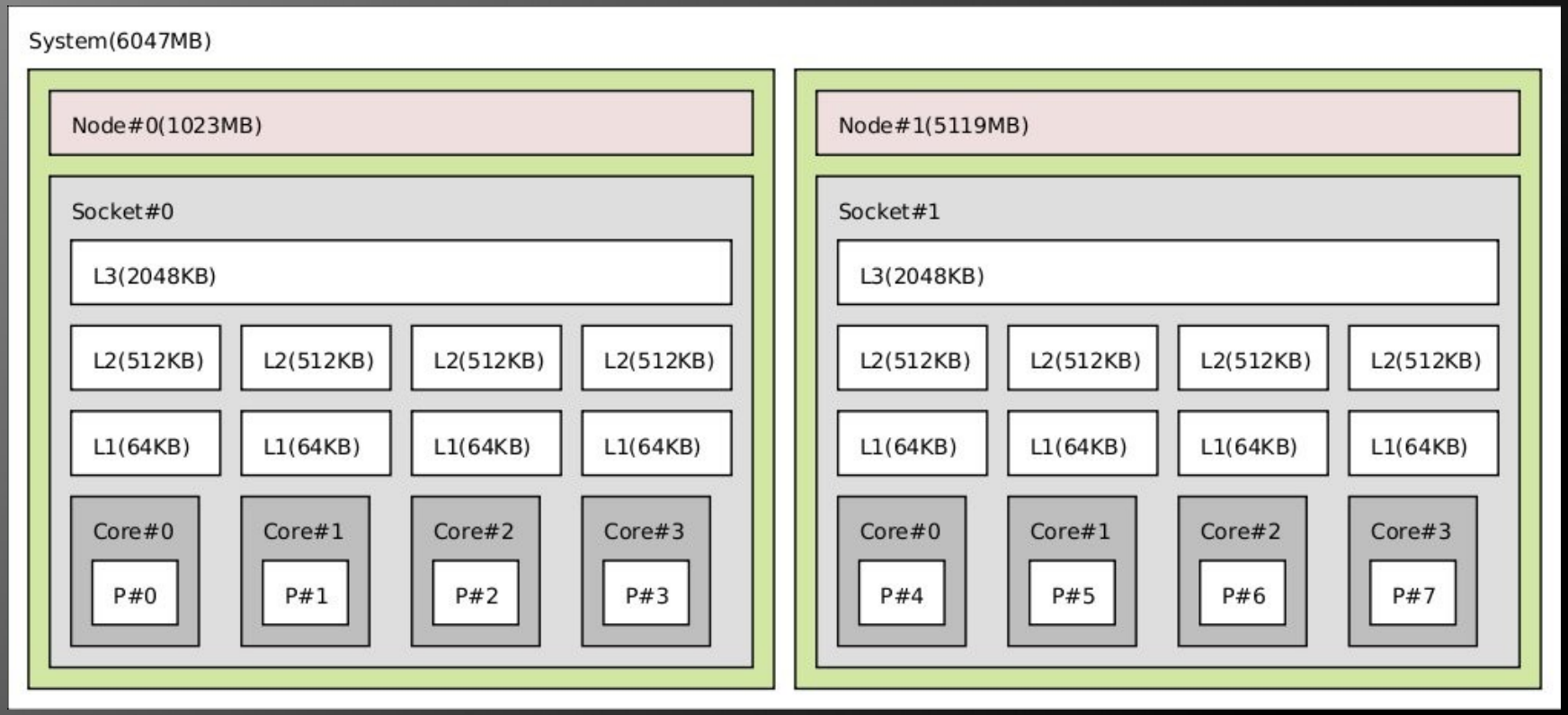


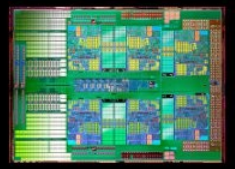




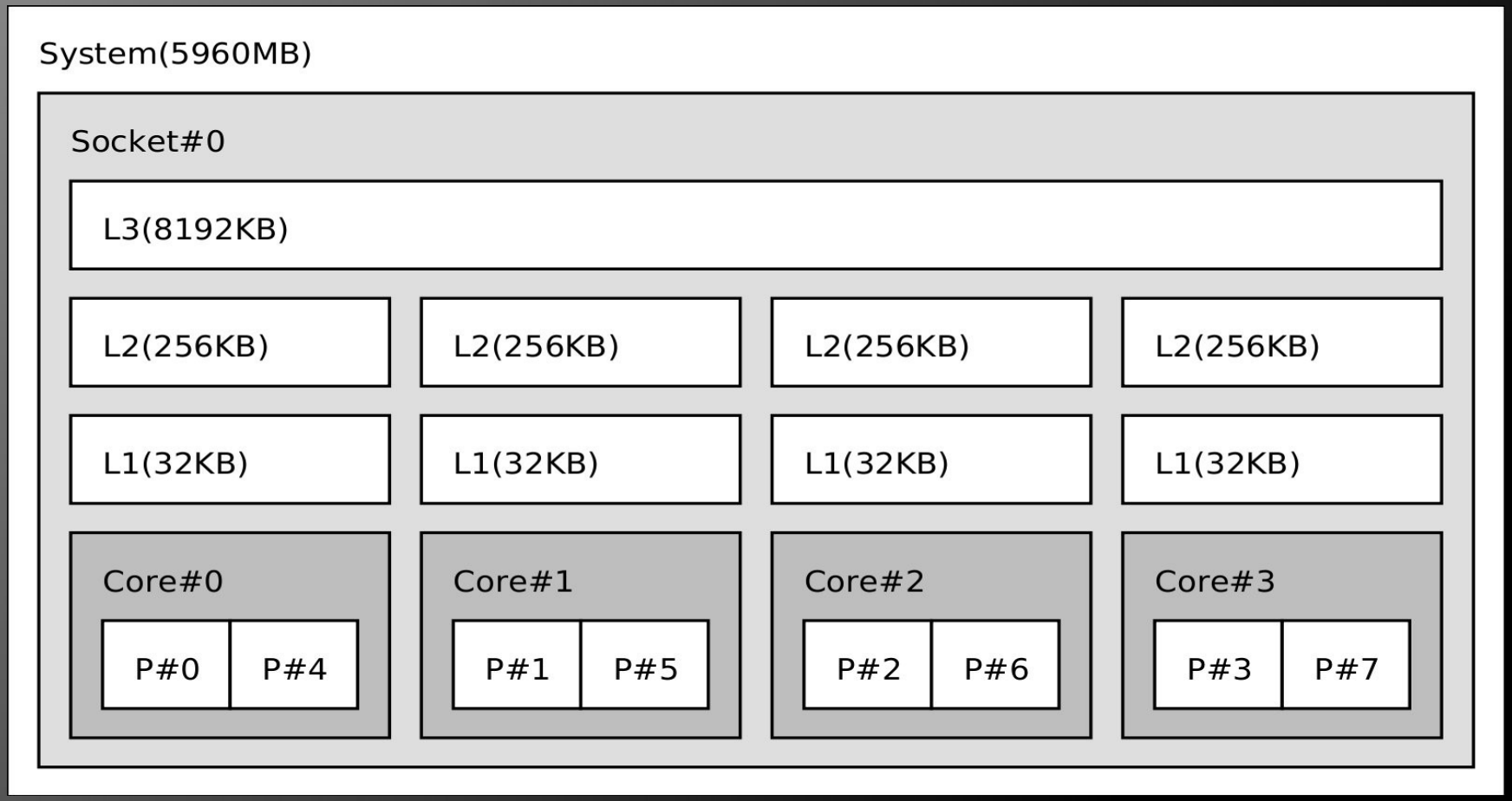


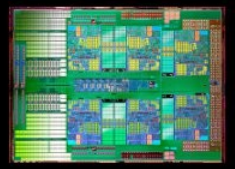
Topology Examples – 2P AMD Barcelona



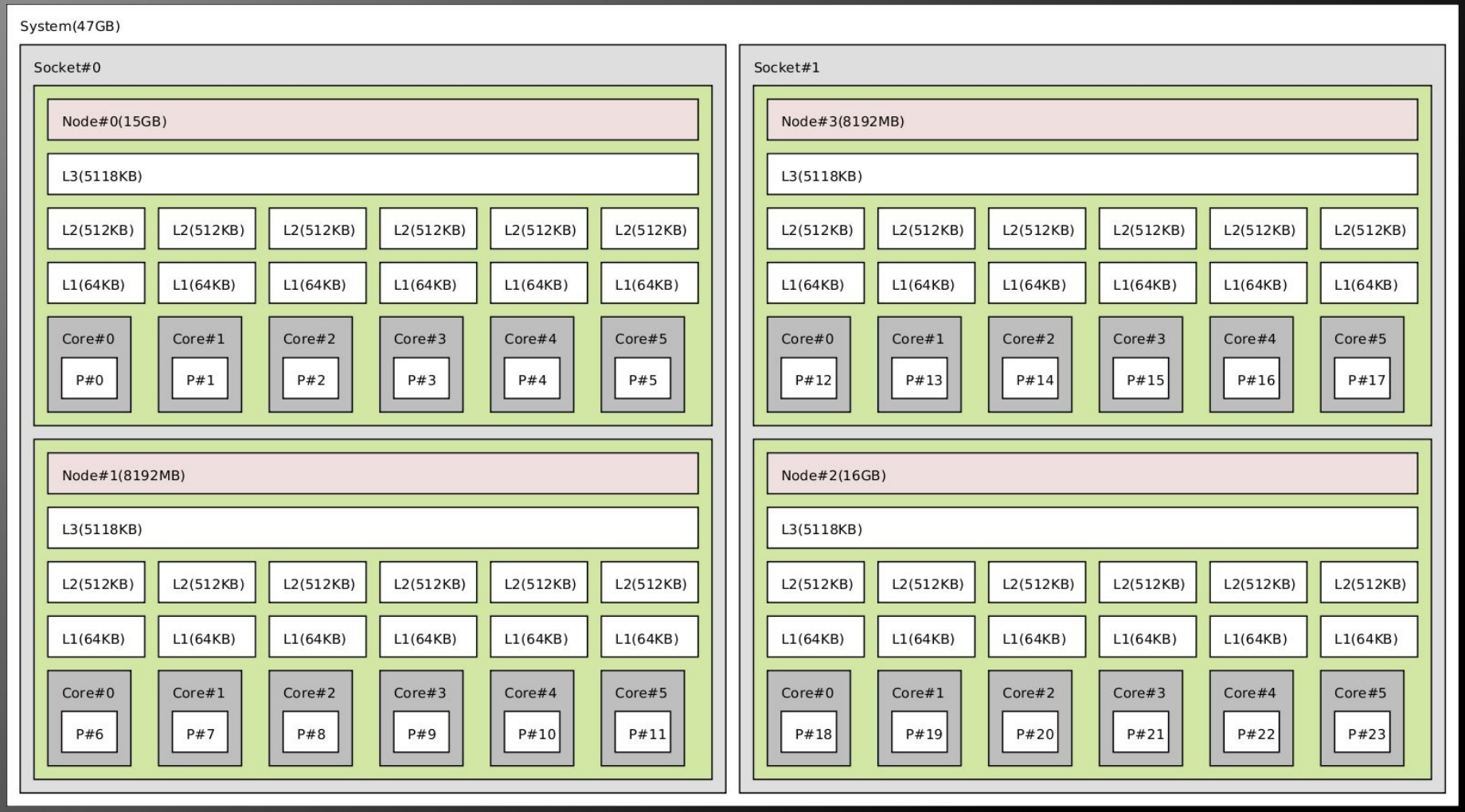


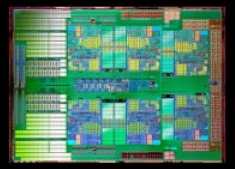
Topology Examples – Intel i7



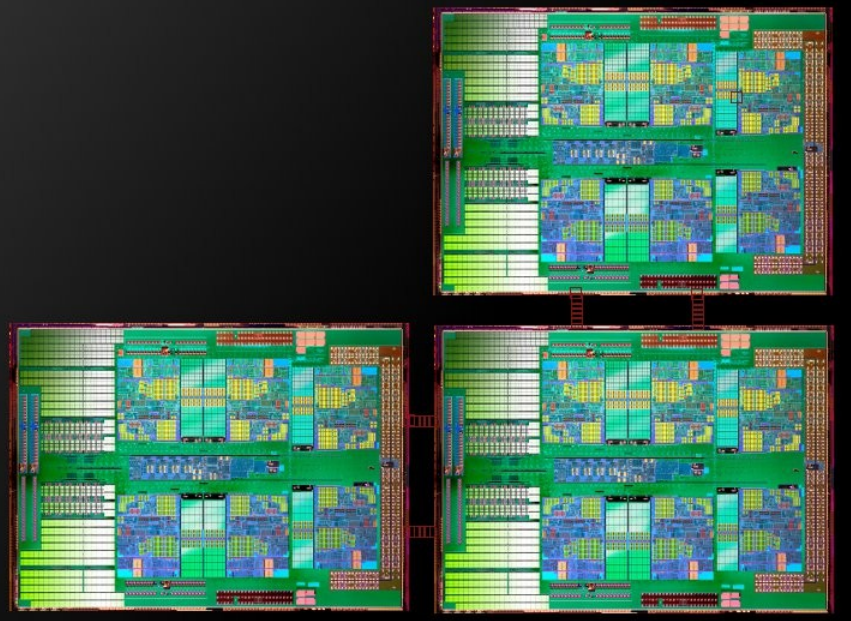


Topology Examples – 2P AMD Magny-Cours





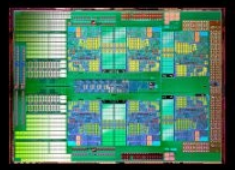
Why does it matter?



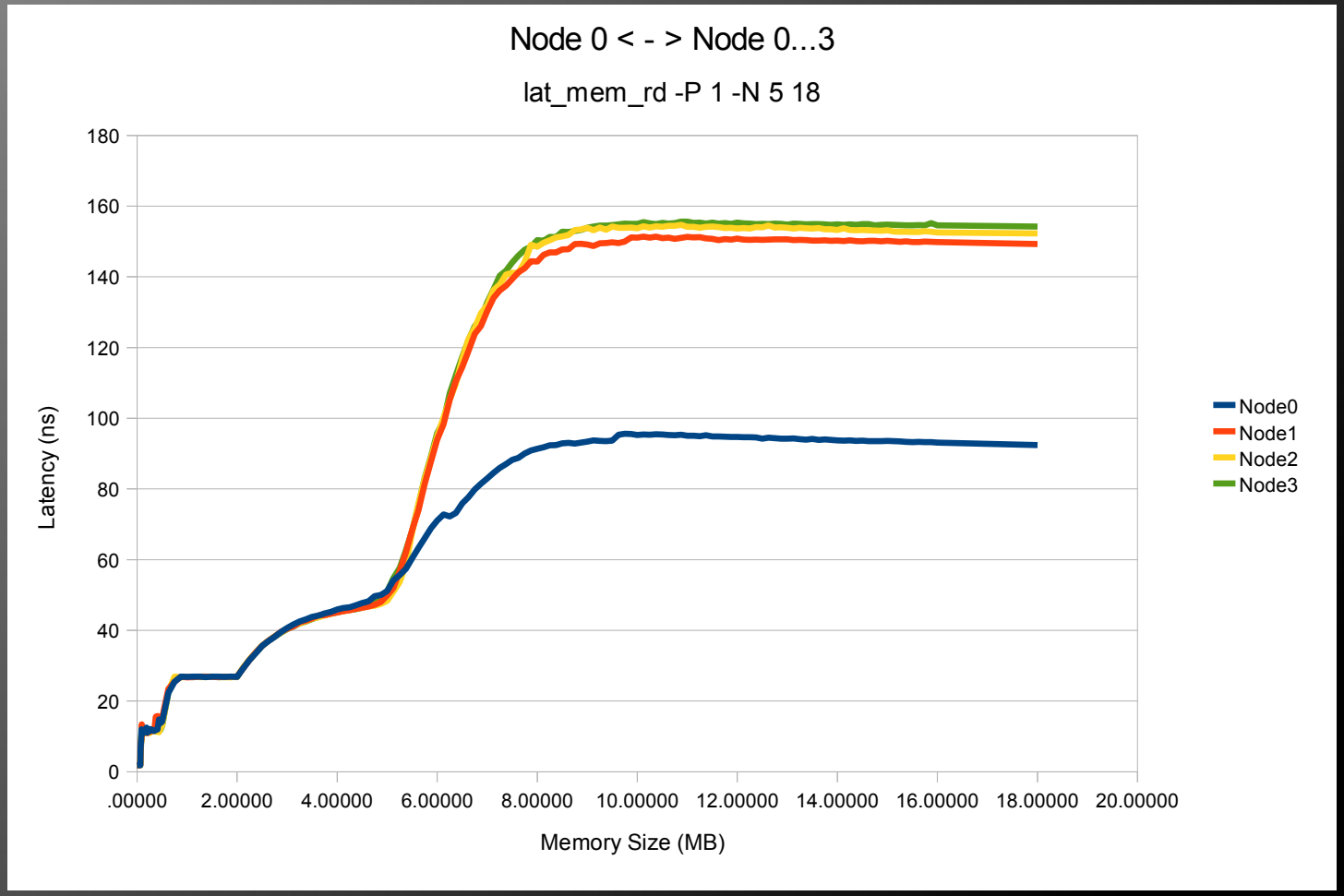
Why does it matter?

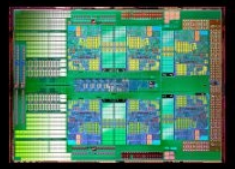
Topology information is vital for

- Reasonable memory allocation
  - Respect memory affinity on NUMA systems
- Scheduling decisions
  - Optimize for best performance and/or
  - Optimize for highest power savings

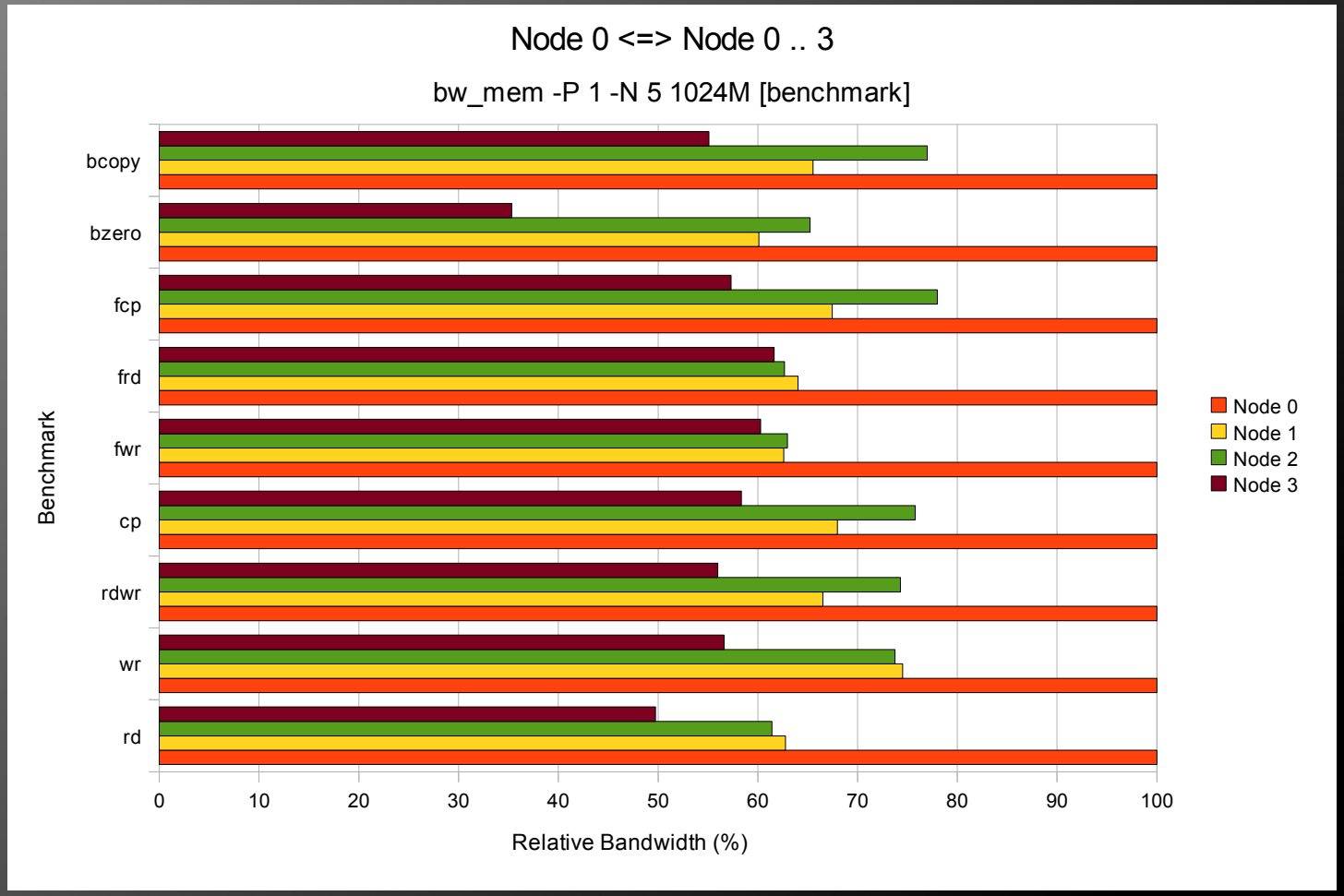


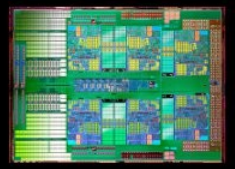
Why does it matter? - NUMA mem latency



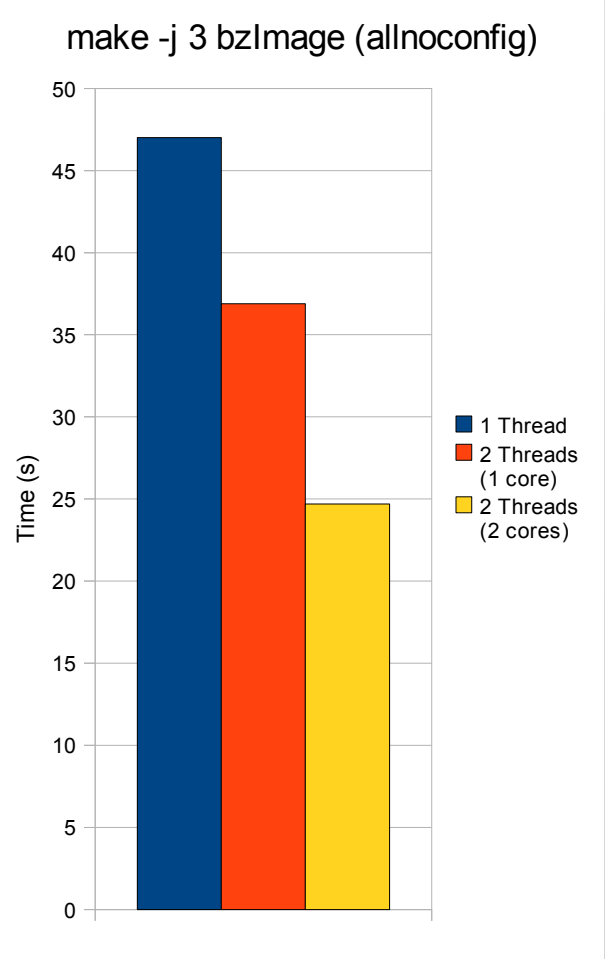


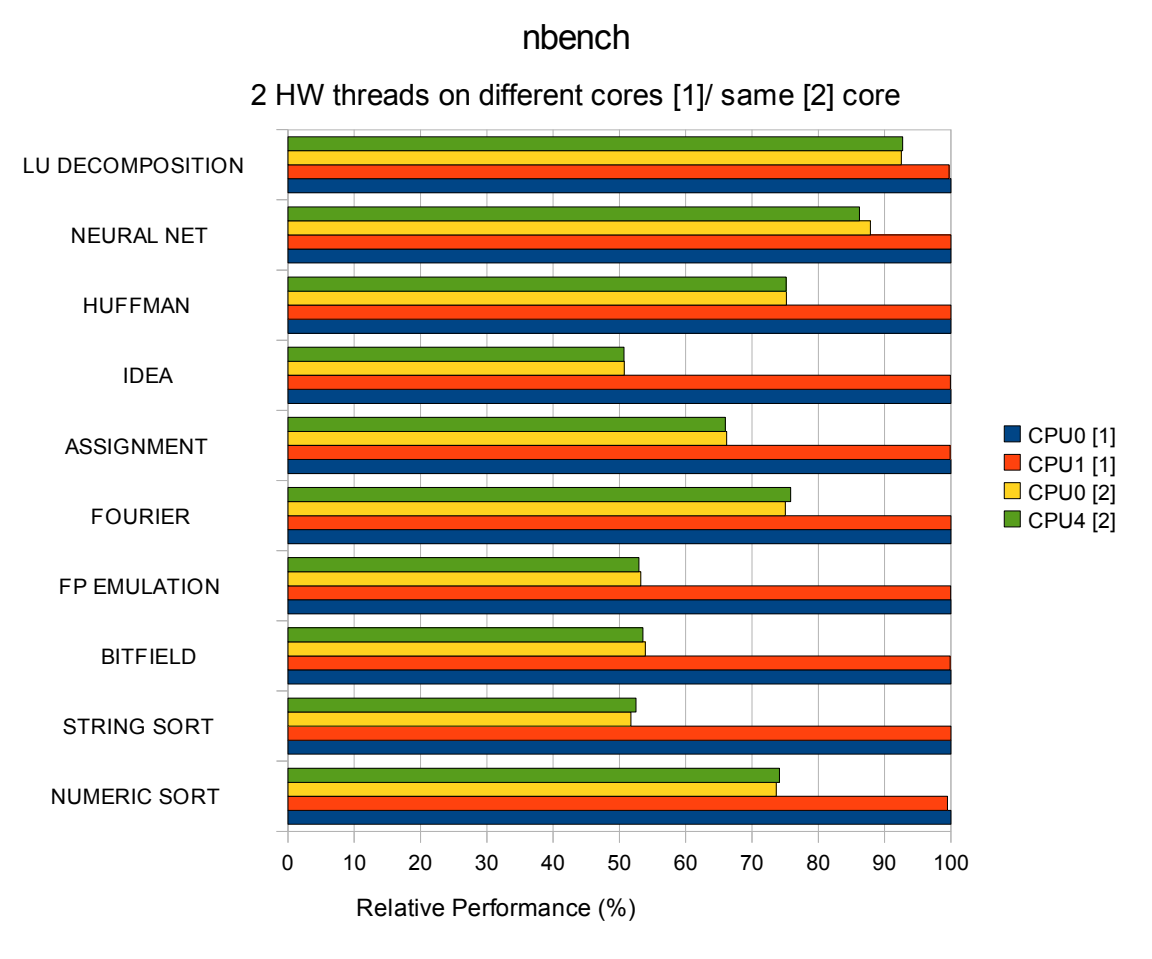
Why does it matter? - NUMA mem bandwidth

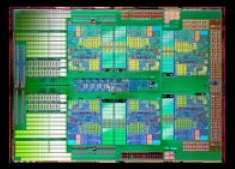




Why does it matter? - SMT effects

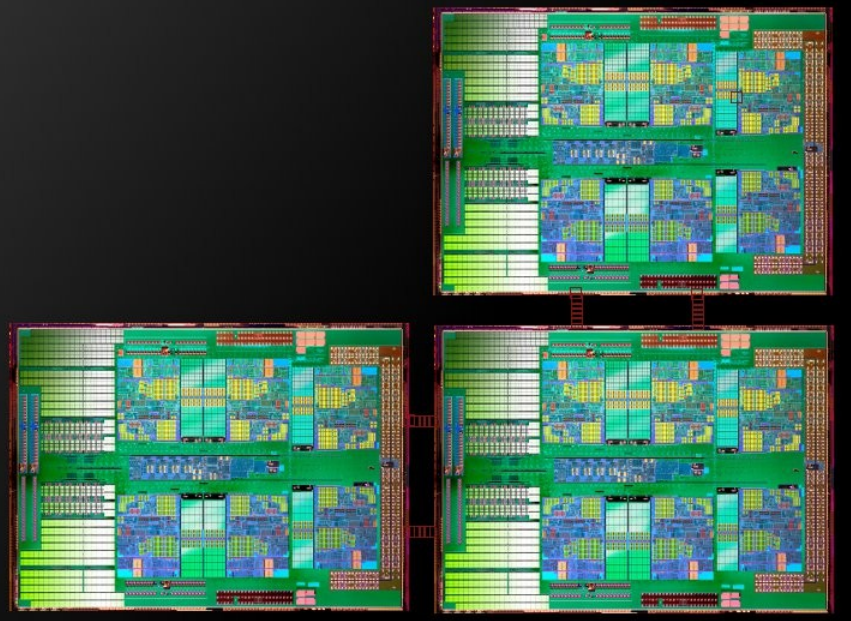






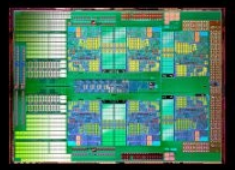
Topology Information Wanted

How to retrieve topology information



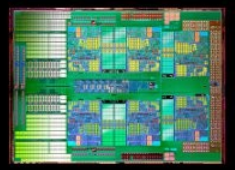
Topology Information Wanted

- Cache topology
  - Intel: CPUID leaf 4 to extract Cache_IDs “specific to a target level cache hierarchy” from APIC ID.
  - AMD: nothing equivalent
- Package topology
  - Intel: CPUID leaf 1 and 4 or on newer processors using CPUID leaf 11 (“the extended topology enumeration leaf”)
  - AMD: nothing equivalent (for multi-node CPUs an MSR was introduced for more convenient retrieval of the NodeId)

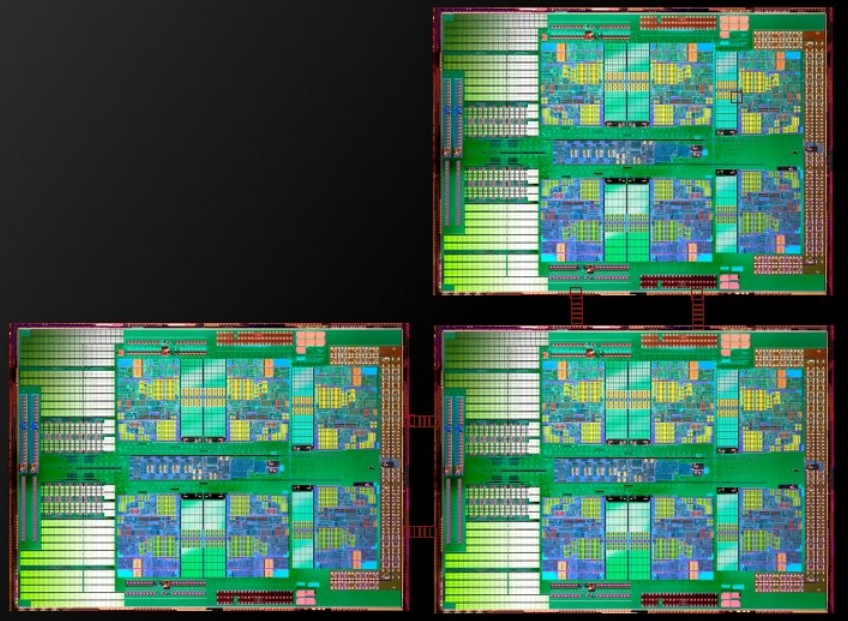


Topology Information Wanted (cont'd)

- NUMA topology
  - ACPI tables to propagate that information to the OS
    - System Resource Affinity Table (SRAT)
    - System Locality Distance Information Table (SLIT)
- “Power topology”
  - ACPI 3.0 _PSD (P-state dependency) objects:
    - Allow to describe P-state dependency domains
    - Can be used to define frequency domains (frequency change of one core causes frequency change of all other cores in that domain).
- So, there are some (well-)defined methods to retrieve the information. Rest is manual work.



Kernel Usage



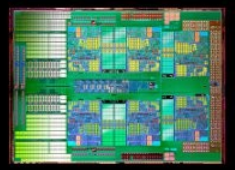
Kernel Usage

- Scheduler considers
  - Cache topology information (CPUs sharing their last level cache)
  - Package topology information (packages, cores, threads)
  - NUMA topology information (NUMA node)
  - Some hardcoded “power topology” information

  when creating scheduling groups and domains. This is the scheduler's tree of all the topology information.

  More details:

  - in `Documentation/scheduler/sched-domains.txt`
  - and in `kernel/sched.c` ;-)
- NUMA aware memory allocation

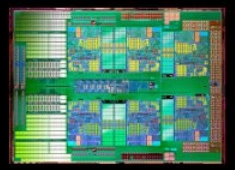


Kernel Usage (cont'd)

- The scheduling domain hierarchy and groups are used to do load-balancing and initial task placement.
- Default policy: best performance (balance tasks between different threads/cores/packages/nodes)
- Some knobs to change this topology-aware scheduling policy:

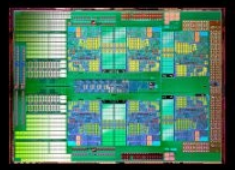
```
/sys/devices/system/cpu/sched_mc_power_savings
/sys/devices/system/cpu/sched_smt_power_savings
```

  Values are
  - 0 (default)
  - 1, 2 (powersavings balance) i.e. fully utilize a (thread)/core/package before other (threads)/cores/packages are utilized

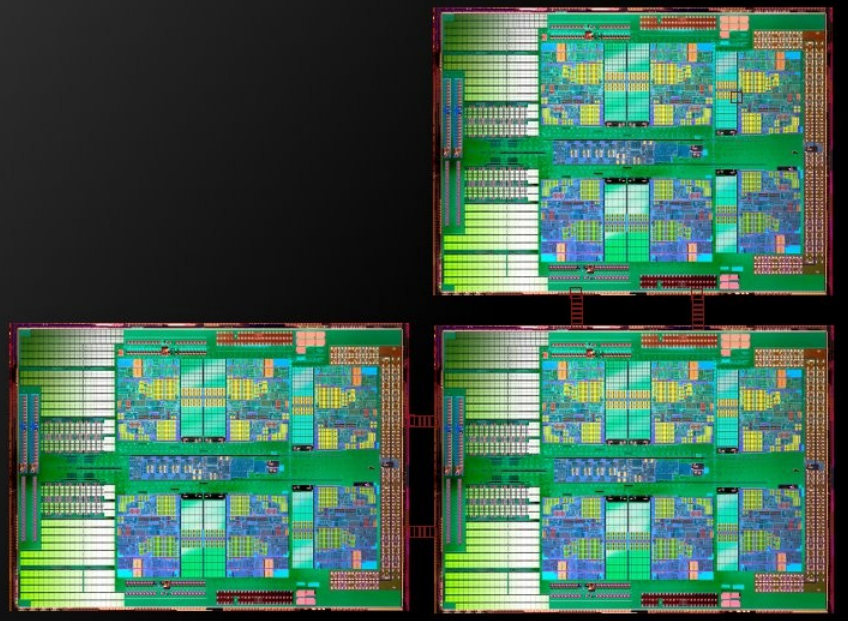


Kernel Usage - Shortcomings

- Current scheduling domain hierarchy cannot properly map topology for multi-node CPUs
  - Impact: powersavings balancing is not fully supported
- No direct connection between scheduler and cpufreq
  - No automatism to propagate power domain information into scheduling domains/groups
  - No uniform way to adapt both frequency scaling and topology aware scheduling policy with one knob.
- Different kinds of Core boosting (aka turbo boost, core performance boost) is another related topic
  - Work in progress



User Space



User Space – Getting the Information

What's in sysfs?

- Cache topology

  `/sys/devices/system/cpu/cpuX/cache/`

- Package topology

  `/sys/devices/system/cpu/cpuX/topology/`

  (Incomplete information for multi-node CPUs)

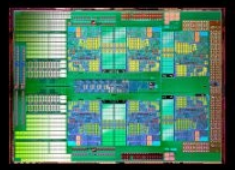
- NUMA topology

  `/sys/devices/system/node/`

- “Power topology”

  `/sys/devices/system/cpu/cpuX/cpufreq`

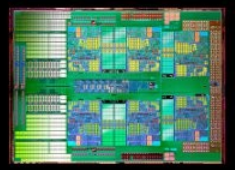
  (For frequency domain see `affected_cpus`)



User Space – Getting the Information (cont'd)

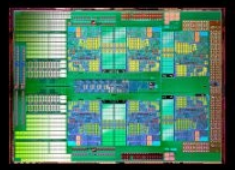
Tools

- Cache topology
  - `lstopo` (from `hwloc`)
- Package topology
  - dito
- NUMA topology
  - dito
  - `numactl`

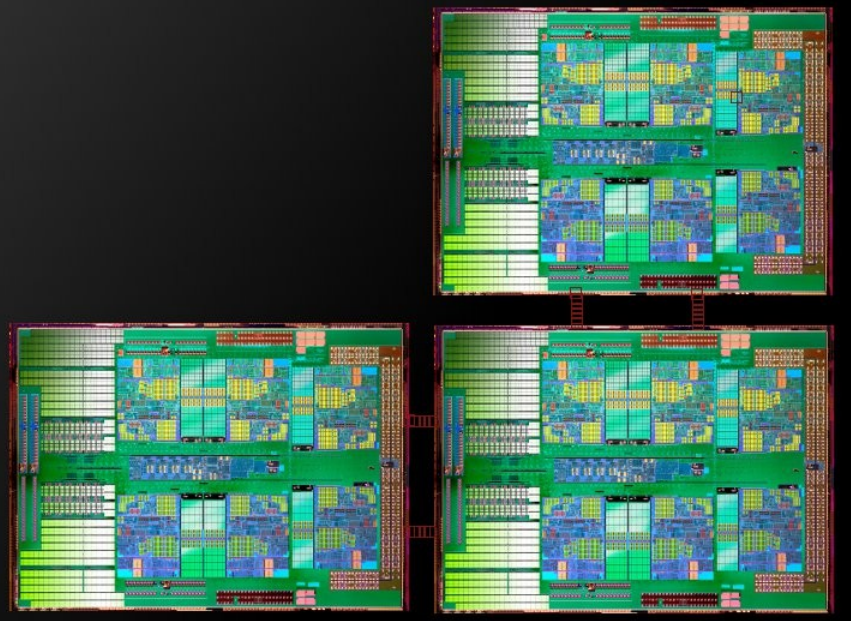


User Space – Using the Information

- If kernel scheduling decisions are inadequate for your application.
- Set CPU affinity mask for your task
  - Adapting your application
    - using `sched_setaffinity` or some library (e.g. `hwloc`)
    - Con: You need the code and the skill for modifications.
    - Pro: It's the most flexible way to control scheduling.
  - Using tools (wrappers for sched_setaffinity)
    - `numactl`, taskset, schedtool
    - Easy to use, but you can't do everything



References



References

- Zhang, E. Z., Jiang, Y., and Shen, X. *Does cache sharing on modern CMP matter to the performance of contemporary multithreaded programs?.* 2010. In Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (Bangalore, India, January 09 - 14, 2010). PPoPP '10. ACM, New York, NY, 203-212. DOI= http://doi.acm.org/10.1145/1693453.1693482
- François Broquedis, Jérôme Clet-Ortega, Stéphanie Moreaud, Nathalie Furmento, Brice Goglin, Guillaume Mercier, Samuel Thibault, and Raymond Namyst. *hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications.* 2010. In Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP2010), Pisa, Italia, February 2010. IEEE Computer Society Press. http://hal.inria.fr/inria-00429889
- *CPUID Specification.* 2008. Rev. 2.28. AMD Publication # 25481.
- *Intel® 64 Architecture Processor Topology Enumeration.* 2009. http://software.intel.com/en-us/articles/intel-64-architecture-processor-topology-enumeration/
- *Intel® Processor Identification and the CPUID Instruction.* Application Note 485. 2009.