Non-uniform memory access (NUMA)

处理器内核与主内存之间的内存访问不一致。

- Memory access between processor core to main memory is not uniform.

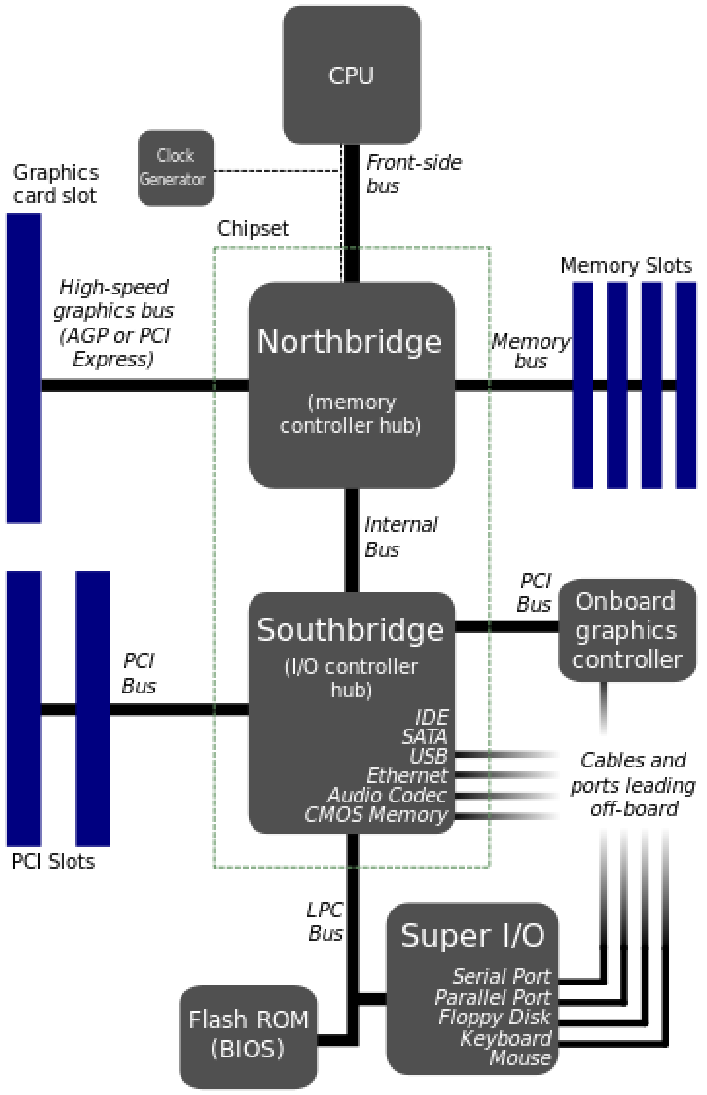
内存位于称为“ NUMA域 ”的单独区域中。

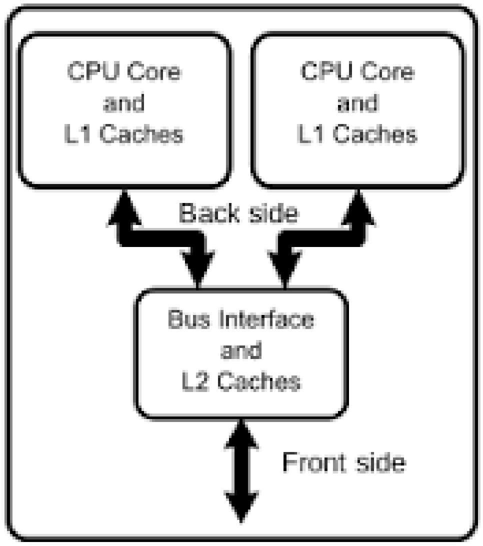
- Memory resides in separate regions called “NUMA domains.”
- For highest performance, cores should only access memory in its nearest NUMA domain.

Memory buses and controllers

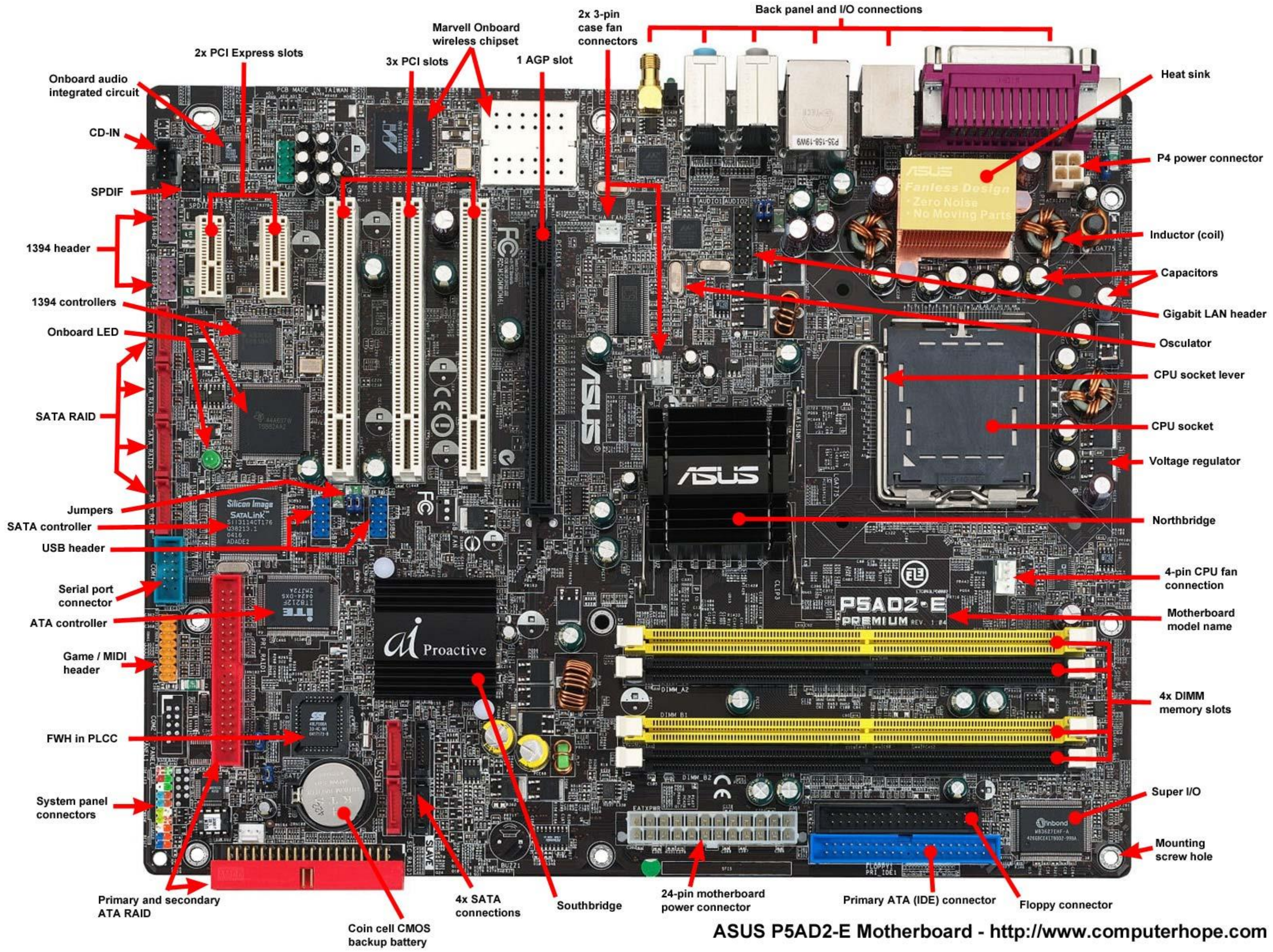
- Pre-2008: Front side bus
- Post-2008: QuickPath interconnect

Front Side Bus (FSB) and Multicore



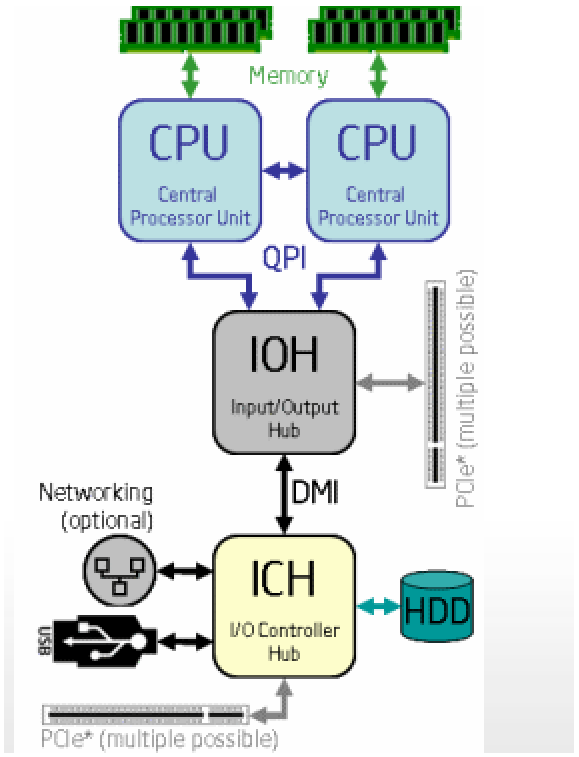


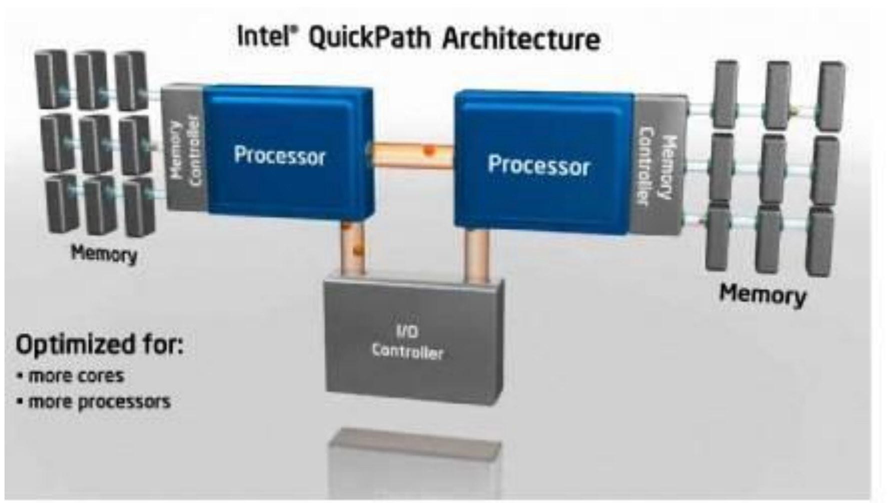
- FSB was used up to the Core microarch. (2008)
- Memory access by many cores or sockets across the FSB is a bottleneck



ASUS P5AD2-E Motherboard - http://www.computerhope.com

Hypertransport & QuickPath Interconnect

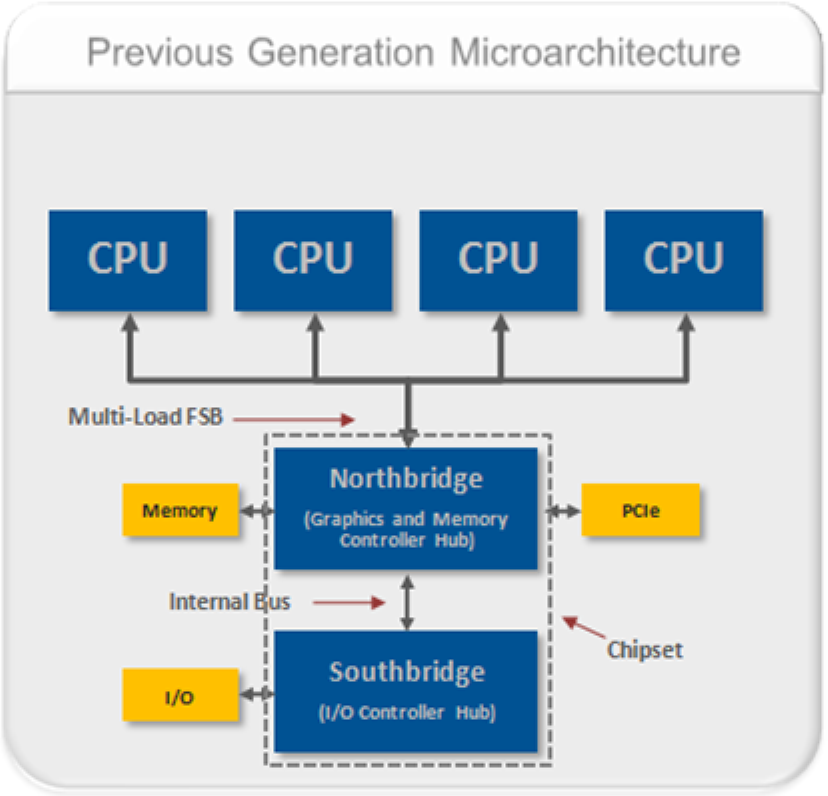




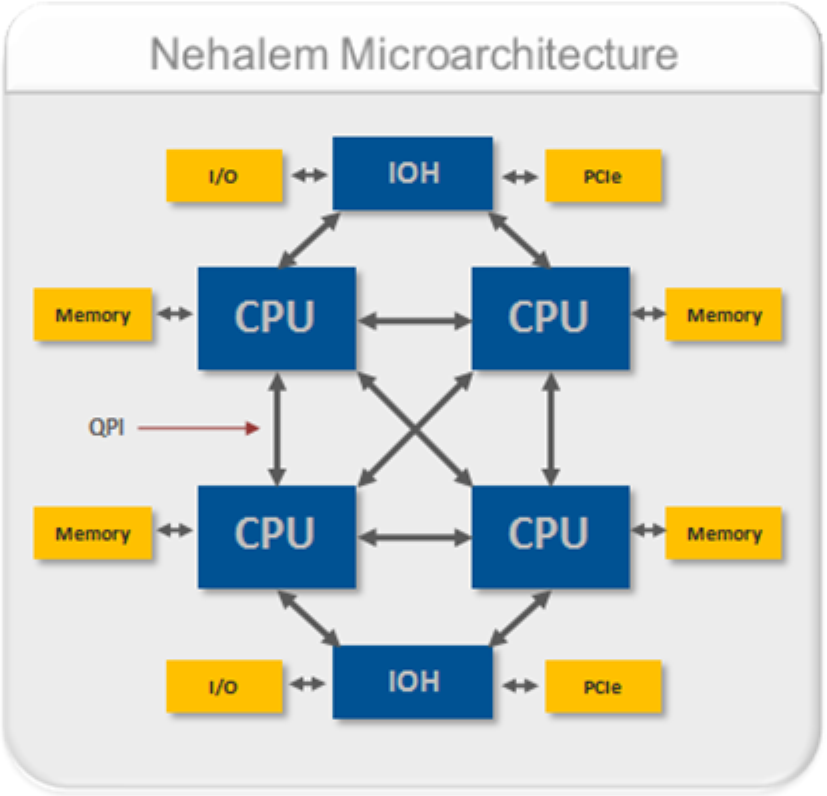
- AMD HyperTransport (2003)
  Intel QuickPath Interconnect (2008)
- Memory controller now on the CPU
- Memory access is now **non-uniform**
  内存访问现在不统一

QuickPathInterconnect

FSB vs QPI



Previous Generation Microarchitecture
CPU
CPU
CPU
CPU
Multi-Load FSB
Memory
Northbridge (Graphics and Memory Controller Hub)
PCIe
Internal Bus
Chipset
I/O
Southbridge (I/O Controller Hub)



UMA = Uniform Memory Access

SMP = Symmetric MultiProcessor

NUMA = Non-Uniform Memory Access

Non-Uniform Memory Access (NUMA)

- NUMA architectures support higher aggregate bandwidth to memory than UMA architectures
- “Trade-off” is non-uniform memory access
- Can NUMA effects be observed?
- Locality domain: a set of processor cores and its locally connected memory (a.k.a. locality “node”)

- View the NUMA structure (on Linux):

  ```
  numactl --hardware
  ```

  numactl can also be used to control NUMA behavior

NUMA

- How does the OS decide where to put data?
- How does OS decide where to place the threads?

OS是如何断定往哪里写数据？在哪里放线程？

- Ref: Hager and Wellein, Sec 4.2, Chap 8, App A
- Ref: Xeon Phi book, Sec 4.4.5

Page placement by first touch

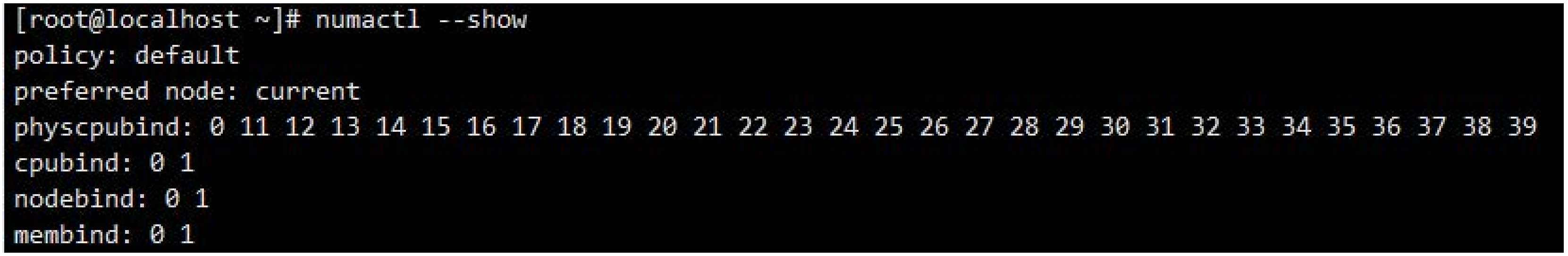
- A page is placed in the locality region of the processor that first touches it (not when memory is allocated)
- If there is no memory in that locality domain, then another region is used
- Other policies are possible (e.g., set preferred locality domain, or round-robin, using numactl)

numactl – set memory affinity

```
numactl [--interleave=nodes] [--preferred=node]
[--membind=nodes] [--localalloc] <command> [args] ...
```

```
numactl --hardware
numactl --show
```

```
numactl –interleave=0,1 <command>
```



```
[root@localhost ~]# numactl --show
policy: default
preferred node: current
physcpubind: 0 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
cpubind: 0 1
nodebind: 0 1
membind: 0 1
```

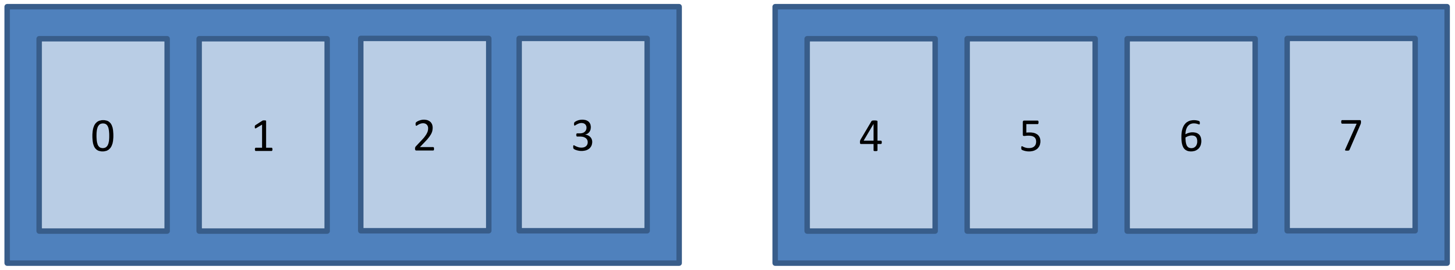
Thread affinity

- Threads can be pinned to cores (otherwise they may move to any core)
  - A thread that migrates to a different core usually will no longer have its data in cache (note, some subset of cores may share some levels of cache)
  - A thread that migrates to a different locality region may have its data in a different locality region
- For programs compiled with Intel compilers, specify affinity at runtime using KMP_AFFINITY environment variable

KMP_AFFINITY environment variable

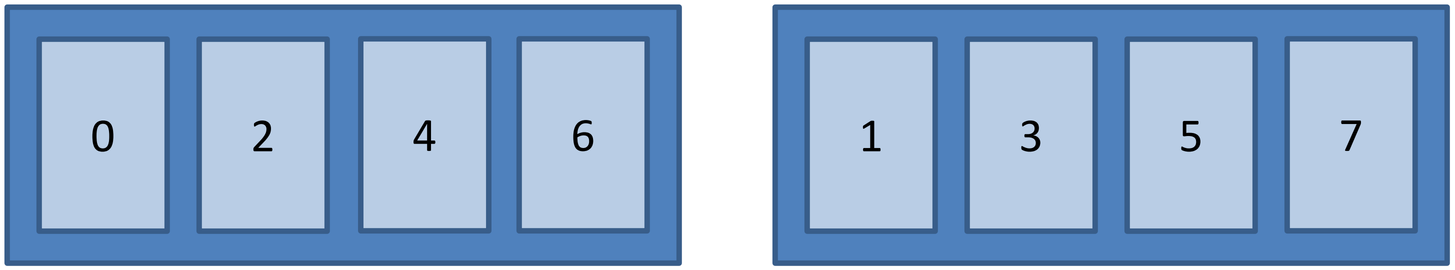
2 sockets, 4 cores/socket

compact 紧凑



Might not use all sockets

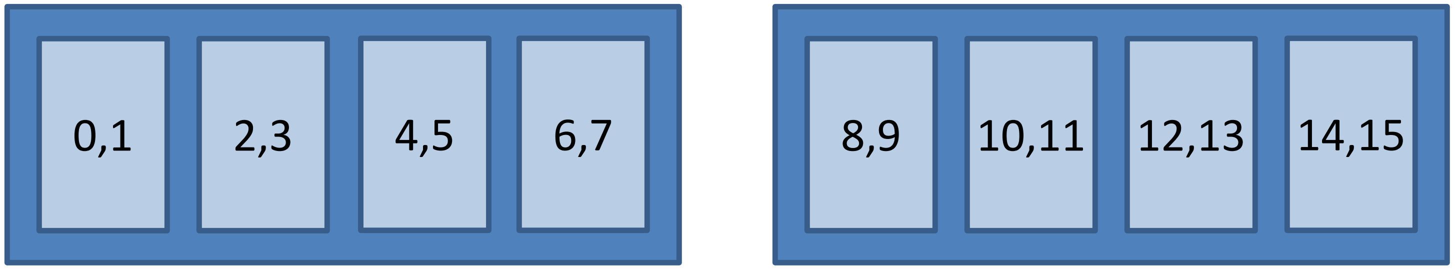
scatter 分散



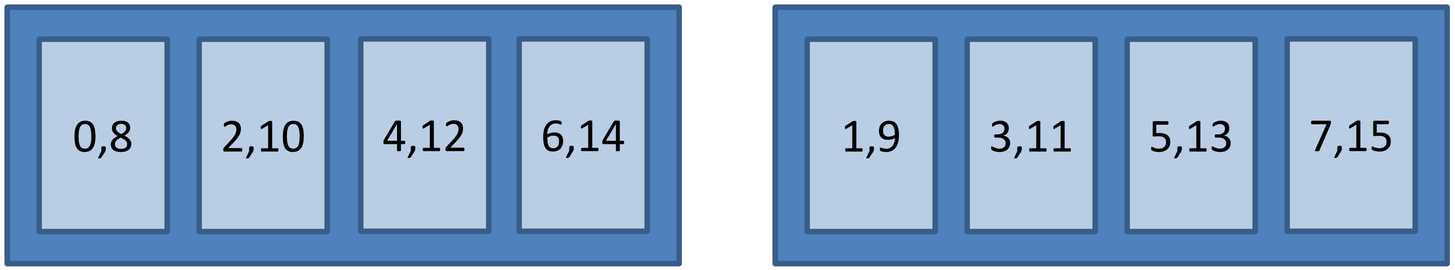
KMP_AFFINITY environment variable

2 sockets, 4 cores/socket, 2-way hyperthreading

granularity=core,compact



granularity=core,scatter

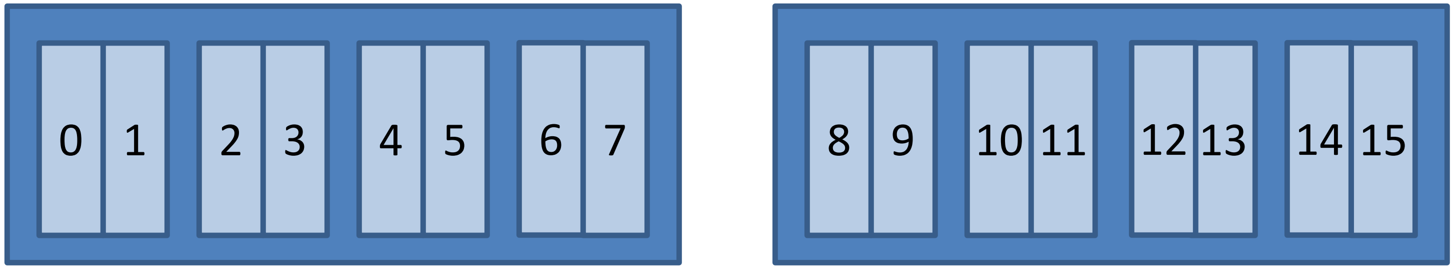


Thread can only float between hardware contexts on a core

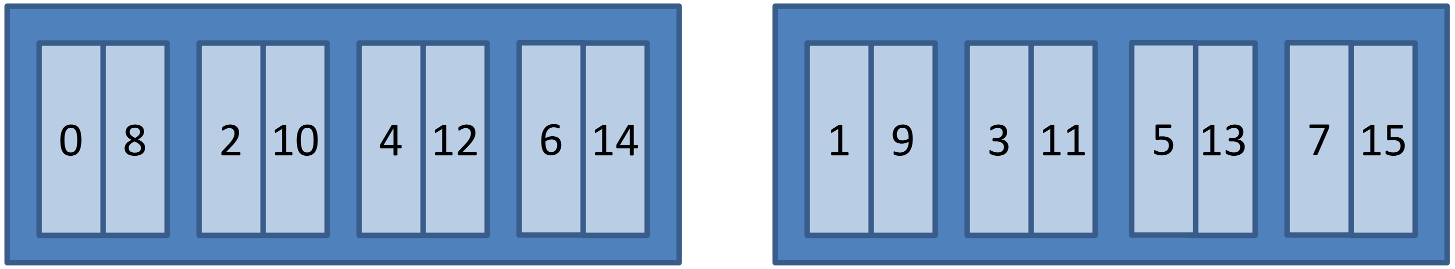
KMP_AFFINITY environment variable

2 sockets, 4 cores/socket, 2-way hyperthreading

granularity=fine,compact

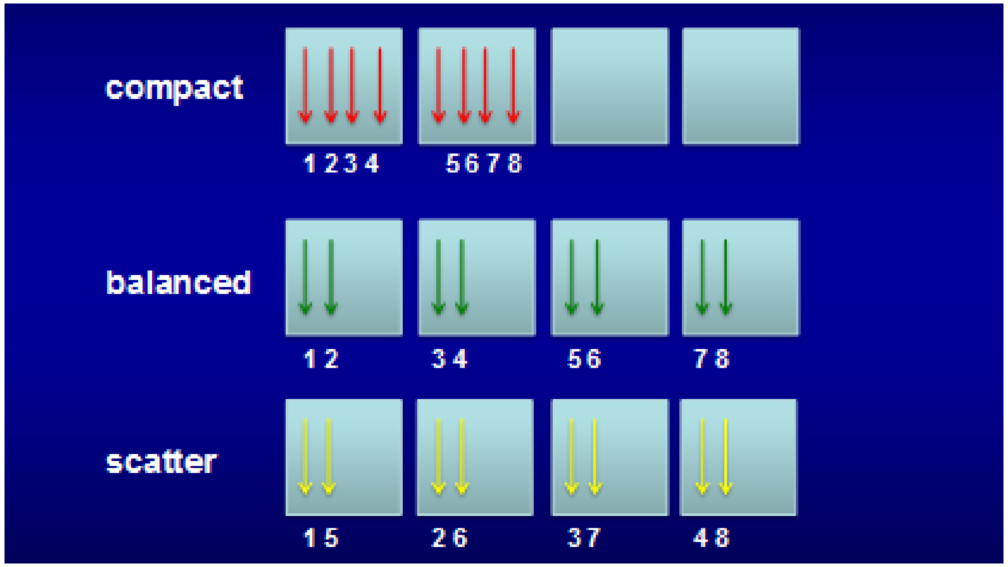


granularity=fine,scatter



Usually no need to bind threads to hardware thread contexts

KMP_AFFINITY on Intel Xeon Phi



4 cores and 4 HW threads per core.
Note: “balanced” is only for Intel Xeon Phi.

Examples

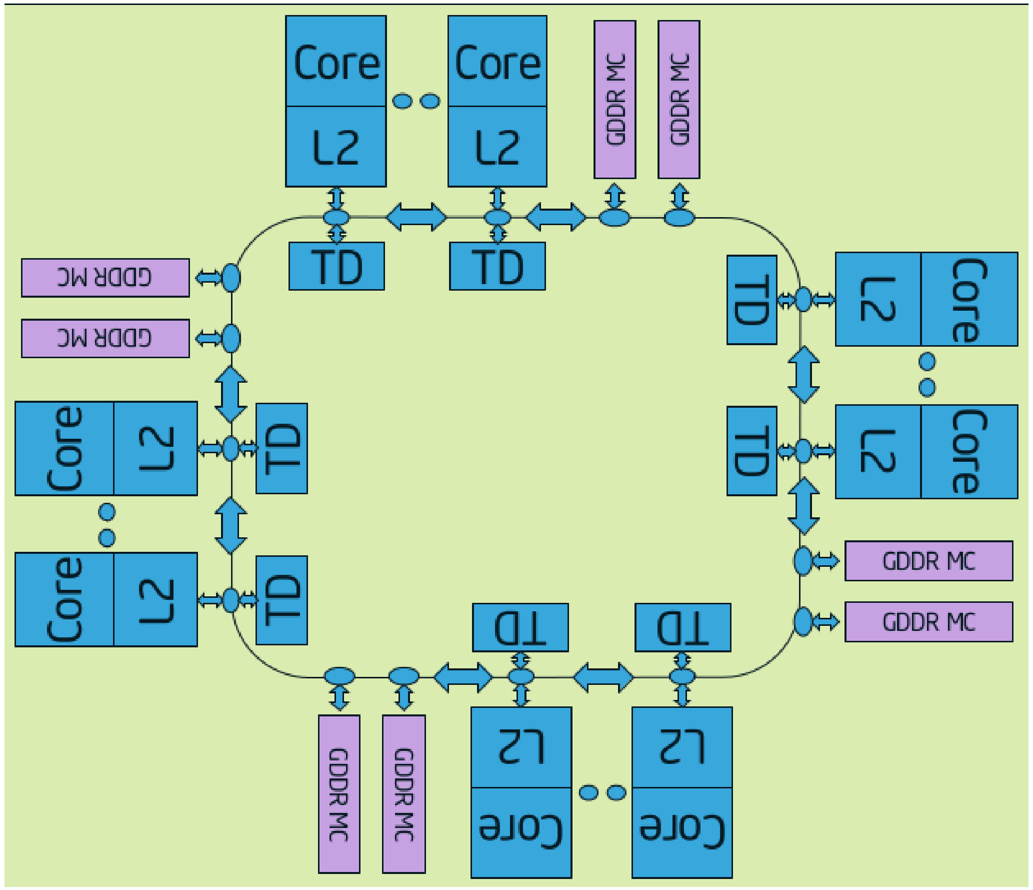
KMP_AFFINITY=“verbose,none” (print affinity information)
KMP_AFFINITY=“balanced”
KMP_AFFINITY=“granularity=fine,balanced” (bind to single hardware thread)
KMP_AFFINITY=“granularity=core,balanced”
(bind to any hardware thread on core; core is default granularity)

How to choose different options

- For BW-bound codes, the optimum number of threads is often less than the number of cores. Distribute and choose number of threads to reduce contention and fully utilize the memory controllers. (Use scatter.)
- For compute-bound codes, use hyperthreading and all available hyperthreads. Threads operating on adjacent data should be placed on the same core (or nearby) in order to share cache. (Use compact.)
- On Intel Xeon Phi, “balanced” often works best. (Probably should have this option on CPUs!)

- Example: bench-dgemm on MIC

Intel Xeon Phi



Technically UMA, but access to L2 cache is nonuniform

Intel compiler optimization reports

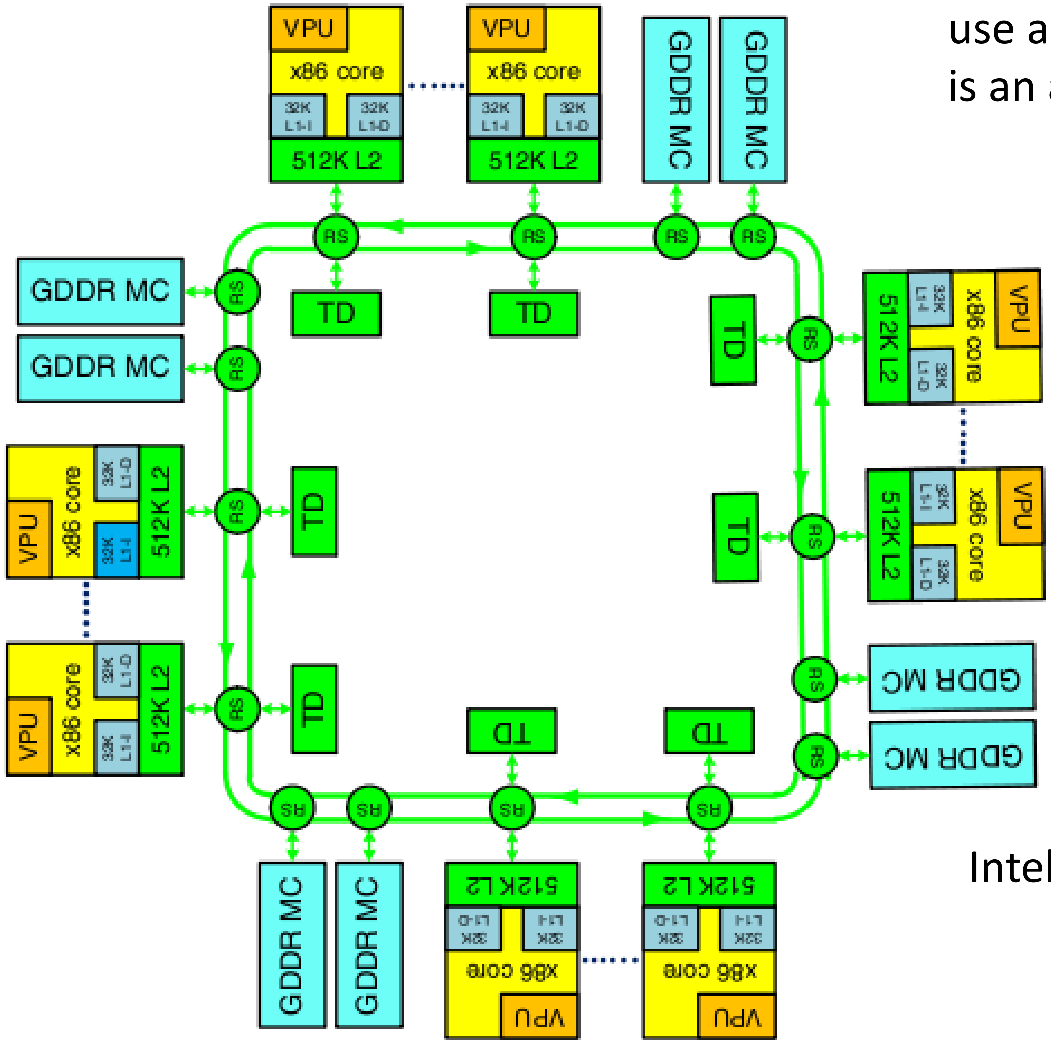
- icc … -qopt-report=[0-5]
  icc … -qopt-report-phase=vec

  compiler will output an optimization report in the file xx.optrpt

Backup Figures

Ring Bus to Connect Cores and Caches/DRAM

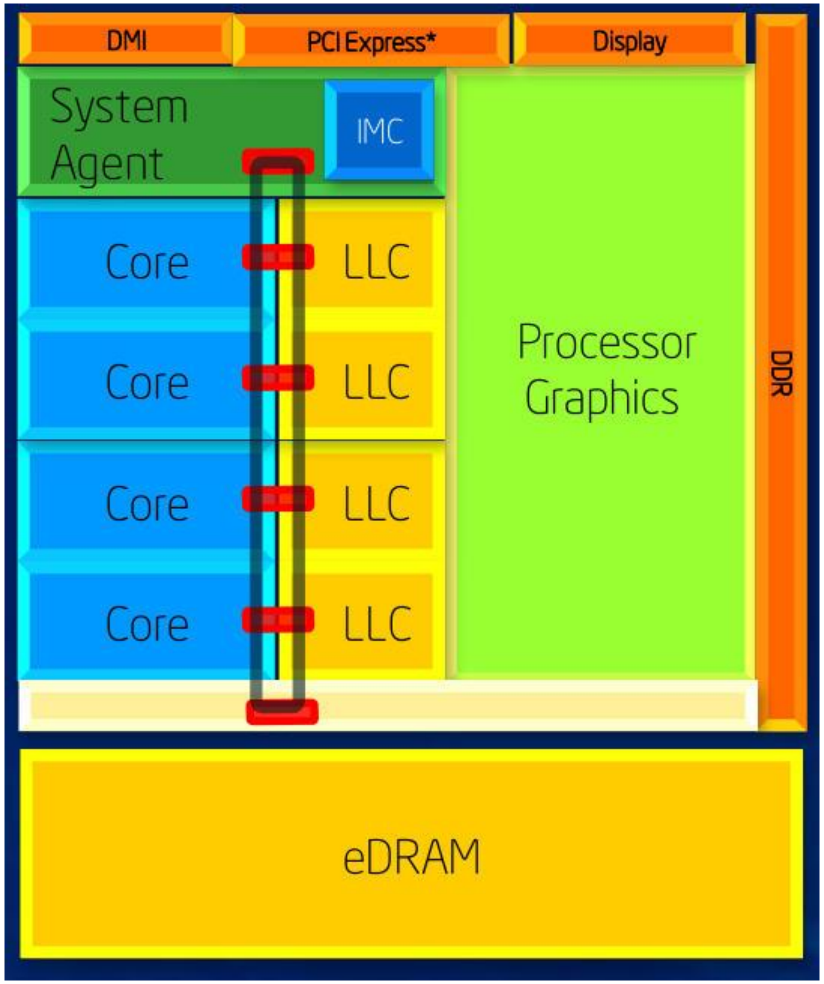
All current Intel CPUs use a ring bus (alternative is an all-to-all switch)



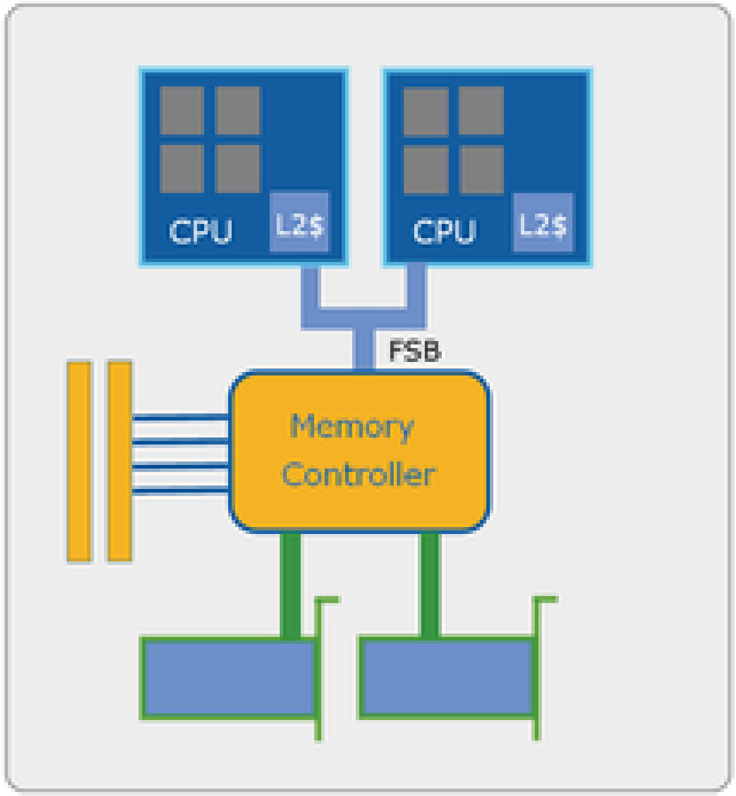
Intel Xeon Phi

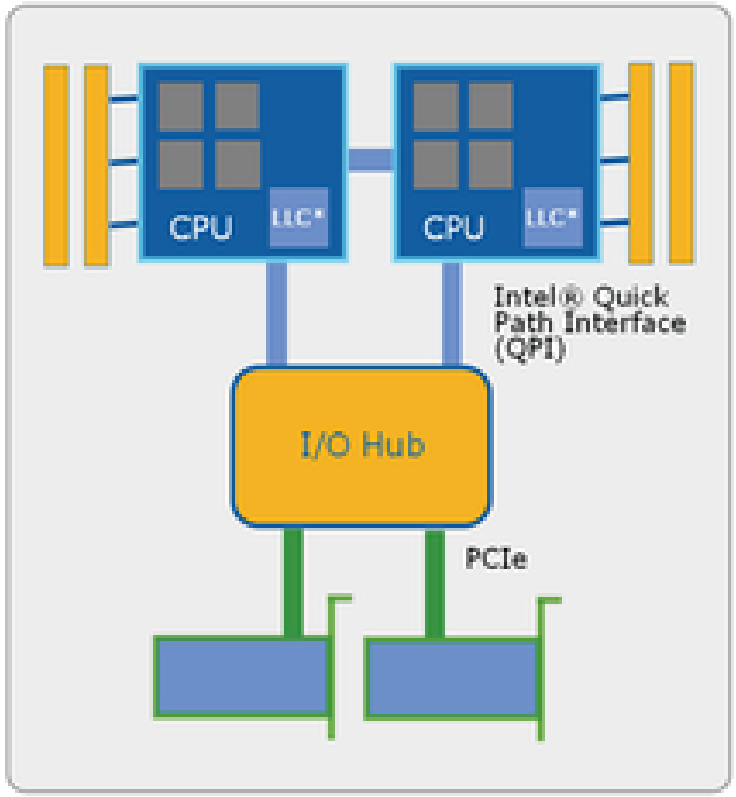
环形总线

Ring Bus on Haswell

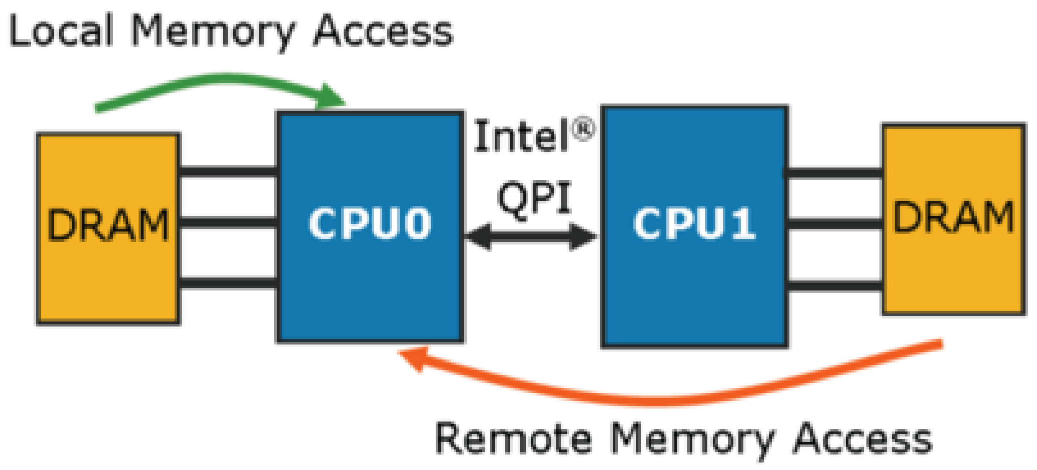


Front Side Bus (FSB) vs. QuickPath Interconnect (QPI)

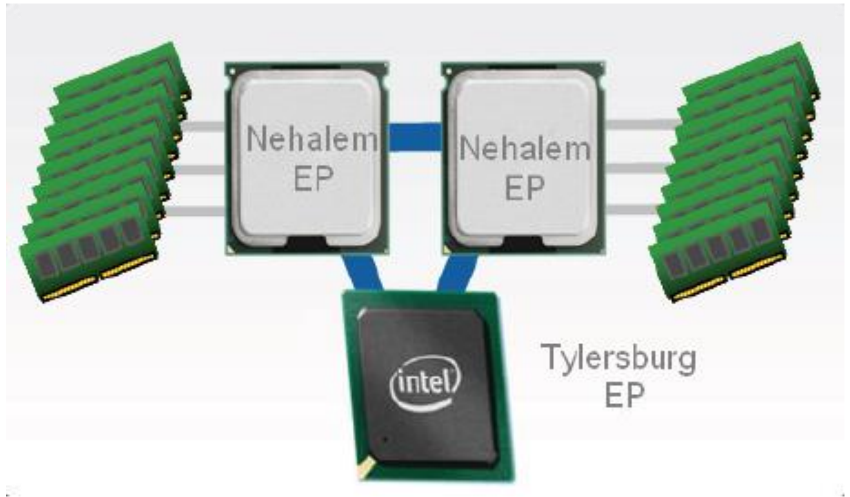




QuickPath Interconnect



NUMA: Non-Uniform Memory Access


Nehalem EP
Nehalem EP
Tylersburg EP