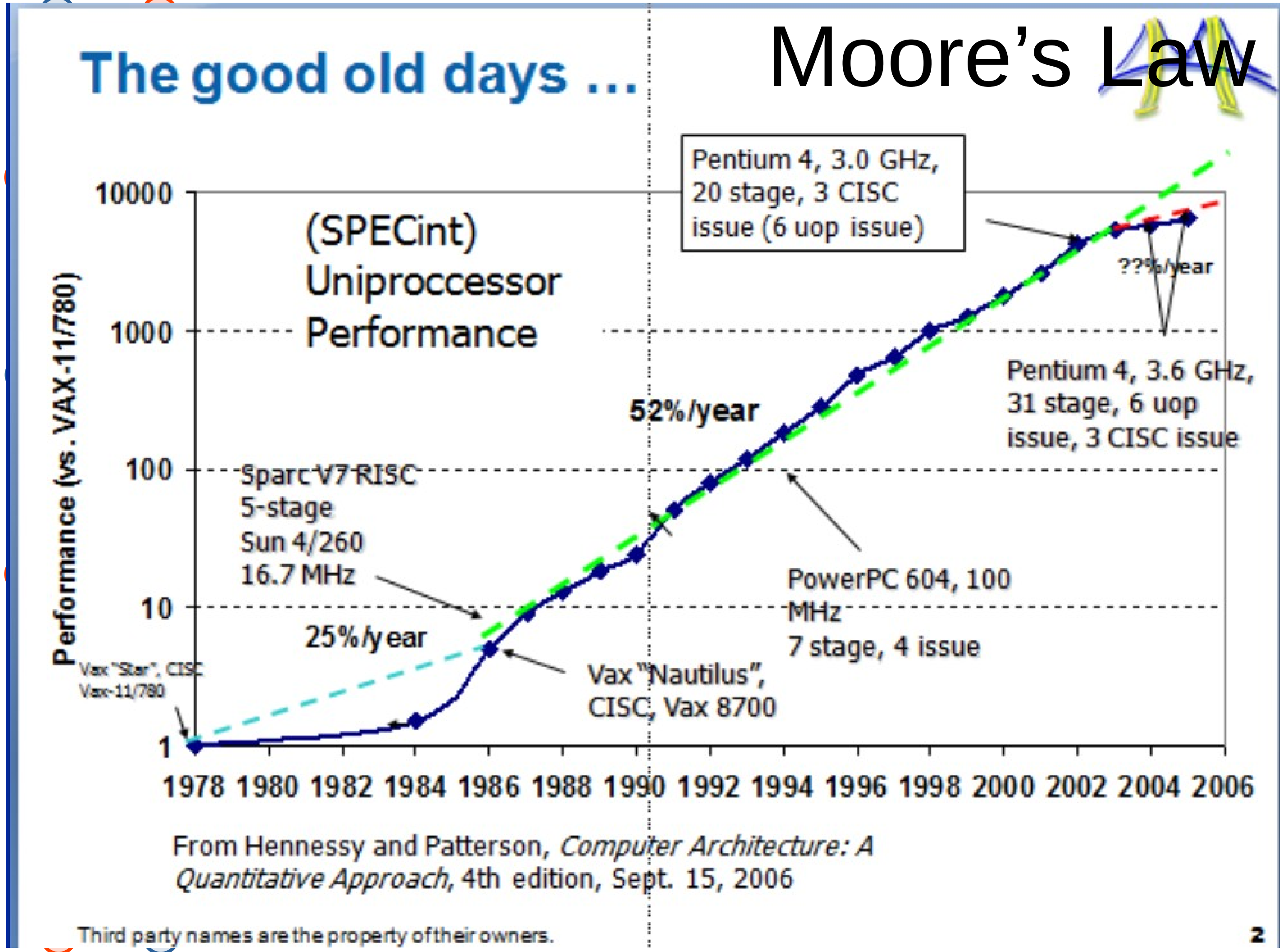
High Performance Computing

Shared Memory Parallel CPU & OpenMP

Computer Eng., KMITL

Assoc. Prof. Dr. Surin. K.

Moore's Law



The good old days …

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

Moore’s Law

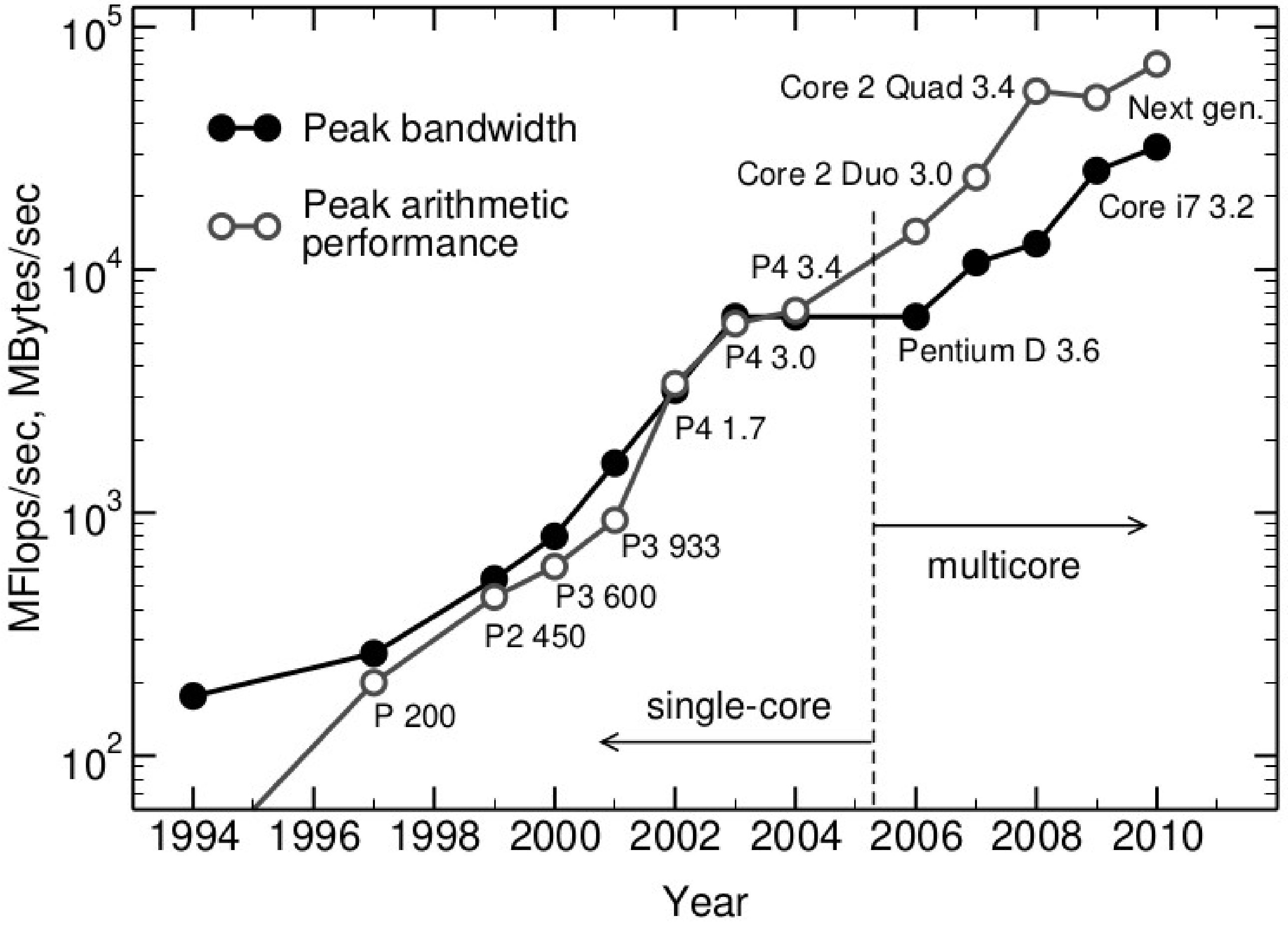
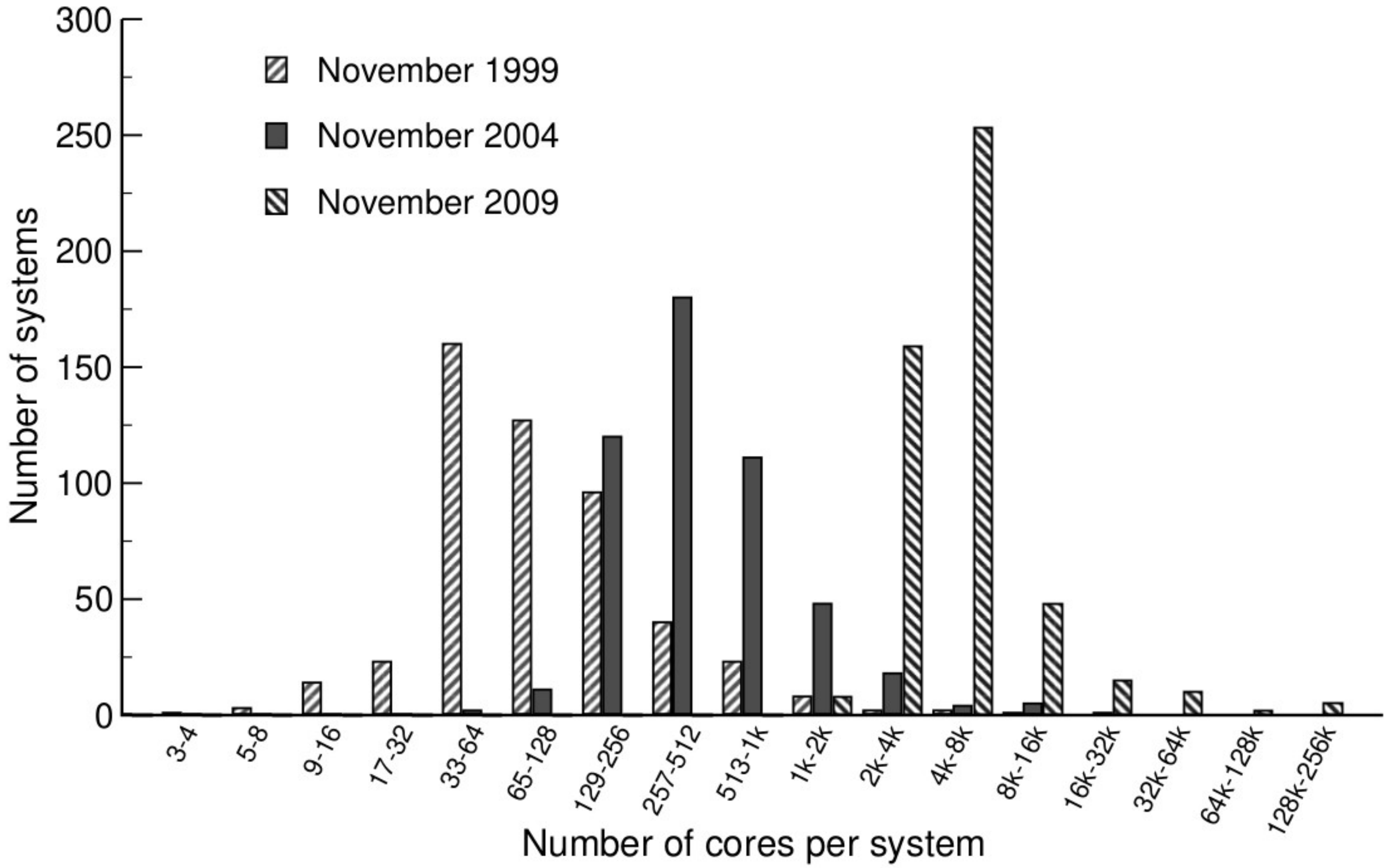


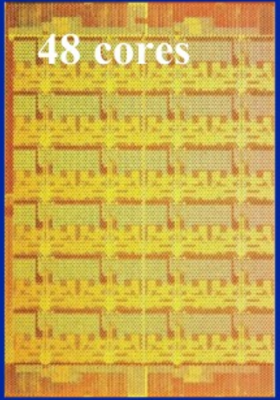
Figure 3.2: Progress of maximum arithmetic performance (open circles) and peak theoretical memory bandwidth (filled circles) for Intel processors since 1994. The fastest processor in terms of clock frequency is shown for each year. (Data collected by Jan Treibig.)

Number of systems versus core count



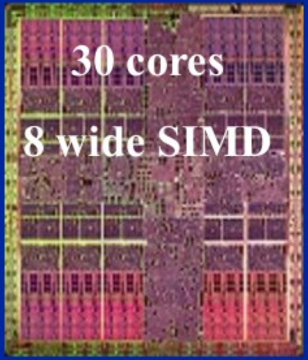
Microprocessor trends

Individual processors are many core (and often heterogeneous) processors.

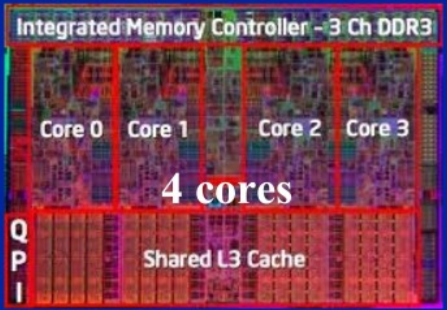


Intel SCC Processor

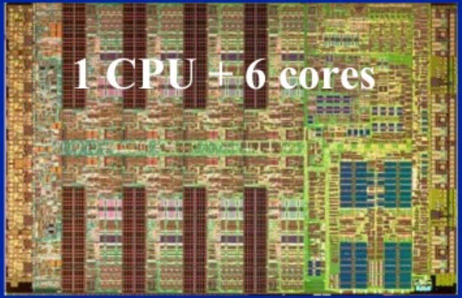
AMD ATI RV770



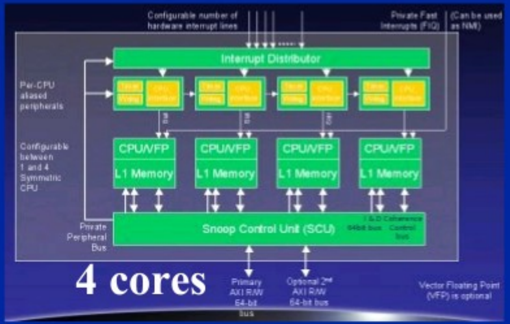
NVIDIA Tesla C1060



Intel® Xeon® processor



IBM Cell

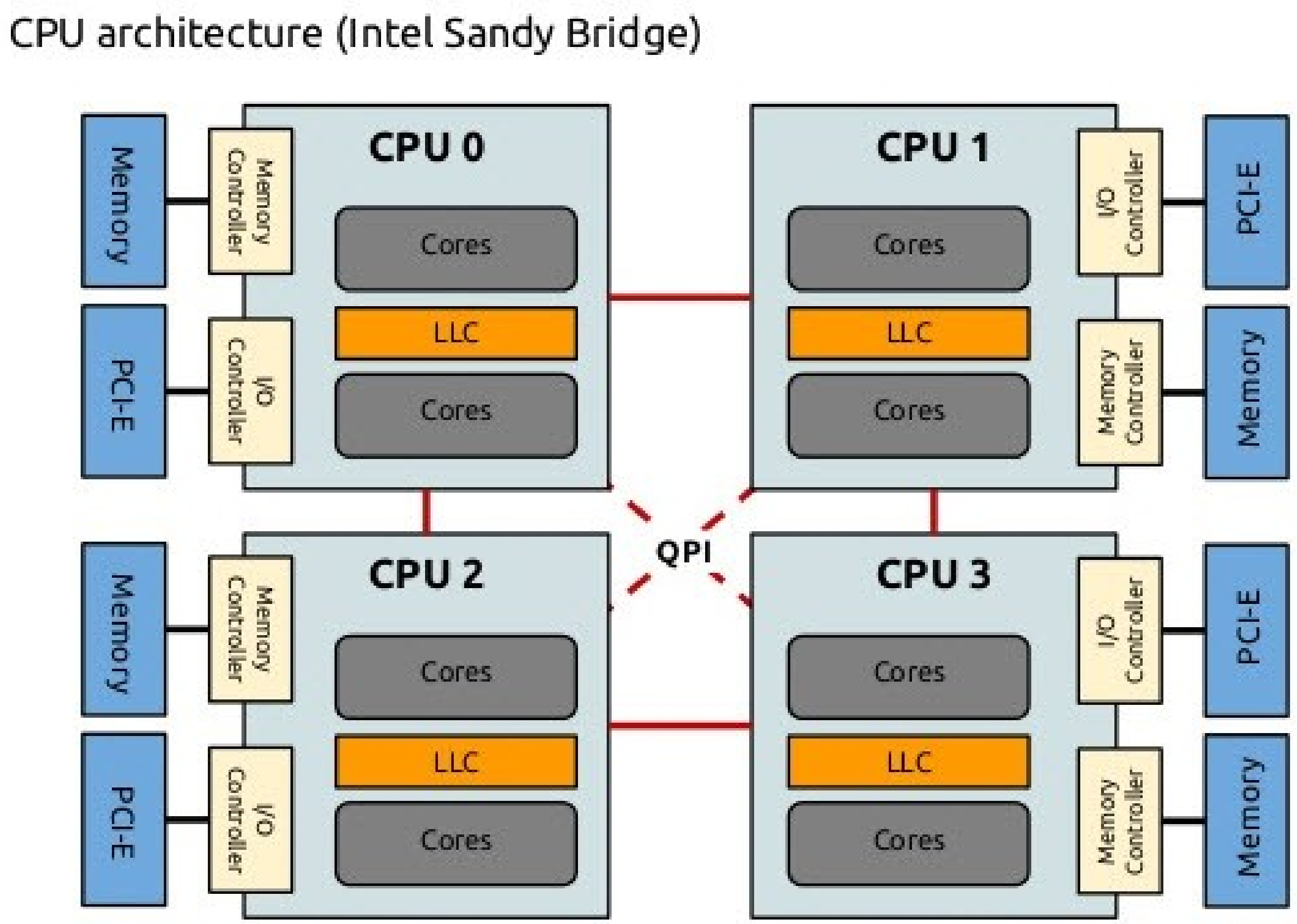

Configurable number of hardware interrupt lines
Private Fast Interrupts (FIQ)
(Can be used as NMI)
Interrupt Distributor
Per-CPU aliased peripherals
Configurable between 1 and 4 Symmetric CPU
CPU/VFP
L1 Memory
CPU/VFP
L1 Memory
CPU/VFP
L1 Memory
CPU/VFP
L1 Memory
Private Peripheral Bus
Snoop Control Unit (SCU)
I & D Coherence 64bit bus
Control bus
4 cores
Primary AXI R/W 64-bit bus
Optional 2nd AXI R/W 64-bit bus
Vector Floating Point (VFP) is optional

ARM MPCORE

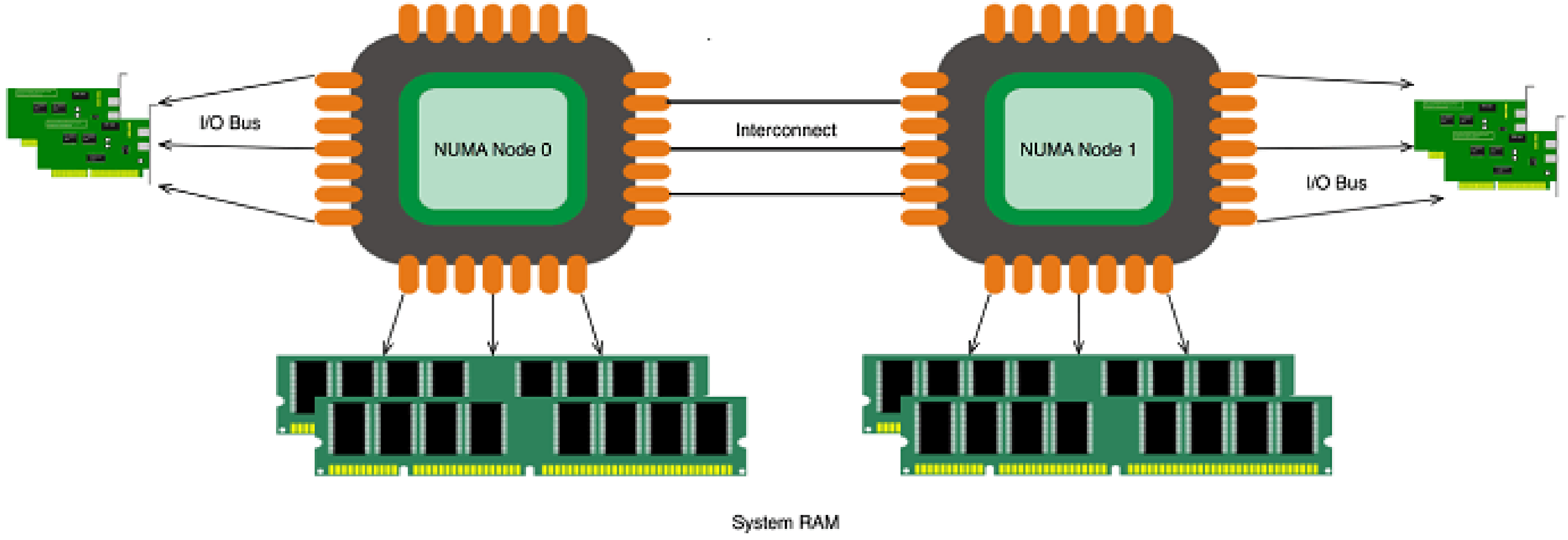
Source: OpenCL tutorial, Gaster, Howes, Mattson, and Lokhmotov, HiPEAC 2011

3rd party names are the property of their owners.

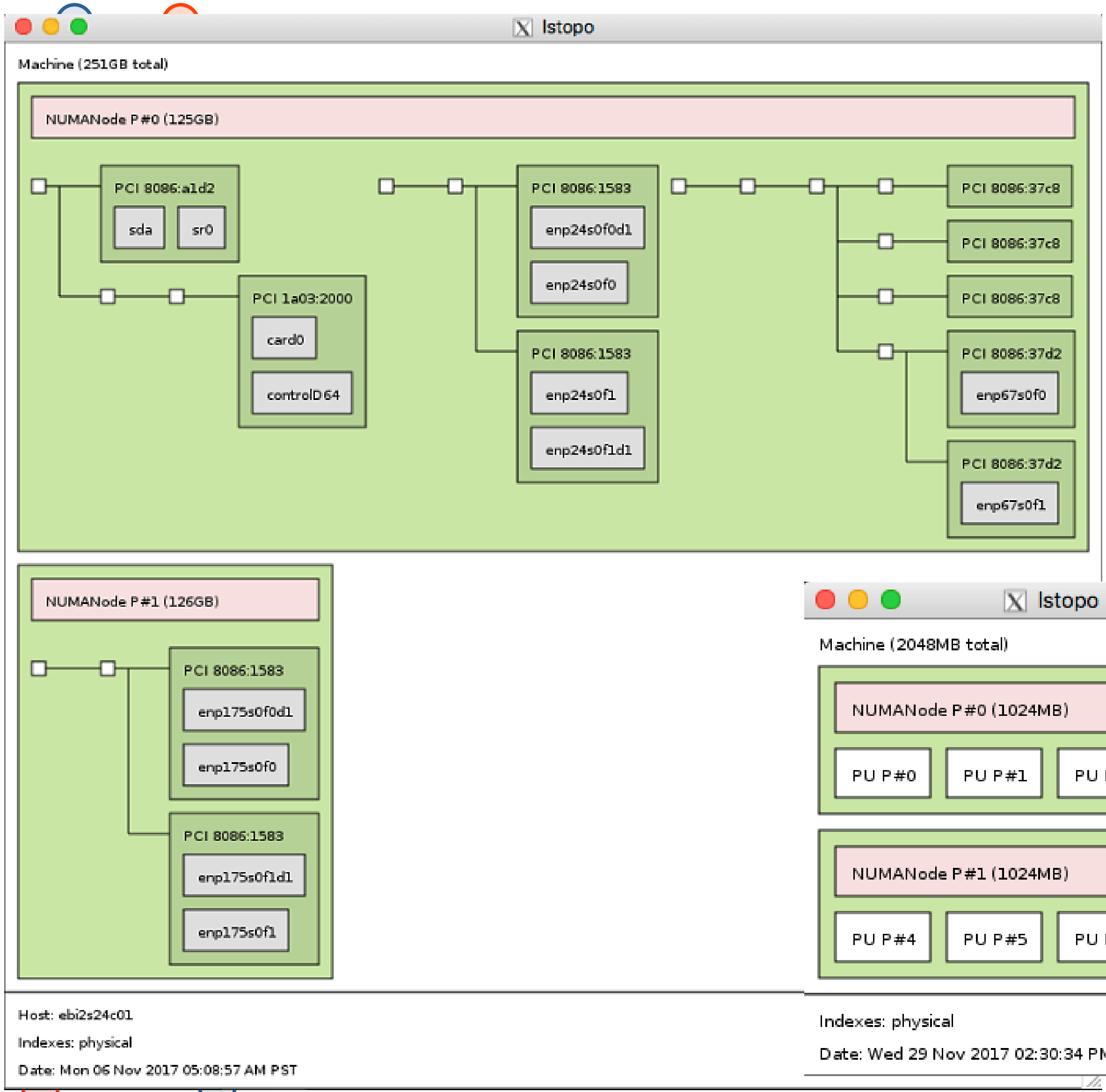
NUMA: Intel Sandy Bridge



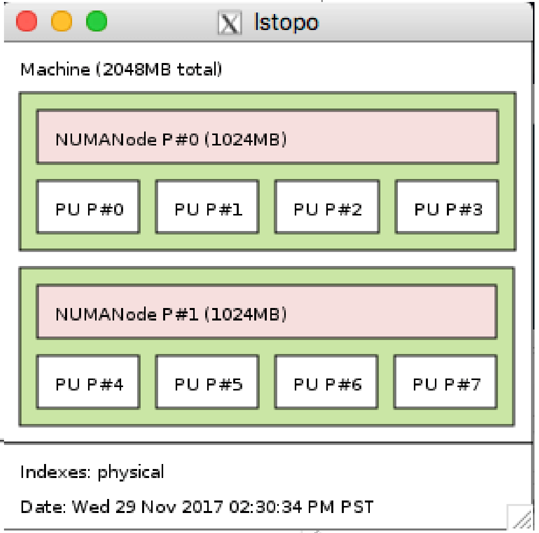
NUMA: Intel Quick Assist Technology



[NUMA: Intel Quick Assist Technology]

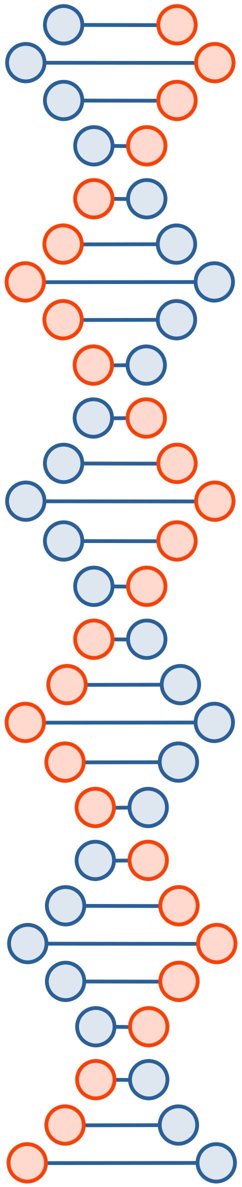


Host: ebi2s24c01
Indexes: physical
Date: Mon 06 Nov 2017 05:08:57 AM PST

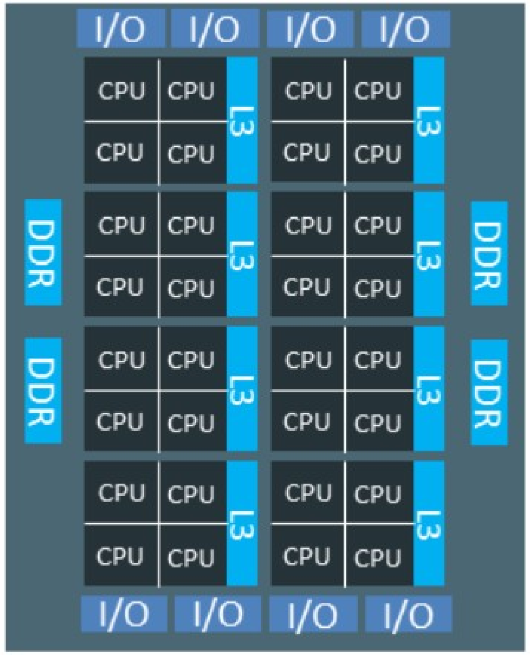


Indexes: physical
Date: Wed 29 Nov 2017 02:30:34 PM PST

NUMA: AMD Epyc



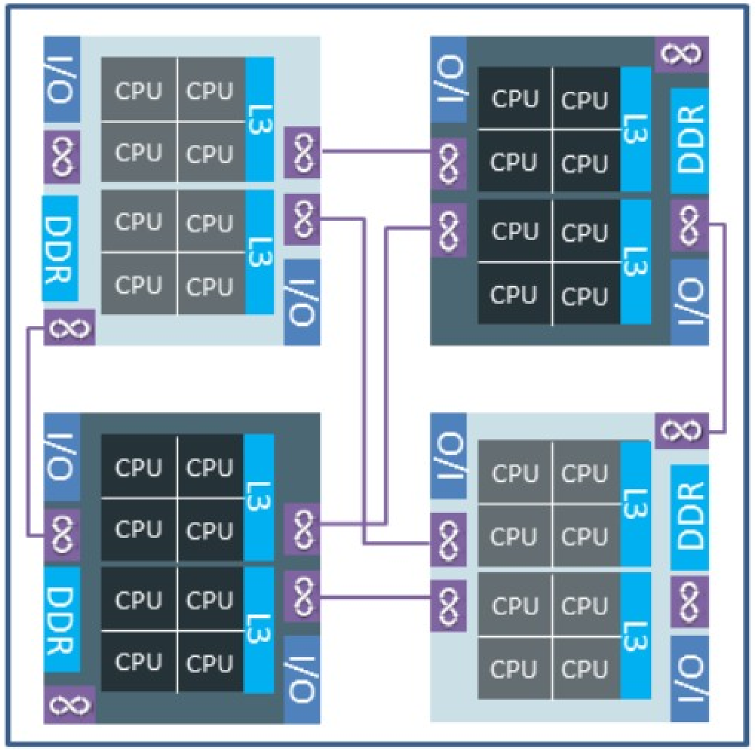
Monolithic Die



32C Die Cost

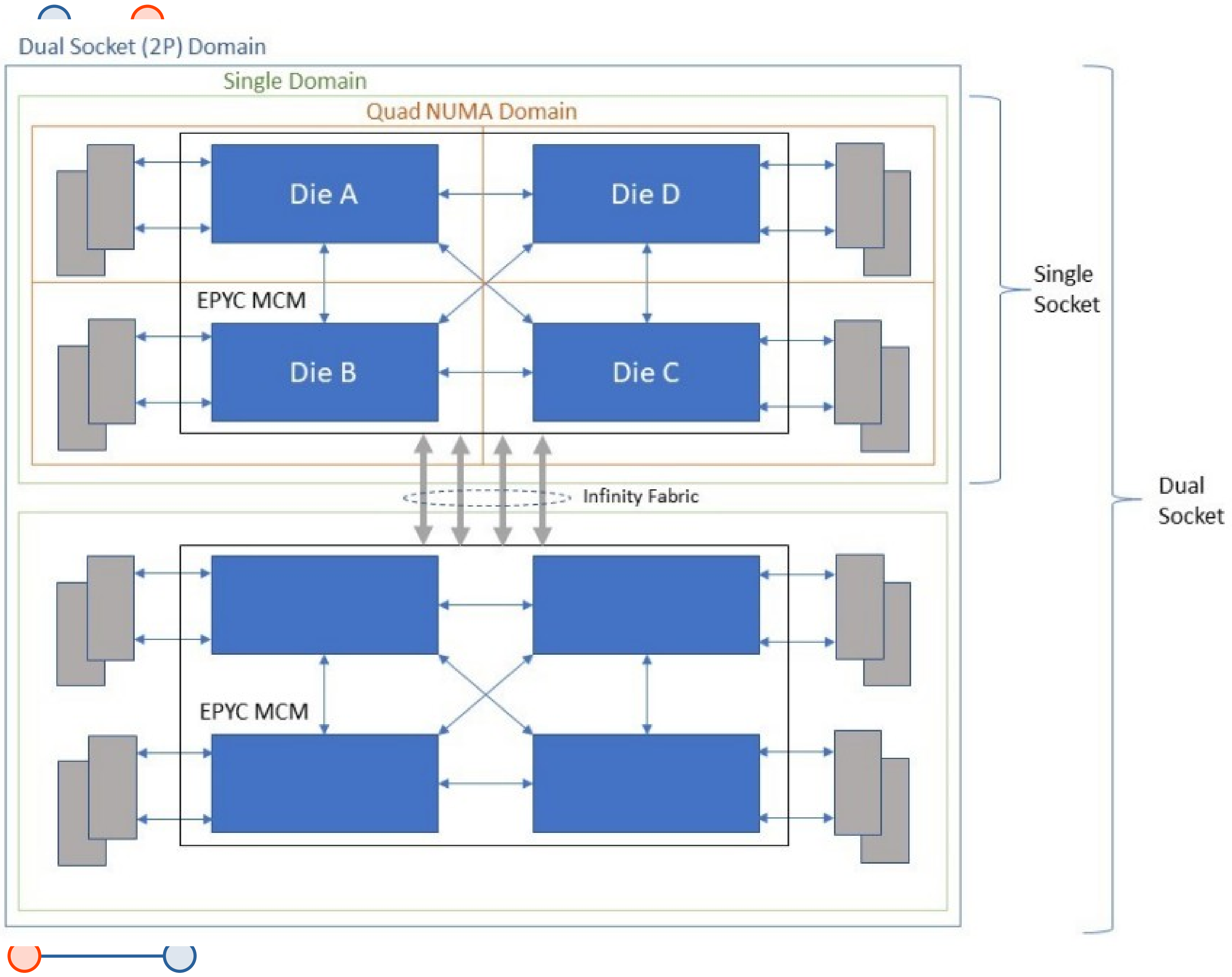
1.0X

EPYC MCM



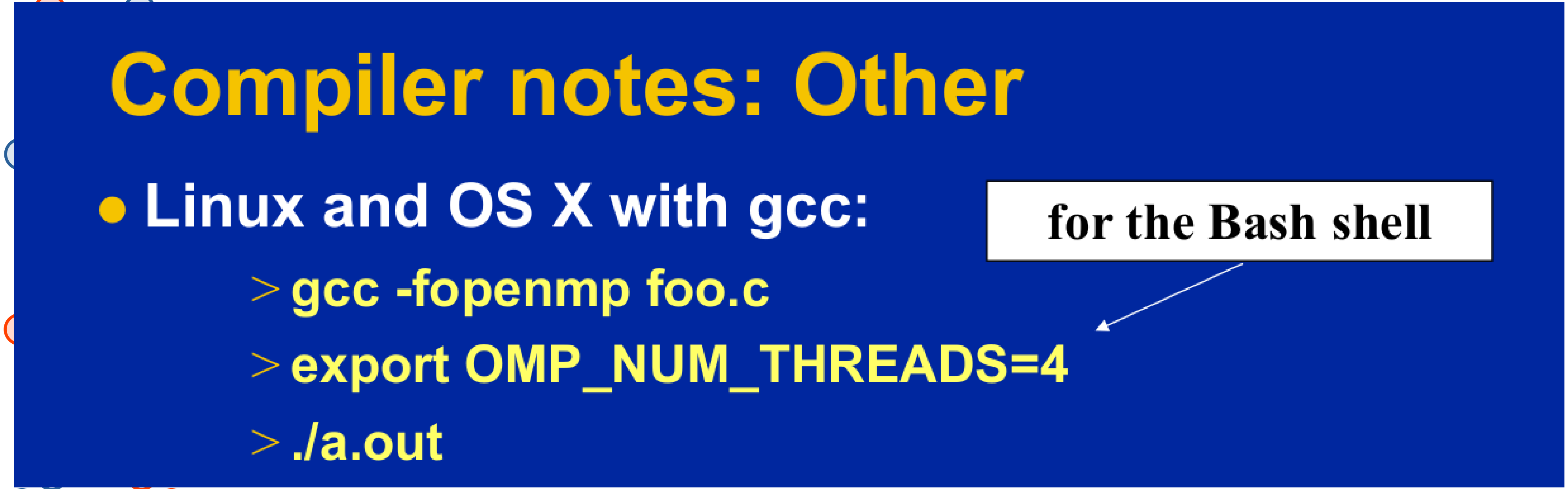
4 x 8C Die Cost

0.59X[1]



NUMA: AMD Epyc

OpenMP



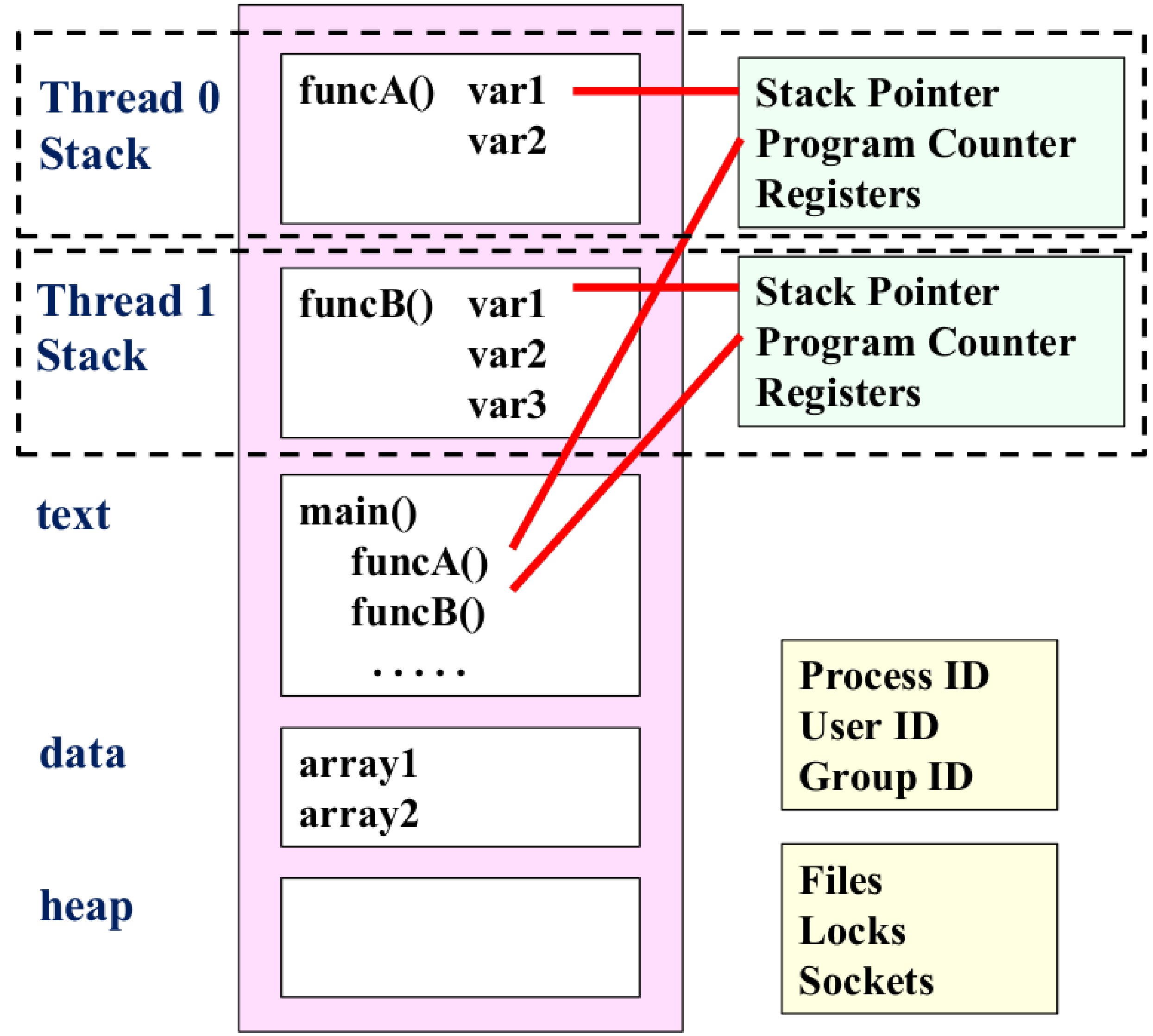
Compiler notes: Other

- Linux and OS X with gcc:

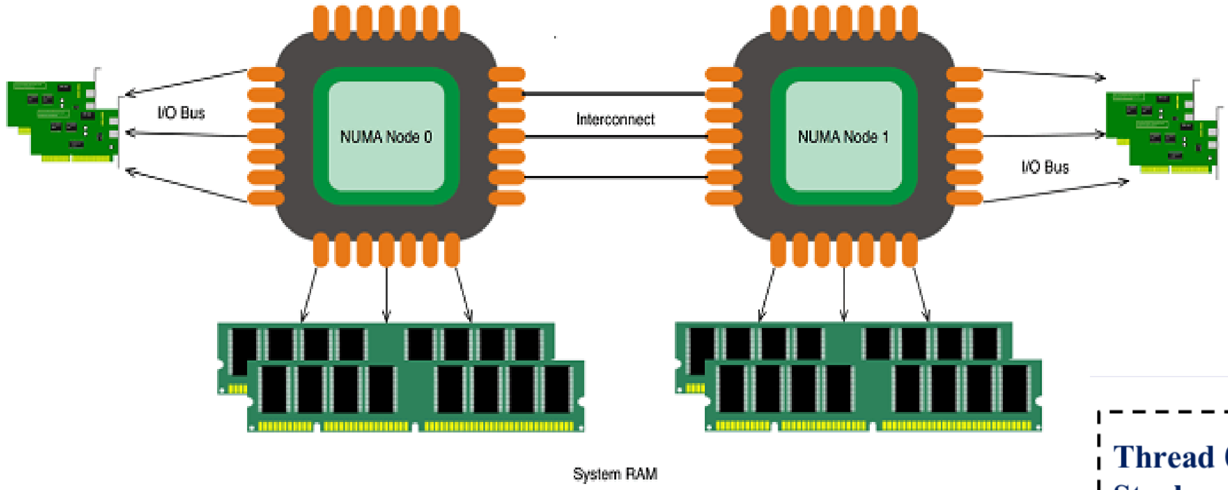
```
> gcc -fopenmp foo.c
> export OMP_NUM_THREADS=4
> ./a.out
```

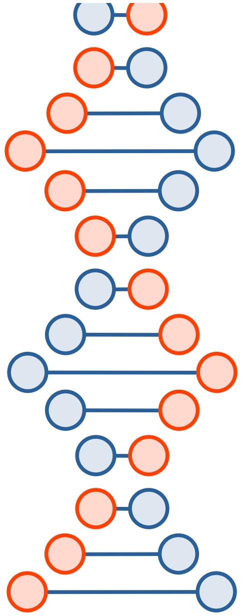
for the Bash shell

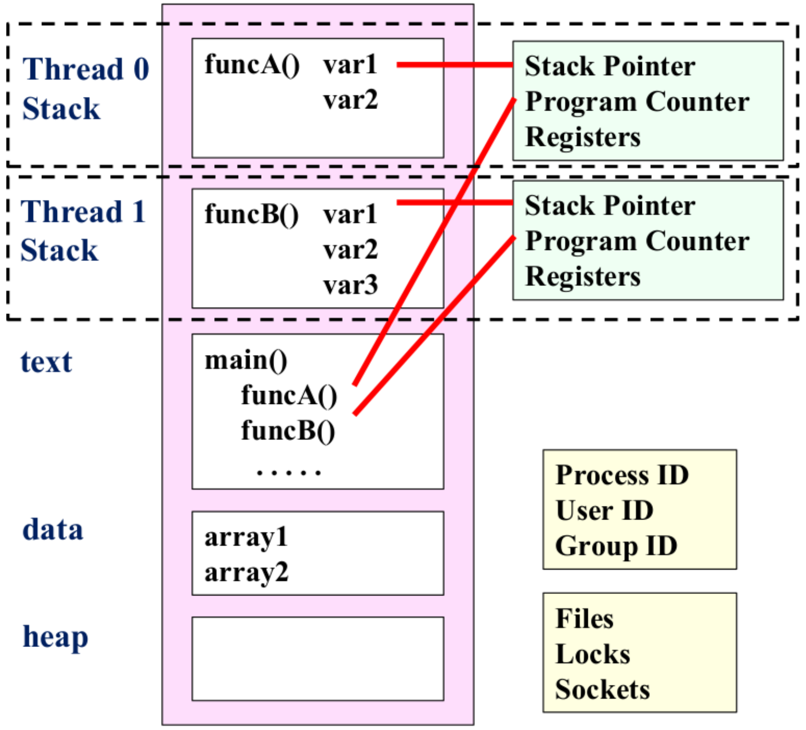
OpenMP & NUMA


Thread 0 Stack
funcA() var1
var2
Stack Pointer
Program Counter
Registers
Thread 1 Stack
funcB() var1
var2
var3
Stack Pointer
Program Counter
Registers
text
main()
funcA()
funcB()
.....
data
array1
array2
heap
Process ID
User ID
Group ID
Files
Locks
Sockets

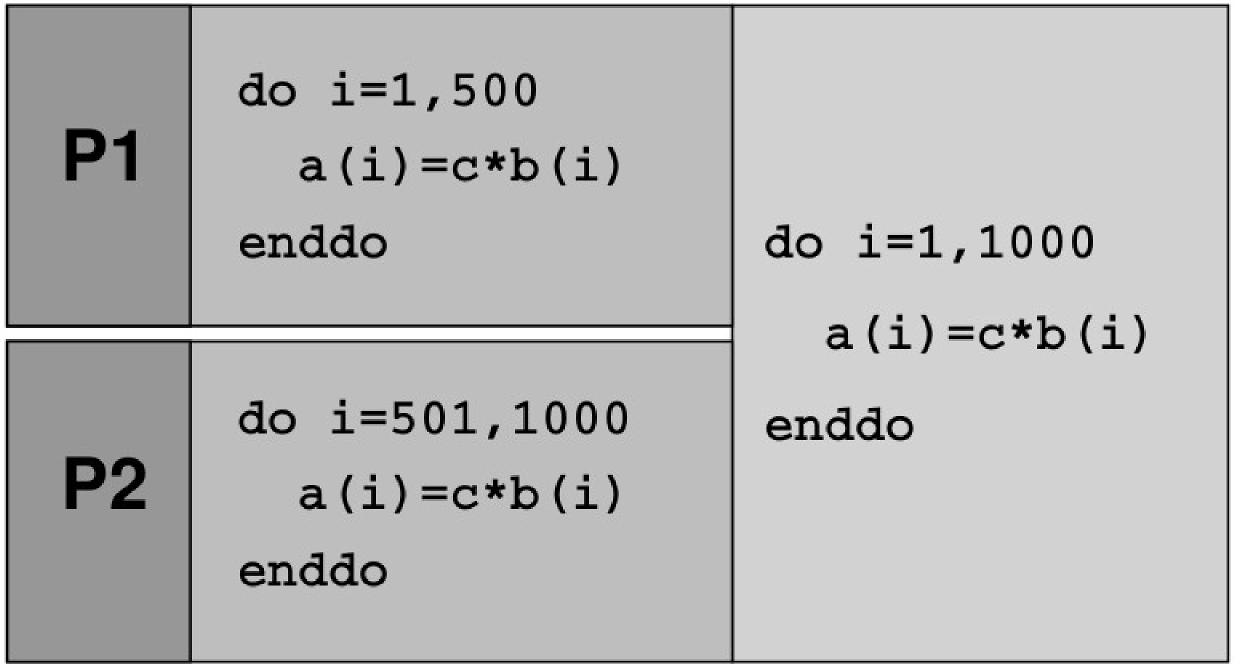
OpenMP & NUMA


I/O Bus
NUMA Node 0
Interconnect
NUMA Node 1
I/O Bus
System RAM




Thread 0 Stack
funcA() var1
var2
Stack Pointer
Program Counter
Registers
Thread 1 Stack
funcB() var1
var2
var3
Stack Pointer
Program Counter
Registers
text
main()
funcA()
funcB()
.....
data
array1
array2
heap
Process ID
User ID
Group ID
Files
Locks
Sockets

Example: Medium-grained loop parallelism Number of systems versus core count



```
do i=1,500
   a(i)=c*b(i)
enddo
```

```
do i=501,1000
   a(i)=c*b(i)
enddo
```

```
do i=1,1000
   a(i)=c*b(i)
enddo
```

Figure 5.1: An example for medium-grained parallelism: The iterations of a loop are distributed to two processors P1 and P2 (in shared memory) for concurrent execution.

5.2.2 Functional parallelism Example: Master-worker scheme

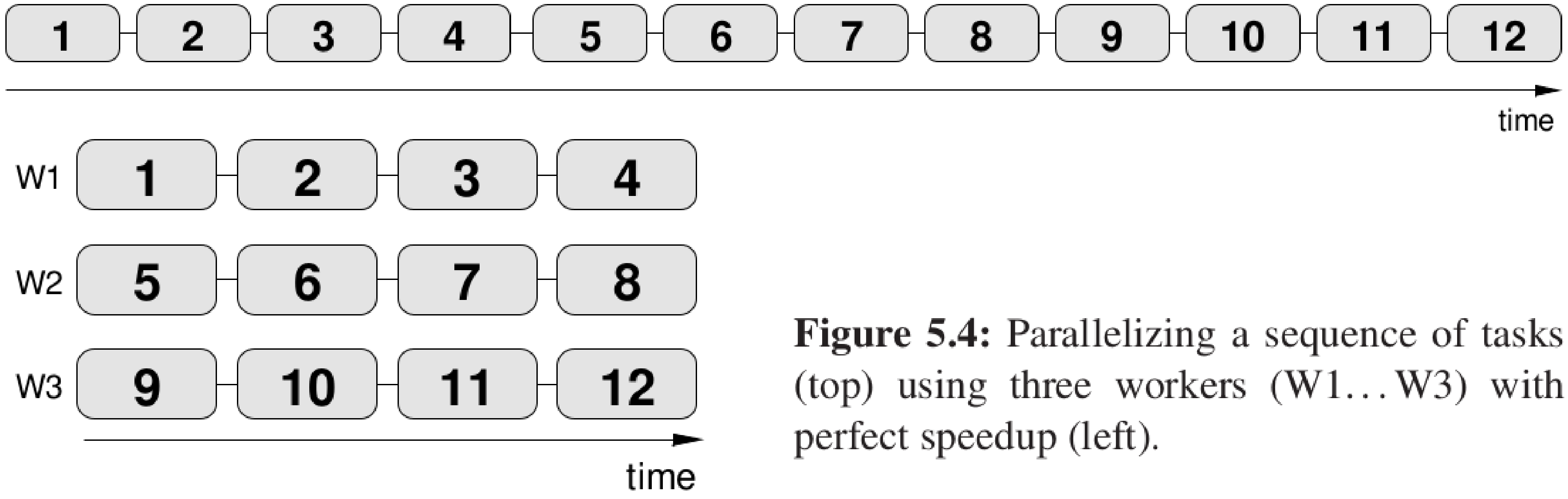


Figure 5.4: Parallelizing a sequence of tasks (top) using three workers (W1...W3) with perfect speedup (left).

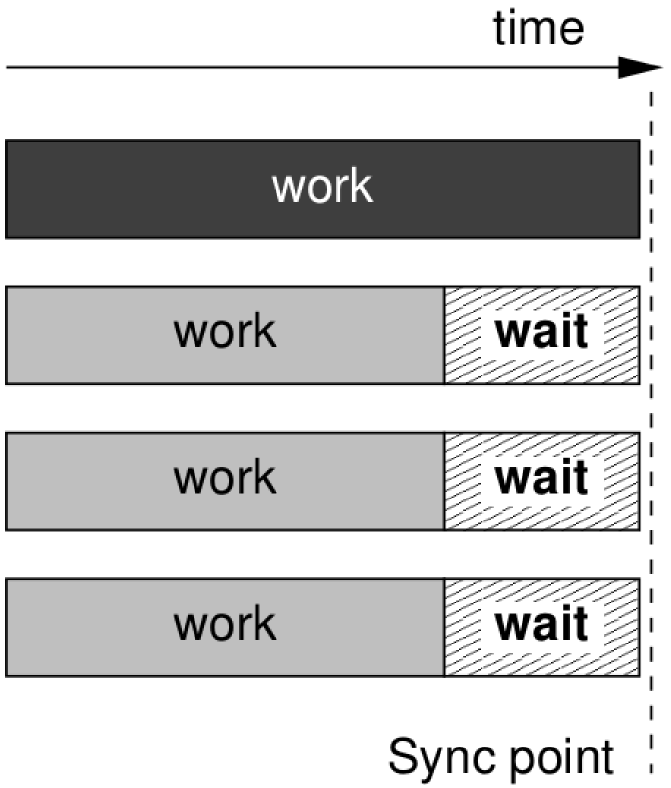


Figure 5.13: Load imbalance with few (one in this case) “laggers”: A lot of resources are underutilized (hatched areas).

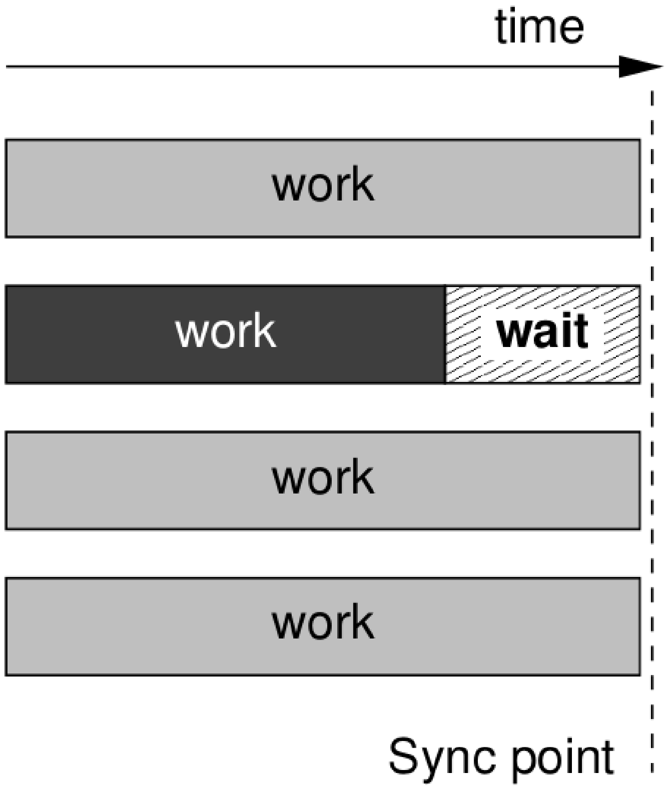
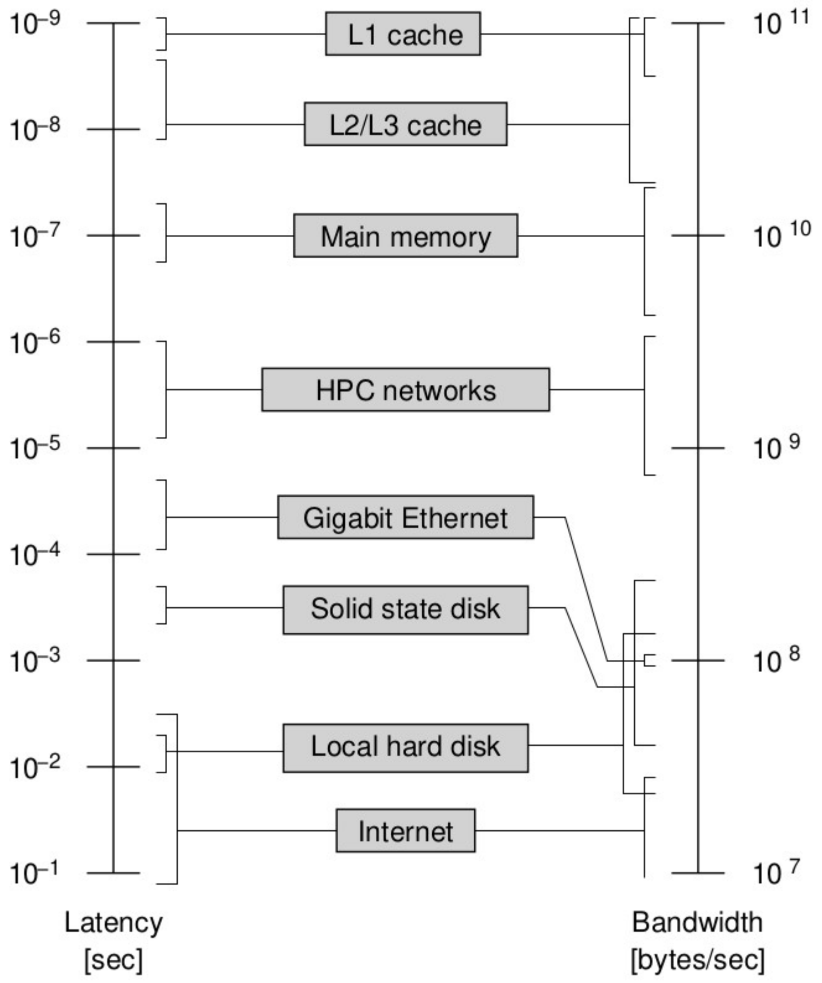


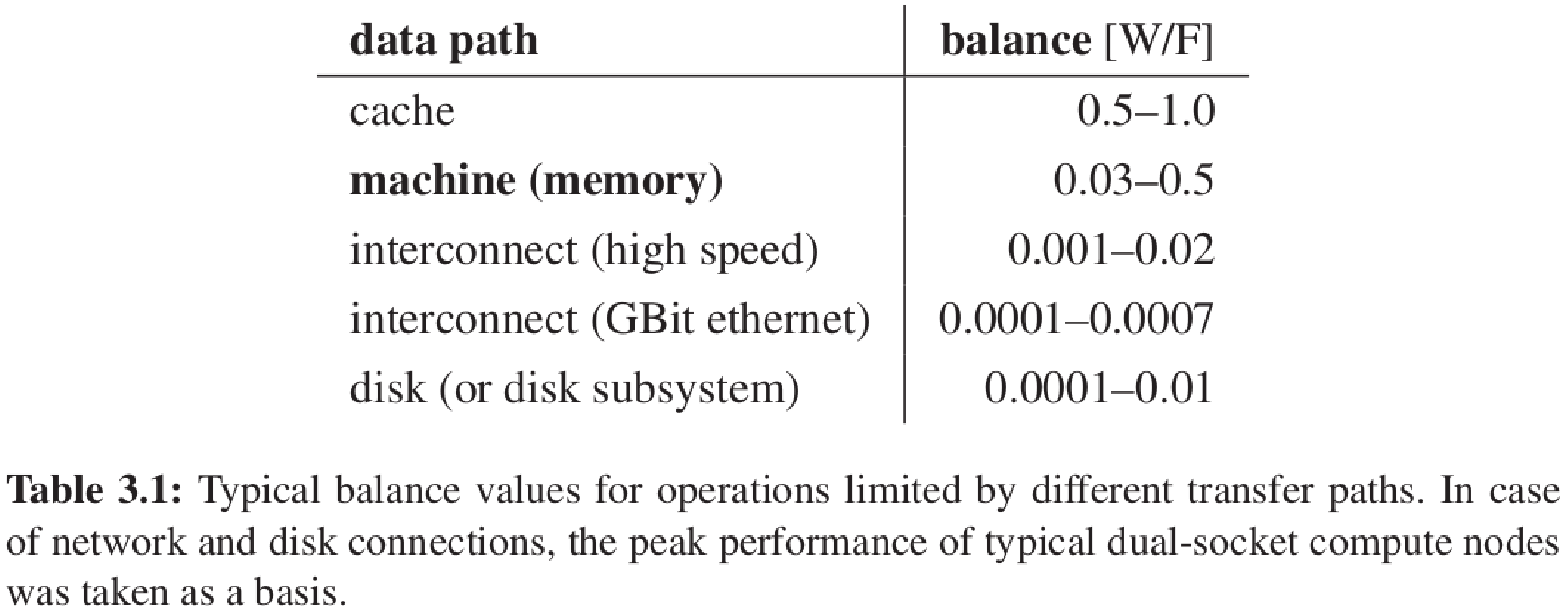
Figure 5.14: Load imbalance with few (one in this case) “speeders”: Underutilization may be acceptable.

Latency vs Bandwidth

$$B_m = \frac{\text{memory bandwidth [GWords/sec]}}{\text{peak performance [GFlops/sec]}} = \frac{b_{\max}}{P_{\max}}$$



Latency vs Bandwidth



data path	**balance** [W/F]
cache	0.5–1.0
machine (memory)	0.03–0.5
interconnect (high speed)	0.001–0.02
interconnect (GBit ethernet)	0.0001–0.0007
disk (or disk subsystem)	0.0001–0.01

Table 3.1: Typical balance values for operations limited by different transfer paths. In case of network and disk connections, the peak performance of typical dual-socket compute nodes was taken as a basis.

1 CPU core

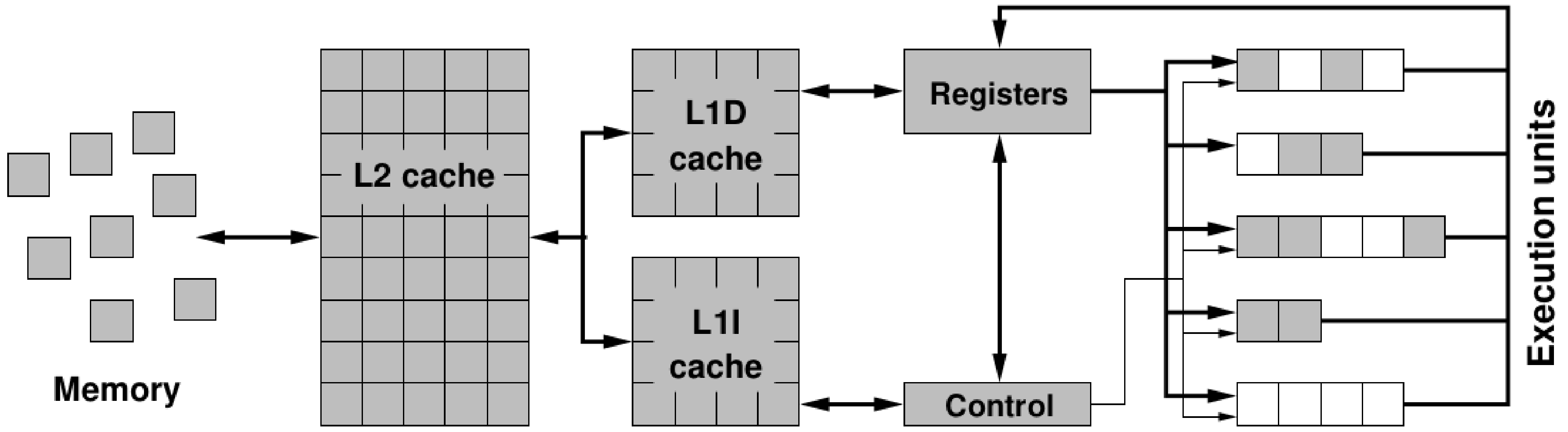
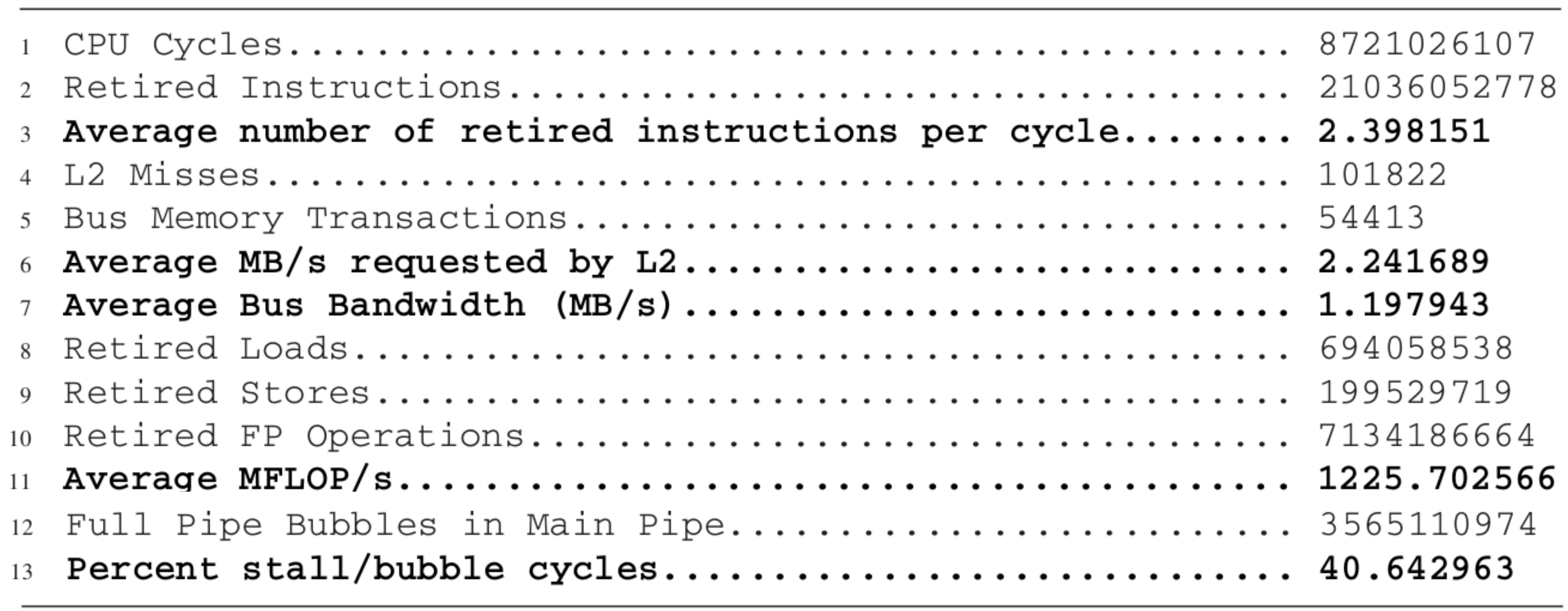


Figure 1.19: Simplified diagram of control/data flow in a (multi-)pipelined microprocessor without SMT. White boxes in the execution units denote pipeline bubbles (stall cycles). Graphics by courtesy of Intel.

Hardware performance counters Per Core



```
CPU Cycles.......................................... 8721026107
Retired Instructions................................ 21036052778
Average number of retired instructions per cycle........ 2.398151
L2 Misses........................................... 101822
Bus Memory Transactions............................. 54413
Average MB/s requested by L2........................ 2.241689
Average Bus Bandwidth (MB/s)........................ 1.197943
Retired Loads....................................... 694058538
Retired Stores...................................... 199529719
Retired FP Operations............................... 7134186664
Average MFLOP/s..................................... 1225.702566
Full Pipe Bubbles in Main Pipe...................... 3565110974
Percent stall/bubble cycles......................... 40.642963
```

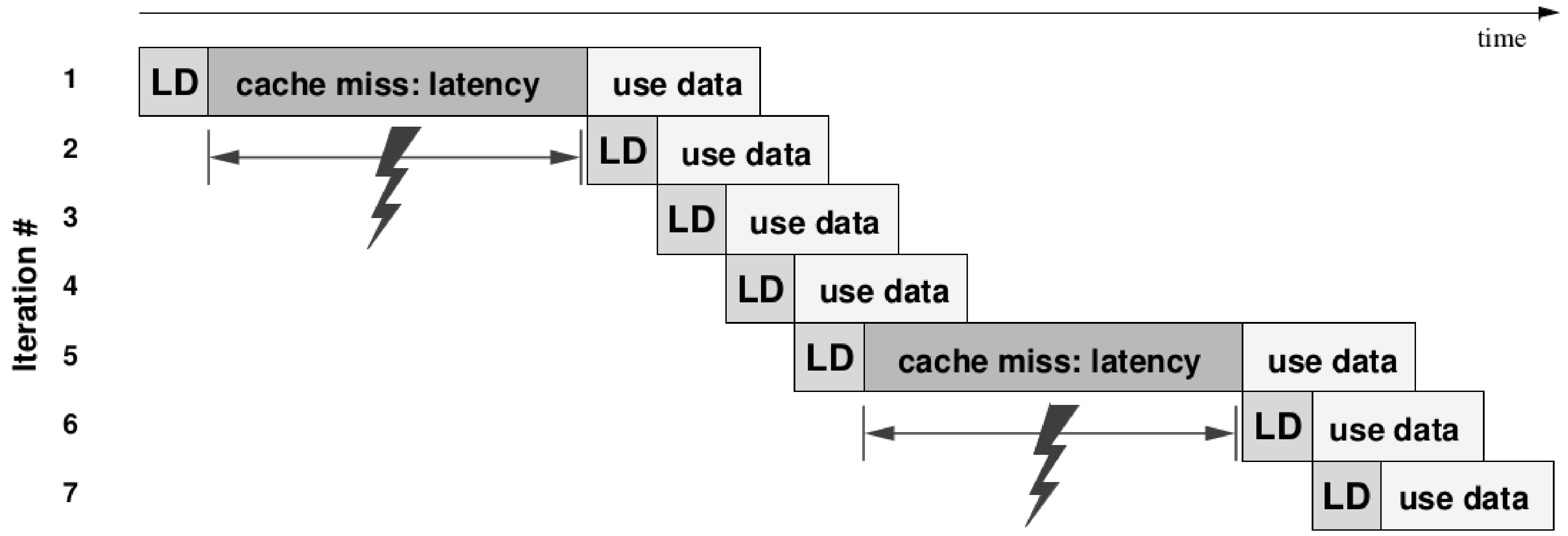


Figure 1.12: Timing diagram on the influence of cache misses and subsequent latency penalties for a vector norm loop. The penalty occurs on each new miss.

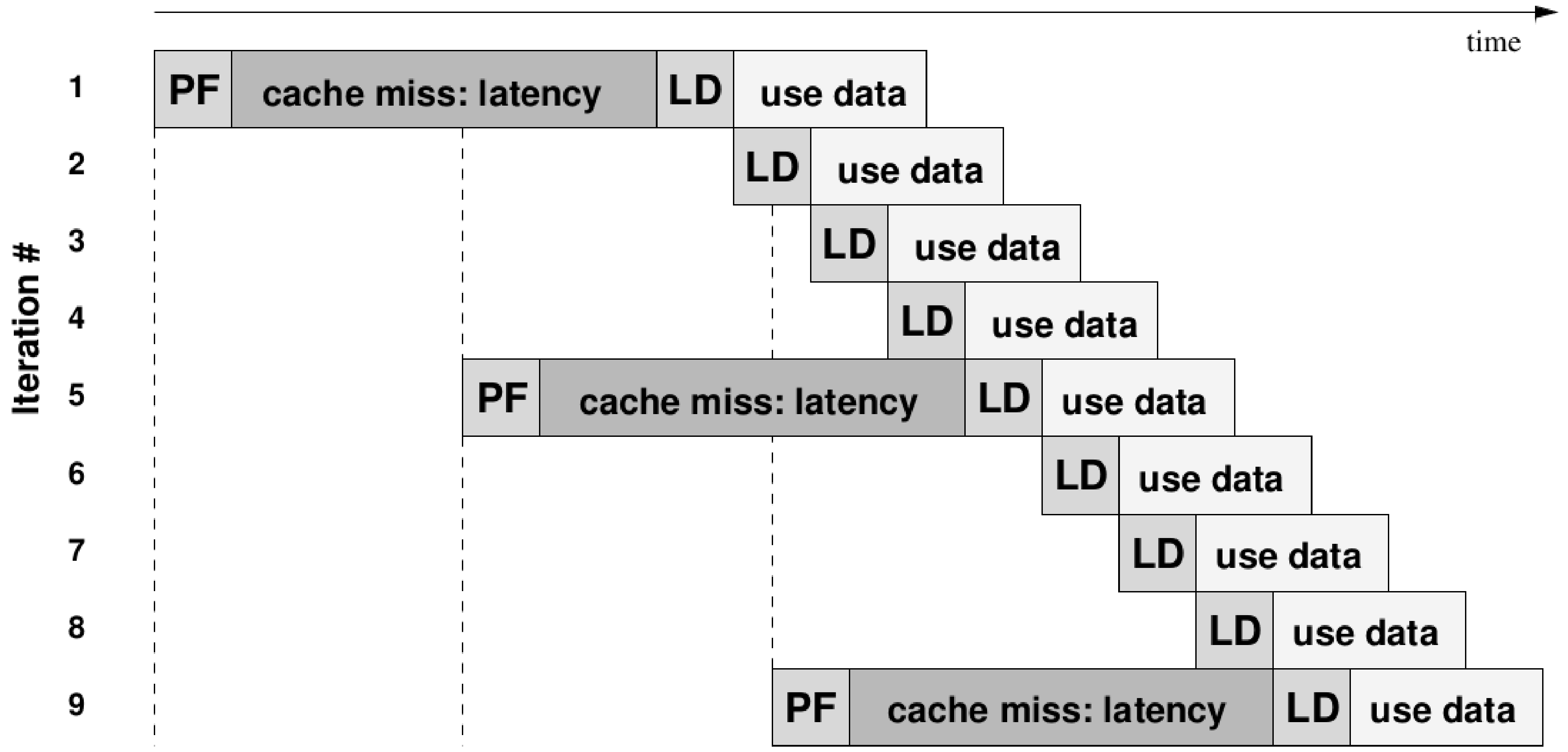


Figure 1.13: Computation and data transfer can be overlapped much better with prefetching. In this example, two outstanding prefetches are required to hide latency completely.

4.5 Networks

Point-to-point connections

Whatever the underlying hardware may be, the communication characteristics of a single point-to-point connection can usually be described by a simple model: Assuming that the total transfer time for a message of size N [bytes] is composed of latency and streaming parts,

$$T = T_\ell + \frac{N}{B} \tag{4.1}$$

and B being the maximum (asymptotic) network bandwidth in MBytes/sec, the effective bandwidth is

$$B_{\text{eff}} = \frac{N}{T_\ell + \frac{N}{B}} \ . \tag{4.2}$$

4.5 Networks

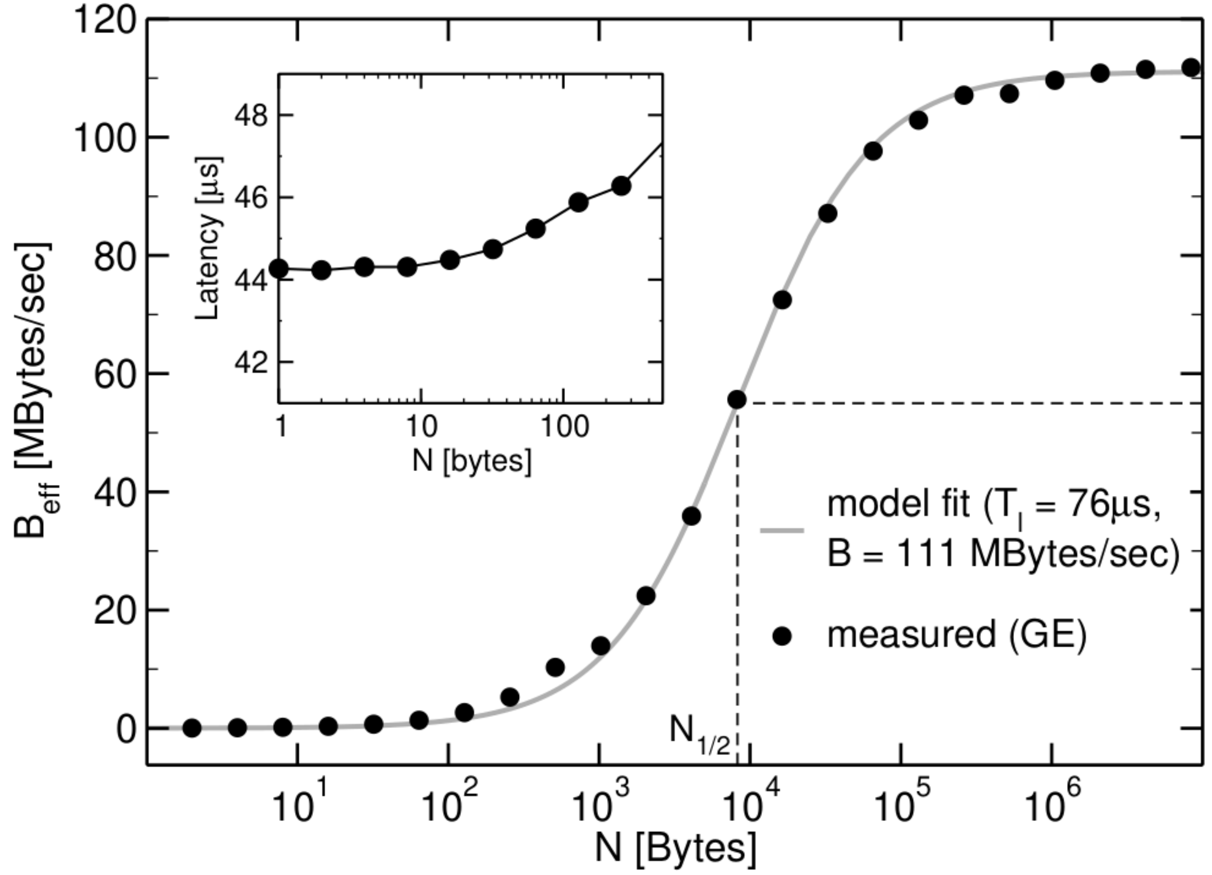


Figure 4.10: Fit of the model for effective bandwidth (4.2) to data measured on a GigE network. The fit cannot accurately reproduce the measured value of T_ℓ (see text). $N_{1/2}$ is the message length at which half of the saturation bandwidth is reached (dashed line).

4.5 Networks

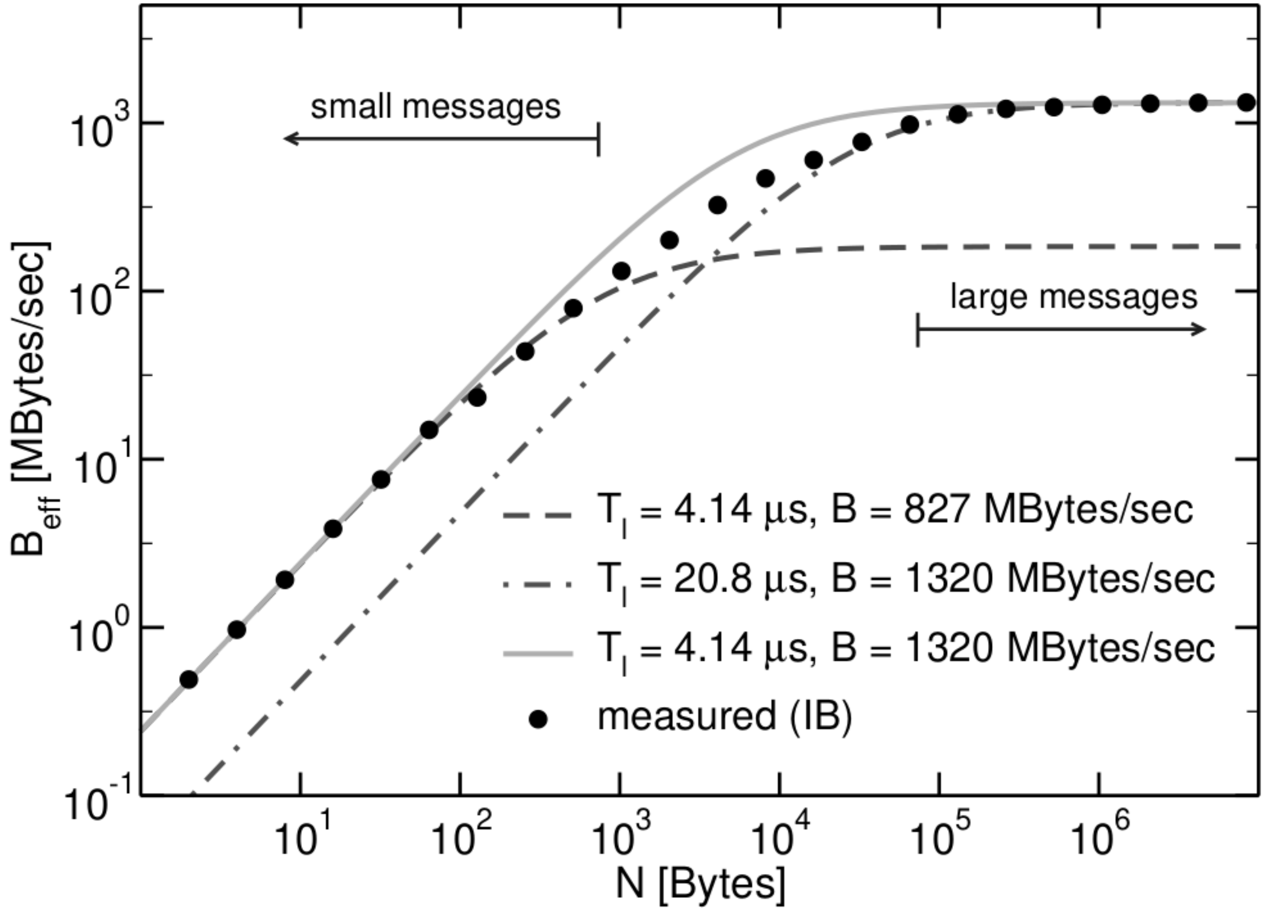


Figure 4.11: Fits of the model for effective bandwidth (4.2) to data measured on a DDR InfiniBand network. "Good" fits for asymptotic bandwidth (dotted-dashed) and latency (dashed) are shown separately, together with a fit function that unifies both (solid).

4.5 Networks

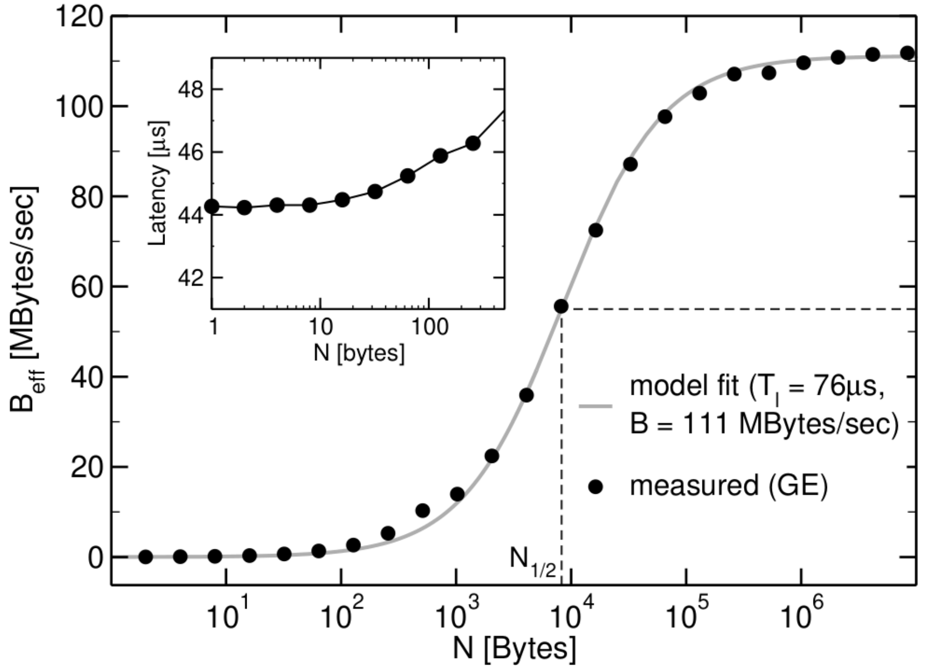


Figure 4.10: Fit of the model for effective bandwidth (4.2) to data measured on a Gig
work. The fit cannot accurately reproduce the measured value of T_ℓ (see text). $N_{1/2}$ i
message length at which half of the saturation bandwidth is reached (dashed line).

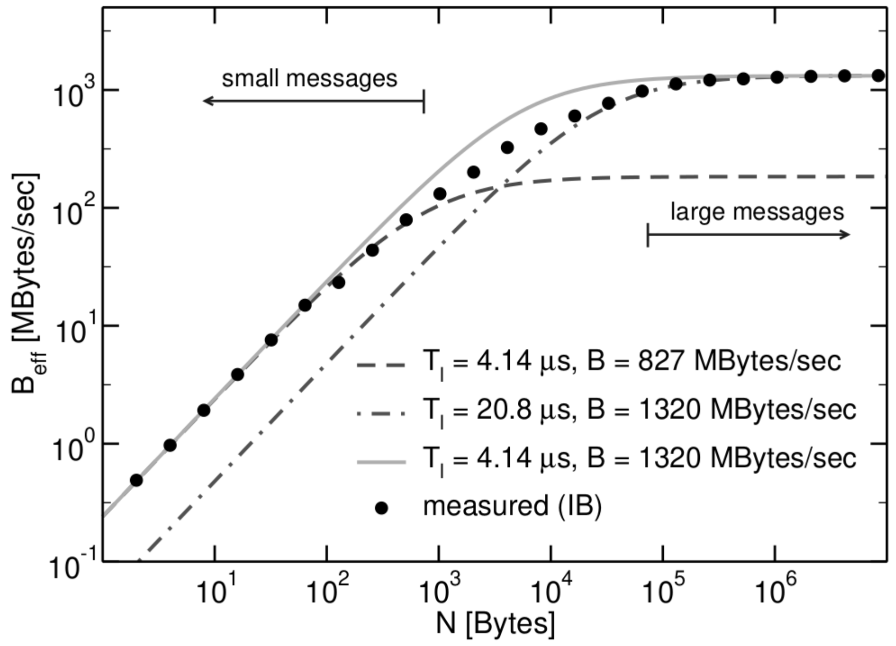


Figure 4.11: Fits of the model for effective bandwidth (4.2) to data measured on a DDR InfiniBand network. "Good" fits for asymptotic bandwidth (dotted-dashed) and latency (dashed) are shown separately, together with a fit function that unifies both (solid).