ASSIGNMENT 1 SOLUTIONS
HPC1 Fall 2013

Due Date: Thursday, September 12
(please submit your report electronically to the instructor via email in one file, PDF preferred, named *hw1-yourUBITname.pdf*)

Problem 1: Consider a simplified model for a two-level memory system (say cache and main memory),

$$\frac{\text{average cycles}}{\text{word access}} = f_c \times \frac{\text{cache cycles}}{\text{word access}} + (1 - f_c) \times \frac{\text{main memory cycles}}{\text{word access}},$$

where f_c is the fraction of cache hits out of all of the memory accesses. This simple model can be used in conjunction with the "machine balance," ρ_{WM}, of the ratio of floating point operations to memory (in words, typically 8 Bytes) accesses, to predict performance. Given a 2GHz processor (2×10^9 cycles/s), $\rho_{WM} = 2$, 2 cycles/word for cache and 100 cycles/word for main memory, what is the predicted performance for f_c of 99% and 1%?

Solution:

Given memory performance of 100 clock cycles per word access for main memory, and 2 cycles/word for the cache memory, $\rho_{WM} = 2$ Flop/word, and a 2GHz processor, we have the performance as

$$\begin{aligned}\text{performance} &= \frac{\rho_{\text{WM}} \cdot (\text{CPU cycles/s})}{[f_c \cdot (\text{cycles/cache word}) + (1 - f_c)(\text{cycles/main memory word})]},\\ &= 4 \times 10^9 \left[2f_c + 100(1 - f_c)\right]^{-1} \text{Flop/s},\end{aligned}$$

where f_c is the cache hit rate. For $f_c = 0.99$ the performance is 1342.3 MFlop/s, and for $f_c = 0.01$ it drops to only 0.0404 GFlop/s = 40.4 MFlop/s. Even in this simplified memory model, one can see the critical importance of making maximum reuse of data in the cache memory hierarchy.

Problem 2: Examine the hardware topology in the cluster at CCR. You can do this by using the **lstopo** command which is part of the OpenMPI **hwloc** project. Have a look at the man page ("**man lstopo**") and the **hwloc** project documentation to understand the details of the output from **lstopo** and what options you may need to use in combination with your slurm requests.

a) Run **lstopo** on the 8-core, 12-core, and both `INTEL` and `AMD` flavors of the 32-core nodes. Request the appropriate nodes through the batch queueing system (slurm). Record your slurm job number and graphical output from **lstopo** (note that you should be able to generate pdf and other graphical formats directly from lstopo).

b) Which node(s) have uniform shared memory, and which use non-uniform shared memory?

c) For each node type, identify the most likely sources for contention in terms of accessing memory from the processing cores.

Solution:

The plots below show the **lstopo** output (graphically) of the three flavors of compute nodes in the cluster. Note that **lstopo** can show you the hardware for the entire node even if you requested only a single core (see the **lstopo** man page), with the current version of **lstopo** that shows up as red in the topology diagram, meaning that you are not going to be able to use them.

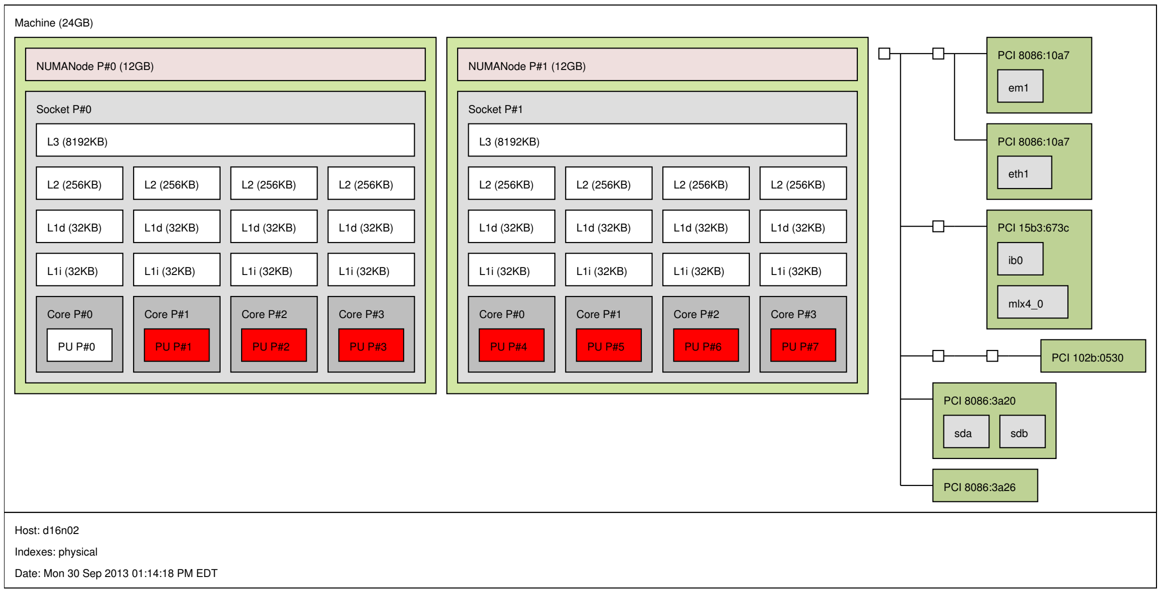


Figure 1: **lstopo** on a 8-core IBM iDataplex M2 node with "Nehalem" Intel Xeon L5520 processors. Non-uniform memory access (NUMA).

For core to memory contention, we have several things to take into account. First is the basic memory hardware itself, in terms of the number and speed of the memory controllers, which is outside the scope of what **lstopo** is showing us. The 8-core and 12-core nodes have 3 memory channels running at 1333MHz, while the 32-core Intel-based nodes have 4 channels at the same speed. For the 32-core Opteron systems we have 4 memory channels also running at 1333MHz. For the newest hardware, of course, there are a lot more cores sharing the improved memory hardware (recall what we noted in class regarding the relatively poor

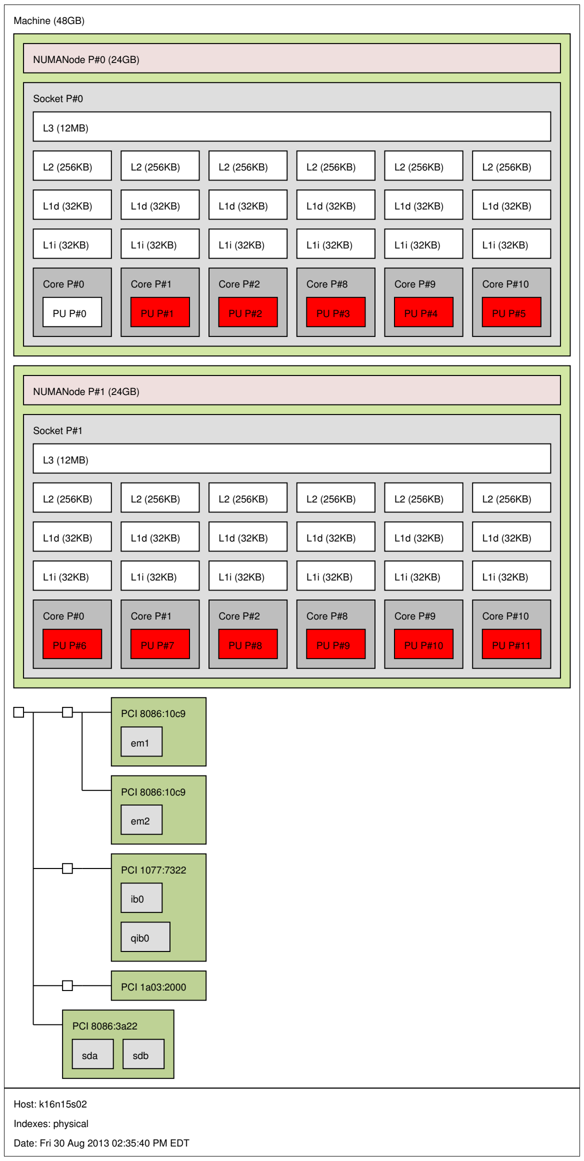


Figure 2: **lstopo** on a 12-core Dell C6100 node with "Westmere" Intel Xeon E5645 processors. Non-uniform memory access (NUMA).

improvements in low-level OpenMP benchmarks over the last decade or so). Second, we have the shared level 3 (L3) caches on the processors for all of the systems. This is a major bottleneck (one reason why larger cache sizes are a selling point for marketing processors), not only due to the size, but also due to *cache-coherency* issues (i.e., when a value is changed in one L3 cache, that change needs to be propagated to the others to maintain consistency in the memory or your program). Note that the network interfaces and other block devices (labeled by their PCIe address in **lstopo**) can also be sources of contention, but the original question was about core to memory.

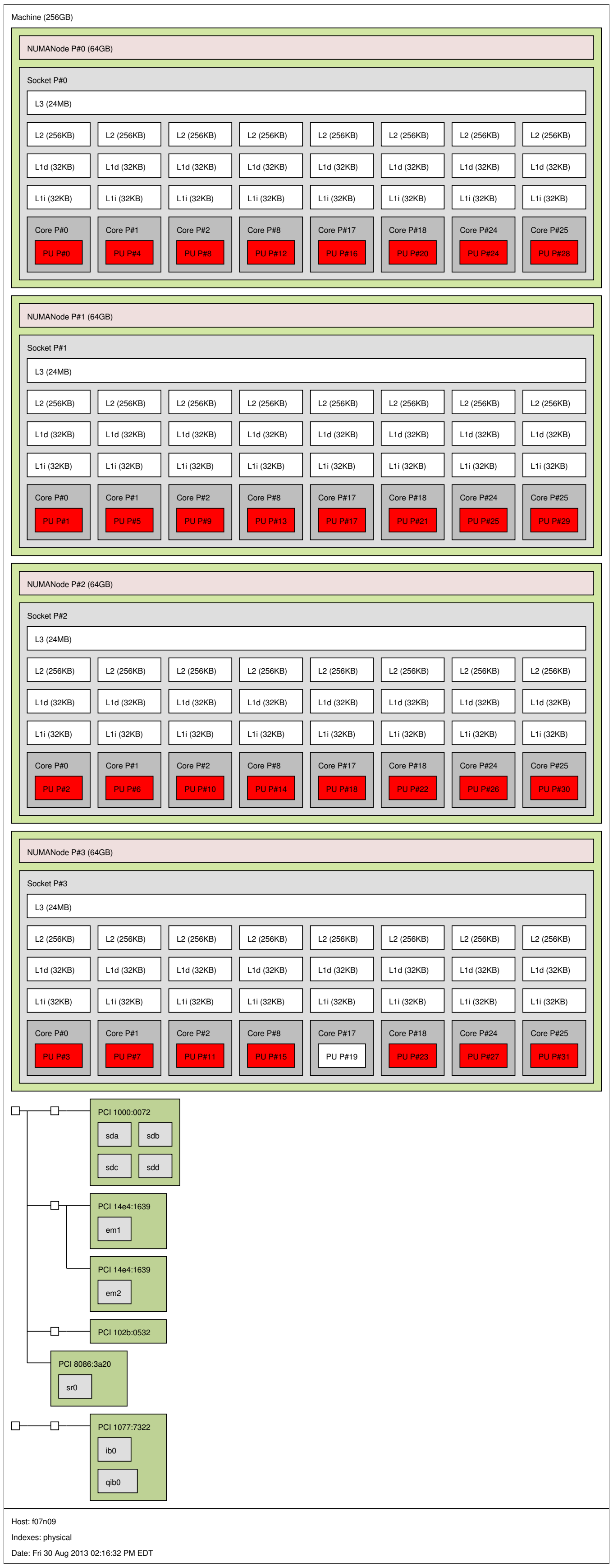
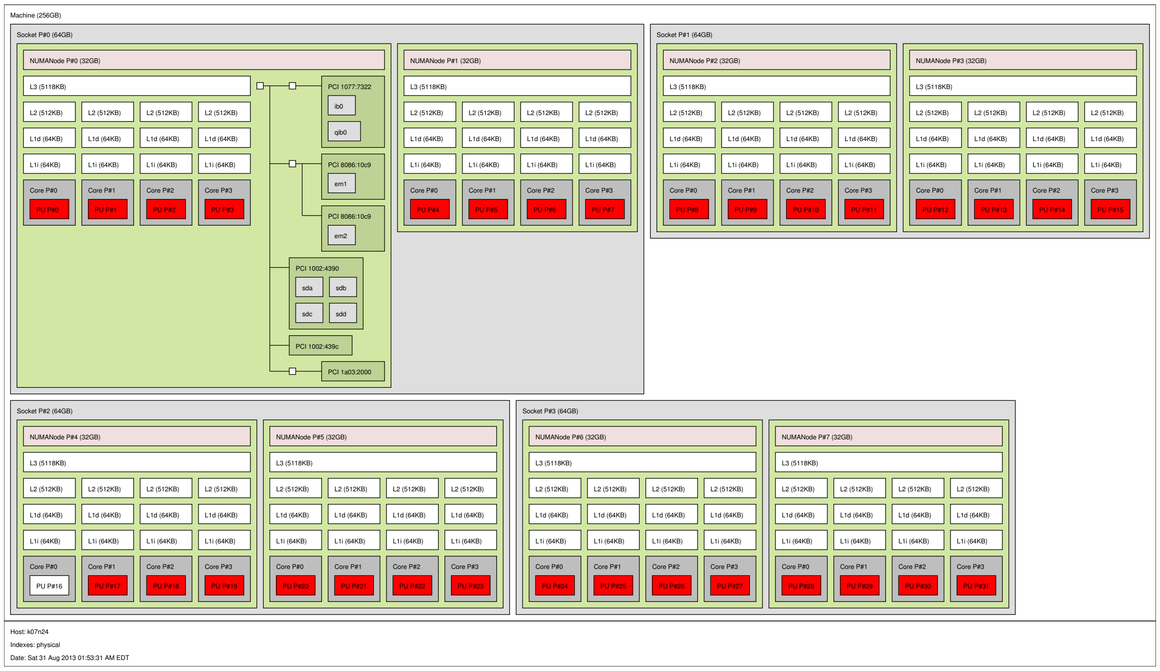


Figure 3: **lstopo** on a 32-core Dell R910 with “Nehalem-EX” Intel Xeon E7-4530 processors. Non-uniform memory access (NUMA).


Machine (256GB)
Socket P#0 (64GB)
NUMANode P#0 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#0 PU P#1 PU P#2 PU P#3
PCI 1077:7322
ib0
qib0
PCI 8086:10c9
em1
PCI 8086:10c9
em2
PCI 1002:4390
sda
sdb
sdc
sdd
PCI 1002:439c
PCI 1a03:2000
NUMANode P#1 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#4 PU P#5 PU P#6 PU P#7
Socket P#1 (64GB)
NUMANode P#2 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#8 PU P#9 PU P#10 PU P#11
NUMANode P#3 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#12 PU P#13 PU P#14 PU P#15
Socket P#2 (64GB)
NUMANode P#4 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#16 PU P#17 PU P#18 PU P#19
NUMANode P#5 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#20 PU P#21 PU P#22 PU P#23
Socket P#3 (64GB)
NUMANode P#6 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#24 PU P#25 PU P#26 PU P#27
NUMANode P#7 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 Core P#1 Core P#2 Core P#3
PU P#28 PU P#29 PU P#30 PU P#31

```
Host: k07n24
Indexes: physical
Date: Sat 31 Aug 2013 01:53:31 AM EDT
```

Figure 4: **lstopo** on a 32-core Dell C6145 with "Magny-Cours" AMD Opteron 6132HE processors. Non-uniform memory access (NUMA). Note that only 4 cores share each L3 cache and the "local" memory is 32GB.