Parallel Computing –Lab 3
Parallel Architectures

In this Lab you are going to explore parallel architectures.

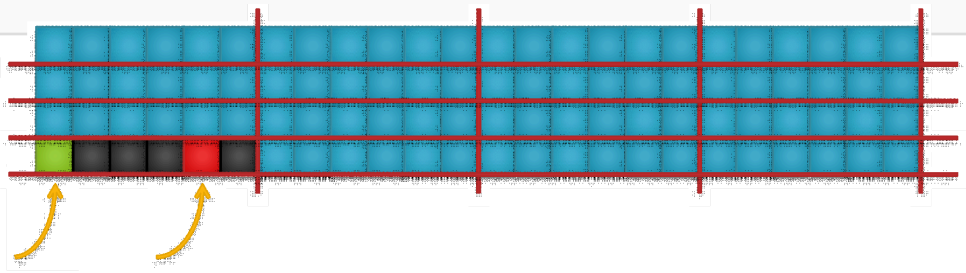
A. Cache coherency and false sharing (5 marks)

The simplest cache coherency protocol is **MESI**. Although cache coherency is important for making sure a program executes correctly, **it leads to false sharing**, which can cause performance problems. In order to remove false sharing, we have to either **move variables in memory** so that **they fit in different cache lines** or **Align shared global data to cache line boundaries**. For example, in the situation shown in this code:

```
struct foo {
    int x;
    int y;
};
int numThreads = 2;

struct foo z;
if (threadID ==0) {
    int sum1 = 0;
    int i;
    for (i = 0; i < 100000; ++i)
        sum1 += z.x;}

if (threadID ==1) {
    int sum2 = 0;
    int j;
    for (j = 0; j < 100000; ++j)
        sum2 += z.y;}
```

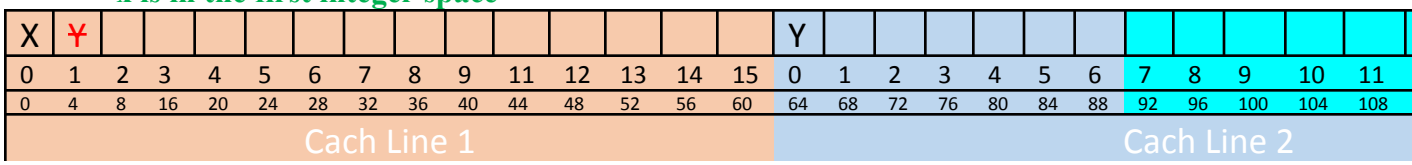


The **two elements of the structure** are **stored** in the **first two adjacent 4 bytes in the same cache line** (which means at **address 0 for x** and **address 4 for y**). If the **cache line size is 64 bytes**, by using the **padding technique** through adding an array of integers between the x and y **to enforce y to move to the next cache line.**

cache line size is 64 bytes = 64/4=16 integer space

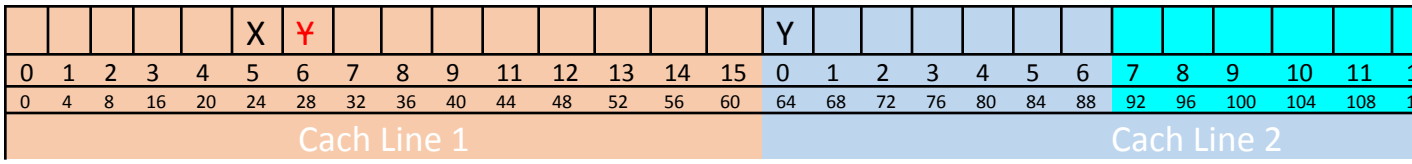
1. What is the minimum number of integer elements the padding array would have to be in order to move **y** to the next cache line?

   x is in the first integer space



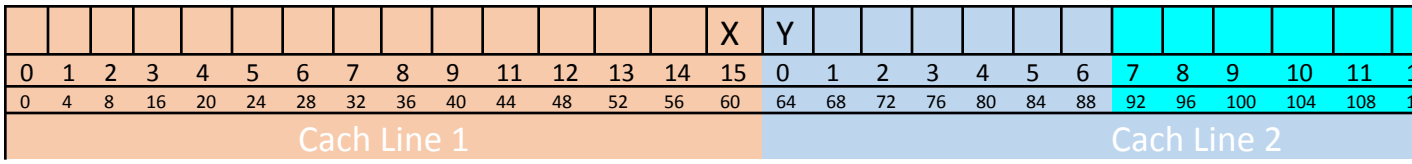
The minimum number of integer elements the padding array is 15

2. If **x** is at **memory address 24** and **y** is at **memory address 28**, what is the minimum number of integer elements the padding array would have to be in order to move y to the next cache line?



The minimum number of integer elements the padding array is 9

3. If **x is at memory address 60** and **y is at memory address 64**, what is the minimum number of integer elements the padding array would have to be in order to move y, so that x and y are in different cache lines?



ZERO THEY ARE IN DIFFERENT CACHE LINE

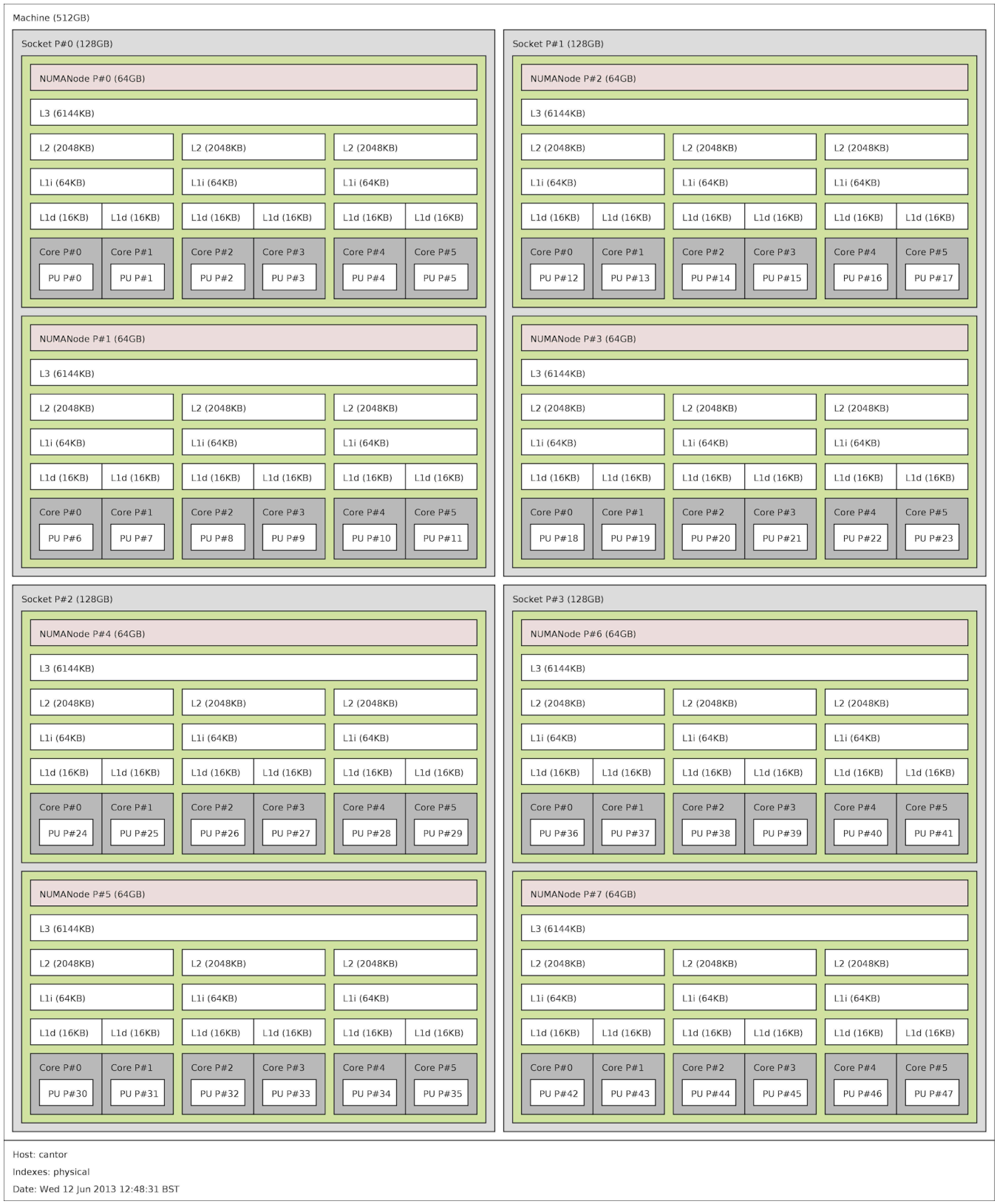
4. what would you suggest as an alternative possible solution the programmer can do to avoid false sharing?

Careful analysis of those parallel loops that play a major part in the execution of an application can reveal performance scalability problems caused by false sharing. In general, false sharing can be reduced by

- making use of private data as much as possible; utilizing the compiler's optimization features to eliminate memory loads and stores.

B. NUMA architecture

Consider the following organization of a NUMA machine:


Machine (512GB)
Socket P#0 (128GB)
NUMANode P#0 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#0
Core P#1
PU P#1
Core P#2
PU P#2
Core P#3
PU P#3
Core P#4
PU P#4
Core P#5
PU P#5
NUMANode P#1 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#6
Core P#1
PU P#7
Core P#2
PU P#8
Core P#3
PU P#9
Core P#4
PU P#10
Core P#5
PU P#11
Socket P#1 (128GB)
NUMANode P#2 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#12
Core P#1
PU P#13
Core P#2
PU P#14
Core P#3
PU P#15
Core P#4
PU P#16
Core P#5
PU P#17
NUMANode P#3 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#18
Core P#1
PU P#19
Core P#2
PU P#20
Core P#3
PU P#21
Core P#4
PU P#22
Core P#5
PU P#23
Socket P#2 (128GB)
NUMANode P#4 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#24
Core P#1
PU P#25
Core P#2
PU P#26
Core P#3
PU P#27
Core P#4
PU P#28
Core P#5
PU P#29
NUMANode P#5 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#30
Core P#1
PU P#31
Core P#2
PU P#32
Core P#3
PU P#33
Core P#4
PU P#34
Core P#5
PU P#35
Socket P#3 (128GB)
NUMANode P#6 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#36
Core P#1
PU P#37
Core P#2
PU P#38
Core P#3
PU P#39
Core P#4
PU P#40
Core P#5
PU P#41
NUMANode P#7 (64GB)
L3 (6144KB)
L2 (2048KB)
L2 (2048KB)
L2 (2048KB)
L1i (64KB)
L1i (64KB)
L1i (64KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
L1d (16KB)
Core P#0
PU P#42
Core P#1
PU P#43
Core P#2
PU P#44
Core P#3
PU P#45
Core P#4
PU P#46
Core P#5
PU P#47
Host: cantor
Indexes: physical
Date: Wed 12 Jun 2013 12:48:31 BST

a- Can you specify the following: [5 Marks]

1- How many NUMA nodes?

8

2- How many NUMA regions?

4

3- How many cores per node?

6

4- How many cores per region?

12

b- Within each NUMA node, can you specify:

1- How many cache levels?

4

2. Which levels of cache are shared? and between which cores?

L1i , L2 is shared between two adjacent cores

L3 is shared between all six cores in the same node

3. Which levels of cache are private?

L1d is private