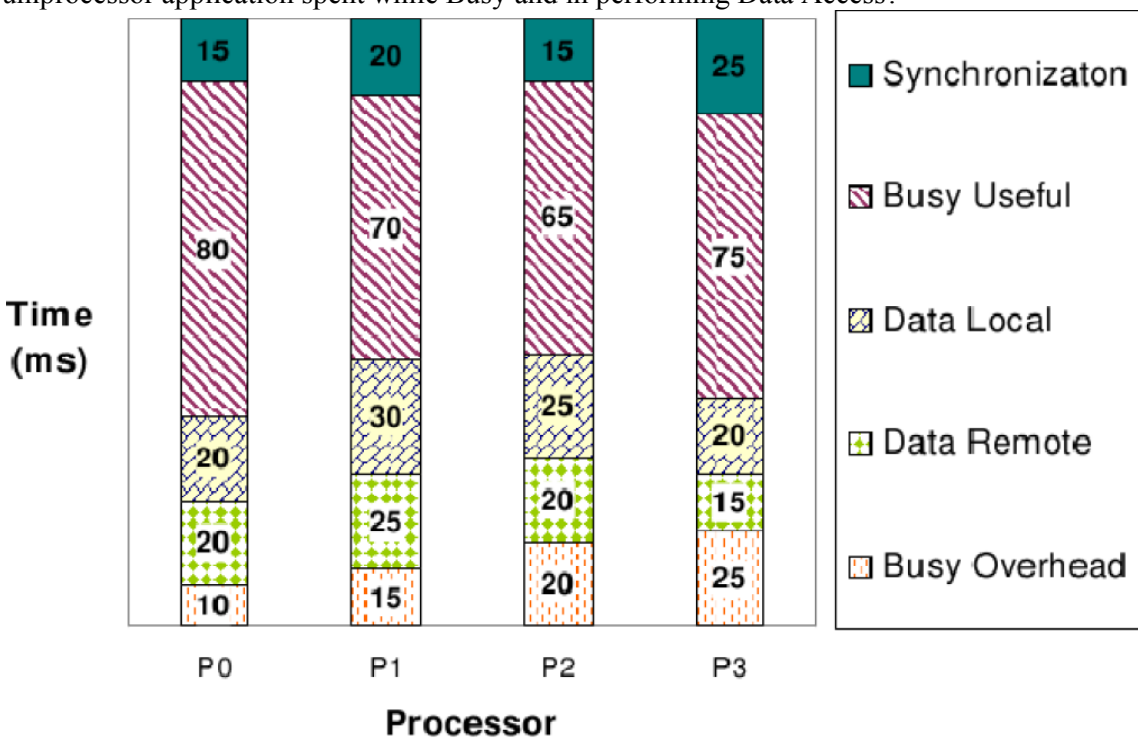
1. A uniprocessor application is parallelized for 4 processors, yielding a 3.8x speedup. Given the time breakdown of the various function seen in the graph, what is the minimum total time that the uniprocessor application spent while Busy and in performing Data Access?



Answer2:

$$\text{Speed Up}(p) <= \frac{SequentialWork}{\text{Max Work on Any Processor}}$$

Looking at each column of the graph and adding the time concerning synchronization, busy useful time, data local time, data remote and busy overhead, you can fund the required execution time of each parallel processor. Afterward, you have to put the longest execution time to the following formula:

$$3.8x < \frac{SequentialWork}{\text{Max Work on Any Processor}} \Rightarrow \quad 3.8x <= \frac{SequentialWork}{\text{Max}(145, 160, 145, 160)}$$

Therefore SequentailWork>= 3.8x * 160

3- Consider a bus-based shared memory multiprocessor system. It is constructed using processors with speed of 10^6 instructions, and a bus with a peak bandwidth of 10^5 fetches. The caches are designed to support a hit rate of 90%.
(a) What is the maximum number of processors that can be supported by this system?
(b) What hit rate is needed to support a 20-processor system?

In this question, (1-hit rate)* processor speed actually determines the rate of bus request form each processor. Hence, the bus will be saturated when total bus requests from the N processors exceed the bus peak bandwidth the following inequality illustrates that

N(1-Hit Rate) * Processor Speed<= Bus Band Width
$N(1-0.9)* 10^6 \leq 10^5$

Therefore N must be 1 for the first part of this question. For the second part of this question we are going to have 20 processors; therefore, we simply place 20 instead of N and find the required hit rate

$20(1-h)* 10^6 \leq 10^5$
$(1-h) \leq 1/200$
$1-1/200 \leq$ hit rate

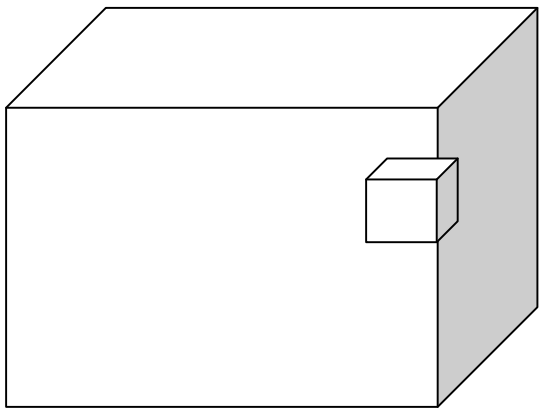
You are given a simple 3D Array of integer which consist of n^3 elements and p processors as a system specification. Using the Domain Decomposition technique, you can be able to varyingly parallelize operations of P processors on this 3D array. What are two different ways of decomposition and the concerning computation and computation overhead of each way.

Answer 7:

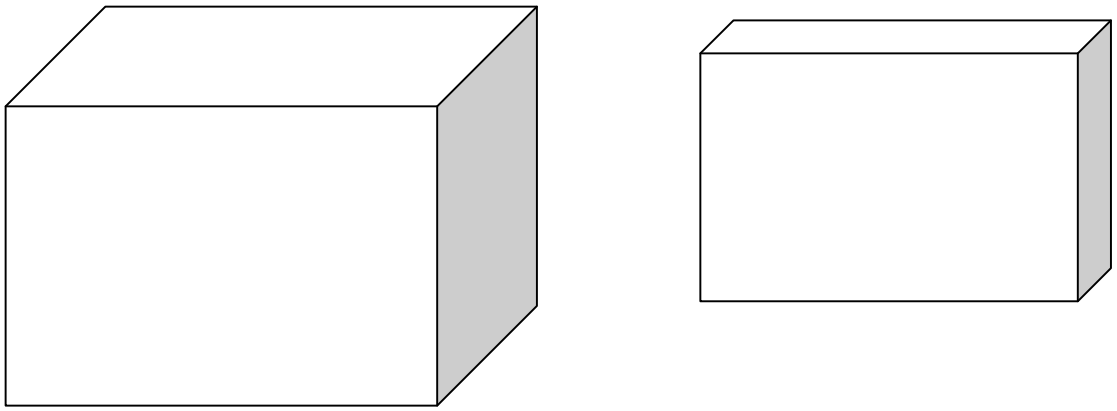
One way of domain decompositions is using blocking,

Communication/computation= LATERAL Area/Volume of each block

$$\text{Communication/computation}=\frac{\left(\frac{n}{\sqrt[3]{p}}\right)^2 \times 6}{\left(\frac{n}{\sqrt[3]{p}}\right)^3}=\frac{6\times p^{\frac{1}{3}}}{n}=\frac{6\times \sqrt[3]{p}}{n}$$



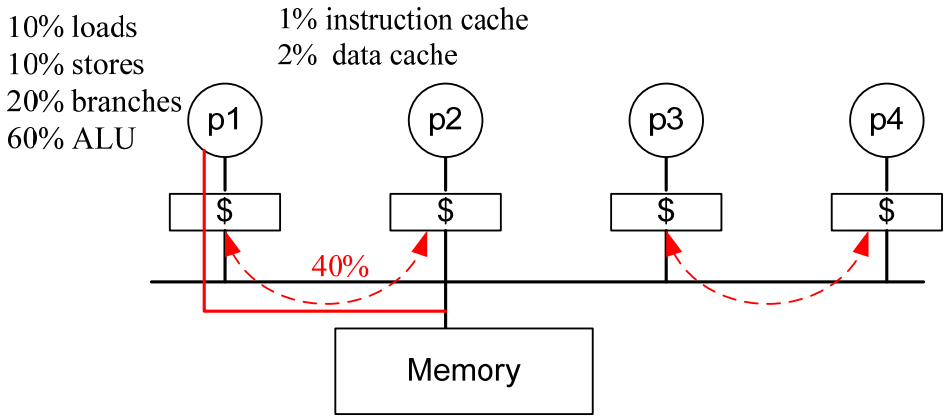
Another way of domain decomposition is using of strip box



Communication/computation= LATERAL Area/Volume of each block

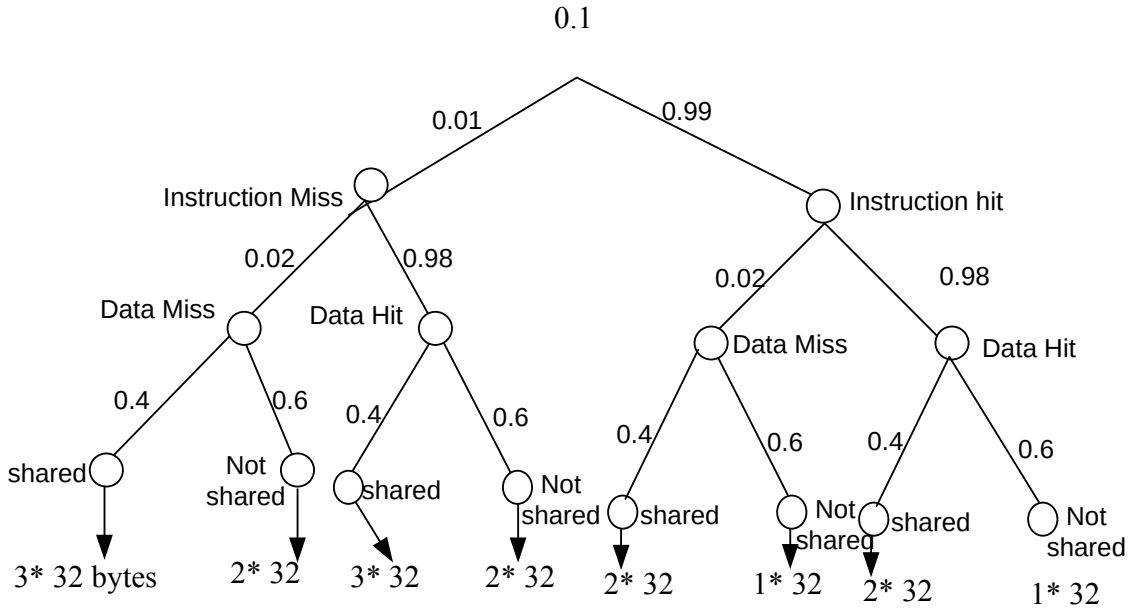
$$\text{Communication/computation}=\frac{n^2\times 2}{\frac{n^3}{p}}=\frac{2\times p}{n}$$

5-Consider a bus-based machine with 4 processors, each running at a 0.5 GIPS and running a workload that consists of: 60% ALU operations, 10% loads, 10% stores and 20% branches. Suppose that the cache miss rate at each processor is 1% for instruction cache and 2% for data cache, and the cache sharing among 2 processors is 40% and zero otherwise. The system bus bandwidth is 8GB/s. Assuming that the cache line is 32 bytes large, and a snooping protocol, determine the bandwidth used. How many processors could the bus accommodate?



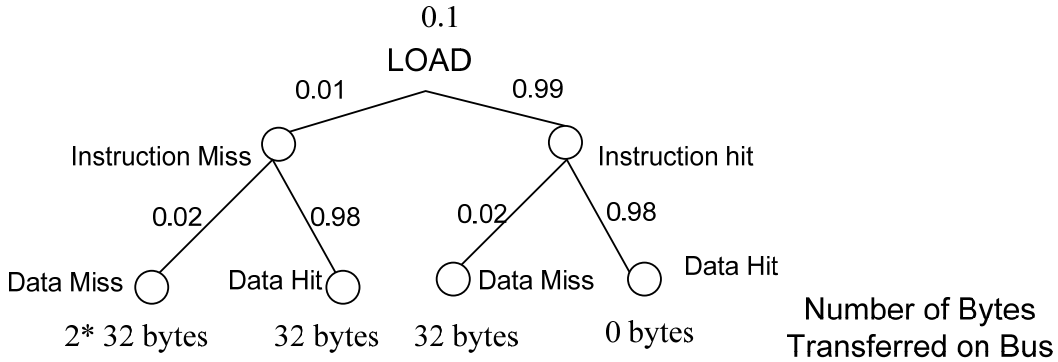
We assume the simple write-through invalidate snooping protocol where every store operation places the address of the block on the bus and cache controllers snoop on the bus and invalidate their block if it was originally in their cache.

In the write-through invalidate snooping snooping protocol, **stores generate a bus transaction in which multiple caches might request to invalidate their blocks. The problem statement states that this happens 40% of the time and only 1 other processor cache will need to get invalidated.**



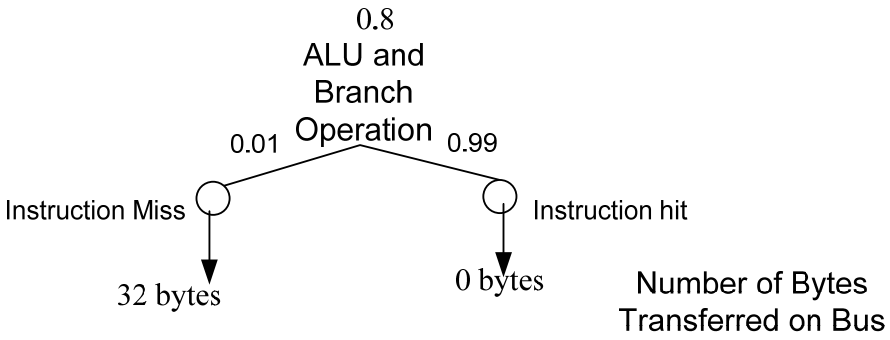
: 0.1*(0.01(0.02*(0.40*3*32bytes+0.60*2*32bytes)+0.98*(0.40*3*32bytes+0.60*2*32bytes))+
0.99(0.02*(0.40*2*32bytes+0.60*1*32bytes)+0.98*(0.40*2*32bytes+0.60*1*32bytes)) = 4.512

Loads might generate a cache misses 2% of the time.



LOAD :
0.1*(0.01(0.02*(2*32bytes)+0.98*(32bytes)) +0.99(0.02*(32bytes)+0.98*(0bytes)))

ALU and Braches do not generate any data misses (all operands that they are working with are inside registers) but can still generate instruction misses like all instructions (1% of the time).



0.8*(0.01*32bytes+0.99*0byte)

To calculate the total bandwidth used, we need to calculate **the number of bytes per instruction transferred on every instruction:**

LOAD:
0.1*(0.01(0.02*(2*32bytes)+0.98*(32bytes)) +0.99(0.02*(32bytes)+0.98*(0bytes)))

SOTRE:

: 0.1*(0.01(0.02*(0.40*3*32bytes+0.60*2*32bytes)+0.98*(0.40*3*32bytes+0.60*2*32bytes))+
0.99(0.02*(0.40*2*32bytes+0.60*1*32bytes)+0.98*(0.40*2*32bytes+0.60*1*32bytes))

ALU, BRANCH:
0.8*(0.01*32bytes+0.99*0byte)

number of bytes per instruction transferred=0.1*(0.96) +0.1*(45.2)+0.8*(0.32) = 4.872 bytes

Processor bandwidth = 0.5 $\frac{GigaInstruction}{s}$ *4.872 $\frac{Bytes}{Instruction}$ = 2.436 $\frac{GB}{s}$

The Bus could accommodate $\frac{8\frac{GB}{s}}{2.436\frac{GB}{s}} = 3.284 \cong 3$

Consider transposing a matrix in parallel from a source matrix to a destination matrix.

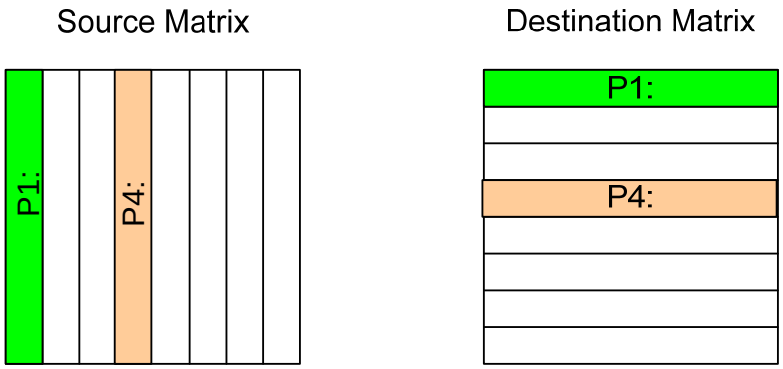
a- How might you partition the two matrices among processes? Discuss some possibilities and the trade-off. Does it matter whether you are programming a shared address space or message-passing machine?

b- Why is interposes communication in a matrix transpose called all-to-all personalized communication?

c- Write simple pseudo code for the parallel matrix transposition in a shared address space and in message passing (just the loops that implement the transpose).

Row and Column based Decomposition

Memory over head: Transpose can not take place



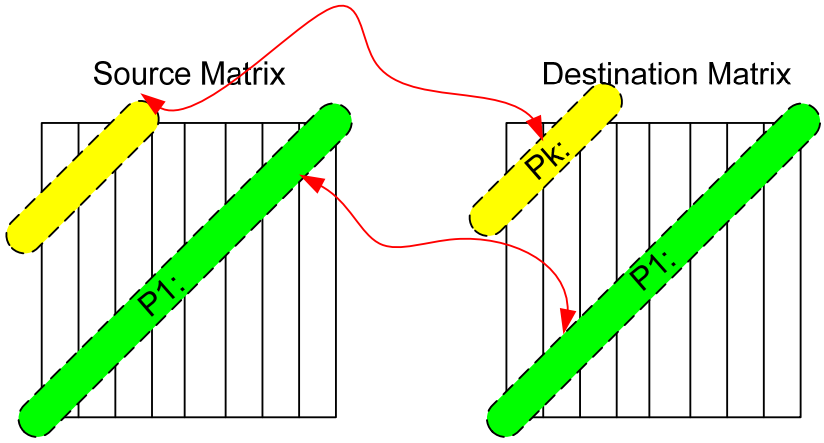
No communication will be needed between processes.

Data Independence

Because of data independency there are no differences in shared memory and message passing approach.

It can be done in N step.

Memory overheard



Diagonal Based Decomposition

No communication will be needed between processes.

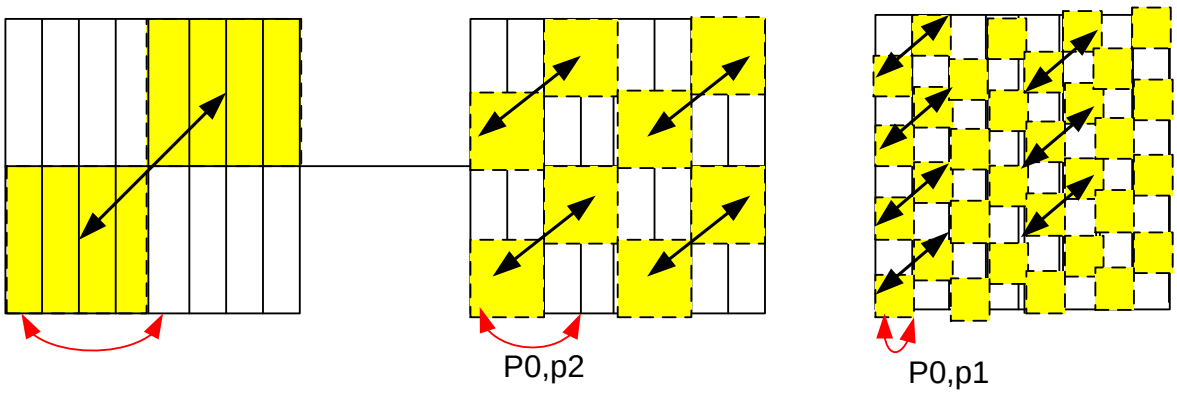
Data Independence

Because of data independency there are no differences in shared memory and message passing approach.

It can be done in $\sqrt{2}$ N step.

Memory overheard

Hypercube method



communication between processes. All to All communication as you can see column 0 gets value by communicating with 4,2 and 1 instead of communication with 7 other column.

It can be done in $\sqrt{N}$ step.

No Memory overheard (In place transpose)

Shared address space code

For k= 0 … log(n) -1

Begin