Homework 4

Questions 1: Consider the MSI protocol for a bus-based system with three processors P1, P2 and P3, each with a direct-mapped cache and assume that the size of a cache line is one word. The following sequence of memory operations access two memory locations (words), A and B, that are mapped to the same cache location.

P_1 writes A=4
P_3 writes B=8
P_2 reads A
P_3 reads A
P_3 writes A= 12
P_2 reads A
P_1 reads B
P_1 writes B = 10

Assume that initially, A = B = 3 and that the cache is initially empty. For each memory access, determine

(a) The content and state of the cache line in each processor
(b) The bus operations caused by the MSI protocol (read request, read data, write back)

Answer:

First, let's clarify the semantics of the bus operations:

1) In a read request (RR) operation, a core puts the address of a cache line on the address lines of the bus and puts a "read" command on the control lines of the bus. When a core requests a cache line, it may request it in the "Shared" or the "Modified" state. This is why we will distinguish between RRS and RRM.
2) In a read data operation (D), the memory puts the data that was to be read on the bus.
3) In a write back operation (WB), the core puts the data and its address on the data and address lines of the bus, along with a "write" command. The memory will then perform the write. Other cores will also monitor WB operations to acquire the data (if needed) or perform invalidation (if needed).

In the following table, S, M and I will be used to indicate the state of the cache line (shared, modified and invalid, respectively).

Event	P1	P2	P3	Bus
Initial State	I	I	I	-
P_1 writes A=4	A=4(M)	I	I	RRM(A), D(A)
P_3 writes B=8	A=4(M)	I	B=8 (M)	RRM(B), D(B)
P_2 reads A	A=4(S)	A=4(S)	B=8 (M)	RRS(A) , WB(A)
P_3 reads A	A=4(S)	A=4(S)	A=4(S)	WB(B), RRS(A), D(A)
P_3 writes A= 12	I	I	A=12(M)	RRM(A)
P_2 reads A	I	A=12(S)	A=12(S)	RRS(A), WB(A)
P_1 reads B	B=8 (S)	A=12(S)	A=12(S)	RRS(B), D(B)
P_1 writes B = 10	B=10 (M)	A=12(S)	A=12(S)	RRM(B)

Question 2: Repeat Question 1 assuming that the size of each cache line is two words and that both A and B are contained in the same cache line.

Answer: In this answer, we are dealing with only one cache line, so we do not have to specify the cache line when describing the bus transaction.

Event	P1	P2	P3	Bus
Initial State	I	I	I	-
P_1 writes A=4	A=4, B=3(M)	I	I	RRM, D
P_3 writes B=8	I	I	A=4, B=8(M)	RRM, WB
P_2 reads A	I	A=4, B=8(S)	A=4, B=8(S)	RRS, WB
P_3 reads A	I	A=4, B=8(S)	A=4, B=8(S)	
P_3 writes A= 12	I	I	A=12, B=8(M)	RRM
P_2 reads A	I	A=12, B=8(S)	A=12, B=8(S)	RRS, WB
P_1 reads B	A=12, B=8(S)	A=12, B=8(S)	A=12, B=8(S)	RRS, D
P_1 writes B = 10	A=12, B=10(M)	I	I	RRM

Question 3: Do Problem 2.16 (a and b) from the textbook

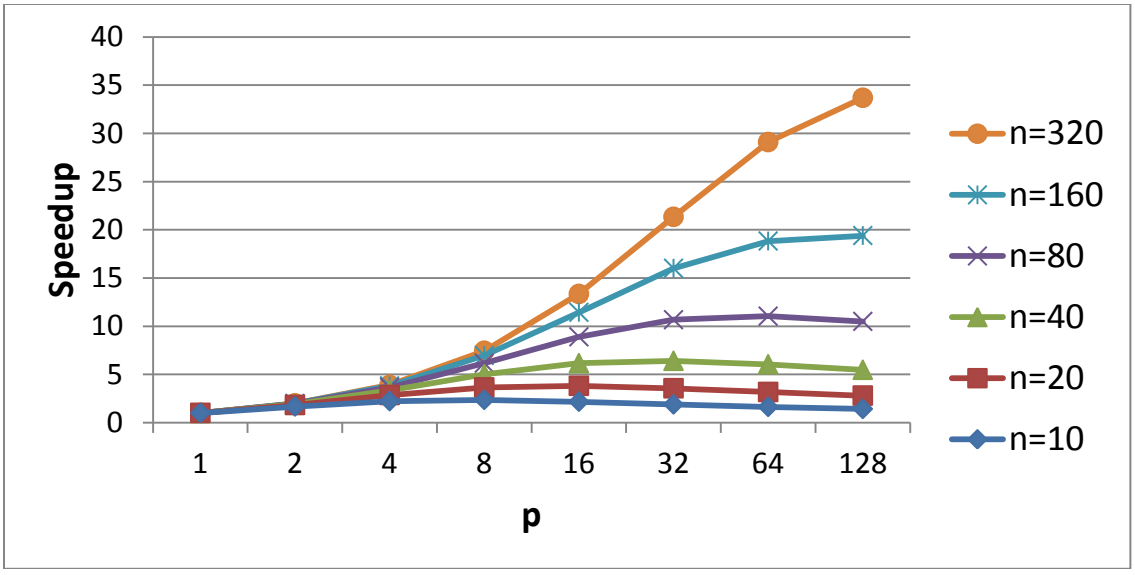
Answer:

(a) $Ts = n^2$

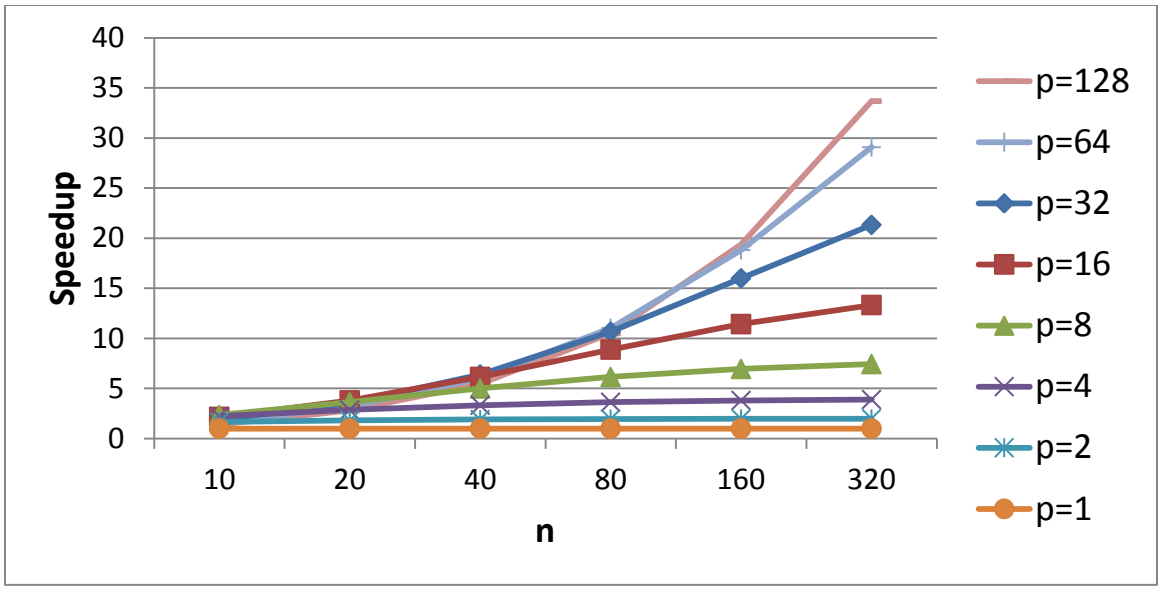
$T_p = n^2/p + \log_2(p)$

$S_p = T_s / T_p = n^2 / (n^2/p + \log_2(p)) = 1 / (1/p + \log_2(p)/ n^2)$

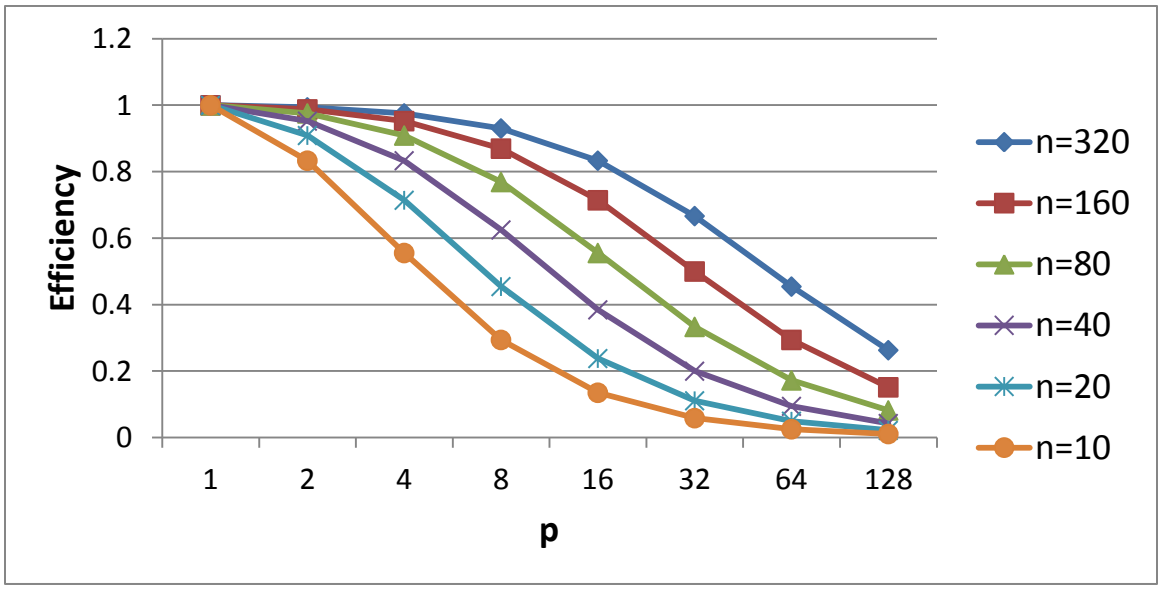
$E_p = S_p / p = 1 / ((1/p + \log_2(p)/ n^2) * p) = 1/ (1 + p \log_2(p)/ n^2)$



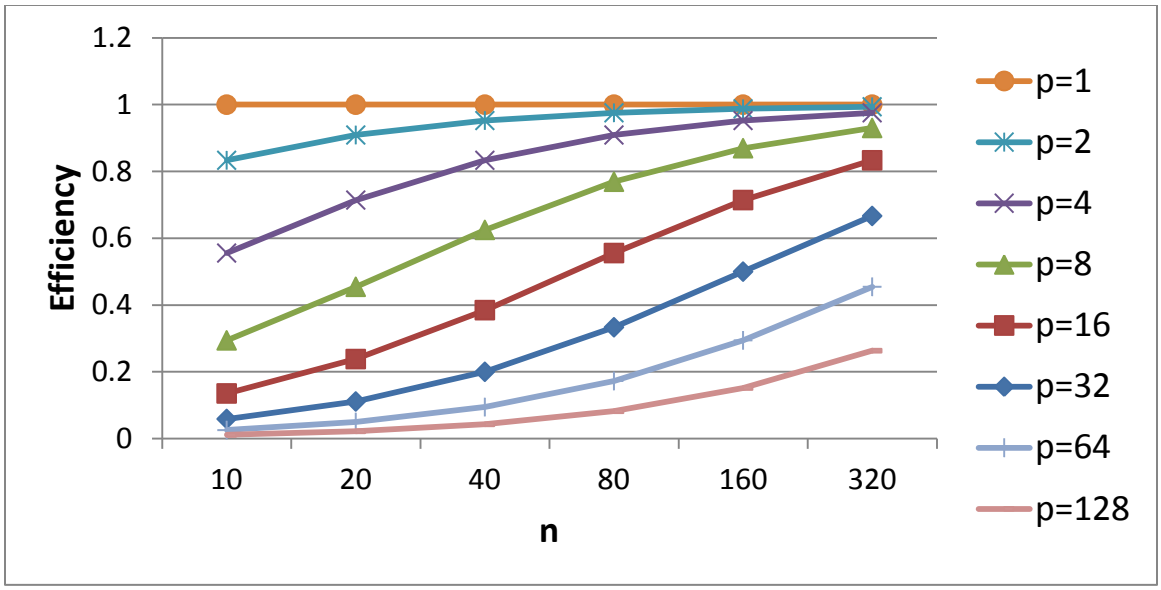
When n is fixed, speedup will increase if p is increased but it will finally decrease after reaching the optimal p.



When p is fixed, speedup will increase if n is increased, but speedup will finally get saturated when n is large enough.



When n is fixed, efficiency will always decrease if p is increased



When p is fixed, efficiency will always increase if n is increased.

(b) $T_p = T_s / p + T_o$
$S_p = T_s / T_p$
$E_p = S_p / p = T_s / ((T_s / p + T_o) * p) = T_s / (T_s + T_o * p) = 1 / (1 + (T_o / T_s) * p)$
If $T_{overhead}$ grows more slowly than T_{serial}, $T_o / T_s \rightarrow 0$, $E_p \rightarrow 1$
If $T_{overhead}$ grows faster than T_{serial}, $T_o / T_s \rightarrow \infty$, $E_p \rightarrow 0$

Question 4: Consider a program that consists of a large number of iterations, where the time to execute one iteration on a single processor is 1000 n.sec. If this program is parallelized on two processors, each iteration would require (500 + M) n.sec, where M is the time to exchange an xbyte message between the two processors. Assume that M = (100 + 10x) n.sec.
a) For what value of x (amount of communication per iteration) would executing the program on two processors result in a speedup larger than 1?
b) For what value of x would executing the program on two processors result in an efficiency larger than 0.7.
c) What is the speedup and efficiency if x = 20 bytes
d) Assume that it is possible to buffer data locally and group the communication such that only one message of 2x bytes is exchanged every two iterations, rather than a message of x bytes every iteration. How would this affect the speedup? Compare the speedup when x=20 bytes (and messages exchanged every two iterations) with the answer of part (c).

Answer:
(a) We compare the times fop one iteration
$T_s = 1000$
$T_p = 500 + M = 500 + (100 + 10x) = 600 + 10x$
$S_p = T_s / T_p = 1000 / (600 + 10x) > 1$
$T_s > T_p$

$1000 > 600 + 10x$
$10x < 400$
$x<40$

x should be smaller than 40 to get a speedup larger than 1.

(b) p=2
$E_p = S_p / p > 0.7$
$T_s / T_p > 0.7 * p = 1.4$
$1000 > (600 + 10x) * 1.4$
$10x < 1000/1.4 – 600$
$x < 12$

x should be smaller than 12 to get an efficiency larger than 0.7.

(c) x=20, p=2
$T_s = 1000$
$T_p = 500 + M = 600 + 10x = 800$
$S_p = T_s / T_p = 1.25$
$E_p = S_p / p = 1.25 / 2 = 0.625$

(d) In this part, we compute the times for two iterations
$T_s = 2 (1000) = 2000$
With a message sent per iteration, the parallel time is:
$T_{p-1} = 2 (500 + M) = 2 (500 + 100 + 10x) = 1200 + 20x = 1600$
With one message sent every two iterations, the parallel time is:
$T_{p-2} = 2 (500) + (100 + 20x) = 1100 + 20x = 1500$
$T_{p-1} > T_{p-2}$
$S_{p-1} < S_{p-2}$

Speedup will increase if two small messages are merged into one large message.
$S_{p-1} = 2000/1600 = 1.25$
$S_{p-2} = 2000 / 1500 = 1.33$