CS1356 Introduction to Information Engineering

Homework 4

Due: Dec 9, 2009 in class

Remember to write your name and student ID

1. Alice is writing Christmas cards to her friends. She has 60 cards to send, so she asks two friends to help with that. To make the work more efficient, she divides the work into four tasks, and the time for each task is given as follows.
   (1) Sign the card (20 seconds)
   (2) Fold the card and put it into an envelope (15 seconds)
   (3) Write the receiver's name and address (25 seconds)
   (4) Put a stamp on envelope and seal the envelope (20 seconds)

   Alice needs to do the first task (sign the cards) in person. **(30%)**

   (a) If Alice does task (1) and (2), and her friends do task (3) and (4) in pipeline, how fast can they finish the work? Justify your answer.

   (b) Can you think a method that is faster than the pipeline in (a)? Can your strategy be used to speed-up the instruction execution on computers? Justify your answer.

   Ans:

   (a) All Alice's friends need to wait Alice to finish the first two tasks, so the time needs to finish 60 cards is (20+15)*60+25+20 = 2145 seconds.

   (b) The shortest time to expect is (20+15+25+20)*60/3= 1600 second. The following is one possible way: (The time needs may be a little bit more than 1600 s)

Alice sign 60 cards (20*60=1200 s)		Alice and her friends do (4) together (60*20/3=400 s)
Friend 1 writes addresses of 30 cards (25*30=750 s)	Friend 1 does (2) for 30 cards(15*30=450 s)	
Friend 2 writes addresses of 30 cards (25*30=750 s)	Friend 2 does (2) for 30 cards(15*30=450 s)	

Both yes and no could be the answer to the second question, depending on your viewpoints. If your point is the stages in the instruction execution need be run in order, then the answer is NO. However, if your point is allowing multiple hardware handle the work in the same stage, then it is YES. In fact, advanced architectures, such as superscalar or VLIW, use similar idea to speed up the program execution.

2. TAs want to compute the average of the midterms by hand. They first need to sum up all the students' grades, and then divide the sum by the total number of students. There are 64 students in the class, and each TA needs 30 seconds to add up two numbers, and 60 seconds to do the division. (They do the calculation carefully.) **(40%)**
   (a) If there are 16 TAs, how fast can they finish the calculation? Justify your answer.
   (b) What is the speedup of 16 TAs comparing to just one TA? Can you modify the Amdahl's law to explain the speedup?

Ans:

(a) They can do it as follows.

(1) In the first 90 seconds, every TA adds 4 numbers. There are 16 partial sums.
(2) In the next 30 seconds, 8 TAs work and 8 TAs rest. Each of working TA adds two partial sums. After this, there are 8 partial sums.
(3) In the next 30 seconds, 4 TAs work and 12 TAs rest. Each of working TA adds two partial sums. After this, there are 4 partial sums.
(4) In the next 30 seconds, 2 TAs work and 14 TAs rest. Each of working TA adds two partial sums. After this, there are 2 partial sums.
(5) In the next 90 seconds, one TA works and 15 TAs rest. The working TA adds two partial sums and do the division.

The total time is 270 seconds.

(b) For one TA, there are 63 additions and 1 division to do. The time is 63*30+60=1950 seconds. So the speedup is 1950/270 = 7.22.
The original Amdahl's law cannot fully explain this speedup. But we can modify it as follows. First, we divide the job into 5 tasks. The first task, as described in (1), can be parallelized by 16 persons; the second task, as described in (2), can be parallelized by 8 persons, but not 16 persons; …; the last task, as described in (5), cannot be parallelized at all. If we use $t_1, t_2, \ldots, t_5$ for the time of task 1, task 2, … task 5 respectively, the speedup is calculated as

$$S = \frac{t_1 + t_2 + t_3 + t_4 + t_5}{t_1/16 + t_2/8 + t_3/4 + t_4/2 + t_5}$$

Of course, you can let $f_i = \frac{t_i}{t_1 + t_2 + t_3 + t_4 + t_5}$ for i =1,..5, and let $n_1 = 16, n_2 = 8, n_3 = 4, n_4 = 2, n_5 = 1$. The above formula becomes

$$S = \frac{1}{f_1/n_1 + f_2/n_2 + f_3/n_3 + f_4/n_4 + f_5/n_5}$$

3. *iPlayer* is a new MP3 player with wirelessly communication ability. It works as follows
   (1) When user selects a song, iPlayer will check if the song is in the local memory.
   (2) If the song is in the local memory, iPlayer will play it.
   (3) If not, iPlayer will download it from a server, store the song, and play it.
   (4) If the local memory has no space to store the downloaded song, iPlayer will randomly delete one or more songs until the downloaded song can be stored. (**30%)**

   (a) Can you show the analogies between how iPlayer works and how virtual memory works? And what are the differences between them?

   (b) For task (4), can you think a better way than random deletion, such that the chance of performing task (3) can be reduced? What is your assumption? And what kind of information need be kept track of?

Ans:

(a) There are many similarities between these two. Here lists a few.

iPlayer	Virtual memory
Local memory	Physical memory
Remote server	Hard disk
Download the requested songs from server	Rotate pages back and forth between main memory and hard disk
Keep track which songs are in the local memory	Virtual address mapping (This one combing the previous one is called “paging”)

The major difference is for iPlayer, the songs not played need not be send back to server. They can be just removed. But for virtual memory, the unused pages need be put back to hard disk. Some other minor differences such as (1) the size of songs is not fixed, but the size of pages is the same. (although there are dynamic sized virtual memory.)

(b) There is no single correct answer to this problem, as long as your assumptions make sense. For example, if the assumption is “people will not listen to the songs they just played for a while”, then the strategy of (4) should be “remove the songs just played”. To implement such function, iPlayer needs to record the history of the played songs. Of course, if you assumptions is “people would like to listen the songs they just played”, then the strategy becomes “remove songs that played long time ago”, which also requires iPlayer to store the playing history.