EEE4084F Quiz 2: Lectures 4-9

########

This quiz is for marks!

PLEASE ANSWER ON A SEPARATE PAGE

MAKE SURE YOUR STUDENT NUMBER IS ON YOUR ANSWER PAGE(s)

TOTAL NUMBER OF QUESTIONS : FIVE

time(min):

#	Question	Marks
Q1	Timing is an important part of establishing whether one solution is better than another. For example, prac 1 involved implementing a median filder and using gettimeofday to measure the time it took. Abswer the following: (a) What exactly is meant by the term 'wall clock time' in the context of benchmarking computation? [2 mark] (b) Present a shot counter argument to the premise: "wall clock time should be measured very accurately, +/- 1us if not better". [2 marks] (c) Briefly describe a recommended practice for timing programs to establish reliable time results (mention at least three methods; 1 mark for quality of answer). [4 marks]	8
A	(a) wall clock time is essentially time for the human running the program, it refers to real / actual time, as in the number of real-time seconds it takes the benchmark program to run. (b) wall clock time is what the human preceives, which tends (from my own experience anyway) tends not to be more precise than about 100ms at best. So going to finer resolution than 1ms would be rather irrelevant to a human. (c) Run the program at least three times. Discard the first run. Run the program on different types of inputs. Close all other programs running on the machine so that they don't use any CPU time. Remove debug messages / printf where relevant as these can effect the repeatability (e.g., scrolling terminal window to show debug output). Use more than one timer (e.g. use your watch to get an impression the time reported is correct).	

	This algorithm relates to both Q2 and Q3.
	
	`Image pic = loadpicture("bigdaddy.jpg"); // load an existing 24-bit image`
	`Image out = new Image(pic.width,pic.height,8); // create a blank 8-bit canvas`
	`out.graymap(); // colour map ranges from RGB(0,0,0) for colour 0 to RGB(255,255,255)`
	`int q = split();`
	`// the above line divides the program into two copies running as separate threads and`
	`// returns: q=0 for thread 1 (original program), q=1 for thread 2 (the new process)`
	`for (int y=q; y<p.height; y+=2) {`
	`  for (int x=0; x<p.width; x++) {`
	`  // do scaled RGB -> gray conversion`
	`  unsigned intensity = (unsigned)((float)`
	`  0.299 * pic.px(x,y).r + 0.587 * pic.px(x,y).g + 0.114 * pic.px(x,y).b);`
	`  // save back into new image`
	`  out.px(x,y) = intensity;`
	`  }`
	`}`
	`if (q==0) out.save("daddy2.gif");`	

Q2	**Answer the following questions in relation to the algorithm above:** (a) In a sentence describe what does the above algorithm does? [2 marks] (b) Is this actually a parallel solution? Or do both threads do the same thing, meaning that you'd get exactly the same result if you removed the int q=slipt(); line in the code (and replaced it with int q=0;). Motivate your answer referring to the code if needed. [4 marks] (c) Choose the letter A-E to specify which type of method has been used to divide up the memory to work on it in parallel in the algorithm. [2 marks] A. Contiguous B. Partitioned C. Interleved D. Interlaced E. Haphazard	8

A	(a) the algorithm above loads in a full-colour image, converts it into an 8-bit gray scale image and then saves it to disk. (b) Yes it is a parallel solution, the threads do slightly different operations: the first thread works on even rows, and the second thread works on odd rows. (c) D	
Q3	**Answer the following questions in relation to the algorithm above:** (a) The above algorithm is a 2-thread solution (the split() function causes the spawning off of a new thread), Briefly describe how you could make this into a 4-thread solution. You can add a code snippet or diagram to help your explanation. [7 marks] (b) The speed-up was a factor of **1.2** when moving from the 1-thread to the 2-thread solution shown above. Assuming that you are running on a 4-core machine, discuss and motivate what speed-up you would expect when moving from this 2-thread solution to a 4-thread solution. [3 marks]	10
A	(a) After the initial split(), another slipt would be needed. Consider that the first thread is A. Then the threads will be divided as follows: A -> A + B A -> A + C B --> B + D so this results in 4 threads. The second split will need another variable (call it r) and r will have the value of 0 or 1. Thus, based on the values q, r you can tell which thread you are in (e.g., q=0 r=0 would be set for the original thread, and q=1 r=1 would be set for the fouth thread). A simple approach would be to change x to increment by 2 also, and to have it start from the value of r. In that way each thread does 1/4 the work. (b) The program can be decomposed as follows: X + Y where X is the sequential part and Y the parallel part. The speed-up could only have happened in the Y section. Now consider the equations: X + Y = 1 (for original program, say it took 1s to run) and X + Z = 0.83 (for 2-thread ver, i.e. this took 0.83 s if there was a 1.2 speedup) Let's assume Z = Y/2 (i.e. best case with no noticeable overhead for thread creation)	
A	In that case: X + Y/2 = 0.83 and thus we have a simultaneous equation we can substitute X giving 1 - Y = 0.83 - Y/2, solve for Y and get Y = 0.34 This means that X = 0.66s (i.e. the non-parallel part) and Y = 0.34s So the time for 4 threads is going to be X + Y/4 = 0.66 + 0.085 = 0.745 The speedup from the original version will then be: 1/0.745 = 1.343 And the speedup for 4-threads over 2-threads will be: 1.343 / 1.2 = 1.12 <= answer And at this point the programmer will say something like “oh how disappointing, I’ll have to parallelise that hairy JPEG codec in order to boost performance substantially or use images with more blank spaces”. Can we be reasonably certain this answer is correct? Here’s a quick check: If the speedup was 1.2 for the 4-thread solution, then the total time (assuming version 1 took 1s) would be 1 * 0.83 * 0.83 = 0.689s this is getting rather close to the 0.66s limit, if we assumed 8-threads gave another 1.2 increase over the 4-thread solution then we come out with 0.83^3 = 0.57 and you’re bust (to use a Blackjack term) because your parallel solution can’t be better than 0.66s since that’s how long it takes to load the image before the split() is called.	

Q4	This question relates to textbook CH13. Refer to formulae below if needed. Consider the following system built around a hypothetical Power Vector Processor (PVP) microprocessor chip. The PVP can do 64 simultaneous register arithmatic operations each clock cycle, and can turn off any number of its vector operations (e.g., running from 1 to a maximum of 64 simultaneous operations) per clock. When each vector operation is active it draws current, but when turned off draws zero current (its so little current you can assume it is 0 anyway). The main application that the PVP run involves the following arithmetic operations in the *main loop* : 10 000 x sequential (single-operation) operations 10 000 x 32 simultaneous vector operations 20 000 x 64 simultaneous vector operations PS: I was planning to remove one of the (a)'s to save you time but forgot to, depending on the marks acheived, I might push all the results up a bit to compensate for this mistake. The VPV operates at a 100MHz clock (note each instruction completes in just 1 clock cycle). (a) What is the relevance of the Required Computation Rate (RCR) metric? Explain the reason for determining this number before completing the design of a system. [2 mark] (a) Assuming system just continues running the main loop of operations as described above, what is the *peak computation rate* for that part of the system? [3 marks] (b) Consider a certain program for which the PVP executes 40 000 sequential instructions (i.e. ops with one vector core turned on) and then ten repeats of the main loop, and then exits. How long would this program take to run? [3 marks] Considering that there are only arithmatic opetations in the main loop, what is the *sustained computation rate* for running this program? [3 marks] (c) If the PVP processor draw on avarage 50W while running the main loop, what can you say about the average power efficiency when the system is running the loop? [3 marks]	11
A	(a) The RCR is useful in determining what kind of speed the system to be designed needs to deliver, which in turn influences the choice of processors (for example if you need to develop custom ASICs or if an existing microcontroller can do the jop). (a) arith ops per clock = (1 * 10, 000 + 32 * 10, 000 + 64 * 20,000)/(1 * 40,000) = (1 + 32 + 128) / 4 = 161/4 = 40.25 PCR = 40.25 * 100 000 000Hz = 4025 x 10^6 FLOPS or 4.025 GFLOPS (b) 40,000 + 10*(10,000 + 10,000 + 20,000) = (4 + 10*(1+1+2)) * 10,000 = 440,000 clocks = 440,000 / 100,000,000 = 4.4ms. (i.e., pretty quick with each instructions only taking one clock cycle to complete) (c) num arith ops = 40,000 + 10*(1 * 10,000 + 32*10,000 + 64*10,000) = 10 100 000 ops sustained computation rate = num arith ops / time to run = 10 100 000 / 4.4ms = 2.3 * 10^9 ops/sec = 2.3 GFLOPS power efficiency = comp. rate / power consumed = 2.3 * 10^9 / 50 = 46 MFLOPS / W	
Q5	(a) Briefly define cloud computing. [3 marks] (b) What is the relation between the hypervisor and the operating system(s) used in a cloud computing system? [2 marks] (c) What is meant by a course-grained problem? Give one example of such a problem or application. [3 marks]	8

A	(a) Cloud computing is a style of computing in which dynamically scalable and usually virtualized computing resources are provided as a service over the internet. (b) The hypervisor is a software technology that connects the hardware platform of a computer in the cloud to an operating system. A cloud supports running different operating systems on the same hardware platform, possibly different O/S at the same time to support different applications running simultaneously on a machine. (c) A course-grained problem is one in which elements of the data are largely independent, thereby allowing the data to be partitioned into blocks that can be processed independently with little sharing of intermediate results.	
Q6	*** BONUS MARK QUESTION: ***	[2]
	In Professor Olukotun talk on 23 March, he spoke about which of the following strategies that the Stanford Pervasive Parallism Lab (PPL) is focusing on as a possible solution to programming parallel systems…? (choose a letter below)	
	A. C++ and Java automatic parallelism B. Partitioning using MPI C. Domain specific lanaguages D. CPU + GPU clusters E. lots of simple processors	
A	The right answer is C !!	
	TOTAL :	45

Appendix -- useful formulae

Required Computation Rate (RCR) = num operations to be executed / time available for computation

Peak Computation Rate = (num arithmetic processor operations per clock cycle) x (maximum clock rate)

Sustained Computation Rate = num arithmetic operations executed by program / time program takes to run

Achievable Efficiency = Sustained Computation Rate / Peak Computation Rate

Power efficiency = computation rate / power consumed

Communication-To-Computation Ratio = time spent calculating / time spent communicating

Power Consumption of CMOS device: $P = CfV^2$

where: C = Gate Capacitance f = Clock Frequency V = Supply Voltage