Module 1: Fundamentals of Quantitative Design and Analysis

Problem 1.

Your company just bought a new quad-core Intel Itanium 9340 processor and you have been tasked with optimizing your software for this processor. You will run two applications on this processor, but the resource requirements are not equal. The first application needs 65% of the resources and the other only 35% of the resources.

1.1. Given that 70% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation?

1.2. Given that 80% of the second application is parallelizable, how much speedup would this application observe if run in isolation?

1.3. This time you are running both applications. Given that 70% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it, but not the second application?

1.4. How much overall system speedup would you achieve if you parallelized both applications, given that 70% of the first application is parallelizable and that 80% of the second application is parallelizable?

Solution: The recommended strategy for the resolution of this problem is to normalize the old (i.e. unenhanced, un-parallelized) execution time to **unity**. Specific solutions to the four sub-problems follow. Also, hand-drawn diagrams are provided in the appendix.

1.1. Since 70% of the first application is parallelizable and it is run in isolation (i.e. the second application is **not** run) and the speedup is 4, the time it takes the first application when parallelized is 0.7/4. Implicitly, we are told that 30% of the time no parallelization is applicable.

Assuming that the unenhanced execution time is unity, this means that the enhanced execution time is the sum of 0.7/4 + 0.3 = 1.9/4. Consequently, the speedup is: S = 1.0 / (1.9/4) = 4/1.9 = 2.1052.

1.2. The solution of this sub-problem is quite similar to the one above. This time, we are told that 80% of the second application is parallelizable and that that application is run in isolation. Also, implicitly, 20% of the time the application is **not** parallelizable.

It follows that the application takes 0.8/4 + 0.2 = 0.4 total time when parallelized and that the speedup is: S = 1.0/0.4 = 2.5.

1.3. In this sub-problem we are asked to find the speedup when both applications are being run. Recall that when both applications are run, the first one requires 65% of the resources while the second only 35%.

Thus, in unenhanced mode, the first application uses 65% of 1.0 which is 0.65 time units while the second one uses 35% of 1.0 which turns out to be 0.35.

Now, 70% of 0.65 is 0.455 and this represents the fraction of the total unenhanced running time devoted to the first application that is parallelizable. The remaining time allocated to the first application, that is, 0.64-0.455 = 0.195 remains unenhanced. Also, recall, we are told that the second application is not parallelized.

This means that out of the 1.0 (unity) total unenhanced execution time only 0.455 is enhanced, the remainder is not. Applying Amdahl's law, we obtain the following speedup:

$$S = 1.0/(0.455/4 + 0.195 + 0.35) = 1.0/ 0.65875 = 1.5180.$$

1.4. In this sub-problem both applications are parallelized: 70% of the first application is parallelized and 80% of the second application is also parallelized. Recall that, as discussed above, the first application uses 65% of the unenhanced time (i.e. 1.0) which is 0.65 time units while the second one uses 35% of 1.0 which turns out to be 0.35. Proceeding as in the preceding sub-problem, we find that total fraction of

time during which the first application is parallelized is 0.455. Next, 80% of 35% is 0.28 and this says that the fraction of time that the second application is parallelized is 0.28. The remainder execution time remains unenhanced and this amounts to 0.195+ 0.07=0.265. Can you see why?

Now, the total enhanced execution time is:

- first application: 0.455/4= 0.11375;
- second application: 0.28/4 = 0.07.

With this behind us, we are now in a position to compute the speedup

S = 1.0/ (0.455/4 + 0.28/4 + 0.265) = 1.0/0.44875 = 2.2284.

Problem 2.

[10/10/20] <1.9> Imagine that your company is trying to decide between a single-processor system and a dual-processor system. Figure 1.26 gives the performance on two sets of benchmarks—a memory benchmark and a processor benchmark. You know that your application will spend 40% of its time on memory-centric computations, and 60% of its time on processor-centric computations.

a. [10] <1.9> Calculate the weighted execution time of the benchmarks.

b. [10] <1.9> How much speedup do you anticipate getting if you move from using a Pentium 4 570 to an Athlon 64 X2 4800+ on a CPU-intensive application suite?

c. [20] <1.9> At what ratio of memory to processor computation would the performance of the Pentium 4 570 be equal to the Pentium D 820?

Chip	# of cores	Clock frequency (MHz)	Memory performance	Dhrystone performance
Athlon 64 X2 4800+	2	2,400	3,423	20,718
Pentium EE 840	2	2,200	3,228	18,893
Pentium D 820	2	3,000	3,000	15,220
Athlon 64 X2 3800+	2	3,200	2,941	17,129
Pentium 4	1	2,800	2,731	7,621
Athlon 64 3000+	1	1,800	2,953	7,628
Pentium 4 570	1	2,800	3,501	11,210
Processor X	1	3,000	7,000	5,000

Figure 1.26 Performance of several processors on two benchmarks.

.

Solution: This problem is typical of an entire series of (similar) problems that are asking us to compare the performance of a number of existing or proposed processors. The basis of the comparison is the performance of these processors on two benchmarks: one for memory performance and the other (Dhrystone, in this case) for CPU performance.

a. To solve this sub-problem, all we have to do is to evaluate the performance of the processors in Figure 1.26 which is a weighted average of 40% memory-centric operations and 60% CPU-centric applications.

As an illustration, the required performance of Athlon 64 X2 4800+ is: 0.40 * 3423 + 0.6 * 20,718 = 13,800. Proceeding similarly, we obtain the following table:

Athlon 64 X2 4800+	13,800
Pentium EE 840	12,627
Pentium D 820	10,332
Athlon 64 X2 3800+	11,454
Pentium 4	5,665
Athlon 64 X2 3000+	5,758
Pentium 4 570	8,126
Processor X	5,800

b. This time we are specifically told that we should evaluate performance, and contemplated speedup, when we migrate from a Pentium 4 570 chip to an Athlon 64 X2 4800+. The basis of the comparison in this case is CPU-performance. Since the Drystone benchmark is a typical of CPU-intensive applications, we use the data in the last column of Figure 1.26. With this in mind, we write the speedup (S) as

$$S = CPU\_Pentium4/CPU\_Athlon4800 = 20{,}718/11{,}210 = 1.8482.$$

c. This time we are interested in comparing the weighted (i.e., overall) performance of the Pentium 4 570 chip with that of the Pentium D 820. For this purpose, a mix of memory-centric and CPU-centric applications. Let x denote the fraction of the mix consisting of memory-centric applications. Evidently, $1-x$ will denote the fraction consisting of CPU-centric applications. In order for the performance of the two chips to be the same, we need to have:

$$3501 * x + 11{,}210 * (1-x) = 3000 * x + 15{,}220 * (1-x).$$

Solving for x, we obtain: $x = 0.8889$, which is to say that for a mix consisting of, roughly, 88.9% memory-centric applications and 11.1% CPU-centric applications, the performances of the two chips under consideration agree.

Problem 3.

[15/10] <1.9> Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time *when the enhanced mode is in use*. Recall that Amdahl's law depends on the fraction of the original, *unenhanced* execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's law.

a. [15] <1.9> What is the speedup we have obtained from fast mode?

b. [10] <1.9> What percentage of the original execution time has been converted to fast mode?

Solution:

This question further explores the effects of Amdahl's Law, but the data given in the question is in a form that cannot be directly applied to the general speedup formula.

a. Because the information given does not allow direct application of Amdahl's Law we start from the definition of speedup:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Time}_{\text{unenhanced}}}{\text{Time}_{\text{enhanced}}}$$

The unenhanced time is the sum of the time that does not benefit from the 10 times faster speedup plus the time that does benefit, but before its reduction by the factor of 10. Thus,

$$\text{Time}_{\text{unenhanced}} = 50\%\ \text{Time}_{\text{enhanced}} + 10 \times 50\%\ \text{Time}_{\text{enhanced}} = 5.5\ \text{Time}_{\text{enhanced}}$$

Substituting into the equation for Speedup yields

$$\text{Speedup}_{\text{overall}} = \frac{5.5\text{Time}_{\text{enhanced}}}{\text{Time}_{\text{enhanced}}} = 5.5$$

b. Using Amdahl's Law, the given value of 10 for the enhancement factor, and the value for $\text{Speedup}_{\text{overall}}$ from part (a), we have

$$5.5 = \frac{\text{Fraction}_{\text{enhanced}}}{1 - \text{Fraction}_{\text{enhanced}} + \frac{\text{Fraction}_{\text{enhanced}}}{10}}$$

Solving shows that the enhancement can be applied 91% of the original time.

Problem 4.

[20/10/10/10/15] <1.9> In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization*. Vectors are discussed in Chapter 4, but you don't need to know anything about how they work to answer this question!

a. [20] <1.9> Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the *y*-axis "Net speedup" and label the *x*-axis "Percent vectorization."

b. [10] <1.9> What percentage of vectorization is needed to achieve a speedup of 2?

c. [10] <1.9> What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?

d. [10] <1.9> What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?

e. [15] <1.9> Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2× speedup in the vector unit (beyond the initial 10×)?

Solution: In what follows, let PV stand for the percentage of vectorization.

a. Plot net speedup $S = \frac{1}{1-PV+\frac{PV}{10}}$ for $0 \leq PV \leq 1$.

b. From the equation in (a), if speedup = 2, then the percentage of vectorization is 5/9 or 56%.

c. For PV=5/9, the time spent in vector mode = $\frac{\frac{PV}{10}}{1-PV+\frac{PV}{10}} = 1/9$ or 11%.

d. From the equation in (a), if speedup = 10/2 =5 then PV = 8/9 or 89%.

e. Observe that the increased percent vectorization has to match a hardware speedup applied to the original 70% vectorization. In other words, we need have

$$\frac{1}{1-PV+\frac{PV}{10}} = \frac{1}{1-0.7+\frac{0.7}{20}}$$

This is an equation that can be solved for PV. Solving shows that the vectorization muct increase to 74%, not a very large increase. Improving the compiler to increase vectorization another 4% may be easier and cheaper than improving the hardware by a factor of two.

Problem 5.

[20/20/15] <1.9> When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's law equation does not take into account this trade-off.

a. [20] <1.9> If the new fast floating-point unit speeds up floating-point operations by, on average, 2×, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

b. [20] <1.9> Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5× slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

Solution: This problem boils down to using Amdahl's law to figure out to speedup obtained under various assumptions.

a. Here, we are told that the floating-point unit, when in use, brings about a speedup of 2. We are also told that the unit can be employed 20% of the time. The basic strategy we employ is to normalize the unenhanced execution time to 1.0. With respect to this, let f=0.2 (i.e. 20%) be the fraction of the unenhanced execution time that the floating-point unit can be used. Let s=2 be the resulting speedup. With this, the overall speedup is determined as follows:

$$S = \frac{1.0}{1 - f + \frac{f}{s}} = \frac{1.0}{1 - 0.2 + \frac{0.2}{2}} = \frac{1.0}{0.8 + 0.1} = \frac{1.0}{0.9} = 1.11$$

b. Here, we have a more complicated scenario. We still have the floating-point unit that can be employed for 20% of the unenhanced execution time and yields a speedup of 2. In addition, we are told that, as a direct effect of using the floating-point unit, cache accesses, accounting for 10% of the unenhanced execution time, are slowed down by a factor of 2/3.

   We let f_1 =0.2 and f_2 = 0.1. And let the corresponding speedups be s_1 = 2 and s_2 = 2/3. In this notation, the overall speedup is:

$$S = \frac{1.0}{1 - f_1 - f_2 + \frac{f_1}{s_1} + \frac{f_2}{s_2}} = \frac{1.0}{1 - 0.2 - 0.1 + \frac{0.2}{2} + \frac{0.1}{2/3}} = \frac{1.0}{0.7 + 0.25} = \frac{1.0}{0.95} = 1.05$$

Problem 6.

[15] <A.9> Compute the effective CPI for MIPS using Figure A.27. Assume we have made the following measurements of average CPI for instruction types:

Instruction	Clock Cycles
All ALU instructions	1.0
Loads-stores	1.4
Conditional branches	
Taken	2.0
Not taken	1.5
Jumps	1.2

Assume that 60% of the conditional branches are taken and that all instructions in the "other" category of Figure A.27 are ALU instructions. Average the instruction frequencies of gap and gcc to obtain the instruction mix.

Solution: If you look carefully at the average instruction frequencies for **gcc** and **gap** in Figure A.27, you will notice that the sum of frequencies for add up to 99.5% and 100.2% This is normal and is attributable to various rounding errors. Nonetheless, it is very desirable to somehow normalize the combined frequencies to 100%. There are several possible solutions to this. One of them is detailed in the table below.

Instruction type	Frequency
ALU	48.7%
Load/store	37.6%
Conditional branches	10.7%
Unconditional branches (jumps)	3.0%

Recall that the effective CPI is a weighted average of the CPIs of individual instruction types weighted by the frequency with which they occur in the instruction mix. Since we are told that for conditional branches the effective CPI depends on whether of not the branch is taken, we need to first compute CPIcb namely the effective CPI of conditional branches.

CPIcb= 2.0 *0.6 + 1.5 * 0.4 = 1.2 + 0.6 = 1.8.

Let CPIalu, CPIloadstore, CPIcb and CPIub denote the individual CPIs of the various instruction types above. Recall that these various CPIs are given in the statement of the problem

with the exception of CPIcb that we just computed. We are now in apposition to evaluate the effective CPI as follows:

CPI = CPIalu * 0.487 + CPIloadstore * 0.376 + CPIcb * 0.107 + CPIub * 0.03
= 1.0 * 0.487 + 1.4 * 0.376 + 1.8 * 0.107 + 1.2 * 0.03
= 1.242 clockcycles/instruction.

Problem 7.

1.18 [10/20/20/20/25] <1.10> When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things: percentage of the application that can be parallelized and the cost of communication. Amdahl's law takes into account the former but not the latter.

a. [10] <1.10> What is the speedup with *N* processors if 80% of the application is parallelizable, ignoring the cost of communication?

b. [20] <1.10> What is the speedup with 8 processors if, for every processor added, the communication overhead is 0.5% of the original execution time.

c. [20] <1.10> What is the speedup with 8 processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

d. [20] <1.10> What is the speedup with *N* processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

e. [25] <1.10> Write the general equation that solves this question: What is the number of processors with the highest speedup in an application in which *P*% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time?

Solution.

(a) Let S denote the speedup. Amdahl's law allows us to write S = $\frac{1}{1-0,8+\frac{0.8}{N}} = \frac{1}{0.2+\frac{0.8}{N}}$.

(b) We are not told that we have 8 processors available and that for each of the 7 additional processors, the communication overhead increased by 0.5%. In this case, the speedup is

$$S = \frac{1}{1-0.8+\frac{0.8}{8}+7*0.005} = 2.98$$

(c) For this problem, observe that you have to double the number of processors three times to go from 1 to 8: 1->2->4->8. In this case the speedup reads

$$S = \frac{1}{1-0.8+\frac{0.8}{8}+3*0.005} = 3.17$$

(d) We continue to assume that 80% of the application is parallelizable. In this case, we have to double the number of processors $\log_2 N$ times to go from 1 processor to N processors. The speedup is

$$S = \frac{1}{1 - 0.8 + \frac{0.8}{N} + 0.005 * \log_2 N}$$

(e) Let *p* be the fraction (percentage) of time the original execution time is parallelizable. We are asked to determine the number *N* of processors that yields the largest speedup, assuming that every time the number of processors is doubled the communication overhead increases by 0.5%. In this case let *S(N)* be the speedup. Proceeding as in (d) above

$$S(N) = \frac{1}{1 - p + \frac{p}{N} + 0.005 * \log_2 N}$$

The number of processors that maximizes the above function of *N* is obtained by setting the derivative of *S(N)* to 0.

$$\frac{d\,S(N)}{d\,N} = -\frac{-\frac{p}{N^2} + 0.005 * \frac{1}{N * \ln 2}}{\left[1 - p + \frac{p}{N} + 0.005 * \log_2 N\right]^2} = 0$$

Solving for *N* yields

$$N = p * \frac{\ln 2}{0.005} \approx 138.63 * p.$$