CS575 Spr 2018 WA3: Speedup & Efficiency

due Mar 11 @ 11:59 pm

Problem 1: [20 pts: 4×5]

A: What is the maximum fraction of execution time that can be spent in performing inherently sequential operations, if a parallel application is to achieve a speedup of 50 over its sequential version?

B: A programmer wrote a parallel program that achieves a speedup of 9 on 10 processors. What is the maximum fraction of the computation that may consist of inherently sequential operations?

C: A parallel program takes 242 seconds to execute on a 16-processor machine. Benchmarking later reveals that 9 seconds were spent on initialization and file I/O, while all the processors were busy doing useful work during the remaining 233 seconds. What is the scaled speedup achieved by the program.

D: Programmer B benchmarks a parallel program and finds that 95% of the time of a program running on 40 processors is spent in perfectly parallel code. Predict a bound on the speedup of the program.

Problem 2: [24 pts: 6×4]

The execution times of six parallel programs, I...VI, were measured on 1,..., 8 processors, as reported in the speedup table below.

# Proc	Speedup					
	I	II	III	IV	V	VI
1	1.00	1.00	1.00	1.00	1.00	1.00
2	1.67	1.89	1.89	1.96	1.74	1.94
3	2.14	2.63	2.68	2.88	2.30	2.82
4	2.50	3.23	3.39	3.67	2.74	3.65
5	2.78	3.68	4.03	4.46	3.09	4.42
6	3.00	4.00	4.62	5.22	3.38	5.15
7	3.18	4.22	5.15	5.93	3.62	5.84
8	3.33	4.35	5.63	6.25	3.81	6.50

For each of the six programs, which of the following statements *best describes* the likely performance on 16 processors. Justify your answers using the Karp Flatt metric.

A: The speedup achieved on 16 processors will probably be at least 40% higher than the speedup achieved on eight processors.

B: The speedup achieved on 16 processors will probably be less than 40% higher than the speedup achieved on eight processors due to the large serial component of the computations.

C: The speedup achieved on 16 processors will probably be less than 40% higher than the speedup achieved on eight processors due to the increase in overhead as processors are added.

Problem 3: [12 pts: 6×2]

Let $W \geq f(p)$ be the iso-efficiency relation of a parallel system. Based on the function f, rank the following parallel programs/systems, from worst to best based on the iso-efficiency metric Next, rank them based on the per-processor scalability function (C is some constant, each instance of C is not necessarily the same).

A: $f(p) = Cp$

B: $f(p) = Cp \log p$

C: $f(p) = C\sqrt{p} \log p$

D: $f(p) = C\sqrt{p}$

E: $f(p) = p^C$, for $1 < C < 2$

F: $f(p) = p^C$, for $C > 2$

Problem 4, 5: [20 pts: 2×10]

Do problems 5.5, and 5.6 of the textbook.

Problem 6: [24 pts:]

This problem is similar the the worked out example in class where we analyzed the scalability of the 2D stencil computation, where over a loop of M iterations (time-steps) an $N \times N$ data array is updated using values from its four nearest neighbors from the previous time step (Jacobi iterations). Assume that an three dimensional array is distributed over P processors in a 3-D grid. Determine the overhead $T_o(n, p)$ assuming the two models in the class: the communication of a single message of size m is (i) $t_w m$, and (ii) $t_s + t_w m$. In each case, analyze the separate terms of the scalability relationship to determine the scalability function [2×10]. Also determine the scalability due to the inherent dependences (i.e., the fact that MN^2 processors cannot be used to compute the answer in constant time) [4 pts].