IACS INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

HARVARD
School of Engineering
and Applied Sciences

HW: A. Parallel Processing Fundamentals

Due Monday, February 15, 2021

Abstract

The objective in this homework is to gain insights into the main aspects to consider in the design, development and distribution of parallel applications.

Contributors

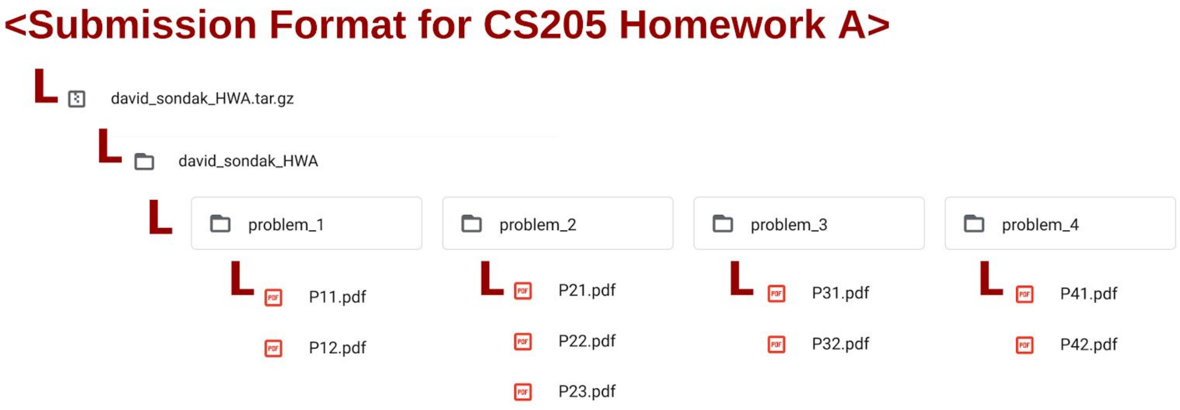
This homework was originally developed by Ignacio Illorente with input from David Sondak, Charles Liu, Matthew Holman, Zudi Lin, Nicholas Stern, Kar-Tong Tan, Dylan Randle, Zijie Zhao, Hayoun Oh, Zhiying Xu, Oluwatosin Alliyu, and Simon Warchol

Guidelines

- The files needed to do the exercises are available for download from the course website. Please follow the submission format below.
- **AWS**
  - Some of the problems in this assignment assume that you already have an AWS account and a key pair and that you are familiar with the AWS EC2 environment. To ensure these prerequisites are met, please refer to the "**First Access to AWS**" guide from Lab 1.
- **Submission**
  - Save each file as a PDF.
  - Upload on **Canvas** the files specified in each assignment.
  - See the Homework Submission Guidelines for submission instructions.

        - Follow these instructions carefully!
    - The grade on this assignment is **10% (100 points) of the final grade**

Table of Contents


<Submission Format for CS205 Homework A>

We understand that it's slightly burdensome to submit a PDF for each question. However, we believe that following the above method is straightforward, and we will not accept submissions that do not follow the above format.

1. Introduction to Parallel Thinking (25 points)

1.1. Revenge on the Campus Store (15 points)

Frustrated with the rising cost of textbooks, you have decided to enact revenge on the campus bookstore; you will pay for your textbooks with a suitcase full of small bags of pennies. Although each bag is correctly labeled with the number of pennies inside it, there are quite a few bags, and the cashiers must sum their totals to verify you are providing the correct amount of money. Armed with their calculators, it takes them 1 second to add two numbers (assume one person can only add two numbers at a time).

1. If one cashier were to add the totals of 256 bags, how long would it take?

2. Seven of the cashier's coworkers approach to see this ridiculous sight. The bright (and perhaps a bit opportunistic) cashier recruits them to help count. By working together, can they verify the total sum faster than the original cashier by herself? If so, how fast can they do this?

3. Imagine the store is big. . . very big. As a matter of fact, the store can hold an arbitrary number of employees. How fast can the employees verify the sum? That is, what are the optimal counting time and number of employees?

4. If you were feeling especially frustrated, perhaps you may use more than 256 bags. Instead, assume there are N bags. (a) How long will it take an arbitrary number of employees, E, to count the N bags? Derive a general mathematical expression in terms of N and E where E is a fixed and finite number but it could get as big as "needed". (b) Using Python, plot the result with bag count on the horizontal and counting time on the vertical. On the plot, create separate curves (around 5 or 6 will do) for different values of E. On the same graph, plot the time it would take the lone cashier to do this by herself.

   Hint: You can use the ceil( ) function in the mathematical expression that you derived.

5. We've been neglecting any overhead associated with people communicating. How does the answer to question 3, for 256 bags, change, if it takes one second to communicate a number between two employees?

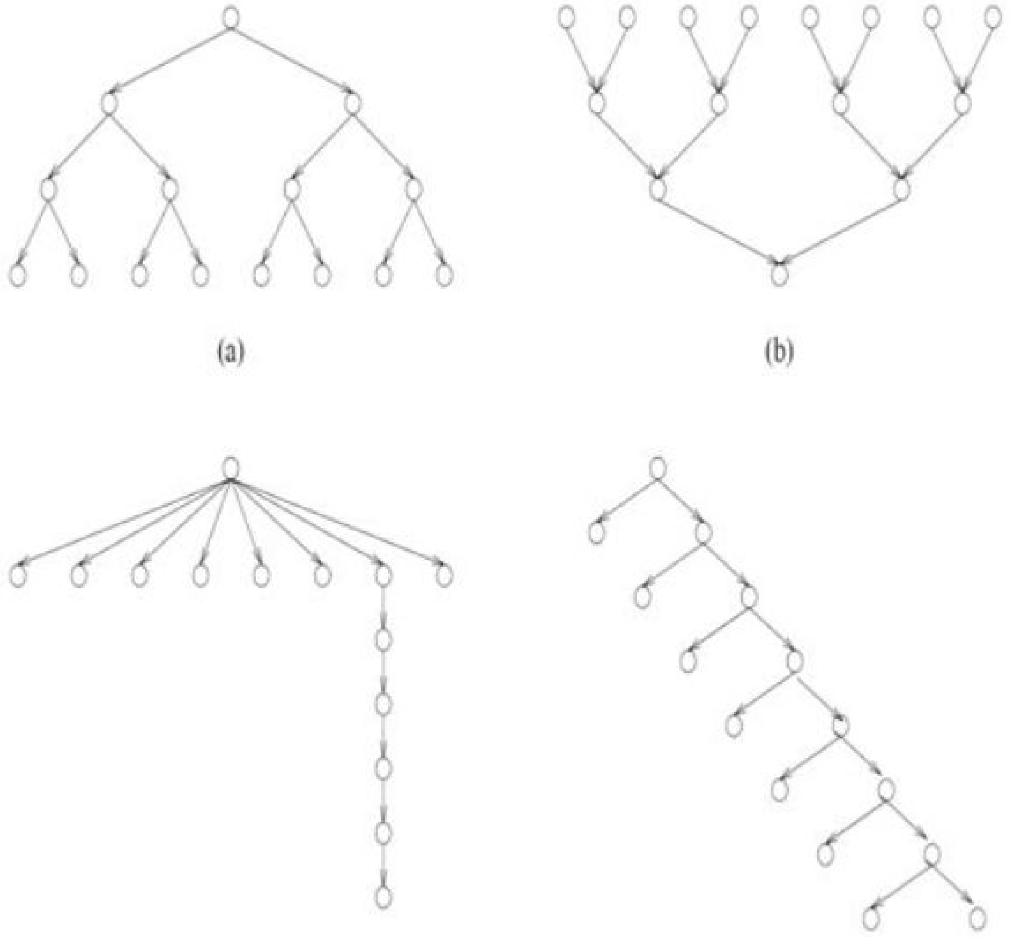
6. Also, while still assuming an arbitrary number of employees, assume it takes one second to hand a bag (or any number of bags) between two people (including from you to a cashier). How long will it take to verify 256 bags worth of coins if each cashier comes to you and takes 1 bag? What about 2 bags? Is there a better strategy? Assume you can't be handing bags to more than one person at a time. Note that, in practice, part of the counting could already start while the handing over of bags is still taking place, but *assume that this does not happen in our simulation.*

Submission
- `P11.pdf`: Answers (including a plot for the fourth question)

1.2. Graph Parallelization (10 points)

For the computation graphs a), b), c) and d) shown in the figure, determine the following assuming unit time for computation:

1. Maximum degree of concurrency
2. Critical path length
3. The minimum number of processors needed to obtain the maximum speed-up
4. The maximum achievable speed-up if the number of processors is limited to (i) 2 and (ii) 5



(a) (b)

(c) (d)

Submission

- `P12.pdf`: Answers

2. Parallel Processing Architectures (25 points)

2.1. Basic Concepts (5 points)

For the following four statements, state if the statement is correct or incorrect, explain why, and, If you think it is incorrect, explain what a correct statement would be.

1. Around 2006, Moore's law did not hold anymore. As a consequence, larger processor manufacturers started to produce multi-core processors.
2. The term power wall typically means that the maximum clock speed has been reached and that chips would be too hot if they were clocked at higher frequencies.
3. New modern programming models and tools for parallel computing enables the parallel implementation of any algorithm.
4. Before multi-core CPUs, shared-memory parallel processing was basically not possible

Submission
- `P21.pdf`: Answers

2.2. Superscalar Processors (10 points)

Modern CPUs contain multiple cores and each core has multiple copies of the same physical resources, such as execution units, issue control or data paths. This allows the execution of more than 1 instruction per core and CPU cycle. For example, the new Intel(R) Core(TM) i9-7900X CPU contains 10 cores, where each core allows the execution of 2 simultaneous threads. Its CPU Core Pipeline functionality is based on the Skylake Microarchitecture with a 6-wide superscalar design per thread able to execute 6 floating point operations per cycle . Please give a brief explanation of each answer.

1. Calculate the theoretical peak performance (in FLOPS – floating point operations per second) of an Intel(R) Core(TM) i9-7900X processor, having a clock frequency of 3.3 GHz.
2. What would be the theoretical execution time (in seconds), on the machine described previously, of calculating the dot products of two full nxn matrices, one having 10^4 x 10^4 elements, represented as floating point numbers with single precision?
3. Consider this processor running at 3.3 GHz. Assume that the average CPI (clock cycles per instruction) is 1. Assume that 10% of all instructions are stores, and that each store writes 8 bytes of data. How many processors will a 800-GB/s bus be able to support without becoming saturated?

Submission
- `P22.pdf`: Answers

2.3. Top500 Showing Signs of Stagnation (10 points)

Since 2008, experts claim that there is a slowing of the pace of performance improvement.

1. Can you estimate this slowdown or demonstrate it by developing a graph that represents the annual performance increase of top500 versus the annual growth predicted by Moore's law?

   Hint: For Moore's law, you can calculate $\frac{current\_year\_transistors}{last\_year\_transistor}$; For top500, you can calculate $\frac{current\_year\_performance}{ast\_year\_performance}$, and feel free to use the supplementary file provided with this prompt.

2. Why do you think supercomputers are not getting faster like they used to?

3. The speed limit for modern computers is now set by power consumption. If all other factors are held constant, the electricity needed to run a processor chip goes up as the cube of the clock rate: doubling the speed brings an eightfold increase in power demand. Since 2005 the main strategy for boosting performance has been to gang together multiple processor "cores" on each chip. The clock rate remains roughly constant, but the total number of operations per second increases if the separate cores can be put to work simultaneously on different parts of the same task. Large systems are assembled from vast numbers of these multicore processors. Can you source evidence to support this phenomenon?.

Submission

- `P23.pdf`: Discussion of the analysis

3. Parallel Processing on the Cloud (25 points)

3.1. Public Cloud Pricing and SLAs (10 points)

Compare the Service Level Agreement (SLA) (i.e. the up-time), penalties (i.e. the credits that you'll receive if the uptime turns out to be lower than what was promised in the SLA). and on-demand pricing (for similar instance capacity with approx 2 vCPUs, 8 GIB Mem and 32 GB SSD) of four main public cloud providers:

- Amazon AWS
- Google Compute
- Microsoft Azure
- IBM Cloud

Explain what are the "nines" given by providers as availability goals? What is the maximum downtime per year?

Submission

- `P31.pdf`: Table with the comparison and discussion

3.2. Linpack on a Single AWS Node (15 points)

Download[1], compile, and run the Linpack benchmark with **4 processes on one t2.2xlarge instance** on AWS running Ubuntu. You may need to install MPI in the node, see the course Infrastructure Guide "MPI on AWS". Tune the execution of the benchmark to maximize performance. Discuss and compare the performance with the historical results of the benchmark.

Submission

- `P32.pdf`: Explain steps and configuration needed to compile and run the benchmark (i.e. high level details such as what instance of AWS and what commands used for compiling), explain the parameters tuned to maximize performance (i.e. report the maximum Gflop), represent the obtained performance in the Top500 performance development graph[2]

[1] https://www.top500.org/project/linpack/

[2] https://www.top500.org/statistics/perfdevel/

4. Designing Parallel Programs (25 points)

4.1 Time Complexity Plots (10 points)

For visual comparison, create a single graph of the following functions of time complexity for the input in the range 0 <= n <= 100. The maximum range of your graph should be no larger than 10^5.

- n^3, 2^n, $\log_2 n$, $n!$, n^2

Submission
- `P41.pdf`: Graph with plots

4.2. Amdahl's Law (15 points)

This problem is concerned with the speedup S of solving a problem. The efficiency E of this parallel approach can be calculated with E = S/p where p is the number of processing elements.

1. Consider a parallel program for which serial work comprises 8%. Given that the speedup is given by

   $$S = \frac{p}{B \times p + (1 - B)},$$

   where B denotes the fraction of serial work to be done (i.e. B = 8%), calculate the maximum number p of processing elements such that the parallel efficiency E is at least 60%.

2. Consider a matrix $A \in \mathrm{IR}^{n \times n}$. A smoothing algorithm calculates the mean of each coefficient a_{ij} and its four neighbours before the new value a'_{ij} is stored at position (i, j) again—one single smoothing calculation takes t_{smooth} time. For a parallel approach with p processing elements the matrix A shall now be divided into p parts, with each part consisting of n/p rows and n columns. For calculating the mean at the border of each part, each processing element has to exchange its border values with its direct neighbours — a single send/receive of one value takes t_{comm} time. The communications between different pairs of processing elements can be performed in parallel, but within the same pair of processing elements, the communications must be in serial. Calculate the maximum amount p of processing elements as a function of n, t_{smooth} and t_{comm} so that the parallel efficiency E is at least 50%.

3. Explain the difference between Amdahl's Law and Gustafson's model. (Hint: Consider the different definitions of the sequential part of a parallel program). Discuss which model is best for measuring the performance of parallel algorithms.

Submission
- `P42.pdf`: Answers