Full Name:
Andrew Id:

15-418/618 Spring 2021
Exercise 2

Assigned:	Wed., Feb. 17
Due:	Wed., Feb. 24, 11:00 pm

Overview

This exercise is designed to help you better understand the lecture material and be prepared for the style of questions you will get on the exams. The questions are designed to have simple answers. Any explanation you provide can be brief—at most 3 sentences. You should work on this on your own, since that's how things will be when you take an exam.

You will submit an electronic version of this assignment to Canvas as a PDF file. For those of you familiar with the LaTeX text formatter, you can download the template and configuration files at:

http://www.cs.cmu.edu/~418/exercises/config-ex2.tex
http://www.cs.cmu.edu/~418/exercises/ex2.tex

Instructions for how to use this template are included as comments in the file. Otherwise, you can use this PDF document as your starting point. You can either: 1) electronically modify the PDF, or 2) print it out, write your answers by hand, and scan it. In any case, we expect your solution to follow the formatting of this document.

Problem 1: Task Assignment for BARNES-HUT

Recall that we discussed a version of the BARNES-HUT application that uses a semi-static assignment strategy. In this problem, we will ask you to compare the semi-static approach with other task assignment approaches for BARNES-HUT. As you discuss the likely differences in performance between these approaches, please relate your answers to the three goals for task assignment (i.e. balance the workload, reduce communication, and minimize extra work).

A. **Semi-Static versus Static Assignment for BARNES-HUT**
   In this problem, we would like for you to compare the semi-static version of BARNES-HUT that we discussed in class with a hypothetical version of BARNES-HUT that was implemented with (pure) static assignment. In particular, imagine that the static version of BARNES-HUT divides up the stars into spatially-contiguous chunks based upon their initial locations (the details of how this takes place are not important, but you can assume that equal numbers of stars are assigned to each thread). Using the metrics above, provide a qualitative discussion of the likely performance differences between these two versions of BARNES-HUT.

   Since the stars accumulate as time goes on, static assignment yields heavy load for later assigned threads. Because of this imbalance, simulation performance decreases while simulation time increases.

B. **Semi-Static versus Fine-Grained Dynamic Assignment for BARNES-HUT**
   Now consider a different hypothetical implementation of BARNES-HUT that uses distributed task queues to dynamically assign individual stars to threads at runtime. Each task in the task queues would represent a particular star. Assume that equal numbers of tasks would be assigned to each task queue at the start of a time step (partitioned in some reasonable way based upon their physical locations), and that task stealing would be used whenever a processor runs out of work. Using the metrics above, provide a qualitative discussion of the likely performance differences between these two versions of BARNES-HUT (i.e. semi-static versus dynamic).

   Dynamic assignment yields better load balance than semi-static assignment. However, dynamic assignment requires more communication than semi-static assignment. Therefore, dynamic assignment may yield better performance than semi-static assignment if the communication overhead is small.

Problem 2: Choosing the Right Grain Size for Dynamic Assignment

Consider the code on slide 12 (entitled Increasing task granularity) of Lecture 6 (entitled Performance Optimization Part 1: Work Distribution and Scheduling). Notice that the `GRANULARITY` variable is a parameter that can be used to adjust the grain size for this example of dynamic task assignment.

If one were to plot execution time of a parallel program (that uses dynamic task assignment) versus grain size, it would be common to see a U-shaped curve where the best performance is achieved at an intermediate happy medium value for the grain size (rather than at the extreme ends of the curve). In this problem, we want you to consider three hypothetical values of grain sizes: one that is too small, one that is too large, and one at the happy medium.

Similar to the previous problem in this assignment, we would like for you to relate your answers to the goals for task assignment. However, to simplify the discussion, please assume that there is no difference in locality as the dynamic task size changes, and therefore you only need to worry about the task assignment goals of workload balancing and minimizing extra work.

A. **Comparing Too Small with Happy Medium**
Using the metrics above, provide a qualitative discussion of the likely performance differences between a version of dynamic assignment where the granularity is *too small* with one where the granularity is at the optimal *happy medium*.

The too small version of dynamic assignment yields worse load balance than the happy medium version. Because computation/communication ratio lowers drastically. If granularity increased to happy medium, computation time increases for the program and performance increases due to low communication overhead compared to fine grained version.

B. **Comparing Too Large with Happy Medium**
Using the metrics above, provide a qualitative discussion of the likely performance differences between a version of dynamic assignment where the granularity is *too large* with one where the granularity is at the optimal *happy medium*.

Synchronization costs decrease with coarse grained version. But since there are too many big chunks of tasks, parallelism could not be achieved as good as happy medium version.