15-418/618 Spring 2021
Exercise 8

Assigned:	Mon., Apr. 12
Due:	Mon., Apr. 19, 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 configuation files at:

http://www.cs.cmu.edu/~418/exercises/ex7.tex
http://www.cs.cmu.edu/~418/exercises/config-ex7.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: Heterogeneous Parallelism

A. Recall the plots shown on lecture 21 slide 7. Qualitatively describe how the plots would change if the perf(r) function were changed to perf(r) = r and perf(r) = $\log r$ instead of $\sqrt{r}$ (you are welcome to consider other functions as well). How would the trade-offs be different compared to the original $\sqrt{r}$ perf function? In these cases, Would the system still benefit from heterogeneity? Why or why not?

$r > \sqrt{r} > \log r$ when r is bigger, resource per one processor is bigger and tasks are better with coarse grained algorithms.
when r is lower, there are too many small cores in system.
Plots already show that heterogeneity is beneficial.
If we increase the perf(r), the system will be more beneficial from heterogeneityi more speedup observed and vice versa. More performance with smaller cores yield power issues, more performance with bigger cores would be beneficial.

Problem 2: Domain-Specific Languages: Liszt

Recall that Liszt was a domain-specific language that was designed for solving PDEs on meshes. Because Liszt programmers do not specify the structure of the mesh itself, the Liszt programming system has considerable flexibility in choosing the best implementation for a particular target architecture.

In class, we discussed two very different parallel orchestration strategies that Liszt uses: (i) graph partitioning (with ghost cells), and (ii) graph coloring. For each of these approaches, please discuss the following: (i) what type of parallel target architecture is this approach well-suited for, and (ii) why is that the case.

A. What type of parallel target architecture is graph partitioning well-suited for, and why is that the case?

Partioning is operating on its own part of the mesh. MPI system would benefit a system like this, no need for shared address space.

B. What type of parallel target architecture is graph coloring well-suited for, and why is that the case?

Graph coloring would benefit from shared address space.