Problem selection

For this programming assignment choose one of the projects below or propose an alternate project of interest to you. If you choose a different project we need to agree in advance of the selection date on the suitability/feasibility of the project and the work to be done. As with the earlier programming assignments, you may work together with a partner and submit a single project. Please email your selection by Tue Nov 6.

Parallel Platforms

- Phaedra is available for shared memory (Cilk) and V100 accelerator projects.
- ITS Research Computing longleaf cluster. This requires a longleaf account. If you do not have one this needs several days of advance work. The general nodes are like Phaedra but do not have Phi accelerators. There are some GPU nodes with GeForce GTX1080 GPUs and some nodes with four V100 GPUs. Note that to use longleaf you must use the slurm job submission system.

Project 1: Parallel quicksort using Cilk or OpenMP tasks

Implement a parallel quicksort on 64-bit double values. Parallel quicksort is easy to express in the W-T model (see the material in lecture 3 [PRAM (2)])]. You will need to use tasking to implement the nested parallelism. Your implementation should parallelize both the partitioning and the divide and conquer steps of quicksort – otherwise you will have an (expected) $O(n)$ step complexity and $O(lg n)$ average available parallelism. Present your results as a description of the implementation and a set of performance graphs that display the sorting performance on random input data sets as a function of problem size and number of processors. Show the performance (speedup) compared with an efficient sequential quicksort. Investigate and analyze any performance bottlenecks.

Project 2: k-means using Nvidia accelerator(s)

Partition a sequence $L$ of $n$ points into $k$ clusters using the Lloyd $k$-means algorithm implemented on a GPU. Each element of $L$ is a 2-tuple interpreted as a coordinate in 2-space. Use Euclidean distance as the distance metric between values in $L$. Initially pick $k$ points from $L$ and use these as the cluster centers. The algorithm repeats an assignment phase and an update phase. In the assignment phase, assign each point in $L$ to the nearest cluster center. In the update phase the cluster centers are updated to be the coordinate-wise mean of points assigned to the cluster (a.k.a. the cluster centroid). Iterate until the cluster assignment stays the same. The output should be a list of the $k$ centroids and a permutation of $L$ partitioned into $k$ segments corresponding to the values assigned to each cluster. Identify the number of points in each segment.
Project 3: Your choice.

This project should be related to your research or other interests and must present a non-trivial parallelization problem. You should talk to me as early as possible to discuss the suitability of the project.