Parallel Computing
COMP 633  Fall 2018

Programming Assignment PA2

Assigned: Thu Oct 25
Project selected: Tue Nov 6
Due: Sat Dec 13, 6PM (earlier submission encouraged!)

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 in the divide and conquer step. In addition you should investigate parallelizing the partitioning step of quicksort – otherwise you will have an (expected) $O(n)$ step complexity with only $O(lg n)$ 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 and specific input data sets (e.g. sorted, reverse sorted) as a function of problem size and number of processors. For expedience use {1, 4, 8, 12, 16, 20} processors. Show the performance (speedup) compared with an efficient sequential quicksort. Investigate and describe 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 phaedra’s Nvidia V100 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 $L$.

The algorithm repeats an assignment phase and an update phase until it reaches a fixpoint in the assignment. In the assignment phase, assign each point in $L$ to the nearest of the $k$ clusters. In the update phase the cluster centers are repositioned to 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 points assigned to each cluster. Identify the number of points in each segment.

For performance analysis we are interested in the computation rate $C = \frac{k \cdot n \cdot h}{t}$, where $1 \leq k \leq 1000$ is the number of clusters, $1 \leq n \leq 10^8$ is the number of 2D points, $h$ is the number of iterations until convergence, and $t$ is the elapsed time. Feel free to stop the implementation after $h_{\text{max}} = 30$ iterations instead of waiting for convergence. However, if convergence is achieved more in less than $h_{\text{max}}$ iterations, do not continue past convergence (because the computation is not representative). Show your performance results for when varying of $k$, $n$ and $h$ individually.

To obtain the elapsed time of the computation on the GPU, without the overhead of transferring data from the host to the device, and without the overhead of the transfer of the result data and the permutation of the data required for the result (which can be done on the host), you can use CUDA GPU timers described in section 8.1.2 in the CUDA C Best Practices document on our class website to time just the $knh$ work at the heart of the algorithm.

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.