Parallel Computing

COMP 633 Fall 2016

Programming Assignment PA2

Assigned:	Thu Oct 27
Project selected:	Tue Nov 1
Due:	Wed Dec 6

Problem selection

For this programming assignment choose one of the projects below or propose some alternate project of interest to you. If you choose a different project we need to agree on the suitability and the work to be done in advance. As with the earlier programming assignment, you may work together with a partner and submit a single project. Please communicate your selection to me by Tue Nov. 1 (let me know if you need an extension beyond this date to investigate an alternate project).

Parallel Platform

- Bass can be used for shared memory OpenMP jobs. It can run MPI jobs, but the configuration is untested. If you want to use MPI, I would suggest the ITS clusters.
- Phaedra is available for shared memory and Xeon Phi accelerator projects. This node has 20 CPU cores, 4 Phi accelerators, and 128 GB main memory. It also has recent Intel compilers and performance analysis tools (Intel C/C++ compilers implement Cilk and OpenMP).
- If you want to work on a GPU problem, you may use gamma-x51-1.cs.unc.edu which has a GeForce GTX 960 card (Maxwell) that is fairly recent (but not the most recent).
- ITS Research Computing facilities killdevil machine. This requires an account to be requested so needs several days of advance work. Killdevil has 750+ nodes (9500+ cores) and 32 nodes with dual Nvidia GPUs. Note that queue wait times for large parallel configurations (48+ cores) can be very long.

Project 1: Parallel quicksort using Cilk or OpenMP 3.0 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 a number of 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: Compute a list of all-pairs n-body interactions using Intel Xeon Phi accelerators

This all pairs computation is a component of optimal $O(n)$ work n-body algorithms. Assume the bodies are interacting through the gravitational potential in 2D as in the first assignment. However in this case we are given our n bodies partitioned into k groups

$$B = [B_1, B_2, \ldots, B_k] = \left[(b_1, \ldots, b_{n_1}), (b_{n_1+1}, \ldots, b_{n_2}), \ \ldots \ , (b_{(n_{k-1}+1)}, \ldots, b_{n_k})\right]$$

(where $n_k = n$) and we are given a list L of pairwise interactions between groups, e.g.

$$L = [(B_1, B_1), (B_1, B_2), (B_1, B_5), (B_2, B_1), (B_2, B_2), (B_2, B_3), (B_2, B_4), \dots, (B_k, B_7), (B_k, B_k), (B_k, B_3)]$$

In the list each group interacts with itself and a variable number of other groups. We have to determine the total force on each body due to the listed interactions, i.e. we have to compute

$$F = [F_1, \dots, F_n]$$

for the totality of all bodies. There number of bodies in each group may vary from one to many. Start by building a fast force evaluation on the Xeon Phi, and then work out a way to evaluate all groupwise interactions and sum the total force for each body. Try to extend your solution to use multiple Phi accelerators (four, in the current configuration).

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 before Nov. 1 so we can discuss the suitability of the project.