A0: Parallel Prefix in OpenMP

Due on Monday 10/23/2017, 11:59pm ET

Problem

Your task in this assignment is to implement efficient C++/OpenMP parallel prefix function `omp_scan`. We make two assumptions: first, the number of input elements is much larger than the number of available processors, second, the number of available processors is relatively small (as in the case of e.g. compute nodes in CCR). To make your job easier, the `main` function and signature of `omp_scan` are provided such that you can focus on the main functionality only.

Instructions

1. Download `A0.tar.bz2` (A0/A0.tar.bz2) that contains the backbone of the project (link (A0/A0.tar.bz2)).
2. Edit `a0.hpp` and implement `omp_scan`. You may include extra files and implement additional functionality as needed, however you are not allowed to change the signature of the `omp_scan`. Moreover, you should not edit `a0.cpp` as this is example of the engine that will be testing your code. Arguments of the `omp_scan` are as follows:
   - `T` type of the elements on which prefix will be applied.
   - `Op` type representing a binary associative operator compatible with `T`.
   - `n` number of input elements.
   - `in` pointer to the array of size `n` storing input items.
   - `out` pointer to the output array of size `n` to store output prefix.
   - `op` object representing associative operator of type `Op`. Just in case, invocation like `out[1] = op(in[0], in[1]);` applies the operator to items `in[0]` and `in[1]` and stores the result in `out[1]`.

   When implementing `omp_scan` you may assume that all arguments are correct, and there is no need to test that. Moreover, you can assume that `omp_scan` is never invoked from within an OpenMP parallel region. Finally, the project backbone provides a simple `Makefile`. You can edit it as needed for your project. However, when invoked without arguments, `Makefile` must produce executable `a0` from `a0.cpp`.
3. Once your solution is ready follow up with submission (see instructions below).

Submission

To submit your code follow these steps:

1. Make sure that `a0.hpp` contains your First, Last and UBIT names as asked for in comments.
2. Remove binary and other unrelated files from your project directory.
3. Make sure that your project directory is named `A0` and make tarball `A0.tar` out of it. Make sure that the resulting tarball indeed contains `A0` directory (i.e. untarring `A0.tar` should produce your `A0` directory).
4. Send your `A0.tar` as an attachment to jzola@buffalo.edu (mailto:jzola@buffalo.edu?subject=[IntroPDP] A0) with the subject `[IntroPDP] A0`. **It is imperative that you use exactly this and not any other subject.** If you make even a small change to the requested subject, your submission will be unrecognized.

Grading

First of all, we are currently working on integrating the CSE Autograder with CCR, but it is going slow. Hopefully, it will change soon! For now your submission will be graded automatically, but without instant feedback.

1. A machine with 64GB of RAM and 20-cores (two 10-core processors) will be used for testing.
2. `gcc` 6.x will be used for compiling.
3. Your code will be tested for scalability on large instances (up to the memory limit) and graded 0-100 points based on the correctness and speedup.
4. If your code does not compile, you will get 0 points.
5. If your code produces systematically incorrect answers, you will get 0 points.
6. **There is no late submission policy! If you do not submit on time, you will get 0 points.**

Final Remarks

1. This is a fun project, which you can showcase to potential employers. Make it well.
2. This is easy project, do not be afraid to experiment to get the best possible scalability.
3. There are plenty of lousy OpenMP prefix implementations in the Internet. I have compiled quite a nice library of those. Do not waste your time on searching. Just code...

4. I use automated plagiarism detection tools. It includes huge database of codes from the Internet and past courses I offered. In other words, if you decide to cheat I will catch you. You will get F and fail the course (see syllabus for details (../UB-IntroPDP-Syllabus.pdf)). This is not because I am mean or do not like you. This is because I value and protect hard working students who do not cheat.