Friday, 9 December 2016

MPCS 51100 HW4 Report

Yan Cui

- **P1**

I implemented the original Dijkstra Algorithm for the single source shortest path problem. In addition, I use a heap to optimize the runtime, the runtime could be reduce from O(n^2) to O(nlogn) for a single source problem.

Here’s the runtime for all source problem on the small graph:

run time of dijsktra with heap: 5.370446

run time of orginal dijsktra: 278.124519

I also tested the runtime for a single source on the high graph:

run time of dijsktra with heap: 0.045965

run time of orginal dijsktra: 38.896706

- **P2**

The parallelize version of all sources shortest path problem should have a complexity of O(n^3 / p) without a heap and O(n^2logn / p) with a heap.
1 thread for all sources on the small graph:

run time of orginal dijsktra: 317.180405

run time of dijsktra with heap: 6.224034

4 thread for all sources on the small graph:

run time of orginal dijsktra: 138.382365

run time of dijsktra with heap: 2.208150

The runtime is longer than O(n^3 / p) and O(n^2logn / p), I think the reason is that some time is spend on accessing shared variables.

- **P3**

I parallelized the single source algorithm through parallelizing the process of finding the vertex with the vertex with the min distance to the source. Each core is assigned with s subset of node, each of them compute the local minimum distance and find the global minimum distance among them.

On the small graph, the runtime for 100 single source problem is:

run time of orginal dijsktra: 5.826444

run time of parallelized dijsktra: 5.651598

The speed up is not noticeable since the time a mainly spend on accessing shared variable.

On the big graph, the runtime for 100 single source problem is:

run time of orginal dijsktra: 40.963460

run time of parallelized dijsktra: 17.554918