Programming Assignment 1- Parallel Dijkstra

Ke Chang
Yesheng Ma

Oct 20th, 2016

1 Abstract

The known Dijkstra algorithm is used to find the single-source shortest path problem. And what we have done in this project is to parallize it in order to complete the task with higher efficiency.

2 Introduction

First let's shed a light on how serial Dijkstra algorithm work. The main body is n-1 iterations. In each iteration, the algorithm selects the vectex which has the smallest distance from the origin vectex among those unknown vectexs. Then it will be put in the known list and all its adjacent vectex will be updated for next iteratoin.
The general idea behind the partition is not to completely seperate the whole algorithm, which will be too complex, but to decompose one single step- to find the min distance among the remaining vertex. During this process, we can divide the whole vertex set into n parts. And each progress finds the one with smallest distance respectively. And progress 0 does the reduce work- to find the desired one among those chosen one.

3 Implementation

3.1 Tools

C language
OpenMPI library
Slurm in Linux environment

3.2 Steps

First, we read the serial algorithm in Dijkstra.c file...
Then we begin to deploy the algorithm on the cluster. We write a slurm script

to do run it on multi-progress(1, 2, 4, 8). And we tested and recorded the time expenditure to do further evaluaton.

4 Result & Analysis

	# of input										
#prog	10	100	300	400	500	800	1000	2000	3000	4000	5000
1	140	130	150	170	170	240	250	600	1290	2210	2950
2	150	170	180	210	210	270	320	730	1290	1870	2880
4	170	160	170	180	190	230	270	710	1280	2030	2910
8	210	170	260	220	250	270	310	710	1170	1990	2870

From these results we can see two important facts: The first is that with the increasement of input, the time consumption increased but not in linear. This may be caused by the IO time, which is considerablely large in this problem. The second is that although the number of progresses increased, the time consumption stays the same. The reason besides IO time may be the most work is done by progress 0, consequently the result is not very good. So the speed-up is 1 and the efficiency is very low. All of these reflects the imperfection in the algorithm itself.