🌲 Tree Search Problem with OpenMPI

East China Normal University

10185101210 陈俊潼, 2020.6.15

简介

本程序使用 OpenMPI 库实现了旅行商问题。三个程序的算法均为循环形式的暴力搜索 DFS。项目使用 macOS 编译，CPU 型号为 Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz。

包含三个程序、一个地图生成脚本和 `makefile` 。这三个程序分别是：

- `tsp_serial.cpp` ： 串行程序
- `tsp1.cpp` ： 每个进程一旦发现新的最佳回路后，要将最佳回路的代价发送给其他所有进程。
- `tsp2.cpp` ： 每个进程有自己的的本地最佳回路数据结构，直到它完成搜索为止。当所有的进程运行完后再调用一个全局归约操作找出最小代价回路。

使用方式

- 生成图： `python generater <node count>` ，用于生成具有 node count 城市的全连接双向图。
- 编译： `make`
- 运行串行程序： `make runs` 或 `./tsp_serial.cpp <graph file>`
- 运行并行程序 1 （广播剪枝）： `make run1` 或 `mpirun -np <node count> tsp1.cpp <graph file>` ，其中 graph file 为图文件。
- 运行并行程序 2 （全局规约）： `make run2` 或 `mpirun -np <node count> tsp2.cpp <graph file>` ，其中 graph file 为图文件。

运行结果对比

为了避免运行时间过长，截图展示的数据是 12 个节点的运行结果。

串行程序：

```
./tsp_serial.out graph12.in
-------- BEST TOUR-------
0->6->2->8->9->5->7->3->1->4->10->11->0, cost = 169
Time elapsed: 29746 ms.
```

并行程序 1（广播剪枝）（部分输出已省略）：

```
<Thread 11>: --------BEST TOUR FOR THREAD 11 -------
<Thread 11>: 0->10->11->6->2->8->9->5->4->7->3->1->0, cost = 187
<Thread 8>: --------BEST TOUR FOR THREAD 8 -------
<Thread 8>: 0->7->3->6->2->8->9->5->1->4->10->11->0, cost = 213
<Thread 7>: Broadcasting shortest cost 221...
<Thread 1>: Broadcasting shortest cost 182...
<Thread 7>: Broadcasting shortest cost 213...
<Thread 2>: Broadcasting shortest cost 226...
<Thread 9>: --------BEST TOUR FOR THREAD 9 -------
<Thread 9>: 0->8->9->5->7->3->1->4->10->11->6->2->0, cost = 264
<Thread 3>: --------BEST TOUR FOR THREAD 3 -------
<Thread 3>: 0->2->8->9->5->6->7->3->1->4->10->11->0, cost = 236
<Thread 1>: Broadcasting shortest cost 180...
<Thread 5>: --------BEST TOUR FOR THREAD 5 -------
<Thread 5>: 0->4->10->11->6->2->8->9->5->7->3->1->0, cost = 227
<Thread 6>: --------BEST TOUR FOR THREAD 6 -------
<Thread 6>: 0->5->6->2->8->9->7->3->1->4->10->11->0, cost = 245
<Thread 7>: Broadcasting shortest cost 205...
<Thread 7>: Broadcasting shortest cost 171...
<Thread 2>: Broadcasting shortest cost 216...
<Thread 10>: --------BEST TOUR FOR THREAD 10 -------
<Thread 10>: 0->9->5->6->7->3->1->4->2->8->10->11->0, cost = 195
<Thread 7>: Broadcasting shortest cost 169...
<Thread 2>: Broadcasting shortest cost 205...
<Thread 7>: --------BEST TOUR FOR THREAD 7 -------
<Thread 7>: 0->6->2->8->9->5->7->3->1->4->10->11->0, cost = 169
<Thread 1>: --------BEST TOUR FOR THREAD 1 -------
<Thread 1>: 0->11->6->2->8->9->5->7->3->1->4->10->0, cost = 180
<Thread 0>: Received result from thread <1>: 180
<Thread 2>: --------BEST TOUR FOR THREAD 2 -------
<Thread 2>: 0->1->4->7->3->9->5->6->2->8->10->11->0, cost = 205
<Thread 0>: Received result from thread <2>: 205
<Thread 0>: Received result from thread <3>: 236
<Thread 0>: Received result from thread <4>: 251
<Thread 0>: Received result from thread <5>: 227
<Thread 0>: Received result from thread <6>: 245
<Thread 0>: Received result from thread <7>: 169
<Thread 0>: Received result from thread <8>: 213
<Thread 0>: Received result from thread <9>: 264
<Thread 0>: Received result from thread <10>: 195
<Thread 0>: Received result from thread <11>: 187
<Thread 0>: Shortest cost of all: 169
<Thread 0>: Time elapsed: 121 ms.
```

并行程序 2（全局规约）：

```
mpirun -np 12 tsp2.out graph12.in
Calculation started.
<Thread 5>: -------- BEST TOUR FOR THREAD 5 -------
<Thread 5>: 0->4->10->11->6->2->8->9->5->7->3->1->0, cost = 227
<Thread 9>: -------- BEST TOUR FOR THREAD 9 -------
<Thread 9>: 0->8->9->5->7->3->1->4->10->11->6->2->0, cost = 264
<Thread 1>: -------- BEST TOUR FOR THREAD 1 -------
<Thread 1>: 0->11->6->2->8->9->5->7->3->1->4->10->0, cost = 180
<Thread 0>: Received result from thread <1>: 180
<Thread 7>: -------- BEST TOUR FOR THREAD 7 -------
<Thread 7>: 0->6->2->8->9->5->7->3->1->4->10->11->0, cost = 169
<Thread 11>: -------- BEST TOUR FOR THREAD 11 -------
<Thread 11>: 0->10->11->6->2->8->9->5->4->7->3->1->0, cost = 187
<Thread 6>: -------- BEST TOUR FOR THREAD 6 -------
<Thread 6>: 0->5->6->2->8->9->7->3->1->4->10->11->0, cost = 245
<Thread 8>: -------- BEST TOUR FOR THREAD 8 -------
<Thread 8>: 0->7->3->6->2->8->9->5->1->4->10->11->0, cost = 213
<Thread 4>: -------- BEST TOUR FOR THREAD 4 -------
<Thread 4>: 0->3->1->4->10->11->6->2->8->9->5->7->0, cost = 251
<Thread 10>: -------- BEST TOUR FOR THREAD 10 -------
<Thread 10>: 0->9->5->6->7->3->1->4->2->8->10->11->0, cost = 195
<Thread 2>: -------- BEST TOUR FOR THREAD 2 -------
<Thread 2>: 0->1->4->7->3->9->5->6->2->8->10->11->0, cost = 205
<Thread 0>: Received result from thread <2>: 205
<Thread 3>: -------- BEST TOUR FOR THREAD 3 -------
<Thread 3>: 0->2->8->9->5->6->7->3->1->4->10->11->0, cost = 236
<Thread 0>: Received result from thread <3>: 236
<Thread 0>: Received result from thread <4>: 251
<Thread 0>: Received result from thread <5>: 227
<Thread 0>: Received result from thread <6>: 245
<Thread 0>: Received result from thread <7>: 169
<Thread 0>: Received result from thread <8>: 213
<Thread 0>: Received result from thread <9>: 264
<Thread 0>: Received result from thread <10>: 195
<Thread 0>: Received result from thread <11>: 187
<Thread 0>: Shortest cost of all: 169
<Thread 0>: Time elapsed: 4331 ms.
```

尝试不同的节点数，对比结果如下：

节点数	串行	并行，广播剪枝	并行，全局规约
4	0 ms	0 ms	0 ms
10	278 ms	12 ms	45 ms
12	29746 ms	121 ms	4331 ms
16	∞	18747 ms	∞