计算机体系结构课程报告

——实现Parallel BFS on Multi-Core

一、实验内容

本次大作业实现的是广度优先搜索（Breadth-First Search，BFS）算法，并利用计算机多个处理核心的计算能力对其搜索过程进行并行加速。BFS 算法通常用于解决图问题，因此这次实验的测试用例是使用BFS寻找一个节点到其他节点的最短距离。

二、实验过程

先实现普通的串行BFS，在这基础上对其改进得到并行BFS，并行BFS在单台主机上的实现可直接通过OpenMP划分节点给不同线程更新计算，而在不同主机上主要使用MPI分布式计算结构，需要划分为不同进程进行计算，因此本次实验主要分为以上三个阶段完成。

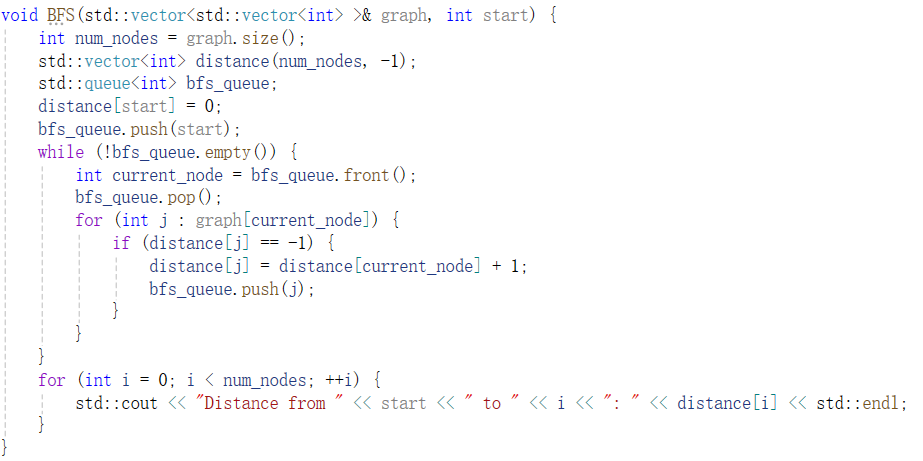
1. 串行BFS

在串行情况下，BFS 从起始节点（设定0为起点）开始，使用队列更新即将访问节点，保证深度低的节点率先更新，实现广度优先，由此逐层访问图中的节点，直到队列不再更新。

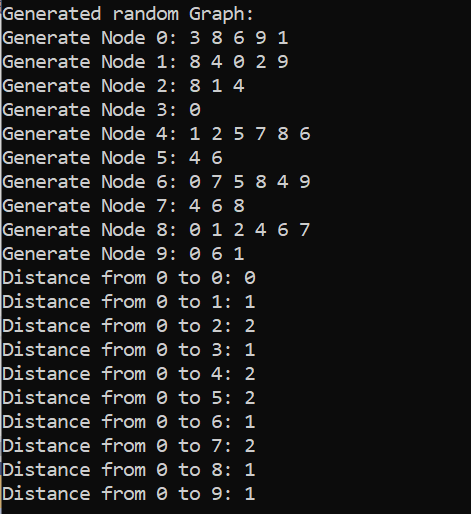
对于图的结构，使用邻接表方式进行存储。首先编写生成随机图的函数，如图，主要构建构成已经注释其中。



实现BFS函数：distance存放指定节点到所有节点的距离。初始化队列bfs\_queue放入第0个节点，通过不断取出队列中节点，遍历该节点的相邻节点并排入队列，子节点若满足未被更新的条件则可更新distance，直到队列中节点被全部取出，搜索结束。



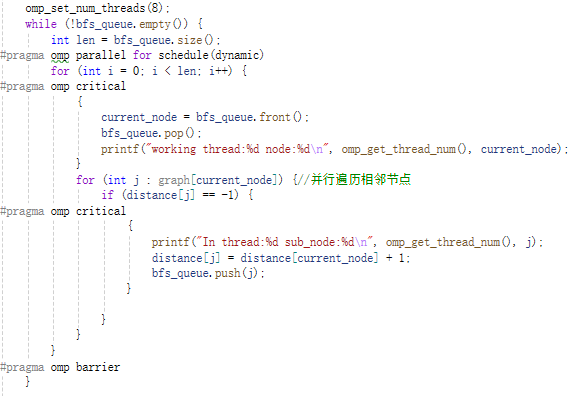
生成10个节点，节点平均邻点数为5，运行结果如下展示：



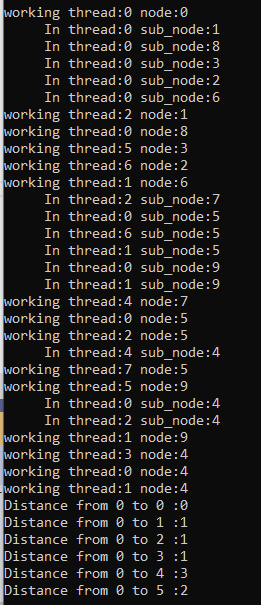
1. 并行BFS ——使用OpenMP编程

并行BFS将图的节点分配给不同的处理核心并同时处理，有助于减少计算时间，特别是对于大型图或需要快速响应的应用场景。并行的特点主要体现在一个节点在分摊搜索遍历相邻节点的过程，实现思路是将每层节点分配到并行线程，遍历搜索子节点。

实现函数在串行基础上进行改进，展示改动部分代码如下：创建OpenMP 并行区域使用一个主线程分配节点搜索更新的任务，从线程进行更新距离和队列的操作，其中创建临界区对这两个共享变量进行原子访问，防止更新冲突。也因为进行原子访问的问题，会带来一定开销，导致在图较小时反而较慢，并行的主要优化在于对for循环遍历的并行。



运行结果如下：成功调用不同线程进行广度搜索



比较串行和并行的时间：对于每种情况展示两次运行结果，其中Total time为串行BFS时间，parallel total time为并行BFS时间

10个节点 平均相邻节点有5个：

8000个节点 平均相邻节点有5000个：



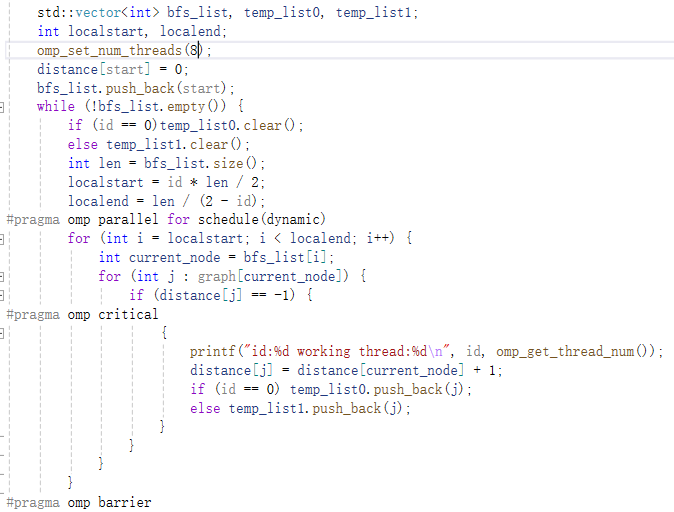
可以看出在图的规模较小的时候，因为并行BFS创建临界区的开销使得其效率低于串行BFS，而在图规模增大之后则体现出并行搜索的优势，效率优于串行搜索。

1. 并行BFS ——OpenMP MPI混合编程

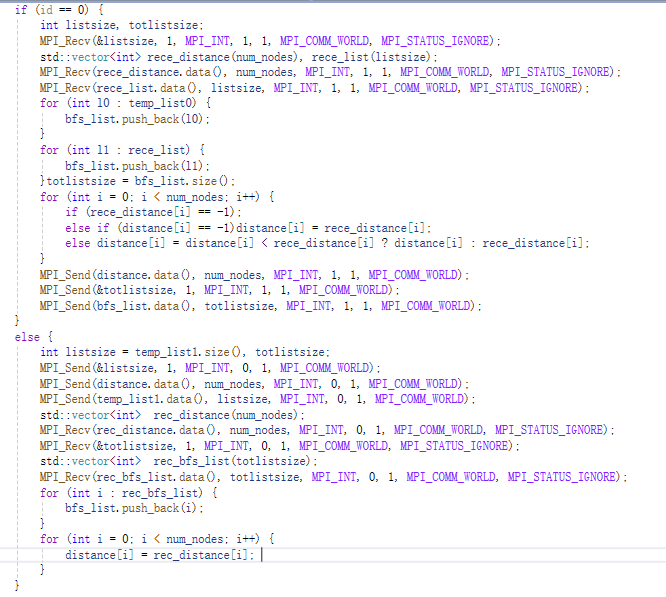
使用两台主机各8个核实现并行BFS，需要使用到MPI实现两台主机之间的分布式通信，在并行搜索期间，每更新一个深度就需要汇总各自更新的距离。两台主机的并行任务划分思路为：在每一层进行搜索时平分节点各自搜索其相邻节点，各自更新distance，各自维护一个队列，每一深度层更新结束后使用MPI通信进行汇总即可。

以下是是主要代码展示：

主要改动在于使用每台主机对应的进程id计算其掌管的节点，对其所有节点使用动态的并行线程分配。

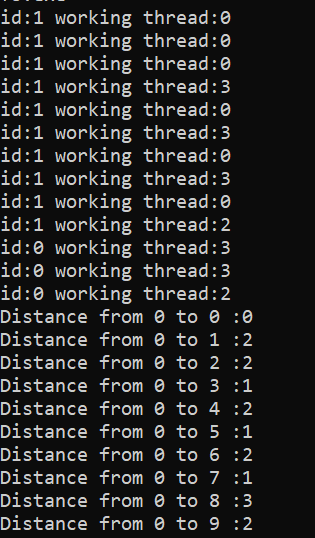
****

接下来是使用MPI\_Send和MPI\_Recv交流信息进行汇总，以下仍在上面代码while循环中，主进程(id = 0)负责接收从进程(id = 1)的distance和queue信息，与自身的更新进行汇总，再发送汇总后的distance和bfs\_queue给从进程，汇总结束加入下一层循环。



在单台主机打开两个进程：mpiexec.exe -n 2 parallel\_bfs.exe

示例如下：能够顺利调用两个进程



接下来使用两台主机进行测试，创建了两台CentOS 7的虚拟机，配置了基础的openmpi环境，并设置两台虚拟机之间的SHH免密登录。

环境配置指令参考：

[centos7配置openmpi+编程环境配置\_centos openmp-CSDN博客](https://blog.csdn.net/qq_45804132/article/details/120520642)

[在CentOS7系统上搭建MPI环境\_centos mpi-CSDN博客](https://blog.csdn.net/wangzhouf/article/details/108222704)

[双机搭建MPI - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/452337327)

两台主机执行BFS命令：

生成可运行文件：mpicc -std=c++11 -lstdc++ -fopenmp -o bfs main.cpp

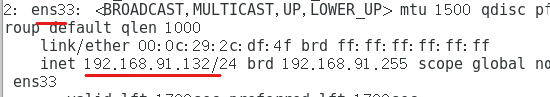
（默认是c++98，支持operator new操作，支持openmp）

单台主机运行：mpirun --allow-run-as-root -np 2 ./bfs

两台虚拟机运行需要二者配置SSH免密登录，并关闭防火墙sudo systemctl stop firewalld

两台主机运行：mpirun --allow-run-as-root -mca btl\_tcp\_if\_include ens33 -np 2 -host 192.168.91.130,192.168.91.132 /root/projects/bfs

（其中ens33来自第二台虚拟机接口）

****

配置完成后在虚拟机中进行操作：正常运行，其中因为图规模较小，只调用了一个线程



比较使用16核与8核的效率：

100个节点 平均相邻节点有50个



1000个节点 平均相邻节点有500个



10000个节点 平均相邻节点有5000个



可以看出随着图规模的扩大，运行在16核上的BFS在图规模更大的搜索中具有更好的效率，达到预期目标。

GitHub链接：<https://github.com/diffwoak/parallel_bfs>