运行介绍

学习内容：

1. 熟悉几种图算法：SSSP, PageRank。

2. 熟悉图计算引擎：接收消息（G），更新消息(G)，发送消息(F)

3. 具体代码：

先看懂了 SumInc 的这几个文件

api:

- examples/analytical_apps/pagerank/pagerank_ingress.h

- examples/analytical_apps/sssp/sssp_ingress.h

worker:

- /grape/worker/ingress_sync_iter_worker.h - 对应 PageRank 类

- /grape/worker/ingress_sync_traversal_worker.h - 对应 SSSP 类

4. 安装运行，见下面详细的介绍

SumInc 安装

运行条件：

- A modern C++ compiler compliant with C++-11 standard. (g++ = 4.8.1 and g++ < 8.0), **gcc 和 g++推荐使用 7.5** 或者 5.5，代码中使用了 cilk 并行库，该库再 gcc-8 版本开始被移除了，所以如果强制使用 gcc-8 则需要额外装 cilk 库，而在 gcc-8 之前 gcc 内含了该库。
- cmake(=2.8), [linux 下安装 cmake 手动安装](#).

- MPI(=3.0)，参考

  https://blog.csdn.net/qq_39451578/article/details/107938560，推荐第二种安装方案。
- Glog，配置参考：

  https://blog.csdn.net/qq_39451578/article/details/115679961
- Gfalgs，同上 Glog
- boost(可以通过修改 makelist，不加载此库，实际上没用到)
  - 如果安装：

boost:

sudo apt-get install libboost-all-dev

  - 如果不安装 boost, 如果运行过程中，cmake 是报错 boost 错误，可以将 cmake 文件中 boost 相关内容注释：

```
# find_package(Boost 1.65.0 COMPONENTS serialization mpi REQUIRED)


target_link_libraries(ingress grape-lite ${MPI_CXX_LIBRARIES}
        ${GLOG_LIBRARIES} ${GFLAGS_LIBRARIES} ${CMAKE_DL_LIBS}
        #-lboost_mpi -lboost_serialization
        )
```

如果还有错误看看：文档最后一节。

SumInc 编译

构建项目

- 在项目根目录下依次执行如下命令

mkdir build

cd build

cmake -DUSE_CILK=true ..

make ingress

运行项目

- ingress 运行命令：

mpirun -n 1 ../build/ingress -application pagerank -vfile /home/yusong/dataset/google/google.v -efile /home/yusong/dataset/google/0.0001/google.base -directed=1 -cilk=true -termcheck_threshold 0.000001 -app_concurrency 1 -sssp_source=16651563 -serialization_prefix /home/yusong/dataset/ser/one -out_prefix /home/yusong/result/Ingress_pagerank

- 压缩系统 SumInc-cmd：

- mpirun -n 1 ../build/ingress -application pagerank -vfile /home/yusong/dataset/google/google.v -efile /home/yusong/dataset/google/0.0001/google.base -directed=1 -cilk=true -termcheck_threshold 0.000001 -app_concurrency 1 -compress=1 -portion=1 -min_node_num=5 -max_node_num=1003 -sssp_source=16651563 -compress_concurrency=1 -build_index_concurrency=1 -compress_type=2 -serialization_prefix /home/yusong/dataset/ser/one -out_prefix /home/yusong/result/Ingress_pagerank -serialization_cmp_prefix /home/yusong/dataset/ser/spnode -message_type=push

- 相关参数解释：

-n 1: 表示启用一个进程，即单机；

-application pagerank: 表示运行 pagerank, 可以改成 sssp, php 等

-vfile: 后面接顶点文件，文件里面每行表示一个顶点 id, 需要包含整个图里面的顶点 id

-efile: 后面接边文件，文件里面每行表示一条边，例如：u v, 如果是运行 sssp 这类带权图，每条边为：u v w。

-directed=1: 表示有向图

-cilk=true: 表示启用 cilk 自动并行，默认是 grape 的线程池并行

-termcheck_threshold：设置收敛阈值，用于 PageRank 类，对于 sssp 用不上

-app_concurrency 1： 表示并行部分，启用 1 个线程进行并行，建议线程数小于或等于机器的核数

-sssp_source=16651563：指定 sssp 的源点

-serialization_prefix：后面接序列化文件存放的地址，如果在此参数，在第一运行是会将数据集读入后，在以二进制存一份(序列化文件)，存在此指定的位置，第二次使用同样参数时，直接加载此二进制文件，读取速度更快

-out_prefix：指定系统运行结果的输出位置

-min_node_num: cluser 最少包含的顶点数(仅仅算 master)，max_node_num 则相反，表示最大数来归纳

-compress：等与 0 表示不压缩，此时为 Ingress，等与 1 表示开启压缩

-portion：表示是否开启优先级调度，仅仅对 pagerank 类算法有效

-serialization_cmp_prefix：graph sketching 序列化存储位置

-message_type：消息处理方式，push/pull, 目前好像没有加 pull.

SumInc 运行

Suminc 工作流程:

- Louvain 压缩

  生成聚类文件，标记每个点属于哪一个类别；

2: 0 1

3: 2 3 4

- Suminc 进行计算
  - 先过滤，生成超点；

    判断同一个类别的点当作一个超点是否满足索引的数量少于内部边的数量，满足则当作超点；
  - 为超点生成索引；
  - 利用索引计算；

运行准备：

为了运行简单，请选择一个目录(root_path),在其下建立 code、dataset、ser、result

文件夹，dataset 下面建立 large、louvain_bin，层次关系如下：

├── code

│ ├── gemini-graph

│ ├── graphbolt

│ ├── Ingress

│ ├── libgrape-lite

│ ├── louvain

│ ├── SumInc

│ └── Test_tools

├── dataset

│ ├── large

```
│   ├── louvain_bin
├── result
│   ├── Ingress_pagerank
│   └── sum_pagerank
└── ser
```

1. 构建测试数据集

在 large 文件夹下，运行下面的代码建立 test/0.0000 文件夹

```
mkdir test
cd test
mkdir 0.0000
```

在/dataset/large/test 下建立文件 test.base, 复制内容如下：

```
0 1
0 2
0 3
1 4
1 5
1 2
2 5
2 3
3 2
3 6
4 7
```

5 7

5 8

5 9

6 2

6 8

7 4

8 9

9 11

10 9

11 14

14 12

14 13

14 15

12 10

12 13

13 15

9 16

16 17

16 18

16 19

17 20

17 21

18 21

21 18

19 21

19 22

20 21

21 23

22 23

22 24

23 24

24 25

24 26

25 27

26 25

26 27

27 25

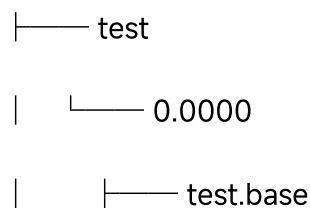
28 27

29 9

30 16

31 23

最后如下：



2. 运行 Louvain

编译 louvain,在 louvain/gen-louvain 下执行

make

将下面的文件中 root_path 修改成自己路径

./code/louvain/gen-louvain/louvain.sh

3. 运行 Suminc

编译 SumInc,在 SumInc 根目录下执行

mkdir build

cd build

cmake .. -DUSE_CILK=true

make ingress

将下面的文件中 root_path 修改成自己路径

./code/SumInc/sh/run_pr.sh

安装过程中的问题

1. 如下错误，能找到 MPI_C 不能找到 MPI_CXX，这是因为在加 Cilk 依赖时导致路径变了。解决办法：检查 cilk 库是否存在，gcc-8 及以上需要受到装 cilk 库，gcc-8 以下，自带 cilk 库。

```
option(USE_CILK "" false) # GCC 7.5.0 is required to make cilk functional
```

```
-- Found MPI_C: /mnt2/neu/software/mpich3/lib/libmpich.a (found version "3.0")
-- Could NOT find MPI_CXX (missing: MPI_CXX_WORKS)
CMake Error at /usr/share/cmake-3.16/Modules/FindPackageHandleStandardArgs.cmake:146 (message):
  Could NOT find MPI (missing: MPI_CXX_FOUND) (found version "3.0")
Call Stack (most recent call first):
  /usr/share/cmake-3.16/Modules/FindPackageHandleStandardArgs.cmake:393 (_FPHSA_FAILURE_MESSAGE)
  /usr/share/cmake-3.16/Modules/FindMPI.cmake:1688 (find_package_handle_standard_args)
  CMakeLists.txt:65 (find_package)
```

建议使用 gcc-5 或者 gcc-7，（注意：如果安装完 gcc-7 或者 gcc-5 之后不希望修改环境变量的前提下，成功 cmake, 可以参照第二步，设置临时环境变量，即无须切换系统默认的 gcc 版本）可参考：

https://blog.csdn.net/qq_39451578/article/details/107402095?csdn_share_tail=%7B%22type%22%3A%22blog%22%2C%22rType%22%3A%22article%22%2C%22rId%22%3A%22107402095%22%2C%22source%22%3A%22qq_39451578%22%7D&ctrtid=pKkUP

- 此外，有时安装了 gcc-7 也可能没有 cilk 文件夹(/usr/lib/gcc/x86_64-linux-gnu/7/include/cilk), 可以从其它地方获取 cilk 文件夹，放到对应位置即可。

2. **指定运行项目的 gcc 版本**

2.1 直接在 cmake 命令中指定：

```
cmake .. -DUSE_CILK=true -DCMAKE_C_COMPILER=gcc-7 -DCMAKE_CXX_COMPILER=g++-7
```

2.2 运行 cmake 前，人为指定临时环境变量

```
export CC=/usr/bin/gcc-7  # 找到可以执行文件，查找命令：whereis gcc
export CXX=/usr/bin/g++-7
cmake .. -DUSE_CILK=true
make
```

2.3 直接在 makelist 中指定好：

注意：需要 cmake_minimum_required()之后，project()之前！！！

```
cmake_minimum_required(VERSION 2.8)


# Need to use the specified gcc-7 version
set (CMAKE_C_COMPILER "/usr/bin/gcc-7")
set (CMAKE_CXX_COMPILER "/usr/bin/g++-7")
message("Note: Artificially specify the GCC version!")


project(AutoInc C CXX)
add_definitions(-w) # close warning
```