umap-apps Documentation

Release 0.0.3

Marty McFadden

Apr 11, 2023

BASICS

This repository contains applications that use the umap library to manage large regions of memory for them.

- Take a look at our Getting Started guide for all you need to get up and running.

CHAPTER

ONE

GETTING STARTED

1.1 Dependencies

At a minimum, the programs under umap-apps depend cmake 3.5.1 or greater and upon umap being installed.

Follow the instructions for installing umap from the umap repository located here.

1.1.1 Umap-apps Build and Installation

Once the prerequisites are taken care of, the following lines should get you up and running:

```
$ git clone https://github.com/LLNL/umap-apps.git
$ mkdir build && cd build
$ cmake -DCMAKE_INSTALL_PREFIX=<path to install umap-apps> -DUMAP_INSTALL_PATH=<path to␣
↪where umap is installed> ..
$ make -j
```

By default, umap-apps will build a Release type build and will use the system defined directories for installation. To specify different build types or specify alternate installation paths, see the *Advanced Configuration*.

Should you wish to also install the applications in their respective installation directories, type:

```
$ make install
```

Umap-apps installs files to the `lib`, `include` and `bin` directories of the `CMAKE_INSTALL_PREFIX`.

CHAPTER

TWO

ADVANCED CONFIGURATION

Listed below are the umap-specific options which may be used when configuring your build directory with cmake. Some CMake-specific options have also been added to show how to make additional changes to the build configuration.

```
cmake -DUMAP_INSTALL_PATH="<path to where umap is installed>"
```

Here is a summary of the configuration options, their default value, and meaning:

Variable	Default	Meaning
UMAP_INSTALL_PATH	not set	Location of umap
CFITS_LIBRARY_PATH	not set	Location of cfitsio library
CFITS_INCLUDE_PATH	not set	Location of cfitsio include files
CMAKE_CXX_COMPILER	not set	C++ compiler to use
DCMAKE_CC_COMPILER	not set	C compiler to use

These arguments are explained in more detail below:

- UMAP_INSTALL_PATH Location of prerequisite umap installation.
- CFITS_INCLUDE_PATH and CFITS_LIBRARY_PATH If these are specified, then the applications that use FITS files as the backing store for umap() will be built.

CHAPTER

THREE

BFS

BFS

Generate Edge List Using an R-MAT Generator

``` ./rmat_edge_generator/generate_edge_list

-o [out edge list file name (required)]

-s [seed for random number generator; default is 123]

-v [SCALE; The logarithm base two of the number of vertices; default is 17]

-e [#of edges; default is 2^{17} x 16]

-a [initiator parameter A; default is 0.57]

-b [initiator parameter B; default is 0.19]

-c [initiator parameter C; default is 0.19]

-r [if true, scrambles edge IDs; default is true]

-u [if true, generates edges for both directions; default is true] ```

- As for the initiator parameters, see [Graph500, 3.2 Detailed Text Description](https://graph500.org/?page_id=12#sec-3_2) for more details.

### Generate Graph 500 Inputs

`bash ./rmat_edge_generator/generate_edge_list -o /mnt/ssd/edge_list -v 20 -e $((2**20*16)) ``

- This command generates a edge list file (/mnt/ssd/edge_list) which contains the edges of a SCALE 20 graph.
- In Graph 500, the number of edges of a graph is #vertices x 16 (16 is called 'edge factor') as an undirected graph.
- Note that #edges generated by this generator is #vertices x 16 x 2 if -u option (generates edges for both directions) is true, which is default.
- This is a multi-threads (OpenMP) program. You can control the number of threads using the environment variable OMP_NUM_THREADS.

## Ingest Edge List (construct CSR graph)

`bash ./ingest_edge_list -g /mnt/ssd/csr_graph_file /mnt/ssd/edge_list1 /mnt/ssd/edge_list2 `

- Load edge data from files /mnt/ssd/edge_list1 and /mnt/ssd/edge_list2 (you can specify an arbitrary number of files). A CSR graph is constructed in /mnt/ssd/csr_graph_file.
- Each line of input files must be a pair of source and destination vertex IDs (unsigned 64bit number).
```

- This program treats inputs as a directed graph, that is, it does not ingest edges for both directions.
- This is a multi-threads (OpenMP) program. You can control the number of threads using the environment variable OMP_NUM_THREADS.
- As for real-world datasets, [SNAP Datasets](http://snap.stanford.edu/data/index.html) is popular in the graph processing community. Please note that some datasets in SNAP are a little different. For example, the first line is a comment; you have to delete the line before running this program.

Run BFS

`bash ./run_bfs -n [#of vertices] -m [#of edges] -g [/path/to/graph_file] -s `

- You can get #of vertices and #of edges by running ingest_edge_list.
- If '-s' is specified, the program uses system mmap instead of umap.
- The interface to the umap runtime library configuration is controlled by environment variables, see [Umap Runtime Environment Variables](https://llnl-umap.readthedocs.io/en/develop/environment_variables.html).
- This is a multi-threads (OpenMP) program. You can control the number of threads using the environment variable OMP_NUM_THREADS. It uses the static schedule.

Tips for Running Benchmark (on large-scale) * The size of generated edge lists could be larger than the constructed CSR graph by a few times. As the rmat_edge_generator writes edges to files sequentially, you should be able to directly generate edge lists to a parallel file systems without an unreasonable execution time. * On the other hand, ingest_edge_list constructs a CSR graph causing a lot of random writes to a file mapped region (the location of the file is specified by -g option). We highly recommend that you should construct a graph on DRAM space, e.g., tmpfs, although you can still read input edge list files from a parallel file system.

Example Run BFS on a SCALE 20 R-MAT graph using 4 threads and system mmap. `bash env OMP_NUM_THREADS=4 ./rmat_edge_generator/generate_edge_list -o /mnt/ssd/edge_list -v 20 -e $((2**20*16)) env OMP_NUM_THREADS=4 ./ingest_edge_list -g /dev/shm/csr_graph_file /mnt/ssd/edge_list* mv /dev/shm/csr_graph_file /mnt/ssd/csr_graph_file sudo sync sudo echo 3 > /proc/sys/vm/drop_caches # drop page cache env OMP_NUM_THREADS=4 ./run_bfs -n $((2**20)) -m $((2**20*16*2)) -g /mnt/ssd/csr_graph_file -s `

CHAPTER

FOUR

UMAP SORT

UMap Sort

Map in a file of integers and then sort it

Example

Sort an array of 96 GB stored in data_file using 4 threads.

```
` drop_page_cache free_mem env UMAP_PAGESIZE=$umap_psize ./umapsort -f ${SSD_MNT_PATH}/
data_file -p $(((96*1024*1024*1024)/umap_psize)) -N 1 -t 4 `
```