About this document

- Version 1.0
- Last revision: 11th August 2015

Requirements

- Compiler with std c++11 support (e.g GNU C/C++ v4.8.1+)
- UNIX-like OS with CUDA 6+ support

Installation, setup and Execution

Downloading

Create an account in https://bitbucket.org Request access for bfs_multinode repository

Available repositories are: (Replace the token Your_Bitbucket_User)

- https://Your_Bitbucket_User@bitbucket.org/julianromera/bfs_multinode.git (Updated daily)
- https://Your_Bitbucket_User@bitbucket.org/jyoung3131/bfs_multinode.git (Updated weekly)

```
$ REPLACE the token Your_Bitbucket_User
$ git clone https://Your_Bitbucket_User@bitbucket.org/julianromera/bfs_multinode.git
$ git checkout -b architectural_tuning
$ git pull origin architectural_tuning
```

Setting up the environment

1. Add CUDA libraries and binaries to your path:

Add this text to your ~/.bashrc

```
UPDATE NEXT LINE
export CUDA_PATH=/usr/local/cuda-7.0
export PATH=$CUDA_PATH/bin:$PATH
export LD_LIBRARY_PATH=$CUDA_PATH/lib64:$CUDA_PATH/lib:$CUDA_PATH/extras/CUPT
```

1. In case of using EXTOLL add these lines to your ~/.bashrc

```
Adjust next line to your actual path
export EXTOLL_HOME=/extoll2
if [ -d /extoll2 ]; then
      source $EXTOLL_HOME/extoll_env.bash
      # ensure this path is correct
      export MPI_PATH=$EXTOLL_HOME/mpi/openmpi-1.6.4
      export PATH=$MPI_PATH/bin:$PATH

      export LD_LIBRARY_PATH=$EXTOLL_HOME/lib:$MPI_PATH/lib:$LD_LIBRARY_PATH
      ulimit -l 4194304
      ulimit -s unlimited
fi
```

Compiling

The code to compile is in the folder `cpu_2d/`

```
$ ssh creek01
$ cd bfs_multinode/cpu_2d
$ make -f Makefile.gcc
$ cd ../eval
```

Test Scenarios

Test scenarios are in the folder `eval/`

SBatch relevant Test-cases in the previous list are:

- o4p2n-roptim.rsh [SCALE_FACTOR]
- o4p2n-roptim.rsh [SCALE_FACTOR]
- o4p2n-roptim.rsh [SCALE_FACTOR]
- o9p8n.rsh [SCALE_FACTOR]
- o16p8n.rsh [SCALE_FACTOR]

Run a Test

Ensure that the Slurm queue is empty

```
$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
```

Use Sbatch to execute the .rsh file. Add a numeric argument at the end (Sf)

```
$ sbatch o4p2n_roptim.rsh 21
Submitted batch job 423
$ sbatch o4p2n_coptim.rsh 21vv
Submitted batch job 424
$ sbatch o4p2n_noptim.rsh 21
Submitted batch job 425
```

Visualizing Results

Generate R-code

Scripts are located under the `scripts/` folder

```
$ cd bfs_multinode
$ # Make a symlink from eval to the script file
$ cd eval
$ ln -s ../scripts/r-ify.sh r-ify.sh
$ chmod u+x r-ify.sh
$ ./r-ify.sh 423 424 425

-> File slurm-423.out. Validation passed (Tasks: 4, GPUS/Task: 1, Scale Factor: 21).
-> File slurm-424.out. Validation passed (Tasks: 4, GPUS/Task: 1, Scale Factor: 21).
-> File slurm-425.out. Validation passed (Tasks: 4, GPUS/Task: 1, Scale Factor: 21).
Enter new labels for the X-Axe? (y/n) [n] y
Enter a total 3 label(s) between quoutes. Separate them with spaces: "4p2n-Roptim" "4p2n-Cop
-> Created file "file-423-424-425.r".
-> R-Code successfully generated.
```

Create a graphic

Open the file `file-423-424-425.r` with your R editor and run the code.

Current Limitations

Out-Of-Memory errors and CUDA memory size limitations:

For some high Score-Factors (e.g 22 as of the day of writing this guide), the resulting Slurm trace will be:

```
Using SCALE-FACTOR 22
Tue Jul 14 15:42:51 CEST 2015
 ========================   JOB MAP   ========================

 Data for node: creek01 Num procs: 2
        Process OMPI jobid: [1404,1] Process rank: 0
        Process OMPI jobid: [1404,1] Process rank: 2

 Data for node: creek02 Num procs: 2
        Process OMPI jobid: [1404,1] Process rank: 1
        Process OMPI jobid: [1404,1] Process rank: 3

 =============================================================
row slices: 2, column slices: 2
graph_generation:               7.749108 s
Input list of edges genereted.
6.710886e+07 edge(s) generated in 8.499692s (7.895447 Medges/s on 4 processor(s))
Adjacency Matrix setup.
2.956432e+06 edge(s) removed, because they are duplicates or self loops.
1.283049e+08 unique edge(s) processed in 18.308235s (7.008041 Medges/s on 4 processor(s
[../b40c/graph/bfs/csr_problem_2d.cuh, 697] CsrProblem cudaMalloc frontier_queues.d_values
[cuda/cuda_bfs.cu, 486] Reset error. (CUDA error 2: out of memory)
MPI_ABORT was invoked on rank 0 in communicator MPI_COMM_WORLD
with errorcode 1.
```

Validation errors for the BFS runs

Looking at your resulting Sbatch trace you may see:

"Validation: failed" / or an output trace similar to the one below:

The issue seems to be related with compiler version errors. The version of the compiler used for the CUDA code may not be met. See the Requirements section above.

```
$ cat slurm-JOBID.out
...
BFS Iteration 24: Finished in 0.111173s
max. local exp.:     0.004383s(3.942371%)
max. queue handling: 0.056909s(51.189917%)
est. rest:           0.049881s(44.867712%)
max. row com.:       0.038369s(34.513278%)
max. col com.:       0.069516s(62.529783%)
max. pred. list. red:0.007413s(6.667911%)
(3:0) Edge [26849(2273):244(244)] with wrong levels: 3 -1
Validation of iteration 24 finished in 0.022564s
Result: Invalid 524279 Edge(s) processed, 4.715888 MTeps
(0:3) Edge [244(244):26849(2273)] with wrong levels: -1 3
(0:3) Edge [244(244):29960(5384)] with wrong levels: -1 4
(3:0) Edge [29960(5384):244(244)] with wrong levels: 4 -1
BFS Iteration 25: Finished in 0.096601s
max. local exp.:     0.000905s(0.937128%)
max. queue handling: 0.048780s(50.496083%)
est. rest:           0.046916s(48.566789%)
max. row com.:       0.032854s(34.009833%)
max. col com.:       0.054057s(55.958921%)
max. pred. list. red:0.009008s(9.324883%)
Validation of iteration 25 finished in 0.020493s
Result: Valid 524280 Edge(s) processed, 5.427272 MTeps
invalid_level
invalid_level
...
```

Troubleshooting

- Problem: In the .out file of Slurm/ Sbatch execution I get the text:

  ```
  S=C=A=N: Abort: No SCOREP instrumentation found in target ../cpu_2d/g500
  ```

- Solution: The instrumentation is activated for the runtime execution (i.e: the binary is being run prefixed with scalasca). Disable it with:

  ```
  $ export G500_ENABLE_RUNTIME_SCALASCA=no
  ```

Profiling and Tracing

Install Score-P, scalasca and CUBE locally

```
cp scripts/scorep_install.sh ~
cd ~
chmod u+x scorep_install.sh
./scorep_install.sh
...
```

Update .bashrc's variables

```
$ cat >> ~/.bashrc << EOF

export LD_LIBRARY_PATH=$HOME/cube/lib:$LD_LIBRARY_PATH
export PATH=$HOME/cube/bin:$PATH

export LD_LIBRARY_PATH=$HOME/scorep/lib:$LD_LIBRARY_PATH
export PATH=$HOME/scorep/bin:$PATH

export LD_LIBRARY_PATH=$HOME/scalasca/lib:$LD_LIBRARY_PATH
export PATH=$HOME/scalasca/bin:$PATH

export MPI_PATH=/home/jromera/openmpi

export PATH=$MPI_PATH/bin:$CUDA_PATH/bin:$PATH
export LD_LIBRARY_PATH=$MPI_PATH/lib:$CUDA_PATHo/lib64:$CUDA_PATHo/lib64/stubs:$

export LD_LIBRARY_PATH=$HOME/scorep/lib:$LD_LIBRARY_PATH
export PATH=$HOME/scorep/bin:$PATH

export SCOREP_CUDA_BUFFER=48M
export SCOREP_CUDA_ENABLE=no
export SCOREP_ENABLE_PROFILING=true
export SCOREP_ENABLE_TRACING=false
export SCOREP_PROFILING_FORMAT=CUBE4
export SCOREP_TOTAL_MEMORY=12M
export SCOREP_VERBOSE=no
export SCOREP_PROFILING_MAX_CALLPATH_DEPTH=330

export G500_ENABLE_RUNTIME_SCALASCA=yes
EOF
```

License

This code contains a subset of Duane Merrill's BC40 repository of GPU-related functions, including his BFS implementation used in the paper, Scalable Graph Traversals.

External Resources

https://www.rstudio.com/products/RStudio/

Goggle Test Framework

To install the package in ~/usr/gtest/ as shared libraries, together with sample build as well:

```
$ mkdir ~/temp
$ cd ~/temp
$ wget http://googletest.googlecode.com/files/gtest-1.7.0.zip
$ unzip gtest-1.7.0.zip
$ cd gtest-1.7.0
$ mkdir mybuild
$ cd mybuild
$ cmake -DBUILD_SHARED_LIBS=ON -Dgtest_build_samples=ON -G"Unix Makefiles" ..
$ make
$ cp -r ../include/gtest ~/usr/gtest/include/
$ cp lib*.so ~/usr/gtest/lib
```

To validate the installation, use the following test.cpp as a simple test example:

```
#include <gtest/gtest.h>
TEST(MathTest, TwoPlusTwoEqualsFour) {
    EXPECT_EQ(2 + 2, 4);
}

int main(int argc, char **argv) {
    ::testing::InitGoogleTest( &argc, argv );
    return RUN_ALL_TESTS();
}
```

To compile the test:

```
$ export GTEST_HOME=~/usr/gtest
$ export LD_LIBRARY_PATH=$GTEST_HOME/lib:$LD_LIBRARY_PATH
$ g++ -I test.cpp $GTEST_HOME/include -L $GTEST_HOME/lib -lgtest -lgtest_main -lpthrea
```