final_notebook

December 14, 2017

1 Parallel BFS performance profiling

- note on formatting: the conversion from python notebook to latex to pdf rendered some issues in image placement. The label on the image should clarify to which entry it belongs

1.1 Entry 1

1.1.1 Submission Script

In [ ]:
```
#SBATCH  -J bfs                          # Job name
#SBATCH  -p development                  # Queue (development or normal)
#SBATCH  -N 1                            # Number of nodes
#SBATCH --tasks-per-node 2               # Number of tasks per node
#SBATCH  -t 00:10:00                     # Time limit hrs:min:sec
#SBATCH  -A TG-TRA170035                 # Allocation
#SBATCH  -o bfs=%j.out                   # Standard output and error log

module use /home1/01236/tisaac/opt/modulefiles
module load petsc/cse6230-double

make test_bfs

git rev-parse HEAD

git diff-files

pwd; hostname; date

#ibrun tacc_affinity ./test_bfs
ibrun tacc_affinity hpcrun -t ./test_bfs -tests 4

date
```

1.1.2 Output

In [ ]:
```
The following have been reloaded with a version change:
  1) petsc/cse6230-single => petsc/cse6230-double
```

```
mpicc -o test_bfs.o -c -fPIC  -wd1572 -g  -g -I/home1/01236/tisaac/
opt/petsc/cse6230-double/include
-I/home1/01236/tisaac/opt/petsc/cse6230-double/include -I
/home1/01236/tisaac/opt/petsc/cse6230-double/include
`pwd`/test_bfs.c
gmake[1]: Nothing to be done for `libc'.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The version of PETSc you are using is out-of-date,
we recommend updating to the new release
 Available Version: 3.8.3   Installed Version: 3.8.1
http://www.mcs.anl.gov/petsc/download/index.html
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
mpicc -fPIC  -wd1572 -g -g -o test_bfs test_bfs.o
true test_bfs
/bin/rm -f -f test_bfs.o
2348ab557cd6754dd64f26e8716c813ebac5d551
/home1/05267/tg845668/cse6230fa17-final-kkc3/Breadth-First-Search/parallel
c455-061.stampede2.tacc.utexas.edu
Wed Dec 13 01:45:46 CST 2017
TACC:  Starting up job 508540
TACC:  Starting parallel tasks...
Running 1 tests of breadth first search
  Test 0: scale 4
    Test: 256 edges
    Test scale 4, 15 keys: Passed, average time (10 tests): 0.0103887,
            ***23102.1 parents per second***.
===Harmonic mean: 23102.1 parents per second===
TACC:  Shutdown complete. Exiting.
Wed Dec 13 01:48:59 CST 2017
```

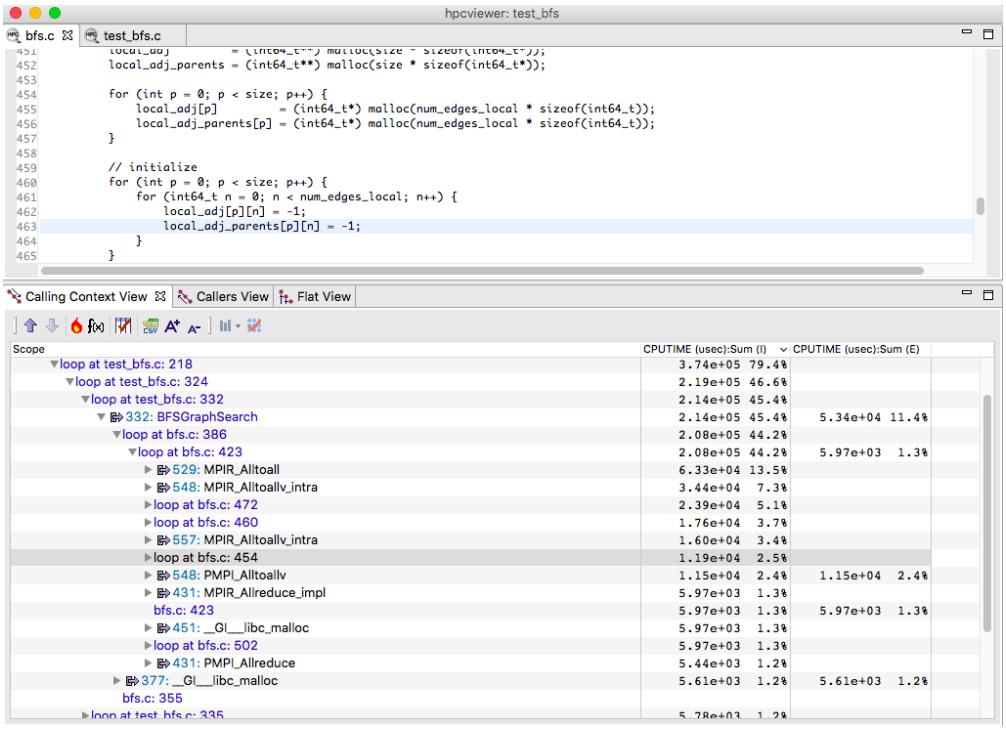
1.1.3 Profiling

In this first round of profiling using `hpcviewer`, I notice loop redundancies at line 454 and 460 when allocating and initializing `local_adj` and `local_adj_parents`. While I do not expect a significant performance boost from merging the two loops given their respective CPU time, it is useful to find and alleviate code inefficiencies so their impact on wall clock time does increase with a larger problem size or number of processes (in this case, the latter). The proposed change is thus to initialize the two arrays in the same loop in which their memory is allocated.

1.1.4 Entry 1b

The same submission script is run with the proposed changes with the results below.

```
In [ ]: mpicc -o test_bfs.o -c -fPIC  -wd1572 -g  -g -I/home1/01236/tisaac/opt
        /petsc/cse6230-double/include -I/home1/01236/tisaac/opt/petsc/cse6230-doubl
        include -I/home1/01236/tisaac/opt/petsc/cse6230-double/include
        `pwd`/test_bfs.c
```



Entry 1: hpcviewer

```
gmake[1]: Nothing to be done for `libc'.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The version of PETSc you are using is out-of-date,
we recommend updating to the new release
 Available Version: 3.8.3   Installed Version: 3.8.1
http://www.mcs.anl.gov/petsc/download/index.html
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
mpicc -fPIC  -wd1572 -g -g -o test_bfs test_bfs.o
true test_bfs
/bin/rm -f -f test_bfs.o
186da4dc43b1791fe729d8f56cfc3ad962fc104f
/home1/05267/tg845668/cse6230fa17-final-kkc3/Breadth-First-Search/parallel
c455-031.stampede2.tacc.utexas.edu
Wed Dec 13 17:33:15 CST 2017
TACC:  Starting up job 511153
TACC:  Starting parallel tasks...
Running 1 tests of breadth first search
  Test 0: scale 4
    Test: 256 edges
    Test scale 4, 15 keys: Passed, average time (10 tests): 0.0122555,
            ***19583.1 parents per second***.
===Harmonic mean: 19583.1 parents per second===
TACC:  Shutdown complete. Exiting.
Wed Dec 13 17:36:32 CST 2017
```

Under commit `9736ce218fe6d03a0d17ac2dcf4e5cfbde2ab8a6`, the wall clock time is tested locally by using `MPI_Wtime()` and executing with the command `mpiexec -n 2 ./test_bfs -tests 4` with the following result. `Average loop time: 0.000005`

Under commit `04719d18155140b4b7ae062e71f545115cb52991`, the wall clock time is tested locally by running with the command `mpiexec -n 2 ./test_bfs -tests 4` to measure the average time spent in the loop after changes were applied. `Average loop time: 0.000005`.

Although the results do not clearly support the claim that one method is faster than the other, it is also possible that the MPI wall clock time measure does not provide sufficient granularity for a meaningful comparison. It could also be the case that the loop ove the number of processes is not big enough to make a large impact when it is done twice instead of once. Had it been a loop over the number of vertices in the frontier, for example, the improvement in speed might have been more noticeable. Although the harmonic mean decreased from the first run to the second, I believe this to be due to the natural variance in the problem as opposed to a direct cause of the changes. Given more time, I would rerun this comparison on a larger problem size and number of checks to be more thorough in the comparison.

1.2 Entry 2

In this entry, the latest version of the code is run with `MPI_Wtime()` around the MPI all-to-all communication routines in `BFSGraphSearch`. The first routine is the `MPI_Allreduce` in charge of checking whether all processes are done with their searches. The second routine timed is `MPI_Alltoall` that transposes the number of adjacencies in each process. The second routine is an `MPI_Alltoallv` (denoted by Alltoallv 1 in the output) which exchanges the adjacencies to their respective owner process, and the final `MPI_Alltoallv` (denoted by Alltoallv 2) does the same but for the parents of these adjacencies. Since the third and fourth transpose operations are communicating the same number of vertices to the same processes, I would expect their timings to be very similar.

1.2.1 Submission Script

In this run, I increase the number of processes to 4 in order to gain a better perspective of work distribution and potential inefficiencies.

```
In [ ]: #SBATCH  -J bfs                          # Job name
        #SBATCH  -p development                  # Queue (development or normal)
        #SBATCH  -N 1                            # Number of nodes
        #SBATCH --tasks-per-node 4               # Number of tasks per node
        #SBATCH  -t 00:10:00                     # Time limit hrs:min:sec
        #SBATCH  -A TG-TRA170035                 # Allocation
        #SBATCH  -o bfs=%j.out                   # Standard output and error log

        module use /home1/01236/tisaac/opt/modulefiles
        module load petsc/cse6230-double

        make test_bfs
```

```
git rev-parse HEAD

git diff-files

pwd; hostname; date

#ibrun tacc_affinity ./test_bfs
ibrun tacc_affinity hpcrun -t ./test_bfs -tests 4 -num_time 1

date
```

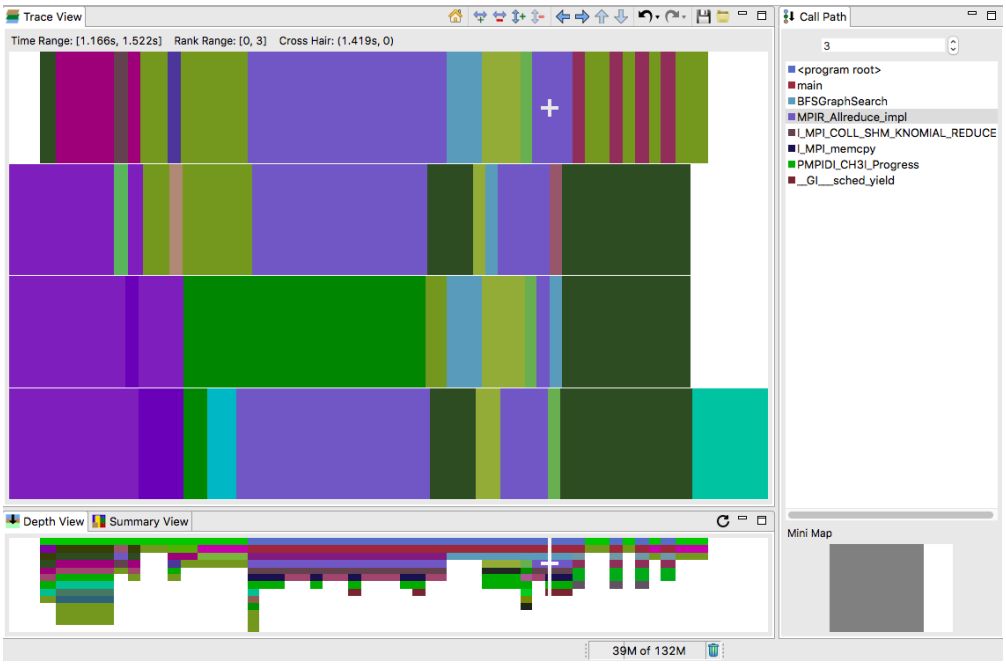
1.2.2 Output

```
In [ ]: mpicc -o test_bfs.o -c -fPIC  -wd1572 -g  -g -I/home1/01236/tisaac/opt/pets
        gmake[1]: Nothing to be done for `libc'.
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        The version of PETSc you are using is out-of-date, we recommend updating to
         Available Version: 3.8.3   Installed Version: 3.8.1
        http://www.mcs.anl.gov/petsc/download/index.html
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        mpicc -fPIC  -wd1572 -g -g -o test_bfs test_bfs.o
        true test_bfs
        /bin/rm -f -f test_bfs.o
        b81e96323cfb9439581b432448762bfcb66049c2
        /home1/05267/tg845668/cse6230fa17-final-kkc3/Breadth-First-Search/parallel
        c455-002.stampede2.tacc.utexas.edu
        Thu Dec 14 00:10:50 CST 2017
        TACC:  Starting up job 511636
        TACC:  Starting parallel tasks...
        Running 1 tests of breadth first search
          Test 0: scale 4
            Test: 256 edges
        rank: 0
        Allreduce avg time:   0.000074
        Alltoall avg time:    0.000031
        Alltoallv 1 avg time: 0.000158
        Alltoallv 2 avg time: 0.000008
        rank: 1
        Allreduce avg time:   0.000077
        Alltoall avg time:    0.000159
        Alltoallv 1 avg time: 0.000013
        Alltoallv 2 avg time: 0.000008
        rank: 2
        Allreduce avg time:   0.000063
        Alltoall avg time:    0.000161
        Alltoallv 1 avg time: 0.000013
        Alltoallv 2 avg time: 0.000008
```

```
rank: 3
Allreduce avg time:   0.000051
Alltoall avg time:    0.000014
Alltoallv 1 avg time: 0.000157
Alltoallv 2 avg time: 0.000008
rank: 0
Allreduce avg time:   0.000007
Alltoall avg time:    0.000165
Alltoallv 1 avg time: 0.000012
Alltoallv 2 avg time: 0.000009
rank: 1
Allreduce avg time:   0.000133
Alltoall avg time:    0.000013
Alltoallv 1 avg time: 0.000025
Alltoallv 2 avg time: 0.000009
rank: 2
Allreduce avg time:   0.000011
Alltoall avg time:    0.000136
Alltoallv 1 avg time: 0.000025
Alltoallv 2 avg time: 0.000009
rank: 3
Allreduce avg time:   0.000012
Alltoall avg time:    0.000149
Alltoallv 1 avg time: 0.000012
Alltoallv 2 avg time: 0.000008
    Test scale 4, 16 keys: Passed, average time
    (1 tests): 0.0137491, ***18619.4 parents per second***.
===Harmonic mean: 18619.4 parents per second===
TACC:  Shutdown complete. Exiting.
Thu Dec 14 00:14:11 CST 2017
```

1.2.3 Profiling

Looking at the `hpctraceviewer` analysis, I notice the process discrepancies of time spent in `MPI_Allreduce`. Given that this routine only communicates the value of a boolean flag (indicating whether the process is done) and reduces it with a logical and operator, I would expect this routine to be equally fast across processes. The viewer shows large variance, however. `MPI_Allreduce` is a blocking communication routine, meaning that each process will have to wait for the other processes to reach this point in order to execute. If some processes have less vertices to process, it could well be the case that they finish their search earlier, and are forced to wait for processes that are still working. To alleviate some of the waiting imabalance, I propose altering the code to implement a non-blocking routine so that processes that arrive at the reduction earlier can continue doing work instead of wasting time waiting. This can be achieved by using `MPI_Iallreduce`.



Entry 2: hpctraceviewer

1.3 Entry 3

To speed up development and testing time, I run the original and the updated BFS locally to compare the performance of the proposed change using the non-blocking routine.

1.3.1 Submission Script

```
In [ ]: make test_bfs

        git rev-parse HEAD

        git diff-files

        pwd; hostname; date

        mpiexec -n 4 ./test_bfs -tests 4 -num_time 1

        date
```

1.3.2 Output (blocking reduction)

```
In [ ]: mpicc -o test_bfs.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unkr
        `pwd`/test_bfs.c
        /usr/bin/ar cr libbfs.a bfs.o
        if test -n ""; then /usr/bin/ar cr  bfs.lo; fi
        /bin/rm -f bfs.o bfs.lo
        mpicc -Wl,-multiply_defined,suppress -Wl,-multiply_defined -Wl,suppress -Wl
```

```
/usr/bin/dsymutil test_bfs
warning: (x86_64) /Users/Karel/cs/cse6230/final/cse6230fa17-final-kkc3/Brea
/bin/rm -f -f test_bfs.o
18aafdeefbe473b14e0d69fe467079139ac050ac
/Users/Karel/cs/cse6230/final/cse6230fa17-final-kkc3/Breadth-First-Search/p
compette
Thu Dec 14 04:07:23 EST 2017
Running 1 tests of breadth first search
  Test 0: scale 4
    Test: 256 edges
rank: 0
Allreduce avg time:   0.000027
Alltoall avg time:    0.000036
Alltoallv 1 avg time: 0.000021
Alltoallv 2 avg time: 0.000021
rank: 1
Allreduce avg time:   0.000028
Alltoall avg time:    0.000027
Alltoallv 1 avg time: 0.000022
Alltoallv 2 avg time: 0.000021
rank: 2
Allreduce avg time:   0.000027
Alltoall avg time:    0.000027
Alltoallv 1 avg time: 0.000025
Alltoallv 2 avg time: 0.000021
rank: 3
Allreduce avg time:   0.000032
Alltoall avg time:    0.000024
Alltoallv 1 avg time: 0.000022
Alltoallv 2 avg time: 0.000024
rank: 0
Allreduce avg time:   0.000043
Alltoall avg time:    0.000040
Alltoallv 1 avg time: 0.000027
Alltoallv 2 avg time: 0.000026
rank: 1
Allreduce avg time:   0.000038
Alltoall avg time:    0.000043
Alltoallv 1 avg time: 0.000025
Alltoallv 2 avg time: 0.000024
rank: 2
Allreduce avg time:   0.000036
Alltoall avg time:    0.000044
Alltoallv 1 avg time: 0.000025
Alltoallv 2 avg time: 0.000025
rank: 3
Allreduce avg time:   0.000031
Alltoall avg time:    0.000039
```

```
Alltoallv 1 avg time: 0.000035
Alltoallv 2 avg time: 0.000025
    Test scale 4, 16 keys: Passed, average time
    (1 tests): 0.00950313, ***26938.5 parents per second***.
===Harmonic mean: 26938.5 parents per second===
Thu Dec 14 04:07:26 EST 2017
```

1.3.3 Output (non-blocking reduction)

In this output, `Allreduce avg time` represents the average time spent in the `Iallreduce` communication.

```
In [ ]: mpicc -o test_bfs.o -c -Wall -Wwrite-strings -Wno-strict-aliasing -Wno-unkr
        make[1]: Nothing to be done for `libc'.
        mpicc -Wl,-multiply_defined,suppress -Wl,-multiply_defined -Wl,suppress -Wl
        /usr/bin/dsymutil test_bfs
        warning: (x86_64) /Users/Karel/cs/cse6230/final/cse6230fa17-final-kkc3/Brea
        /bin/rm -f -f test_bfs.o
        85505a92c5b358577baa67e9de736412340ea7de
        /Users/Karel/cs/cse6230/final/cse6230fa17-final-kkc3/Breadth-First-Search/p
        compette
        Thu Dec 14 03:56:53 EST 2017
        Running 1 tests of breadth first search
          Test 0: scale 4
            Test: 256 edges
        rank: 0
        Allreduce avg time:   0.000005
        Alltoall avg time:    0.000084
        Alltoallv 1 avg time: 0.000056
        Alltoallv 2 avg time: 0.000043
        rank: 1
        Allreduce avg time:   0.000005
        Alltoall avg time:    0.000087
        Alltoallv 1 avg time: 0.000046
        Alltoallv 2 avg time: 0.000044
        rank: 2
        Allreduce avg time:   0.000005
        Alltoall avg time:    0.000070
        Alltoallv 1 avg time: 0.000056
        Alltoallv 2 avg time: 0.000054
        rank: 3
        Allreduce avg time:   0.000004
        Alltoall avg time:    0.000084
        Alltoallv 1 avg time: 0.000048
        Alltoallv 2 avg time: 0.000049
        rank: 0
        Allreduce avg time:   0.000005
        Alltoall avg time:    0.000058
```

```
Alltoallv 1 avg time: 0.000039
Alltoallv 2 avg time: 0.000028
rank: 1
Allreduce avg time:   0.000004
Alltoall avg time:    0.000049
Alltoallv 1 avg time: 0.000033
Alltoallv 2 avg time: 0.000037
rank: 3
Allreduce avg time:   0.000004
Alltoall avg time:    0.000053
Alltoallv 1 avg time: 0.000037
Alltoallv 2 avg time: 0.000028
rank: 2
Allreduce avg time:   0.000004
Alltoall avg time:    0.000049
Alltoallv 1 avg time: 0.000033
Alltoallv 2 avg time: 0.000039
    Test scale 4, 16 keys: Passed, average time
    (1 tests): 0.00913596, ***28021.1 parents per second***.
===Harmonic mean: 28021.1 parents per second===
Thu Dec 14 03:56:54 EST 2017
```

Interestingly, there is a noticeable difference in the average time spent in the reduction, as well as variance across processes:

	Mean Time Spent	Standard Deviation
`Allreduce`	0.00003275	0.00000543
`Iallreduce`	0.00000457	0.0000005
% decrease	86%	91%

The significant decrease in the average time spent in the communication suggest that the non-blocking behavior worked as hoped. Processes, on average, are spending less time waiting for others to catch up. This decrease in variance also helps validate this claim; processes are each spending roughly equal lengths of time in the routine. However, I am hesitant to claim that this evidence suffices to prove my initial proposition correct. For one thing, I would need to run these tests for a large number of checks (`num_time`) to approach a 'truer' average of these measurements. Secondly, I would require successful execution of the updated code to run on Stampede2 with a large number of checks. Despite my unsteady confidence in these results, a key takeaway is that improvements in work balance across processes could lie in switching blocking communications to non-blocking, where allowed.

1.4 Entry 4

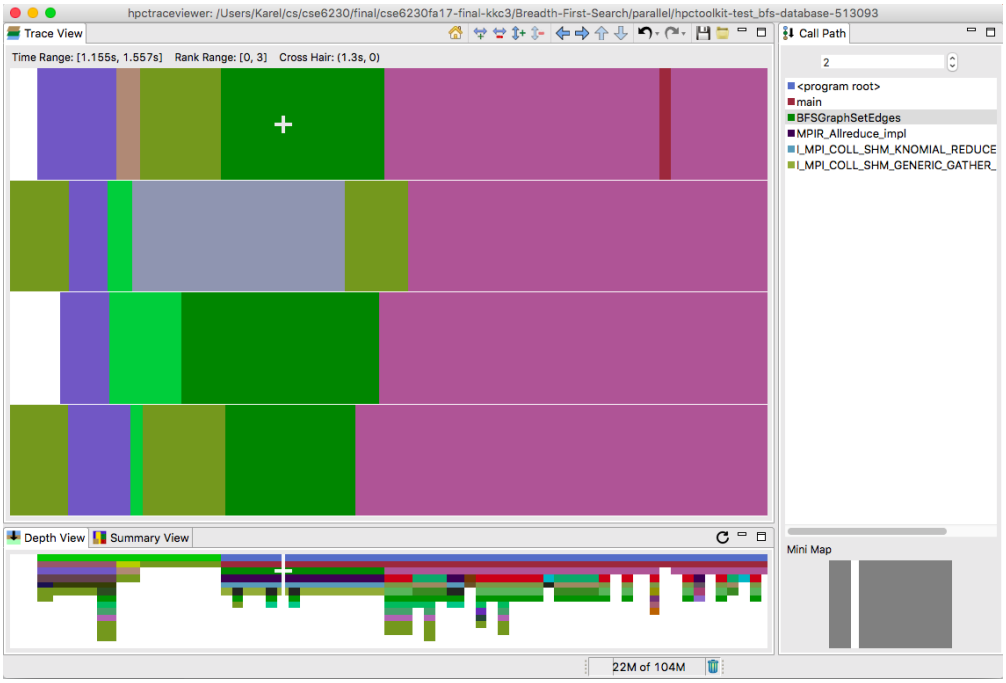
1.4.1 Submission Script

```
In [ ]: #SBATCH  -J bfs                          # Job name
        #SBATCH  -p development                  # Queue (development or normal)
        #SBATCH  -N 1                            # Number of nodes
```

```
#SBATCH --tasks-per-node 4              # Number of tasks per node
#SBATCH  -t 00:10:00                    # Time limit hrs:min:sec
#SBATCH  -A TG-TRA170035                # Allocation
#SBATCH  -o bfs=%j.out                  # Standard output and error log

module use /home1/01236/tisaac/opt/modulefiles
module load petsc/cse6230-double

make test_bfs

git rev-parse HEAD

git diff-files

pwd; hostname; date

#ibrun tacc_affinity ./test_bfs
ibrun tacc_affinity hpcrun -t ./test_bfs -tests 4 -num_time 1000

date
```

1.4.2 Output

```
In [ ]: mpicc -o test_bfs.o -c -fPIC  -wd1572 -g  -g -I/home1/01236/tisaac/
        opt/petsc/cse6230-double/include -I/home1/01236/tisaac/opt/petsc/cse6230-do
        include -I/home1/01236/tisaac/opt/petsc/cse6230-double/include    `pwd`/tes
        gmake[1]: Nothing to be done for `libc'.
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        The version of PETSc you are using is out-of-date, we recommend updating to
         Available Version: 3.8.3   Installed Version: 3.8.1
        http://www.mcs.anl.gov/petsc/download/index.html
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        mpicc -fPIC  -wd1572 -g -g -o test_bfs test_bfs.o
        true test_bfs
        /bin/rm -f -f test_bfs.o
        b9977774a23d95122a69bf985382c806c0724233
        /home1/05267/tg845668/cse6230fa17-final-kkc3/Breadth-First-Search/parallel
        c455-014.stampede2.tacc.utexas.edu
        Thu Dec 14 13:08:50 CST 2017
        TACC:  Starting up job 513093
        TACC:  Starting parallel tasks...
        Running 1 tests of breadth first search
          Test 0: scale 4
            Test: 256 edges
            Test scale 4, 16 keys: Passed, average time (1000 tests): 0.00479865,
                    ***53348.4 parents per second***.
        ===Harmonic mean: 53348.4 parents per second===
        TACC:  Shutdown complete. Exiting.
```



Entry 4: hpctraceviewer

```
Thu Dec 14 13:12:16 CST 2017
```

1.4.3 Profiling

Looking at the time spent in `BFSGraphSetEdges`, I notice that each process does not spend an equal amount of time in this graph set-up stage. When assigning edges to the different processes, I distribute the graph 1-dimensionally across the rows of the equivalent adjacency matrix so that each process owns approximately n/p vertices. This redistribution of edges according to vertex number is already an improvement in load balancing over the non-redistributing approach. However, the disparity still present could be further improved. Instead of assigning an equal sized range of vertex numbers to each process, I could instead use the total number of edges `E` to assign each process E/p edges. With this change, each process should own the same number of edges which to iterate over, thereby evening out the workload distribution. Another potential improvement would be changing the algorithm to employ a 2-D partition. Edges (u,v) would then be assigned to a mesh of pxq processes according to the ID of vertex u and of vertex v. Since vertices would be distributed across two dimensions, I would expect the average load balance to incurr less variance and the time spent in setting the edges to be more even.