CSE 6230 PROJECT2

Hansol Suh
Arjun Chintapalli

Introduction

It should be noted that we started the project individually, and later decided to work together as a group. Hansol's repository (https://bitbucket.org/dhsuh/cse6230fa17-proj2-hsuh7) has the commits for the Quicksort Pivot entry, whereas the other commits are in the submitted. It also should be noted that, it was decided to convert Arjun Chintapalli's repository as the group repository and form a pair, when Hansol faced insormountable trouble with his Sample Sort implementation. Therefore, on the final group repository, it would appear that Arjun alone did most of the git commits. Nevertheless, in order to "prove" that we worked as group, Hansol's repository will also be submitted. We tried to add Hansol's repository as a branch of the main but it was giving a variety of errors and we didn't want to risk ruining the current files.

In this report we go over the three optimizations we implemented as the entries: local quicksort pivot modification (modified qsuicksort.c and swensort.h), sample sort (modified quicksort.c), and synchronous nonblocking Isend communication instead of Ssend (modified bitonic.c).

Quicksort Pivot

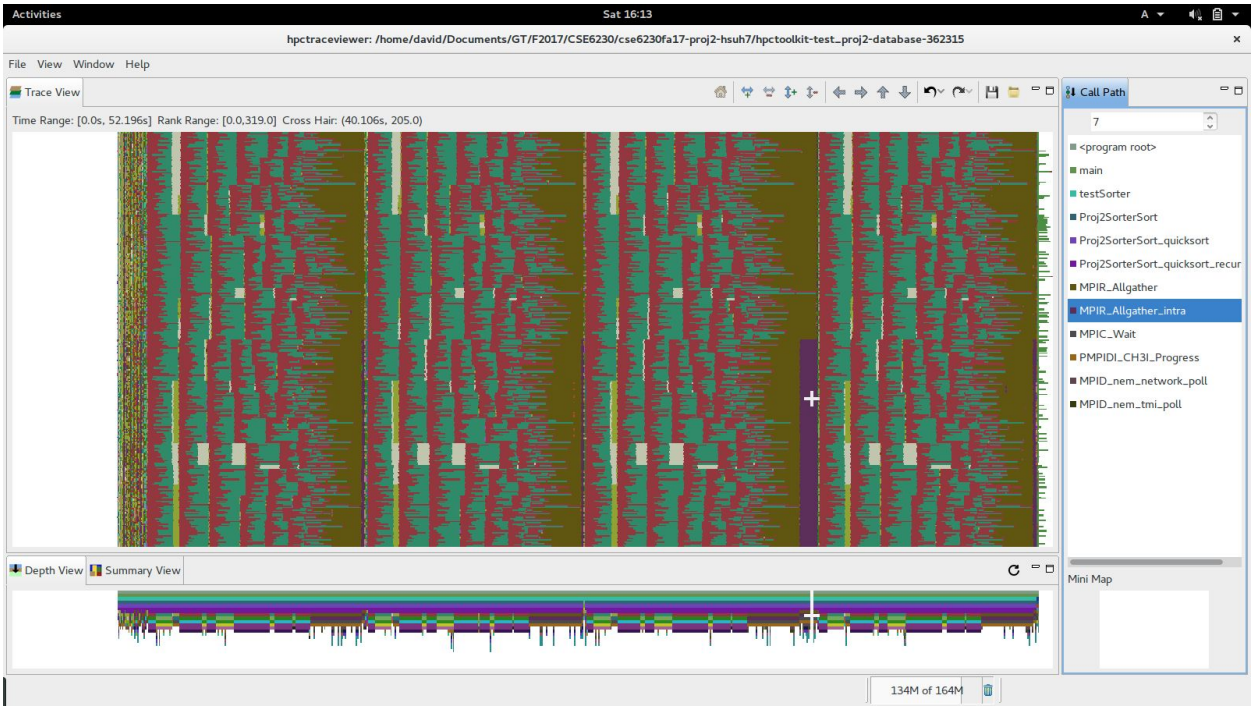

Figure 1. Original and Unaltered

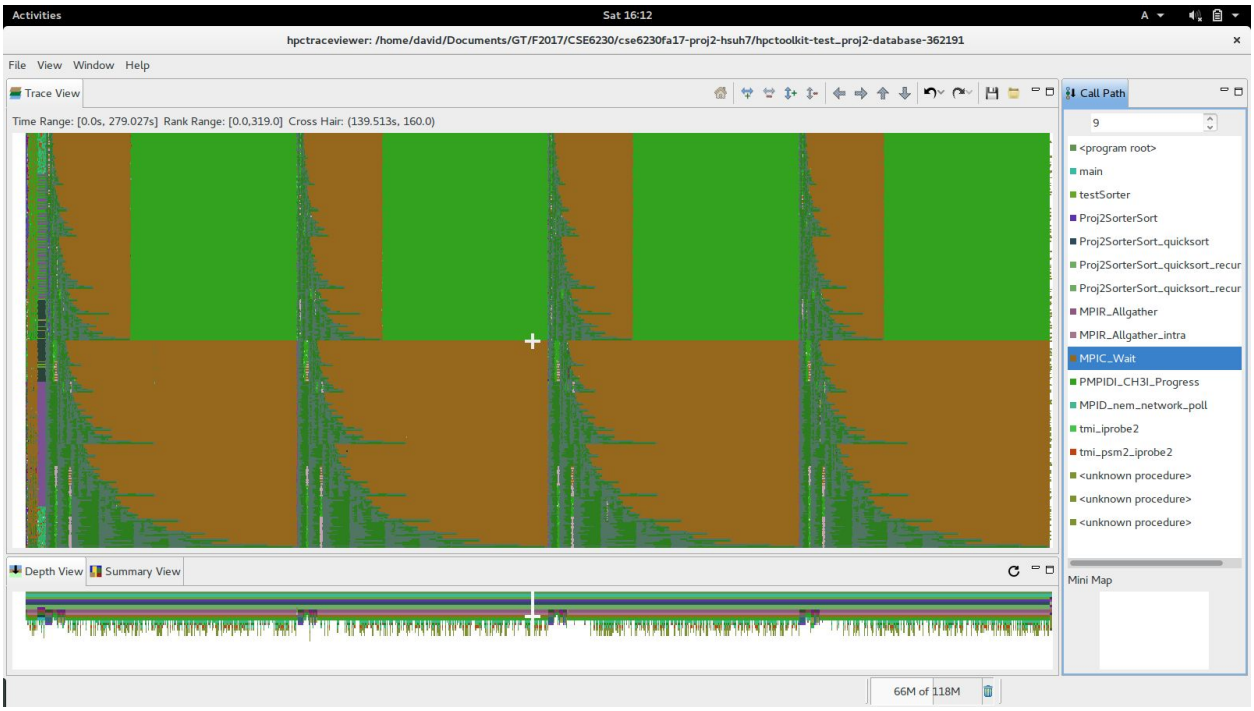

Figure 2. Changed Two Pivots

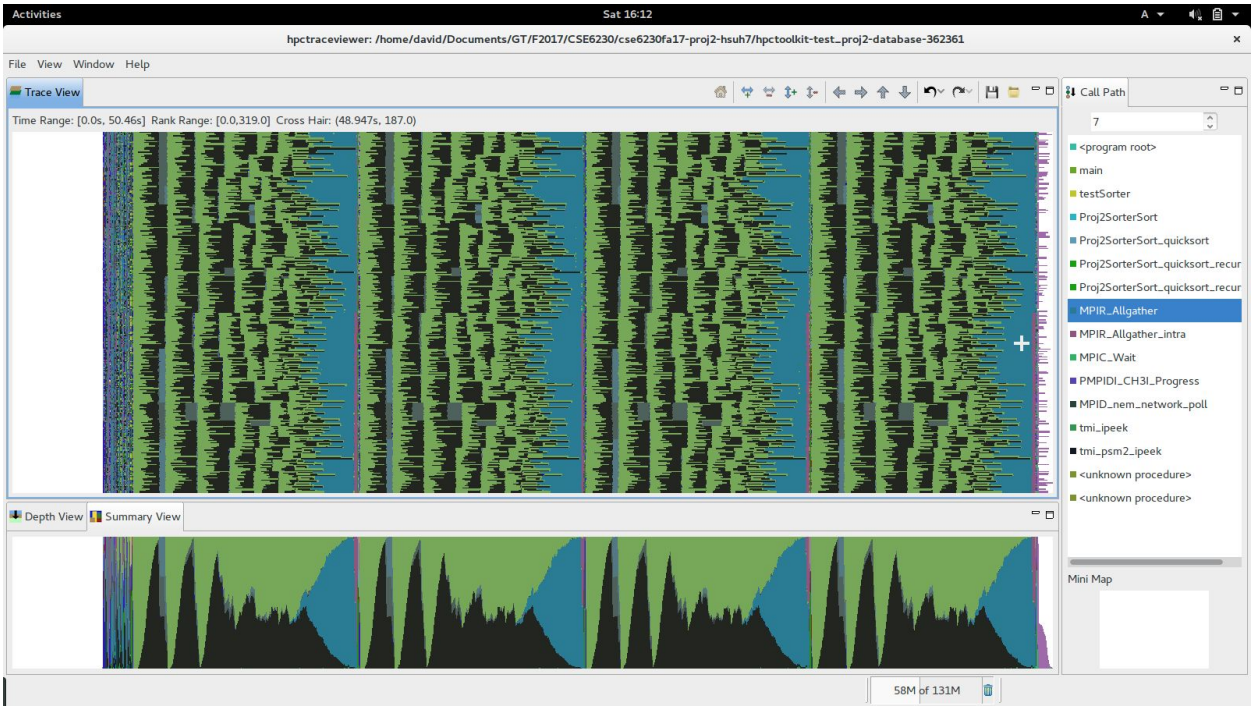

Figure 3. Changed swenson pivot option 1.

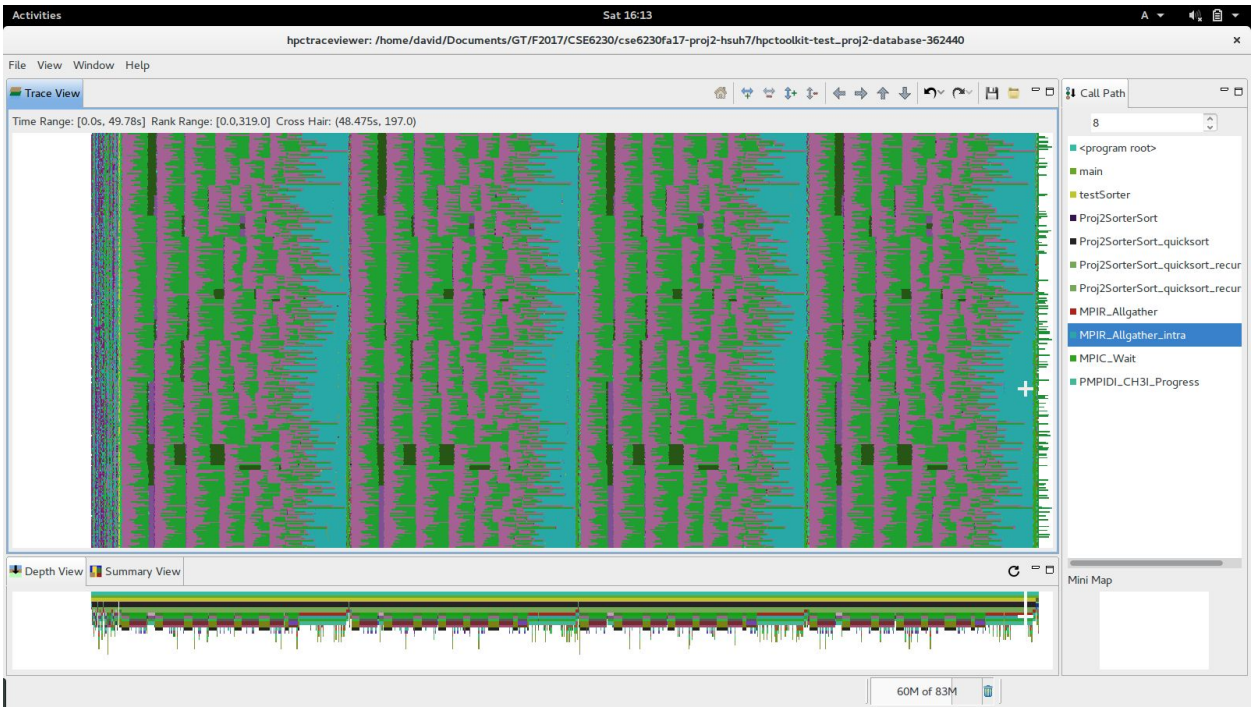

Figure 4. Changed swenson pivot option 2.

All four cases are from Stampede2, and the input was, “80, 100000, 32, 0, 4”. Number of nodes was 5, in order to investigate case for quicksort. Figure 1 is the initial output for original, unaltered file. Figure 2 is where pivot inside ChoosePivot of quicksort.c was changed from “/2” to “/3”, and where swensonsort’s sort.h’s qucisort_recursive’s pivot was changed from middle to about 0.375 from left. Figure 3 had same changes in swensort as Figure 2, but no pivot change inside quicksort.c. Finally, Figure 4 is where swensonsort’s pivot was changed to quarter, instead of original half. Following is the table for bandwidth results of four cases.

Table 1. Bandwidth Results

Cases	Bandwidth
1	5.001147e+06
2	3.448216e+06
3	5.233713e+06
4	5.740526e+06

Following are the output for four cases.

Case 1: Original

9dedfe1c98e4ce578c9b3c448dcabd861eb294cb

```
make: `test_proj2' is up to date.
/home1/05236/tg845360/cse6230fa17-proj2-hsuh7
c455-004.stampede2.tacc.utexas.edu
Sat Oct 28 21:57:23 CDT 2017
TACC:  Starting up job 371161
TACC:  Starting parallel tasks...
[0] test_proj2 minKeys 80 maxKeys 100000 mult 32 seed 0
[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800
[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 2.109964e+06
[0] Testing numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600
[0] Tested numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600: average bandwidth 3.998749e+07
[0] Testing numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200
[0] Tested numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200: average bandwidth 1.596283e+07
[0] Harmonic average bandwidth: 5.001147e+06
TACC:  Shutdown complete. Exiting.
Sat Oct 28 21:58:38 CDT 2017
```

Case 2: Pivot changes in both quicksort.c and sort.h

```
892a12f13beefbbf5c375293950f4b921f322b4f
mpicc -I../cse6230fa17-proj2-hsuh7/utils/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c
mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm
mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2
/home1/05236/tg845360/cse6230fa17-proj2-hsuh7
c455-032.stampede2.tacc.utexas.edu
Sat Oct 28 21:42:17 CDT 2017
TACC:  Starting up job 371048
TACC:  Starting parallel tasks...
[0] test_proj2 minKeys 80 maxKeys 100000 mult 32 seed 0
[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800
[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 2.227218e+06
[0] Testing numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600
[0] Tested numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600: average bandwidth 1.040593e+07
[0] Testing numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200
[0] Tested numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200: average bandwidth 3.077629e+06
[0] Harmonic average bandwidth: 3.448216e+06
TACC:  Shutdown complete. Exiting.
Sat Oct 28 21:47:23 CDT 2017
```

Case 3: PIvot change in sort.h, >>2 + >>3

```
da4c13f3786764686c364f55800f180f20262c6b
mpicc -I../cse6230fa17-proj2-hsuh7/utils/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c
mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm
mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2
/home1/05236/tg845360/cse6230fa17-proj2-hsuh7
c455-021.stampede2.tacc.utexas.edu
Sat Oct 28 21:39:09 CDT 2017
```

```
TACC:  Starting up job 371005
TACC:  Starting parallel tasks...
[0] test_proj2 minKeys 80 maxKeys 100000 mult 32 seed 0
[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800
[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 2.015564e+06
[0] Testing numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600
[0] Tested numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600: average bandwidth 4.271941e+07
[0] Testing numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200
[0] Tested numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200: average bandwidth 1.863615e+07
[0] Harmonic average bandwidth: 5.233713e+06
TACC:  Shutdown complete. Exiting.
Sat Oct 28 21:40:24 CDT 2017
```

Case 4: Pivot change in sort.h: >>2

```
afbc3b2158dd95ca4fc9fed323eb3c9ad080e181
mpicc -I../cse6230fa17-proj2-hsuh7/utils/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c
mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm
mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2
/home1/05236/tg845360/cse6230fa17-proj2-hsuh7
c455-032.stampede2.tacc.utexas.edu
Sat Oct 28 21:53:55 CDT 2017
TACC:  Starting up job 371116
TACC:  Starting parallel tasks...
[0] test_proj2 minKeys 80 maxKeys 100000 mult 32 seed 0
[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800
[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 2.241405e+06
[0] Testing numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600
[0] Tested numKeysLocal 2560, numKeysGlobal 819200, total bytes 6553600: average bandwidth 4.147513e+07
[0] Testing numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200
[0] Tested numKeysLocal 81920, numKeysGlobal 26214400, total bytes 209715200: average bandwidth 1.910564e+07
[0] Harmonic average bandwidth: 5.740526e+06
TACC:  Shutdown complete. Exiting.
Sat Oct 28 21:55:13 CDT 2017
```

Following is the rationale for each modifications. First, as it can be seen from Figure 1, big blocks of MPI_Allgather was seen at end of each iterations, especially pronounced in third iteration. As after first localsort in quicksort, each input would be in shape of localized saw-tooth, it was hypothesized that setting the pivot to be mechanical center, was sub-optimal decision. Thus, pivots in both quicksort.c and sort.h in swensonsort were modified. The former was changed from middle to be first third, and latter was changed from middle to quarter plus one-eighth. However, as it can be seen from Figure 2, it creates heavy uneven workload, with additional large MPI_Wait, with bandwidth getting halved. Thus, it was determined to remove change in pivot from quicksort.c, while leaving the pivot change in sort.h. As it can be seen from Figure 3 and Figure 4, MPI_Allgather wait block seen from Figure 1 was removed, with about 10~15% improvement in bandwidth.

It should be noted that as this process was trivial, the same batch script was used for all the tests and is as follows:

```
#!/bin/sh
#SBATCH  -J proj2                  # Job name
#SBATCH  -p development            # Queue (development or normal)
#SBATCH  -N 5                      # Number of nodes
#SBATCH --tasks-per-node 64        # Number of tasks per node
#SBATCH  -t 00:06:00               # Time limit hrs:min:sec
#SBATCH  -A TG-TRA170035           # Our allocation
#SBATCH  -o proj2-%j.out           # Standard output and error log

git rev-parse HEAD

git diff-files

make test_proj2

pwd; hostname; date

ibrun tacc_affinity hpcrun -t test_proj2 80 100000 32 0 4

date
```

It should be noted that all works on pivot were done on David's repository (https://bitbucket.org/dhsuh/cse6230fa17-proj2-hsuh7), instead of combined repository, as David and Arjun have decided to work together at later time to collaborate. Thus, David's repo was ignored after pivot part.

Sample Sort

One caveat with quicksort was that it chooses one pivot, and sorts the two subarrays recursively. However, as it chooses pivot only once, it is more prone to random deviation. Therefore, we have implemented samplesort, which is found inside quicksort.c as Proj2SorterSort_samplesort(), for sake of brevity in implementation. For each MPI processes' keys, samples with a regular interval of (# of keys)/(# of processors) were chosen, and they were gathered to processor 0. Then these splitter samples at the root processor would get sorted, and the final splitters would be chosen with the prior regular interval of a regular interval of (# of keys)/(# of processors). These final global splitters would be broadcasted to all processors, and each processor would partition its keys with the given splitters. Each processor would then send the splitted buckets to the according processor, and receive its corresponding buckets from the other processors. Finally, the key would have to be balanced so that each processor only has localNumKeys amount of keys. The default implementation within quicksort was used for this process.

This sample sort implementation, which theoretically has the same time complexity, would have less space complexity, and less stacks, as it is not recursive. Thus, sample sort works for the largest sample key size whereas quicksort crashes. This is because quicksort requires 1 + log(Processors) work arrays, so in our case this would lead to

O(9n) space complexity, whereas for sample sort the space complexity is O(2n) because we set the buffer to receive buckets from other processors as 2 *LocalNumKeys.

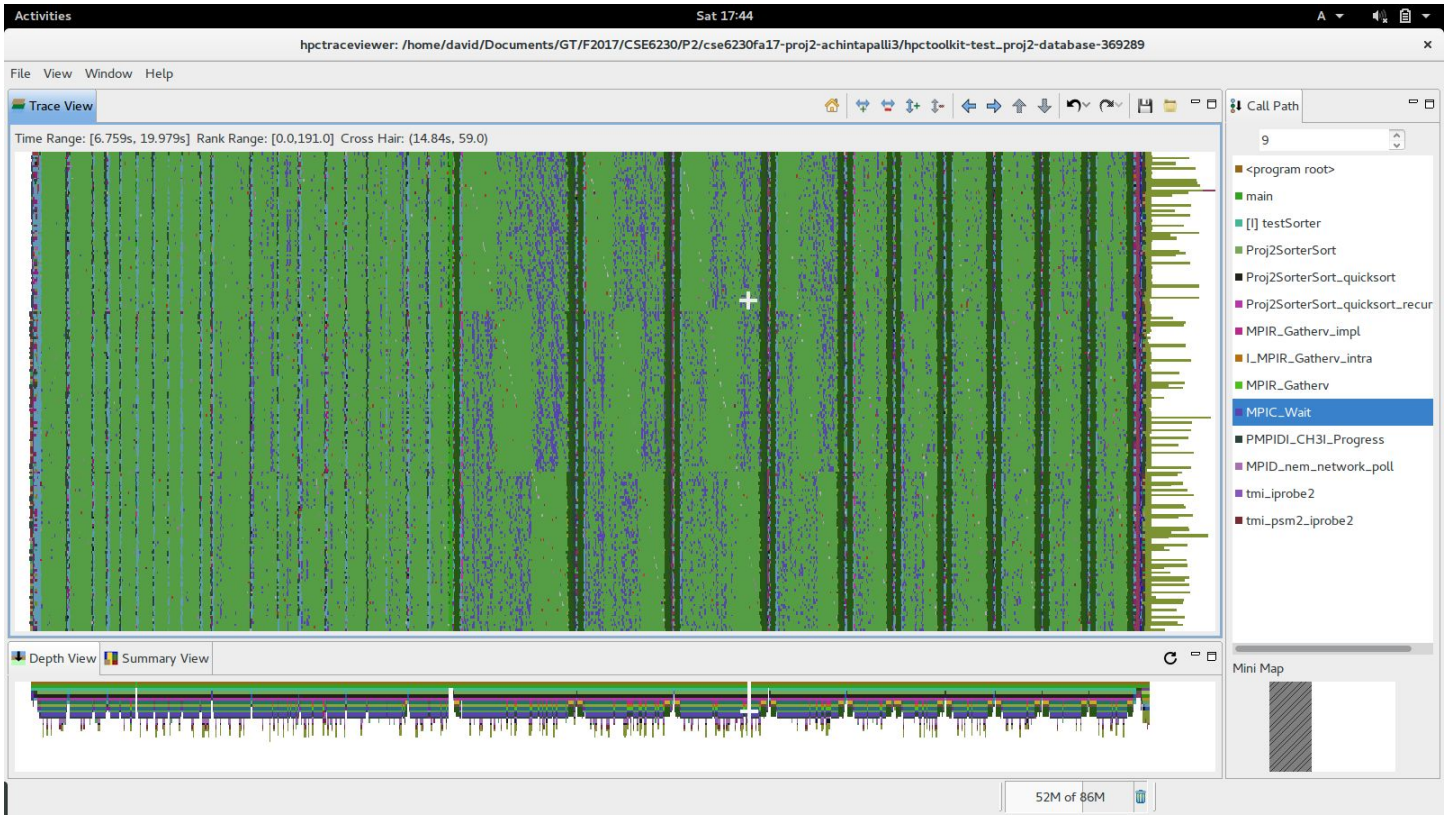

Figure 5. Sample Sort

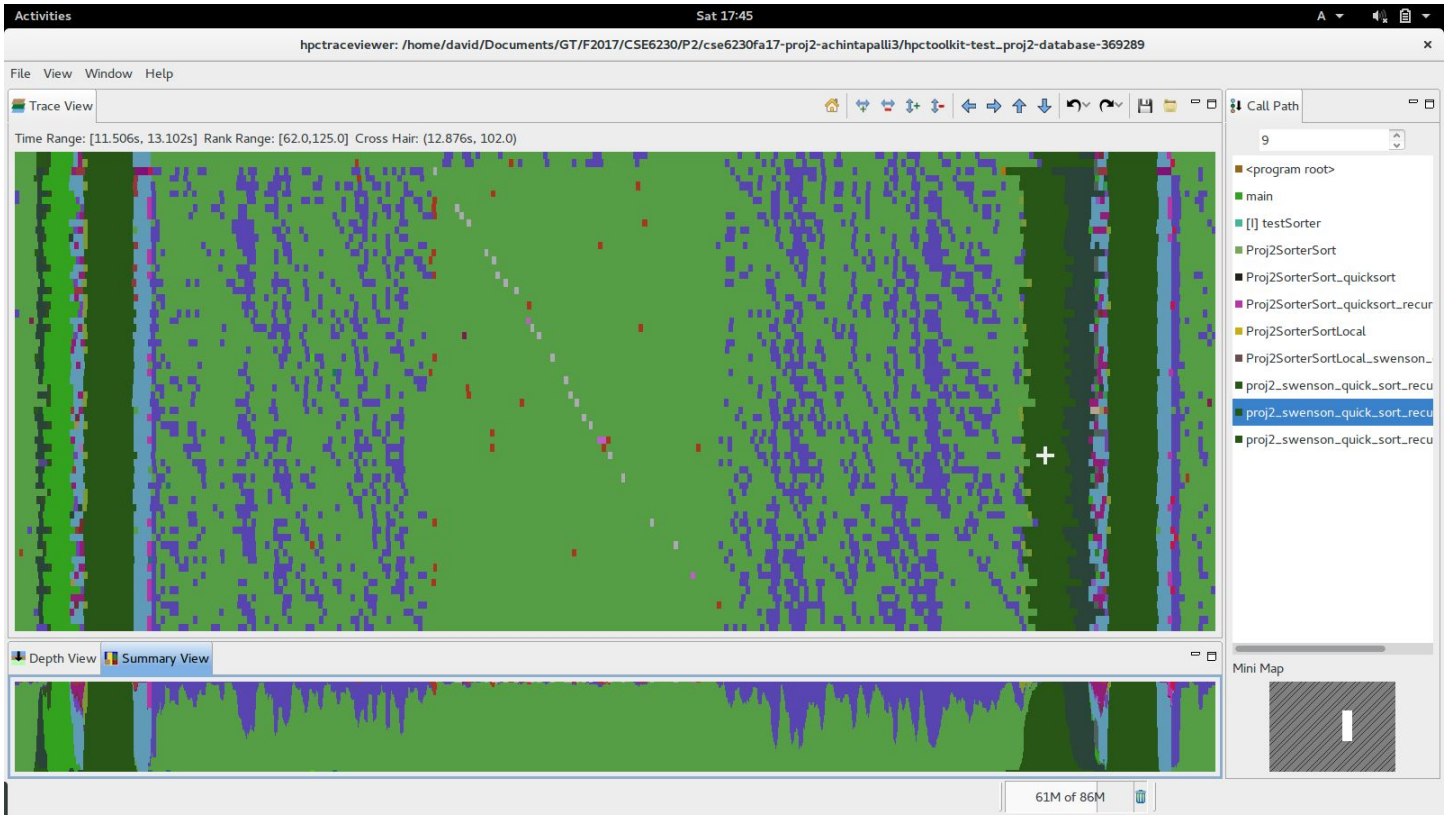

Figure 6. Sample Sort Zoom

Following is the output of all quicksort, with five nodes.

---

ac3615ba752b7c013ab6b80b0d79d9d53a0453d3

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o test_proj2.o test_proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2.o proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2sorter.o proj2sorter.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o local.o local.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o bitonic.o bitonic.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c

mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm

mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2

/home1/05236/tg845354/proj2/cse6230fa17-proj2-achintapalli3

c455-022.stampede2.tacc.utexas.edu

Sat Oct 28 20:32:46 CDT 2017

TACC: Starting up job 370600

TACC: Starting parallel tasks...

[0] test_proj2 minKeys 80 maxKeys 10000000 mult 64 seed 0

[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800

[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 1.246252e+07

[0] Testing numKeysLocal 5120, numKeysGlobal 1638400, total bytes 13107200

[0] Tested numKeysLocal 5120, numKeysGlobal 1638400, total bytes 13107200: average bandwidth 5.229182e+08

[0] Testing numKeysLocal 327680, numKeysGlobal 104857600, total bytes 838860800

[0] Tested numKeysLocal 327680, numKeysGlobal 104857600, total bytes 838860800: average bandwidth 1.014209e+09

[0] Harmonic average bandwidth: 3.608418e+07

TACC: Shutdown complete. Exiting.

Sat Oct 28 20:32:59 CDT 2017

Following is the output for all samplesort.

6531ff4537104e0ac7cf38d5c9e018c4427ba1bd

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o test_proj2.o test_proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2.o proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2sorter.o proj2sorter.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o local.o local.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o bitonic.o bitonic.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c

mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm

mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2

/home1/05236/tg845354/proj2/cse6230fa17-proj2-achintapalli3

c455-062.stampede2.tacc.utexas.edu

Sat Oct 28 20:38:56 CDT 2017

TACC: Starting up job 370631

TACC: Starting parallel tasks...

[0] test_proj2 minKeys 80 maxKeys 10000000 mult 64 seed 0

[0] Testing numKeysLocal 80, numKeysGlobal 25600, total bytes 204800

[0] Tested numKeysLocal 80, numKeysGlobal 25600, total bytes 204800: average bandwidth 2.577491e+06

[0] Testing numKeysLocal 5120, numKeysGlobal 1638400, total bytes 13107200

[0] Tested numKeysLocal 5120, numKeysGlobal 1638400, total bytes 13107200: average bandwidth 3.958957e+07

[0] Testing numKeysLocal 327680, numKeysGlobal 104857600, total bytes 838860800

[0] Tested numKeysLocal 327680, numKeysGlobal 104857600, total bytes 838860800: average bandwidth 7.902554e+08

[0] Harmonic average bandwidth: 7.237657e+06

TACC: Shutdown complete. Exiting.

Sat Oct 28 20:39:11 CDT 2017

Following is the batch script for above two outputs.

```
#!/bin/sh
#SBATCH  -J proj2               # Job name
#SBATCH  -p development         # Queue (development or normal)
#SBATCH  -N 5                   # Number of nodes
#SBATCH --tasks-per-node 64     # Number of tasks per node
#SBATCH  -t 00:07:00            # Time limit hrs:min:sec
#SBATCH  -A TG-TRA170035        # Our allocation
#SBATCH  -o proj2-%j.out        # Standard output and error log

git rev-parse HEAD

git diff-files

make test_proj2

pwd; hostname; date

ibrun tacc_affinity test_proj2 80 10000000 64 0 5

date
```

As it can be seen from Figure 5, it appears samplesort's queues looks more "plateaued" compared to quicksort. However, as it can be seen from Figure 6, sparsely populated MPI_Wait would delay the communication more so than concentrated cases for quicksort. The bandwidth for samplesort case was 7.237657e+06, clearly a lot slower than quicksort's 3.608418e+07.

Despite of samplesort's slow speed, one notable aspect is that whereas quicksort fails for the final slurm script at the biggest key size, due to large space complexity, O(9n), of the quicksort implementation; samplesort succeeds, as it has lower space complexity, O(2n) for our implementation, and lack of recursive call stacks.

The default code to distribute keys between processors to ensure the each processor has LocalNumKeys number of keys was left in place. Although the idea of using prefix summing so as to improve on the default performance was tried, a successful implementation could not be found within the the time limit.

Bitonic

Bitonic sort is the default choice when the number of nodes is even because in this case it dramatically outperforms both quick sort and sample sort. And as bitonic sort was very optimized for this project, one had to look at specific MPI calls to investigate for optimization. For the default bitonic sort, the bandwidth was 4.962060e+07.

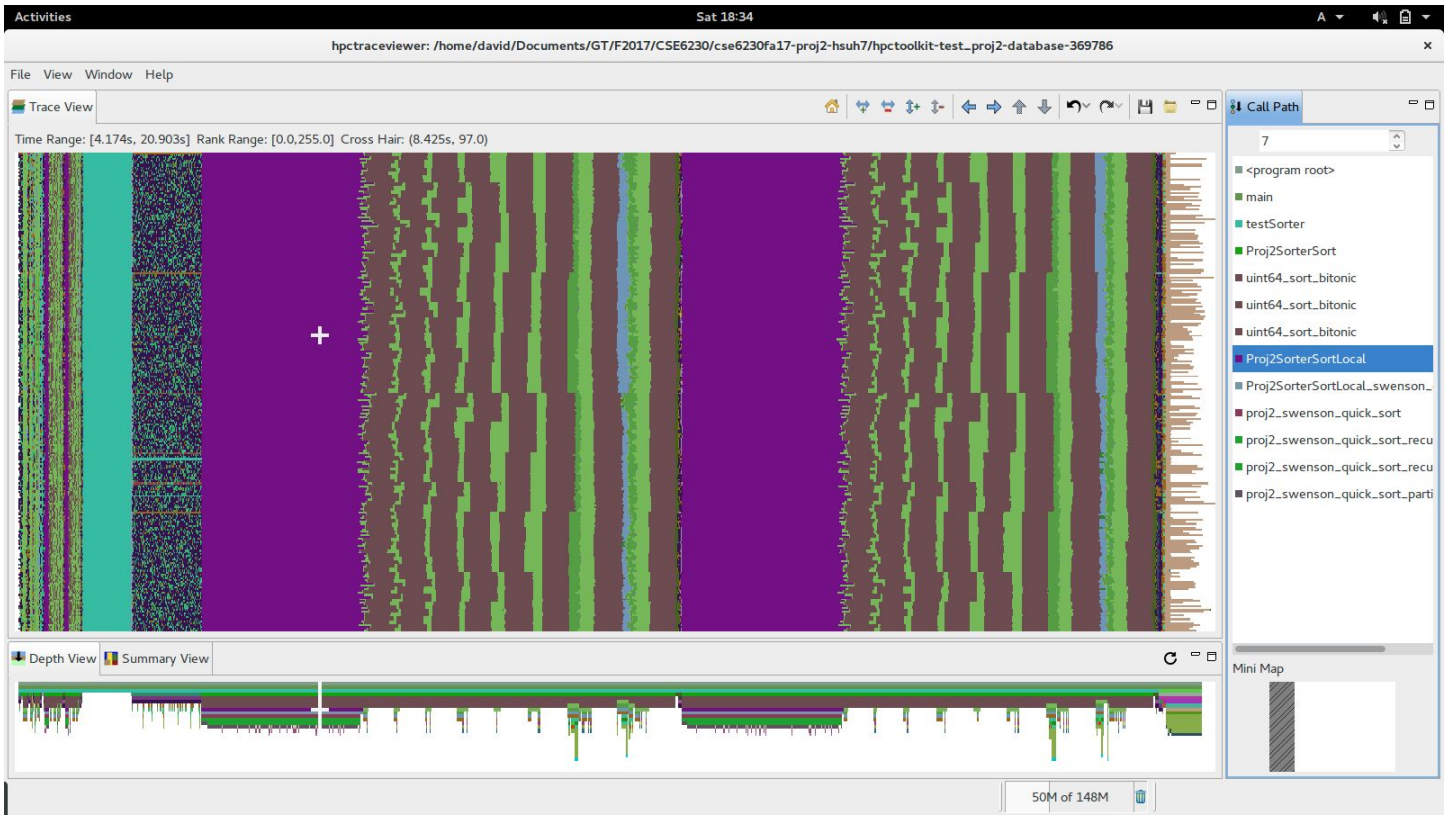

Figure 8. Overview of Default Bitonic

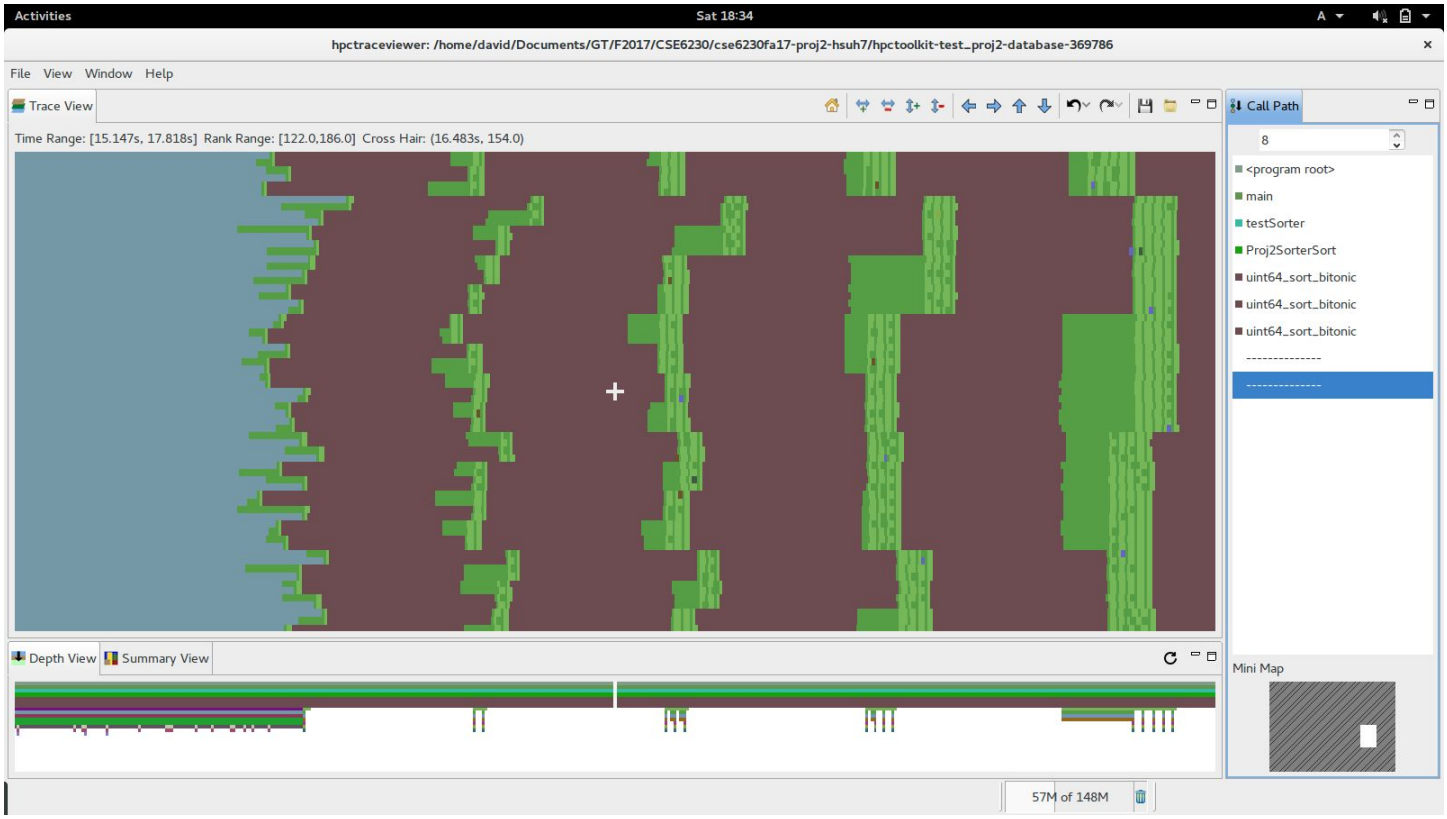

Figure 9. Closeup of Default Bitonic

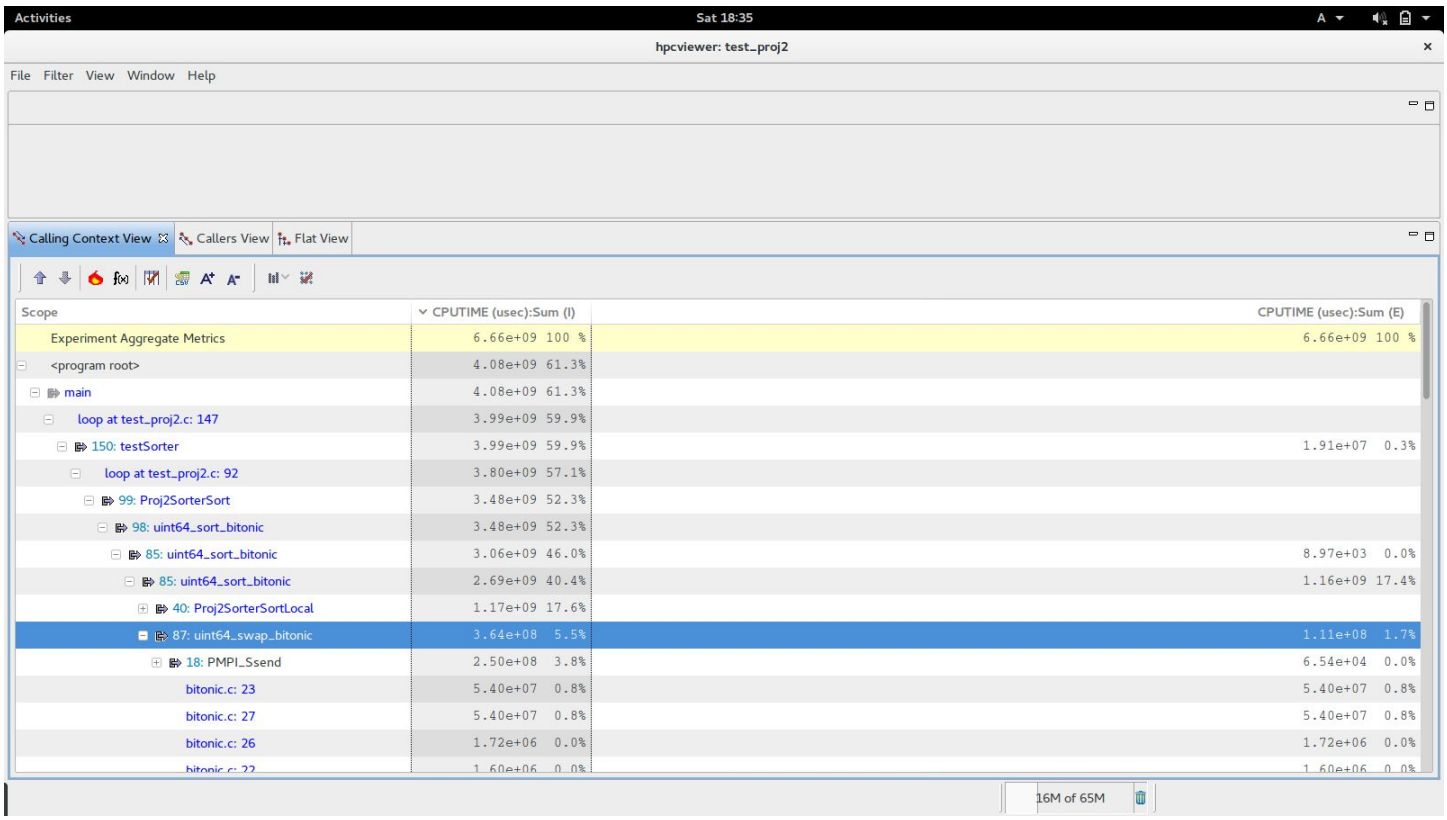

Figure 10. Hpcviewer of Default Bitonic

It can be seen that for default bitonic, Ssend is taking a fair amount of time because the MPI_Ssend will not return until the matching receive posted.

MPI_Ssend is a blocking communication that requires the completion of the communication for the function to exit, in contrast MPI_Isend is a non-blocking communication that allows the function to return immediately even if the communication isn't finished. Thus, the use of MPI_Isend for our purpose will reduce the wait time for the communication.

Therefore, in order to optimize, Ssend in **uint64 swap_bitonic()**, was changed to Isend.

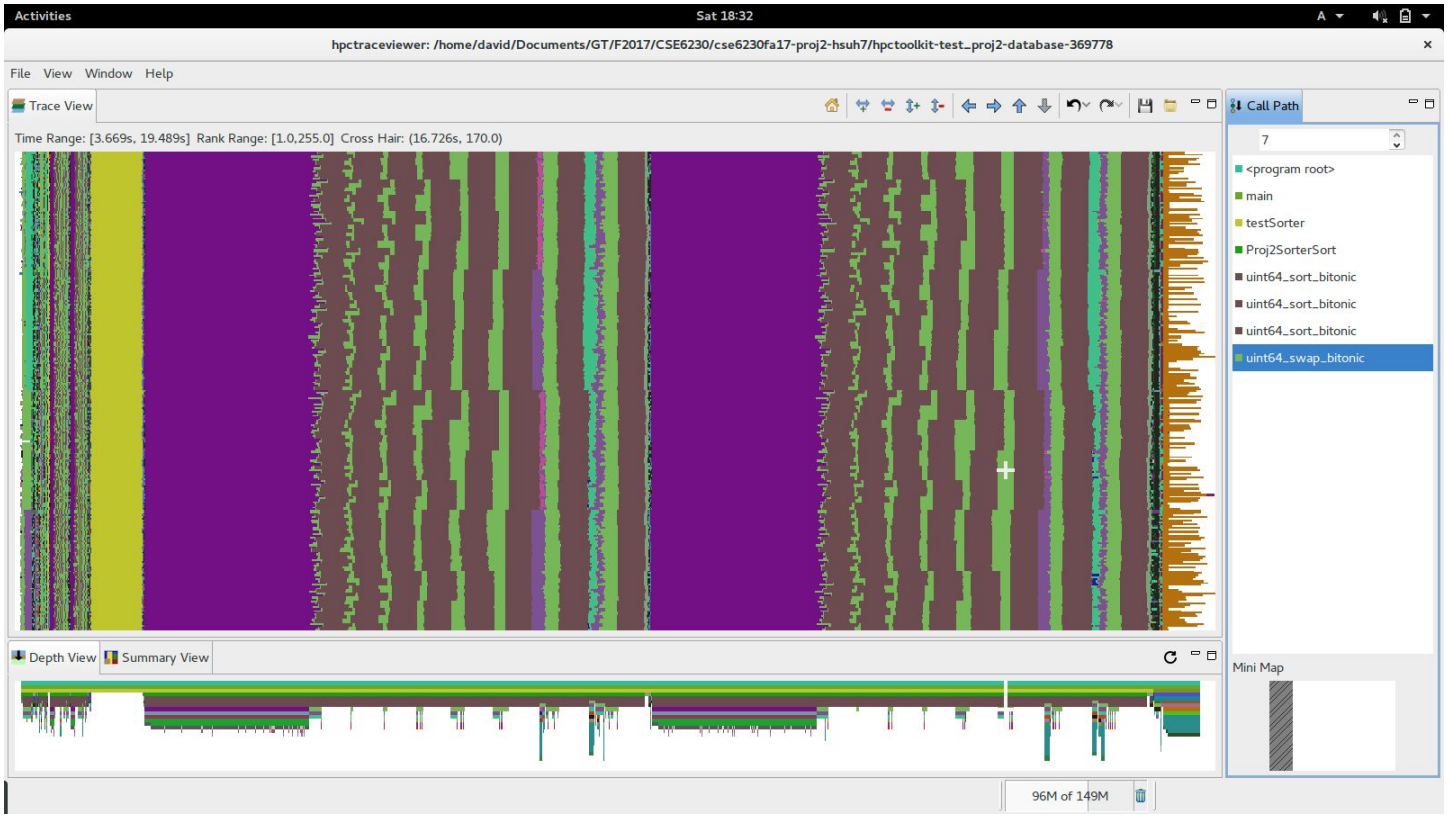

Figure 11. Isend Bitonic

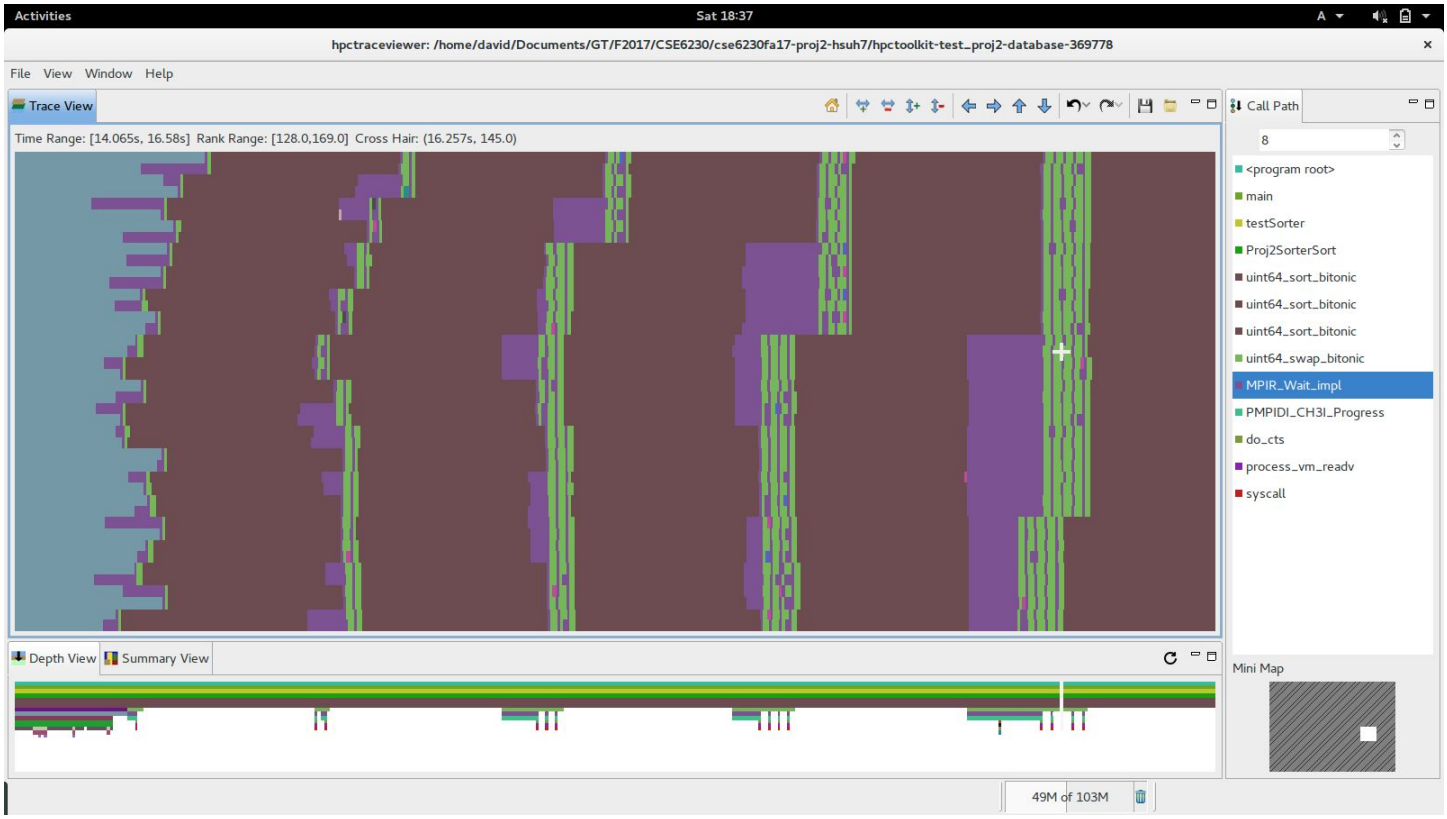

Figure 12. Isend Bitonic Closeup

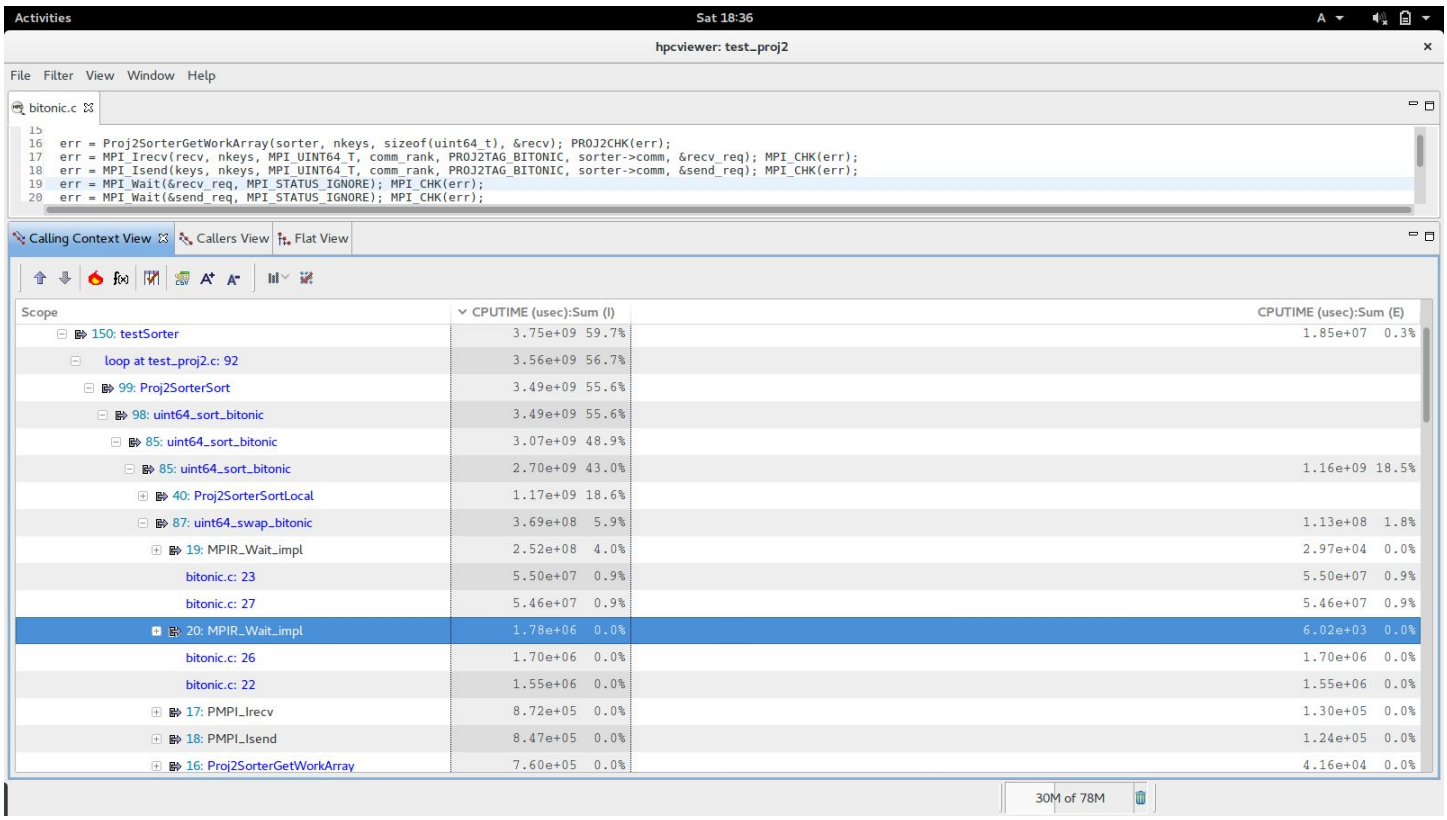

Figure 13. Isend Bitoni Hpcviewer

From Figure 11,12, and 13, it can be seen that, even though it is marginal, that all the MPI_Wait incurred by MPI_Ssend, were mitigated, by changing MPI_Ssend to MPI_Isend. The bandwidth for bitonic sort with MPI_Isend was 5.791104e+07, about 20% better. Following table, Table 2, is the summary for bitonic sort.

Table 2. Bitonic Sort

Cases	Bandwidth
Ssend	4.962060e+07
Isend	5.791104e+07

Following are the output files for bitonic sorts, and batch script for both bitonic sorts.

Ssend:

---

ac7ac123c83b8605e579fdd759a1a5fd9b30f0f0

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o test_proj2.o test_proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2.o proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2sorter.o proj2sorter.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o local.o local.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o bitonic.o bitonic.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c

mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm

mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2

/home1/05236/tg845354/proj2/cse6230fa17-proj2-achintapalli3

c455-012.stampede2.tacc.utexas.edu

Sat Oct 28 19:36:58 CDT 2017

TACC: Starting up job 370404

TACC: Starting parallel tasks...

[0] test_proj2 minKeys 80 maxKeys 100000000 mult 64 seed 0

[0] Testing numKeysLocal 80, numKeysGlobal 20480, total bytes 163840

[0] Tested numKeysLocal 80, numKeysGlobal 20480, total bytes 163840: average bandwidth 1.340436e+07

[0] Testing numKeysLocal 5120, numKeysGlobal 1310720, total bytes 10485760

[0] Tested numKeysLocal 5120, numKeysGlobal 1310720, total bytes 10485760: average bandwidth 2.517927e+08

[0] Testing numKeysLocal 327680, numKeysGlobal 83886080, total bytes 671088640

[0] Tested numKeysLocal 327680, numKeysGlobal 83886080, total bytes 671088640: average bandwidth 1.150988e+09

[0] Testing numKeysLocal 20971520, numKeysGlobal 5368709120, total bytes 42949672960

[0] Tested numKeysLocal 20971520, numKeysGlobal 5368709120, total bytes 42949672960: average bandwidth 8.556014e+08

[0] Harmonic average bandwidth: 4.962060e+07

TACC: Shutdown complete. Exiting.

Sat Oct 28 19:42:39 CDT 2017

Isend

8b6c73b762e1902a0d47a9f718c4d73ad84fbcab

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o test_proj2.o test_proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2.o proj2.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o proj2sorter.o proj2sorter.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o local.o local.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o bitonic.o bitonic.c

mpicc -I../cse6230fa17-proj2-achintapalli3/Random123/include -g -Wall -std=c99 -fpic -O3 -c -o quicksort.o quicksort.c

mpicc -shared -o libproj2.so proj2.o proj2sorter.o local.o bitonic.o quicksort.o -lm

mpicc -L./ -Wl,-rpath,./ -o test_proj2 test_proj2.o -lproj2

/home1/05236/tg845354/proj2/cse6230fa17-proj2-achintapalli3

c455-022.stampede2.tacc.utexas.edu

Sat Oct 28 19:27:33 CDT 2017

TACC: Starting up job 370384

TACC: Starting parallel tasks...

[0] test_proj2 minKeys 80 maxKeys 100000000 mult 64 seed 0

[0] Testing numKeysLocal 80, numKeysGlobal 20480, total bytes 163840

[0] Tested numKeysLocal 80, numKeysGlobal 20480, total bytes 163840: average bandwidth 1.608759e+07

[0] Testing numKeysLocal 5120, numKeysGlobal 1310720, total bytes 10485760

[0] Tested numKeysLocal 5120, numKeysGlobal 1310720, total bytes 10485760: average bandwidth 2.047780e+08

[0] Testing numKeysLocal 327680, numKeysGlobal 83886080, total bytes 671088640

[0] Tested numKeysLocal 327680, numKeysGlobal 83886080, total bytes 671088640: average bandwidth 1.144203e+09

[0] Testing numKeysLocal 20971520, numKeysGlobal 5368709120, total bytes 42949672960

[0] Tested numKeysLocal 20971520, numKeysGlobal 5368709120, total bytes 42949672960: average bandwidth 8.662129e+08

[0] Harmonic average bandwidth: 5.791104e+07

TACC: Shutdown complete. Exiting.

Sat Oct 28 19:33:11 CDT 2017

Following lines are bash script for running Bitonic sorts, for both Ssend and Isend.

```
#!/bin/sh
#SBATCH  -J proj2               # Job name
#SBATCH  -p development                 # Queue (development or normal)
#SBATCH  -N 4                   # Number of nodes
#SBATCH --tasks-per-node 64             # Number of tasks per node
#SBATCH  -t 00:06:00                    # Time limit hrs:min:sec
#SBATCH  -A TG-TRA170035                # Our allocation
#SBATCH  -o proj2-%j.out                # Standard output and error log

git rev-parse HEAD

git diff-files

module load hpctoolkit

make test_proj2

pwd; hostname; date

ibrun tacc_affinity hpcrun -t test_proj2 80 10000000 64 0 4

date
```

Conclusion

Overall, three optimizations were found. The first optimization is changing the pivot in swensonsort's recursive quicksort, to abuse the symmetrical saw-tooth nature of keys. The recursive queue sort algorithm requires sorting of two subarrays that are already sorted. Therefore taking the median value here to pivot by would result in taking value at the highend or low end of the key distribution, which would dramatically decrease performance time due to quicksort having worst case performance on already sorted arrays. Thus, the performance was improved by taking the value at ⅓ of the length as the pivot. The second optimization was utilizing sample sort instead of quicksort for large keys size, which was determined to be twenty million for numKeysLocal, as space complexity of samplesort was less than that of quicksort. Third optimization was for bitonic sort, where the inefficient and bottleneck of using Ssend in **uint64_swap_bitonic** was switched to Isend to avoid such bottleneck.

Thus in our final submission if there are an even number of nodes than bitonic sort is utilized, and if there are an odd number of nodes and if there are less than 20,000,000 keys then quicksort is utilized, otherwise sample sort is utilized.