Parallel PageRank Report

Mandy Chan, Eric Dazet, Nathik Salam, Garrett Summers

CPE 419 and CPE 466

November 17, 2015

Table of Contents

Implementation

Original

The original implementation of the pagerank algorithm was all in python and adhered to the pagerank pseudocode. No changes have really been made that would impact the results. The algorithm was implemented using python v.2.6. At the start of the program, a file is taken as input. There is also a flag that can be used to specify if the file is using the second and fourth column values to determine direction(for the football file). Otherwise, it is assumed that the graph is directed. Graphs can be considered “undirected” when they include the mirrored edge value. Given a menu, the user can specify if the file is a csv or if it is a SNAP file.

This file is then parsed one line at a time. We create the graph that the data represents as each line comes in. This is done by creating two dictionaries to store information. Each dictionary has a key that is associated with a vertex in the graph and each value is a list. One dictionary stores the lists of outward edges of the current node to other nodes. The second dictionary stores the lists of inward edges to the current node from the other nodes.

In implementing pagerank, we made a vector with each dimension made up of the pagerank keyed to each vertice. The initial pagerank values for each vertex is set to 1/n where n is the total number of vertices in the graph. The pagerank algorithm is then ran on each vertex in the graph one time. Once every vertex has been updated, the new vector that has been created is compared to the old vector. We use a “D” value of .85 for the random link followed chance. If the differences in the lengths of each vector is less than 0.0000000001, then the algorithm sorts the values in order of highest rank. The program then outputs the top twenty results.

Parallel

In both parallel implementations, instead of making matrices that map the connections between pages, a set of arrays were used. In order to make this plausible, we altered the python parser from the original implementation to create an internal mapping between the names of the pages to a list of consecutive numbers starting at zero; this was only necessary for the smaller CSV files and the wiki-Vote SNAP file because all the other files were read in as integers, initially sorted, and started at node zero. The python driver then invoked the C programs, included the number of nodes, edges, and iterations for the desired file as command-line arguments, and then wrote all of the needed data, edges, in-degrees, and out-degrees, to stdin such that the C program could read from stdin via the pipe that was created when it was invoked.

Both C programs initially allocate memory for two integer arrays, one for the in-degrees and other for out-degrees, both with a size equal to the number of nodes. Since the node numbers

are provided by the python program in sorted consecutive order, the node's in-degrees and out-degrees are stored at the index of the array corresponding to the node's number.

Each C program then allocates two more arrays, one with a size equal to number of nodes called running indices and the other with a size equal to number of edges called edges. The edges array holds all of the in-degrees id's for each node. However, with an array like edges, it is not readily apparent where the in-degrees for each node starts or ends. Therefore, to solve this problem, the running indices array stores the beginning of in-degrees list for each node. The figure below explains the use of both the edges array and the running indices array.

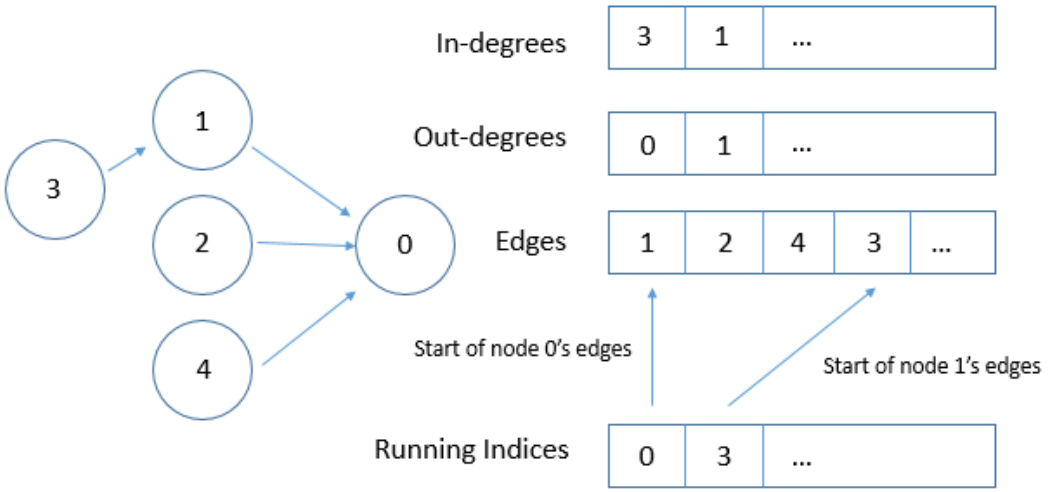


Figure 1: Picture depicting the arrays used in the parallel implementation

We chose to use a makeshift "structure of arrays" methodology because we were not able to use standard matrices given the size of the some of the datasets. Also, because of the amount of memory the arrays used as oppose to a matrix-like structure, we were able to transfer all of the data to the device in both implementations, keep it there, and only bring back the final pagerank values.

The pagerank values are then calculated 'iteration' number of times using the following equation.

$$pageRankNew(node) = (1-d) \cdot \frac{1}{number\ of\ nodes} + d \cdot \sum_{indegrees\ of\ node} \frac{1}{number\ of\ outdegrees\ for\ indegree} \cdot pageRankOld(node)$$

Each C program then prints the final pagerank values to stdout, which are subsequently read by the python program via the pipe. The python program then maps the nodes to their corresponding names from the initial read, if necessary, sorts the entries by pagerank value, and then prints the top twenty pages and their values.

Xeon Phi

In the Xeon phi implementation, two pagerank arrays are allocated, one called old and the other called new. The old one's values are initially assigned to 1/nodes. At the end of each iteration when new pagerank values are calculated, the array pointers are swapped and fed through the loop again, such that the just computed values become the values used for the next iteration.

All of the arrays were allocated using the memalign function and were aligned to 64-byte boundaries.

It is also worth noting that the phi version was designed to read in the edge values first because this was always the largest single array. After all the edges were read, the edges array was asynchronously transferred to the Xeon Phi while the other arrays were being populated.

Within the offload call, there are three main loops: one for the number of iterations, one that goes through each node and another that goes through the in-degrees of each node. Both inner loops were parallelized using OpenMP and the second loop used static scheduling.

Cuda

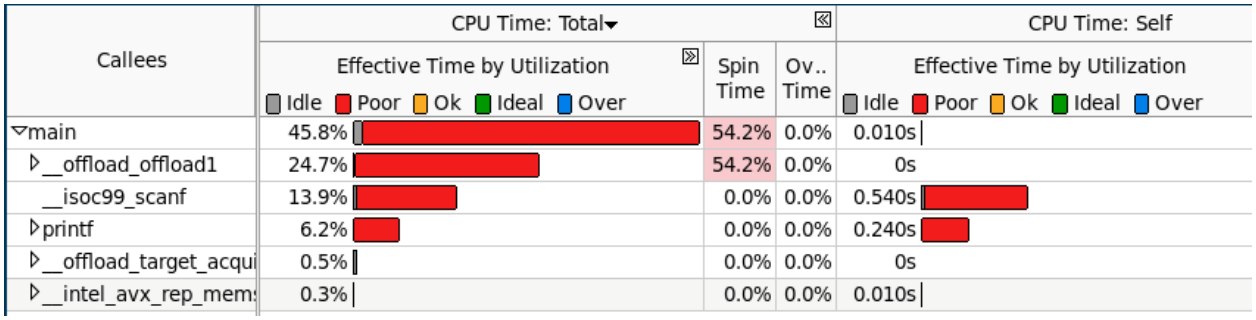
For the Cuda implementation, again two arrays are created to hold new and old pagerank values; however unlike the phi version, the new array just temporarily holds the new values as a synchronization mechanism until all of the threads finish calculating the new pagerank values; at this time, the new values are directly copied to the old array to be used for the next iteration. Each thread is assigned a single node as a part of a one dimensional block of threads with the maximum size of 1024. The grid is also one dimensional with an x-dimension size equal to the ceiling of number of nodes divided by 1024.

Profiling

We used Intel's VTune and NVIDIA's nvprof to profile our Xeon Phi and Cuda versions respectively. We also did not profile our two implementations until we were confident that they were working. It is also important to note that because of the way we were using a python driver to run our parallel implementations, in order to profile, we needed to use the python driver to instead generate input files that we could then pass to the parallel versions when within their profilers. Profile results vs actual results are explained below.

Xeon Phi

Below is a screenshot of Intel's VTune profiler after running our program with the amazon0505.txt dataset.



Callees	CPU Time: Total — Effective Time by Utilization (Idle / Poor / Ok / Ideal / Over)	Spin Time	Ov.. Time	CPU Time: Self — Effective Time by Utilization (Idle / Poor / Ok / Ideal / Over)
main	45.8%	54.2%	0.0%	0.010s
__offload_offload1	24.7%	54.2%	0.0%	0s
__isoc99_scanf	13.9%	0.0%	0.0%	0.540s
printf	6.2%	0.0%	0.0%	0.240s
__offload_target_acqui	0.5%	0.0%	0.0%	0s
__intel_avx_rep_mem:	0.3%	0.0%	0.0%	0.010s

Figure 2: Intel's VTune profiler output

From the picture above, we interpreted the spin time to be the time when work was being done on the Xeon Phi. As for the work done by the CPU, the most time was spent moving the data to the card, scanning the data in, and printing the data out.

These results are slightly misleading because when the Xeon Phi implementation is called from the python driver using the same dataset, the read time, which is printed out, takes longer than the computation, the opposite of the results shown in the picture.

Cuda

We tested the Cuda implementation using the karate.csv and amazon0505.txt datasets. The nvprof outputs are shown below.

```
Time(%)      Time     Calls       Avg       Min       Max  Name
 99.19%  719.29us        38  18.928us  18.656us  20.064us  CalcPageRank(int, int, int*, int*, int*, int*, double*, double*)
  0.57%  4.1280us         5     825ns     736ns     960ns  [CUDA memcpy HtoD]
  0.24%  1.7600us         1  1.7600us  1.7600us  1.7600us  [CUDA memcpy DtoH]

==26613== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 99.47%  296.53ms         6  49.421ms  6.8380us  296.49ms  cudaMalloc
  0.18%  545.53us        83  6.5720us     153ns  281.24us  cuDeviceGetAttribute
  0.17%  502.62us         6  83.769us  5.1140us  455.17us  cudaMemcpy
  0.09%  254.54us        38  6.6980us  5.2620us  30.720us  cudaLaunch
  0.04%  133.90us         6  22.316us  6.1460us  92.835us  cudaFree
  0.02%  65.903us       304     216ns     188ns  2.2410us  cudaSetupArgument
  0.01%  38.683us         1  38.683us  38.683us  38.683us  cuDeviceTotalMem
  0.01%  32.674us         1  32.674us  32.674us  32.674us  cuDeviceGetName
  0.00%  9.9630us        38     262ns     207ns  1.0510us  cudaConfigureCall
  0.00%  2.5360us         2  1.2680us     243ns  2.2930us  cuDeviceGetCount
  0.00%     619ns         2     309ns     167ns     452ns  cuDeviceGet
```

Figure 3: NVPROF output for karate.csv

```
Time(%)      Time     Calls       Avg       Min       Max  Name
 98.90%  341.65ms        85  4.0195ms  3.7906ms  4.4386ms  CalcPageRank(int, int, int*, int*, int*, int*, double*, double*)
  0.97%  3.3423ms         6  557.04us  146.82us  2.0376ms  [CUDA memcpy HtoD]
  0.13%  446.46us         1  446.46us  446.46us  446.46us  [CUDA memcpy DtoH]

==21658== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 53.58%  344.89ms         7  49.270ms  283.98us  340.78ms  cudaMemcpy
 46.04%  296.34ms         6  49.389ms  176.67us  295.43ms  cudaMalloc
  0.19%  1.2160ms        85  14.305us  12.221us  54.208us  cudaLaunch
  0.07%  444.25us         6  74.041us  60.572us  124.40us  cudaFree
  0.06%  389.46us        83  4.6920us     159ns  208.79us  cuDeviceGetAttribute
  0.04%  239.63us       680     352ns     259ns  6.6140us  cudaSetupArgument
  0.01%  52.733us        85     620ns     430ns  5.6690us  cudaConfigureCall
  0.01%  36.920us         1  36.920us  36.920us  36.920us  cuDeviceTotalMem
  0.00%  31.343us         1  31.343us  31.343us  31.343us  cuDeviceGetName
  0.00%  2.2440us         2  1.1220us     352ns  1.8920us  cuDeviceGetCount
  0.00%     574ns         2     287ns     179ns     395ns  cuDeviceGet
```

Figure 4: NVPROF output for amazon0505.txt

From these results, we saw that most of the time spent on the GPU was spent doing the pagerank calculation, which was what we wanted. We also noticed that for the smaller datasets, the largest percentage of time was spent allocating memory. However, for the larger datasets, the largest percentage of time was spent copying memory, which was also not surprising.
For the Cuda implementation, we also tried using the nvvp profiler, but ran into problems. Aside from the results above, similar to the Xeon Phi implementation, the time to read the data in from python took longer than the actual computation and moving of the data, especially for the larger datasets.

Results

NCAA Football

This is the only dataset that requires the -w flag in order to achieve the correct results. Refer to Appendix A for information about how to properly run our program with the -w flag.

The output from the parallel versions were in the same order as the original version, with the only difference being some very miniscule floating-point differences.

U.S. State Borders

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being some very miniscule floating-point differences.

Karate

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being some very miniscule floating-point differences.

Dolphins

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being the floating-point values.

Les Miserables

No special settings are needed to run this dataset.

The output from the parallel versions for this dataset were the only outputs that differed from the original version by more than just the pagerank values. In the original dataset, there were three instances where two pages had the same pagerank value: two instances are readily visible, the third involved the last printed value and a value that wasn't printed. For both parallel implementations, all three instances were printed in the opposite order; i.e: "Joly" and then "Bahorel" in the original version became "Bahorel" and then "Joly" in both parallel versions.

Political Blogs

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being the floating-point values.

Wiki Vote

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being the floating-point values.

P2P Gnutella

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being the floating-point values.

SlashdotZoo

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being the floating-point values.

Amazon Product Co-Purchasing Network

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being some floating-point differences.

Live Journal

No special settings are needed to run this dataset.

The output from the parallel versions were in the same order as the original version, with the only difference being some floating-point differences.

Overall Performance Summary

Right away, something we noticed was that the pagerank values for the parallel versions weren't exactly equal to those of the original version. They were however within a reasonable tolerance of correctness so this wasn't too big of a deal.

In terms of timing, for the smaller datasets, the Xeon Phi implementation was often slower than the original and Cuda versions in total runtime and compute time. However, when run using the larger datasets, the Xeon Phi and Cuda implementations largely outperformed the original version in terms of both compute time and overall runtime.

Below is a table that tabulates what we thought was the most important statistic, the computation time. This table contains the computation times for all three implementations on all datasets for an easy side-by-side comparison. The more detailed results for each dataset can be found in Appendix B.

Table 1: Computation Time Comparisons

	Original (s)	Xeon Phi (s)	Cuda (s)
NCAA_Football	0.00479102134705	0.402700	0.000106
U.S. State Borders	0.0144698619843	0.372428	0.001235
Karate	0.00424194335938	0.345820	0.000848
Dolphins	0.0157480239868	0.288894	0.000864
Les Miserables	0.0307269096375	0.372734	0.002492
Political Blogs	0.840591192245	0.297704	0.025829
Wiki Vote	2.60404109955	0.302456	0.024248
P2P Gnutella	0.591051101685	0.297327	0.024120
SlashdotZoo	31.9098169804	0.773699	0.190957
Amazon Product	282.415816784	3.351078	0.339811
Live Journal	5646.19970703	76.405948	5.242004

Appendix A

README

How to Build:

Typing “make” will build both programs.
Typing “make clean” will remove all object files and executables.

How to Run:

python pagerank.py file_name version [-w]

file_name: name of input file (file does not need to be in the current working directory)

version: three possible values

- phi -> runs only Xeon Phi version
- cuda -> runs only Cuda version
- both -> runs both versions sequentially (Xeon Phi first, Cuda second)

-w : Flag for the NCAA-Football dataset. This flag is used to determine which team won and therefore, what team the arrow between the two teams points to. **MAKE SURE TO PUT -w AS THE LAST COMMAND-LINE ARGUMENT IF NEEDED.**

Program Notes:

We were able to successfully run the Live Journal dataset and get a significant speedup with both parallel implementations. However, the results in the overall performance summary above and in the detailed results below were gathered using different values of ‘d’ and ‘vector difference’ than those required to be considered for extra credit.

Appendix B

Detailed Outputs

NCAA Football

Original

```
North Dakota 0.00111112440191
Weber State 0.00109834244703
Montana 0.00108396274778
South Dakota 0.00102208646617
Northwestern State 0.000969627192982
Florida 0.000935009398496
Iona 0.000934210526316
Richmond 0.000922049916458
Utah 0.000915215121136
Oklahoma 0.000900754727728
Central Arkansas 0.000898083751044
Grambling State 0.00089595342523
James Madison 0.000894089390142
Sacred Heart 0.000887972772841
Savannah State 0.000887609649123
Texas Tech 0.000878386306676
Texas 0.000869768170426
South Carolina State 0.000862400793651
Bryant University 0.000856017885623
Liberty 0.000840032372598
Mississippi 0.000839943609023

Read Time: 0.0145938396454 seconds
Process Time: 0.00479102134705 seconds
Number of Iterations: 3
```

Xeon Phi

```
North Dakota 0.001111124401914
Weber State 0.001098342447027
Montana 0.001083962747779
South Dakota 0.001022086466165
Northwestern State 0.000969627192982
Florida 0.000935009398496
Iona 0.000934210526316
Richmond 0.000922049916458
Utah 0.000915215121136
Oklahoma 0.000900754727728
Central Arkansas 0.000898083751044
Grambling State 0.000895953425230
James Madison 0.000894089390142
Sacred Heart 0.000887972772841
Savannah State 0.000887609649123
Texas Tech 0.000878386306676
Texas 0.000869768170426
South Carolina State 0.000862400793651
Bryant University 0.000856017885623
Liberty 0.000840032372598
Mississippi 0.000839943609023

Read Time (Python): 0.0140960216522 seconds
Read Time (C): 0.206961 seconds
Compute Time: 0.402700 seconds
Total Program Time: 3.736552 seconds
Number of Iterations: 3 (passed as parameter)
```

Cuda

```
North Dakota 0.001111124401914
Weber State 0.001098342447027
Montana 0.001083962747779
South Dakota 0.001022086466165
Northwestern State 0.000969627192982
Florida 0.000935009398496
Iona 0.000934210526316
Richmond 0.000922049916458
Utah 0.000915215121136
Oklahoma 0.000900754727728
Central Arkansas 0.000898083751044
Grambling State 0.000895953425230
James Madison 0.000894089390142
Sacred Heart 0.000887972772841
Savannah State 0.000887609649123
Texas Tech 0.000878386306676
Texas 0.000869768170426
South Carolina State 0.000862400793651
Bryant University 0.000856017885623
Liberty 0.000840032372598
Mississippi 0.000839943609023

Read Time (Python): 0.0140960216522 seconds
Read Time (C): 0.0054 seconds
Compute Time: 0.000106 seconds
Total Program Time: 0.398892940521 seconds
Number of Iterations: 3 (passed as parameter)
```

U.S. State Borders

Original

```
MA 0.028209302551
NY 0.0250633995942
TN 0.0247847335561
PA 0.0234381845554
ID 0.0234038238942
AR 0.0227189493141
MO 0.0226252674367
KY 0.0220637836169
OK 0.0213082182744
GA 0.0212646649958
VA 0.0209480015822
NV 0.0208126268499
TX 0.0202331457831
NH 0.0200858490565
MD 0.019040431089
UT 0.018517897023
OR 0.0180315654402
WY 0.0179985321193
CO 0.0179731863717
NB 0.01782675111
SD 0.0178131225666

Read Time: 0.00240206718445  seconds
Process Time: 0.0144698619843 seconds
Number of Iterations: 93
```

Xeon Phi

```
MA 0.028209302580805
NY 0.025063399619763
TN 0.024784733572549
PA 0.023438184575082
ID 0.023403823907041
AR 0.022718949329105
MO 0.022625267450729
KY 0.022063783631434
OK 0.021308218287788
GA 0.021264665007989
VA 0.020948001595945
NV 0.020812626861731
TX 0.020233145795917
NH 0.020085849075147
MD 0.019040431102530
UT 0.018517897033417
OR 0.018031565449846
WY 0.017998532128575
CO 0.017973186381779
NB 0.017826751119527
SD 0.017813122574706

Read Time (Python): 0.00179600715637 seconds
Read Time (C): 0.195711 seconds
Compute Time: 0.372428 seconds
Total Program Time: 4.80342507362 seconds
Number of Iterations: 93 (passed as parameter)
```

Cuda

```
MA 0.028209302551042
NY 0.025063399594177
TN 0.024784733556139
PA 0.023438184555401
ID 0.023403823894220
AR 0.022718949314094
MO 0.022625267436669
KY 0.022063783616941
OK 0.021308218274360
GA 0.021264664995797
VA 0.020948001582217
NV 0.020812626849872
TX 0.020233145783109
NH 0.020085849056525
MD 0.019040431089033
UT 0.018517897022970
OR 0.018031565440249
WY 0.017998532119340
CO 0.017973186371712
NB 0.017826751110021
SD 0.017813122566575

Read Time (Python): 0.00179600715637 seconds
Read Time (C): 0.000061 seconds
Compute Time: 0.001235 seconds
Total Program Time: 0.392478942871 seconds
Number of Iterations: 93 (passed as parameter)
```

Karate

Original

```
34 0.100919162391
1 0.0969973059667
33 0.071693210407
3 0.0570785089414
2 0.052876929866
32 0.0371580816843
4 0.0358598626948
24 0.0315225071646
9 0.0297660540734
14 0.0295364580469
7 0.0291111668711
6 0.0291111668711
30 0.0262885313584
28 0.0256397627862
31 0.0245901528135
8 0.0244905002157
11 0.0219779605028
5 0.0219779605028
25 0.0210760294319
26 0.0210061929599
20 0.0196046373318

Read Time: 0.00192284584045 seconds
Process Time: 0.00424194335938 seconds
Number of Iterations: 38
```

Xeon Phi

```
34 0.100919162390698
1 0.096997305966699
33 0.071693210406958
3 0.057078508941365
2 0.052876929865968
32 0.037158081684259
4 0.035859862694826
24 0.031522507164556
9 0.029766054073440
14 0.029536458046873
6 0.029111166871070
7 0.029111166871070
30 0.026288531358414
28 0.025639762786208
31 0.024590152813452
8 0.024490500215732
5 0.021977960502774
11 0.021977960502774
25 0.021076029431910
26 0.021006192959876
20 0.019604637331763

Read Time (Python): 0.00199604034424 seconds
Read Time (C): 0.178796 seconds
Compute Time: 0.345820 seconds
Total Program Time: 4.39518713951 seconds
Number of Iterations: 38 (passed as parameter)
```

Cuda

```
34 0.100919162390698
1 0.096997305966699
33 0.071693210406958
3 0.057078508941365
2 0.052876929865968
32 0.037158081684259
4 0.035859862694826
24 0.031522507164556
9 0.029766054073440
14 0.029536458046873
6 0.029111166871070
7 0.029111166871070
30 0.026288531358414
28 0.025639762786208
31 0.024590152813452
8 0.024490500215732
5 0.021977960502774
11 0.021977960502774
25 0.021076029431910
26 0.021006192959876
20 0.019604637331763

Read Time (Python): 0.00199604034424 seconds
Read Time (C): 0.000051 seconds
Compute Time: 0.000848 seconds
Total Program Time: 0.392958991241 seconds
Number of Iterations: 38 (passed as parameter)
```

Dolphins

Original

```
Grin 0.0321444715593
Jet 0.0317281771596
Trigger 0.0312993379427
Web 0.0300954086468
SN4 0.029875320328
Topless 0.0295141841384
Scabs 0.0284230515197
Patchback 0.0264585331357
Gallatin 0.0261569114829
Beescratch 0.0246507369799
Kringel 0.0246409066077
SN63 0.0239392298089
Feather 0.0234585111566
SN9 0.0219663567815
Stripes 0.0216911115742
Upbang 0.0216509011437
SN100 0.0206133868192
DN21 0.0200536556942
Haecksel 0.0198830695071
Jonah 0.0193955377775
TR99 0.0192319326478

Read Time: 0.0042359828949 seconds
Process Time: 0.0157480239868 seconds
Number of Iterations: 49
```

Xeon Phi

```
Grin 0.032144466787378
Jet 0.031728185423684
Trigger 0.031299333678421
Web 0.030095417026593
SN4 0.029875316218384
Topless 0.029514179765920
Scabs 0.028423047424093
Patchback 0.026458529299939
Gallatin 0.026156919467666
Beescratch 0.024650741506713
Kringel 0.024640903855267
SN63 0.023939226527911
Feather 0.023458518311140
SN9 0.021966354709195
Stripes 0.021691108529537
Upbang 0.021650906557532
SN100 0.020613386230803
DN21 0.020053661629942
Haecksel 0.019883066849945
Jonah 0.019395534879100
TR99 0.019231929939034

Read Time (Python): 0.00309300422668 seconds
Read Time (C): 0.179856 seconds
Compute Time: 0.288894 seconds
Total Program Time: 4.30766415596 seconds
Number of Iterations: 49 (passed as parameter)
```

Cuda

```
Grin 0.032144471559265
Jet 0.031728177159611
Trigger 0.031299337942673
Web 0.030095408646755
SN4 0.029875320328037
Topless 0.029514184138400
Scabs 0.028423051519748
Patchback 0.026458533135668
Gallatin 0.026156911482948
Beescratch 0.024650736979890
Kringel 0.024640906607707
SN63 0.023939229808943
Feather 0.023458511156620
SN9 0.021966356781454
Stripes 0.021691111574202
Upbang 0.021650901143667
SN100 0.020613386819185
DN21 0.020053655694163
Haecksel 0.019883069507113
Jonah 0.019395537777519
TR99 0.019231932647788

Read Time (Python): 0.00309300422668 seconds
Read Time (C): 0.000074 seconds
Compute Time: 0.000864 seconds
Total Program Time: 0.40810300827 seconds
Number of Iterations: 49 (passed as parameter)
```

Les Miserables

Original

```
Valjean 0.0754301223136
Myriel 0.0427792833019
Gavroche 0.0357673176562
Marius 0.0308949358468
Javert 0.0303027358967
Thenardier 0.0279265255633
Fantine 0.0270227049443
Enjolras 0.0218820329331
Cosette 0.0206112150563
MmeThenardier 0.0195011346455
Bossuet 0.0189595297273
Courfeyrac 0.0185784420744
Eponine 0.0177939117009
Mabeuf 0.0174780219455
Joly 0.0171998776172
Bahorel 0.0171998776172
Babet 0.0166918379631
Gueulemer 0.0166918379631
Claquesous 0.0165610202438
MlleGillenormand 0.0162602085942
Feuilly 0.0158921243334

Read Time: 0.00648808479309 seconds
Process Time: 0.0307269096375 seconds
Number of Iterations: 61
```

Xeon Phi

```
Valjean 0.075430122583863
Myriel 0.042779283104112
Gavroche 0.035767317497056
Marius 0.030894935737374
Javert 0.030302735898445
Thenardier 0.027926525523114
Fantine 0.027022704949399
Enjolras 0.021882032815598
Cosette 0.020611215048335
MmeThenardier 0.019501134629540
Bossuet 0.018959529613510
Courfeyrac 0.018578441952448
Eponine 0.017793911654443
Mabeuf 0.017478021845097
Bahorel 0.017199877500500
Joly 0.017199877500500
Gueulemer 0.016691837940144
Babet 0.016691837940144
Claquesous 0.016561020220295
MlleGillenormand 0.016260208581018
Combeferre 0.015892124227425

Read Time (Python): 0.0101189613342 seconds
Read Time (C): 0.182639 seconds
Compute Time: 0.372734 seconds
Total Program Time: 4.49520684242 seconds
Number of Iterations: 61 (passed as parameter)
```

Cuda

```
Valjean 0.075430122313551
Myriel 0.042779283301896
Gavroche 0.035767317656156
Marius 0.030894935846765
Javert 0.030302735896721
Thenardier 0.027926525563276
Fantine 0.027022704944294
Enjolras 0.021882032933070
Cosette 0.020611215056328
MmeThenardier 0.019501134645530
Bossuet 0.018959529727321
Courfeyrac 0.018578442074404
Eponine 0.017793911700874
Mabeuf 0.017478021945469
Bahorel 0.017199877617236
Joly 0.017199877617236
Gueulemer 0.016691837963134
Babet 0.016691837963134
Claquesous 0.016561020243758
MlleGillenormand 0.016260208594221
Combeferre 0.015892124333444

Read Time (Python): 0.0101189613342 seconds
Read Time (C): 0.000095 seconds
Compute Time: 0.002492 seconds
Total Program Time: 0.400594005585 seconds
Number of Iterations: 61 (passed as parameter)
```

Political Blogs

Original

```
155 0.0117132335908
55 0.00994072561462
1051 0.00824181775149
855 0.00815474381255
641 0.00811666933456
1153 0.00712240166018
963 0.00699267730083
729 0.00688415624254
1245 0.00583295903484
798 0.00562304801973
323 0.00555703989525
1112 0.0055347969462
1461 0.00468328129442
1306 0.00456958979674
1463 0.00445700644687
1179 0.00441149814043
1041 0.00437195019729
1437 0.00420687885325
535 0.00406617883232
990 0.00393442428027
180 0.00363415891448

Read Time: 0.111705064774 seconds
Process Time: 0.840591192245 seconds
Number of Iterations: 71
```

Xeon Phi

```
155 0.011713233618724
55 0.009940725643493
1051 0.008241817773532
855 0.008154743825097
641 0.008116669356900
1153 0.007122401677591
963 0.006992677308798
729 0.006884156263840
1245 0.005832959048560
798 0.005623048033480
323 0.005557039911032
1112 0.005534796959424
1461 0.004683281308207
1306 0.004569589808710
1463 0.004457006460899
1179 0.004411498152664
1041 0.004371950207810
1437 0.004206878860599
535 0.004066178844916
990 0.003934424289102
180 0.003634158925447

Read Time (Python): 0.118390083313 seconds
Read Time (C): 0.190171 seconds
Compute Time: 0.297704 seconds
Total Program Time: 4.34534692764 seconds
Number of Iterations: 71 (passed as parameter)
```

Cuda

```
155 0.011713233559095
55 0.009940725581905
1051 0.008241817726437
855 0.008154743798228
641 0.008116669309182
1153 0.007122401640439
963 0.006992677291731
729 0.006884156218331
1245 0.005832959019269
798 0.005623048004130
323 0.005557039877382
1112 0.005534796931181
1461 0.004683281278783
1306 0.004569589783168
1463 0.004457006430980
1179 0.004411498126558
1041 0.004371950185341
1437 0.004206878844898
535 0.004066178818043
990 0.003934424270259
180 0.003634158902050

Read Time (Python): 0.172997951508 seconds
Read Time (C): 0.054998 seconds
Compute Time: 0.025829 seconds
Total Program Time: 0.626409215927 seconds
Number of Iterations: 71 (passed as parameter)
```

Wiki Vote

Original

```
4037 0.00192379827873
15 0.00153658553046
6634 0.00149774701877
2625 0.00137114263933
2398 0.00108927702371
2470 0.00105384087064
2237 0.00104250603246
4191 0.000946977446693
7553 0.00090600533601
5254 0.000897808548444
2328 0.000851525252608
1186 0.000849969512974
1297 0.000812516629124
4335 0.000808725841007
7620 0.000806770846466
5412 0.000801273672891
7632 0.000796608094171
4875 0.00078244050251
6946 0.000755135691423
3352 0.000744919211647
6832 0.000738332838156

Read Time: 0.1811170578 seconds
Process Time: 2.60404109955 seconds
Number of Iterations: 38
```

Xeon Phi

```
4037 0.001923798278725
15 0.001536585530460
6634 0.001497747018769
2625 0.001371142639333
2398 0.001089277023711
2470 0.001053840870641
2237 0.001042506032459
4191 0.000946977446693
7553 0.000906006005336010
5254 0.000897808548444
2328 0.000851525252608
1186 0.000849969512974
1297 0.000812516629124
4335 0.000808725841007
7620 0.000806770846466
5412 0.000801273672891
7632 0.000796608094171
4875 0.000782440502510
6946 0.000755135691423
3352 0.000744919211647
6832 0.000738332838156

Read Time (Python): 0.218215942383 seconds
Read Time (C): 0.422093 seconds
Compute Time: 0.302456 seconds
Total Program Time: 4.81841798782 seconds
Number of Iterations: 38 (passed as parameter)
```

Cuda

```
4037 0.001923798266375
15 0.001536585517454
6634 0.001497746975760
2625 0.001371142624993
2398 0.001089277009918
2470 0.001053840868148
2237 0.001042506027860
4191 0.000946977437021
7553 0.000906006005326940
5254 0.000897808540425
2328 0.000851525244640
1186 0.000849969510318
1297 0.000812516621207
4335 0.000808725830573
7620 0.000806770838026
5412 0.000801273661134
7632 0.000796608083384
4875 0.000782440494312
6946 0.000755135668254
3352 0.000744919204113
6832 0.000738332829019

Read Time (Python): 0.243498086929 seconds
Read Time (C): 0.306466 seconds
Compute Time: 0.024248 seconds
Total Program Time: 0.965697040558 seconds
Number of Iterations: 38 (passed as parameter)
```

p2p-Gnutella05

Original

```
1676 0.000258797944699
1020 0.000253264016481
386 0.000241780769507
222 0.00023943613222
227 0.000232734971143
388 0.000229984921195
389 0.000228891358599
688 0.00022017999474
226 0.000215716044397
842 0.000215279443899
876 0.000213099432143
223 0.000199179250058
31 0.000198125104824
391 0.000195953158986
279 0.000191967934127
271 0.000191219938609
225 0.000189992196276
277 0.000189644350947
274 0.000187645870201
272 0.000185816063374
887 0.000184364673257

Read Time: 0.0811309814453 seconds
Process Time: 0.591051101685 seconds
Number of Iterations: 21
```

Xeon Phi

```
1676 0.000258797959749
1020 0.000253264030044
386 0.000241780781670
222 0.000239436144812
227 0.000232734984205
388 0.000229984936274
389 0.000228891370462
688 0.000220180005841
226 0.000215716052203
842 0.000215279453229
876 0.000213099439385
223 0.000199179258842
31 0.000198125114418
391 0.000195953167865
279 0.000191967943373
271 0.000191219946926
225 0.000189992204859
277 0.000189644359681
274 0.000187645881521
272 0.000185816073128
887 0.000184364681159

Read Time (Python): 0.087277173996 seconds
Read Time (C): 0.268418 seconds
Compute Time: 0.297327 seconds
Total Program Time: 4.563862084 seconds
Number of Iterations: 21 (passed as parameter)
```

Cuda

```
1676 0.000258797936200
1020 0.000253264009225
386 0.000241780762973
222 0.000239436125553
227 0.000232734964225
388 0.000229984913205
389 0.000228891352198
688 0.000220179988862
226 0.000215716040177
842 0.000215279438748
876 0.000213099428495
223 0.000199179245421
31 0.000198125099653
391 0.000195953154213
279 0.000191967929126
271 0.000191219934296
225 0.000189992191704
277 0.000189644346333
274 0.000187645864267
272 0.000185816058339
887 0.000184364669019

Read Time (Python): 0.15852189064 seconds
Read Time (C): 0.154719 seconds
Compute Time: 0.024120 seconds
Total Program Time: 0.867552804947 seconds
Number of Iterations: 21 (passed as parameter)
```

SlashdotZoo

Original

```
75 0.00182291024155
43 0.00178588742063
749 0.0016401304579
184 0.00108396456414
38 0.00107890705845
625 0.000806629019236
163 0.000640533824724
1810 0.000569742729214
57 0.000527369725609
34 0.000515342333507
74 0.00049639441129
15 0.000485631035697
85 0.000484300454018
651 0.000482566832481
53 0.00047768212115
50 0.00047169022
1808 0.000467661439849
1832 0.000422770338298
877 0.000383776583609
3335 0.000379523473038
1116 0.000361499020606

Read Time: 0.997929811478 seconds
Process Time: 31.9098169804 seconds
Number of Iterations: 64
```

Xeon Phi

```
75 0.001822745287228
43 0.001785725816481
749 0.001639982043238
184 0.001083866476681
38 0.001078809428640
625 0.000806556027740
163 0.000640475863108
1810 0.000569691173452
57 0.000527322004161
34 0.000515295700412
74 0.000496349492784
15 0.000485587091164
85 0.000484256629889
651 0.000482523165227
53 0.000477638895911
50 0.000471647536965
1808 0.000467619121376
1832 0.000422732082001
877 0.000383741855840
3335 0.000379489130131
1116 0.000361466308723

Read Time (Python): 1.09603095055 seconds
Read Time (C): 2.177345 seconds
Compute Time: 0.773699 seconds
Total Program Time: 8.176141976 seconds
Number of Iterations: 64 (passed as parameter)
```

Cuda

```
75 0.001822745286769
43 0.001785725814144
749 0.001639982042872
184 0.001083866476547
38 0.001078809428432
625 0.000806556027360
163 0.000640475862946
1810 0.000569691173240
57 0.000527322003985
34 0.000515295699703
74 0.000496349492580
15 0.000485587090692
85 0.000484256629747
651 0.000482523165129
53 0.000477638895379
50 0.000471647536557
1808 0.000467619121210
1832 0.000422732081925
877 0.000383741855740
3335 0.000379489130045
1116 0.000361466308650

Read Time (Python): 1.10741114616 seconds
Read Time (C): 2.079228 seconds
Compute Time: 0.190957 seconds
Total Program Time: 4.01777801514 seconds
Number of Iterations: 64 (passed as parameter)
```

Amazon Product Co-Purchasing Network

Original

```
593 0.00174566565029
595 0.00145189096493
591 0.00143551834066
89 0.00128833715223
590 0.00102845383708
972 0.000997447823712
976 0.000845201641661
974 0.000825553626934
975 0.000770241847984
978 0.000756642436588
120 0.000733637566414
977 0.000717121715774
634 0.000714444253983
2612 0.000693695462301
598 0.000629822166527
597 0.000558468769431
585 0.000542502107877
162 0.000530391837118
596 0.000479633437179
4455 0.00047392782044
88 0.000445002737534

Read Time: 7.00430202484 seconds
Process Time: 282.415816784 seconds
Number of Iterations: 85
```

Xeon Phi

```
593 0.001745665586833
595 0.001451890901514
591 0.001435518277549
89 0.001288337159178
590 0.001028453792358
972 0.000997447776753
976 0.000845201602301
974 0.000825553588220
975 0.000770241815401
978 0.000756642397112
120 0.000733637570796
977 0.000717121700173
634 0.000714444258344
2612 0.000693695467221
598 0.000629822138845
597 0.000558468740129
585 0.000542502086023
162 0.000530391840858
596 0.000479633415107
4455 0.000473927824178
88 0.000445002739615

Read Time (Python): 7.82812404633 seconds
Read Time (C): 14.715285 seconds
Compute Time: 3.351078 seconds
Total Program Time: 31.55948305 seconds
Number of Iterations: 85 (passed as parameter)
```

Cuda

```
593 0.001745665139174
595 0.001451890460938
591 0.001435517838894
89 0.001288337010674
590 0.001028453482150
972 0.000997447453412
976 0.000845201329973
974 0.000825553321301
975 0.000770241589451
978 0.000756642128656
120 0.000733637472640
977 0.000717121581721
634 0.000714444150515
2612 0.000693695439402
598 0.000629821947219
597 0.000558468541124
585 0.000542501933814
162 0.000530391753236
596 0.000479633264028
4455 0.000473927803263
88 0.000445002696382

Read Time (Python): 7.69310188293 seconds
Read Time (C): 13.104072 seconds
Compute Time: 0.339811 seconds
Total Program Time: 23.4144890289 seconds
Number of Iterations: 85 (passed as parameter)
```

LiveJournal

Original

```
8737 0.000137746984371
2914 0.000121458776084
18964 8.49247967938e-05
1220 7.34615042362e-05
2409 6.93806551654e-05
10029 6.7543940206e-05
214538 6.2578084095e-05
7343 5.91828962791e-05
39295 5.57154614538e-05
38283 5.55105459343e-05
18963 5.54432655543e-05
40509 4.93423210477e-05
1918 4.86858997758e-05
4494 4.85248159272e-05
3407 4.68677082217e-05
214406 4.61029926611e-05
56913 4.55604260649e-05
39633 4.51233679761e-05
1772 4.37890780561e-05
503 4.351284833e-05
1689 4.22236052645e-05

Read Time: 254.646777153 seconds
Process Time: 5646.19970703 seconds
Number of Iterations: 67
```

Xeon Phi

```
8737 0.000137747002826
2914 0.000121458790151
18964 0.000084924802478
1220 0.000073461514607
2409 0.000069380666031
10029 0.000067543948661
214538 0.000062578094723
7343 0.000059182904465
39295 0.000055715464982
38283 0.000055510558466
18963 0.000055443274841
40509 0.000049342326103
1918 0.000048685904212
4494 0.000048524822515
3407 0.000046867711959
214406 0.000046102939996
56913 0.000045560433626
39633 0.000045123372952
1772 0.000043789083465
503 0.000043512852451
1689 0.000042223612485

Read Time (Python): 260.597575903 seconds
Read Time (C): 258.362584 seconds
Compute Time: 76.405948 seconds
Total Program Time: 625.9693229 seconds
Number of Iterations: 85 (passed as parameter)
```

Cuda

```
8737 0.000137746894079
2914 0.000121458707278
18964 0.000084924767367
1220 0.000073461453520
2409 0.000069380601950
10029 0.000067543896440
214538 0.000062578032084
7343 0.000059182856257
39295 0.000055715443189
38283 0.000055510484397
18963 0.000055443217523
40509 0.000049342296325
1918 0.000048685878124
4494 0.000048524783722
3407 0.000046867690046
214406 0.000046102920422
56913 0.000045560389021
39633 0.000045123343773
1772 0.000043789051614
503 0.000043512828202
1689 0.000042223569663

Read Time (Python): 254.43066597 seconds
Read Time (C): 244.415066 seconds
Compute Time: 5.242004 seconds
Total Program Time: 530.684278965 seconds
Number of Iterations: 85 (passed as parameter)
```