Parallel Breadth-First Search Using OpenMP

HingOn Miu (hmiu) An Wu (anwu)
Carnegie Mellon University

Top-down Approach

Implementation Details

We use most of the same implementation detail as the sequential top-down solution provided: two vertex_set (frontier and new_frontier). We parallelize the BFS by adding #pragma omp parallel for, and made the frontier check atomic, using __sync_bool_compare_and_swap. However, since we don't even need to try to swap if the entry is visited before, we do a test before calling __sync_bool_compare_and_swap to reduce synchronization. Also, if the swap is successful, we use a __sync_add_and_fetch to atomically add the counter in the new frontier. Also, we passed a few constants and pointers into top_down_step function like graph -> num_nodes and graph -> outgoing_starts, so that the threads don't need to access memory to fetch them every time.

Optimization Process

First we implemented details above except doing the test before __sync_bool_compare_and_ swap. The performance of our solution is not quite stable, and the performance can go higher than 125% of reference solution sometimes. Then we went to office hours and realized that synchronization could take a lot of costs, so we started to think about what synchronization steps can be "eliminated". After a while we came up with the idea that we don't really need to swap the entry in distance array if it's already visited, and thus add a test before calling the atomic function.

Performance analysis

- Where is the synchronization in your solution? Do you do anything to limit the overhead of synchronization?

There are two places that synchronization happens: to (atomically) compare and swap in the distance array, and to (atomically) add the counter in new_frontier. We did a test before doing the compare and swap in order to reduce synchronization cost.

- Why do you think your code is unable to achieve perfect speedup?

The workload should be relatively balanced for top-down approach because frontiers are evenly divided among all processors, and every frontier should contain the same amount of work on average. However, there are considerable communication/synchronization in the top down approach (two synchronization steps). There isn't much artificial data movement in top down step.

- When you run your code on Blacklight with more than 16 threads, do you see a noticeable drop off in performance? Why do you think this might be the case?

We didn’t see a noticeable drop off in performance of top-down approach when the thread counts go over 16 on Blacklight.

Bottom-up Approach

Implementation Details

We use two boolean arrays (frontier and new_frontier) to keep track of frontiers and new frontiers. There is no synchronization in this approach. For every vertex, we can check the distance array to see if it has been visited before, and if not, we can go through all its incoming neighbors, and check frontier array to see if any neighbor was visited before. If it is, then it must be a new frontier, so we set the respective entry in new_frontier to true, and update this node’s distance. We used “guided” scheduling for the approach. In definition, guided is pretty much like dynamic, but at the start of the program every thread will be assigned more work than designated, and gradually every thread will be assigned less and less work, with a minimum of the designated number. We think this approach will decrease the scheduling overhead for large graphs. Also, we passed a few constants and pointers into bottom_up_step function like graph->num_nodes and graph-> incoming_starts, so that the threads don’t need to access memory to fetch them every time.

Optimization Process

Reading the spec, we first realize that doing linear check to see if a vertex was visited before, or if its incoming neighbors were visited before is probably going to be too slow. Thus, the first solution we came up with is one that uses distances array to check both conditions in O(1). Also there were no synchronizations in the solution, because every node only write to its own entry in distances array. Surprisingly the solution was not fast enough. Confused, we went to office hour. After the TA read several similar solutions that look alike, he concluded that reading and writing one array for all threads may suffer from false sharing, and suggested us to add more data structures to separate read and write. Thus we came up the approach to use frontier (read-only in a round) and new_frontier (write-only in a round), and use boolean array for both in order to increase the number of items that can be held in a cache line. After implementing the optimization above, the performance of our bottom-up approach increased a lot, but still can be unsatisfactory for rmat graphs. Thus, we made an extra optimization to pass constants and pointers to bottom_up_step so that they don’t need to be dereferenced every time to be used, and the performance of rmat graphs became better.

Performance analysis

- Where is the synchronization in your solution? Do you do anything to limit the overhead of synchronization?

There is no explicit synchronization in the solution (no atomic function calls/critical sections). However, cache coherence problem can happen sometimes, because many professors can potentially read from and write to the same cache line. We use several redundant arrays to separate read and write in order to reduce false sharing.

- Why do you think your code is unable to achieve perfect speedup?

The workload should be relatively balanced for in our approach, because we used dynamic scheduling. However, there are some implicit synchronization/communication costs (false sharing). These costs can also be attributed to data movement because when cache are invalidated, data move from one cache to main memory/another cache.

- When you run your code on Blacklight with more than 16 threads, do you see a noticeable drop off in performance? Why do you think this might be the case?

Yes we did. Since the main cost of bottom-up approach is from cache coherence, adding the number of cores could hurt speedup.

Hybrid Approach

Implementation Details

The intuition for hybrid is that top-down is fast when there are few frontiers (because it's likely that there are not many new frontiers, so the bottom-up's approach will waste time traversing all vertices), and bottom-up is fast when there are many frontiers (because top-down will have many new frontiers overlap, increasing synchronization overhead and repeated checks).

We use the same structure and algorithm for hybrid as our parallel bottom-up and top-down solution, except that we keep track of bottom-up's frontier array when we are doing top-down. Once we switch to bottom-up when there are many frontiers, we never switch back again because switching between data structures are expensive. Our policy of switching is that if the ratio of the number of frontiers to the total number of vertices is greater than 0.02, we do the switching. This "magic number" is obtained by experiment.

Optimization Process

First our switching policy is that if the number of frontiers exceed a certain number, then we do the switching. However, this solution is not scalable because graph size (and thus frontier size) could differ greatly. Therefore, we changed our policy to ratio, and after several runs, we decided that 0.02 is a good ratio, favoring large graphs.

Also, we keep track of both approaches' data structures so we can switch back and forth. However, this method proved to be too slow because keeping track of top-down approach's data structures requires synchronization. Then we tried converting one type of data structure to the other when we are about to do the switching, and it still proved to be too slow. Finally, we realized that when the frontier gets large, there shouldn't be too many iterations before the graph is completed processed, so it doesn't hurt too much to keep using bottom-up and never switch back. This approach proved to be sufficiently fast.

Performance analysis

- Where is the synchronization in your solution? Do you do anything to limit the overhead of synchronization?

The only explicit synchronization is the synchronization for top-down approach in our solution. We didn't do any extra step compared to top-down approach. However, we decided not to convert top-down's data structure to that of bottom-up when we want to do the switching, but instead keep track of it when we are doing top-down to reduce the synchronization step needed to convert. Also we decide not to switch back to top-down approach after we take bottom-up because either keeping track of top-down approach's data structures or converting bottom-up approach's data structures to that of top-down could add synchronization costs.

- Why do you think your code is unable to achieve perfect speedup?

The workload should be balanced according to the analysis in the previous two approaches. There are explicit synchronization costs in top-down approach, and there are implicit cache communication costs in both approaches. Also, there are extra data movement in top-down approach because it needs to keep track of bottom-up approach's data structure.

- When you run your code on Blacklight with more than 16 threads, do you see a noticeable drop off in performance? Why do you think this might be the case?

We saw some drop off in performance of hybrid approach when the thread counts go over 16 on Blacklight, though it not as noticeable as the bottom-up approach. We think the reason is that the drop off is counterbalanced by the top-down approach

Tables:

Runtime on ghc39.ghc.andrew.cmu.edu (rmat_32m.graph):

```
----------------------------------------------------------
Timing Summary
Threads  Top Down            Bottom Up           Hybrid
   1:    6.0032 (1.0000x)    6.9139 (1.0000x)    3.4972 (1.0000x)
   2:    4.0748 (1.4732x)    3.8161 (1.8118x)    1.9381 (1.8045x)
   4:    2.9347 (2.0456x)    2.2236 (3.1093x)    1.1487 (3.0445x)
   6:    2.6054 (2.3041x)    1.8216 (3.7956x)    0.9440 (3.7046x)
   8:    2.3794 (2.5230x)    1.7619 (3.9242x)    0.8942 (3.9112x)
  12:    2.0224 (2.9684x)    1.6827 (4.1089x)    0.8406 (4.1603x)
----------------------------------------------------------
Reference Summary
Threads  Top Down            Bottom Up           Hybrid
   1:    6.4014 (1.0000x)    7.8318 (1.0000x)    3.8490 (1.0000x)
   2:    4.2107 (1.5203x)    4.3779 (1.7889x)    2.2216 (1.7325x)
   4:    3.3601 (1.9051x)    2.5808 (3.0346x)    1.4140 (2.7221x)
   6:    3.2257 (1.9845x)    2.1347 (3.6688x)    1.2215 (3.1511x)
   8:    2.8480 (2.2477x)    2.0497 (3.8209x)    1.2122 (3.1752x)
  12:    2.3861 (2.6828x)    1.9423 (4.0322x)    1.1111 (3.4642x)
----------------------------------------------------------
For grading reference (based on execution times)

Correctness:

Timing:
Threads  Top Down      Bottom Up      Hybrid
   1:    93.78p     88.28p     90.86p
   2:    96.77p     87.17p     87.24p
   4:    87.34p     86.16p     81.24p
   6:    80.77p     85.33p     77.29p
   8:    83.55p     85.96p     73.76p
  12:    84.76p     86.63p     75.66p
```

Runtime on ghc39.ghc.andrew.cmu.edu (random_50m.graph):

```
----------------------------------------------------------
Timing Summary
Threads  Top Down               Bottom Up             Hybrid
   1:    14.6249 (1.0000x)      34.1084 (1.0000x)     5.6928 (1.0000x)
   2:    9.6571 (1.5144x)      21.0551 (1.6200x)     3.6465 (1.5611x)
   4:    6.9414 (2.1069x)      12.8242 (2.6597x)     2.8747 (1.9803x)
   6:    5.9244 (2.4686x)      15.9294 (2.1412x)     2.3804 (2.3915x)
   8:    5.5000 (2.6590x)      17.0268 (2.0032x)     2.2875 (2.4886x)
  12:    6.0930 (2.4003x)      10.3422 (3.2980x)     2.3312 (2.4420x)
----------------------------------------------------------
Reference Summary
Threads  Top Down               Bottom Up             Hybrid
   1:    15.7651 (1.0000x)      36.7491 (1.0000x)     8.7110 (1.0000x)
   2:    10.5329 (1.4967x)      25.2021 (1.4582x)     4.7253 (1.8435x)
   4:    7.5428 (2.0901x)      17.9428 (2.0481x)     3.2726 (2.6618x)
   6:    6.9795 (2.2588x)      16.3647 (2.2456x)     3.1684 (2.7493x)
   8:    6.2725 (2.5134x)      17.0879 (2.1506x)     3.2396 (2.6889x)
  12:    6.3549 (2.4808x)      16.3004 (2.2545x)     2.8149 (3.0946x)
----------------------------------------------------------
For grading reference (based on execution times)

Correctness:

Timing:
Threads  Top Down       Bottom Up       Hybrid
   1:    92.77p     92.81p     65.35p
   2:    91.68p     83.54p     77.17p
   4:    92.03p     71.47p     87.84p
   6:    84.88p     97.34p     75.13p
   8:    87.68p     99.64p     70.61p
  12:    95.88p     63.45p     82.82p
```

Runtime on Blacklight (rmat_32m.graph):

```
--------------------------------------------------------
Max system threads = 1
Running with 1 threads
--------------------------------------------------------
Loading graph...

Graph stats:
   Edges: 199491925
   Nodes: 33554432
Running with 1 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
--------------------------------------------------------
Timing Summary
Threads Top Down          Bottom Up       Hybrid
      1: 16.0188          11.6702           6.9809
--------------------------------------------------------
Reference Summary
Threads Top Down          Bottom Up       Hybrid
      1: 16.3109          14.0961           7.2650
--------------------------------------------------------


--------------------------------------------------------
Max system threads = 2
Running with 2 threads
--------------------------------------------------------
Loading graph...

Graph stats:
   Edges: 199491925
   Nodes: 33554432
Running with 2 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
--------------------------------------------------------
Timing Summary
Threads Top Down          Bottom Up       Hybrid
      2: 15.8147          10.9551           4.4316
--------------------------------------------------------
Reference Summary
Threads Top Down          Bottom Up       Hybrid
      2: 15.7932          8.1047            4.8431
--------------------------------------------------------
```

Max system threads = 4
Running with 4 threads

Loading graph...

Graph stats:
  Edges: 199491925
  Nodes: 33554432
Running with 4 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
4:	11.4126	8.6600	2.6278

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
4:	11.2683	4.0421	2.7536

Max system threads = 8
Running with8 threads

Loading graph...

Graph stats:
  Edges: 199491925
  Nodes: 33554432
Running with 8 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
8:	8.3346	7.5327	1.6313

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
8:	11.2405	2.1812	1.7437

Max system threads = 16
Running with 16 threads

Loading graph...

Graph stats:
  Edges: 199491925
  Nodes: 33554432
Running with 16 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
16:	10.6528	6.5767	1.9410

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
16:	11.3013	1.4047	1.5714

Max system threads = 32
Running with 32 threads

Loading graph...

Graph stats:
  Edges: 199491925
  Nodes: 33554432
Running with 32 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
32:	24.1170	21.5944	3.9215

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
32:	24.6346	4.6247	4.4369

Runtime on Blacklight (random_50m.graph):

\--
Max system threads = 1
Running with 1 threads
\--
Loading graph...

Graph stats:
  Edges: 499999944
  Nodes: 50000000
Running with 1 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
\--
Timing Summary

Threads	Top Down	Bottom Up	Hybrid
1:	46.3898	55.7806	11.2587

\--
Reference Summary

Threads	Top Down	Bottom Up	Hybrid
1:	44.9548	68.4181	15.2585

\--

\--
Max system threads = 2
Running with 2 threads
\--
Loading graph...

Graph stats:
  Edges: 499999944
  Nodes: 50000000
Running with 2 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
\--
Timing Summary

Threads	Top Down	Bottom Up	Hybrid
2:	36.3385	35.1577	7.7612

\--
Reference Summary

Threads	Top Down	Bottom Up	Hybrid
2:	36.2882	36.0831	10.1436

\--

```
----------------------------------------------------------
Max system threads = 4
Running with 4 threads
----------------------------------------------------------
Loading graph...

Graph stats:
   Edges: 499999944
   Nodes: 50000000
Running with 4 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
----------------------------------------------------------
Timing Summary
Threads Top Down          Bottom Up       Hybrid
    4: 21.6723            24.1341           4.8610
----------------------------------------------------------
Reference Summary
Threads Top Down          Bottom Up       Hybrid
    4: 21.5067            18.7704           5.6025
----------------------------------------------------------


----------------------------------------------------------
Max system threads = 8
Running with 8 threads
----------------------------------------------------------
Loading graph...

Graph stats:
   Edges: 499999944
   Nodes: 50000000
Running with 8 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid
----------------------------------------------------------
Timing Summary
Threads Top Down          Bottom Up       Hybrid
    8: 13.4517            15.6372           2.9507
----------------------------------------------------------
Reference Summary
Threads Top Down          Bottom Up       Hybrid
    8: 17.6449            10.1982           3.9919
----------------------------------------------------------
```

Max system threads = 16
Running with 16 threads

Loading graph...

Graph stats:
  Edges: 499999944
  Nodes: 50000000
Running with 16 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
16:	19.3297	17.6897	4.1888

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
16:	20.6625	7.5523	3.8894

Max system threads = 32
Running with 32 threads

Loading graph...

Graph stats:
  Edges: 499999944
  Nodes: 50000000
Running with 32 threads
Testing Correctness of Top Down
Testing Correctness of Bottom Up
Testing Correctness of Hybrid

Timing Summary

Threads	Top Down	Bottom Up	Hybrid
32:	46.4034	31.5984	10.0992

Reference Summary

Threads	Top Down	Bottom Up	Hybrid
32:	56.1906	39.2686	13.8575

Runtime on unix.andrew.cmu.edu (random_50m.graph):

```
----------------------------------------------------------
Timing Summary
Threads  Top Down            Bottom Up            Hybrid
   1:    15.8681 (1.0000x)    25.8125 (1.0000x)     4.4828 (1.0000x)
   2:    13.4549 (1.1794x)    17.0456 (1.5143x)     3.4020 (1.3177x)
   4:    9.1159 (1.7407x)    9.1072 (2.8343x)     2.0885 (2.1464x)
   8:    6.9225 (2.2922x)    5.2636 (4.9040x)     1.5145 (2.9599x)
  16:    5.8690 (2.7037x)    3.3598 (7.6828x)     1.2651 (3.5434x)
  32:    4.6955 (3.3794x)    3.3549 (7.6939x)     1.0765 (4.1641x)
----------------------------------------------------------
Reference Summary
Threads  Top Down            Bottom Up            Hybrid
   1:    16.3994 (1.0000x)    27.4077 (1.0000x)     5.7011 (1.0000x)
   2:    14.3532 (1.1426x)    17.7396 (1.5450x)     4.3947 (1.2973x)
   4:    10.7889 (1.5200x)    10.2806 (2.6660x)     2.8440 (2.0046x)
   8:    9.2487 (1.7732x)    6.9772 (3.9282x)     2.1605 (2.6388x)
  16:    8.2343 (1.9916x)    5.4853 (4.9966x)     1.8675 (3.0527x)
  32:    6.2013 (2.6445x)    5.2117 (5.2588x)     1.5797 (3.6091x)
----------------------------------------------------------
For grading reference (based on execution times)

Correctness:

Timing:
Threads  Top Down       Bottom Up      Hybrid
   1:    96.76p     94.18p     78.63p
   2:    93.74p     96.09p     77.41p
   4:    84.49p     88.59p     73.44p
   8:    74.85p     75.44p     70.10p
  16:    71.27p     61.25p     67.74p
  32:    75.72p     64.37p     68.15p
```