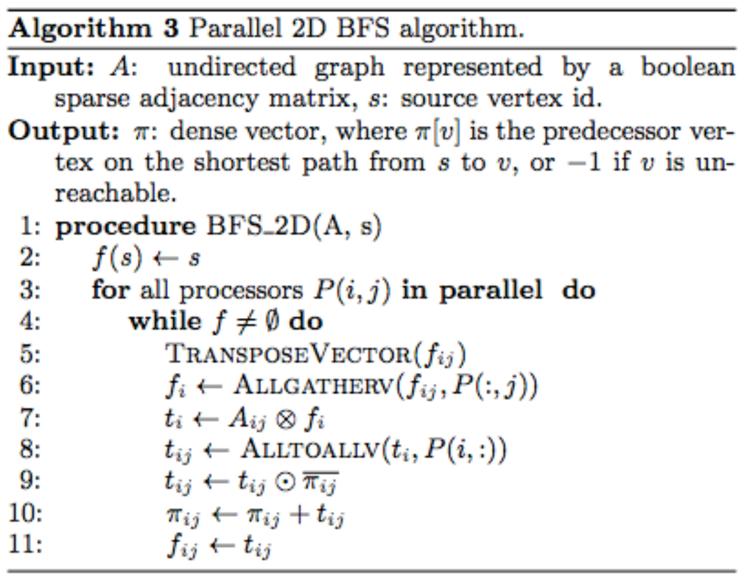
Parallel Breadth-First Search

Computing Platform

For this parallel implementation of BFS, TACC's Stampede2 is used, equipped with Intel Xeon Phi KNL compute nodes at 68 cores per KNL node and 4 hardware threads per core.

Algorithm Pseudocode



Algorithm 3 Parallel 2D BFS algorithm.

Input: A: undirected graph represented by a boolean sparse adjacency matrix, s: source vertex id.

Output: π: dense vector, where $\pi[v]$ is the predecessor vertex on the shortest path from s to v, or -1 if v is unreachable.

```
1:  procedure BFS_2D(A, s)
2:      $f(s) \leftarrow s$
3:      for all processors $P(i, j)$ in parallel do
4:          while $f \neq \emptyset$ do
5:              TRANSPOSEVECTOR($f_{ij}$)
6:              $f_i \leftarrow$ ALLGATHERV($f_{ij}, P(:, j)$)
7:              $t_i \leftarrow A_{ij} \otimes f_i$
8:              $t_{ij} \leftarrow$ ALLTOALLV($t_i, P(i, :)$)
9:              $t_{ij} \leftarrow t_{ij} \odot \overline{\pi_{ij}}$
10:             $\pi_{ij} \leftarrow \pi_{ij} + t_{ij}$
11:             $f_{ij} \leftarrow t_{ij}$
```

[1]

$\otimes$ denotes the matrix-vector multiplication operation on a special semiring, $\odot$ denotes element-wise multiplication, and overline represents the complement operation. *f* represents the current frontier, and *t* is a vector that holds the parent information for the current iteration. For a vector *v*, v_{ij} denotes the local *n/p* sized piece of the vector owned by the *P(i, j)th* processor.

Asymptotic Analysis

The parallel 2D BFS algorithm will have a number of iterations on the order of the diameter of the graph *O(D)*. For each level, each process must learn whether the vertices it owns have been added to the BFS tree. In the worst case, every vertex could try to add all of its neighbors to the tree. In this case, there will be a total of *2*E_c* messages: for a given edge e=(u,v), the owner process of u will try to add v, and the owner process of v will try to add u. *E_c* represents the number of edges across a graph partition, which in the worst case would be equal to the number of edges in the graph. This sending and receiving of messages will be the factors dominating the asymptotic performance. With this algorithm, we can then expect the time complexity to be *O(DE)*.

Machine Characteristics

Assuming the code is running with the MCDRAM on nodes configured in flat mode, Stampede2 is capable of reaching a peak bandwidth of 450 GB/s. As further discussed in the following section, bandwidth can become a bottleneck to performance as the number of edges and the number of vertices grow relative to the number of processes. In terms of network performance, the Stampede2 network provides 100 Gbits/s peak bandwidth, with point-to-point exchange performance measured at over 11 GB/s for a single task pair across nodes. As the parallel BFS algorithm relies heavily on the use of *Allgatherv* and *Alltoallv*, the speed at which communications between processors can occur will be a large determinant of performance.

Performance Bottlenecks

The adjacency matrix is decomposed across two dimensions such that there are p_r rows of processes and p_c columns, for a total of p_r x p_c processes. Therefore, a process' local adjacency matrix is of size n/p_r x n/p_c . The total memory required for the full matrix is thus the same as in the serial case. The size of the frontier, which is used as the local input, and of the output are n/p_r and n/p_c , respectively. This means that the amount of memory that a given process requires for a given time interval will be a lot larger than in the serial case. This may result in the cache not being able to hold the requested data, causing the process to wait until it can be retrieved from higher levels of cache or RAM. Therefore, we can expect that the larger the ratio of n/p becomes, the more of a bottleneck the size of the cache will be.

Aggregated over all iterations, the *AllGatherv* step input is *O(n)*, and is distributed such that each process receives $1/p_c$ sized portion. If we represent the memory access latency by α and the inverse of bandwidth by β, then the communication cost of this procedure can be represented as $p_r\, \alpha_N + (n/p_c)\, \beta_{N,ag}$. From this cost analysis, it is evident that the bandwidth will start becoming a bottleneck to performance as the ratio of n to p_c increases. During the *AlltoAllv* stage, the input can be the size of *O(m)* in the worst case. Each process will handle $1/p$ of this data, resulting in a communication cost of $p_c\, \alpha_N + (m/p)\, \beta_{N,a2a}$. For this procedure, the ratio of the number of edges to the processor count will become a hinderance to performance as the ratio grows.

References
[1] *Buluç, A. Madduri, K. Parallel Breadth-First Search on Distributed Memory Systems. 2011.*
[2] https://portal.tacc.utexas.edu/user-guides/stampede2#phase-1-compute-nodes-knl
[3] https://piazza.com/class/j6digpe6lrd35k?cid=210
[4] *Chow, Henderson, Yoo. Distributed Breadth-First Search with 2-D Partitioning.* https://pdfs.semanticscholar.org/c2b5/27d721ba6082f82e4b2f670fd2e48aa70a52.pdf