THE MORE THE MERRIER

EFFICIENT MULTI-SOURCE GRAPH TRAVERSAL

Then et Al.  Jeremy Bogle
BACKGROUND

- Graph analytics
- Multi core machines
- Graph traversal on same graph from different sources
  - Calculating graph centralities
  - Enumerating neighborhoods for all vertices
  - All-pairs shortest distance problem
BFS

- Textbook BFS
- Building block for other graph traversals
- Max levels - diameter(G)
- Random memory accesses every time it checks if a neighbor has been visited

Listing 1: Textbook BFS algorithm.

1. **Input:** $G, s$
2. $seen \leftarrow \{s\}$
3. $visit \leftarrow \{s\}$
4. $visitNext \leftarrow \emptyset$
5. 
6. while $visit \neq \emptyset$
7.   for each $v \in visit$
8.     for each $n \in neighbors_v$
9.       if $n \notin seen$
10.       \hspace{1cm} seen \leftarrow seen \cup \{n\}$
11.       $visitNext \leftarrow visitNext \cup \{n\}$
12.       do BFS computation on $n$
13.     $visit \leftarrow visitNext$
14. $visitNext \leftarrow \emptyset$
OPTIMIZING TEXTBOOK BFS

- Level by level parallelization

- Beamer et. All

  - Bottom up approach - Explores based on unvisited nodes

  - Hybrid approach - Uses bottom up for large frontiers, top up otherwise

\[
Work: \quad T_1(n) = \Theta(m+n)
\]

\[
Span: \quad T_\infty(n) = \Theta(d)
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>Tuning parameter</td>
</tr>
<tr>
<td>beta</td>
<td>Tuning parameter</td>
</tr>
<tr>
<td>m_f</td>
<td># Edges in frontier</td>
</tr>
<tr>
<td>m_u</td>
<td># Unexplored vertices</td>
</tr>
<tr>
<td>n_f</td>
<td># Vertices in frontier</td>
</tr>
<tr>
<td>n</td>
<td># Vertices</td>
</tr>
</tbody>
</table>

\[
m_f > \frac{m_u}{\alpha} = C_{TB}
\]

\[
n_f < \frac{n}{\beta} = C_{BT}
\]
MOTIVATION

- Large graphs often must be searched from various starting nodes
- Lots of overlap when executing BFS from multiple nodes
- Small world graphs have even more overlap - large fanout, each level grows rapidly

![Small World Graphs](image)

**Figure 1:** Percentage of vertex explorations that can be shared per level across 512 concurrent BFSs.

**Small World Graphs**

The distance between any two vertices is very small compared to the size of the graph, and the number of vertices discovered in each iteration of the BFS algorithm grows rapidly.
MS-BFS: EXAMPLE

- Increase the dimensionality of textbook BFS to allow for multiple BFS at once
- Shared exploration of nodes
- Finish all BFSs executions in parallel

```
Listing 2: The MS-BFS algorithm.
1 Input: G, B, S
2 seen1 ← \{b_i\} for all b_i ∈ B
3 visit ← \bigcup_{b_i ∈ B} \{(s_i, \{b_i\})\}
4 visitNext ← ∅
5 while visit ≠ ∅
6   for each v in visit
7     B' v ← ∅
8     for each (v', B') ∈ visit where v' = v
9       B' v ← B' v ∪ B'
10      for each n ∈ neighbors_v
11         D ← B' v \ seen
12         if D ≠ ∅
13           visitNext ← visitNext ∪ \{(n, D)\}
14           seen_n ← seen_n ∪ D
15         do BFS computation on n
16     visit ← visitNext
17     visitNext ← ∅
```
OPTIMIZATIONS FOR MS-BFS

- Bit operations
- Aggregated neighbor processing
- Direction optimized
- Neighbor prefetching
- Sharing heuristic
OPTIMIZATIONS FOR MS-BFS

Listing 2: The MS-BFS algorithm.

Input: $G, \mathcal{B}, S$

seen$_{si} \leftarrow \{b_i\}$ for all $b_i \in \mathcal{B}$

visit $\leftarrow \bigcup_{b_i \in \mathcal{B}} \{(s_i, \{b_i\})\}$

visitNext $\leftarrow \emptyset$

while visit $\neq \emptyset$

for each $v$ in visit

for each $(v', \mathcal{B}') \in \text{visit where } v' = v$

$\mathcal{B}' \leftarrow \mathcal{B}' \cup \mathcal{B}'$

for each $n \in \text{neighbors}_v$

$\mathcal{D} \leftarrow \mathcal{B}' \setminus \text{seen}_n$

if $\mathcal{D} \neq \emptyset$

visitNext $\leftarrow \text{visitNext} \cup \{(n, \mathcal{D})\}$

seen$_n \leftarrow \text{seen}_n \cup \mathcal{D}$

do BFS computation on $n$

visit $\leftarrow \text{visitNext}$

reset visitNext

visitNext $\leftarrow \emptyset$

Listing 3: MS-BFS using bit operations.

Input: $G, \mathcal{B}, S$

for each $b_i \in \mathcal{B}$

seen[$s_i$] $\leftarrow 1 << b_i$

visit[$s_i$] $\leftarrow 1 << b_i$

reset visitNext

while visit $\neq \emptyset$

for $i = 1, \ldots, N$

if visit[$v_i$] = $\mathcal{B}_\emptyset$, skip

for each $n \in \text{neighbors}_v$

if $\mathcal{D} \neq \mathcal{B}_\emptyset$

visitNext[$n$] $\leftarrow \text{visitNext}[n] \mid \mathcal{D}$

seen[$n$] $\leftarrow \text{seen}[n] \mid \mathcal{D}$

do BFS computation on $n$

visit $\leftarrow \text{visitNext}$

reset visitNext

Listing 4: MS-BFS algorithm using ANP.

Input: $G, \mathcal{B}, S$

for each $b_i \in \mathcal{B}$

seen[$s_i$] $\leftarrow 1 << b_i$

visit[$s_i$] $\leftarrow 1 << b_i$

reset visitNext

while visit $\neq \emptyset$

for $i = 1, \ldots, N$

if visit[$v_i$] = $\mathcal{B}_\emptyset$, skip

for each $n \in \text{neighbors}_v$

if visitNext[$v_i$] = $\mathcal{B}_\emptyset$, skip

seen[$v_i$] $\leftarrow \text{seen}[v_i]$ $\mid \text{visitNext}[v_i]$

if visitNext[$v_i$] $\neq \mathcal{B}_\emptyset$

do BFS computation on $v_i$

visit $\leftarrow \text{visitNext}$

reset visitNext
EVALUATION AND RESULTS

- Running BFS from all nodes as number of vertices increases
- Traversed edges per second
- Improvement benefits from various optimizations
STRENGTHS

- Comparison with existing approaches
- Leverage existing optimizations
- Large scale evaluations
WEAKNESSES

- Must be overlapping during the same iterations
  - No “memory” of previously searched nodes
- Evaluation on non “small-world” graphs
- Perform optimizations independently
- Evaluation in a distributed system
- MS-BFS with parallelization at each level
DISCUSSION

▸ What did you guys think were the strengths and weaknesses?

▸ On what types of graphs is MS-BFS NOT useful
  ▸ How could it be improved to be useful on these graphs?
  ▸ How does MS-BFS perform compared to textbook BFS in these scenarios