The More the Merrier: Efficient Multi-Source Graph Traversal

Manuel Then* then@in.tum.de
Moritz Kaufmann* kaufmanm@in.tum.de
Fernando Chirigati† fchirigati@nyu.edu
Tuan-Anh Hoang-Vu† tuananh@nyu.edu
Kien Pham† kien.pham@nyu.edu
Huy T. Vo† huy.vo@nyu.edu
Alfons Kemper* kemper@in.tum.de
Thomas Neumann* neumann@in.tum.de

* Technische Universität München † New York University

Motivation for Multi-Source Traversal

Many graph algorithms run **multiple breadth-first searches**

- Shortest paths
- Closeness centrality
- K-hop neighborhoods

...

+

Real-world graphs are often **small-world networks**

- Social networks
- Web graphs
- Communication networks

...

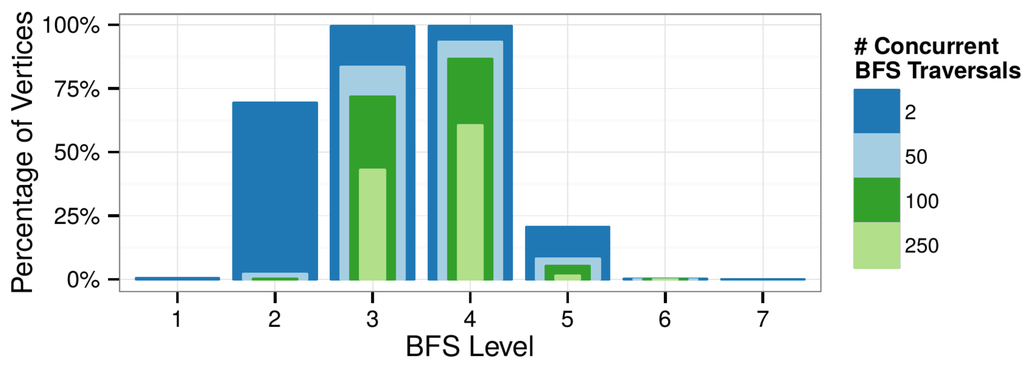
Goals

Leverage multi-source traversal in graph algorithms.

Optimize data access patterns for this kind of traversal and **avoid redundant computation** to **improve** graph analytics **performance.**

Challenges

- Traversals require **random data accesses** with **bad cache behavior** and often cause **CPU stalls**
- Single bit accesses **waste memory bandwidth**
- Independent BFS runs **redundantly visit vertices multiple times**

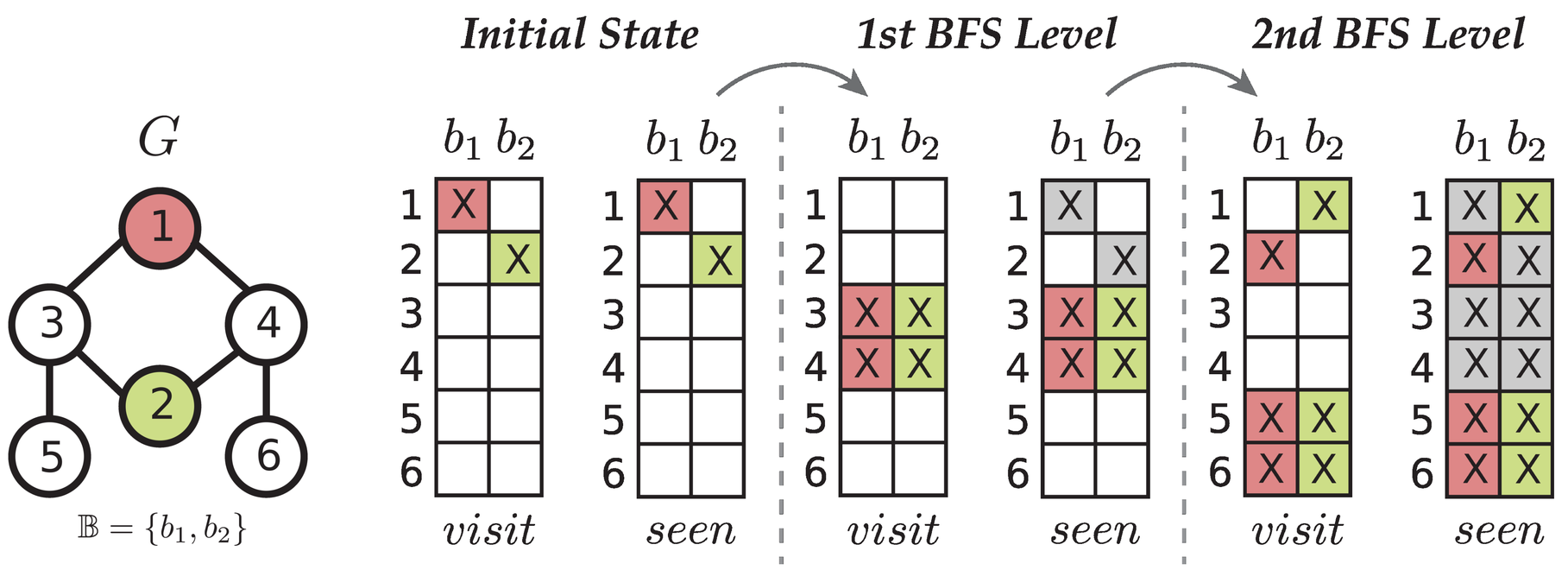


Multi-Source BFS (MS-BFS)

- Concurrently run **many independent BFS traversals** on the **same graph**
- **Share traversals** whenever possible

- Store **BFS state** per vertex **as a bitset** (3 bits per vertex and traversal)
- Represent BFS as **SIMD bit operations**
- **Fully utilize** cache line-sized **memory accesses** of modern CPUs

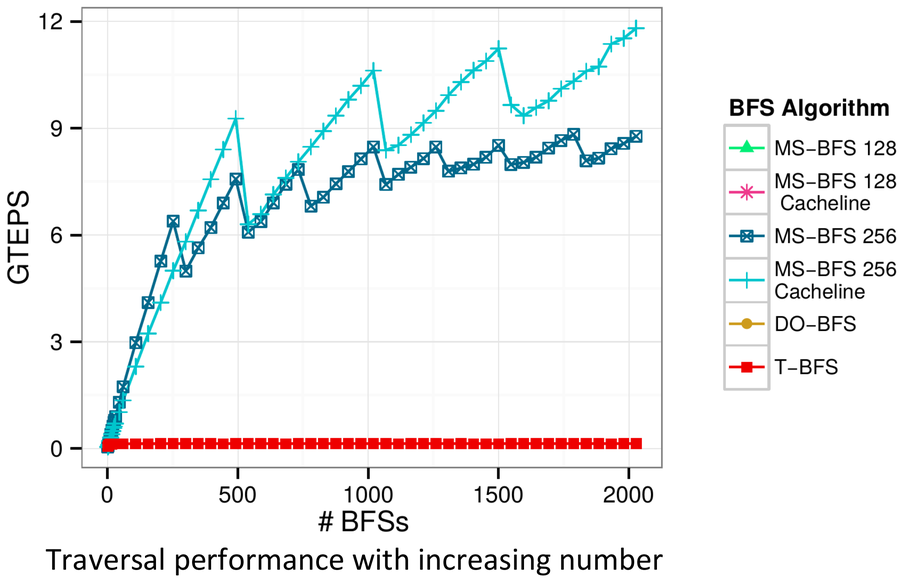
$$
\begin{aligned}
&\textbf{for } i = 1, \ldots, N \\
&\quad \textbf{if } visit[v_i] = \mathbb{B}_\varnothing\textbf{: skip} \\
&\quad \textbf{for each } n \in neighbors[v_i] \\
&\qquad \mathbb{D} \leftarrow visit[v_i] \ \&\ \sim seen[n] \\
&\qquad \textbf{if } \mathbb{D} \neq \mathbb{B}_\varnothing \\
&\qquad\quad visitNext[n] \leftarrow visitNext[n] \mid \mathbb{D} \\
&\qquad\quad seen[n] \leftarrow seen[n] \mid \mathbb{D}
\end{aligned}
$$



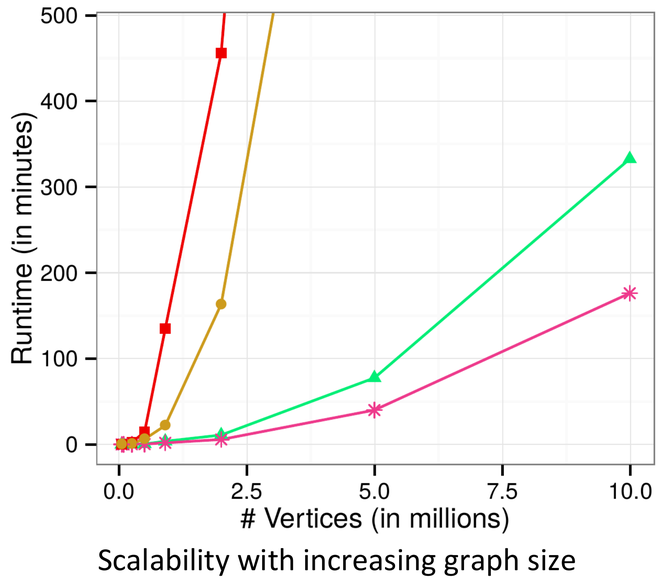
Further improvements: **Aggregated neighbor processing**, Direction-optimizing, **Prefetching**, Batching for **maximum sharing**

Evaluation

MS-BFS-based closeness centrality $\frac{(C_v - 1)^2}{(N-1) * \Sigma_{u \in V} d(v,u)}$ on 4x Intel Xeon E7-4870v2, 1TB memory



Traversal performance with increasing number of sources - The more the merrier



Scalability with increasing graph size

Graph	MS-BFS	Speedup over T-BFS	Speedup over DO-BFS
LDBC 1M	0:02h	73.8x	12.1x
LDBC 10M	2:56h	88.5x	28.7x
Wikipedia	0:26h	75.4x	29.5x
Twitter (1M)	2:52h	54.6x	12.7x

Runtime and speedup of MS-BFS compared to state-of-the-art competitors

Source available at
https://github.com/mtodat/ms-bfs