Ligra: A Lightweight Graph Processing Framework for Shared Memory

(Julian Shun and Guy E. Blelloch 2013)

Presented by Armins B. Stepanjans

Core Contributions

A simple interface to the optimised framework.

A framework for fast graph computation on multicore systems.

Competitive experimental results.

What problem is the paper solving?

Processing large graphs

On multiple parallel workers with shared memory

Many CPUs and/or cores (paper uses 40)

Interface is Beautifully Simple

```
vertexSubset

size

edgeMap

vertexMap
```

Vertex Subset

[3, 4, 2] List of vertex indices (*any* order)

[0, 0, 1, 1, 1, 0, 0, 0] List of booleans

Size

$\textbf{SIZE}(U : \mathit{vertexSubset}) : \mathbb{N}$.

Returns $|U|$.

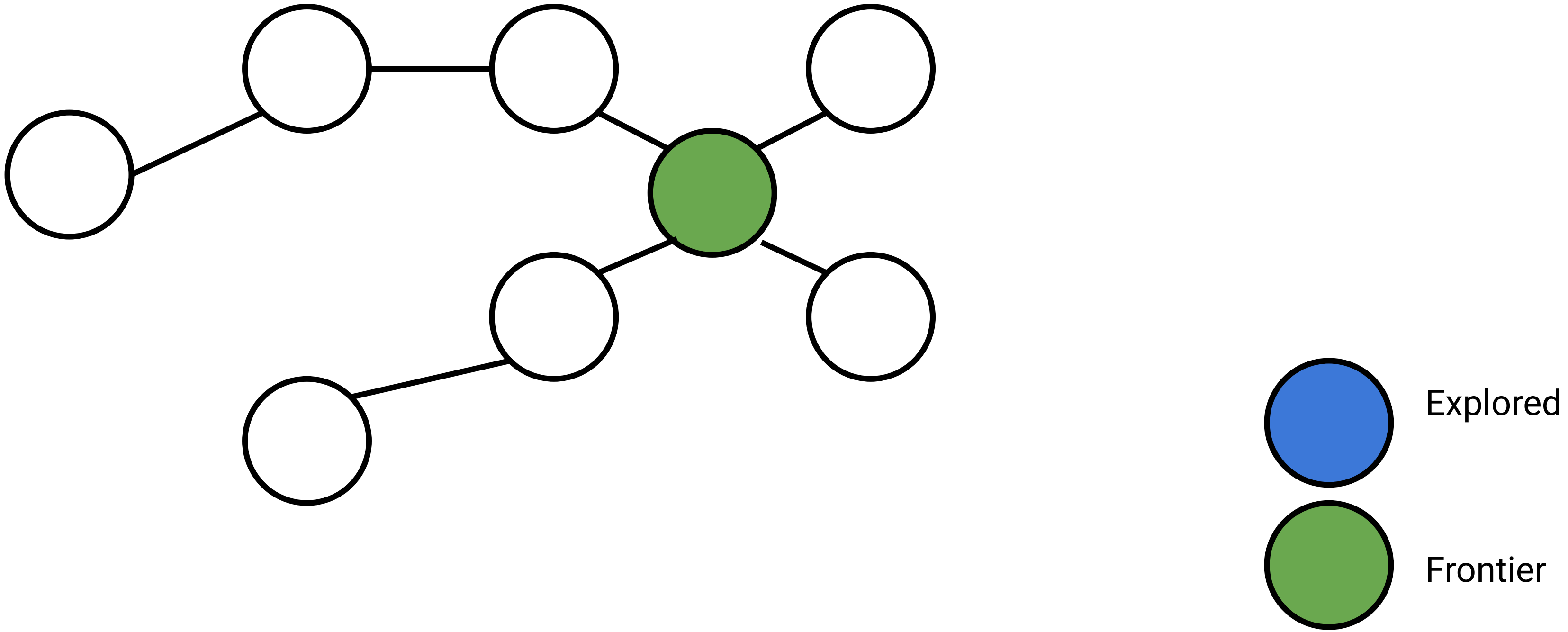
Edge Map

$$
\begin{aligned}
\textsc{edgeMap}(G &: \textit{graph},\\
U &: \textit{vertexSubset},\\
F &: (\textit{vertex} \times \textit{vertex}) \mapsto \textit{bool},\\
C &: \textit{vertex} \mapsto \textit{bool}) : \textit{vertexSubset}.
\end{aligned}
$$

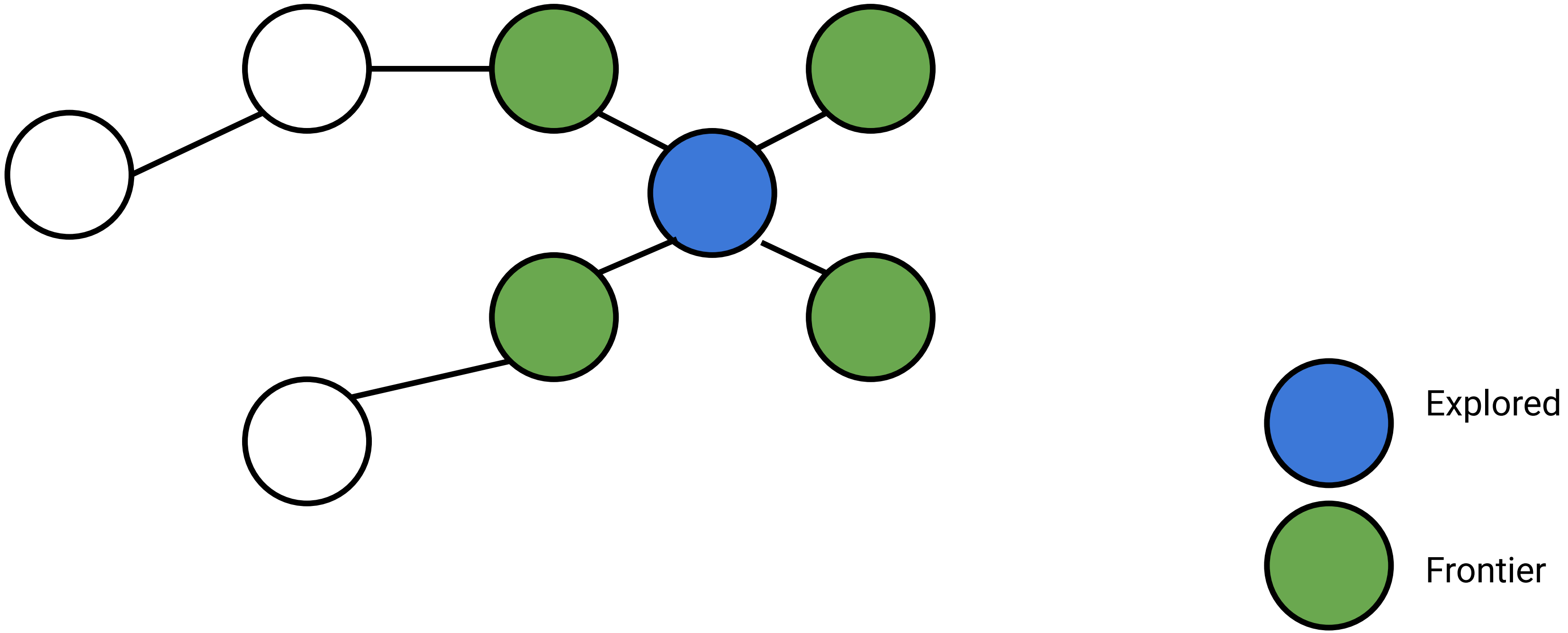
Vertex Map

$$\textsc{vertexMap}(U : \mathit{vertexSubset},\\ F : \mathit{vertex} \mapsto \mathit{bool}) : \mathit{vertexSubset}.$$

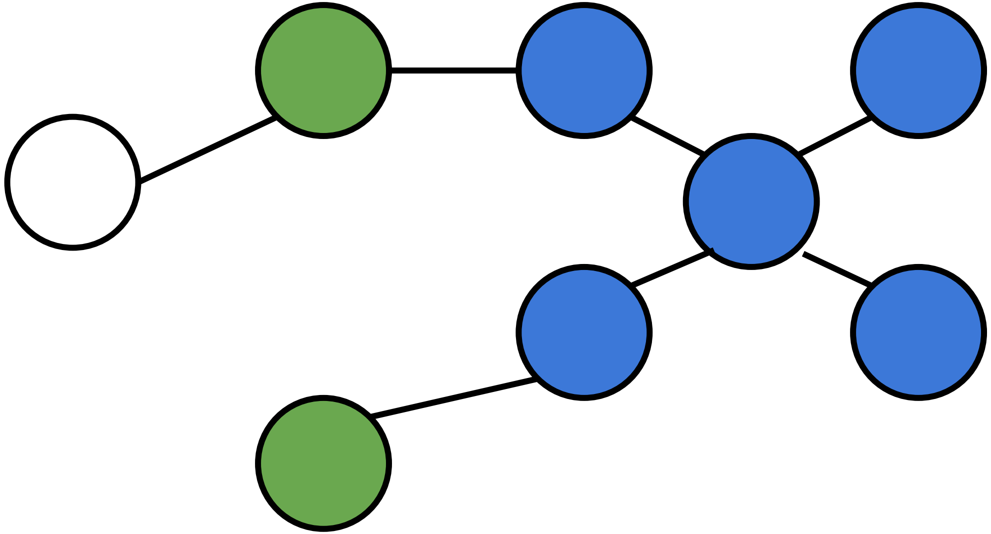
Breadth First Search step 1

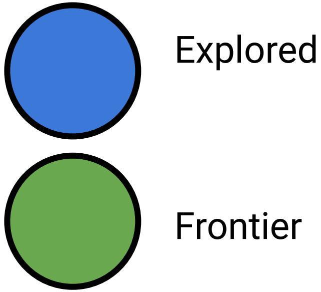


Breadth First Search step 2

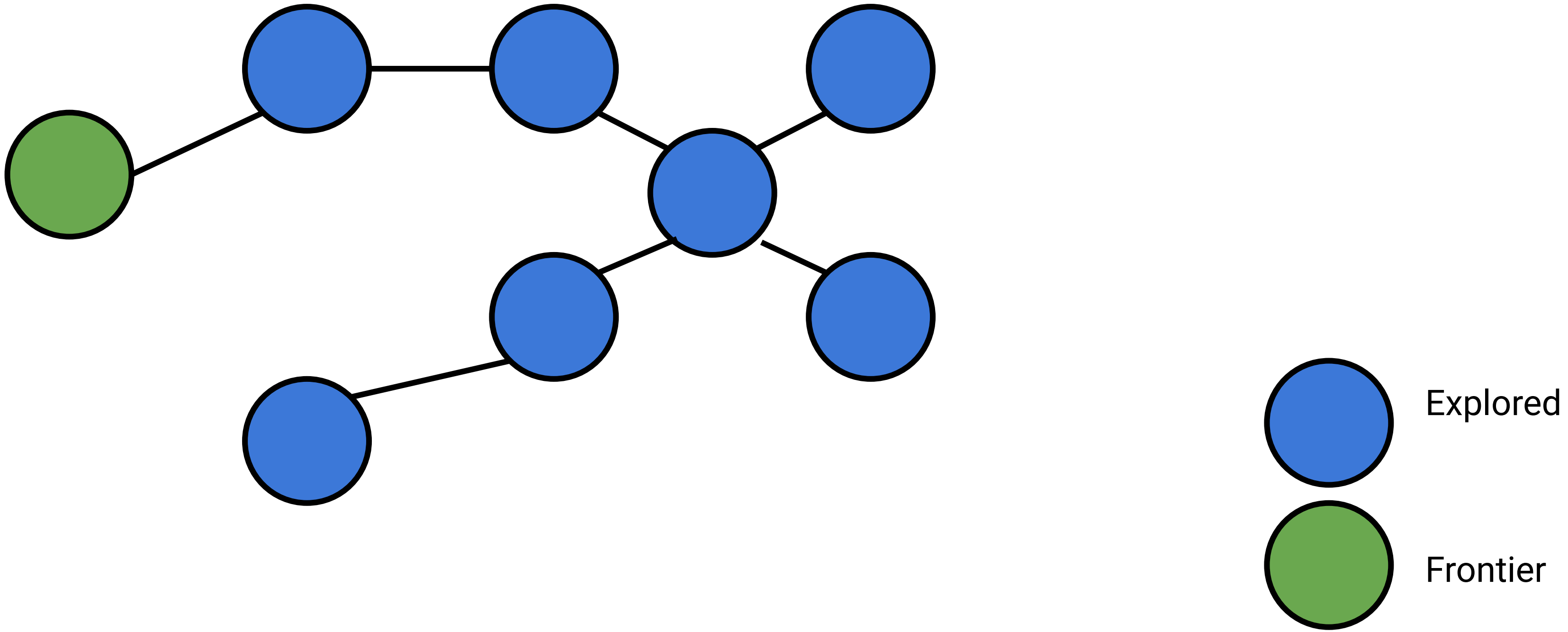


Breadth First Search step 3

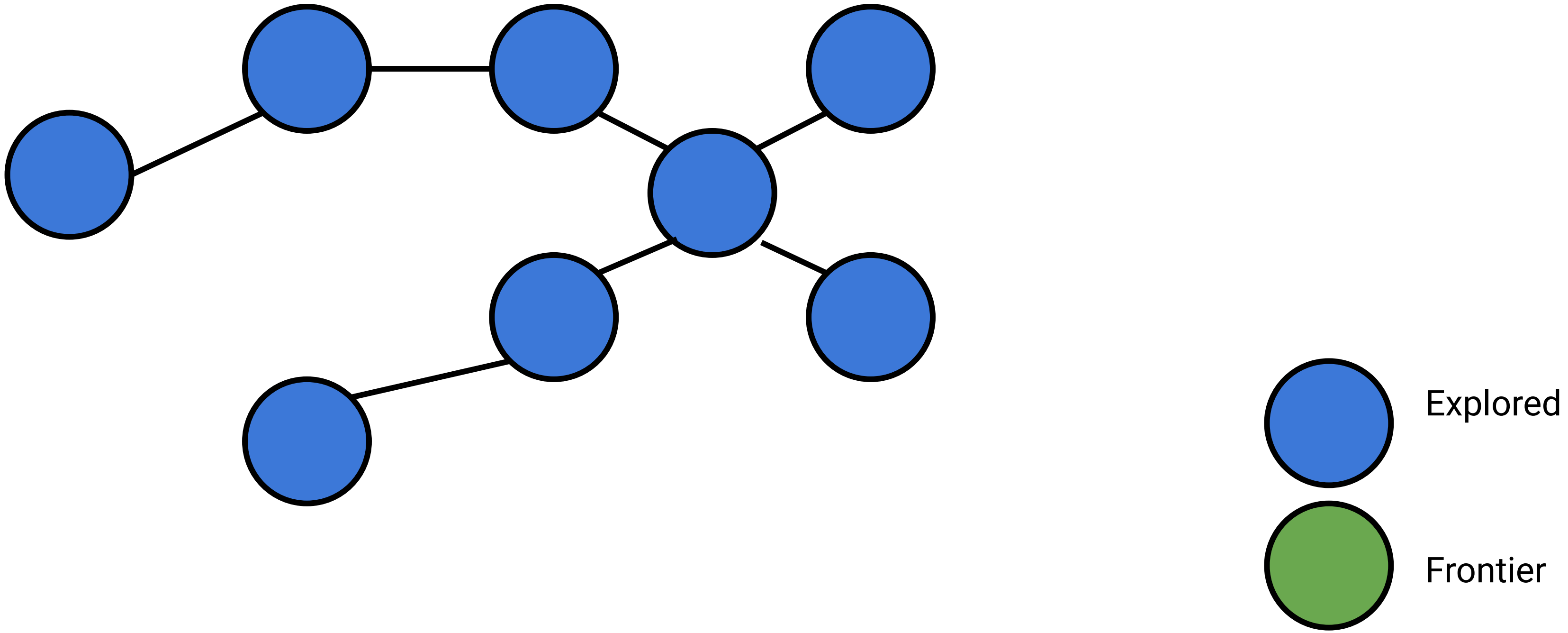




Breadth First Search step 4



Breadth First Search step 5



BFS in Ligra

```
Parents = {−1, . . . , −1}                                  ▷ initialized to all -1's

procedure UPDATE(s, d)
    return (CAS(&Parents[d], −1 , s ))

procedure COND(i)
    return (Parents[i] == −1)

procedure BFS(G, r)                                          ▷ r is the root
    Parents[r] = r
    Frontier = {r}                ▷ vertexSubset initialized to contain only r
    while (SIZE(Frontier) ≠ 0) do
        Frontier = EDGEMAP(G, Frontier, UPDATE, COND)
```

Optimised graph computation

Following Beamer et al. (2011) sparse or dense representations are used depending on the subgraph.

Algorithm 1 EDGEMAP

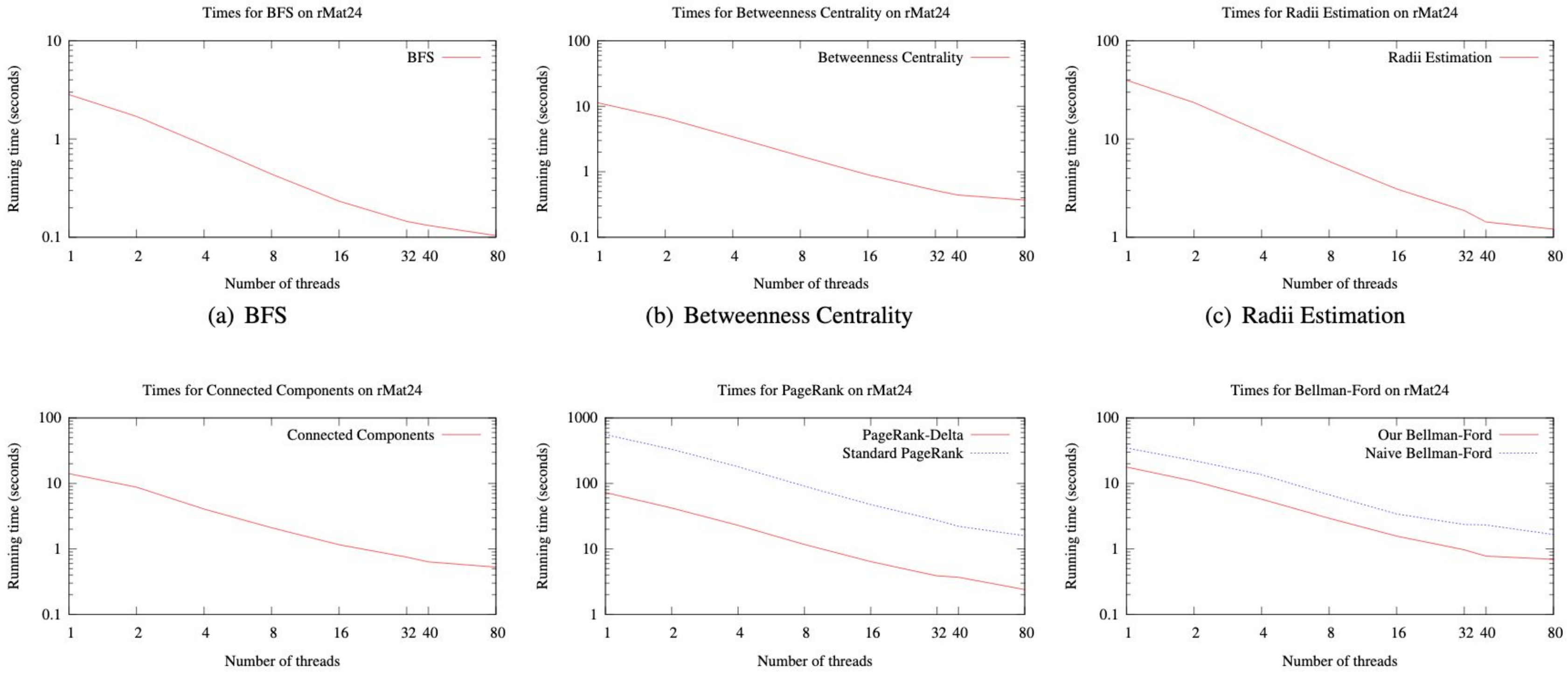
procedure EDGEMAP(G, U, F, C)
  if ($|U|$ + sum of out-degrees of $U >$ threshold) **then**
    return EDGEMAPDENSE(G, U, F, C)
  else return EDGEMAPSPARSE(G, U, F, C)

Evaluation Setting

Evaluated on 40 cores (4x10 core CPUs) with 256GB RAM 1066MHz bus

Input	Num. Vertices	Num. Directed Edges
3D-grid	10^7	6×10^7
random-local	10^7	9.8×10^7
rMat24	1.68×10^7	9.9×10^7
rMat27	1.34×10^8	2.12×10^9
Twitter	4.17×10^7	1.47×10^9
Yahoo*	1.4×10^9	12.9×10^9

Close to linear parallelization



(a) BFS (b) Betweenness Centrality (c) Radii Estimation

(d) Connected Components (e) PageRank (f) Bellman-Ford

Comparative performance

Ligra v GPS (30 instances each with 4 virtual cores 7.5GB RAM) on Yahoo PageRank (20s v. 104s)

Ligra v. PowerGraph (8x64 cores) on Twitter PageRank (2.91s v. 3.6s)

Ligra v. Pregel (300 commodity PCs) Bellman-Ford on 1B vertex binary tree (2s v. 20s)

Shortcomings

Requires a very specific configuration
"We also ran experiments on a 64-core AMD Opteron machine, but the results are slower than the ones from the Intel machine"

Framework doesn't handle insertions/deletions

Framework doesn't handle loading and caching of values associated with the graph.

Since 2013 a lot has been built on top of Ligra

Ligra+ for compressed graphs (2015)

Julienne for bucketting (2017)

Aspen for updating graphs (2019)

Extension to hypergraphs (2020)

Ligra is *efficient* and *simple* parallel graph processing framework

But
it demands specific hardware structure

doesn’t handle writes

and it is up to the user to maintain associated data.

Please ask questions!

Thank you for your time!