Ligra: A Lightweight Graph Processing Framework for Shared Memory

Julian Shun and Guy E. Blelloch (2013)

(Or How to Win a Prestigious ACM Prize)

Presented by Timi Adeniran

How does one process a graph with 20 billion edges?

According to Pregel:

According to Pregel:

Scale out!

According to Powergraph:

According to Powergraph:

Scale out!

Before Ligra

Predominant approach was to distribute load among many machines.

Before Ligra

- Predominant approach was to distribute load among many machines.

- This model worked:
    - Google created Pregel (2010) for large-scale graph processing.
    - PowerGraph outperformed SOTA graph-parallel frameworks.

Can we do better?

Key Insights

A single commodity machine can fit billions of edges in memory.

Key Insights

- Commodity machines can fit billions of edges in memory.

Their many cores can be used for parallel processing

Key Insights

- Commodity machines can fit billions of edges in memory.
- Their many cores can be used for parallel processing.

Communication is cheaper on shared memory systems.

Key Insights

- Commodity machines can fit billions of edges in memory.
- Their many cores can be used for parallel processing.
- Communication is cheaper on shared memory systems.

Most graph algorithms only work on a small subset at a time.

Key Insights

- Commodity machines can fit billions of edges in memory.
- Their many cores can be used for parallel processing.
- Communication is cheaper on shared memory systems.
- Most graph algorithms only work on a small subset at a time.

Graph processing on a single commodity machine can be more efficient.

Ligra is a framework for shared memory machines

Ligra

- Simple API:
    - Operates on a \`vertexSubset\`.
    - Provides \`edgeMap\` and \`vertexMap\` functions.
- Parallelizes graph operations.
- Has optimizations for sparse and dense graphs.

BFS Pseudocode

```
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)
```

Evaluation

- Ran experiments on a 40-core Intel machine with 256GB of main memory.

- Achieved better performance than SOTA for common graph algorithms: BFS, PageRank, Connected Components, etc.

Comparison with PowerGraph

- PowerGraph setup: 8 machines with 64 cores each.
- PageRank on Twitter graph: 41.7m vertices and 1.47b edges.
- Ligra outperforms PowerGraph: 2.91s vs 3.6s per iteration.

Comparison with GPS

- GPS setup: 30 instances with 4 cores each and 7.5GB memory.
- PageRank on a graph with 3.7 billion edges took 86s per iteration.
- PageRank on a Yahoo graph with more edges (6.6 billion) took <20s per iteration in Ligra.

Paper Observations (1)

Interesting evaluation choices:

“We also ran experiments on a 64-core AMD Opteron machine, but the results are slower than the ones from the Intel machine so we only report the latter”

Paper Observations (2)

Interesting evaluation choices:

> "The single-source shortest paths algorithm of Pregel [34] for **a binary tree with 1 billion vertices** takes almost 20 seconds on a cluster of 300 multicore commodity PCs. We ran our Bellman-Ford algorithm on a **larger binary tree with 2^{27}(≈ 1.68 × 10^7) vertices**, and it completed in under 2 seconds"

$$1 \text{ billion} >> 2^{27} > 1.68 \times 10^7$$

Limitations

What if I don’t have have a large machine?

Limitations

What if I don’t have a large machine?

Ligra+ (2015) compresses the graph.

Limitations

- What if I don't have a large machine?
  - Ligra+ (2015) compresses the graph.

What about dynamic graphs?

Limitations

- What if I don't have a large machine?
  - Ligra+ (2015) compresses the graph.

- What about dynamic graphs?
  - Aspen (2019) extends Ligra+ to support dynamic graphs.

Impact (1)

- Has influenced many other systems:
  - Polymer (2015): Zhang et al. “NUMA-Aware Graph-Structured Analytics,” PPOPP 2015.
  - Gemini (2016): Zhu et al. “Gemini: A Computation-Centric Distributed Graph Processing System, ” OSDI 2016
  - Aspen (2019): Dhulipala et al. “Low-Latency Graph Streaming using Compressed Purely-Functional Trees, ” PLDI 2019.
  - Kairos (2023): da Trindade et al. “Kairos: Efficient Temporal Graph Analytics on a Single Machine,” NEDB 2023

Impact (2)

- So good it won its authors an award:
  - 2023 ACM Paris Kanellakis Theory and Practice Award "for contributions to algorithm engineering, including the Ligra, GBBS, and Aspen frameworks which revolutionized large-scale graph processing on shared-memory machines."
  - "One important upshot of this work was the paradigm-changing demonstration that shared-memory computers are an ideal platform for analyzing large graphs. **At the time Ligra was first developed, the predominant approach used to analyze large graphs was distributed systems such as Pregel (developed by Google)**. This was overturned when, for many important large real-world graph problems, **the Ligra approach turned out to be much more efficient in terms of energy, cost, and end-to-end running time."**

Questions?