UNIVERSITY OF CAMBRIDGE

Ligra: A Lightweight Graph Processing Framework for Shared Memory

J. Shun and G. Blelloch

Presented by Dmitry Kazhdan

Overview

- Background (briefly)
- Existing work
- Key contributions
- Results
- Criticism
- Conclusions
- Questions

Background

Background

Motivation:

- Efficient Graph Processing

Opportunities:

- Parallelism
- Large number of cores + RAM in a single server

Related Work

- Pregel
- PowerGraph
- Green-Marl
- X-Stream
- Pegasus
- GraphLab
- …

Ligra

Ligra

- Graph processing framework
- Relying on multicore machines with shared memory
- Offering parallel processing

Contributions

- Implemented in memory (single machine)
- Lightweight (contains a few thousand lines of C++)
- Easily extendable/customisable
- Minimal (offering a small number of primitives/abstractions)
- Ligra+ offers graph compression

Abstractions

- Graph and VertexSubset datatypes.
- EdgeMap and VertexMap functions.

Algorithm 4 VERTEXMAP

```
1: procedure VERTEXMAP(U, F)
2:     Out = {}
3:     parfor u ∈ U do
4:         if (F(u) == 1) then Add u to Out
5:     return Out
```

Algorithm 2 EDGEMAPSPARSE

```
1: procedure EDGEMAPSPARSE(G, U, F, C)
2:     Out = {}
3:     parfor each v ∈ U do
4:         parfor ngh ∈ N+(v) do
5:             if (C(ngh) == 1 and F(v, ngh) == 1) then
6:                 Add ngh to Out
7:     Remove duplicates from Out
8:     return Out
```

Use Cases

- Breadth-first Search
- Betweenness Centrality
- Graph Radii Estimation
- Connected Components
- Page Rank
- Bellman-Ford Shortest Paths

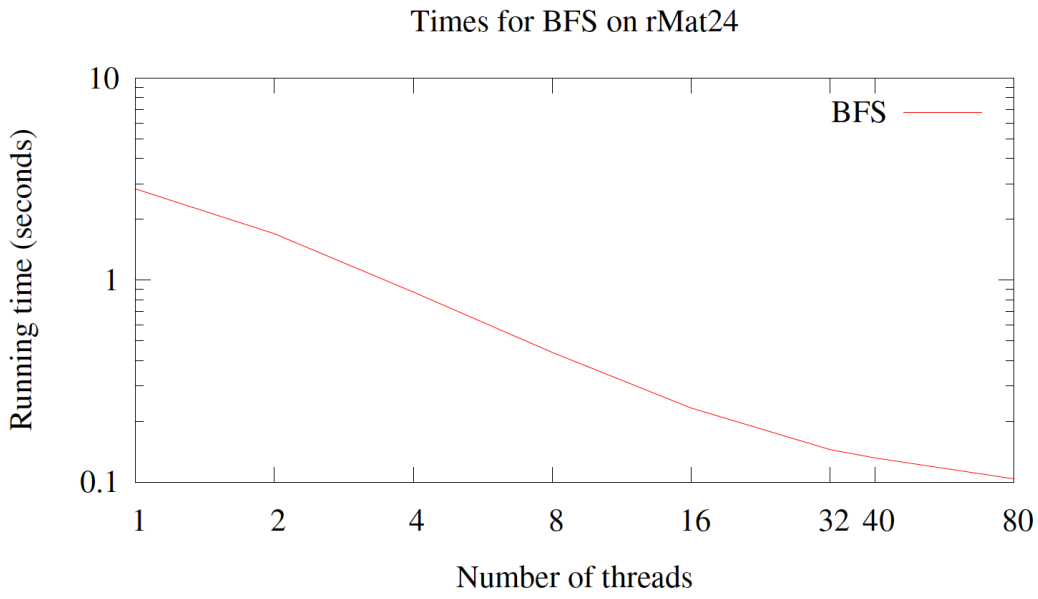
Algorithm 8 Connected Components

```
1:  IDs = {0, ..., |V| − 1}                    ▷ initialized such that IDs[i] = i
2:  prevIDs = {0, ..., |V| − 1}                ▷ initialized such that prevIDs[i] = i
3:
4:  procedure CCUPDATE(s, d)
5:      origID = IDs[d]
6:      if (WRITEMIN(&IDs[d], IDs[s])) then
7:          return (origID == prevIDs[d])
8:      return 0
9:
10: procedure COPY(i)
11:     prevIDs[i] = IDs[i]
12:     return 1
13:
14: procedure CC(G)
15:     Frontier = {0, ..., |V| − 1}           ▷ vertexSubset initialized to V
16:     while (SIZE(Frontier) ≠ 0) do
17:         Frontier = VERTEXMAP(Frontier, COPY)
18:         Frontier = EDGEMAP(G, Frontier, CCUPDATE, C_true)
19:     return IDs
```

Evaluation

Evaluation

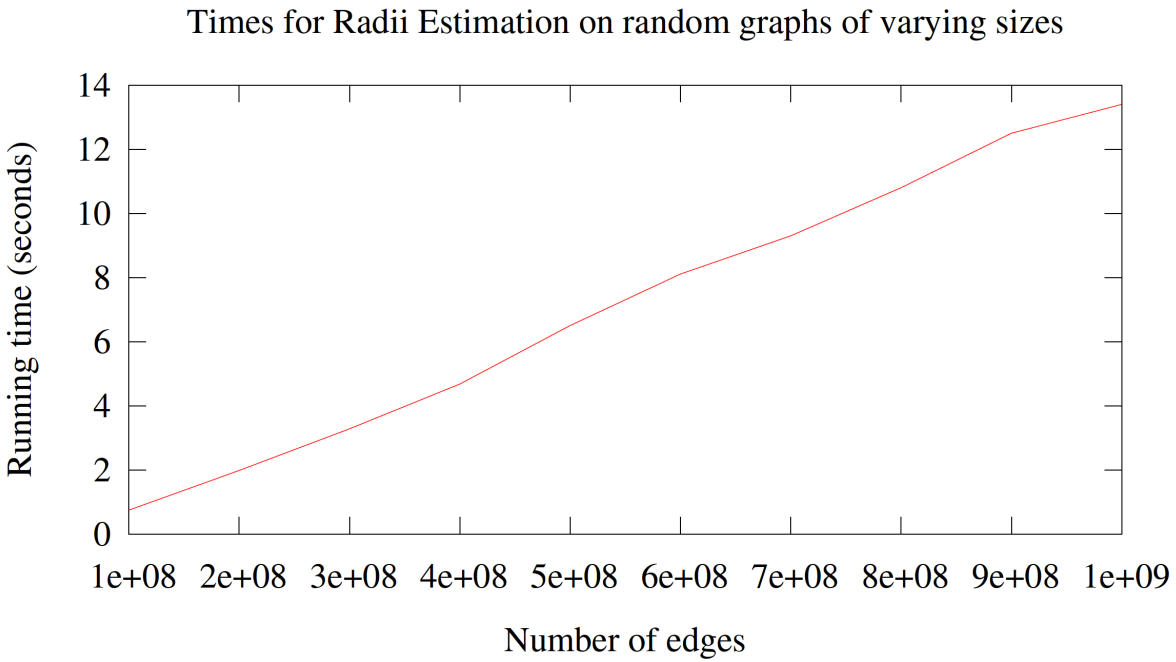
- Evaluated algorithm performance on set of selected graphs
- Showed effects of using more threads
- Gave performance comparisons to other systems (albeit brief, often using different setups)



(a) BFS

Evaluation

- Showed system scalability on randomly generated graphs



(c) Radii Estimation

Resources

http://jshun.github.io/ligra/index.html

https://github.com/jshun/ligra

Criticism

Criticism/Discussion

"and our code is much simpler than theirs"

```
class PageRankVertex
    : public Vertex<double, void, double> {
 public:
  virtual void Compute(MessageIterator* msgs) {
    if (superstep() >= 1) {
      double sum = 0;
      for (; !msgs->Done(); msgs->Next())
        sum += msgs->Value();
      *MutableValue() =
          0.15 / NumVertices() + 0.85 * sum;
    }

    if (superstep() < 30) {
      const int64 n = GetOutEdgeIterator().size();
      SendMessageToAllNeighbors(GetValue() / n);
    } else {
      VoteToHalt();
    }
  }
};
```

Figure 4: PageRank implemented in Pregel.

Algorithm 9 PageRank

1: $p_{curr} = \{\frac{1}{|V|}, \ldots, \frac{1}{|V|}\}$ ▷ initialized to all $\frac{1}{|V|}$
2: $p_{next} = \{0.0, \ldots, 0.0\}$ ▷ initialized to all 0.0
3: diff $= \{\}$ ▷ array to store differences
4:
5: **procedure** PRUPDATE(s, d)
6: ATOMICINCREMENT($\&p_{next}[d]$, $\frac{p_{curr}[s]}{deg^+(s)}$)
7: **return** 1
8:
9: **procedure** PRLOCALCOMPUTE(i)
10: $p_{next}[i] = (\gamma \times p_{next}[i]) + \frac{1-\gamma}{|V|}$
11: $\text{diff}[i] = |p_{next}[i] - p_{curr}[i]|$
12: $p_{curr}[i] = 0.0$
13: **return** 1
14:
15: **procedure** PAGERANK(G, γ, ϵ)
16: Frontier $= \{0, \ldots, |V| - 1\}$ ▷ vertexSubset initialized to V
17: error $= \infty$
18: **while** (error $> \epsilon$) **do**
19: Frontier = EDGEMAP(G, Frontier, PRUPDATE, C_{true})
20: Frontier = VERTEXMAP(Frontier, PRLOCALCOMPUTE)
21: error = sum of diff entries
22: SWAP(p_{curr}, p_{next})
23: **return** p_{curr}

Criticism/Discussion

- How representative are the examples?
- Too much time spent describing algorithms

Criticism/Discussion

- Single experimental setup (“but the results are slower than the ones from the Intel machine so we only report the latter”)
- Performance comparisons not detailed or not meaningful (“we achieved faster results”)

Conclusions

Conclusions

- Introduced Ligra
- Compared to existing systems
- Presented evaluation results
- Criticism

Questions?