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
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

- 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:    for u ∈ U do
4:        if (F(u) == 1) then Add u to Out
5:    return Out

Algorithm 2 EDGE MAP SPARSE

1: procedure EDGE MAP SPARSE(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
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)
• Showed system scalability on randomly generated graphs
Resources

http://jshun.github.io/ligra/index.html
https://github.com/jshun/ligra
Criticism
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.
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?