MapReduce is 10 years old...
...what's next?

Sergei Vassilvitskii

Google

MapReduce: Overview

"MapReduce is a framework for processing embarrassingly parallel problems across huge datasets using a large number of computers" [Wikipedia]

In practice:

- Initially developed for dealing with the web graph
  - Building inverted indexes, computing pagerank, log processing
- Now widely deployed:
  - Google, Yahoo, FB, Microsoft, Netflix, NYTimes, ...)
  - Easy to access through Amazon Elastic Computing (AWS, EC2)
- Used far beyond building inverted indexes:
  - Machine learning, recommendations, data analysis,

Why MapReduce

Simple:

- Easy to understand model (takes 10 minutes to explain)
- Good level of abstraction (can't do everything, but don't have to think about concurrency!)

Scalable:

- Just add more hardware!
- Although...interesting scheduling challenges emerge

Robust:

- Nodes will fail!
  - but the system automatically checkpoints & restarts jobs as needed
- Allows users to think about algorithmics and not reliability

Thinking about MR

Modeling MapReduce

- Series of papers: [FMSSS '08], [KSV '10], [GSZ '12], [PPRSU '12]
- Converging towards a model:
  - Sublinear number of machines
  - Sublinear memory
  - Aim for constant number of rounds (expose memory/rounds tradeoffs)

MR: Known Knowns

Easily parallelizable questions

- Counting & statistics: sorting, median, moments, etc.

Graph algorithms

- Matchings, connectivity (when all nodes fit in memory)
- Triangle counting, densest subgraph (all graphs)

Other:

- Greedy algorithms (set cover, other submodular optimizations)
- Clustering: k-{median, means, center}, EM

MR: Known Unknowns

The exact power of the model:

- Known: can simulate PRAMs round per round with MR [KSV '10]
- Unknown: strong lower bounds
  - First progress: dense matrix multiplication [PPRSF '12]
- Unknown: sparse graph problems
  - e.g. CCs on 2-regular graphs (count the number of cycles)
  - See me offline for a lower bound puzzle

Beyond MapReduce: Graphs

The MapReduce paradigm is seemingly not very good for graph computation

Beyond MapReduce: Graphs

The MapReduce paradigm is seemingly not very good for graph computation

Working with graphs:

- Imagine a simple machine at every node, (synchronously) passing messages to other nodes.
- BFS, SSSP are very easy in this view
  - But can they be even better?
- In practice:
  - Pregel[Google], Giraph[Yahoo/Open source], GraphLab
  - Again: simple, scalable, robust

Beyond MR: Unknown Unknowns

What is the shape of large scale parallel computation?

- Must be:
  - Simple, scalable, robust
- Needs to play well with others:
  - Multiple “paradigms” sharing the same set of resources (same cluster)

What kind of computation?

- Past: MapReduce – simple batch computation
- Present: Graph algorithms via message passing along edges
- Future: Repeated & fixed point computations

Thank You

sergeiv@google.com