Distributed Data: Problems from Social Networks

Ashish Goel

Stanford University

Data Model #1: Map Reduce

- An immensely successful idea which transformed offline analytics and bulk-data processing. Hadoop (initially from Yahoo!) is the most popular implementation.

- **MAP:** Transforms a (key, value) pair into other (key, value) pairs using a UDF (User Defined Function) called Map. Many mappers can run in parallel on vast amounts of data in a distributed file system

- **SHUFFLE:** The infrastructure then transfers data from the mapper nodes to the "reducer" nodes so that all the (key, value) pairs with the same key go to the same reducer

- **REDUCE:** A UDF that aggregates all the values corresponding to a key. Many reducers can run in parallel.

A Motivating Example: Continuous Map Reduce

- There is a stream of data arriving (eg. tweets) which needs to be mapped to timelines [Pub-Sub]
- Simple solution?
  - Map: (user u, string tweet, time t) → (v1, (tweet, t)) (v2, (tweet, t)) … (vK, (tweet, t))

    where v1, v2, …, vK follow u.
  - Reduce : (user v, (tweet_1, t1), (tweet_2, t2), … (tweet_J, tJ)) → sort tweets in descending order of time

Data Model #2: Active DHT

- DHT (Distributed Hash Table): Stores key-value pairs in main memory on a cluster such that machine H(key) is responsible for storing the pair (key, val)
- Active DHT: In addition to lookups and insertions, the DHT also supports running user-specified code on the (key, val) pair at node H(key)
- Like Continuous Map Reduce, but reducers can talk to each other

Problem #1: Incremental PageRank

- Assume social graph is stored in an Active DHT
- Estimate PageRank using Monte Carlo: Maintain a small number R of random walks from each node; Store these in an Active DHT and update them on each edge arrival
- Suited for Social Networks
- Assume N nodes, M edges
- Amount of work to update PageRank estimates of every node when the M-th edge arrives $= (RN/\varepsilon^2)/M$ which goes to 0 even for moderately dense graphs

[Bahmani, Chowdhury, Goel; 2011]

Data Model #3: Batched + Stream

- Part of the problem is solved using Map-Reduce/some other offline system, and the rest solved in real-time
- Example: The incremental PageRank solution for the Batched + Stream model: Compute PageRank initially using a Batched system, and update in real-time
- Another Example: Social Search [Bahmani, Goel; 2012]

Problem #2: Real-Time Social Search

- Find a piece of content that is exciting to the user's extended network right now and matches the search criteria

- Hard technical problem: imagine building 100M real-time indexes over real-time content

[Bahmani, Goel; 2012]

More Data Models? Problems?

- Data Models: Flash drives? Flash as Cache to Disk? Optimum architecture?

- Problems: An Oracle for Personalized PageRank; Personalized Trends; Audience Estimation

OTHER PROBLEMS? MODELS?