Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Scalable Data Analysis Algorithms

Potsdam, October 18, 2011

Felix Naumann, Arvid Heise

Outline

Goals

- Learn about large-scale data mining problems
- Implement efficient and scalable algorithms
- Evaluation of two Map-Reduce platforms

Seminar Mode

- 6 LP project seminar
- Mostly, consultation instead of group meetings
- 4 teams with 2 students each
- **Only for students who did not participate in a master course about Hadoop**

Grading

- Two short and one long presentations
- Implementation (strategies)
- Final report with evaluation
- Participation in discussions/consultations

Outline

Basic Idea

- Introduced by Google in 2004 [1]

- Usual program is 1 – 10 MB large
- We want to process GBs up to TBs of data
- Inefficient to bring data to program
- Solution: bring the program to the data
- Divide the program in map and reduce parts

Map, Reduce

- Map and reduce are *second-order functions* and origin in functional programming
- Take a set of `data` and a first-order function `func` as parameters

  ```
  map(data, func)
  ```

- Apply `func` to tuples of `data`

- Example: `foreach([1, 2, 3], &println)`

  → `1`
  `2`
  `3`

Key-Value Data Model

- Each tuple in `data` is a key-value pair `(k, v)`
- `k` and `v` may be arbitrary user-defined types
- Within one `data` set, `k` and `v` must be homogeneous
- `k` is comparable/sortable
- `k` and `v` must be serializable

Map

- Every key-value pair in `data` is processed independently:
  - Transfer `func` to all nodes to the cluster
  - Apply `func` to each local pair
  - No need to transfer any data over the network

- Definition: $map([(k_1, v_1), \ldots, (k_n, v_n)], func)$
  $\rightarrow [func(k_1, v_1), \ldots, func(k_n, v_n)]$

- Example:

```
udf(key, value) { return (key, value + "x") }
map([(1, "a"), (2, "b"), (1, "c")], &udf)

→ (1, "ax")
  (2, "bx")
  (1, "cx")
```

Reduce

- Groups all tuples with same key:
  - Globally partition data
  - Transfer `func` to all nodes to the cluster
  - Apply `func` to each partition
  - Might cause heavy network traffic

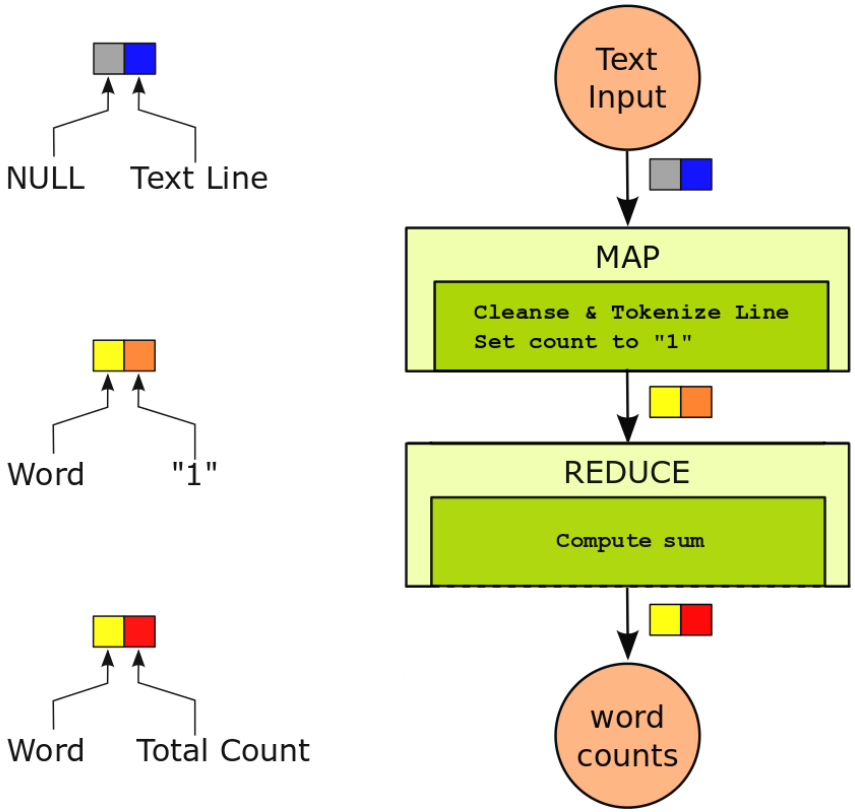
- Definition:
  $reduce([(k_1, v_1), (k_1, v_2), \ldots, (k_n, v_{m-1}), (k_n, v_m)], func)$
  $\rightarrow [func(k_1, [v_1, v_2, \ldots]), \ldots, func(k_n, [\ldots, v_{m-1}, v_m])]$

- Example:

```
udf(key, values) { return (key, concat(values)) }
reduce([(1, "a"), (2, "b"), (1, "c")], &udf)
→ (1, "ac")
  (2, "b")
```

Word Count

- Common example as it can be perfectly ported to Map-Reduce
- Two phases: tokenize sentences, count occurences



Source: http://www.stratosphere.eu/projects/Stratosphere/wiki/WordCountExample

Outline

- Largest open source implementation of Map-Reduce
- Apache project since 2007
- Written in Java
- Used by Yahoo, Facebook and many more

- Very measure, reliable project
- Strong focus on fault-tolerance

Complex Program

- Programs are formulated in jobs
- Job consists always of a map and a reduce
- For complex programs, a driver program queues multiple jobs
- Data flow has to be explicitly specified by file names

Map in Hadoop

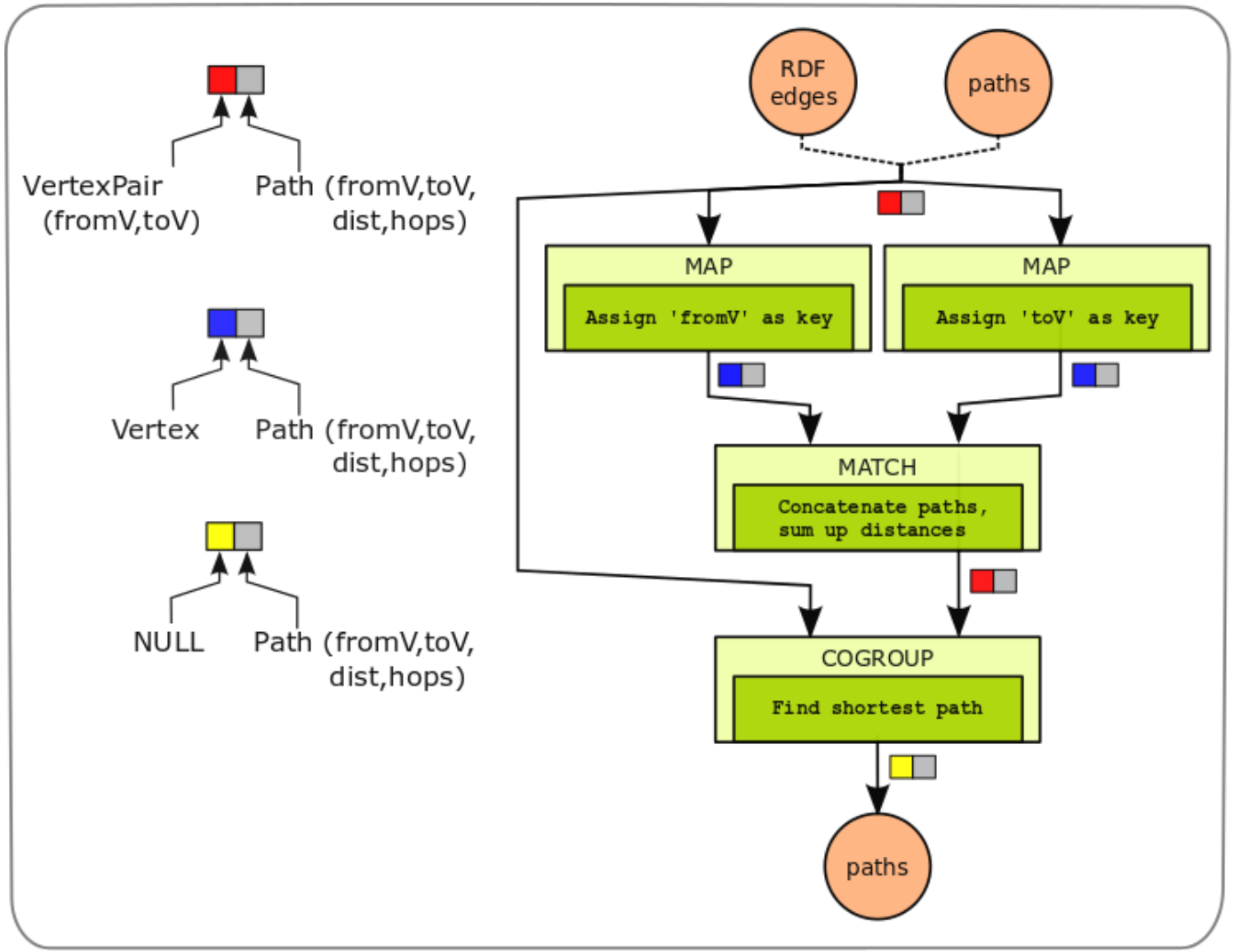

```
public static class Map extends
    Mapper<LongWritable, Text, Text, IntWritable> {
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value, Context contex
      throws IOException, InterruptedException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
      word.set(tokenizer.nextToken());
      context.write(word, one);
    }
  }
}
```

Source: http://wiki.apache.org/hadoop/WordCount

- Research project by HU, TU, and HPI
- Improve some short-comings of Hadoop for complex data flows
- Programs are specified as acyclic directed graphs
- Additional second-order functions with two inputs
- Robust and adaptive query optimization
- Exploit elasticity of clouds (automatically book/unbook VMs)
- (Intelligent fault tolerance)

Stratosphere Plan



Outline

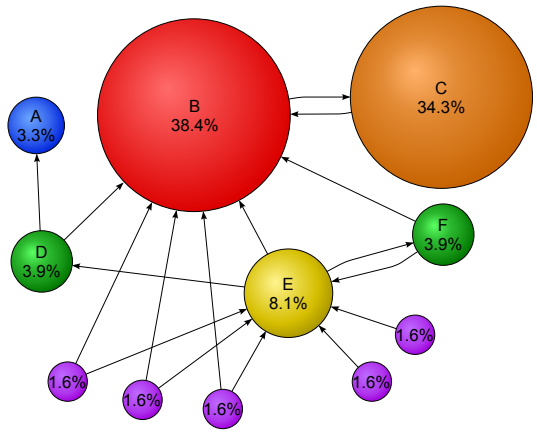
Overview

- Common problems when encountering large data sets
- Apply techniques of machine learning (unsupervised learning algorithm, optimization)
- Map-Reduce offers a standard solution to parallelize algorithms

- Data sets for the tasks are recommendations

Link Analysis

- Google's PageRank: collectively assigns weights to nodes of a graph
- Downrates spam pages in search engines
- Task
  - Implement PageRank
  - Extend to either TrustRank or SpamMass
  - Apply to Wikipedia link graph

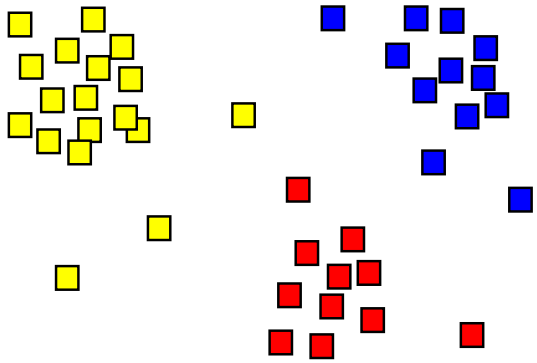


Source: http://en.wikipedia.org/wiki/PageRank

Clustering

- Automatically group similar items
- Traditionally, metric distance measure d
  $d(a, b) + d(b, c) \geq d(a, c)$
- Saves a lot of computation costs
- Here: clustering in non-euclidian space

- Task
  - Implement GRGPF
  - Evaluate different implementation strategies
  - Apply to Wikipedia link graph



Source: http://en.wikipedia.org/wiki/Cluster˙analysis

Frequent Itemsets

- Find commonly co-occurring item sets
- For example, two movies that are often purchased together
- May be used to mine association rules

- Task
  - Implement SON
  - Infer association rules
  - Apply to DBPedia (cleansed, triplified Wikipedia info boxes)

Source: http://www.amazon.de/gp/product/B004MKNBJG

Collaborative Filtering

- Recommend items to users
- Content-based recommendation: find explicit categorization of items and users
- Collaborative filtering: use large amounts of data to deduce data

- Task
  - Implement Distributed Stochastic Gradient Descent
  - Find improvements, especially on Stratosphere
  - Apply to Netflix or KDD Cup

Source: http://www.amazon.de/

Outline

Dates

- Roughly 2011: Hadoop implementation
- 2012: Stratosphere implementation and comparison

- October 18: Topic introduction
- October 22: **Submission of topic wishlist**
- October 24: Notification
- November 15: Task **presentation** and ideas (15+5 min)
- December 20: Intermediate **presentation** (15+5 min)
- January 31, February 7: Final **presentation** (30+10 min)
- End of March: Final **report** (6-8 pages)

Consultation

- Every team gets a 45 min slot per week
- May be canceled (>1 day in advance)
- Mandatory in the first two weeks to discuss the topic
- Mandatory one week before each presentation to discuss slides
- Mandatory in February to discuss paper outline

Infrastructure

- Common infrastructure
  - Mailing list
  - Common repository
  - Trac/Wiki?
  - Cluster with ten nodes (access and time schedule tbd)
- Individual infrastructure
  - Linux (Ubuntu) (VM) recommended for easy installation
  - Hadoop local and pseudo-distributed mode for testing
  - Stratosphere local testing
  - Unit tests for map/reduce tasks

Wish List

- Mail top 3 list to Arvid.Heise@hpi...
- Optionally add team partner
- If top 3 is identically with partner's wish list
  - One mail per team is enough
  - But add teammate in CC, so I can assume agreement
- If more than 8 students apply
  - Randomly select students with first wish for same topic
  - Fill in gaps with second wishes and so on

References

1. Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM 51.
2. http://hadoop.apache.org/
3. http://www.stratosphere.eu/
4. Anand Rajaraman and Jeff Ullman. 2010. Mining of Massive Datasets. http://infolab.stanford.edu/ũllman/mmds.html
5. Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, and Jörg Schad. 2010. Hadoop++: making a yellow elephant run like a cheetah (without it even noticing). In Proceedings of the VLDB Endowment.