Map Reduce and Design Patterns
Lecture 1

Fang Yu

Software Security Lab.
Department of Management Information Systems
College of Commerce, National Chengchi University
http://soslab.nccu.edu.tw

Cloud Computation, March 10, 2015

About Me

Yu, Fang

- 2014-present: Associate Professor, Department of Management Information Systems, National Chengchi University
- 2010-2014: Assistant Professor, Department of Management Information Systems, National Chengchi University
- 2005-2010: Ph.D. and M.S., Department of Computer Science, University of California at Santa Barbara
- 2001-2005: Institute of Information Science, Academia Sinica
- 1994-2000: M.B.A. and B.B.A., Department of Information Management, National Taiwan University

Hadoop and MapReduce Refresh

Hadoop: The Definitive Guide or the Apache Hadoop website.

- Hadoop MapReduce jobs are divided into a set of map tasks and reduce tasks that run in a distributed fashion on a cluster of computers.
- Each task works on the small subset of the data it has been assigned so that the load is spread across the cluster.
- The map tasks generally load, parse, transform, and filter data.
- Each reduce task is responsible for handling a subset of the map task output.

Summarization

Grouping similar data together and then performing an operation such as calculating a statistic, building an index, or just simply counting

- Numerical summarizations
- Inverted index, and
- Counting with counters

Numerical Summarization

Group records together by a key and calculate a numerical aggregate per group

- Consider θ to be a generic numerical summarization function
- Over some list of values $(v_1, v_2, \ldots, v_n)$, find $\lambda = \theta(v_1, v_2, \ldots, v_n)$
- θ could be minimum, maximum, average, median, and standard deviation

Motivation

Consider that your website logs each time a user logs onto the website, enters a query, clicks ads, or performs any other notable action

- When your website is more active?
- How affective your ads are?

Minimum, maximum, and count example

Problem: Given a list of users comments, determine the first and last time a user commented and the total number of comments from that user.

- Key: User ID, Value: MinMaxCountTuple
- Mapper?
- Reducer?

Ideas

- Mapper: For each comment, generate a pair <UserId, (CommentTime, CommentTime, 1)>
- Reducer: For each group by UserID, find min, max, and aggregate count

More details about the Reducer

For each value in a group:

- If the output results minimum is not yet set, or the values minimum is less than results current minimum, we set the results minimum to the input value.
- Same to the maximum, except using a greater than operator
- Each values count is added to a running sum

Remark: the reducer code can be used as a combiner as associativity is preserved.

Average example

Problem: Given a list of users comments, determine the average comment length per hour of day.

- <Hour, CommentLength>
- Mapper?
- Reducer?
- Can the reducer code be used as a combiner?

Average example

To calculate an average, we need two values for each group: the sum of the values that we want to average and the number of values that went into the sum.

- <Hour, (Count, AvgCommentLength)>
- Mapper: For each comment, generate a pair
  <Hour, (1, CommentLength)>
- Reducer: For each group by Hour, accumulate Count and Sum, and compute AvgCommentLength as Sum/Count. Set the pair as
  <Hour, (Count, AvgCommentLength)>
- The reducer code can be used as a combiner

Median and Standard Deviation

Could be more complicated

- Median requires sorting
- Standard deviation requires the average to be discovered prior to reduction

Median and Standard Deviation

Problem: Given a list of users comments, determine the median and standard deviation of comment lengths per hour of day

- A naive idea: <Hour, CommentLength>
- Mapper: For each comment, generate a pair <Hour, CommentLength>
- Reducer: For each group by Hour, sort the comment lengths in a list to find the median value, and accumulate count and sum to calculate mean. Revisit the list to accumulate sum of deviations by squaring the difference between each comment length and the mean and compute standard deviation
- A combiner cannot be used in this implementation. Can we do better?

Memory-conscious median and standard deviation

Instead of having a list whose scaling is O(n) where n = number of comments, the number of key/value pairs in our map is O(max(m)) where m = maximum comment length.

- <Hour, A sorted map of (CommentLength, Count)>
- Mapper: For each comment, generate a pair <Hour, A singleton map with (CommentLength, 1)>
- Reducer: For each group by Hour, maintain the sorted map <Hour, A sorted map of (CommentLength, Count)> Revisit the map to find the median and sum, and accumulate sum of deviations by multiplying the count with the squaring of the difference between each comment length and the mean to compute standard deviation
- A combiner can be used to aggregate the sorted map