CMPE 282 Cloud Services
MapReduce Design Patterns - Summarization

Instructor: Kong Li

Content

- What and Why
- MapReduce refresher
- Summarization Patterns
- Filtering Patterns
- Data Organization Patterns
- Join Patterns

Design Patterns

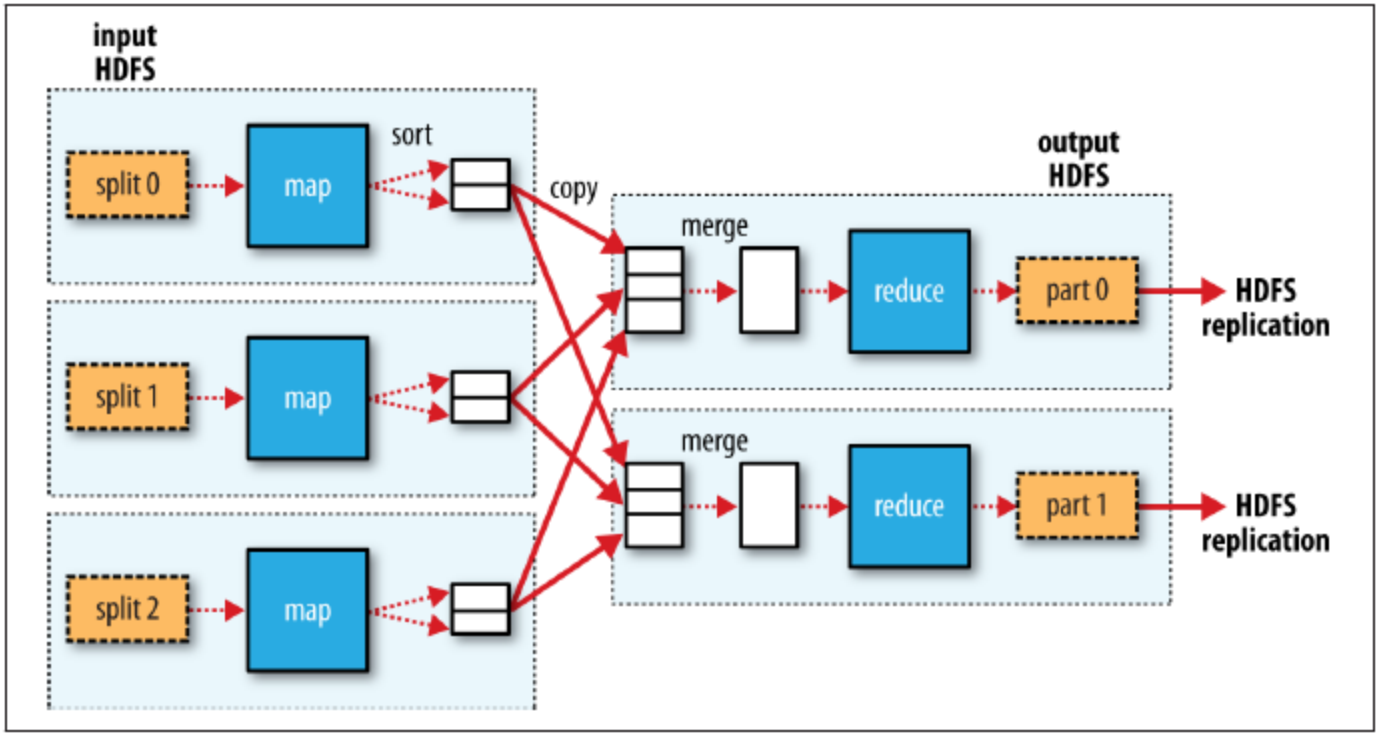
- What
  - Reusable solutions to problems
  - Domain independent
  - Not a cookbook, but a guide
  - Not a finished solution
- Why
  - Makes the intent of code easier to understand
  - Provides a common language for solutions
  - Be able to reuse code
  - Known performance profiles and limitations of solutions

Why MapReduce Design Patterns

- Recurring patterns in data-related problem solving
- Groups are building patterns independently
- Lots of new users every day
- MapReduce is a new way of thinking
- Foundation for higher-level tools (Pig, Hive, …)
- Community is reaching the right level of maturity

MapReduce: Refresher

- Required
  - map: (K1, V1) → list(K2, V2)
  - reduce: (K2, list(V2)) → list(K3, V3)
- Optional
  - Combiner F: (K2, list(V2)) → list(K2, V2)
    - Part of map phase
    - Often the combiner and reduce functions are the same
  - Partition F: (K2, V2) → integer



- HDFS: job input & output
- Driver program
- JobTracker, Task Tracker
- Resource manager, node manager

Designing MapReduce Algorithms

- Key decision: What should be done by `map`, and what by `reduce`?
  - `map` can do something to each individual key-value pair, but it can't look at other key-value pairs
    - Ex: Filtering out key-value pairs we don't need
  - `map` can emit more than one intermediate key-value pair for each incoming key-value pair
    - Ex: Incoming data is text line, `map` produces (word,1) for each word
  - Output value from `map` is a class which can have several properties
    - Ex: `Map` output can be (key, {min, max})
  - `reduce` can aggregate data; it can look at multiple values, as long as `map` has mapped them to the same (intermediate) key
    - Ex: Count the number of words, add up the total cost, ...
- Need to get the intermediate format right
  - If `reduce` needs to look at several values together, `map` must emit them using the same key
- Multiple MapReduce jobs can be chained together

Summarization Patterns

What: top-down summaries

Why: I only want a top-level of my data

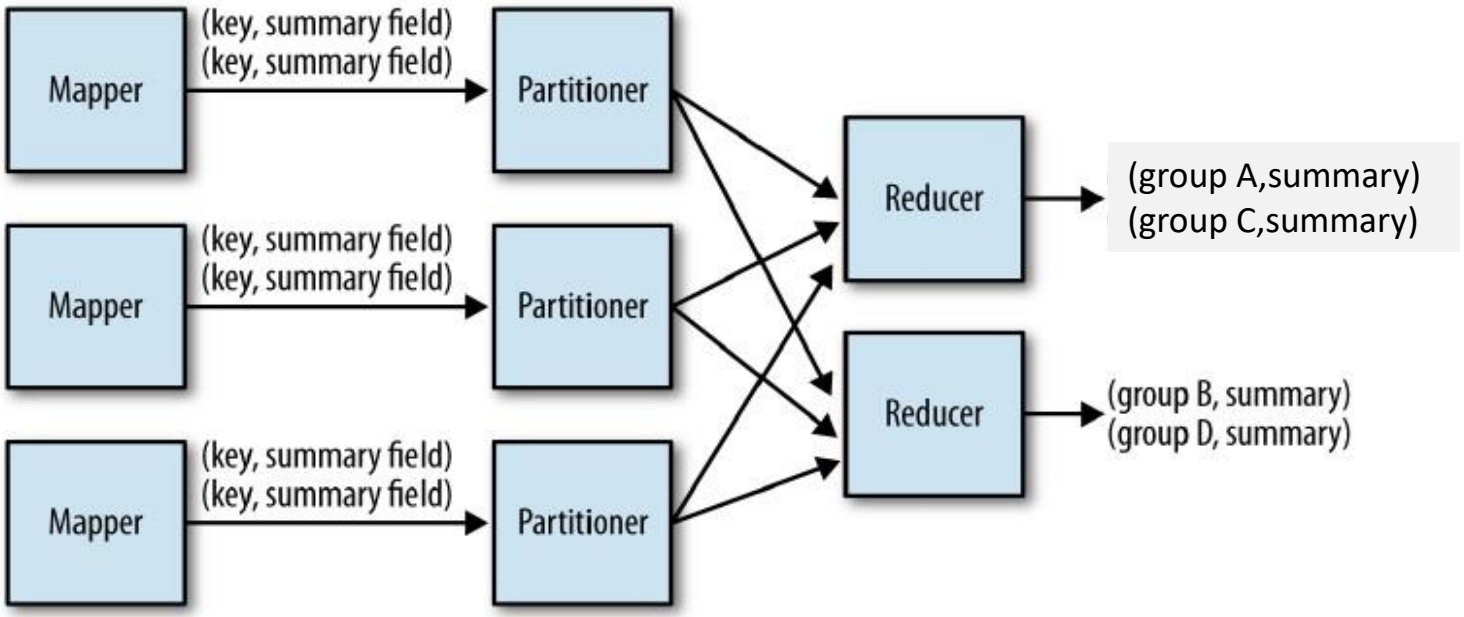
- **Numerical summarizations**
- **Inverted index**
- **Counting with counters**

Numerical Summarizations 1/5

- Intent - Group records together by a key field and calculate a numerical aggregate per group to get a top-level view of the larger data set
- Motivation
  - Many data sets are too large for a human to consume
  - minimum, maximum, average, median, and standard deviation
- Applicability
  - You are dealing with numerical data or counting
  - The data can be grouped by specific fields

Numerical Summarizations 2/5

- Structure
  - Mapper: outputs (key, value)
    - Keys: group-by fields
    - Values: pertinent numerical items (summary of the key)
  - Reducer: receives a set of numerical values (v_1, v_2, v_3, …, v_n) associated w/ a group-by key to perform aggregation function λ. The value of λ is output w/ the given group-by key

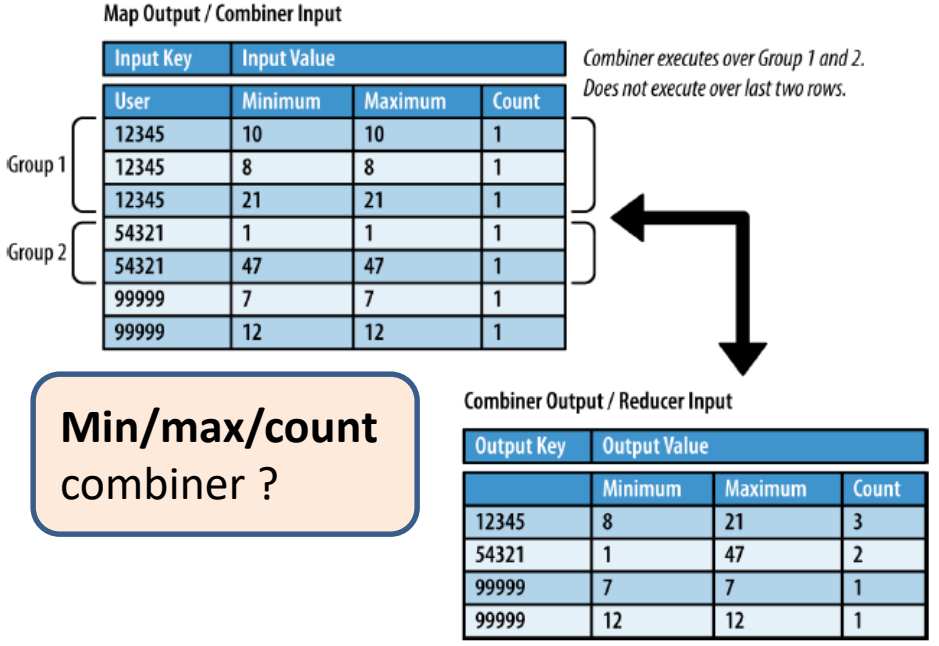


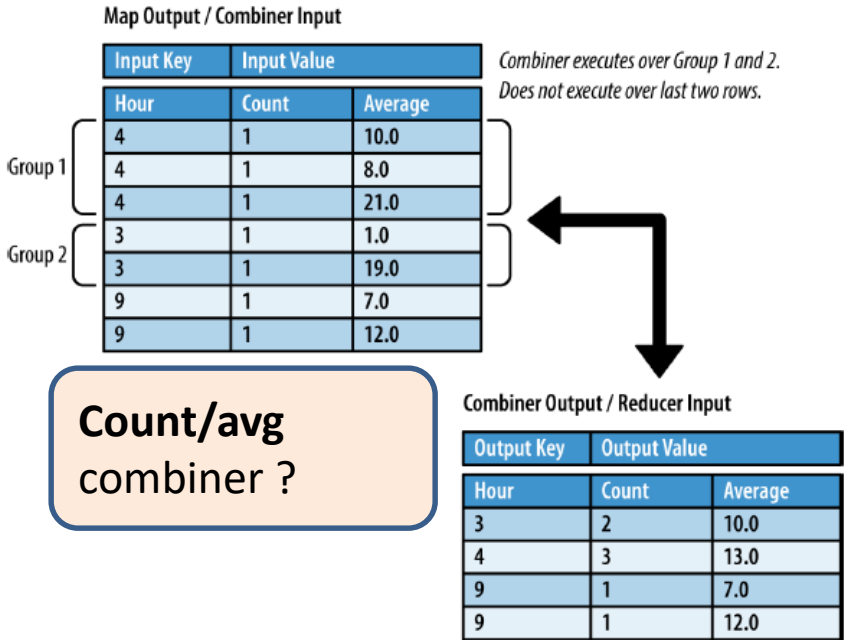
Numerical Summarizations 3/5

- Known uses: Count, Min, Max, Average, Median, etc.
- SQL

```
SELECT groupcol2, MIN(numericalcol1), MAX(numericalcol1), COUNT(*)
FROM table
GROUP BY groupcol2;
```

- Performance: combiner optimization





Numerical Summarizations 4/5

- MinMaxCountDriver.java: Given a list of user's comments, determine the first and last time a user commented and the total # of comments from that user
- In: Comments.xml

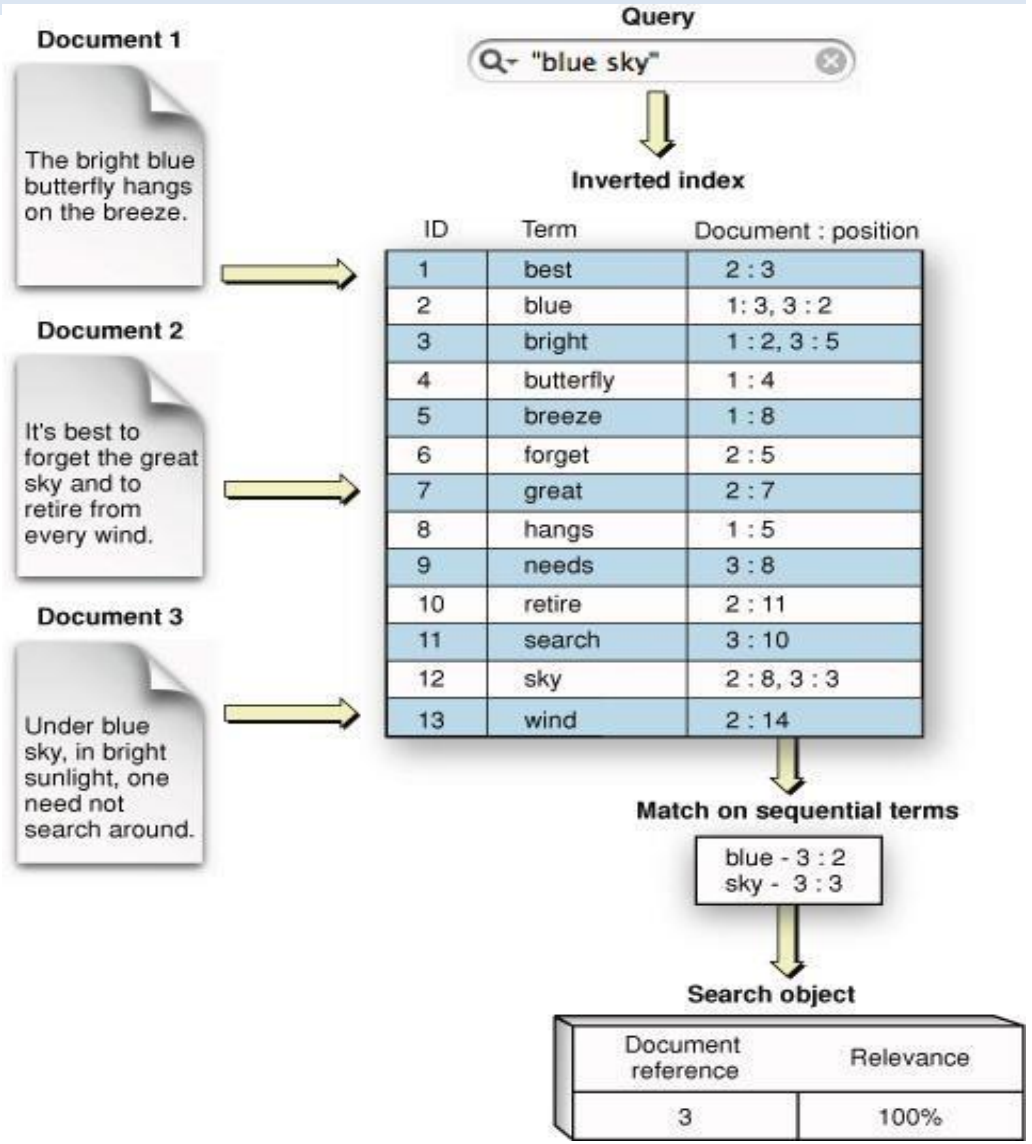
```
public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

    Map<String, String> parsed = transformXmlToMap(value.toString());

    // Grab the "CreationDate" field since it is what we are finding
    // the min and max value of
    String strDate = parsed.get("CreationDate");

    // Grab the "UserID" since it is what we are grouping by
    String userId = parsed.get("UserId");
    // Parse the string into a Date object
    Date creationDate = frmt.parse(strDate);

    // Set the minimum and maximum date values to the creationDate
    outTuple.setMin(creationDate);
    outTuple.setMax(creationDate);

    // Set the comment count to 1
    outTuple.setCount(1);

    // Set our user ID as the output key
    outUserId.set(userId);

    // Write out the hour and the average comment length
    context.write(outUserId, outTuple);
}
```

Numerical Summarizations 5/5

```
public void reduce(Text key, Iterable<MinMaxCountTuple> values,
        Context context) throws IOException, InterruptedException {

    // Initialize our result
    result.setMin(null);
    result.setMax(null);
    result.setCount(0);
    int sum = 0;

    // Iterate through all input values for this key
    for (MinMaxCountTuple val : values) {
        // If the value's min is less than the result's min
        // Set the result's min to value's
        if (result.getMin() == null ||
                val.getMin().compareTo(result.getMin()) < 0) {
            result.setMin(val.getMin());
        }

        // If the value's max is more than the result's max
        // Set the result's max to value's
        if (result.getMax() == null  ||
                val.getMax().compareTo(result.getMax()) > 0) {
            result.setMax(val.getMax());
        }

        // Add to our sum the count for value
        sum += val.getCount();
    }

    // Set our count to the number of input values
    result.setCount(sum);
    context.write(key, result);
}
```

Inverted Index Summarizations 1/6

- Inverted index: mapping from content to its locations

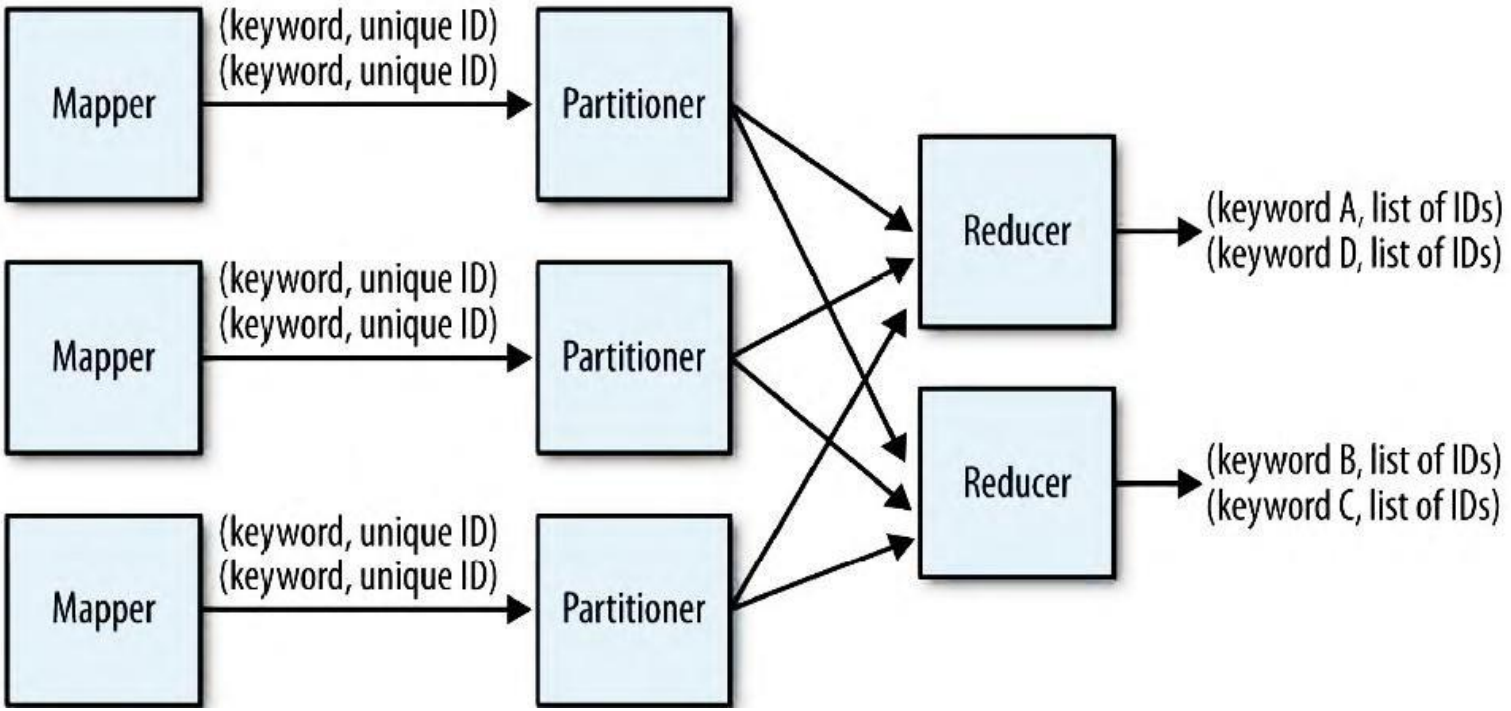


Inverted Index Summarizations 2/6

- Intent - Generate an index from a data set to allow for faster searches
- Motivation
  - To index large data sets on keywords, so that searches can trace terms back to records that contain specific values
  - Search performance of search engine
- Applicability
  - You are requiring quick query responses
  - The results of such a query can be preprocessed and ingested into a database

Inverted Index Summarizations 3/6

- Structure



Inverted Index Summarizations 4/6

- Performance
  - Parsing (computation) cost in Mapper
  - The cardinality of the index keys: increase # of reducers to increase parallelism
  - Common keys, e.g., “the”, causes busy Reducer
    - Ignore common index keys
    - A custom partition function
  - Combiner optimization

Inverted Index Summarizations 5/6

- WikipediaIndex.java: Given a set of user's comments, build an inverted index of Wikipedia URLs to a set of answer post IDs
- In: Posts.xml

map: do link.set only if txt != null

```
public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

    Map<String, String> parsed = MRDPUtils.transformXmlToMap(value
            .toString());

    // Grab the necessary XML attributes
    String txt = parsed.get("Body");
    String posttype = parsed.get("PostTypeId");
    String row_id = parsed.get("Id");
    // if the body is null, or the post is a question (1), skip
    if (txt == null || (posttype != null && posttype.equals("1"))) {
        return;
    }

    // Unescape the HTML because the SO data is escaped.
    txt = StringEscapeUtils.unescapeHtml(txt.toLowerCase());

    link.set(getWikipediaURL(txt));
    outkey.set(row_id);
    context.write(link, outkey);
}
```

Inverted Index Summarizations 6/6

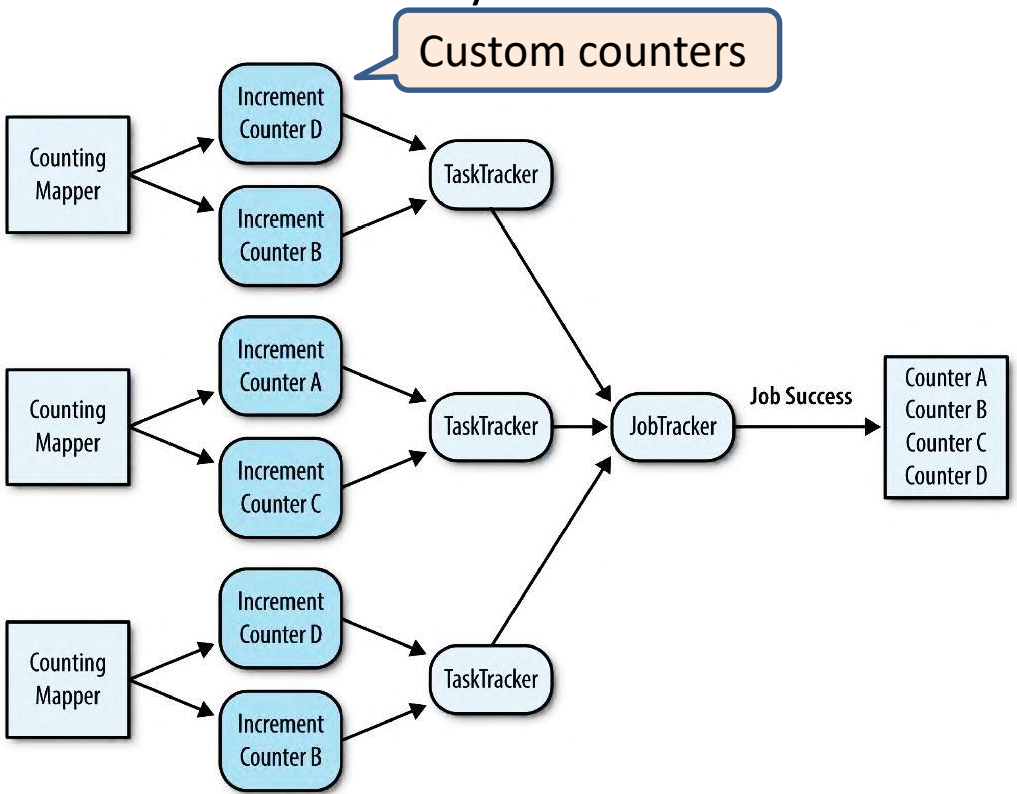
```
public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {

    StringBuilder sb = new StringBuilder();
    boolean first = true;
    for (Text id : values) {
        if (first) {
            first = false;
        } else {
            sb.append(" ");
        }
        sb.append(id.toString());
    }

    result.set(sb.toString());
    context.write(key, result);
}
```

Counting with Counters 1/4

- Intent
  - An efficient means to retrieve count summarizations of large data sets
- Motivation
  - A count or summation can tell you about data as a whole
  - Use the framework's counters
    - Counting is done in the map phase – custom counters
    - TaskTracker and JobTracker aggregate counter values from mapper
    - no combiner F, no partition F, nor reducer required
- Applicability
  - Gather counts or summations over large data sets
  - The number of counters you are going to create is small

Counting with Counters 2/4

- Structure
  - Mapper: processes each input record at a time to increment custom counters based on certain criteria
  - Counter: (a) incremented by one if counting a single instance (b) incremented by some number if executing a summation



Builtin counters

```
:29 INFO mapred.JobClient: Running job: job_200912300823_0013
:30 INFO mapred.JobClient:  map 0% reduce 0%
:40 INFO mapred.JobClient:  map 100% reduce 0%
:46 INFO mapred.JobClient:  map 100% reduce 100%
:48 INFO mapred.JobClient: Job complete: job_200912300823_0013
:48 INFO mapred.JobClient: Counters: 18
:48 INFO mapred.JobClient:   Job Counters
:48 INFO mapred.JobClient:     Launched reduce tasks=1
:48 INFO mapred.JobClient:     Launched map tasks=2
:48 INFO mapred.JobClient:     Data-local map tasks=2
:48 INFO mapred.JobClient:   FileSystemCounters
:48 INFO mapred.JobClient:     FILE_BYTES_READ=2521661
:48 INFO mapred.JobClient:     HDFS_BYTES_READ=1259430
:48 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=5043392
:48 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=366678
:48 INFO mapred.JobClient:   Map-Reduce Framework
:48 INFO mapred.JobClient:     Reduce input groups=33783
:48 INFO mapred.JobClient:     Combine output records=0
:48 INFO mapred.JobClient:     Map input records=22109
:48 INFO mapred.JobClient:     Reduce shuffle bytes=2521667
:48 INFO mapred.JobClient:     Reduce output records=33783
:48 INFO mapred.JobClient:     Spilled Records=430274
:48 INFO mapred.JobClient:     Map output bytes=2091381
:48 INFO mapred.JobClient:     Map input bytes=1257289
:48 INFO mapred.JobClient:     Combine input records=0
:48 INFO mapred.JobClient:     Map output records=215137
:48 INFO mapred.JobClient:     Reduce input records=215137
```

Counting with Counters 3/4

- Known uses
  - Count number of records (over a given time period)
  - Count a small number of unique instances
  - Counters can be used to sum fields of data together
- Performance
  - Using counters is very fast
    - data is read in through the mapper and no output is written
  - Performance depends on # of map tasks and how much time it takes to process each record

Counting with Counters 4/4

- CountNumUsersByStateDriver.java: Count # of users from each state using Hadoop custom counters
- In: Users.xml
- Driver:

job.setNumReduceTasks(0);

```
int code = job.waitForCompletion(true) ? 0 : 1;

if (code == 0) {
    for (Counter counter : job.getCounters().getGroup(
            CountNumUsersByStateMapper.STATE_COUNTER_GROUP)) {
        System.out.println(counter.getDisplayName() + "\t"
                + counter.getValue());
    }
}

// Clean up empty output directory
FileSystem.get(conf).delete(outputDir, true);

System.exit(code);
```

```
public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

    Map<String, String> parsed = MRDPUtils.transformXmlToMap(value
            .toString());

    // Get the value for the Location attribute
    String location = parsed.get("Location");

    // Look for a state abbreviation code if the
    // location is not null or empty
    if (location != null && !location.isEmpty()) {

        // Make location uppercase and split on white space
        String[] tokens = location.toUpperCase().split("\\s");

        // For each token
        boolean unknown = true;
        for (String state : tokens) {

            // Check if it is a state
            if (states.contains(state)) {
                // If so, increment the state's counter by 1
                // and flag it as not unknown
                context.getCounter(STATE_COUNTER_GROUP, state)
                        .increment(1);
                unknown = false;
                break;
            }
        }

        // If the state is unknown, increment the UNKNOWN_COUNTER counte
        if (unknown) {
            context.getCounter(STATE_COUNTER_GROUP, UNKNOWN_COUNTER)
                    .increment(1);
        }
    } else {
        // If it is empty or null, increment the
        // NULL_OR_EMPTY_COUNTER counter by 1
        context.getCounter(STATE_COUNTER_GROUP,
                NULL_OR_EMPTY_COUNTER).increment(1);
    }
}
```

References

- Donald Miner and Adam Shook, *MapReduce Design Patterns*.
  - http://oreil.ly/mapreduce-design-patterns
  - https://github.com/adamjshook/mapreducepatterns