Map Reduce

Matthew Arnold & Benedikt Schesch

Google MapReduce Goals and Achievements

Goals

- Express real world problems using simple model
- Process large datasets without specialized per-project software
- Hide systems engineering behind abstraction, conceptually straightforward

Achievements

- Widely used by Google (0-900 instances within a year)
- “Good enough” performance compared to custom solutions
- Typically much less code, easier to reason about

Map and Reduce Primitives

- Map - takes input pair and produces intermediate kv pairs
- Reduce - accepts intermediate kv pairs and performs some operation, emitting final list of values

Example: count number of occurrences of each unique word

- Map input **(file name, file contents)** kvp and outputs list of **(word, count)** kvp
- Intermediate groups together all intermediate kvp of the same key (i.e. all words’ counts)
- Reduce input **(word, count list)** kvp and outputs **list of counts**

Hadoop vs Google MapReduce Word Counter

```java
public class WC_Mapper extends MapReduceBase
 implements Mapper<LongWritable, Text, Text, IntWritable> {
 private static final IntWritable one = new IntWritable(1);
 private Text word = new Text();
 public void map(LongWritable key, Text value,
   OutputCollector<Text, IntWritable> output, Reporter
reporter
 ) throws IOException {
   String text = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(text);


   while (tokenizer.hasMoreTokens()) {
     word.set(tokenizer.nextToken());
     output.collect(word, one);
   }
 }
}
```

```cpp
class WordCounter : public Mapper {
public:


 virtual void Map(const MapInput &input) {


   const string &text = input.value();
   const int n = text.size();


for (int i = 0; i < n;) {
     while ((i < n) && isspace(text[i])) i++;   // Find word start
     int start = i;
     while ((i < n) && !isspace(text[i])) i++;   // Find word end
     if (start < i)
       Emit(text.substr(start, i - start), "1");
   }
 }
};
```

My takeaway - frontend APIs functionally equivalent

Hadoop vs Google MapReduce Word Counter

```java
public class WC_Reducer
 extends MapReduceBase
 implements Reducer<Text, IntWritable, Text, IntWritable> {

 public void reduce(
    Text key,
    Iterator<IntWritable> values,
    OutputCollector <Text, IntWritable> output,
    Reporter reporter
 ) throws IOException {

    int value = 0;
    while (values.hasNext()) {
      value += values.next().get();
    }


    output.collect(key, new IntWritable(value));
  }
}
```

```cpp
class Adder : public Reducer {

 virtual void Reduce(ReduceInput *input) {

    // Iterate over all entries with the
    // same key and add the values
    int64 value = 0;
    while (!input->done()) {
      value += StringToInt(input->value());
      input->NextValue();
    }
    // Emit sum for input->key()
    Emit(IntToString(value));
  }
};
```

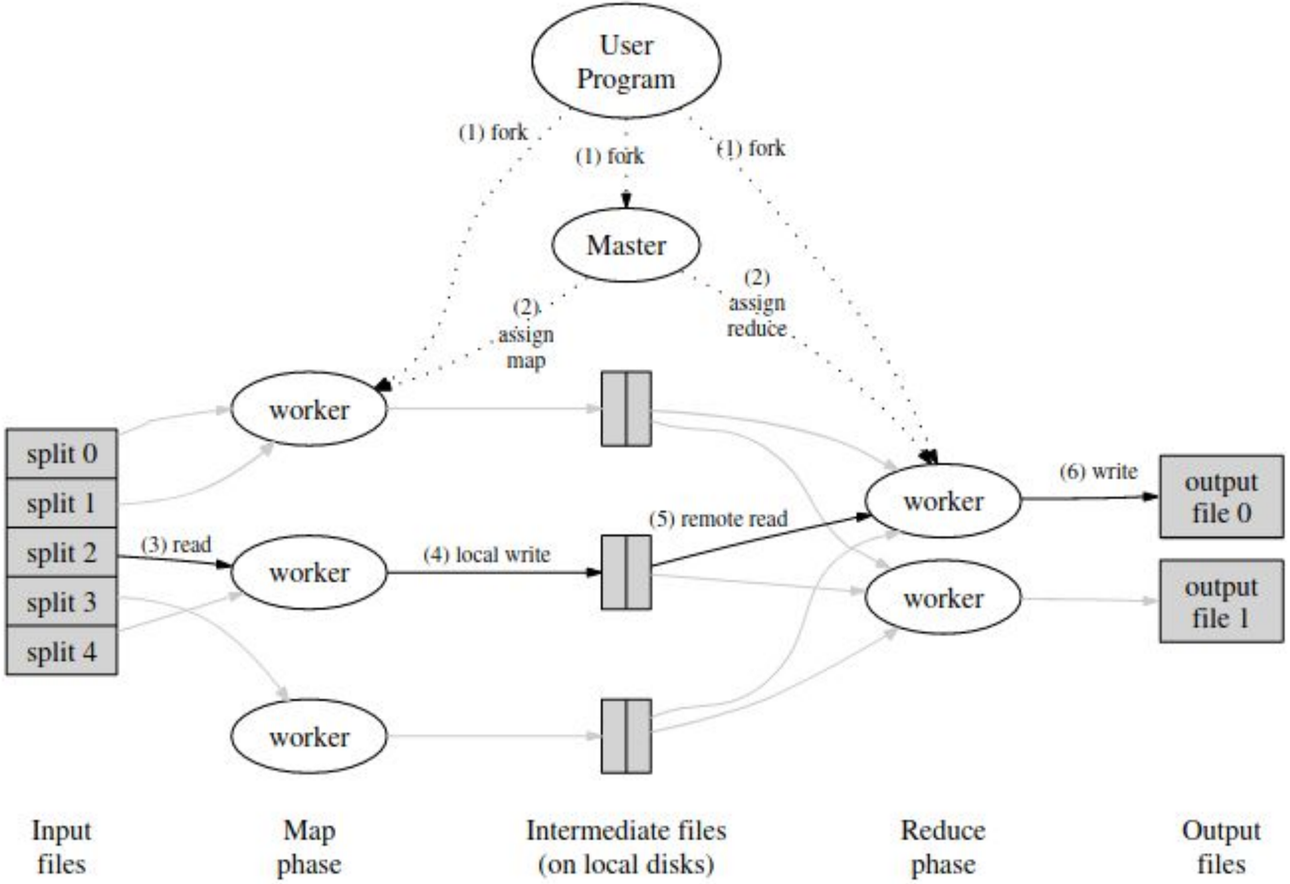


Figure 1: Execution overview

Characterizing Throughput

Why do throughput graphs look the way they look?

Why is input rate higher than shuffle/output rate?

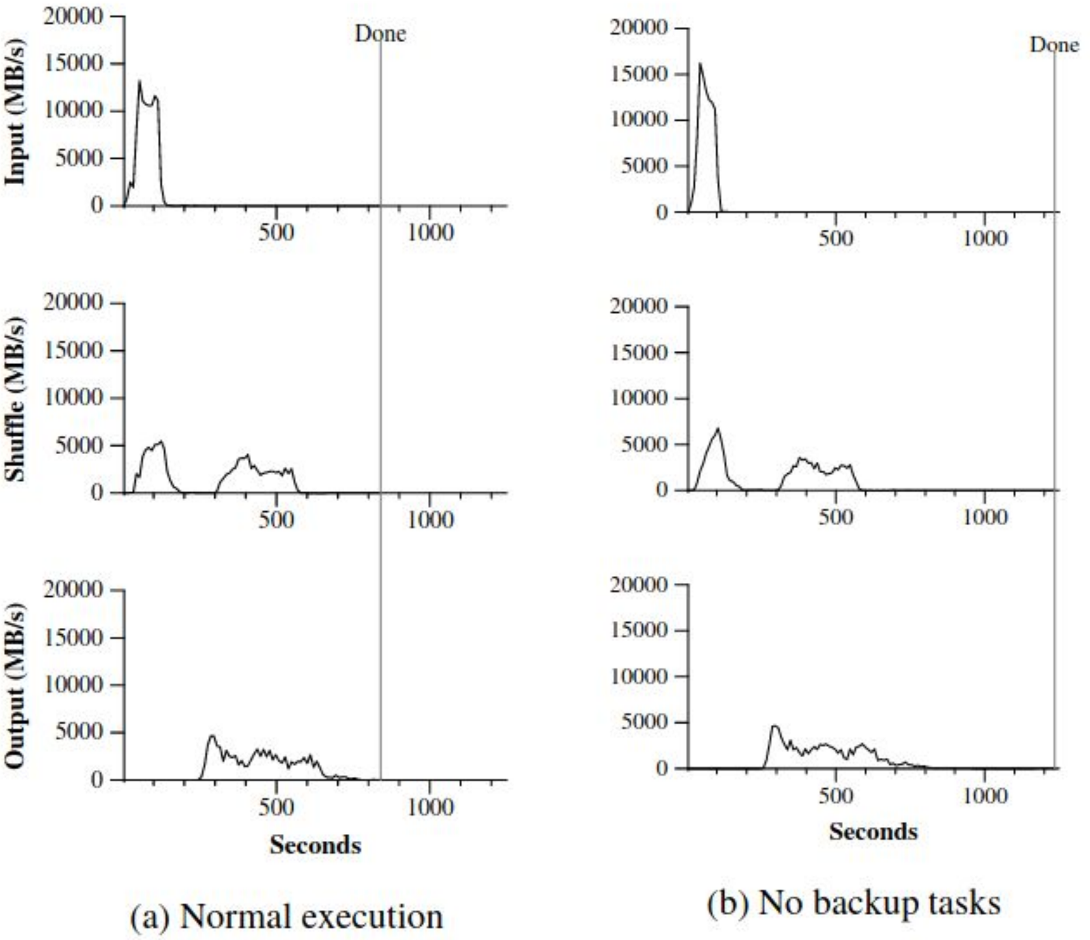
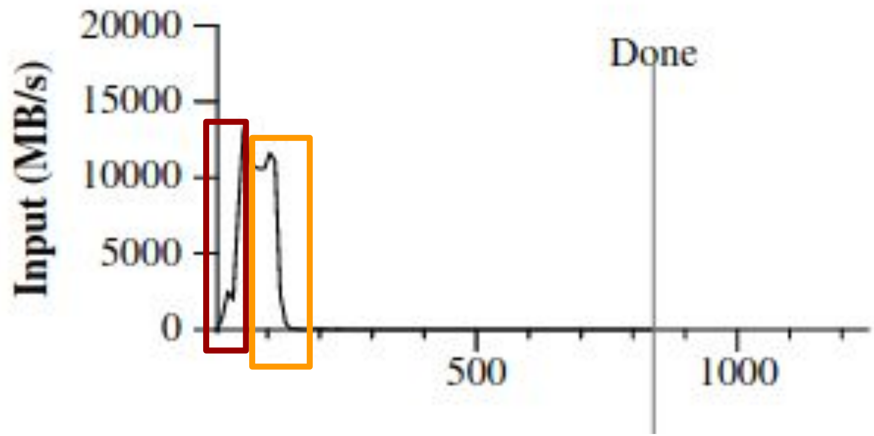


Figure 3: Data transfer rates over time for different executic

Characterizing Throughput

Why does input throughput graph look the way it does?

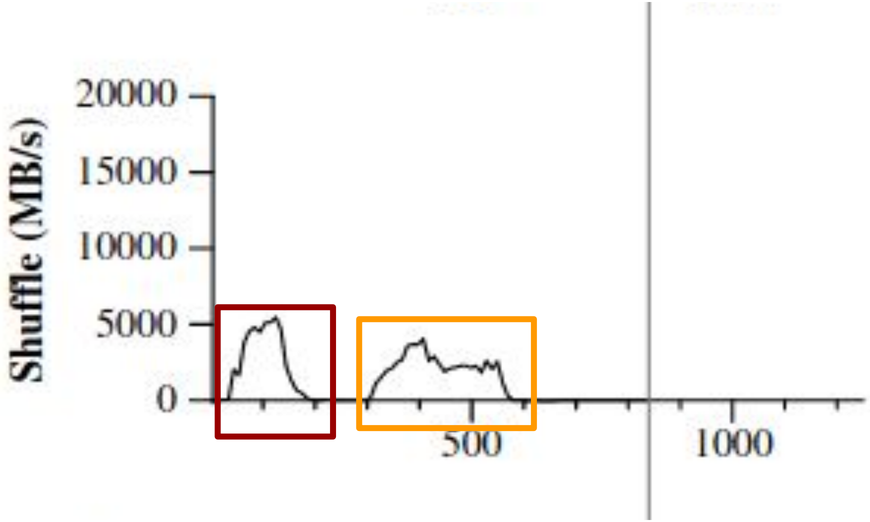
- Forking process across cluster (slow startup)
- Execution of map worker and cooldown as backups/slower processes finish



Characterizing Throughput

Why does shuffle throughput graph look the way it does?

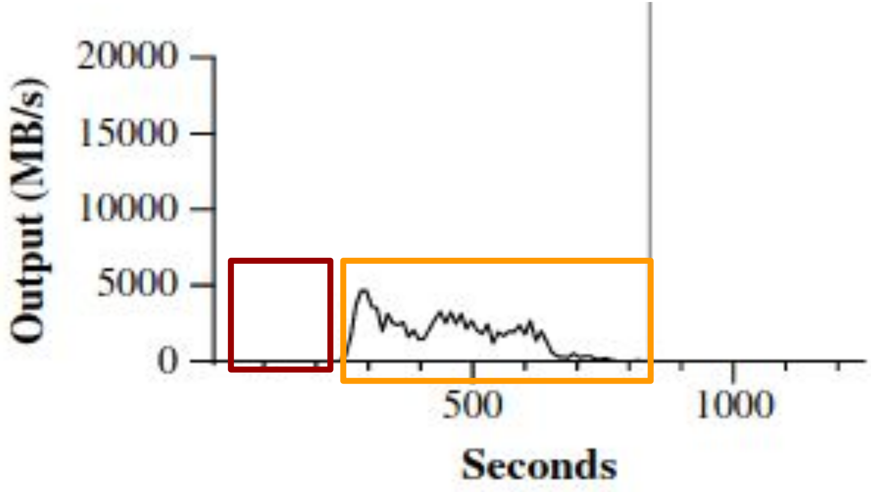
- First wave of map workers finishing
- Second wave of map workers finishing, backups working



Characterizing Throughput

Why does shuffle throughput graph look the way it does?

- Waiting for mapping and intermediate shuffling
- Processing (throughput < shuffling because output replicated



Discussion

1. Are there any optimizations you can make to reduce resources (energy, memory, compute, communication etc) used by MapReduce. Does your proposal introduce another complexity?

Discussion

1. Are there any optimizations you can make to reduce resources (energy, memory, compute, communication etc) used by MapReduce. Does your proposal introduce another complexity?

- Optimizing backup tasks
  - Earlier select tasks that fail or straggle to reduce tail
  - Aren't randomly rescheduling remaining tasks, targeting tasks that are lagging behind
  - How do we characterize a worker as a straggler?
- Replication of master to avoid restart when master fails
- Reducing latency between map and reduce
  - Optimizing spatial locality s.t. map and reduce workers that access intermediate data close to each other
  - Combiner function ran by map worker attempts to increase throughput by reducing intermediate repetition

Discussion

2. Describe an unlikely yet interesting use case of the MapReduce system.

Discussion

2. Describe an unlikely yet interesting use case of the MapReduce system.

Anywhere where big data exists

- Social network platforms - count number of memes in social networking platform, etc.
- MapReduce for real-time data
  - Paper written when you have all data in db, but MapReduce has some desirable characteristics for analyzing real-time data (think IOT sensor)
  - Reducing scheme fits real-time data well
    - Reduce becomes reduce(prevResults, reduce(newData…))
    - Continuous MapReduce: https://github.com/estuary/flow

Discussion

3. In DeWitt and Stonebreaker’s response http://craig-henderson.blogspot.com/2009/11/dewitt-and-stonebrakers-mapreduce-major.html, they say: “Given the experimental evaluations to date, we have serious doubts about how well MapReduce applications can scale.” This seems, at its face, ridiculous. Discuss what they might sensibly mean here.

Discussion

3. In DeWitt and Stonebreaker’s response http://craig-henderson.blogspot.com/2009/11/dewitt-and-stonebrakers-mapreduce-major.html, they say: “Given the experimental evaluations to date, we have serious doubts about how well MapReduce applications can scale.” This seems, at its face, ridiculous. Discuss what they might sensibly mean here.

Discussion

4. MR is an exemplar of the different design methodologies of the systems and DB communities. Who is right?

Discussion

4. MR is an exemplar of the different design methodologies of the systems and DB communities. Who is right?

- MP vs Parallel DBMS
    - Parallel DBMS - DM system running over multiple nodes, supporting SQL queries
    - MP's purpose is to process data
    - DBMS has multiple purposes, one of which includes processing, but also storage and management
    - DBMS better at simpler queries, MP more expressive

Discussion

- "A Comparison of MapReduce and Parallel Database Management Systems"
  - Competing paradigms
    - Large data volumes
    - Analytics - Parallel DBMS optimized for simple queries, for complex algorithms MR can be more efficient
  - Complementary paradigms
    - MR doesn't suffer from Parallel DBMS issue of load time, but once loaded Parallel good for repeated queries
    - Analytics again - both serve different purposes

Criticism

- DeWitt and Stonebraker's "MapReduce: A major step backwards" criticise the MapReduce approach

1. MapReduce is a step backwards in database access

2. MapReduce is a poor implementation

3. MapReduce is missing features

4. MapReduce is not novel

5. MapReduce is incompatible with the DBMS tools

Criticism

1. MapReduce is a step backwards in database access

   Schemas are good since they allow to separate the structure of data with the algorithms that run on it. Two approaches to DBMS access programing:

   - **By stating what you want - rather than presenting an algorithm for how to get it (relational view)**
   - **By presenting an algorithm for data access (Codasyl view)**

   Makes it difficult to understand a program from an exterior perspective

Criticism

2. MapReduce is a poor implementation

No indexing only allows for brute force computations, imagine you have a query that only looks at a very small subset of the data.

Assume in the map phase when there is wide variance in the distribution of records with the same key. Some reduce instances will take much longer than others.

Push vs Pull (Each Reduce will ask each Map its file)

Criticism

3. MapReduce is missing features

No way to update data

No Transactions, parallel updates and failure recovery

No Constraints/Integrity checks to filter out bad data

No Indexing

Criticism

4. MapReduce is not novel

Similar approaches have already been created a long time ago.
“xapping/reduction” was found in 1985 Danny Hillis’s Thesis.

5. MapReduce is incompatible with the DBMS tools

All tools build on top of SQL are no longer usable.
E.g. Oracle Data Mining to discover structure in large datasets

Criticism

Paper: A Comparison of Approaches to Large-Scale Data Analysis

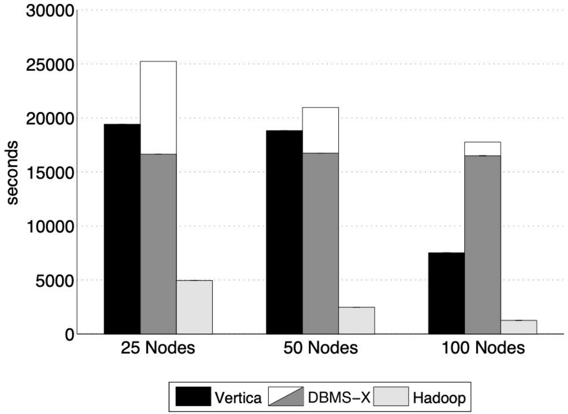


Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

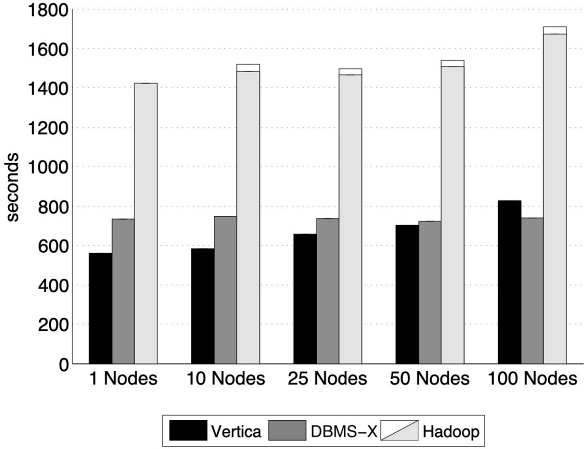


Figure 7: Aggregation Task Results (2.5 million Groups)

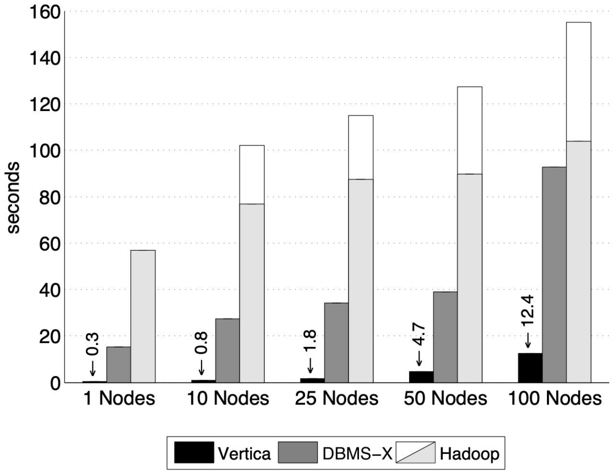


Figure 6: Selection Task Results

Hive

- MapReduce is surprisingly expressive
- One can express certain SQL queries with MapReduce operations
- Writing and maintaining Map/Reduce operations is difficult

```
FROM (SELECT a.status, b.school, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.userid = b.userid and
               a.ds='2009-03-20' )
      ) subq1
INSERT OVERWRITE TABLE gender_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.gender, COUNT(1) GROUP BY subq1.gender
INSERT OVERWRITE TABLE school_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```

SQL and Query plan to generate daily counts of status updates by school and gender (3 map-reduce jobs for multi-table insert query)