CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2016
Doc 22 Hadoop Examples
Nov 10, 2016

WordMean.java

Shows aggregation of results after map & reduce are done

Issues

- Assumes only one reducer is used
  - How to insure that on a cluster? On AWS EMR?

- Output goes to standard out
  - Where can you find it
  - Where can you find in on AWS EMR

WordMedian.java

Use of counters in calculation

WordStandardDeviation.java

Multiple types of key-value pairs produced in map

MultiFileWordCount.java

Shows how to write custom RecordReader

Uses builtin reducer

org.apache.hadoop.mapreduce.lib.reduce

IntSumReducer

LongSumReducer

org.apache.hadoop.mapreduce.lib.map

InverseMapper
MultithreadedMapper
RegexMapper
TokenCounterMapper
WrappedMapper

org.apache.hadoop.mapreduce.lib.chain

ChainMapper

ChainReducer

AggregateWordCount.java

There are common types of operations - Hadoop has some built in

- DoubleValueSum
- LongValueMax
- LongValueMin
- LongValueSum
- StringValueMax
- StringValueMin
- UniqValueCount

Example shows how to extend them

```
public ArrayList<Entry<Text, Text>> generateKeyValPairs(Object key,
                                                        Object val) {
  String countType = LONG_VALUE_SUM;
  ArrayList<Entry<Text, Text>> retv = new ArrayList<Entry<Text, Text>>();
  String line = val.toString();
  StringTokenizer itr = new StringTokenizer(line);
  while (itr.hasMoreTokens()) {
    Entry<Text, Text> e = generateEntry(countType, itr.nextToken(), ONE);
    if (e != null) {
      retv.add(e);
    }
  }
  return retv;
 }
}
```

Mapper

cleanup(org.apache.hadoop.mapreduce.Mapper.Context context)

Called once at the end of the task

map(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Mapper.Context context)

run(org.apache.hadoop.mapreduce.Mapper.Context context)

setup(org.apache.hadoop.mapreduce.Mapper.Context context)

Called once at the beginning of the task.

What can we do with Context?

Where is the API docs?

public KEYIN getCurrentKey() throws IOException, InterruptedException
public VALUEIN getCurrentValue() throws IOException, InterruptedException
public boolean nextKeyValue() throws IOException, InterruptedException
public Counter getCounter(Enum counterName)

public void write(KEYOUT key, VALUEOUT value) throws

public String getStatus() {
public void setStatus(String msg)
public Configuration getConfiguration()

public int getNumReduceTasks()

public RawComparator<?> getSortComparator()
public Path getWorkingDirectory() throws IOException
public void progress()

public Iterable<VALUEIN> getValues() throws
public boolean nextKey() throws IOException, InterruptedException
public float getProgress()

Where is LineReader?

import org.apache.hadoop.util.LineReader;

Issue - Lots of IO

map writes lot of key value pairs

The word “**and”** appears many times

Why write out “and 1” multiple times

Aggregate the values in map

Aggregation in Pseudo code

```
public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

   private Hashmap<String,Integer> wordCount = new Hashmap();

   public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {
     StringTokenizer itr = new StringTokenizer(value.toString());
     while (itr.hasMoreTokens()) {
          word = itr.nextToken()
          wordCount[word] += wordCount[word]
     }
   }

public void cleanUp(Context context) {
   write out the contents wordCount
}
```

Real Pseudo Code

```
class Mapper
    method Initialize
        H ← new AssociativeArray

    method Map(docid a, doc d)
        for all term t ∈ doc d do H{t} ← H{t} + 1      ◁ Tally counts across documents

    method Close
        for all term t ∈ H do
        Emit(term t,count H{t})
```

Partial Aggregation

Aggregation trades memory for disk IO

What if we run out of memory?

Every n’th word write out the results

Reducer Class

cleanup(org.apache.hadoop.mapreduce.Reducer.Context context)

Called once at the end of the task.

reduce(KEYIN key, Iterable<VALUEIN> values,
org.apache.hadoop.mapreduce.Reducer.Context context)

This method is called once for each key.

run(org.apache.hadoop.mapreduce.Reducer.Context context)

setup(org.apache.hadoop.mapreduce.Reducer.Context context)

Called once at the start of the task.

MapReduceBase

org.apache.hadoop.mapred.MapReduceBase

Base class for implementing Map or Reduce

Contains two additional methods

close()

Called when map or reduce is done

configure(JobConf)

Called before map

Aggregation in Reducer

```
public static class MeanReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

  private int sum = 0;
  private int count = 0;
  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {

    for (IntWritable val : values) {
      count++
      sum += val.get();
    }
   }

  public void cleanUp(Context context) {
    write out sum/count
  }
}
```

Computing Mean

If keep track of sum and number of values we can compute mean

Rather than map output (key, number)

Output (key, (sum, count))

count = 1

sum = number

```
class Mapper
    method Map(string t, integer r)
        Emit(string t, pair (r, 1))

class Combiner
    method Combine(string t, pairs [(s1, c1), (s2, c2) . . .])
        sum ← 0
        cnt ← 0
        for all pair (s, c) ∈ pairs [(s1, c1), (s2, c2) . . .] do
            sum ← sum + s
            cnt ← cnt + c
    Emit(string t, pair (sum, cnt))
```

Pairs & Stripes - Motivation

Count the number of times any word pair occurs

Map output key becomes (w1, w2)

Increases the number of keys needed

Pairs

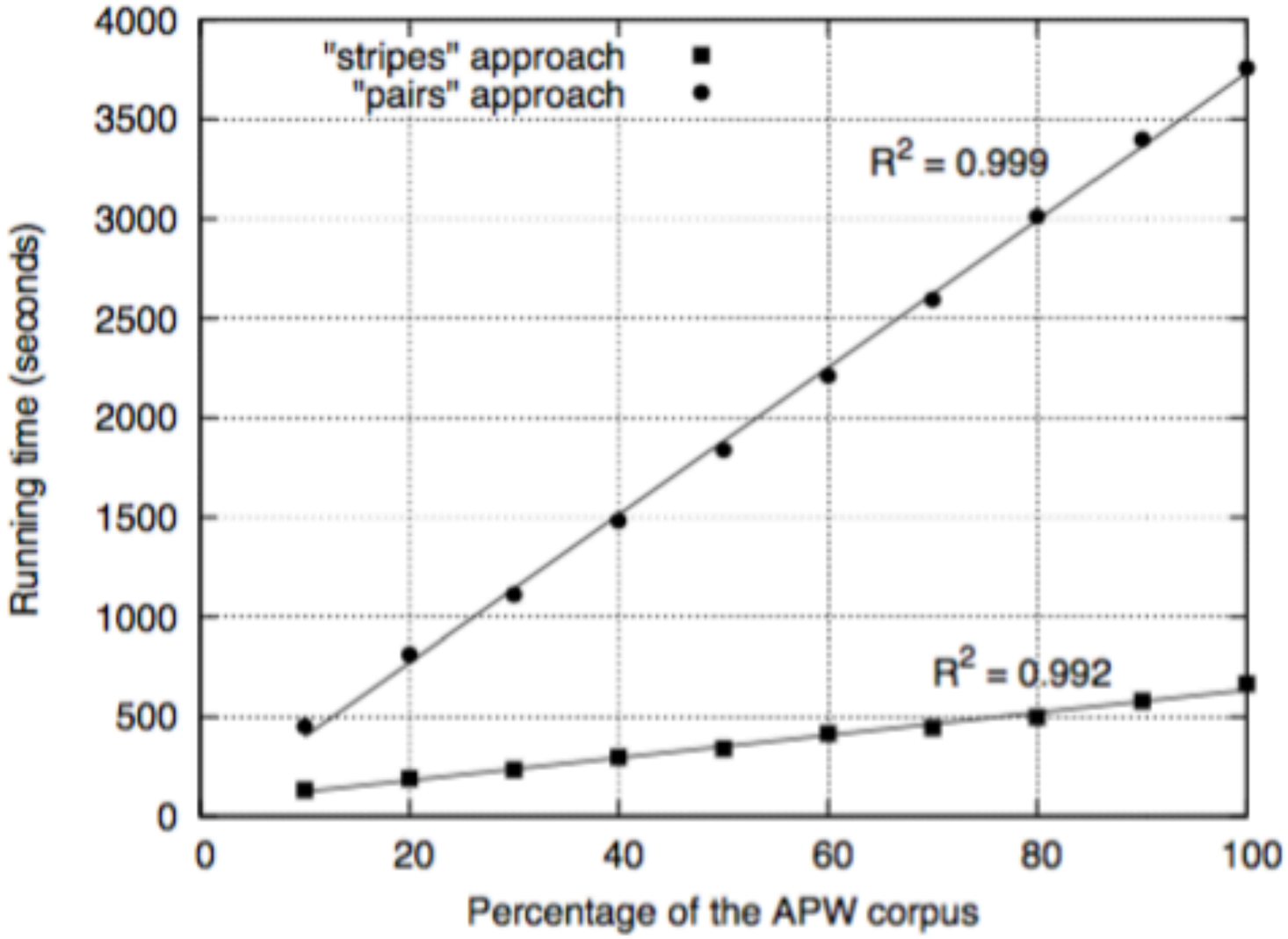
```
class Mapper
    method Map(docid a, doc d)
        for all term w ∈ doc d do
            for all term u ∈ Neighbors(w) do
                Emit(pair (w, u), count 1) ◁ Emit count for each co-occurrence

class Reducer
    method Reduce(pair p, counts [c1, c2, . . .])
        s←0
        for all count c ∈ counts [c1,c2,...] do
            s ← s + c ◁ Sum co-occurrence counts
        Emit(pair p, count s)
```

Stripes

```
class Mapper
    method Map(docid a, doc d)
        for all term w ∈ doc d do
            H ← new AssociativeArray
            for all term u ∈ Neighbors(w) do
                H{u} ← H{u} + 1 ◁ Tally words co-occurring with w
            Emit(Term w, Stripe H)
```

```
class Reducer
    method Reduce(term w, stripes [H1, H2, H3, . . .])
    Hf ← new AssociativeArray
    for all stripe H ∈ stripes [H1, H2, H3, . . .] do
        Sum(Hf,H) ◁ Element-wise sum
    Emit(term w, stripe Hf )
```



2.27 million documents

5.7 GB