Information Systems Group

Hasso Plattner Institut | Universität Potsdam

MapReduce on Hadoop

Übung – DBS2

Felix Naumann

Alexander Albrecht

Motivation

- Problem: Verarbeiten großer Datenmengen (z.B. Wikipedia Corpus)
  - Häufigkeiten zählen, Invertierten Index berechnen, ...
- Alternative Szenarien
  - Log-File Analyse
  - Google Page Rank
  - Wetterdaten …

Verteiltes Rechnen

- **Bekannte Beispiele**
  - **S**earch for **E**xtra**t**errestrial **I**ntelligence
  - IBM Deep Blue
  - Google (PageRank)
  - Rendering
  - Wettervorhersage
  - ...

Motivation

Beispiel: Worthäufigkeit

```
01: String[] input={"this is the foo text",
                    "this is the bar text","even more foo text"};
02: HashMap<String, Integer> wordcount = new HashMap<String,
                                                    Integer>();
03: for (int i = 0; i < input.length; i++) {
04:  String[] words = input[i].split(" ");
05:  for (int j = 0; j < words.length; j++) {
06:   if (wordcount.containsKey(words[j])) {
07:    wordcount.put(words[j], wordcount.get(words[j]) + 1);
08:   } else {
09:    wordcount.put(words[j], 1);
10:   }
11:  }
12: }
13: for (String s : wordcount.keySet())
14:  System.out.println(s + ":" + wordcount.get(s));
```

Motivation

Beispiel: Worthäufigkeit

```
01: String[] input={"this is the foo text",
                    "this is the bar text","even more foo text"};
02: HashMap<String, Integer> wordcount = new HashMap<String,

03: for (int i = 0; i < input.length; i++) {
04:  String[] words = input[i].split(" ");
05:  for (int j = 0; j < words.length; j++) {
06:   if (wordcount.containsKey(words[j])) {
07:    wordcount.put(words[j], wordcount.get(wor
08:   } else {
09:    wordcount.put(words[j], 1);
10:   }
11:  }
12: }
13: for (String s : wordcount.keySet())
14:  System.out.println(s + ":" + wordcount.get(s));
```

text:3
is:2
more:1
even:1
foo:2
the:2
bar:1
this:2

Motivation

- Verteiltes Rechnen
  - Programmiermodell erforderlich
  - Infrastruktur erforderlich (Load balancing, data distribution, messaging, hardware failures)

- Google Strategie: Berechnung auf mehrere Rechner verteilen
  - Billige Standard-Hardeware
  - **Shared Nothing**
  - Geographisch verteilt
  - Repliziert

Motivation

- Beispiel: Worthäufigkeit

- Phase 1
  - **Input:**
    ```
    (1, "this is the foo text")
    (2, "this is the bar text")
    (3, "even more foo text")
    ```
  - **Output:**
    ```
    ("this", 1),("is", 1),("the", 1), ("foo", 1), ("text", 1)
    ("this", 1),("is", 1),("the", 1), ("bar", 1), ("text", 1)
    ("even", 1),("more", 1),("foo", 1), ("text", 1)
    ```

Motivation

- Beispiel: Worthäufigkeit

- Phase 2 / Task 1
  - **Input:**
    ("this", 1), ("this", 1)
  - **Output:**
    ("this", 2)
- ...
- Phase 2 / Task 8
  - **Input:**
    ("text", 1), ("text", 1), ("text", 1)
  - **Output:**
    ("text", 3)

Motivation

- Beispiel: Worthäufigkeit

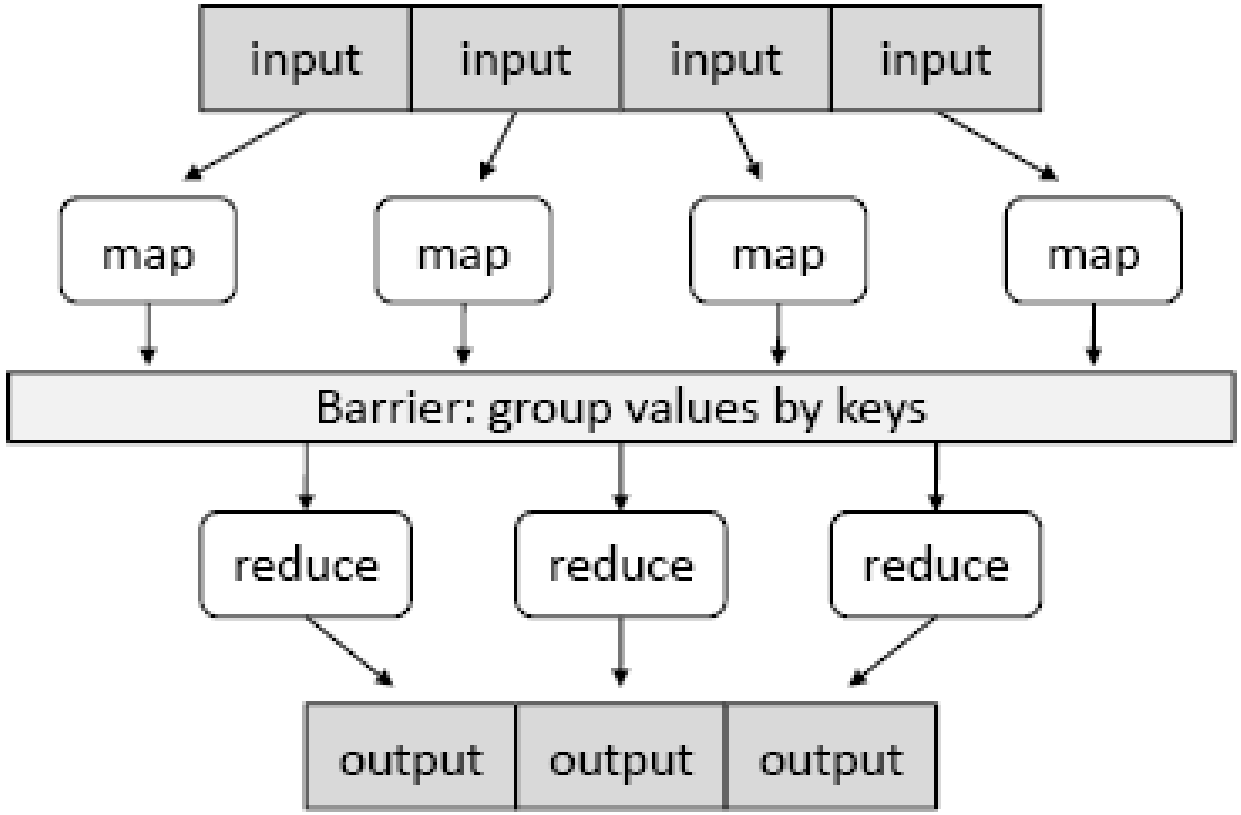
- Phase 2 / Task 1
  - **Input:**
    ("this", 1), ("this", 1)
  - **Output:**
    ("this", 2)
- ...
- Phase 2 / Task 8
  - **Input:**
    ("text", 1), ("text", 1), ("text", 1)
  - **Output:**
    ("text", 3)

text:3
is:2
more:1
even:1
foo:2
the:2
bar:1
this:2

Map/Reduce

- **Beobachtung:** Datenverarbeitungsschritte können oft mit Hilfe der zwei Funktionen map() und reduce() beschrieben werden
  - map – Bildet ein Eingabepaar (k_1, v_1) auf eine Liste von Ausgabepaaren (k_2, v_2) ab, z.B. **map(k_1, v_1) -> list (k_2, v_2)**
  - reduce – Erhält als Eingabe einen Schlüssel und eine Liste von Werten und bildet die Eingabe auf ein (oder auch mehrere) Ausgabepaar (k_2, v_2) ab, z.B. **reduce(k_2, list(v_2)) -> (k_2, v_3)**
- Map => Reduce
  - Gruppiere alle Zwischenergebnisse mit gleichem Schlüssel **k_2** und leite die Liste von Werten **list(v_2)** an reduce() weiter

Map/Reduce


input
input
input
input
map
map
map
map
Barrier: group values by keys
reduce
reduce
reduce
output
output
output

Map/Reduce

- Beispiel: Worthäufigkeit
  - map()
    `emit (word, 1)` für jedes Wort im Dokument
  - reduce()
    summiert alle Werte für ein Wort auf
    `emit (word, total count)`

Map/Reduce on Hadoop

- Beispiel: Worthäufigkeit (Mapper)
-

```
01: public static class TokenizerMapper extends Mapper<Object,
                                                Text, Text, IntWritable>
02: {
03:  private final static IntWritable one = new IntWritable(1);
04:  private Text word = new Text();
05:  public void map(Object key, Text value, Context context)
                        throws IOException, InterruptedException
06:  {
07:   StringTokenizer itr =new StringTokenizer(value.toString());
08:   while (itr.hasMoreTokens()) {
09:    word.set(itr.nextToken());
10:    context.write(word, one);
11:   }
12:  }
13: }
```

Map/Reduce on Hadoop

- Beispiel: Worthäufigkeit (Reducer)
-

```
01: public static class IntSumReducer extends Reducer
                          <Text, IntWritable, Text, IntWritable>
02: {
03:  private IntWritable result = new IntWritable();
04:  public void reduce(Text key, Iterable<IntWritable> values,
        Context context) throws IOException, InterruptedException
05:  {
06:   int sum = 0;
07:   for (IntWritable val : values) {
08:    sum += val.get();
09:   }
10:   result.set(sum);
11:   context.write(key, result);
12:  }
13: }
```

Map/Reduce

- Combiner
- Beispiel: Worthäufigkeit
  - Mapper erzeugen für eine Eingabe wiederholt gleiche Ausgaben, z.B. **("is", 1)**
  - (Fast) alle Ausgaben gehen über das Netzwerk (shuffle process)
  - Problem wird durch Anwendung einer Combiner-Methode auf dem Map-Rechner vermindert
  - Verwende Code von reduce() wird für combine()
    - **job.setCombinerClass(IntSumReducer.class);**
    - **job.setCombinerClass(InvertedIndexReducer.class);**

MapReduce auf Hadoop

- Open Source Implementation von Apache
- Version 0.20
- Wer setzt Hadoop ein? http://wiki.apache.org/hadoop/PoweredBy
  - Yahoo!
    - *Biggest cluster: 2000 nodes, used to support research for Ad Systems and Web Search.*
  - Amazon
    - *Process millions of sessions daily for analytics, using both the Java and streaming APIs. Clusters vary from 1 to 100 nodes.*
  - Facebook
    - *Use Hadoop to store copies of internal log and dimension data sources and use it as a source for reporting/analytics. 600 machine cluster*.
  - ...