Information Systems Group

Hasso Plattner Institut | Universität Potsdam

Advanced MapReduce Algorithms on Hadoop

Projektseminar Master

Felix Naumann

Alexander Albrecht

Christoph Böhm

Agenda

- **MapReduce**
- Organisatorisches
- Themen

Motivation

- Problem: Verarbeiten großer Datenmengen (z.B. Wikipedia Corpus)
  - Häufigkeiten zählen, Invertierten Index berechnen, ...
- Alternative Szenarien
  - Log-File Analyse
  - Google Page Rank
  - Wetterdaten …
- Master-Projektseminar im WS 09/10
  - Advanced Map-Reduce-Algorithms on Hadoop

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

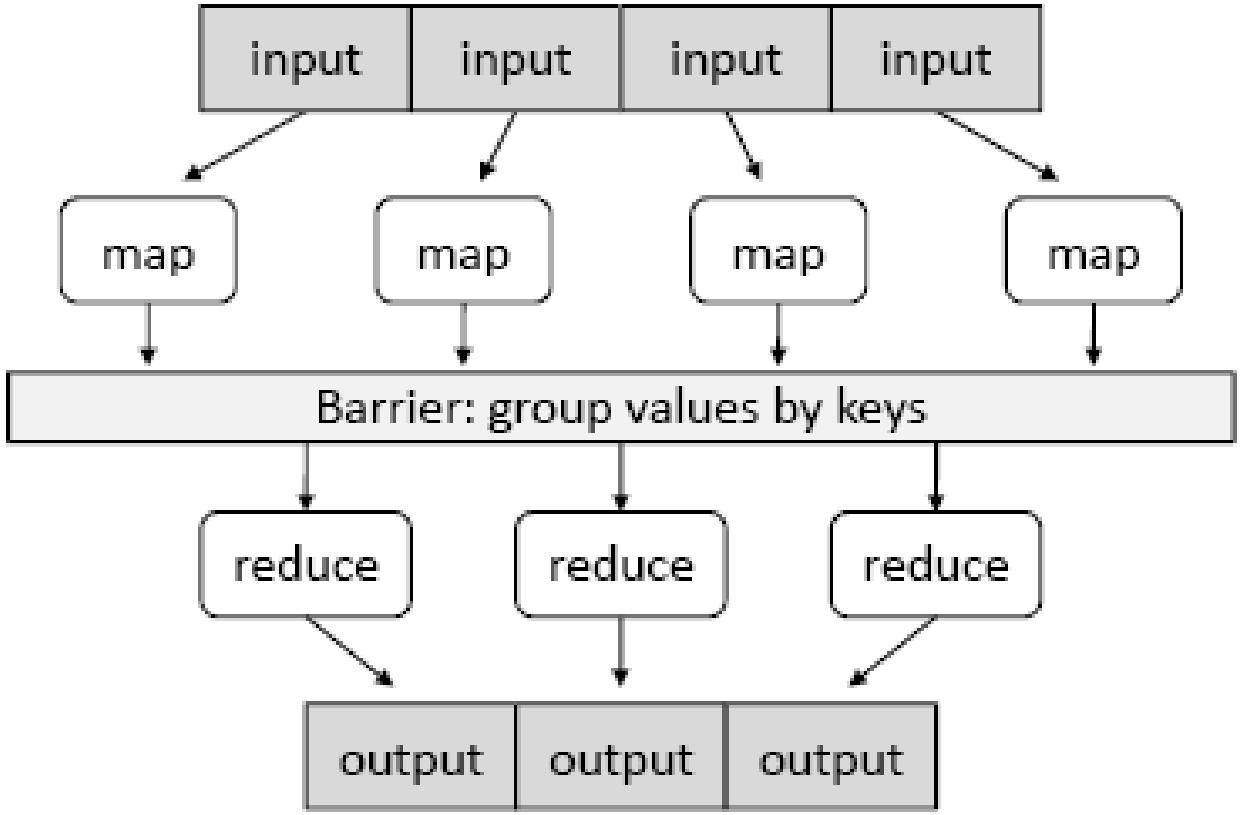
- Phase 2 / Task 1
  - **Input:**
    ("this", 1), ("this", 1)
  - **Output:**
    ("this", 2)
- ...
- Phase 2 / Task 8
  - **Input:**
    ("text", 1), ("text", 1), ("text", 1)
  - **Output:**
    ("text", 3)

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: }
```

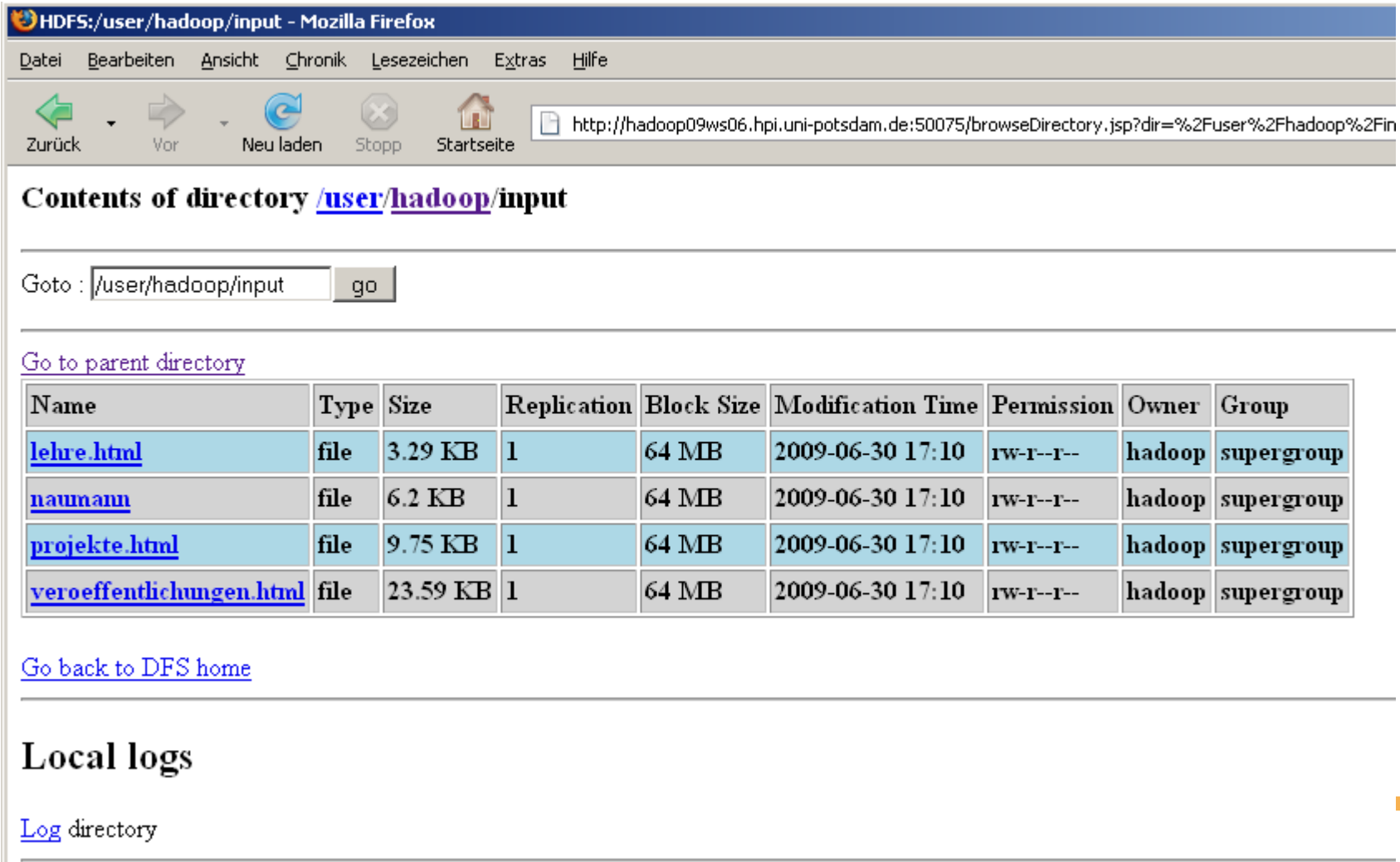
Map/Reduce on Hadoop

- 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

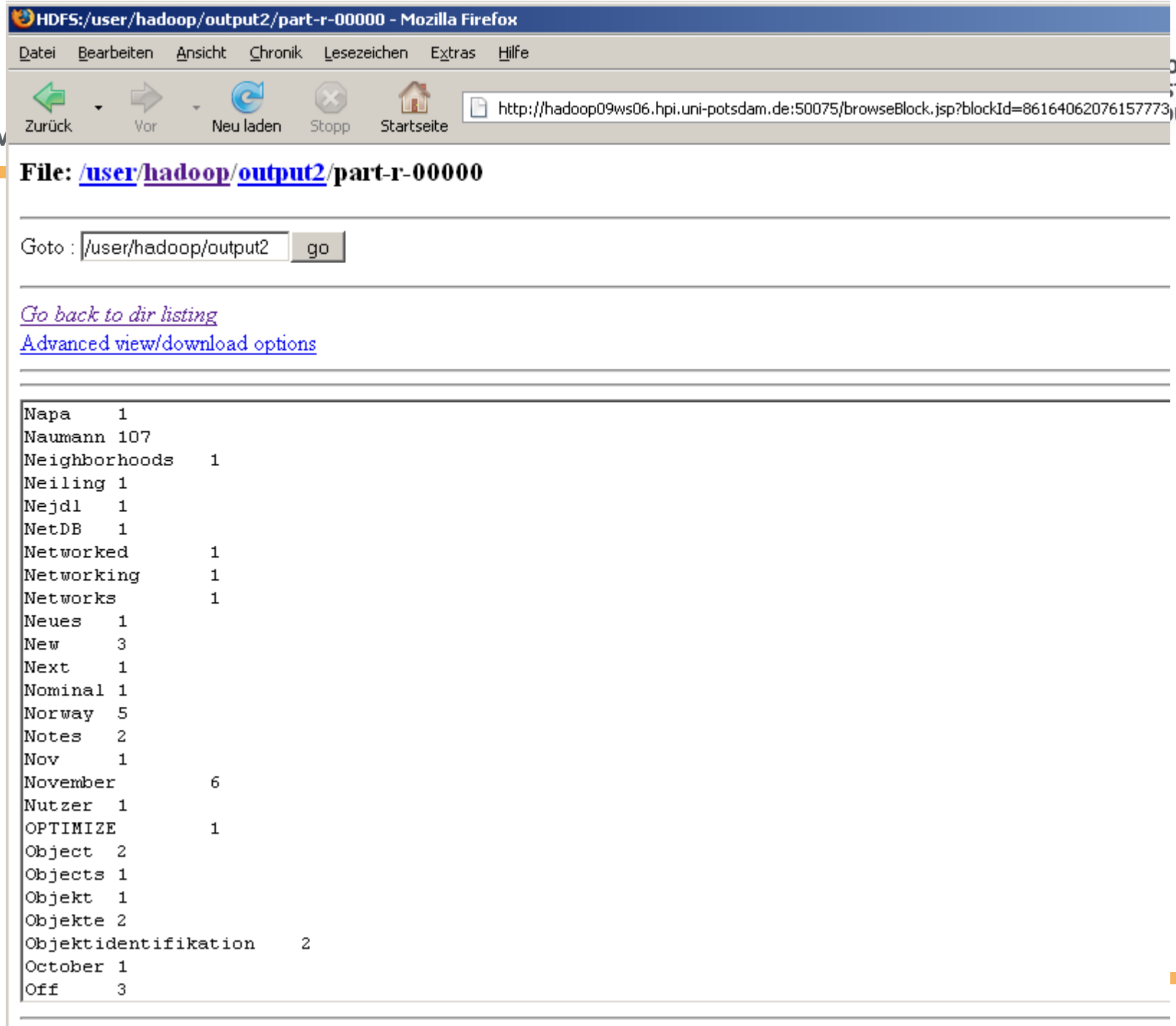
- `./bin/hadoop fs -put web input`



Map/Reduce

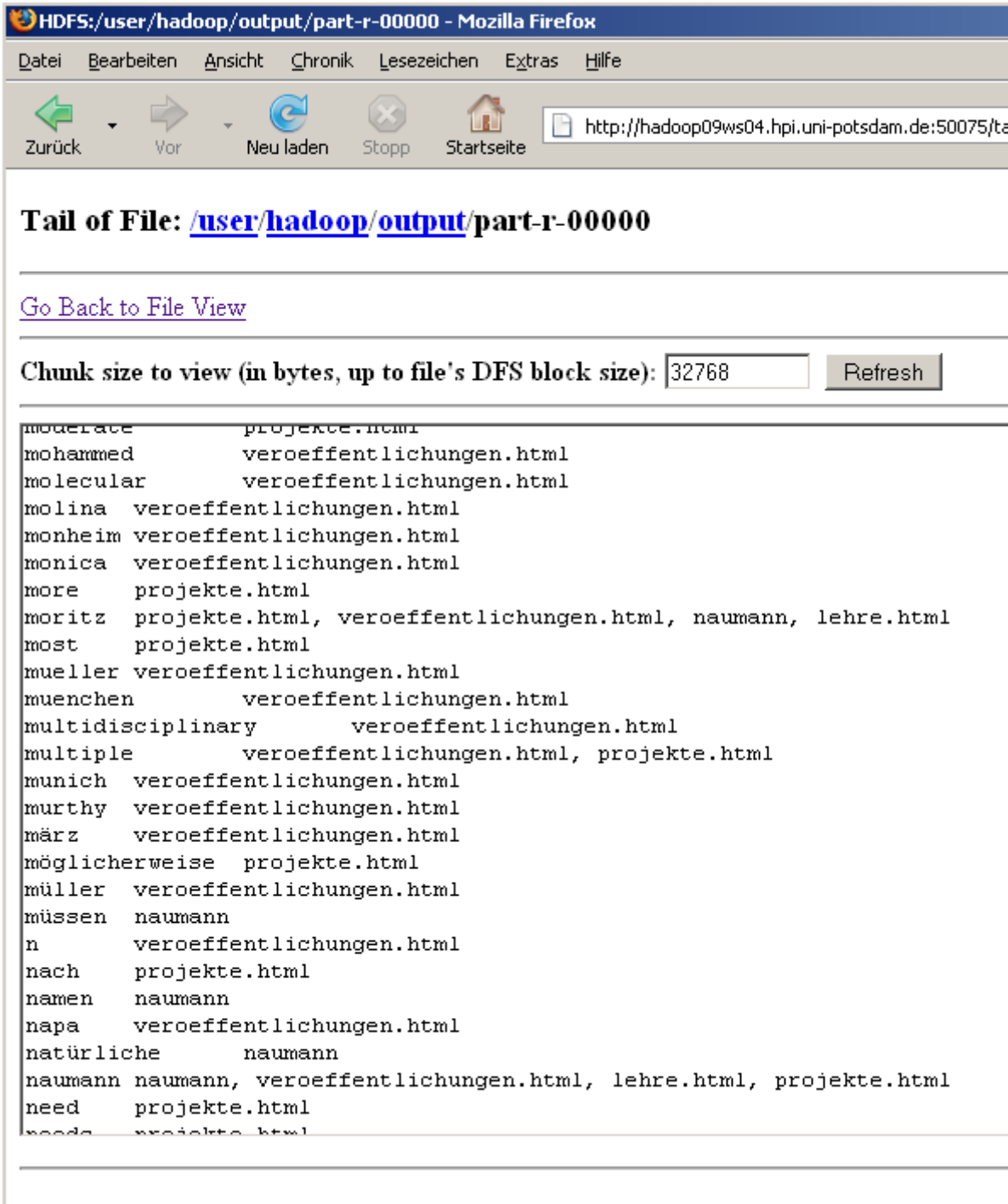
- `./bin/hadoop jar SearchEngines.jar WordCount input output2`

```
doop@hadoop09ws10 hadoop]$ ./bin/hadoop jar SearchEngines.jar WordCount input out
07/02 12:06:48 INFO input.FileInputFormat: Total input paths to process : 4
07/02 12:06:48 INFO mapred.JobClient: Running job: job_200905181507_1017
07/02 12:06:49 INFO mapred.JobClient:  map 0% reduce 0%
07/02 12:06:59 INFO mapred.JobClient:  map 50% reduce 0%
07/02 12:07:00 INFO mapred.JobClient:  map 75% reduce 0%
07/02 12:07:02 INFO mapred.JobClient:  map 100% reduce 0%
07/02 12:07:08 INFO mapred.JobClient:  map 100% reduce 25%
07/02 12:07:17 INFO mapred.JobClient:  map 100% reduce 100%
07/02 12:07:19 INFO mapred.JobClient: Job complete: job_200905181507_1017
07/02 12:07:19 INFO mapred.JobClient: Counters: 18
07/02 12:07:19 INFO mapred.JobClient:   Job Counters
07/02 12:07:19 INFO mapred.JobClient:     Launched reduce tasks=1
07/02 12:07:19 INFO mapred.JobClient:     Rack-local map tasks=3
07/02 12:07:19 INFO mapred.JobClient:     Launched map tasks=4
07/02 12:07:19 INFO mapred.JobClient:     Data-local map tasks=1
07/02 12:07:19 INFO mapred.JobClient:   FileSystemCounters
07/02 12:07:19 INFO mapred.JobClient:     FILE_BYTES_READ=34476
07/02 12:07:19 INFO mapred.JobClient:     HDFS_BYTES_READ=43856
07/02 12:07:19 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=69098
07/02 12:07:19 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=17411
07/02 12:07:19 INFO mapred.JobClient:   Map-Reduce Framework
07/02 12:07:19 INFO mapred.JobClient:     Reduce input groups=0
07/02 12:07:19 INFO mapred.JobClient:     Combine output records=2509
07/02 12:07:19 INFO mapred.JobClient:     Map input records=855
07/02 12:07:19 INFO mapred.JobClient:     Reduce shuffle bytes=27410
07/02 12:07:19 INFO mapred.JobClient:     Reduce output records=0
07/02 12:07:19 INFO mapred.JobClient:     Spilled Records=5018
```


HDFS:/user/hadoop/output2/part-r-00000 - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Zurück Vor Neu laden Stopp Startseite
http://hadoop09ws06.hpi.uni-potsdam.de:50075/browseBlock.jsp?blockId=8616406207615773
File: /user/hadoop/output2/part-r-00000
Goto : /user/hadoop/output2 go
Go back to dir listing
Advanced view/download options
Napa 1
Naumann 107
Neighborhoods 1
Neiling 1
Nejdl 1
NetDB 1
Networked 1
Networking 1
Networks 1
Neues 1
New 3
Next 1
Nominal 1
Norway 5
Notes 2
Nov 1
November 6
Nutzer 1
OPTIMIZE 1
Object 2
Objects 1
Objekt 1
Objekte 2
Objektidentifikation 2
October 1
Off 3

Map/Reduce

- **Beispiel**
  Invertierter Index
  - Ideen?



Map/Reduce on Hadoop

- Beispiel: Invertierter Index (Mapper)

```
01: public static class InvertedIndexMapper extends Mapper
                                    <LongWritable, Text, Text, Text> {
02:  private Text word = new Text();
03:  private final static Text location = new Text();
04:  public void map(LongWritable key, Text val, Context context)
                          throws IOException, InterruptedException {
05:   FileSplit fileSplit = (FileSplit)context.getInputSplit();
06:   location.set(fileSplit.getPath().getName());

07:   String line = val.toString().toLowerCase();
08:   StringTokenizer i=new StringTokenizer(line.toLowerCase());
09:   while (i.hasMoreTokens()) {
10:    word.set(i.nextToken());
11:    context.write(word, location);
12:   }
13:  }
14: }
```

Map/Reduce on Hadoop

- Beispiel: Invertierter Index (Reducer)
-

```
01: public static class InvertedIndexReducer extends Reducer
                                          <Text,Text,Text,Text> {
02:  public void reduce(Text key, Iterable<Text> values, Context
                context) throws IOException, InterruptedException {
03:   boolean first = true;
04:   StringBuilder toReturn = new StringBuilder();
05:   for (Text val : values) {
06:    String fileName = val.toString();
07:    if (toReturn.indexOf(fileName) < 0) {
08:     if (!first) toReturn.append(", ");
09:     first=false;
10:     toReturn.append(fileName);
11:    }
12:   }
13:   context.write(key, new Text(toReturn.toString()));
14:  }
15: }
```

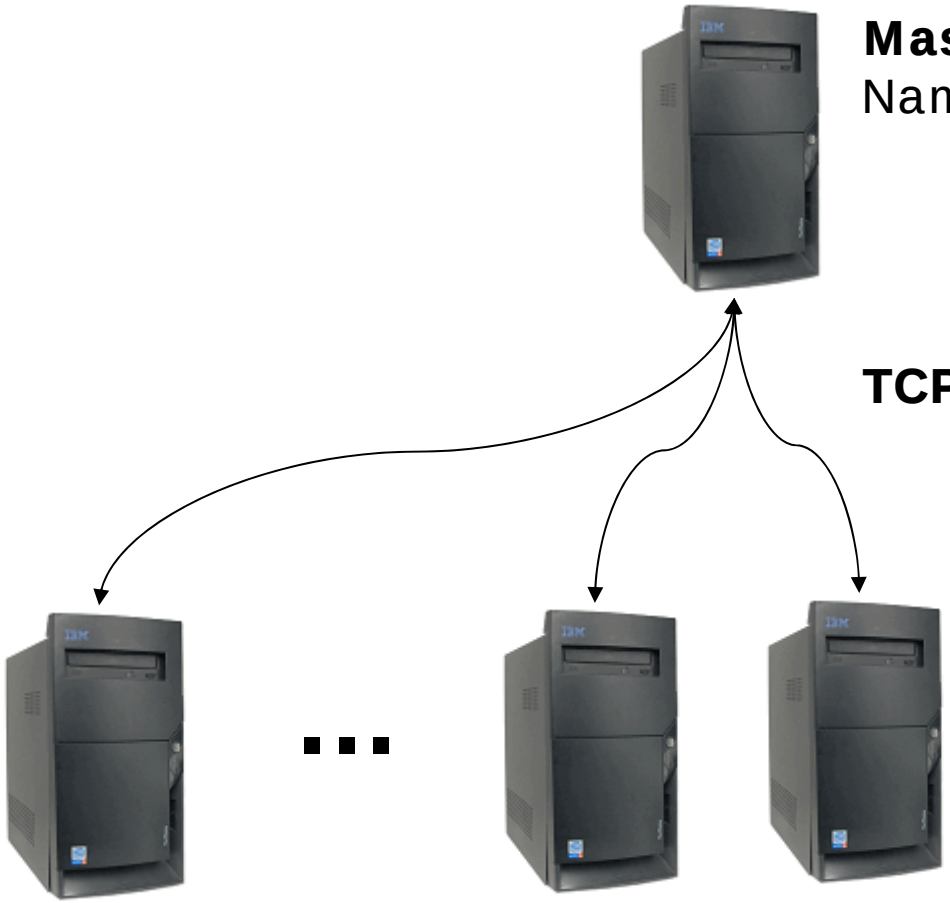
Map/Reduce

- 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*.
  - ...

Hadoop an der FG Naumann



Master
NameNode / JobTracker

TCP / IP

Slaves
DataNodes / TaskTracker

Agenda

- MapReduce
- **Organisatorisches**
- Themen

Organisatorisches

- Ablauf
  - Teilnehmer schicken Themenwunschliste an die Betreuer bis zum **23. Oktober**
  - Themen und Teams werden gemäß Wunschliste vergeben
  - Benachrichtigung erfolgt per E-Mail und im WWW
  - Paper lesen ab 24.10.09
  - Nächster Termin **27. Oktober**

Organisatorisches

- Anforderungen
  - Teilnahme an allen Seminarterminen
  - Anwesenheit bei Konsultationen
  - Implementierung einer MapReduce Lösung in Java
  - Präsentation der implementierten Lösung, inkl. Demo, als Vortrag am Ende des Semesters. Dauer: 30 Minuten Vortrag + 15 Minuten Diskussion
  - Ausarbeitung (4 - 5 Seiten)

Organisatorisches

- Abschlussnote berücksichtigt die folgenden Punkte
  - Implementierte Lösung
  - Präsentation / Vortrag
  - Ausarbeitung
  - Mündliche Beteiligung