MapReduce am Beispiel MongoDB

Joëlle Simonet, Lucius Bachmann

12. November 2013

Inhaltsverzeichnis

Motivation

- Das automatische Erfassen von Daten (Log-Dateien, Aktienkurse, Social Networks, ...) erzeugt riesige Datenmengen
- Speicherkapazitäten von Festplatten sind in den letzten Jahrzehnten stark angestiegen
- Mittlere Zugriffszeiten und Transferraten haben sich vergleichsweise wenig verbessert

Problem

Ein Rechen-Job soll auf einer riesigen Datenmenge ausgeführt werden. Die Datenmege ist auf verschiedenen Systemen verteilt.

Entstehung des Programmiermodells

- Berechnungen lassen sich leichter verteilen als Daten
- Wir teilen den Job in zwei Phasen auf, die sich gut verteilen und parallelisieren lassen

$\rightarrow$ Google, Inc. stellt die Idee des MapReduce vor

MapReduce: Simplified Data Processing on Large Clusters[1]

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The runtime system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required intermachine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terrabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

[1] Jeffrey Dean and Sanjay Ghemawat, (Google, Inc.), 2004, S.1.

Funktionen

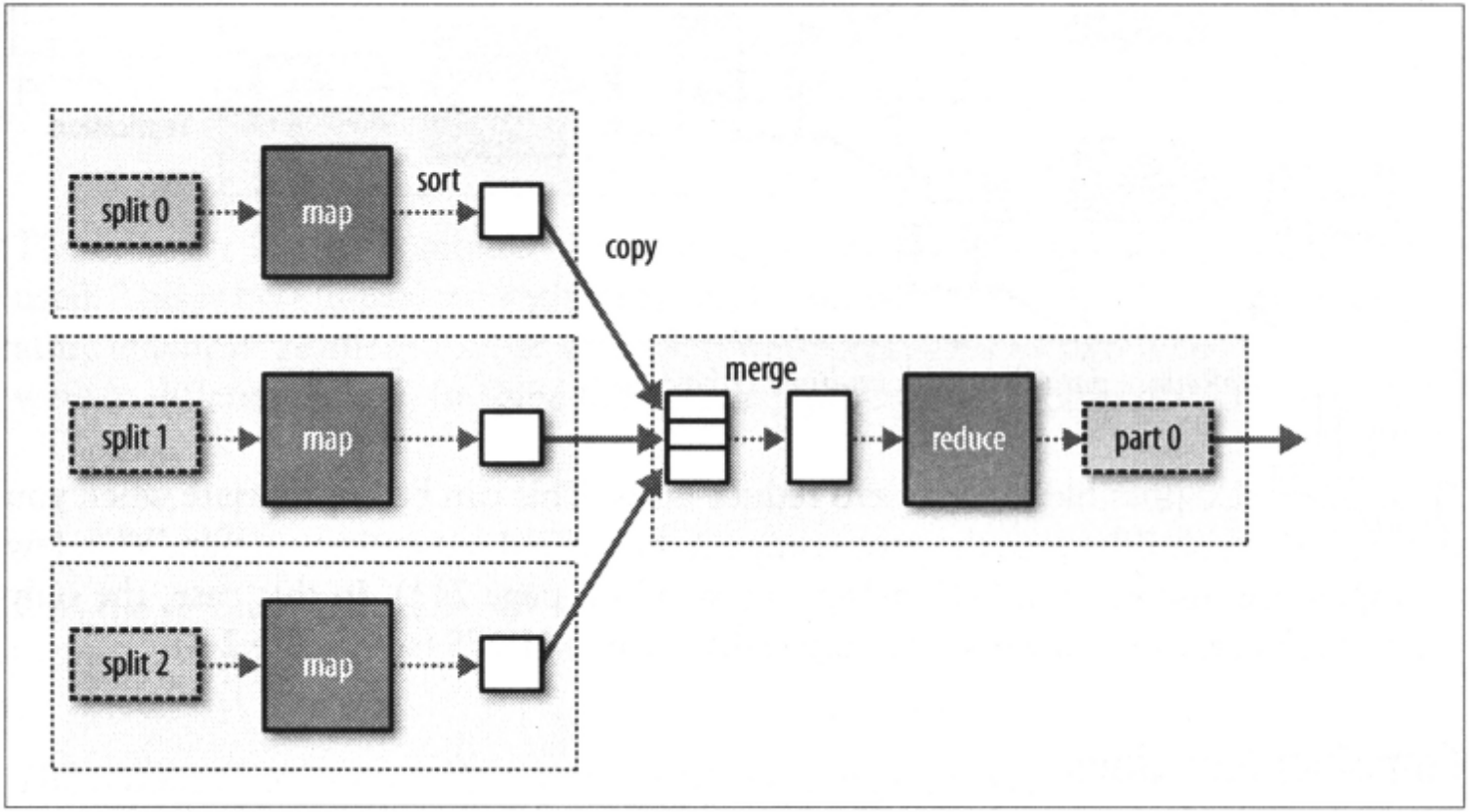
- **Map:** (k1, v1) $\rightarrow$ list(k2, v2)
  Abbildung von Key/Value-Paaren auf andere Key/Value-Paare
- **Reduce:** (k2, list(v2)) $\rightarrow$ list(v2)
  Zusammenfassung der Zwischenergebnisse

Schritte I

1. unstrukturierte Daten einlesen
2. Zuordnung von Datei-Inhalt zu Positionen
3. Map: Transformieren dieser Schlüssel-/Werte-Paare in intermediate Schlüssel-/Werte-Paare
   - benötigte Daten werden herausgenommen
   - es entstehen viele Key/Value-Paare

Schritte II

4. Map: Erzeugen von gruppierten Schlüssel-/Werte-Paaren
   - Sortieren der Schlüssel
   - Zuordnen von Werten zu einem Schlüssel
   - Jeder Mapper schreibt den sortierten Output ins Filesystem
5. Reduce: Für jeden Key aus dem Map-Schritt die Reduce-Funktion aufrufen
   - Zusammenfassung der Werte
   - Pro Schlüssel nur noch ein Wert
6. Output der Ergebnisse

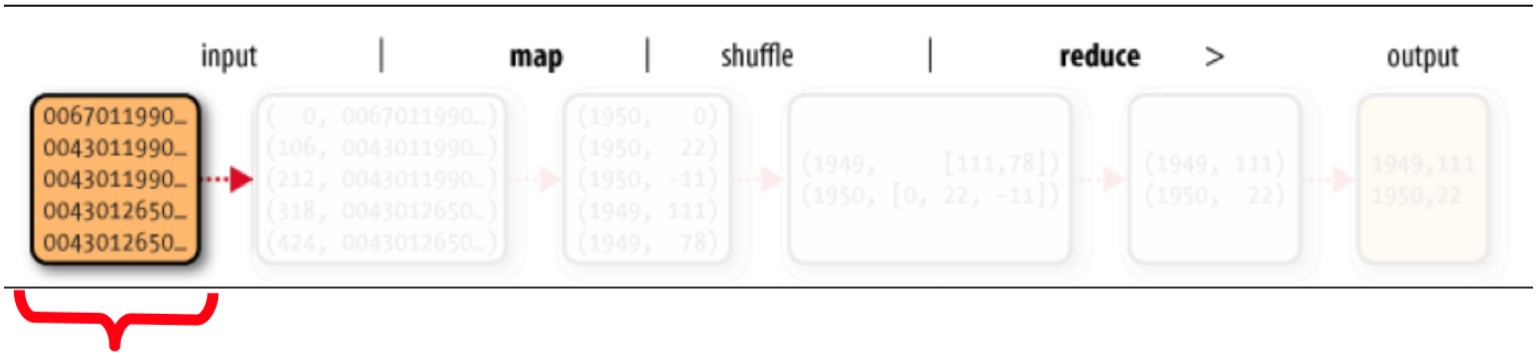

split 0
map
sort
split 1
map
split 2
map
copy
merge
reduce
part 0

Beispiel I

Ziel:
Wir wollen die höchste Temperatur eines Jahres finden.

Daten-Input:
Wir haben unstrukturierte Wetterdaten.

Beispiel II



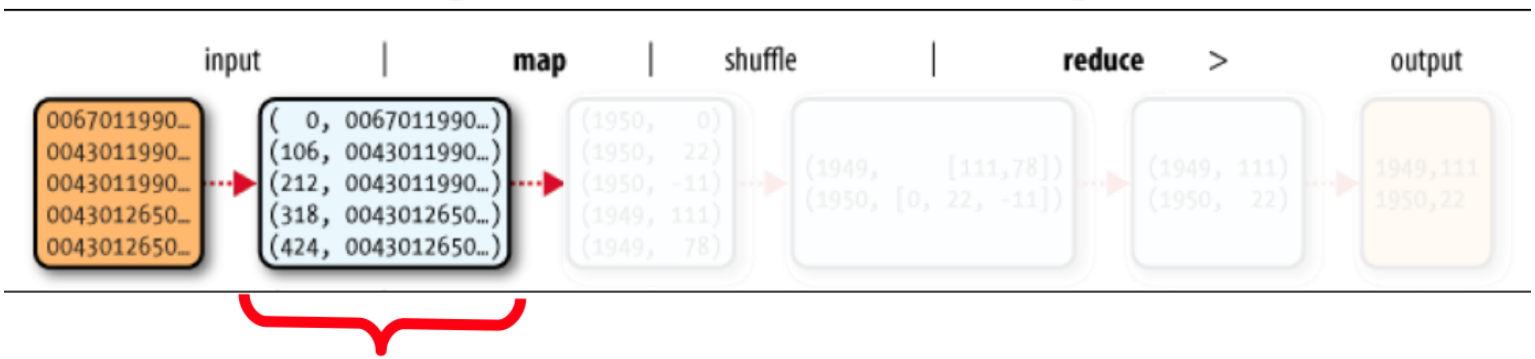
1. unstrukturierte Wetterdaten einlesen

```
0029029070999991901010106004+64333+023450FM-12+00059
9999V0202701N015919999999N0000001N9-00781+9999910200
1ADDGF108991999999999999999999

0029029070999991901010113004+64333+023450FM-12+00059
9999V0202901N008219999999N0000001N9-00721+9999910200
1ADDGF104991999999999999999999
```

1901-01-01
13:00
-7,2°C

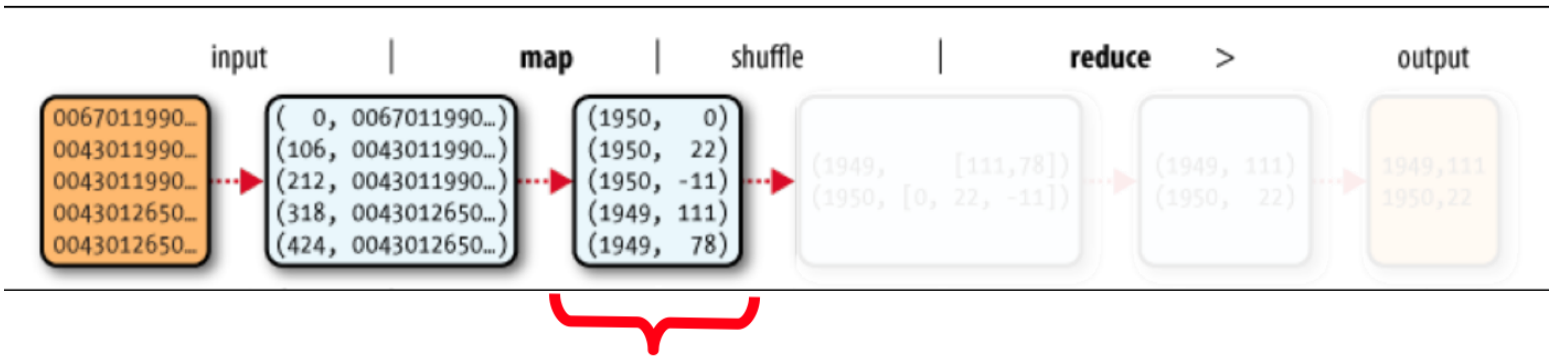
Beispiel III



2. Zuordnung von Datei-Inhalt zu Positionen

- Jede Zeile wird anhand des Byte-Offsets identifiziert
- Byte-Offset verweist jeweils auf den Beginn der Zeile

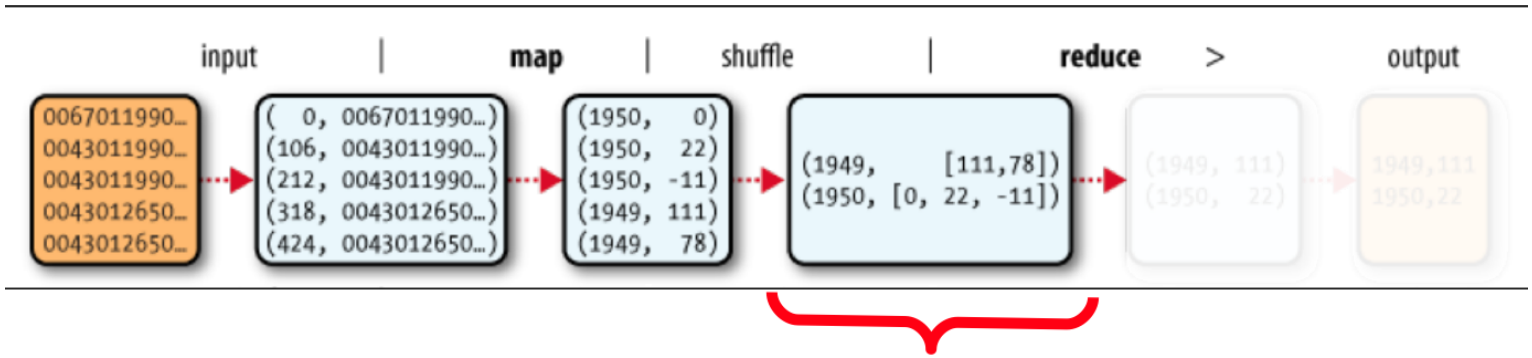
Beispiel IV



3. Map: Transformieren dieser Schlüssel-/Werte-Paare in intermediate Schlüssel-/Werte-Paare

- benötigte Daten (Jahreszahlen, Temperaturwerte) werden aus den Zeilen extrahiert
- es entstehen viele Key/Value-Paare der Form (Jahr, Temperatur)

Beispiel V

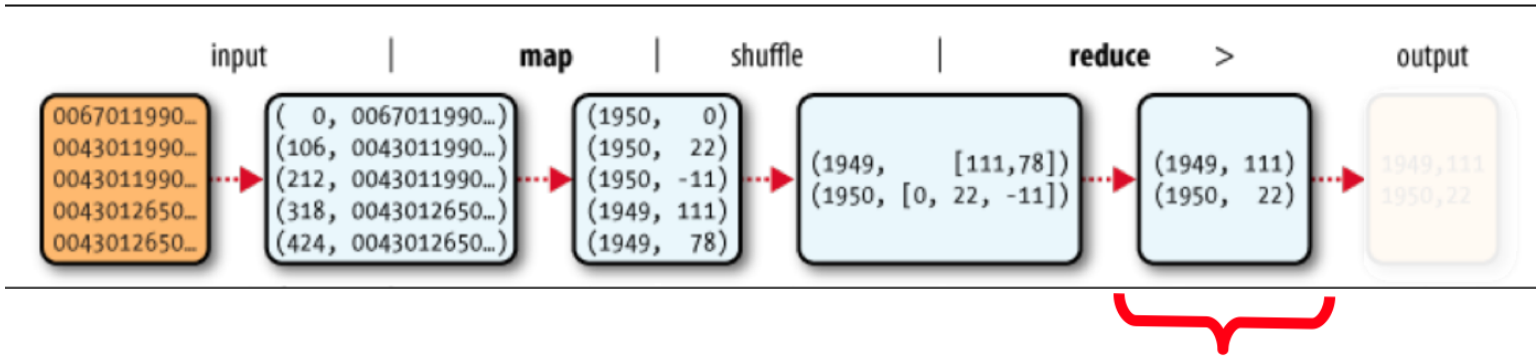


4. Map: Erzeugen von gruppierten Schlüssel-/Werte-Paaren

- Sortieren der Schlüssel
- Zuordnen von Werten zu einem Schlüssel: alle Temperaturwerte zu einem Jahr werden gesammelt
- Jeder Mapper schreibt den sortierten Output ins Filesystem

Jahr	Temperatur
1949	111
	78
1950	0
	22
	-11

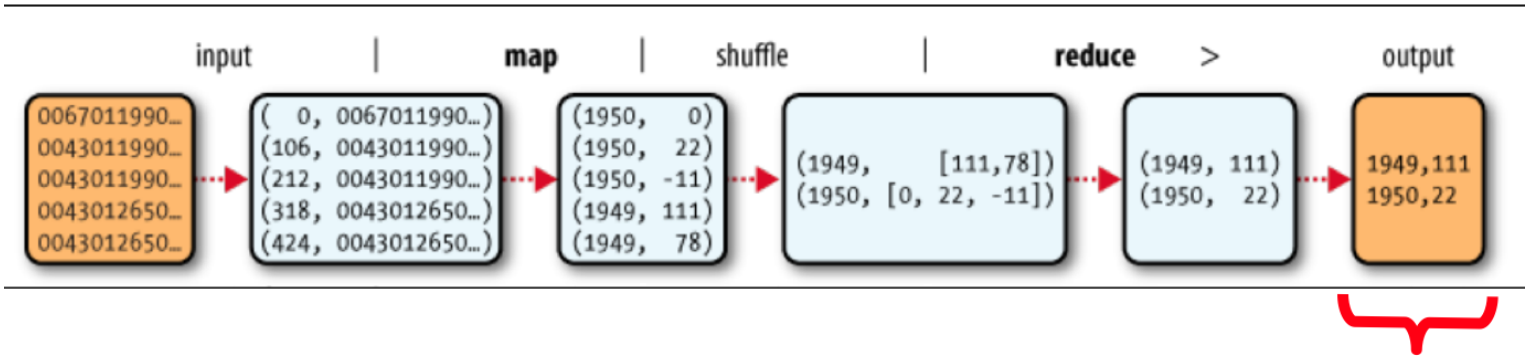
Beispiel VI



5. Reduce:
   - Zusammenfassung der Werte, hier: Maximum finden
   - Pro Schlüssel nur noch ein Wert

Jahr	Temperatur
1949	111
1950	22

Beispiel VII



6. Output der Ergebnisse

MapReduce:
- Batch-Verarbeitung von Daten
- unstrukturierte/semi-strukturierte Daten
- nicht-normalisierte Daten

Relationale DBS:
- gezielte Abfragen und Updates
- strukturierte Daten
- normalisierte Daten

Praxis in MongoDB

Hintergrund

- Okt. 2007 entwickelt von 10gen (Heute MongoDB inc.) als Teil eines PaaS Systems.
- 2009 wechselte die Firma auf Open Source, 10gen bietet kommerziellen Support dafür an.
- MongoDB ist unter GNU AGPL, die Language Drivers unter der Apache License frei verfügbar.
- eBay, SourceForge, NY Times, SAP in SAP PaaS, Forbes, The Guardian, CERN LHC verwenden MongoDB

Begriffe

- Collection : Sammlung von Documents, tritt als eine Datei im Ordner Data auf. (SQL : table)
- Document: Ein Eintrag in der Collection (SQL : entry)
- replica set : Eine Gruppe von mongod Prozessen, ein primary und n secondary. Die Secondaries duplizieren die Aktionen des Primarys auf die Kopien der Datenbank.
- Sharding: Verteilen der Datenbank auf mehrere Server. 1 Shard $=$ 1 Server $+$ Teil der Datenbank
- namespace : dbname.collectionname

CRUD Operations

- Create:
  db.person.insert({vorname : “lucius“, nachname : “bachmann“});
  SQL: INSERT INTO person(vorname, nachname) VALUES (‘lucius‘, ‘bachmann‘);
- Read:
  db.person.find({hausnr : {$gt : 5 , $lt : 20}}, {vorname : 1, nachname : 1});
  SQL: SELECT vorname, nachname FROM person WHERE hausnr BETWEEN 5 AND 20;

CRUD Operations 2

- Update:
  db.person.update({vorname : “lucius“} , {nachname : “baumann“}, {upsert : false}, {multi : true} });
  SQL: UPDATE person SET nachname = ‘baumann‘WHERE vorname = ‘lucius‘;
- Delete;
  db.person.remove({vorname :“Lucius“}, 0);
  SQL: DELETE FROM person WHERE vorname = “Lucius “;

Syntax

```
db.person.mapReduce(
                  {
                    map: <function>,
                    reduce: <function>,
                    out: <output>,
                    query: <document>,
                    sort: <document>,
                    limit: <number>,
                    finalize: <function>,
                    scope: <document>,
                    jsMode: <boolean>,
                    verbose: <boolean>
                  }
                )
```

map (Bsp Durchschnitt)

```
1  function map(){
2    var size=this.size;
3    emit(this.countries, {totalSize : size,
4                numAdded : 1
5              }
6      );
7  }
```

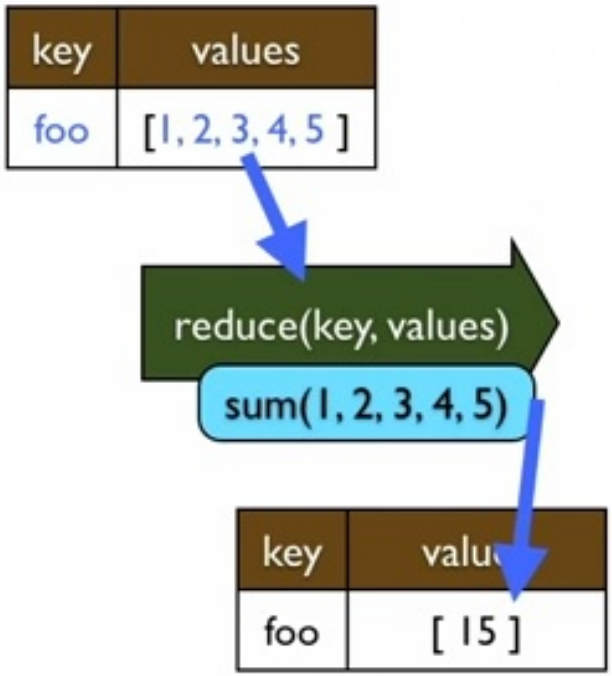
reduce (Bsp Durchschnitt)

```
1  function reduce(key, values){
2    var returnObj = { totalSize : 0,
3              numAdded : 0
4              };
5    for (var i in values){
6      returnObj.totalSize += values[i].totalSize;
7      returnObj.numAdded+= values[i].numAdded;
8      }
9    return returnObj;
10 }
```

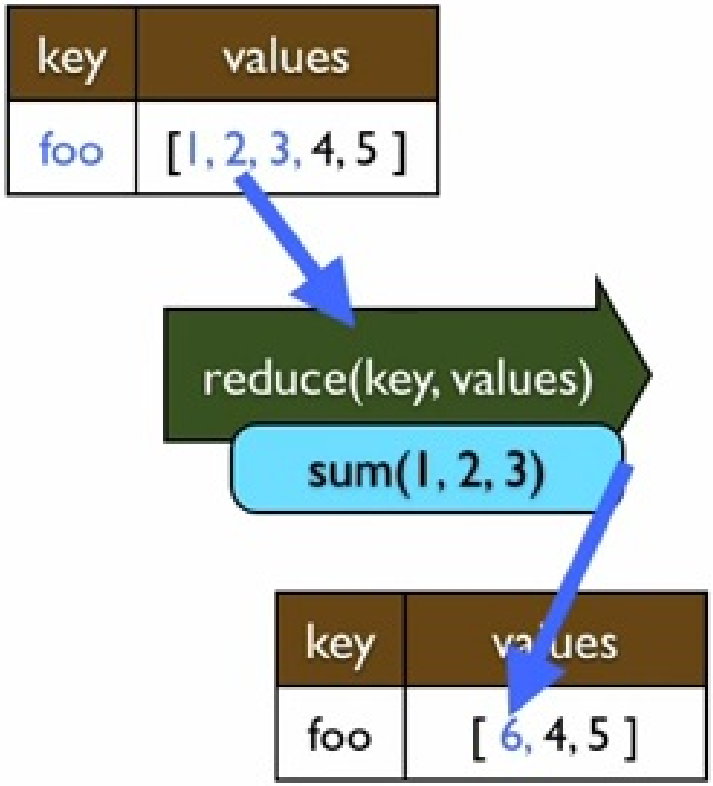
finalize (Bsp Durchschnitt)

```
1  function finalize (key, reducedValue){
2    if(reducedValue == null){}
3    else{
4      var returnObj={ totalSize : reducedValue.
           totalSize,
5                numAdded : reducedValue.numAdded,
6                average : reducedValue.totalSize/
                   reducedValue.numAdded
7              };
8      return returnObj;
9      }
10 }
```

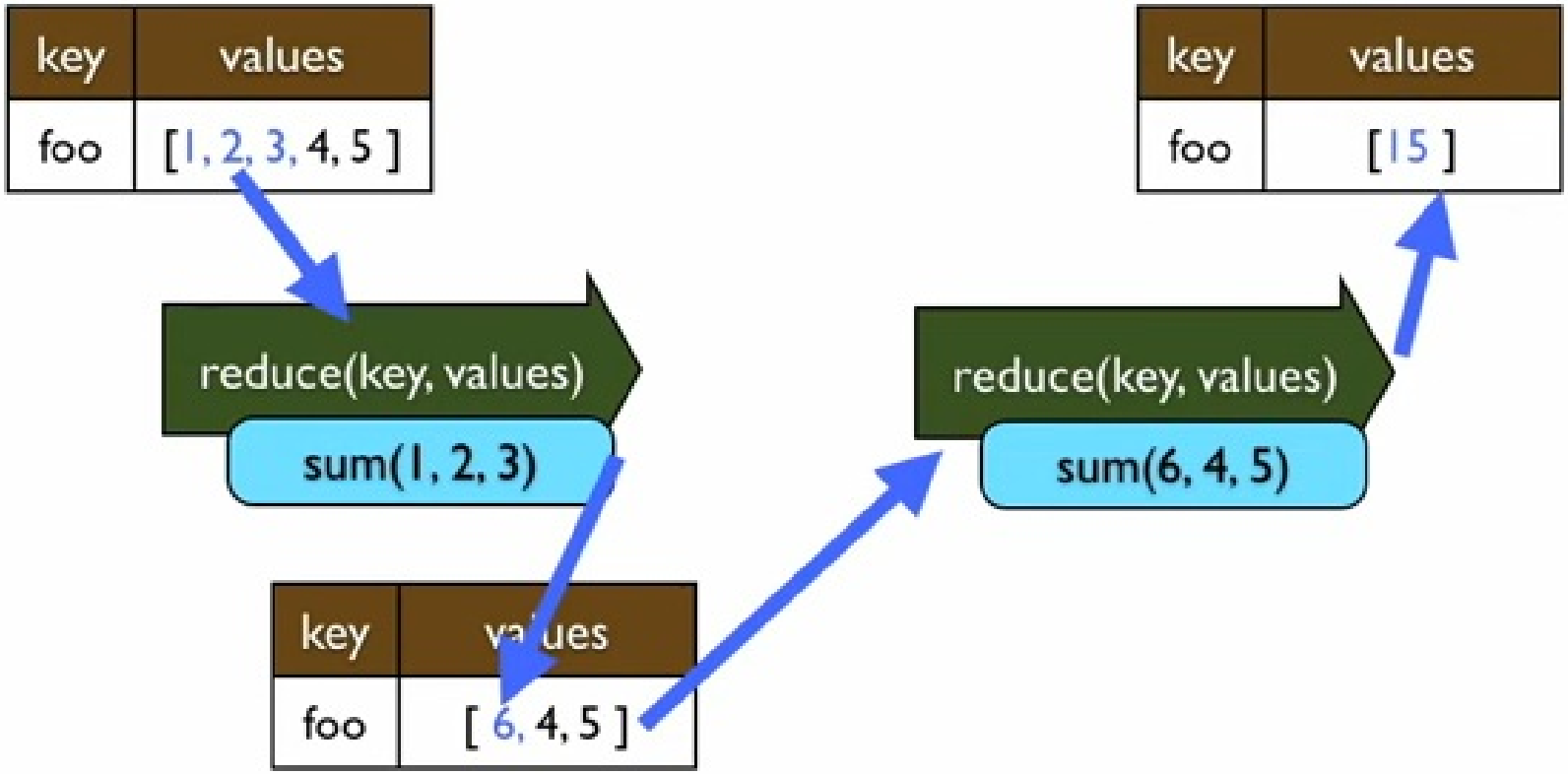
Stolperstein



Stolperstein



Stolperstein



Idempotence

- reduce(key, value) gleicher Typ wie reduce(key, reduce(key, value))
- in der reduce function keine Arrays returnieren
- Wenn es sein muss, dann in Objekte verpacken.
- Nach dem Key sortieren

Vorinformation 1

- Verbindet euch mit [Lucius IP]

Vorinformation 2

```
1  MongoDB> db.createCollection("MeinName_output");
2  MongoDB> var map=function(){emit(1,1);}
3  MongoDB> var reduce = function (key, values)
4  ...{return 1;}
5  MongoDB> var finalize = function(key, value)
6  ...{return 1;}
7  MongoDB> db.testdata.mapReduce(map, reduce,
8  ...{out : "MeinName_output",
9  ...finalize : finalize}).find()
10
11 { "_id" : 1, "value" : 1 }
12
13 MongoDB> db.MeinName_output.find()
14
15 { "_id" : 1, "value" : 1 }
```

Aufgabe

- Stichprobe 30‘000, Attr: name, countries, tshirtcolor, size
- Von jedem countries wollen wir wissen:
  - Alle vorkommenden Namen
  - Durchschnittsgrösse
  - Am häufigsten getragene T-Shirt Farbe

Noch Fragen?

Fragen?