Cloud & Big Data Technologien

Prof. Dr. Wolfgang Blochinger

Hintergrund: Google, Facebook und Co.

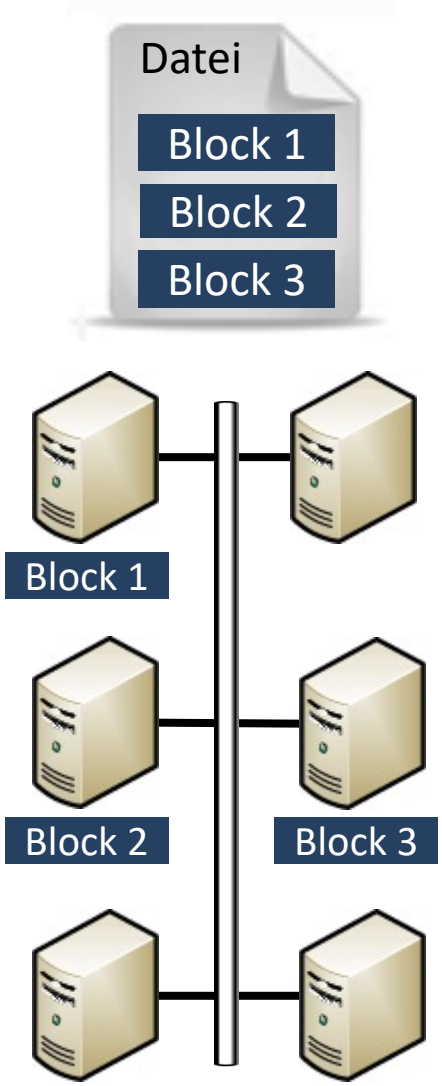
- Beispiel Google:
  - Datenbestand 100 Petabyte (1 Petabyte = 1015 Byte)
  - 100 Terabyte neue Daten pro Tag (1 Terabyte = 1012 Byte)
- Beispiel Facebook:
  - Datenbestand 21 Petabyte
  - 25 Terabyte Log-Daten pro Tag
- Zwei wesentliche Probleme:
  - Speicherung extrem großer Datenmengen
  - Geringe Verarbeitungsgeschwindigkeit: Auslesen von 1TB Daten dauert ca. 2 Stunden.
- **Lösung: Verteilte Datenspeicherung und -verarbeitung in Rechen-Clustern.**

Hintergrund: Google, Facebook und Co.

Map/Reduce Framework

- **Map/Reduce** wurde ursprünglich von Google zur Berechnung des Page Rank entwickelt.
- Grundprinzip von Map/Reduce: “Function Shipping”
  - Daten werden in einem verteilten Dateisystem gespeichert.
  - Funktionen zur parallelen Verarbeitung (“Mapper”, “Reducer”) werden am Speicherort der Daten ausgeführt.
- **Apache Hadoop:** Open Source Implementierung von Map Reduce
  - Es entwickelte sich ein umfassendes “Hadoop Ökosystem”.
  - Es werden Hadoop Cluster mit tausenden von Knoten betrieben.
    - Bsp. Facebook, Yahoo and Twitter.

Verteilte Datenspeicherung in Clustern

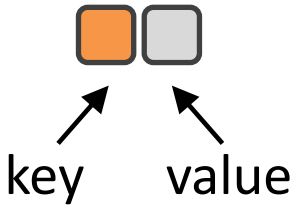


- **Verteilte Datenspeicherung:**
  - Eine Datei wird in mehrere, gleichgroße Datenblöcke aufgeteilt.
  - Blöcke werden auf verschiedenen Rechnern des Clusters gespeichert.
- **Vorteile:**
  - Speicherung sehr großer Dateien möglich (größer als einzelne Festplatte).
  - Blöcke einer Datei können parallel verarbeitet (z.B. durchsucht) werden.
- **Problem:** Programmierung der parallelen Abläufe.

MapReduce ist Programmierkonzept für die parallele und verteilte Verarbeitung großer Datenmengen.

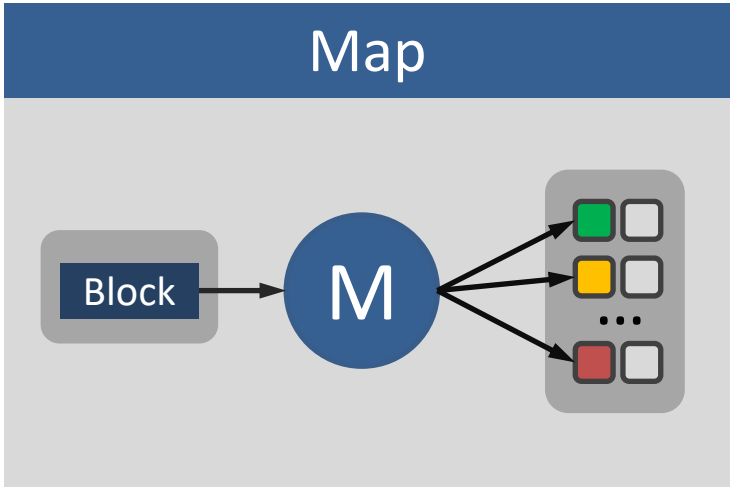
Grundlagen von MapReduce

- MapReduce basiert auf dem **Skeleton-Prinzip:**
  - Struktur und Ablauf des Gesamtprogramms sind durch ein Programm-Skelett fest vorgegeben.
  - Programmierer schreibt Programmcode für zwei Funktionen: **Map** und **Reduce**.
  - Map und Reduce bilden die anwendungsspezifischen Komponenten des Skeletts.
- Ein- und Ausgabe von Map und Reduce bestehen aus **Key/Value Paaren**.
  - Beispiele: (4711, 1.9) oder („Haus“, 1)
  - Beliebige Datentypen für Key und Value möglich.
  - Grafische Notation:



Map-Funktion

Aufgabe: Extraktion und Klassifizierung benötigter Informationen aus gespeicherten Daten.

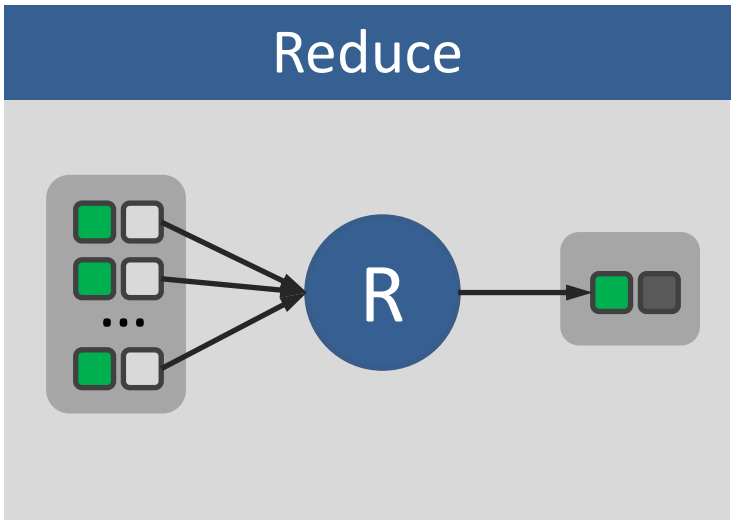


- **Eingabe:**
  - Datenblock
- **Ausgabe:**
  - Liste von Key/Value Paaren
  - Schlüssel kodiert typischerweise eine Eigenschaft des Werts

- Beispiel „Grep“: Extraktion aller Zeilen eines Textdokuments, welche vorgegebene Worte enthalten.
  - Key: Wort
  - Value: Textzeile

Reduce-Funktion

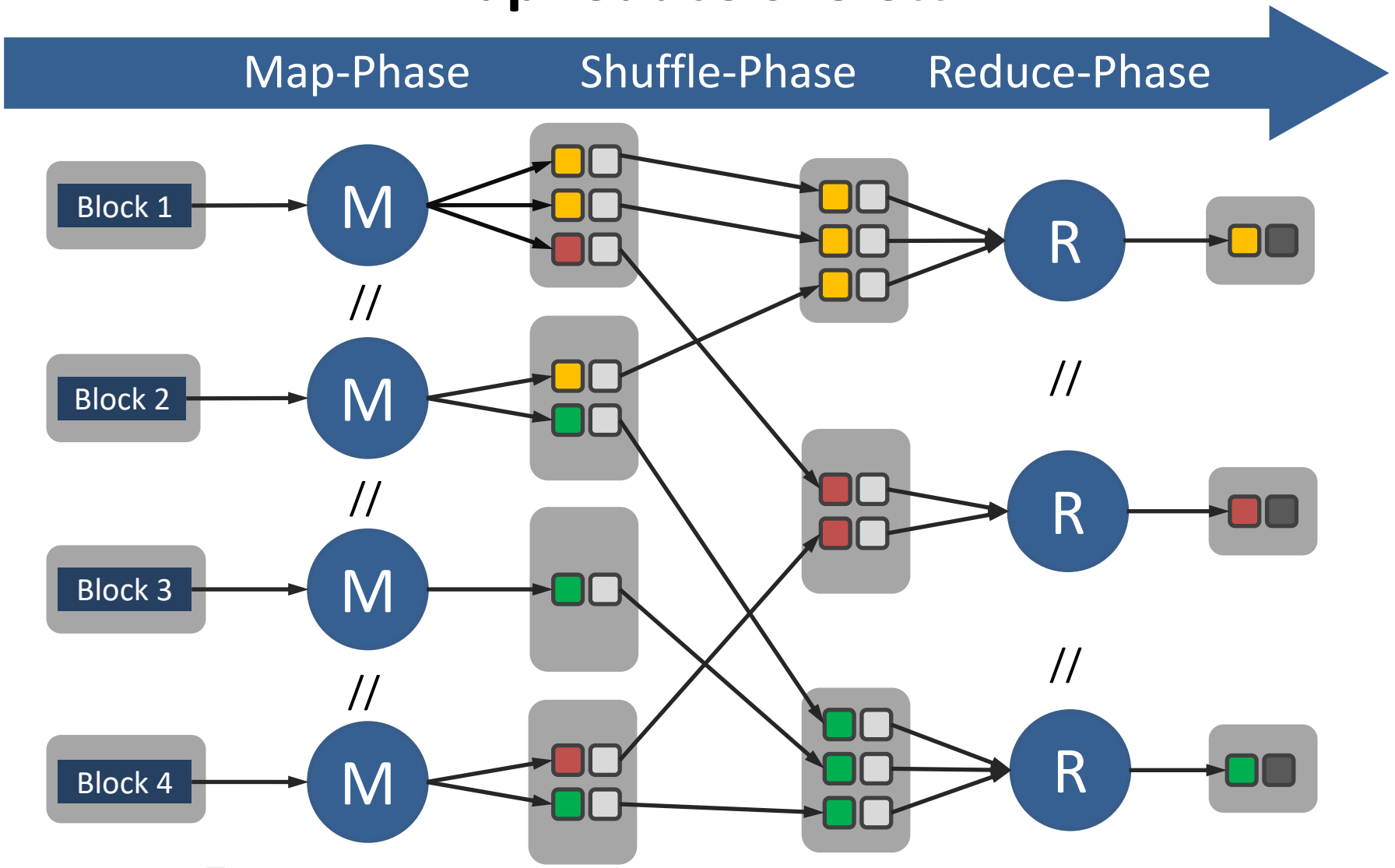
Aufgabe: Reduktion (z.B. Zusammenfassung oder Filterung) gleichartiger Informationen.



- **Eingabe:**
  - Liste von Key/Value Paaren mit identischem Key
- **Ausgabe:**
  - Key/Value Paar
  - Schlüssel gleich wie bei Eingabe
  - Wert ist Ergebnis der Reduktion

- **Beispiel „Grep“:** Reduktion ist hier die Konkatenation aller Zeilen, die das selbe Wort enthalten.
  - Ergebnis: Dokument, bei dem jede Zeile das durch den Schlüssel gegebene Wort enthält.

MapReduce Skelett

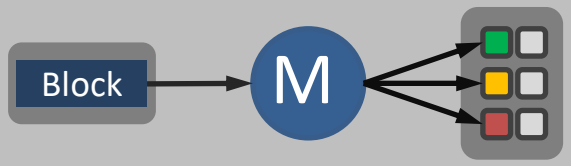


Beispiel: Word-Count mit MapReduce

Problemstellung: Bestimme für jedes Wort der Länge > 3
die Häufigkeit des Vorkommens in verschiedenen Dokumenten.

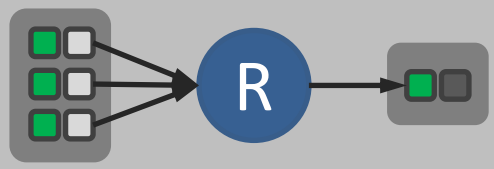
Map Pseudocode

```
map(String key, String value)
// key: document name
// value: document contents
for each word w in value:
  if (length(w) > 3)
    Emit-KV-Pair(w, “1”);
```

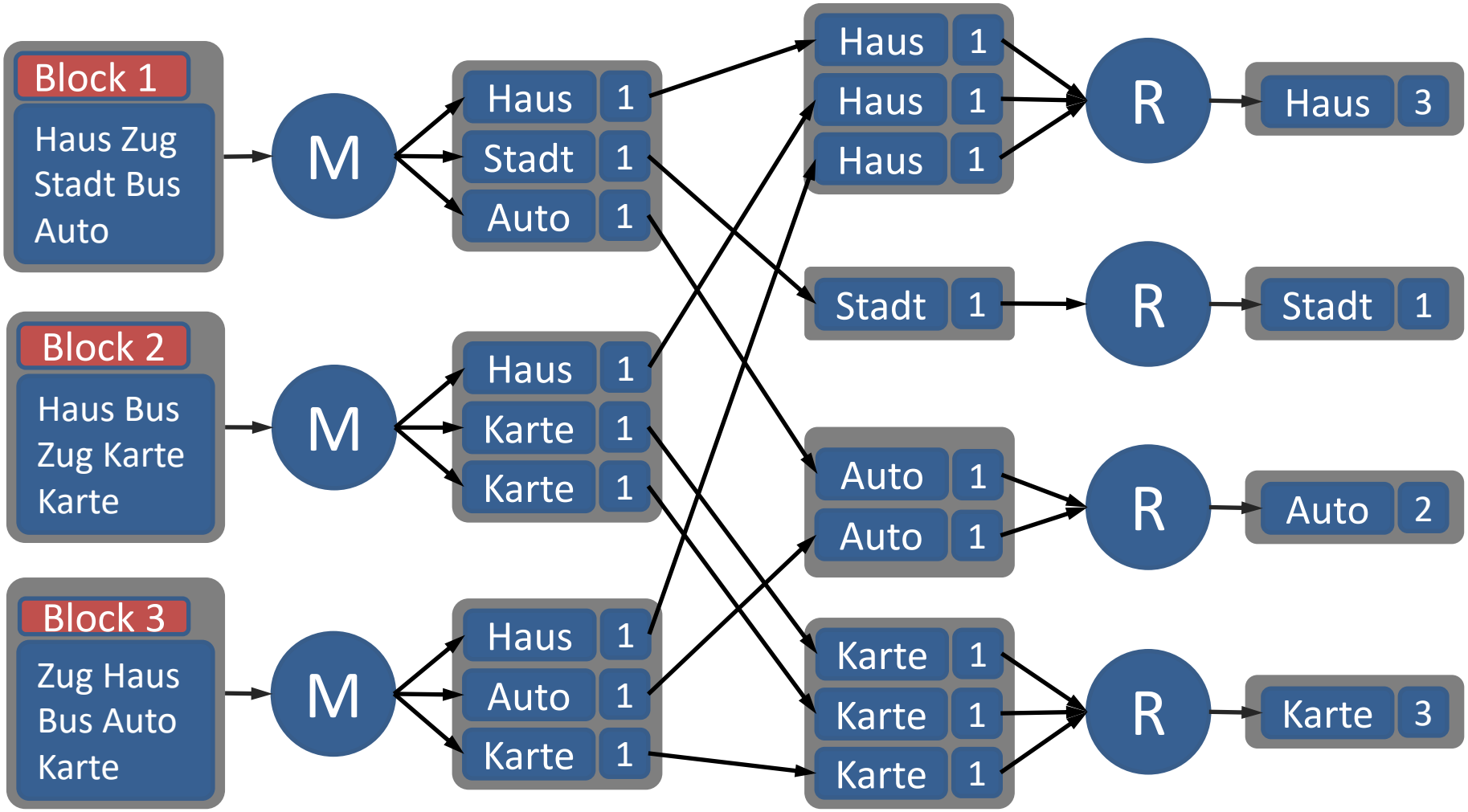


Reduce Pseudocode

```
reduce(String key, Iterator values)
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
  result += ParseInt(v);
Emit-KV-Pair(key, toString(result));
```



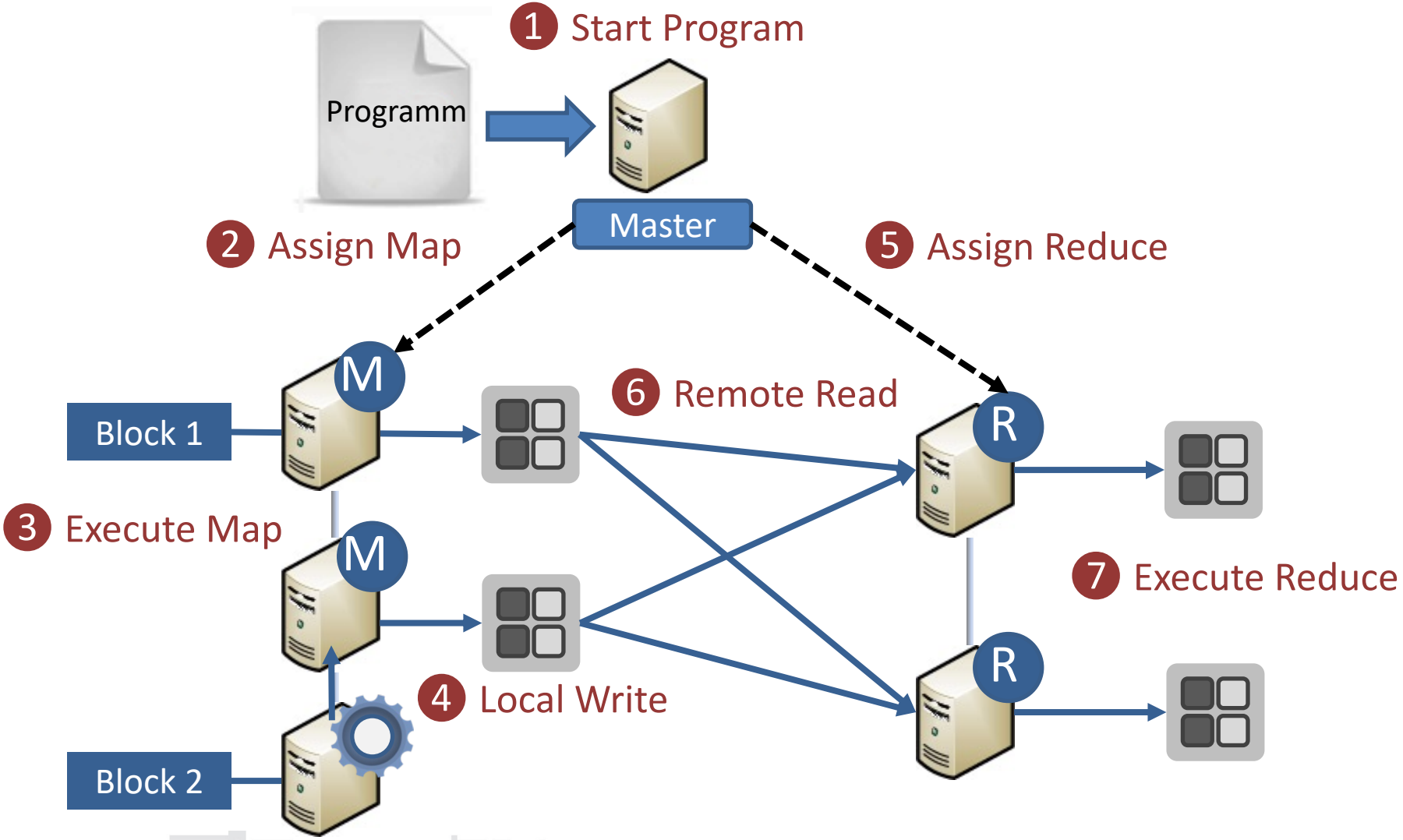
Beispiel: Word-Count mit MapReduce



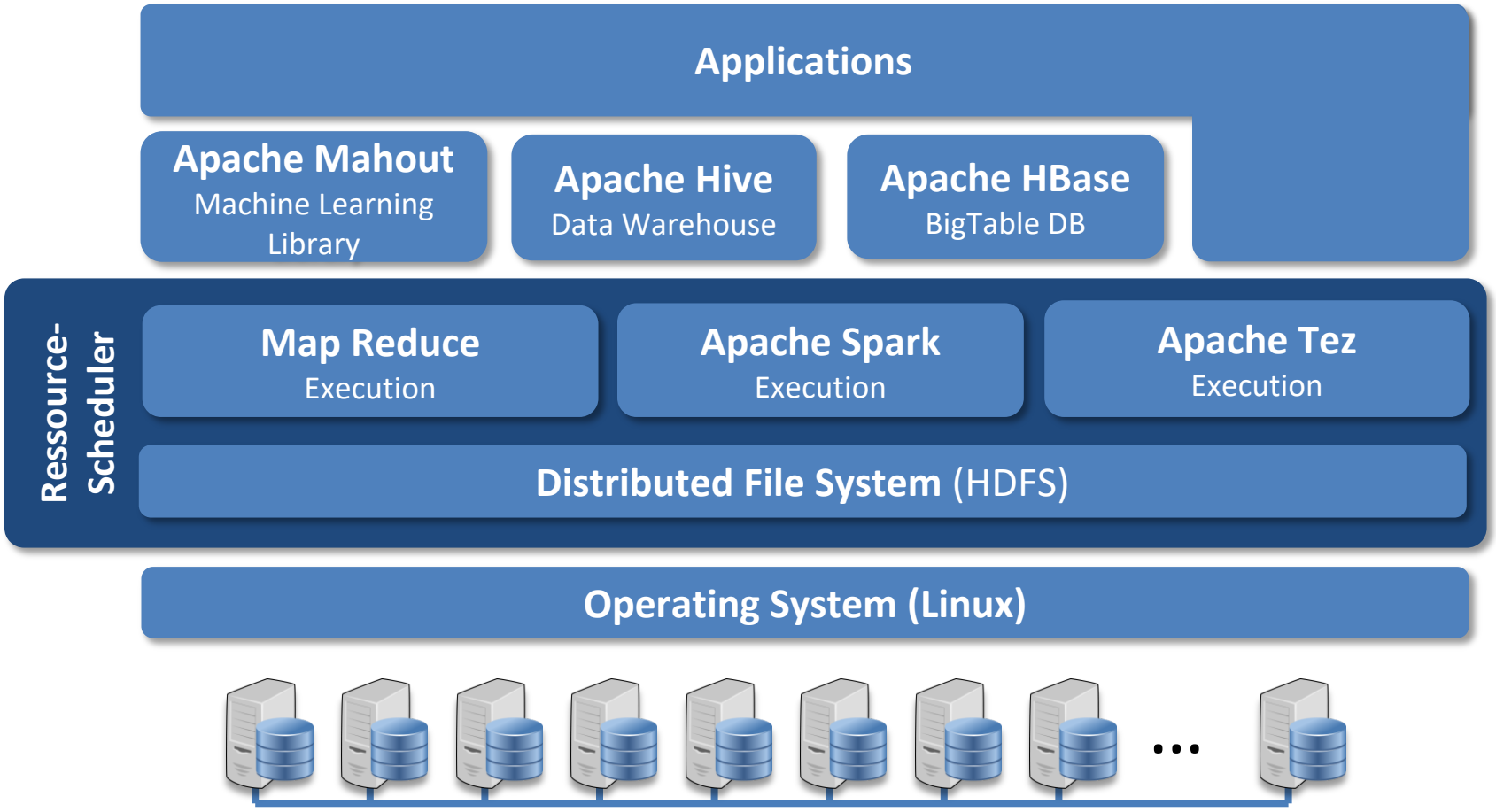
Ausführung eines MapReduce Programms

- MapReduce Programme werden im **Batch-Betrieb** ausgeführt.
  - Programm wird vom System in Warteschlange gestellt.
  - Ausführung startet, wenn genügend Rechner im Cluster zur Verfügung stehen.
  - Ende der Ausführung wird z.B. per E-Mail angezeigt.
- Ein Rechner des Clusters übernimmt die Rolle des **Masters**, alle anderen sind **Worker.**
  - Worker führen Map und Reduce Funktionen aus.
  - Master koordiniert die Berechnung:
    - Weist freien Workern Arbeit zu.
    - Führt Buch über den Speicherort aller Zwischenergebnisse.

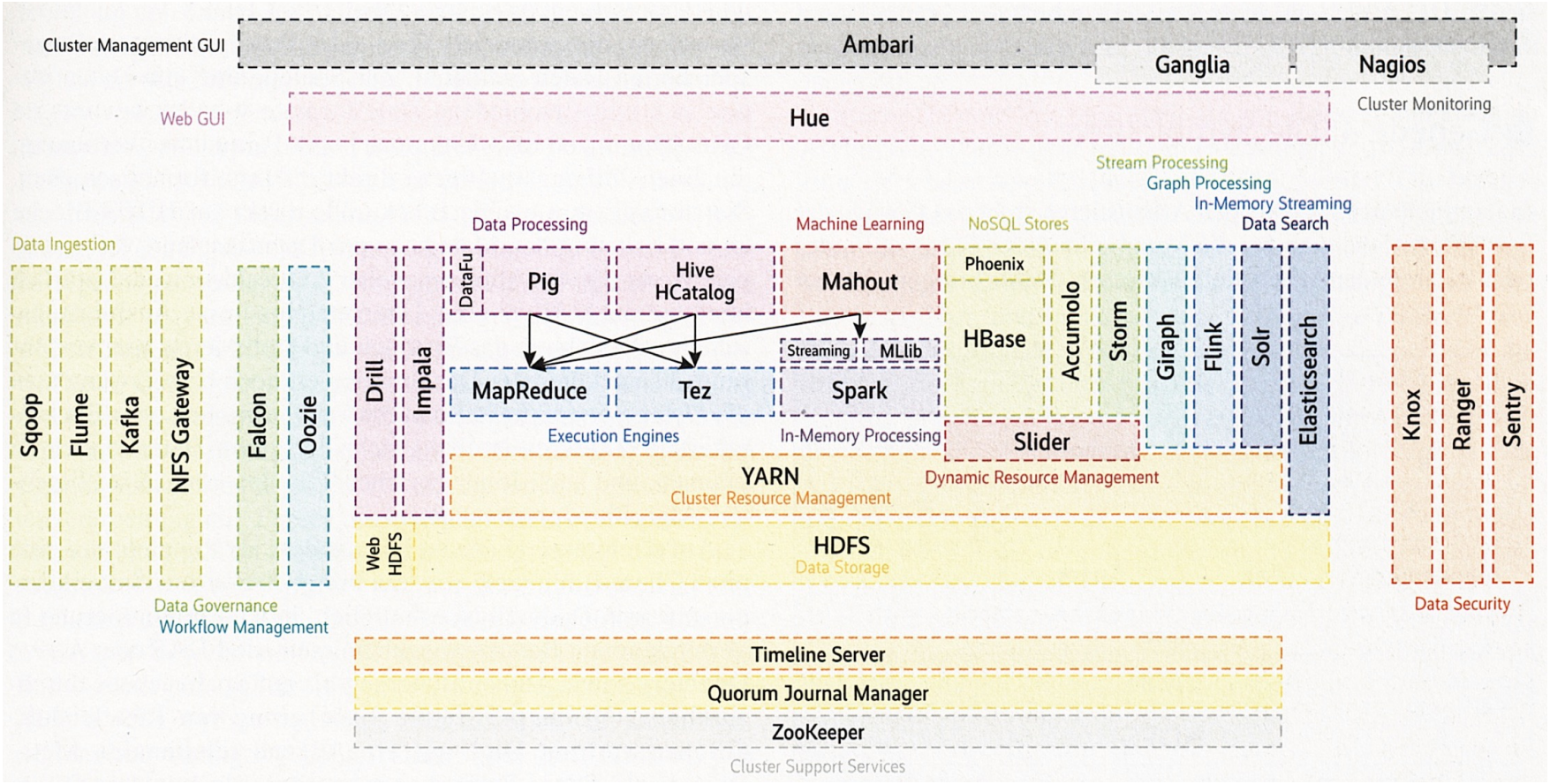
Ausführung eines MapReduce Programms



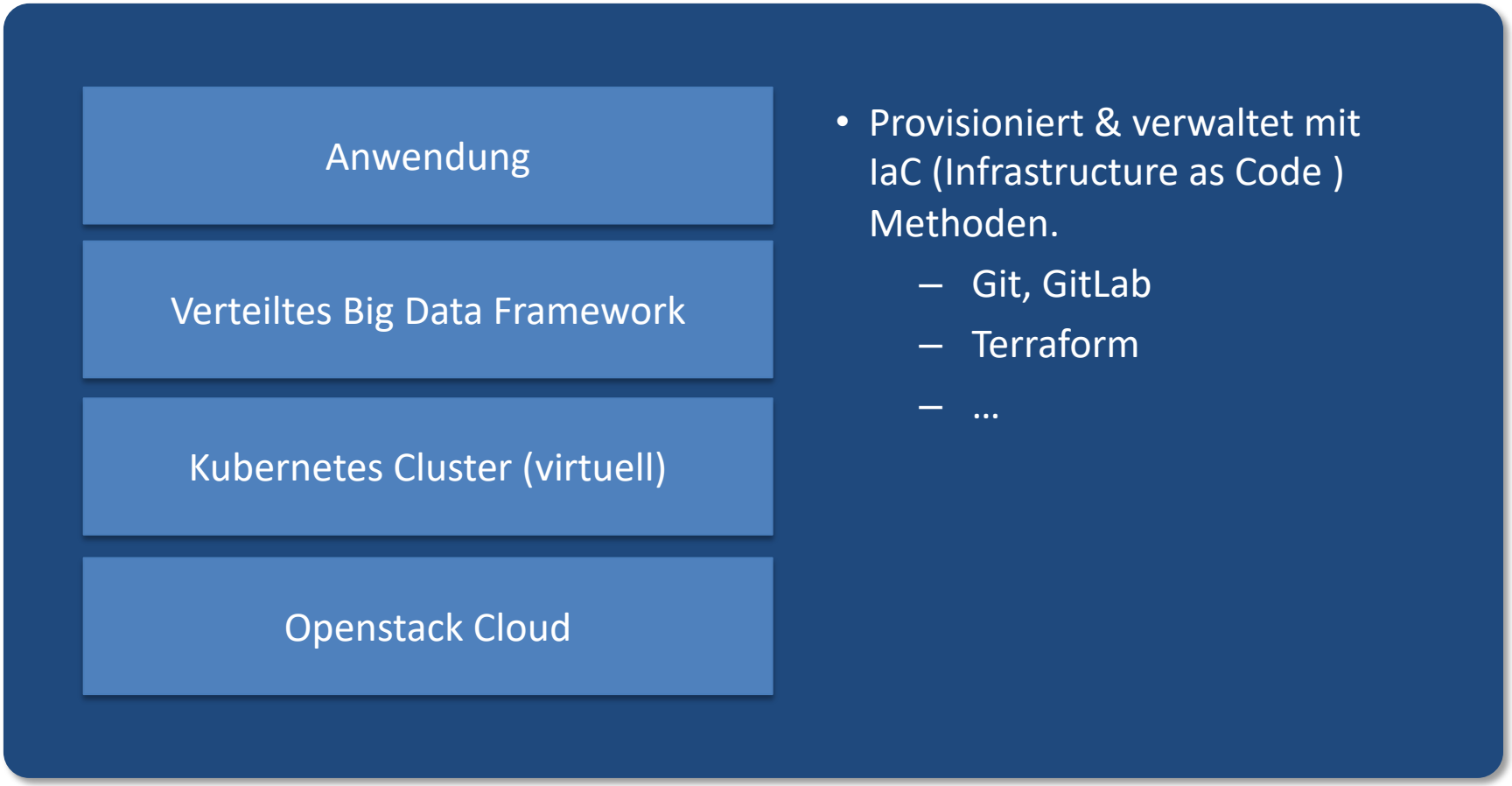
Hadoop Software Stack (Core Components)



Hadoop Software Stack



Unsere Ziel-Architektur



- Provisioniert & verwaltet mit IaC (Infrastructure as Code) Methoden.
  - Git, GitLab
  - Terraform
  - …