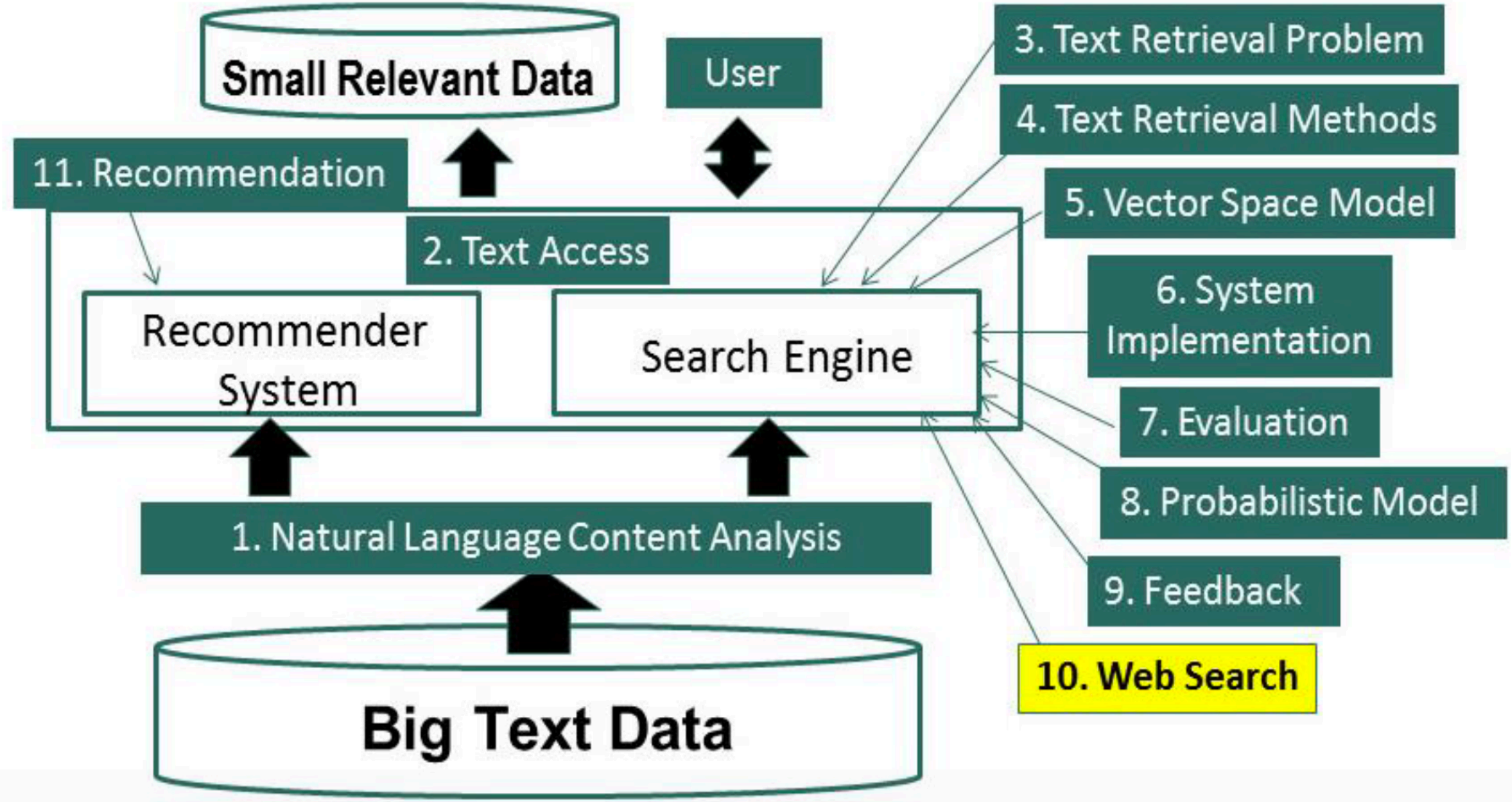
Information Retrieval & Text Mining

Web Search

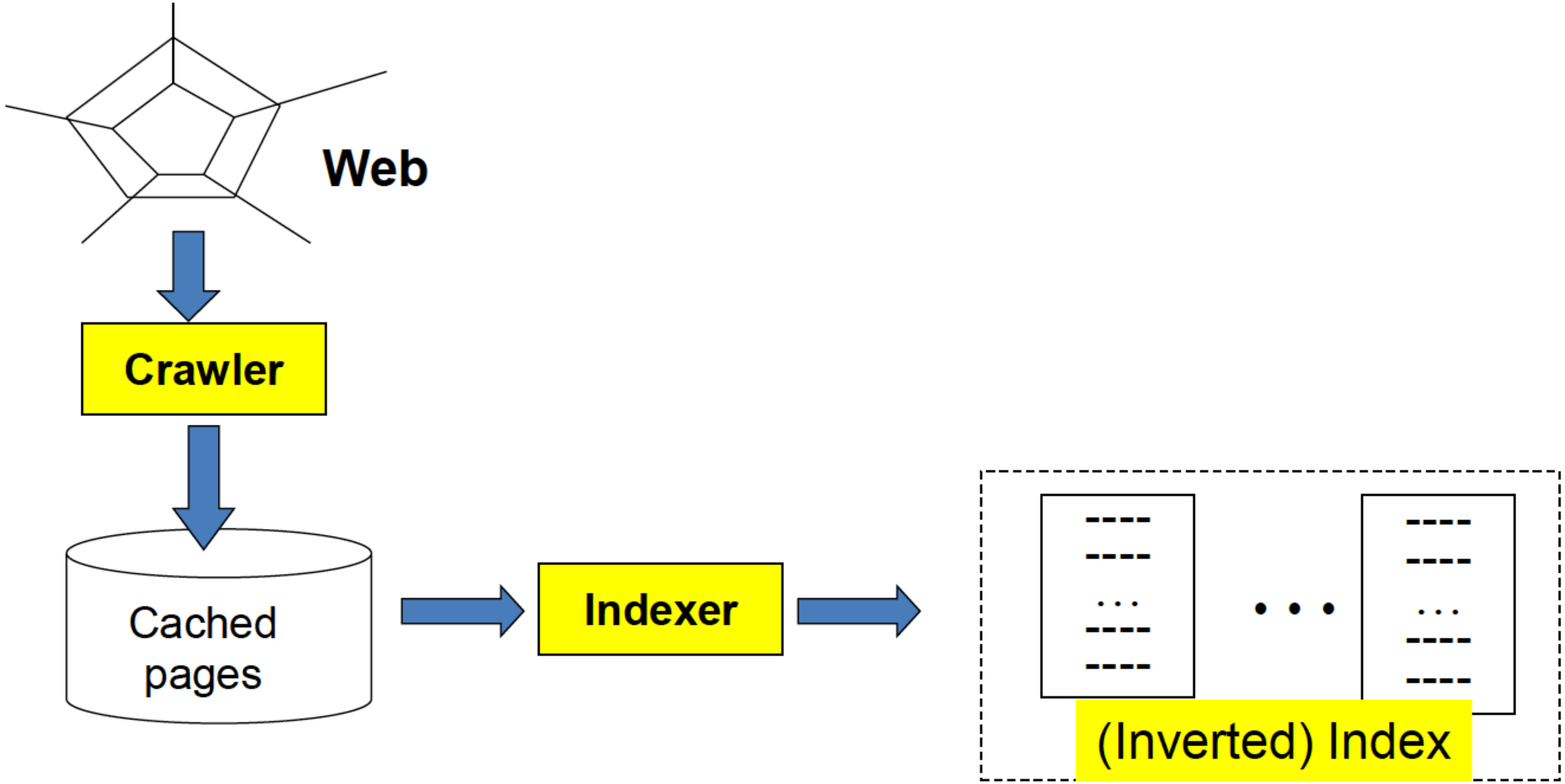
Web Index

Dr. Iqra Safder
Information Technology University

Course Schedule



Basic Search Engine Technologies



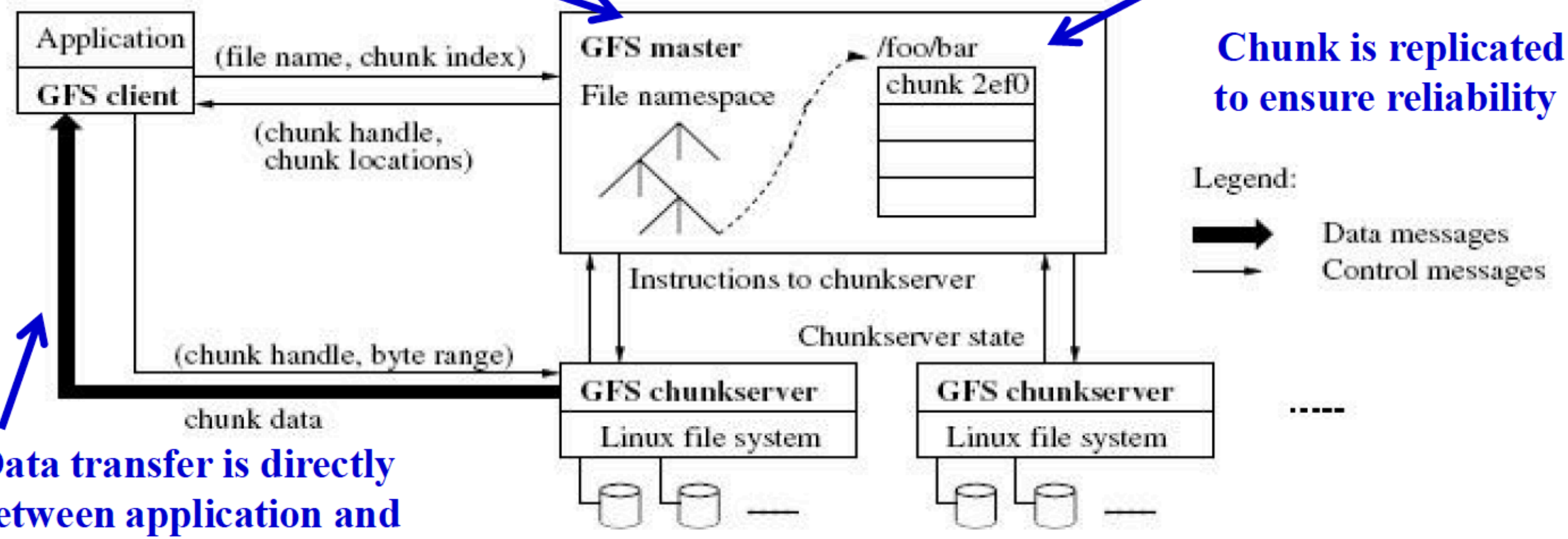
Overview of Web Indexing

- Standard IR techniques are the basis, but insufficient
  - Scalability
  - Efficiency
- Google’s contributions:
  - Google File System (GFS): distributed file system
  - MapReduce: Software framework for parallel computation
  - Hadoop: Open source implementation of MapReduce

GFS Architecture

Simple centralized management

Fixed chunk size (64 MB)



Chunk is replicated to ensure reliability

Data transfer is directly between application and chunk servers

Figure 1: GFS Architecture

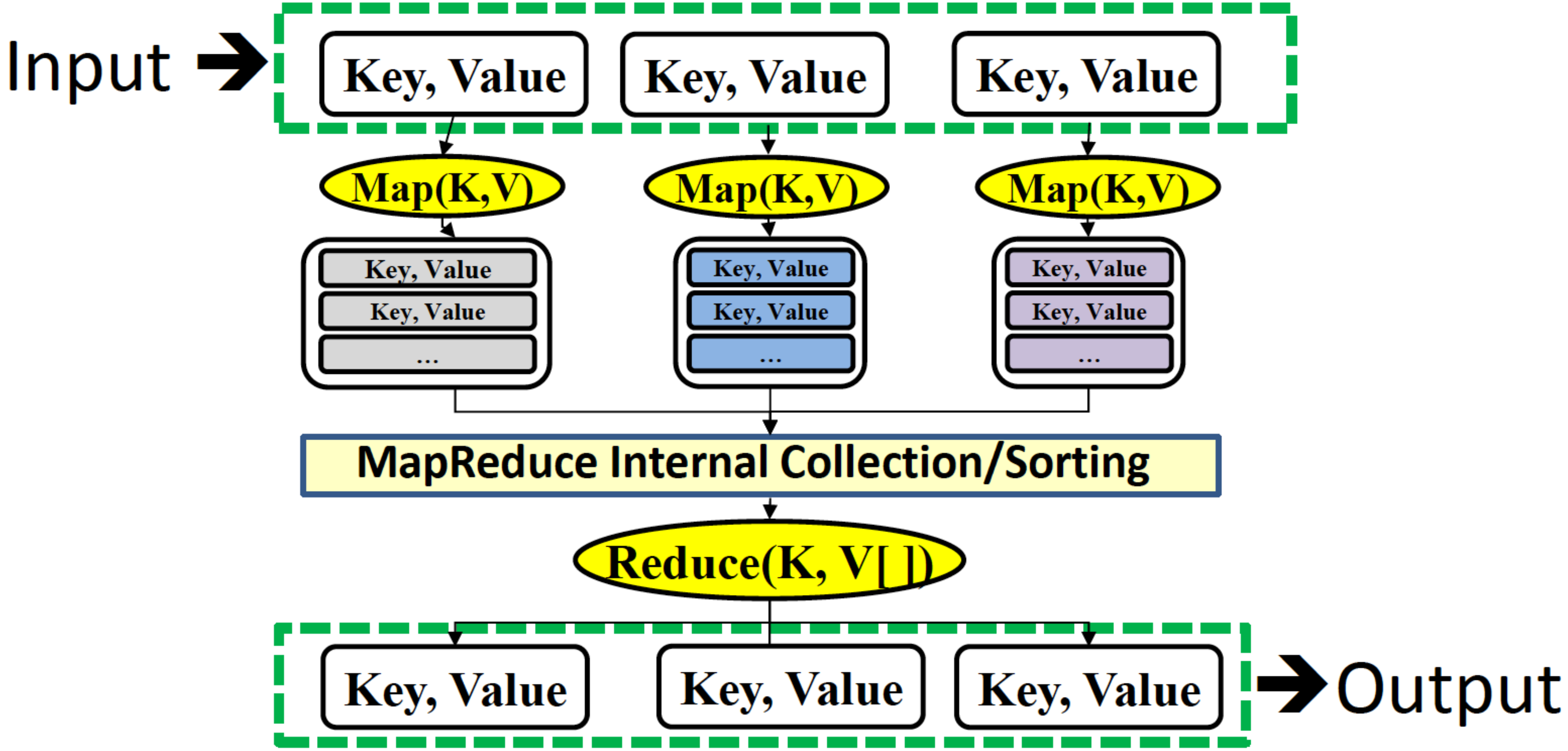
GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The google file system. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles (New York, NY, USA, 2003), ACM, pp. 29–43.

http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf

MapReduce: A Framework for Parallel Programming

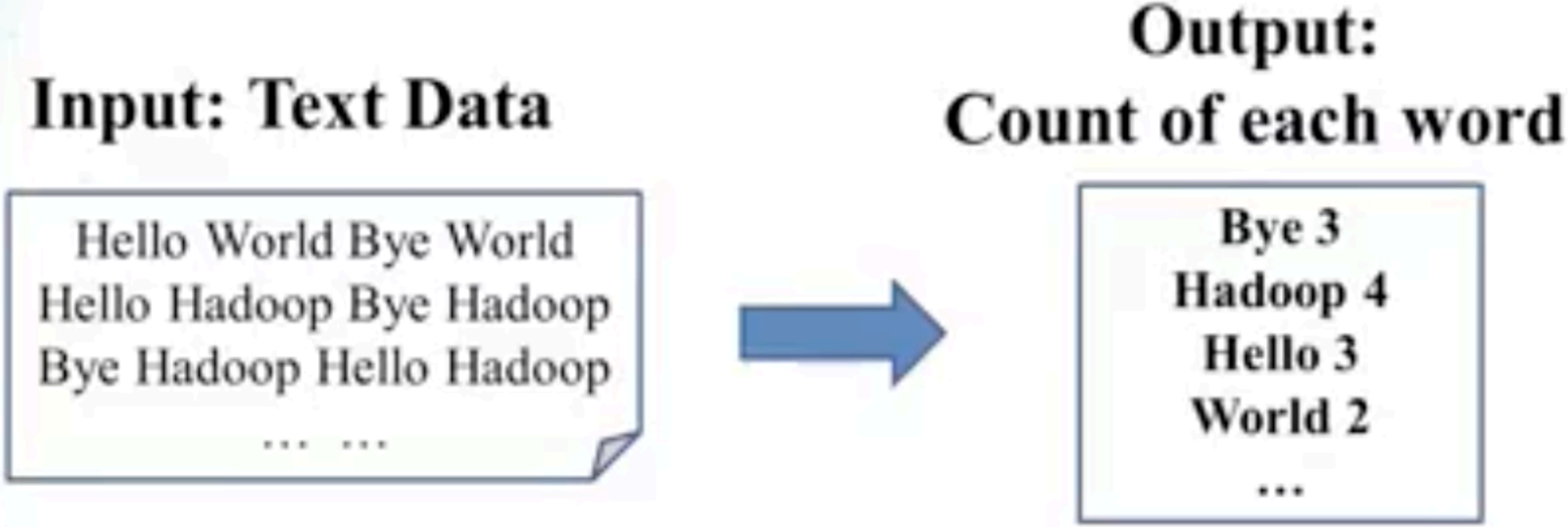
- Minimize effort of programmer for simple parallel processing tasks
- Features
  - Hide many low-level details (network, storage)
  - Built-in fault tolerance
  - Automatic load balancing

MapReduce: Computation Pipeline



Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Word Counting



Input: Text Data

Hello World Bye World
Hello Hadoop Bye Hadoop
Bye Hadoop Hello Hadoop
… …

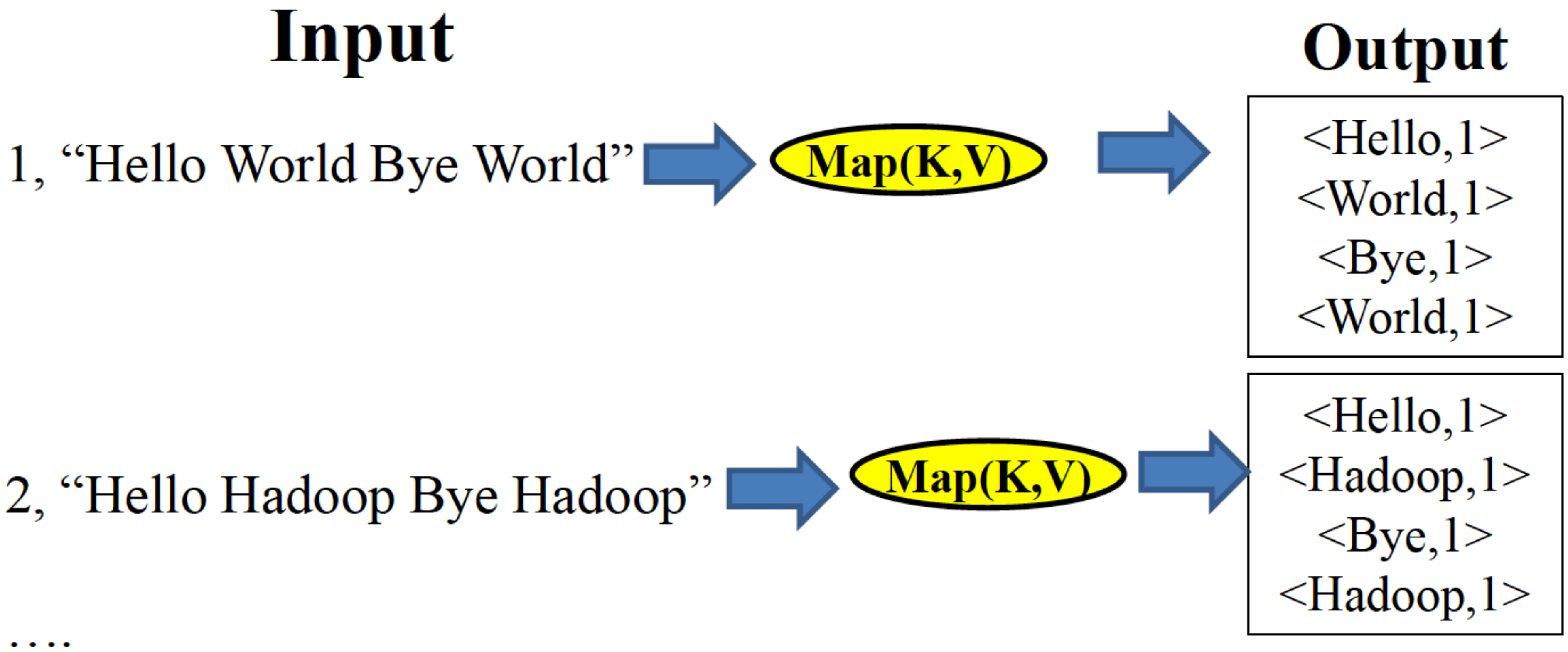
Output:
Count of each word

Bye 3
Hadoop 4
Hello 3
World 2
…

How can we do this within the MapReduce framework?

Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Word Counting: Map Function



Input

Output

1, “Hello World Byes World”

<Hello,1>
<World,1>
<Bye,1>
<World,1>

2, “Hello Hadoop Bye Hadoop”

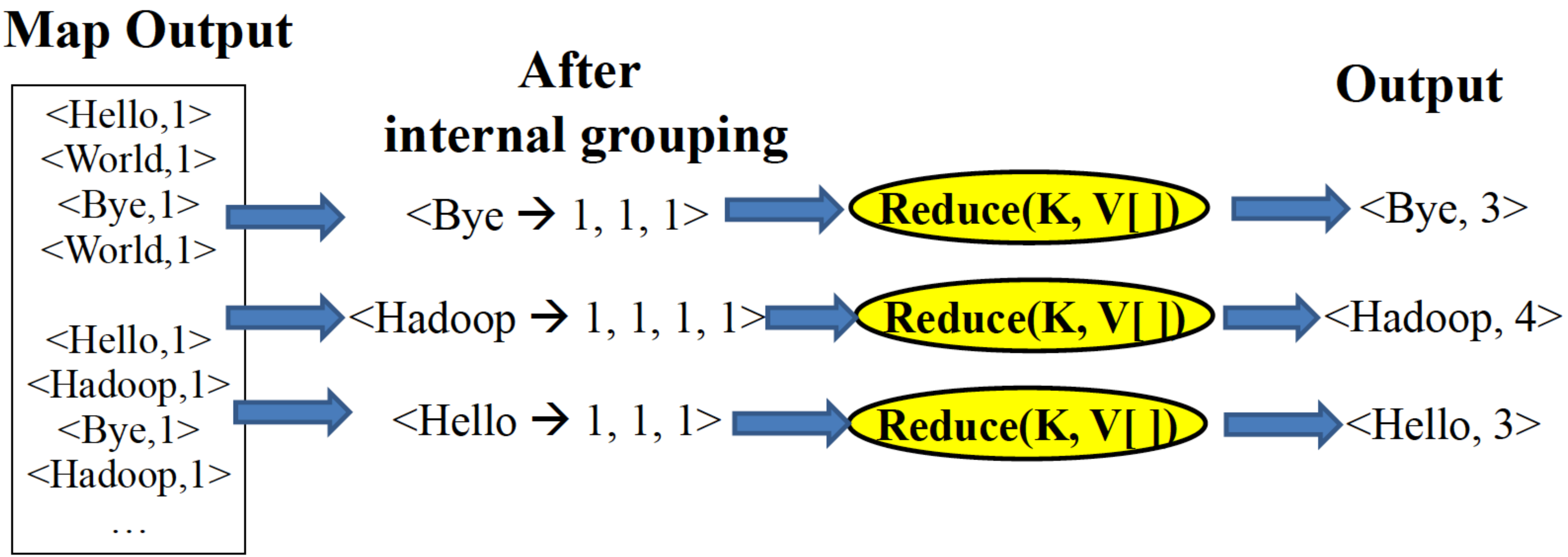
<Hello,1>
<Hadoop,1>
<Bye,1>
<Hadoop,1>

….

Map(K, V)
{ For each word w in V, Collect(w, 1);}

Slide adapted from Alexander Behm & Ajey Shah’s presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Word Counting: Reduce Function



Map Output

<Hello,1>
<World,1>
<Bye,1>
<World,1>

<Hello,1>
<Hadoop,1>
<Bye,1>
<Hadoop,1>
…

After internal grouping

<Bye → 1, 1, 1>
<Hadoop → 1, 1, 1, 1>
<Hello → 1, 1, 1>

Reduce(K, V[ ])
Reduce(K, V[ ])
Reduce(K, V[ ])

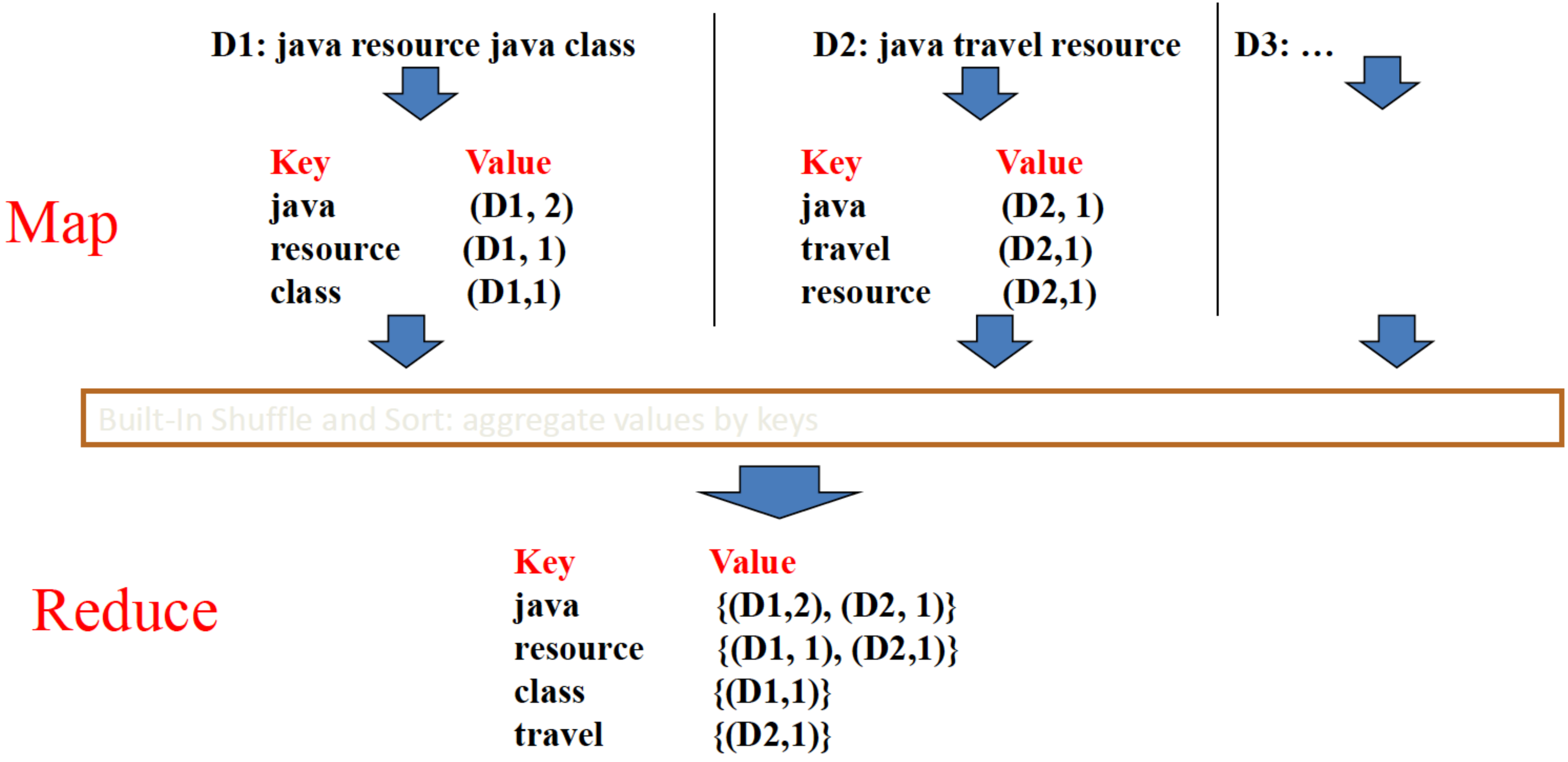
Output

<Bye, 3>
<Hadoop, 4>
<Hello, 3>

Reduce(K, V[ ])
{ Int count = 0; For each v in V, count += v; Collect(K, count); }

Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Inverted Indexing with MapReduce



Map

D1: java resource java class

Key	Value
java	(D1, 2)
resource	(D1, 1)
class	(D1,1)

D2: java travel resource

Key	Value
java	(D2, 1)
travel	(D2,1)
resource	(D2,1)

D3: …

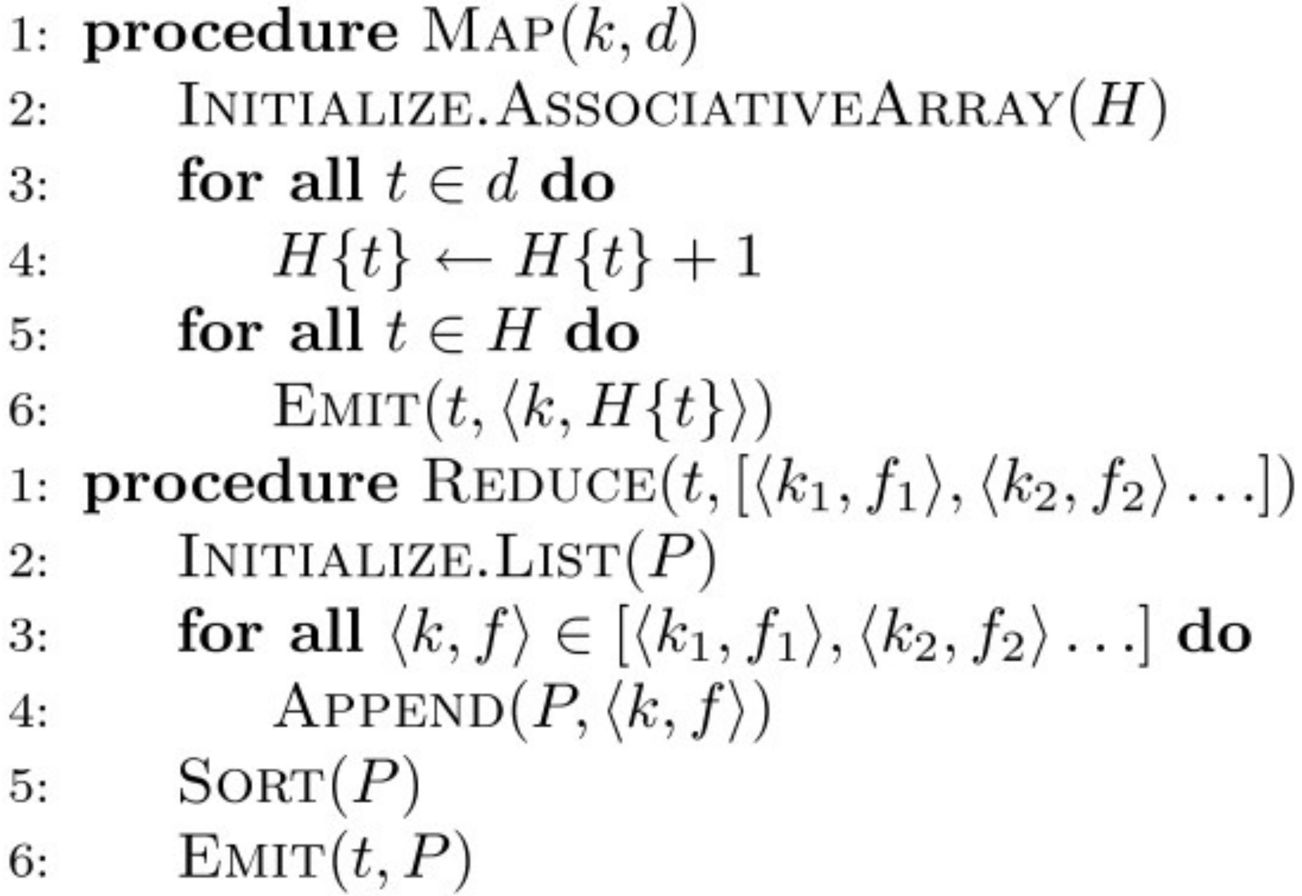
Built-In Shuffle and Sort: aggregate values by keys

Reduce

Key	Value
java	{(D1,2), (D2, 1)}
resource	{(D1, 1), (D2,1)}
class	{(D1,1)}
travel	{(D2,1)}
…	

Slide adapted from Jimmy Lin's presentation

Inverted Indexing – Pseudo code



```
procedure Map($k, d$)
    Initialize.AssociativeArray($H$)
    for all $t \in d$ do
        $H\{t\} \leftarrow H\{t\} + 1$
    for all $t \in H$ do
        Emit($t, \langle k, H\{t\} \rangle$)
procedure Reduce($t, [\langle k_1, f_1 \rangle, \langle k_2, f_2 \rangle \ldots]$)
    Initialize.List($P$)
    for all $\langle k, f \rangle \in [\langle k_1, f_1 \rangle, \langle k_2, f_2 \rangle \ldots]$ do
        Append($P, \langle k, f \rangle$)
    Sort($P$)
    Emit($t, P$)
```

Summary

- Web scale indexing requires
  - Storing the index on multiple machines (GFS)
  - Creating the index in parallel (MapReduce)
- Both GFS and MapReduce are general infrastructures