BigData

Relational database vs BigData

- Structured data vs semi-structured data, graph data
- Data from a single enterprise
- BigData requires high degree of parallelism (storage and processing)
- Sharding, key-value storage systems and documents stores

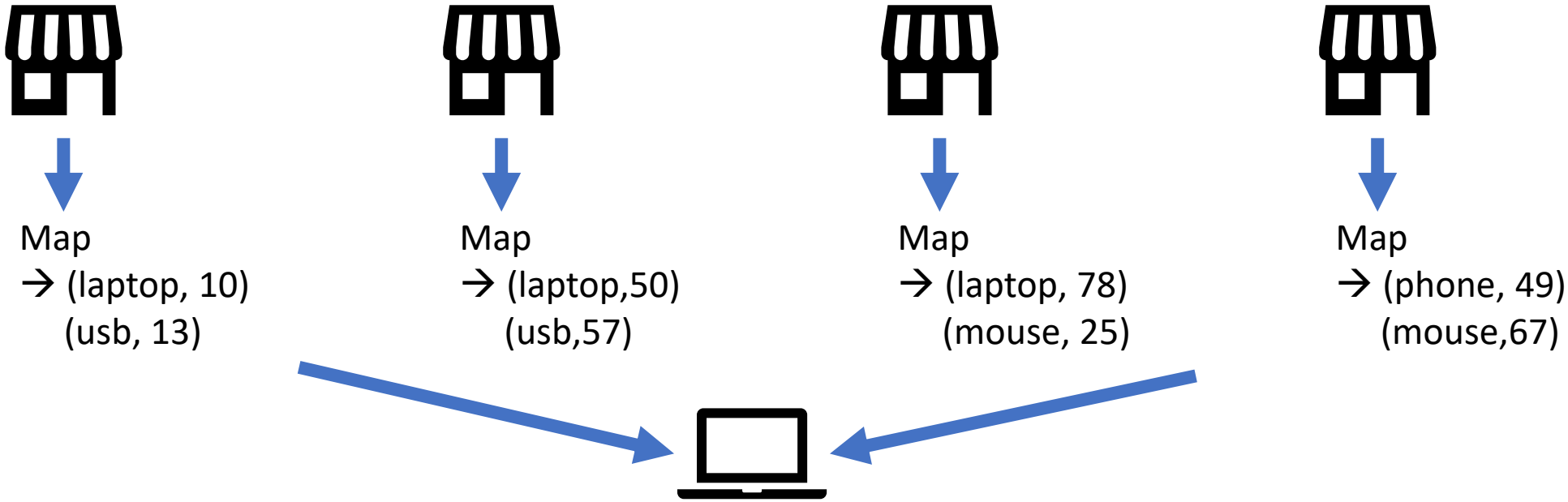
Map-reduce

Map Reduce algorithms

- Used in parallel processing.
- Fault tolerant.
- Programming paradigm (model) → framework,
  - examples Hadoop, Google
- Allows to process large volumes of data.
- Input in different formats.

Map Reduce example

- Counting product that clients entering local buy.
  - Input collected by multiple machines in parallel.
  - Data processed by multiple machines.

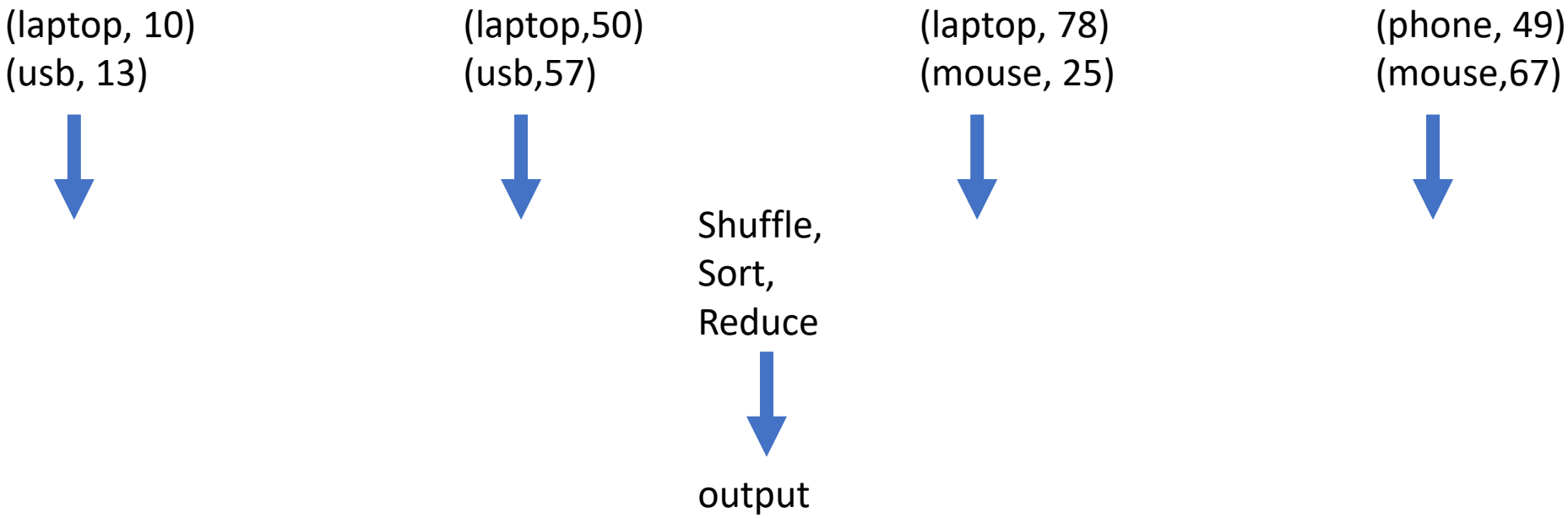


Map Reduce example

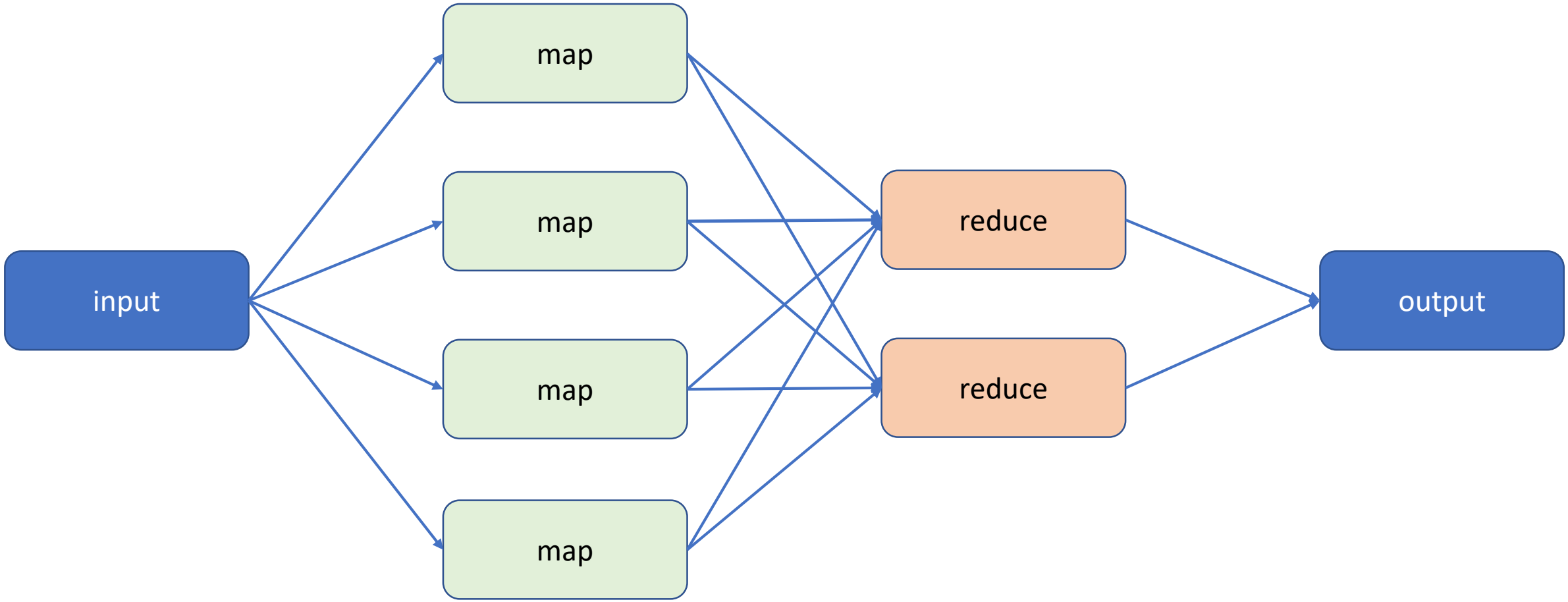
- MAP phase
  - map function provided by the developer will run on multiple nodes in parallel, process input data.

Map Reduce example

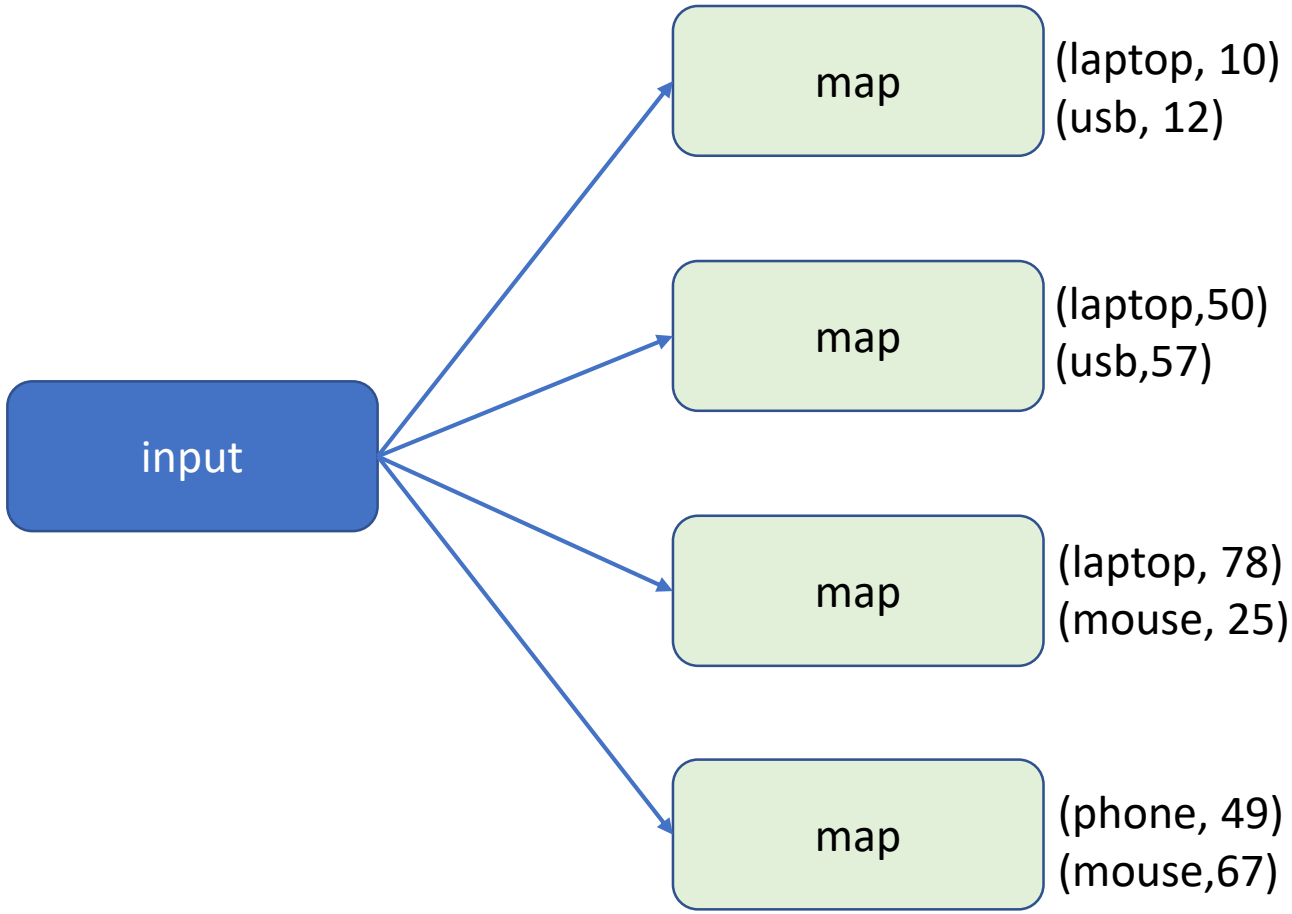
- REDUCE phase
  - reduce function provided by the developer, reduce the output produced by map functions, aggregate.
  - a call for a reduce function is for a single reduced key.



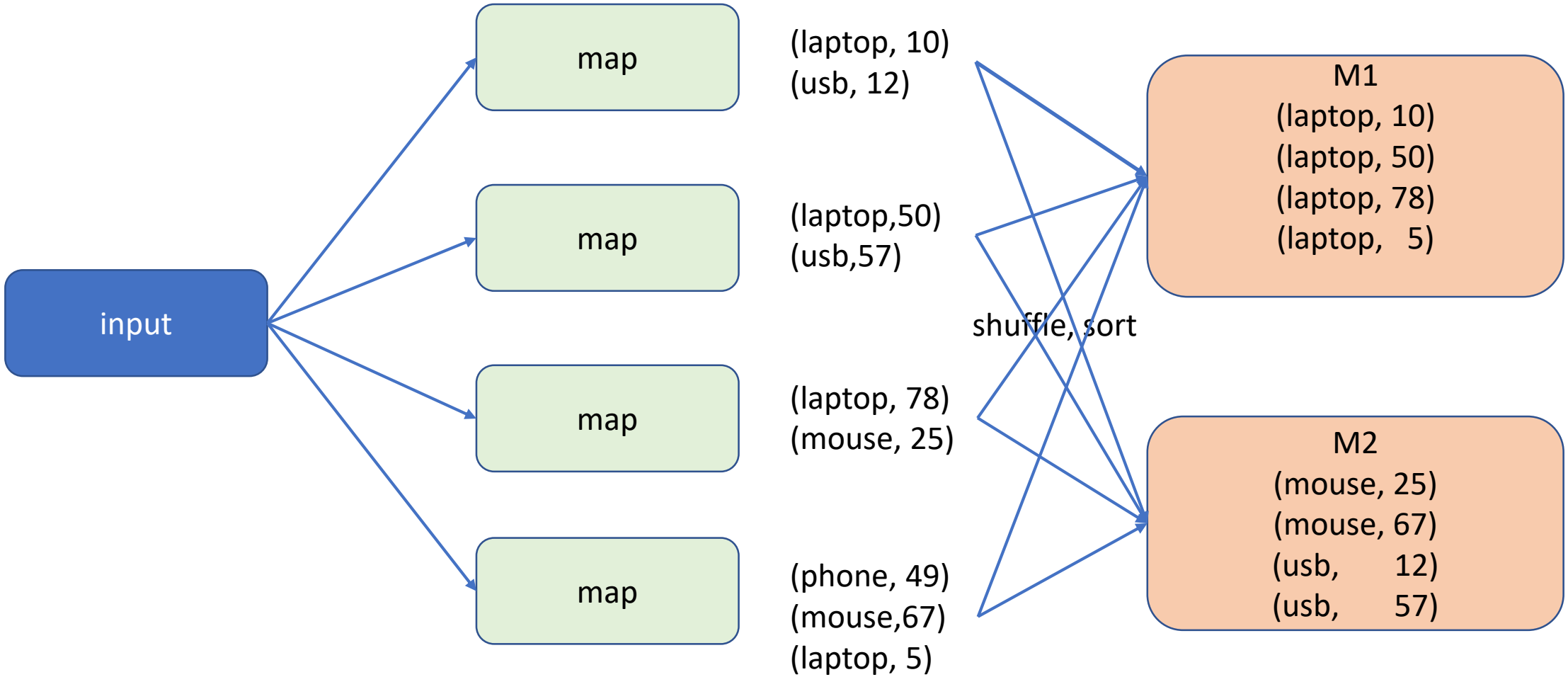
Map reduce



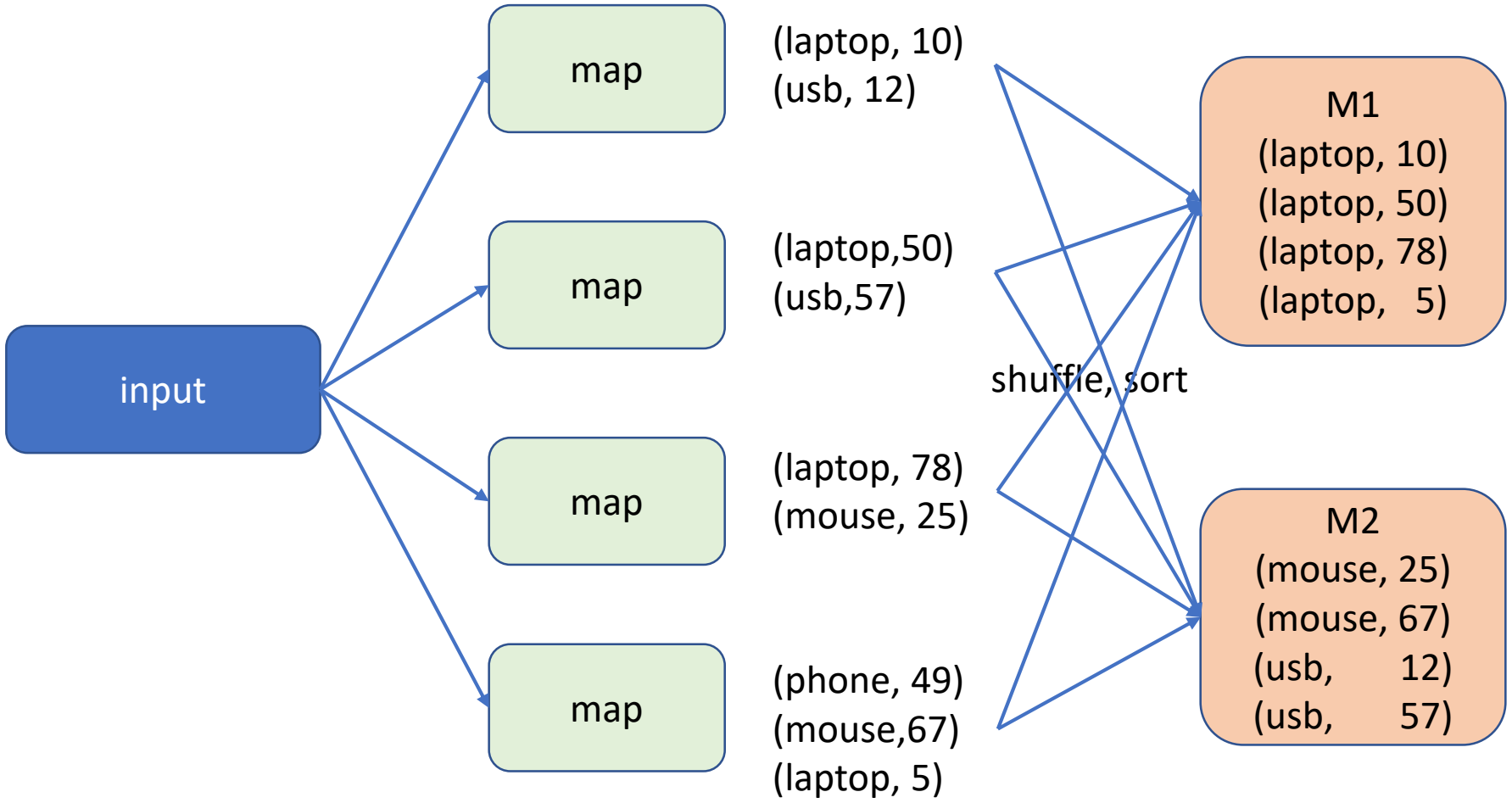
Map reduce


input
map
map
map
map
(laptop, 10)
(usb, 12)
(laptop,50)
(usb,57)
(laptop, 78)
(mouse, 25)
(phone, 49)
(mouse,67)

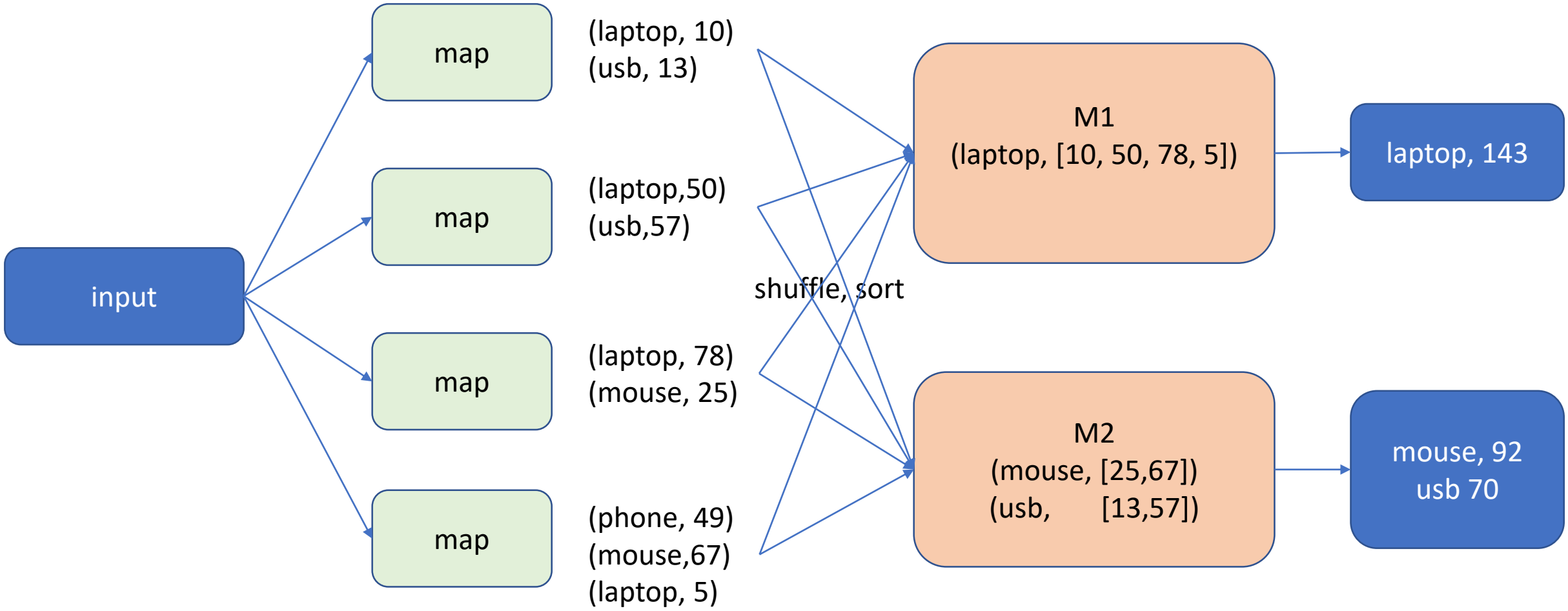
Map reduce


input
map
map
map
map
(laptop, 10)
(usb, 12)
(laptop,50)
(usb,57)
(laptop, 78)
(mouse, 25)
(phone, 49)
(mouse,67)
(laptop, 5)
shuffle, sort
M1
(laptop, 10)
(laptop, 50)
(laptop, 78)
(laptop, 5)
M2
(mouse, 25)
(mouse, 67)
(usb, 12)
(usb, 57)

Map reduce


input
map
map
map
map
(laptop, 10)
(usb, 12)
(laptop,50)
(usb,57)
(laptop, 78)
(mouse, 25)
(phone, 49)
(mouse,67)
(laptop, 5)
shuffle, sort
M1
(laptop, 10)
(laptop, 50)
(laptop, 78)
(laptop, 5)
M2
(mouse, 25)
(mouse, 67)
(usb, 12)
(usb, 57)

Map reduce


input
map
map
map
map
(laptop, 10)
(usb, 13)
(laptop,50)
(usb,57)
(laptop, 78)
(mouse, 25)
(phone, 49)
(mouse,67)
(laptop, 5)
shuffle, sort
M1
(laptop, [10, 50, 78, 5])
M2
(mouse, [25,67])
(usb, [13,57])
laptop, 143
mouse, 92
usb 70

MapReduce Hadoop

- Open source from Apache. https://hadoop.apache.org/

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

- Written in Java, also provide implementations in C++/Python.

- Components
  - MapReduce
  - Hadoop Distributed file system HDFS
    - Each file is stored as a sequence of blocks
    - Fault tolerant: Each block is replicated

- Master-slave architecture: NameNode (master), DataNodes (slaves).

MapReduce Hadoop

- map, reduce and combine function.
- combine perform partial aggregation before maps sends the result to reduce.
- combine -- reduce the amount of data sent over the network.
- combine -- Decrease the shuffling cost
- A MapReduce job can be configured to process map function phase only

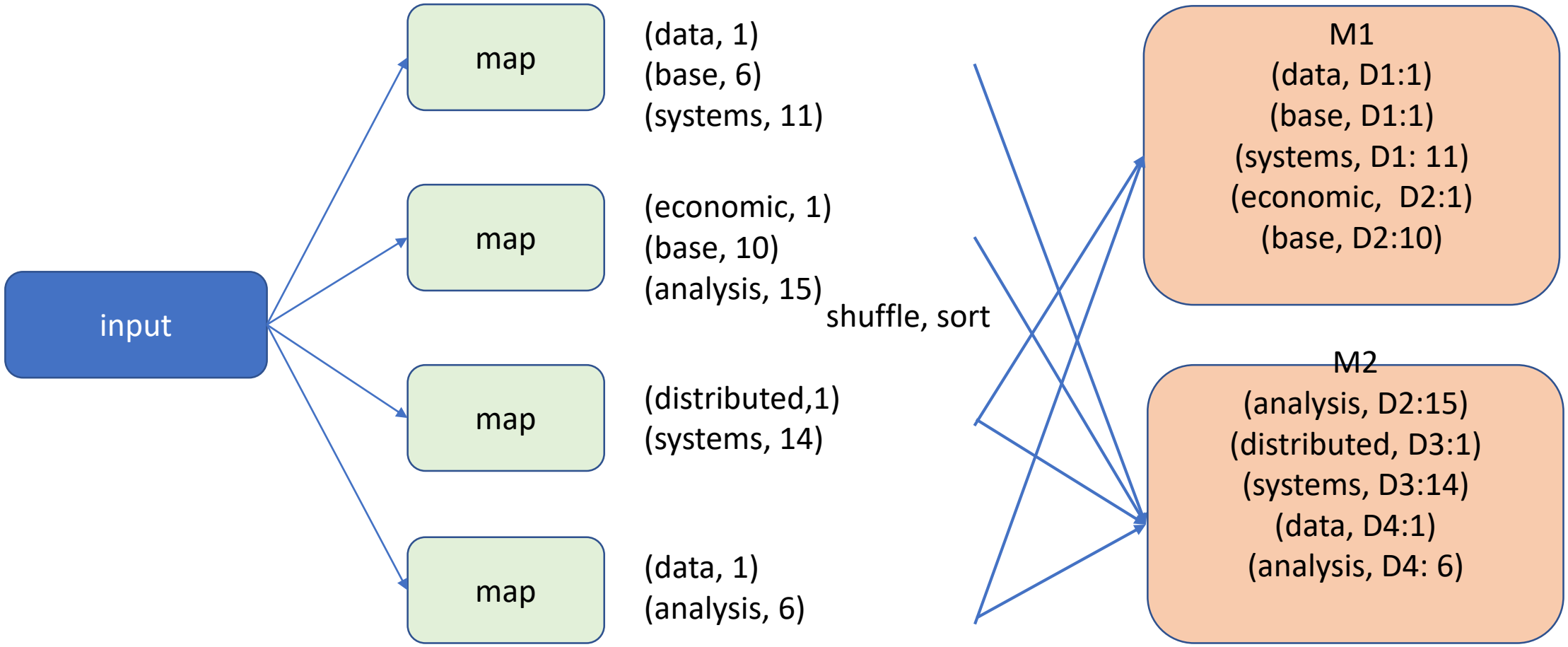
Inverted index

MapReduce Inverted index

- Web search engines (including Google).
- Maps content to location.
- Fast text search.
- PageRank-ing

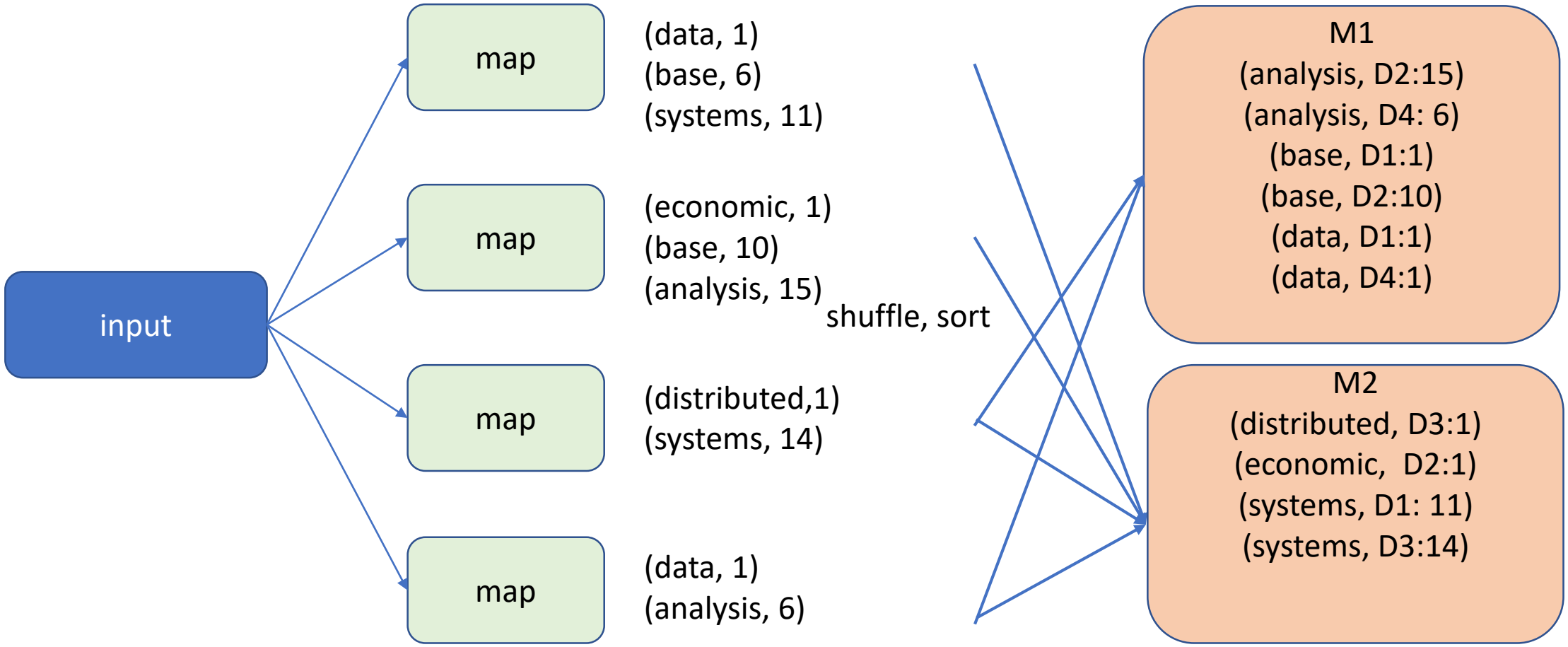
Inverted index

D1: data base systems,
D2: economic base analysis
D3: distributed systems
D4: data analysis



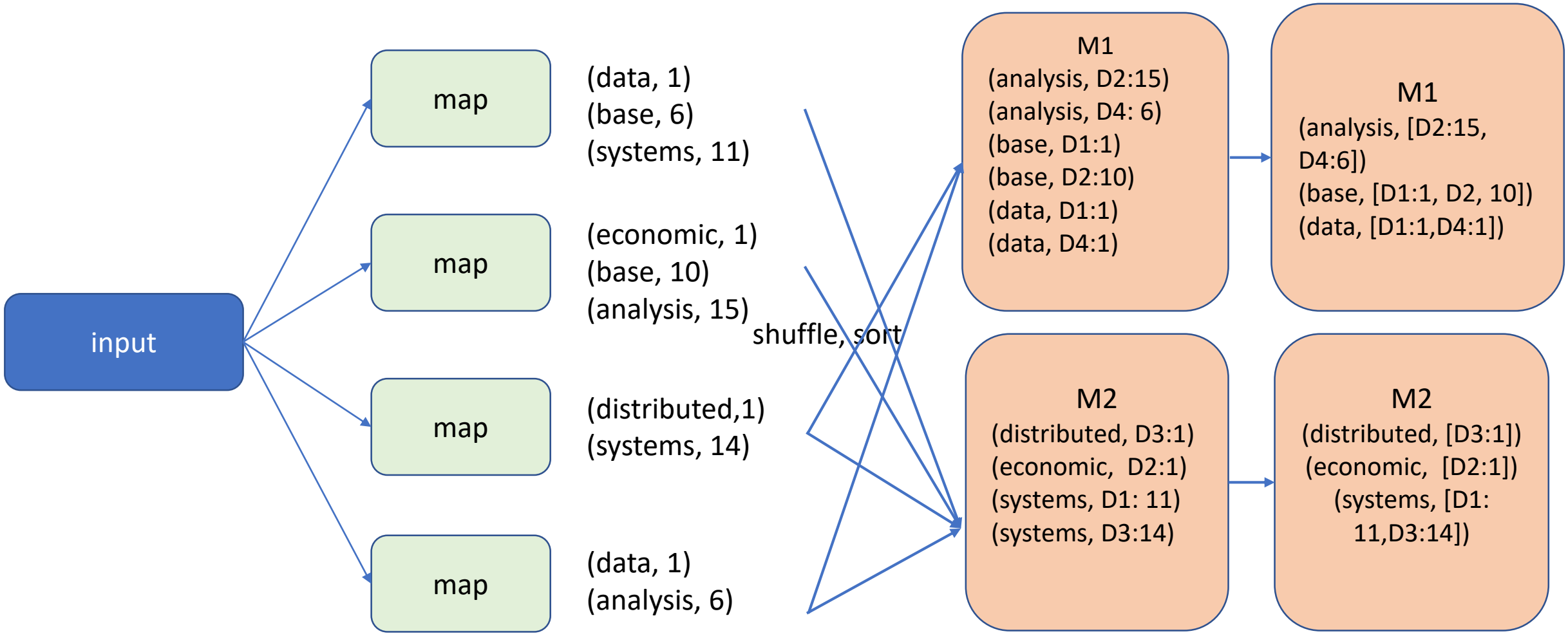
Inverted index

D1: data base systems,
D2: economic base analysis
D3: distributed systems
D4: data analysis



Inverted index

D1: data base systems,
D2: economic base analysis
D3: distributed systems
D4: data analysis



Sql operators

MapReduce: Sql operators

- Selection
- Group by
- Join

EMPLOYEES		
emp_id	name	dep_id
100	Steven King	90
102	Lex De Hann	90
108	Nancy Greenberg	100
116	Shelli Baida	30
117	Sigal Tobias	30

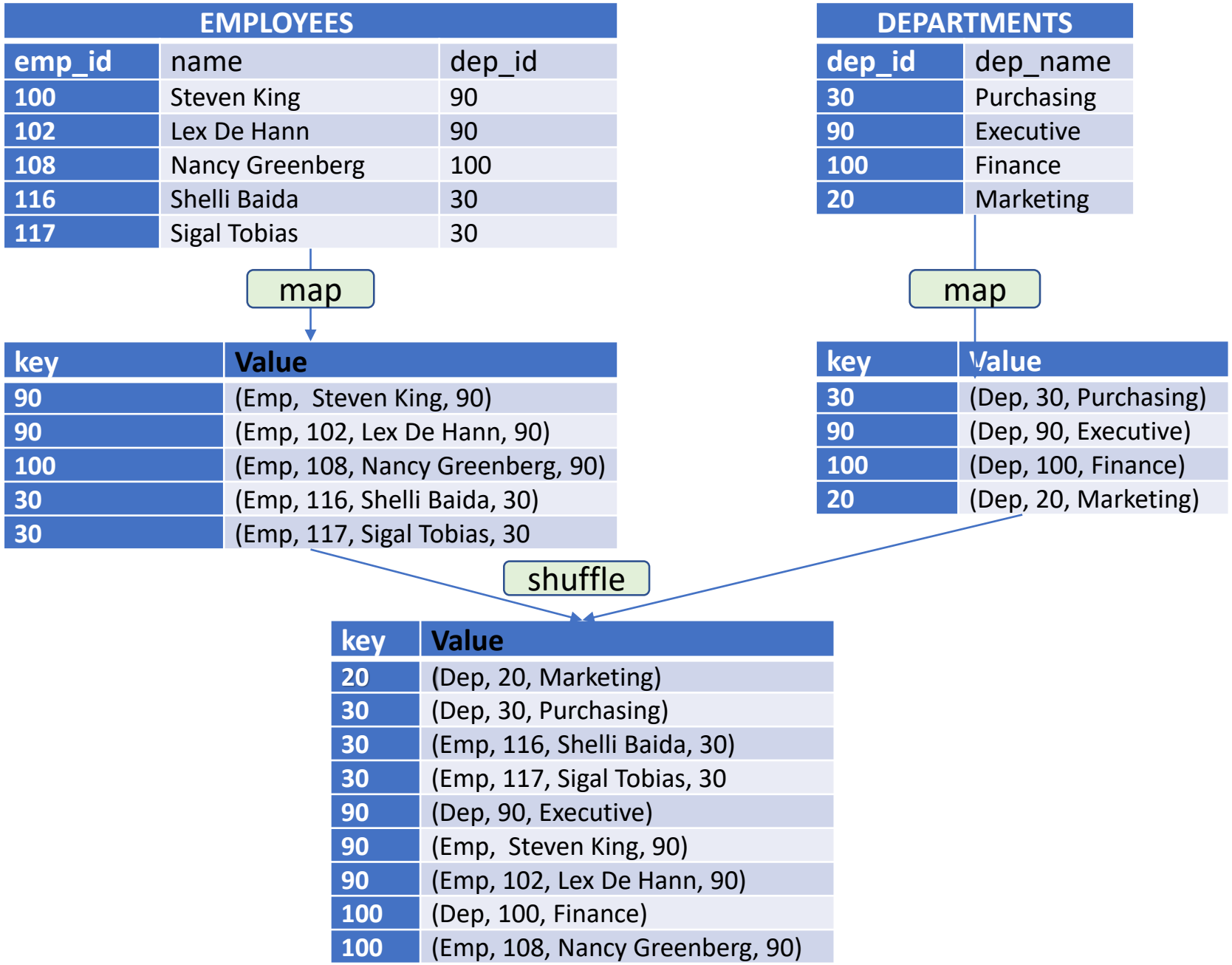
map

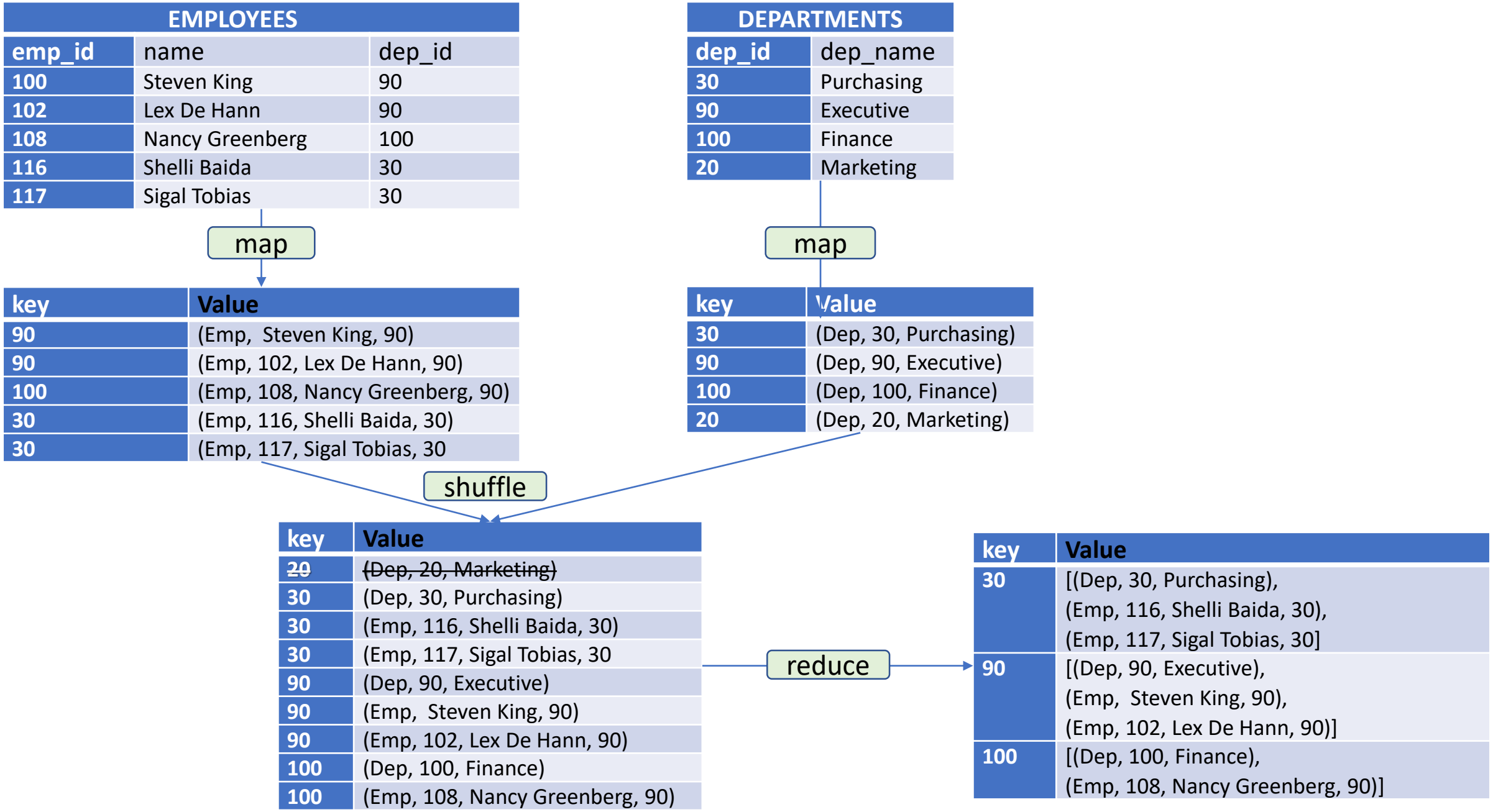
key	Value
90	(Emp, Steven King, 90)
90	(Emp, 102, Lex De Hann, 90)
100	(Emp, 108, Nancy Greenberg, 90)
30	(Emp, 116, Shelli Baida, 30)
30	(Emp, 117, Sigal Tobias, 30

DEPARTMENTS	
dep_id	dep_name
30	Purchasing
90	Executive
100	Finance
20	Marketing

map

key	Value
30	(Dep, 30, Purchasing)
90	(Dep, 90, Executive)
100	(Dep, 100, Finance)
20	(Dep, 20, Marketing)


EMPLOYEES
emp_id
name
dep_id
100
Steven King
90
102
Lex De Hann
90
108
Nancy Greenberg
100
116
Shelli Baida
30
117
Sigal Tobias
30
map
key
Value
90
(Emp, Steven King, 90)
90
(Emp, 102, Lex De Hann, 90)
100
(Emp, 108, Nancy Greenberg, 90)
30
(Emp, 116, Shelli Baida, 30)
30
(Emp, 117, Sigal Tobias, 30
DEPARTMENTS
dep_id
dep_name
30
Purchasing
90
Executive
100
Finance
20
Marketing
map
key
Value
30
(Dep, 30, Purchasing)
90
(Dep, 90, Executive)
100
(Dep, 100, Finance)
20
(Dep, 20, Marketing)
shuffle
key
Value
20
(Dep, 20, Marketing)
30
(Dep, 30, Purchasing)
30
(Emp, 116, Shelli Baida, 30)
30
(Emp, 117, Sigal Tobias, 30
90
(Dep, 90, Executive)
90
(Emp, Steven King, 90)
90
(Emp, 102, Lex De Hann, 90)
100
(Dep, 100, Finance)
100
(Emp, 108, Nancy Greenberg, 90)


EMPLOYEES
emp_id
name
dep_id
100
Steven King
90
102
Lex De Hann
90
108
Nancy Greenberg
100
116
Shelli Baida
30
117
Sigal Tobias
30
DEPARTMENTS
dep_id
dep_name
30
Purchasing
90
Executive
100
Finance
20
Marketing
map
map
key
Value
90
(Emp, Steven King, 90)
90
(Emp, 102, Lex De Hann, 90)
100
(Emp, 108, Nancy Greenberg, 90)
30
(Emp, 116, Shelli Baida, 30)
30
(Emp, 117, Sigal Tobias, 30
key
Value
30
(Dep, 30, Purchasing)
90
(Dep, 90, Executive)
100
(Dep, 100, Finance)
20
(Dep, 20, Marketing)
shuffle
key
Value
20
(Dep, 20, Marketing)
30
(Dep, 30, Purchasing)
30
(Emp, 116, Shelli Baida, 30)
30
(Emp, 117, Sigal Tobias, 30
90
(Dep, 90, Executive)
90
(Emp, Steven King, 90)
90
(Emp, 102, Lex De Hann, 90)
100
(Dep, 100, Finance)
100
(Emp, 108, Nancy Greenberg, 90)
reduce
key
Value
30
[(Dep, 30, Purchasing),
(Emp, 116, Shelli Baida, 30),
(Emp, 117, Sigal Tobias, 30]
90
[(Dep, 90, Executive),
(Emp, Steven King, 90),
(Emp, 102, Lex De Hann, 90)]
100
[(Dep, 100, Finance),
(Emp, 108, Nancy Greenberg, 90)]