An introduction to MapReduce

Dimitris Chatzopoulos

Input: a big dataset

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
3	30	F	R1	X
4	50	M	R2	√
5	19	M	R2	X
6	60	M	R2	√
7	20	F	R2	X
8	61	F	R3	√
9	21	F	R3	X
10	62	F	R3	√
11	63	M	R4	√
12	20	M	R4	X
13	64	F	R4	√
14	21	F	R4	X
15	65	M	R4	√
16	25	F	R4	X
17	67	M	R4	√
18	27	M	R4	X
19	28	F	R4	X
20	40	F	R4	√
21	41	F	R4	√
22	70	M	R4	√
23	29	F	R4	X
24	77	M	R5	√
25	22	M	R5	X
26	83	M	R5	√
27	82	F	R5	√
28	81	F	R5	√
29	80	M	R5	√

MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

Input: a big dataset

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
3	30	F	R1	X
4	50	M	R2	√
5	19	M	R2	X
6	60	M	R2	√
7	20	F	R2	X
8	61	F	R3	√
9	21	F	R3	X
10	62	F	R3	√
11	63	M	R4	√
12	20	M	R4	X
13	64	F	R4	√
14	21	F	R4	X
15	65	M	R4	√
16	25	F	R4	X
17	67	M	R4	√
18	27	M	R4	X
19	28	F	R4	X
20	40	F	R4	√
21	41	F	R4	√
22	70	M	R4	√
23	29	F	R4	X
24	77	M	R5	√
25	22	M	R5	X
26	83	M	R5	√
27	82	F	R5	√
28	81	F	R5	√
29	80	M	R5	√

Output: information

e.g., vaccination percentage
e.g., average age of vaccinated people

MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

Input: a big dataset

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
3	30	F	R1	X
4	50	M	R2	√
5	19	M	R2	X
6	60	M	R2	√
7	20	F	R2	X
8	61	F	R3	√
9	21	F	R3	X
10	62	F	R3	√
11	63	M	R4	√
12	20	M	R4	X
13	64	F	R4	√
14	21	F	R4	X
15	65	M	R4	√
16	25	F	R4	X
17	67	M	R4	√
18	27	M	R4	X
19	28	F	R4	X
20	40	F	R4	√
21	41	F	R4	√
22	70	M	R4	√
23	29	F	R4	X
24	77	M	R5	√
25	22	M	R5	X
26	83	M	R5	√
27	82	F	R5	√
28	81	F	R5	√
29	80	M	R5	√

Output: information

e.g., vaccination percentage
e.g., average age of vaccinated people

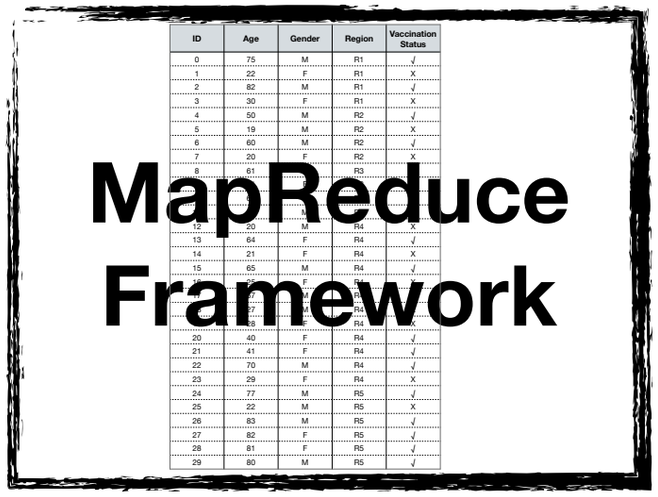
MapReduce Framework

MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

Input: a big dataset

Output: information

e.g., vaccination percentage
e.g., average age of vaccinated people



MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

Input: a big dataset

Output: information

e.g., vaccination percentage
e.g., average age of vaccinated people

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
3	30	F	R1	X
4	50	M	R2	√
5	19	M	R2	X
6	60	M	R2	√
7	20	F	R2	X
8	61		R3	
12	20	M	R4	X
13	64	F	R4	√
14	21	F	R4	X
15	65	M	R4	√
20	40	F	R4	√
21	41	F	R4	√
22	70	M	R4	√
23	29	F	R4	X
24	77	M	R5	√
25	22	M	R5	X
26	63	M	R5	√
27	82	F	R5	√
28	81	F	R5	√
29	80	M	R5	√

MapReduce Framework

60% of the population is vaccinated

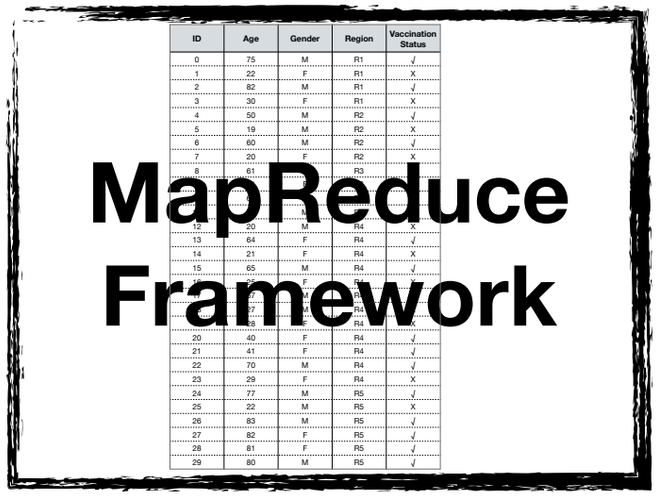
MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

Input: a big dataset

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
3	30	F	R1	X
4	50	M	R2	√
5	19	M	R2	X
6	60	M	R2	√
7	20	F	R2	X
8	61	F	R3	√
9	21	F	R3	X
10	62	F	R3	√
11	63	M	R4	√
12	20	M	R4	X
13	64	F	R4	√
14	21	F	R4	X
15	65	M	R4	√
16	25	F	R4	X
17	67	M	R4	√
18	27	M	R4	X
19	28	F	R4	X
20	40	F	R4	√
21	41	F	R4	√
22	70	M	R4	√
23	29	F	R4	X
24	77	M	R5	√
25	22	M	R5	X
26	83	M	R5	√
27	82	F	R5	√
28	81	F	R5	√
29	80	M	R5	√

Output: information

e.g., vaccination percentage
e.g., average age of vaccinated people



60% of the population is vaccinated

MapReduce is a programming paradigm for processing big datasets in a distributed fashion.

In MapReduce, data are structured in (key, value) pairs

In more detail…

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
…	…	…	…	…
28	81	M	R5	√
29	80	F	R5	√

MapReduce Framework

In more detail…

Partition dataset based on region and ignore age and gender

ID	Age	Gender	Region	Vaccination Status
0	75	M	R1	√
1	22	F	R1	X
2	82	M	R1	√
…	…	…	…	…
28	81	M	R5	√
29	80	F	R5	√

MapReduce Framework

In more detail…

Partition dataset based on region and ignore age and gender

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

MapReduce Framework

In more detail…

Partition dataset based on region and ignore age and gender

ID	Vaccination Status
11	√
12	X
28	√
19	X
29	√
20	√
21	√
22	√
23	X

MapReduce Framework

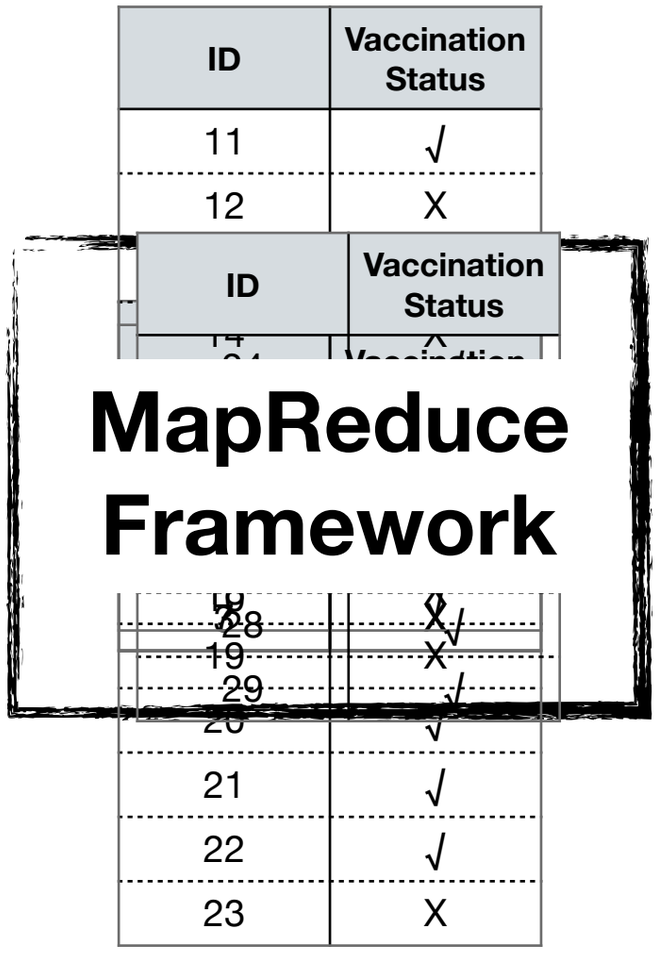
In more detail…

Partition dataset based on region and ignore age and gender

MapReduce Framework

60% of the population is vaccinated

In even more detail…



In even more detail…

MapReduce Framework

2 out of 4 are vaccinated

2 out of 4 are vaccinated

2 out of 3 are vaccinated

7 out of 13 are vaccinated

5 out of 6 are vaccinated

In even more detail…

MapReduce Framework

18 out of 30 are vaccinated

In even more detail…

MapReduce Framework

60% of the population is vaccinated

In even more detail…

A MapReduce program is composed of a ***map method*** and a ***reduce method.***

In even more detail…

Map

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

5 out of 6 are vaccinated

7 out of 13 are vaccinated

2 out of 3 are vaccinated

2 out of 4 are vaccinated

2 out of 4 are vaccinated

A MapReduce program is composed of a ***map method*** and a ***reduce method.***

- **map** performs filtering and sorting (e.g., sorting entries by region into queues, one queue for each region) and produces intermediate results

In even more detail…

Map

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Reduce

5 out of 6 are vaccinated

7 out of 13 are vaccinated

2 out of 3 are vaccinated

2 out of 4 are vaccinated

2 out of 4 are vaccinated

A MapReduce program is composed of a ***map method*** and a ***reduce method.***

- **map** performs filtering and sorting (e.g., sorting entries by region into queues, one queue for each region) and produces intermediate results

- **reduce** performs a summary operation (e.g., counting the people in each queue, yielding vaccination frequencies)

In even more detail…

Map

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Reduce

18 out of 30
are vaccinated

A MapReduce program is composed of a ***map method*** and a ***reduce method.***

- **map** performs filtering and sorting (e.g., sorting entries by region into queues, one queue for each region) and produces intermediate results

- **reduce** performs a summary operation (e.g., counting the people in each queue, yielding vaccination frequencies)

In even more detail…

Map

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Reduce

60% of the population is vaccinated

A MapReduce program is composed of a ***map method*** and a ***reduce method.***

- **map** performs filtering and sorting (e.g., sorting entries by region into queues, one queue for each region) and produces intermediate results

- **reduce** performs a summary operation (e.g., counting the people in each queue, yielding vaccination frequencies)

Map

Single-threaded processing

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Parallel processing

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Map

Single-threaded processing

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Parallel processing

ID	Vaccination Status
0	√
1	X
2	√
3	X

ID	Vaccination Status
4	√
5	X
6	√
7	X

ID	Vaccination Status
8	√
9	X
10	√

ID	Vaccination Status
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X

ID	Vaccination Status
24	√
25	X
26	√
27	√
28	√
29	√

Map

Single-threaded processing

ID	Vaccination Status
0	√
1	X
2	√
3	X
4	√
5	X
6	√
7	X
8	√
9	X
10	√
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X
24	√
25	X
26	√
27	√
28	√
29	√

Parallel processing

ID	Vaccination Status
0	√
1	X
2	√
3	X
4	√
5	X
6	√
7	X
8	√
9	X
10	√
11	√
12	X
13	√
14	X
15	√
16	X
17	√
18	X
19	X
20	√
21	√
22	√
23	X
24	√
25	X
26	√
27	√
28	√
29	√

Given a set of computing servers (*nodes*) that store parts of the dataset, each node runs map to the locally store data to generate intermediate results.

The number of the nodes and the distribution of the dataset impacts the running time of map.

Reduce

Reduce

Node 1 **Node 2** **Node 3** **Node 4** **Node 5**

Reduce

Node 1	Node 2	Node 3	Node 4	Node 5
√: 2/4	√: 2/4	√: 2/3	√: 5/6	√: 7/13
X: 2/4	X: 2/4	X: 1/3	X: 1/6	X: 6/13

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i **Node j**

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

√: 2/4
√: 2/4
√: 2/3
√: 5/6
√: 7/13

Node j

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

√: 2/4
√: 2/4
√: 2/3
√: 5/6
√: 7/13

Node j

X: 2/4
X: 2/4
X: 1/3
X: 1/6
X: 6/13

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

X: 2/4
X: 2/4
X: 1/3
X: 1/6
X: 6/13

Node j

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

√: 18/30

Node j

X: 2/4
X: 2/4
X: 1/3
X: 1/6
X: 6/13

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i **Node j**

√: 18/30

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

√: 18/30

Node j

X: 12/30

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

1) **Prepare the input to map** by selecting a key (region in this example)

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

1) **Prepare the input to map** by selecting a key (region in this example)

2) **Execute map** in each node and generate output based on another key (vaccination in this example)

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

1) **Prepare the input to map** by selecting a key (region in this example)

2) **Execute map** in each node and generate output based on another key (vaccination in this example)

3) **Shuffle** the output from map to the reduce nodes (i and j in this example)

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

1) **Prepare the input to map** by selecting a key (region in this example)

2) **Execute map** in each node and generate output based on another key (vaccination in this example)

3) **Shuffle** the output from map to the reduce nodes (i and j in this example)

4) **Execute reduce** using the shuffled output from map

Reduce

Node 1

√: 2/4
X: 2/4

Node 2

√: 2/4
X: 2/4

Node 3

√: 2/3
X: 1/3

Node 4

√: 5/6
X: 1/6

Node 5

√: 7/13
X: 6/13

i and j can be nodes 1,2,3,4,5 or separate nodes

Node i

√: 18/30

Node j

X: 12/30

Final Output: 60% of the population is vaccinated

MapReduce in a nutshell

1) **Prepare the input to map** by selecting a key (region in this example)

2) **Execute map** in each node and generate output based on another key (vaccination in this example)

3) **Shuffle** the output from map to the reduce nodes (i and j in this example)

4) **Execute reduce** using the shuffled output from map

5) **Produce the final output**

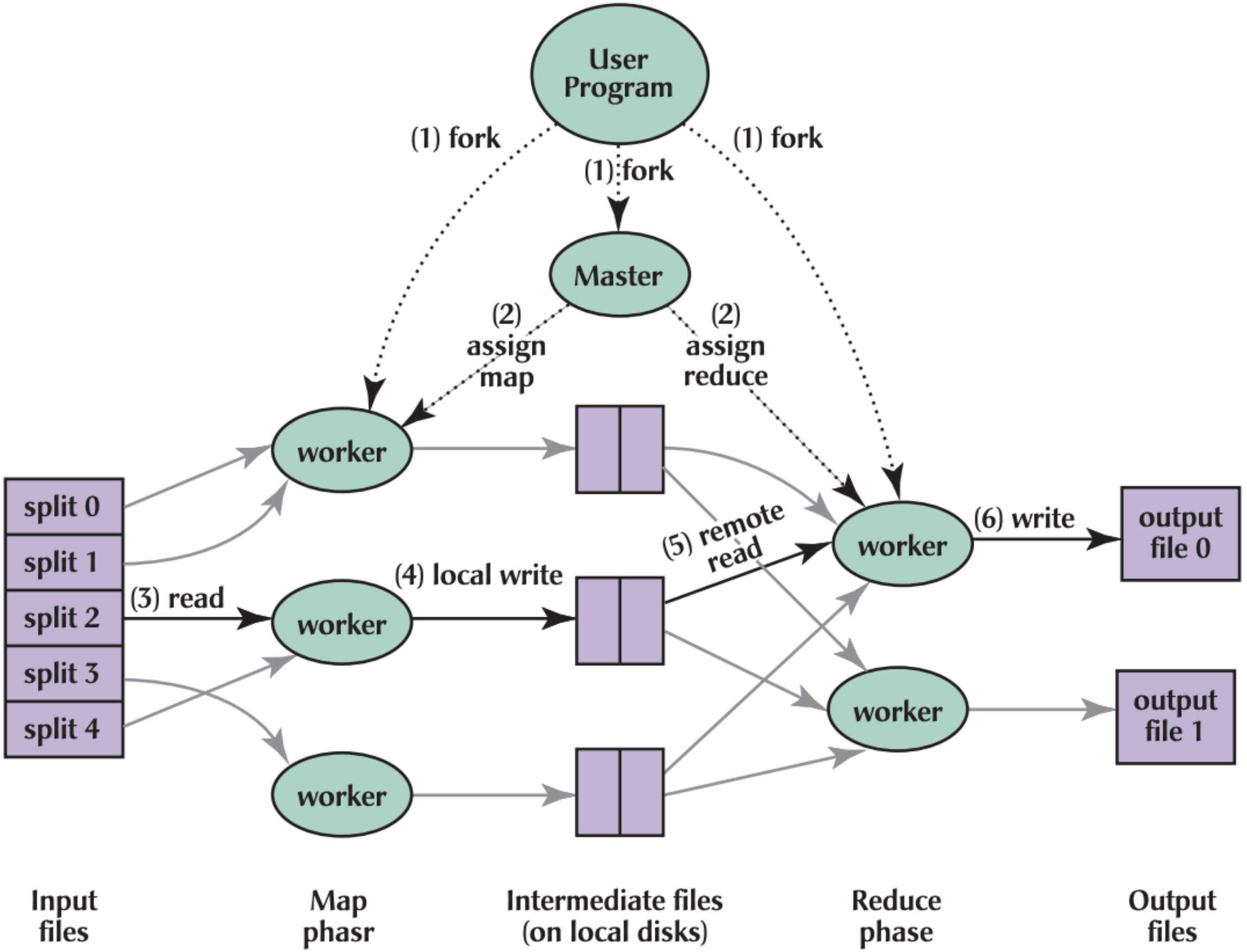


Fig. 1. Execution overview.

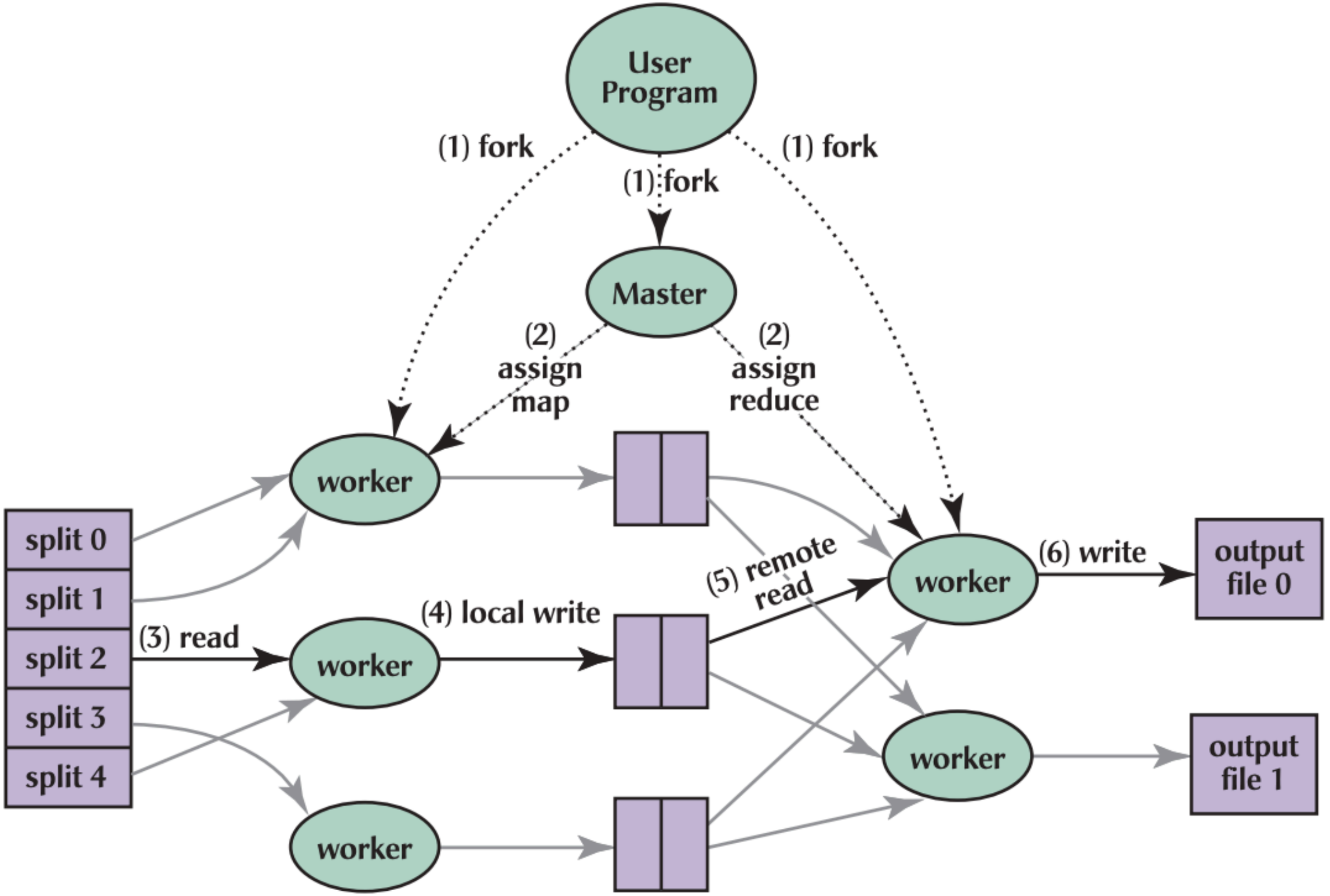


Fig. 1. Execution overview.

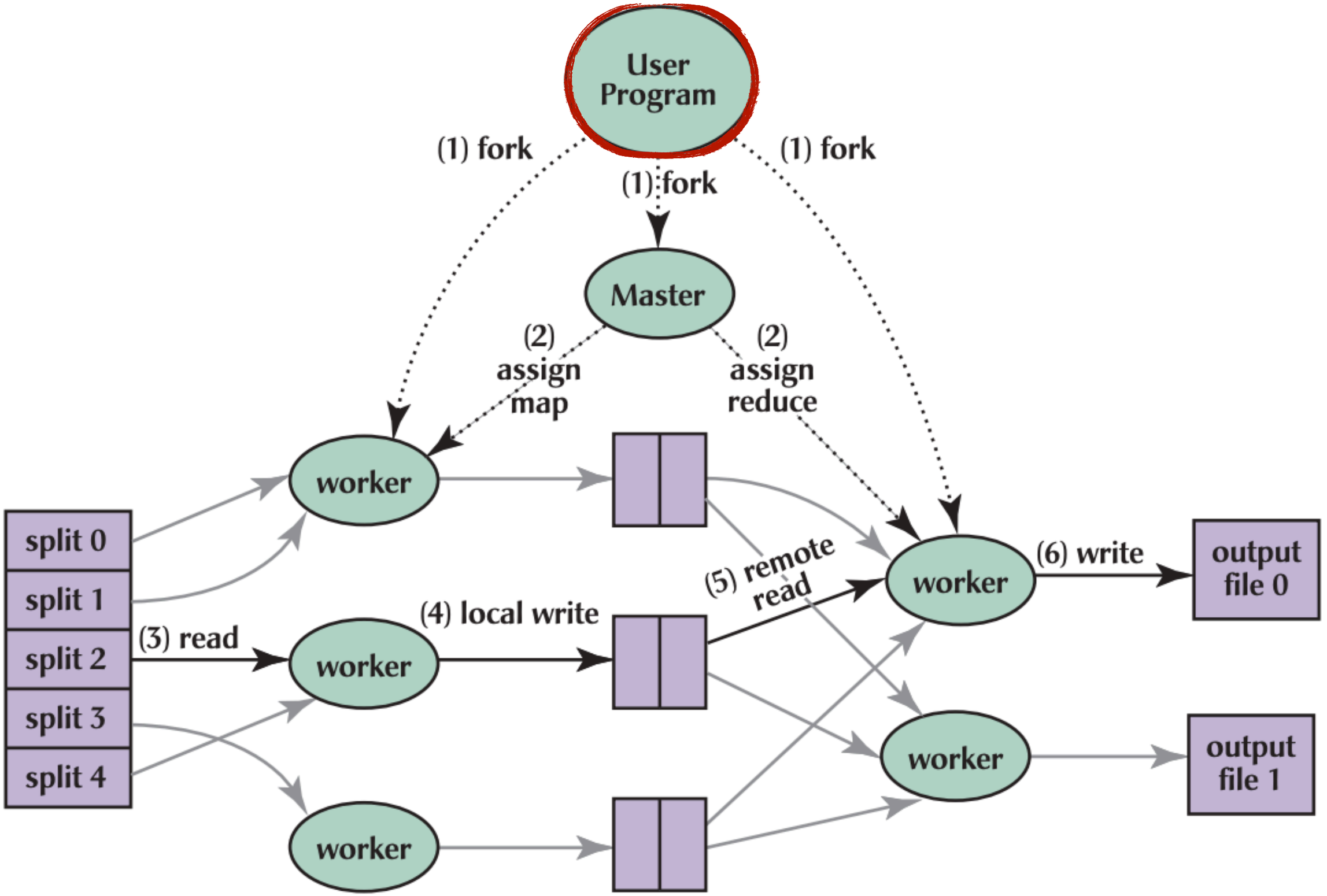


Fig. 1. Execution overview.

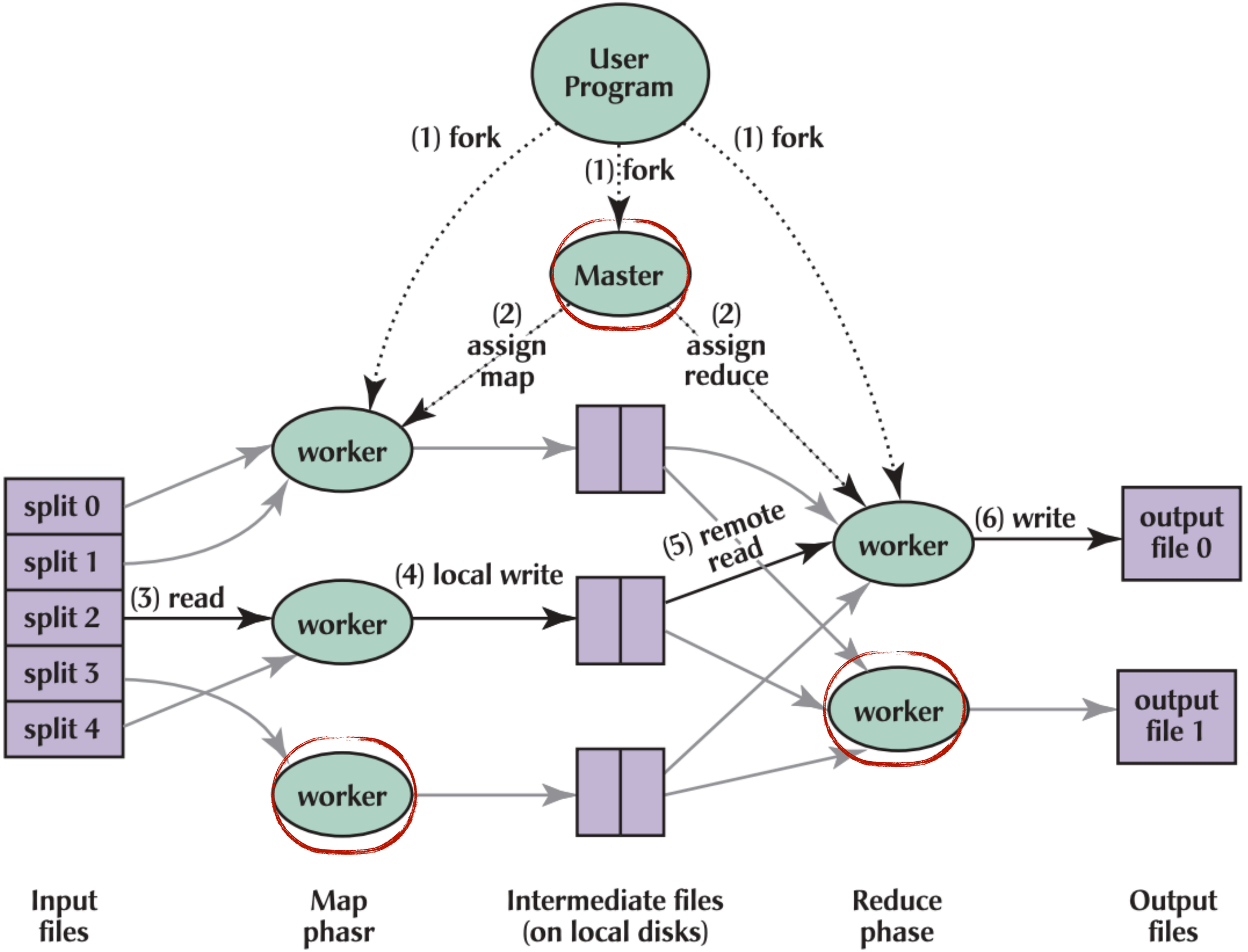


Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes

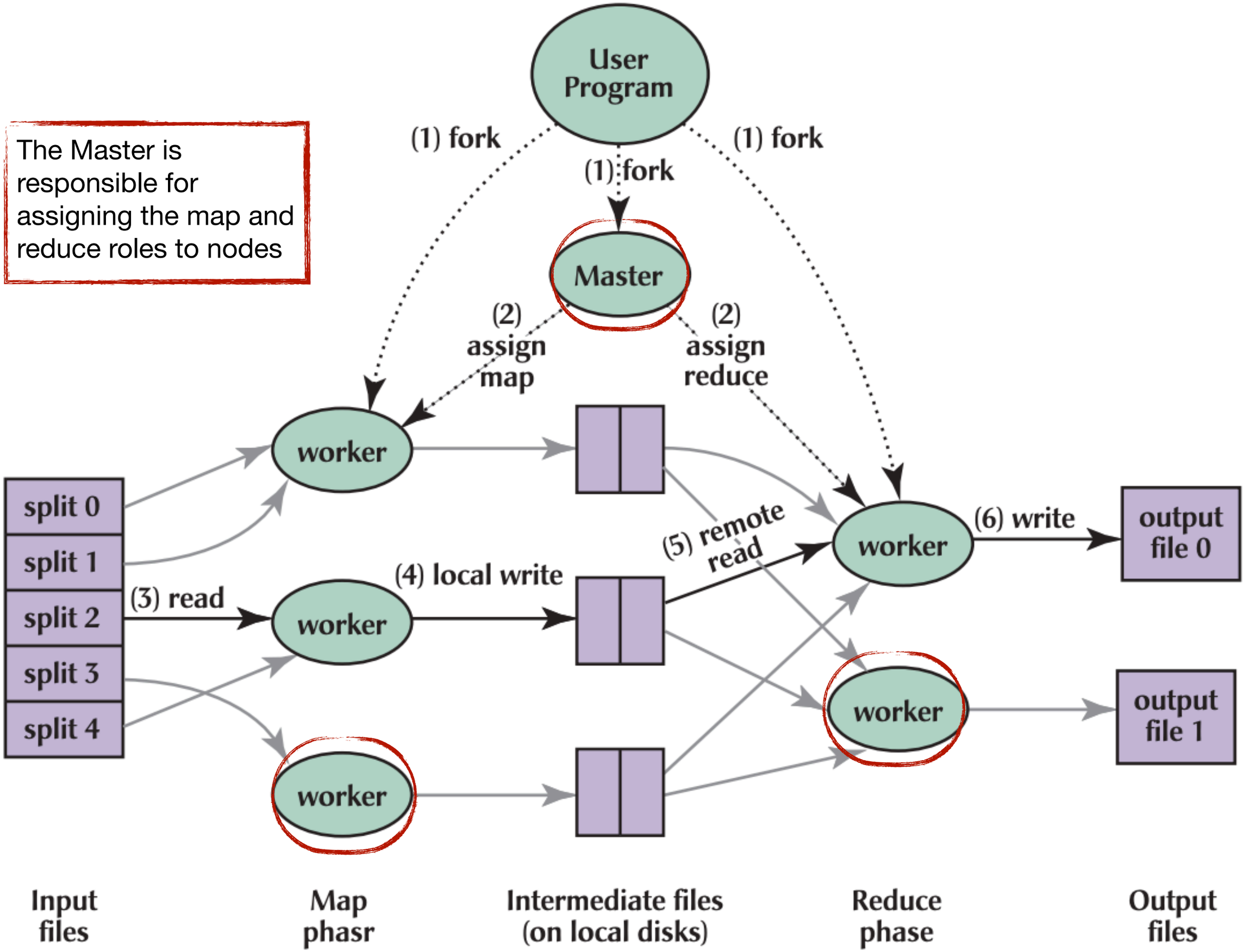
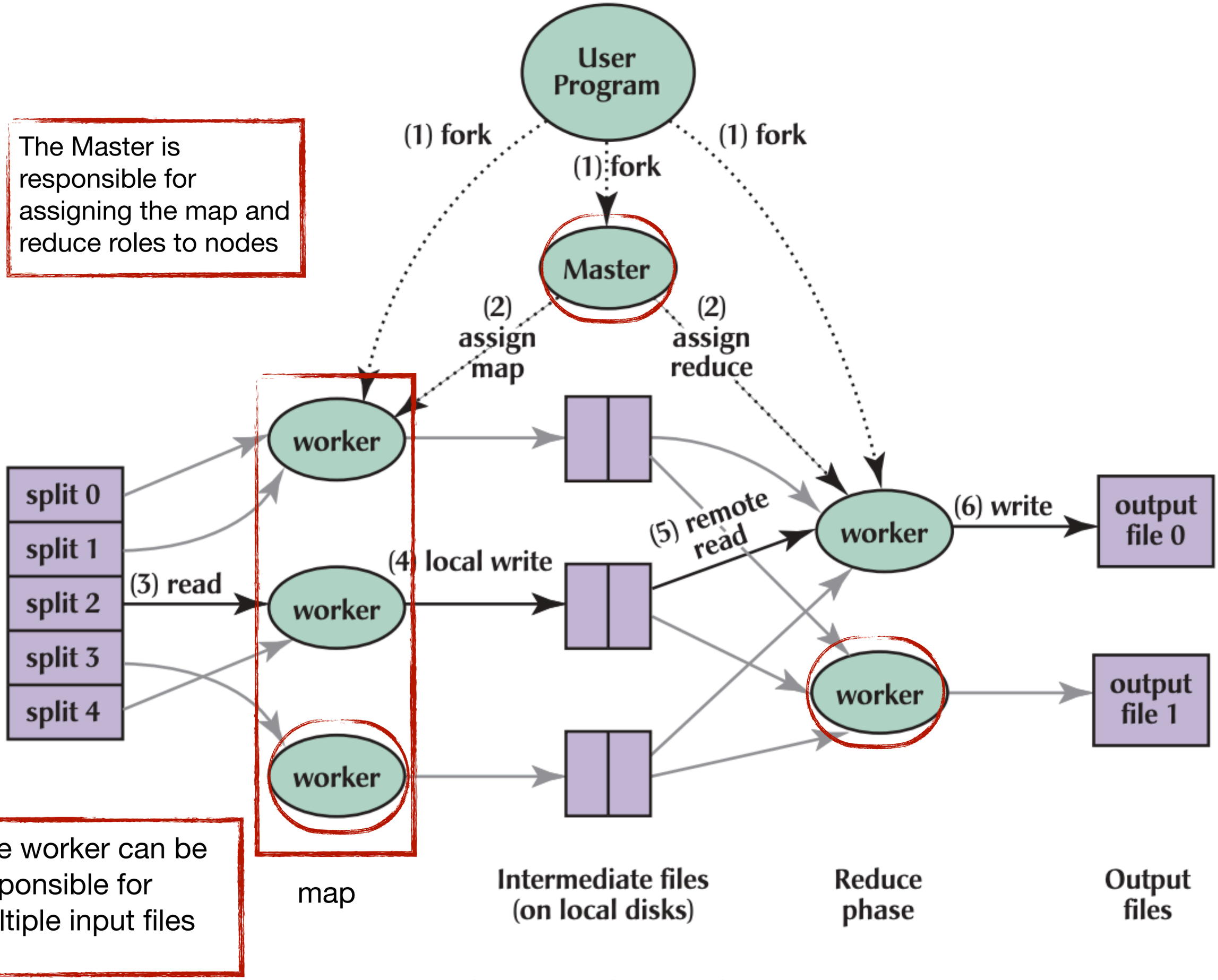


Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes

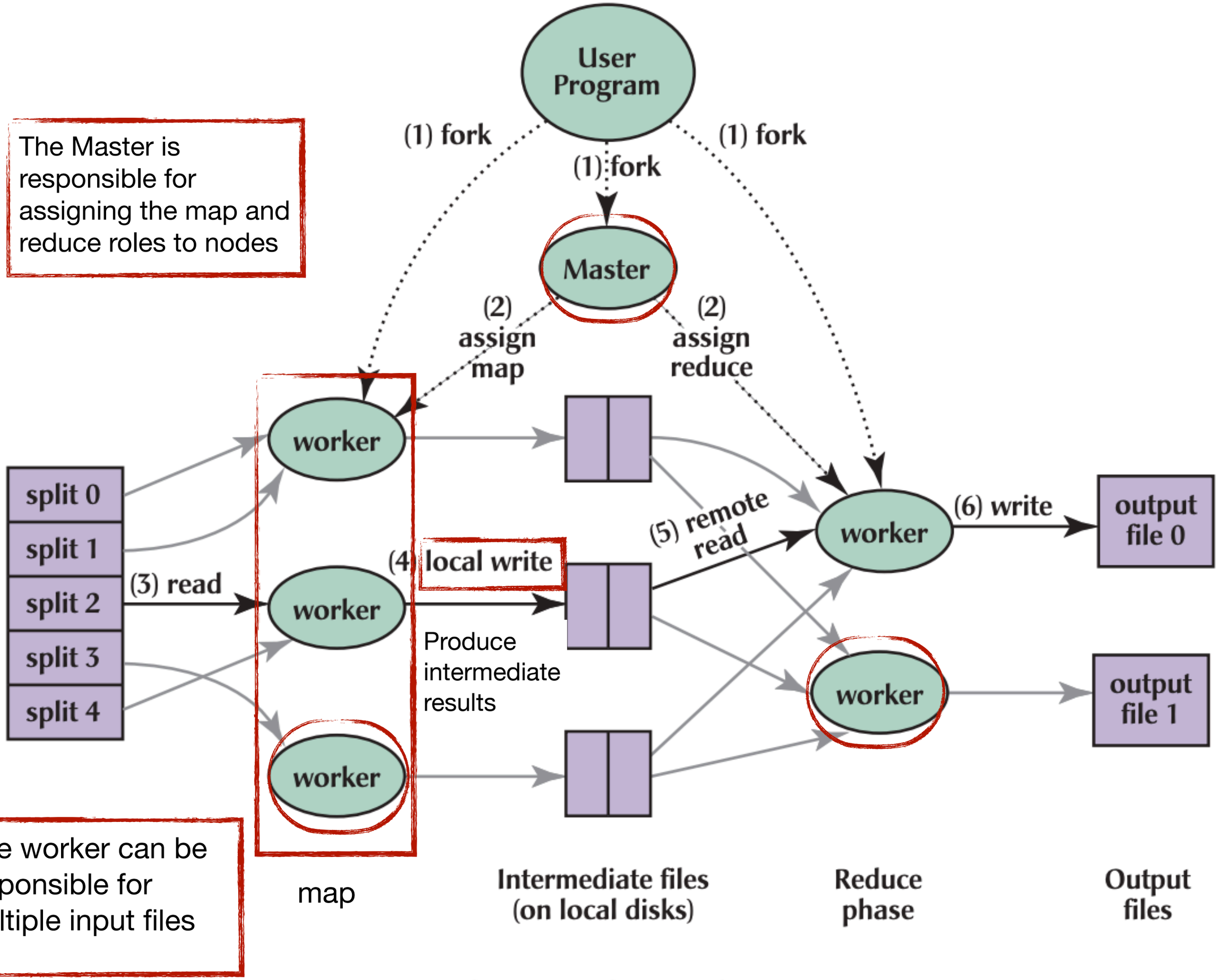


One worker can be responsible for multiple input files

Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes

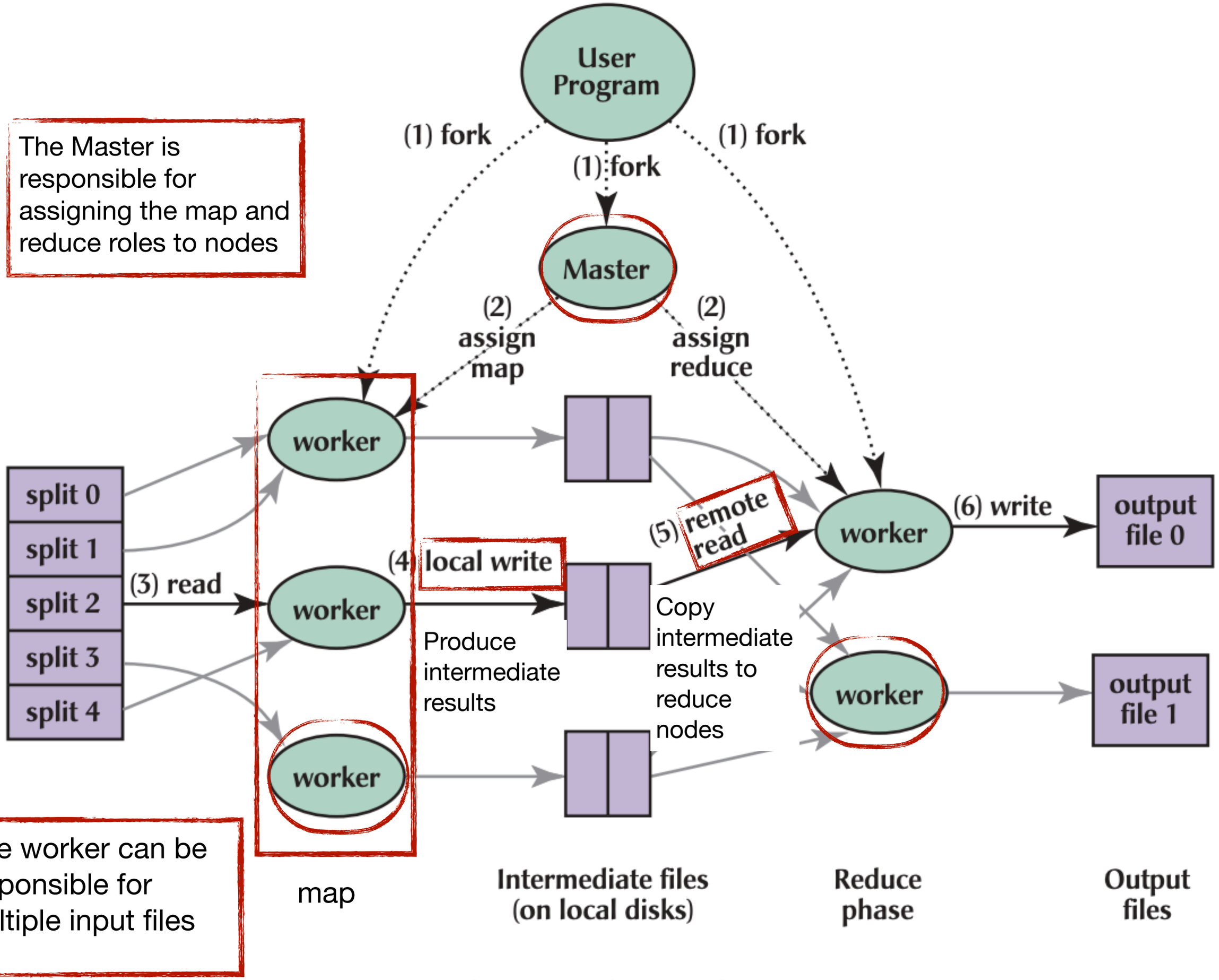


One worker can be responsible for multiple input files

Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes

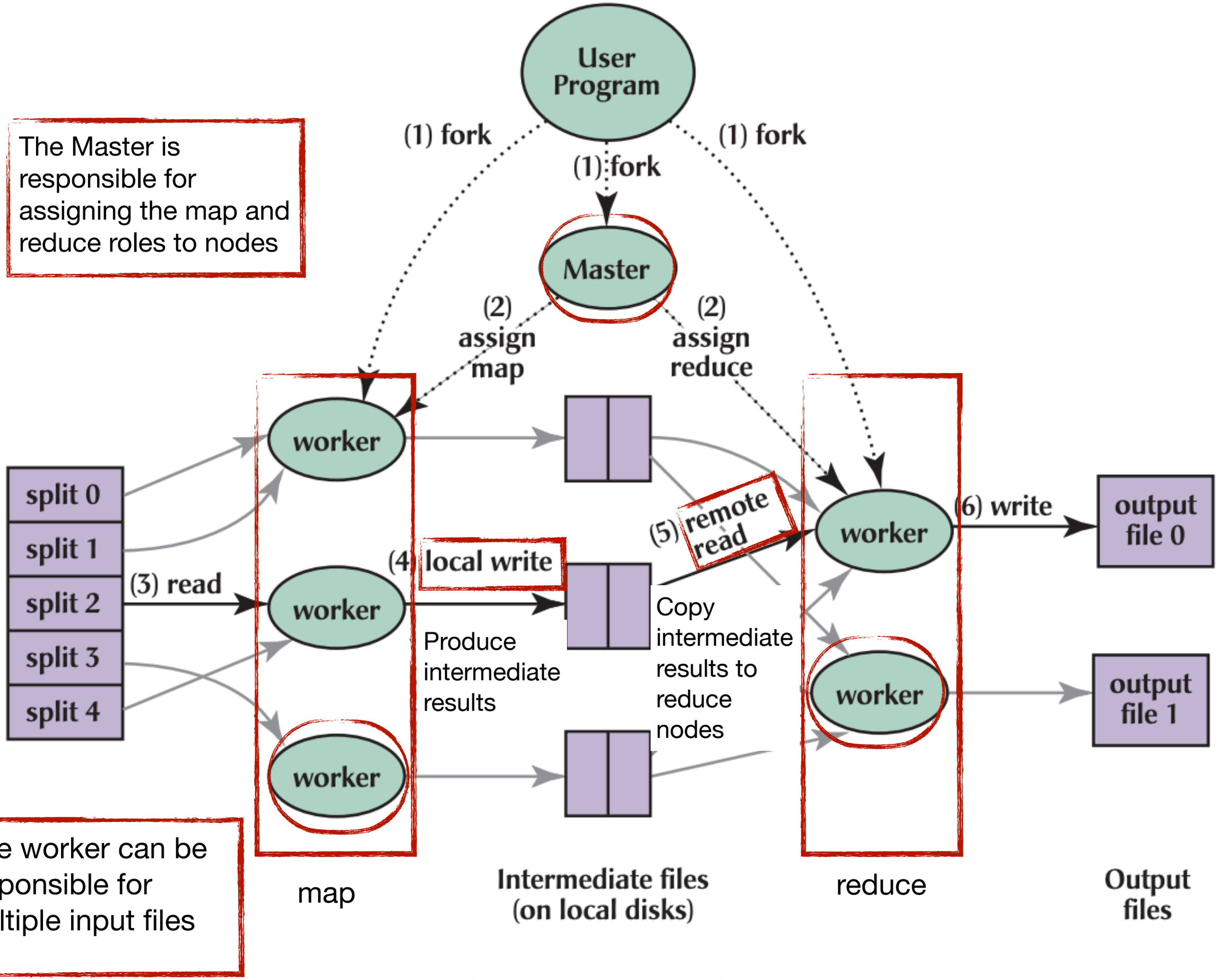


One worker can be responsible for multiple input files

Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes



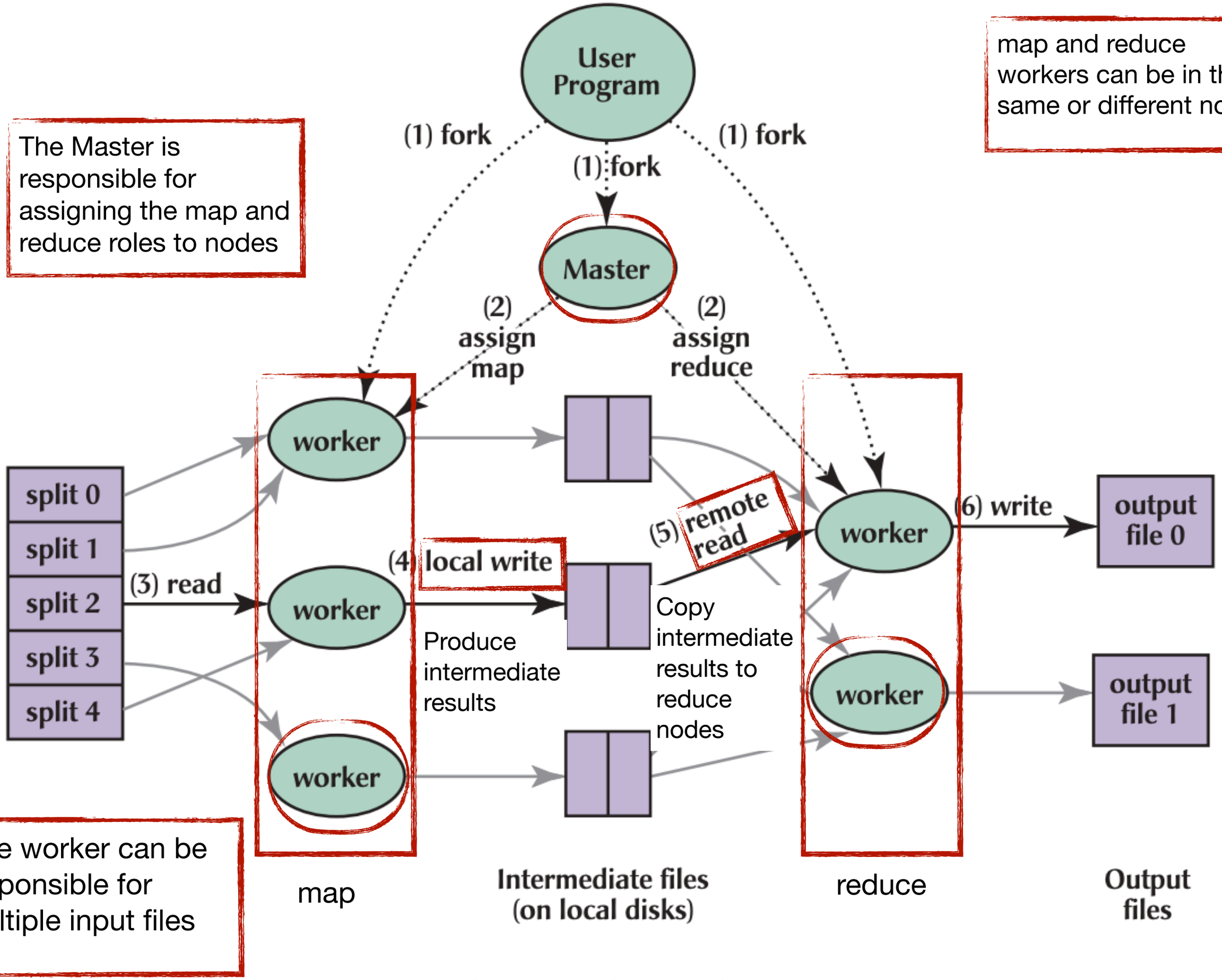
One worker can be responsible for multiple input files

Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

The Master is responsible for assigning the map and reduce roles to nodes

map and reduce workers can be in the same or different nodes



One worker can be responsible for multiple input files

Fig. 1. Execution overview.

https://dl.acm.org/doi/pdf/10.1145/1327452.1327492

In summary

In summary

Performance trade-off: The *higher the number of the computing nodes,* **the higher the computation speedup** and the **higher the data volume of the intermediate reports that need to be exchanged.**

In summary

Performance trade-off: The *higher the number of the computing nodes,* **the higher the computation speedup** and the **higher the data volume of the intermediate reports that need to be exchanged.**

Suitability of MapReduce: highly parallelisable computing tasks that involve big datasets.

Examples: sorting, searching, indexing, classification

In summary

Performance trade-off: The *higher the number of the computing nodes,* **the higher the computation speedup** and the **higher the data volume of the intermediate reports that need to be exchanged.**

Suitability of MapReduce: highly parallelisable computing tasks that involve big datasets.

Examples: sorting, searching, indexing, classification

Do not use MapReduce when:

- the dataset is small and can be processed by one computing node
- you need to process data in real-time
- you process data streams
- the nodes need to communicate a lot with each other

In summary

Performance trade-off: The *higher the number of the computing nodes,* **the higher the computation speedup** and the **higher the data volume of the intermediate reports that need to be exchanged.**

Suitability of MapReduce: highly parallelisable computing tasks that involve big datasets.

Examples: sorting, searching, indexing, classification

Do not use MapReduce when:

- the dataset is small and can be processed by one computing node
- you need to process data in real-time
- you process data streams
- the nodes need to communicate a lot with each other

For more details on the design of MapReduce, check the paper:

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

https://static.usenix.org/publications/library/proceedings/osdi04/tech/full_papers/dean/dean.pdf

In summary

Performance trade-off: The *higher the number of the computing nodes,* **the higher the computation speedup** and the **higher the data volume of the intermediate reports that need to be exchanged.**

Suitability of MapReduce: highly parallelisable computing tasks that involve big datasets.

Examples: sorting, searching, indexing, classification

Do not use MapReduce when:

- the dataset is small and can be processed by one computing node
- you need to process data in real-time
- you process data streams
- the nodes need to communicate a lot with each other