SPARK+AI
SUMMIT EUROPE

Towards Writing Scalable Spark Applications

Philipp Brunenberg, Independent Big Data Consultant

philipp-brunenberg.de @p_brunenberg Philipp Brunenberg

#AmazingSpark

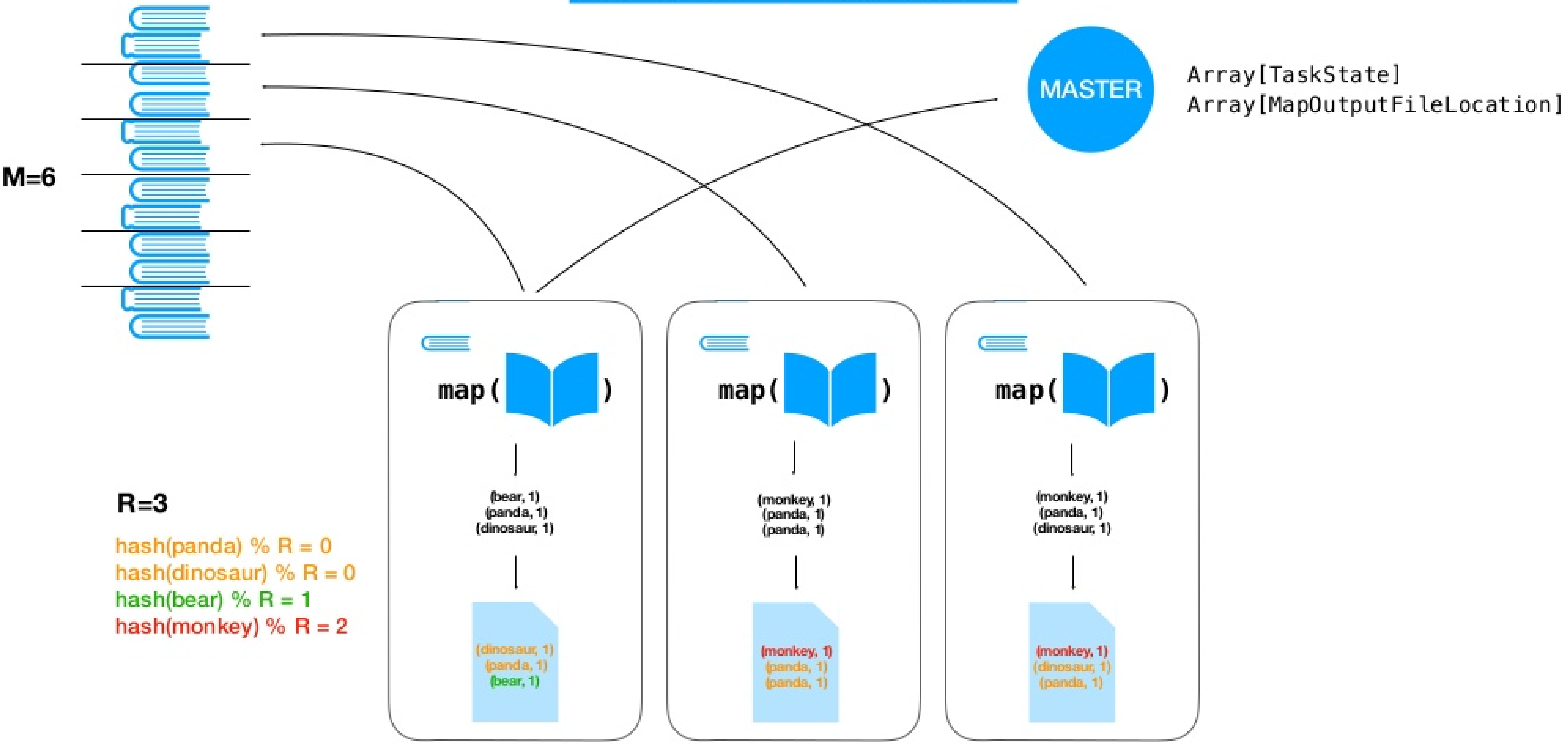
Data decomposition

independently of
all others

```
map(k1, v1) -> list(k2, v2)

reduce(k2, iterator(v2))
```

`map(k1, v1) -> list(k2, v2)`



M=6

MASTER

```
Array[TaskState]
Array[MapOutputFileLocation]
```

R=3

hash(panda) % R = 0
hash(dinosaur) % R = 0
hash(bear) % R = 1
hash(monkey) % R = 2

`map(` `)`

(bear, 1)
(panda, 1)
(dinosaur, 1)

(dinosaur, 1)
(panda, 1)
(bear, 1)

`map(` `)`

(monkey, 1)
(panda, 1)
(panda, 1)

(monkey, 1)
(panda, 1)
(panda, 1)

`map(` `)`

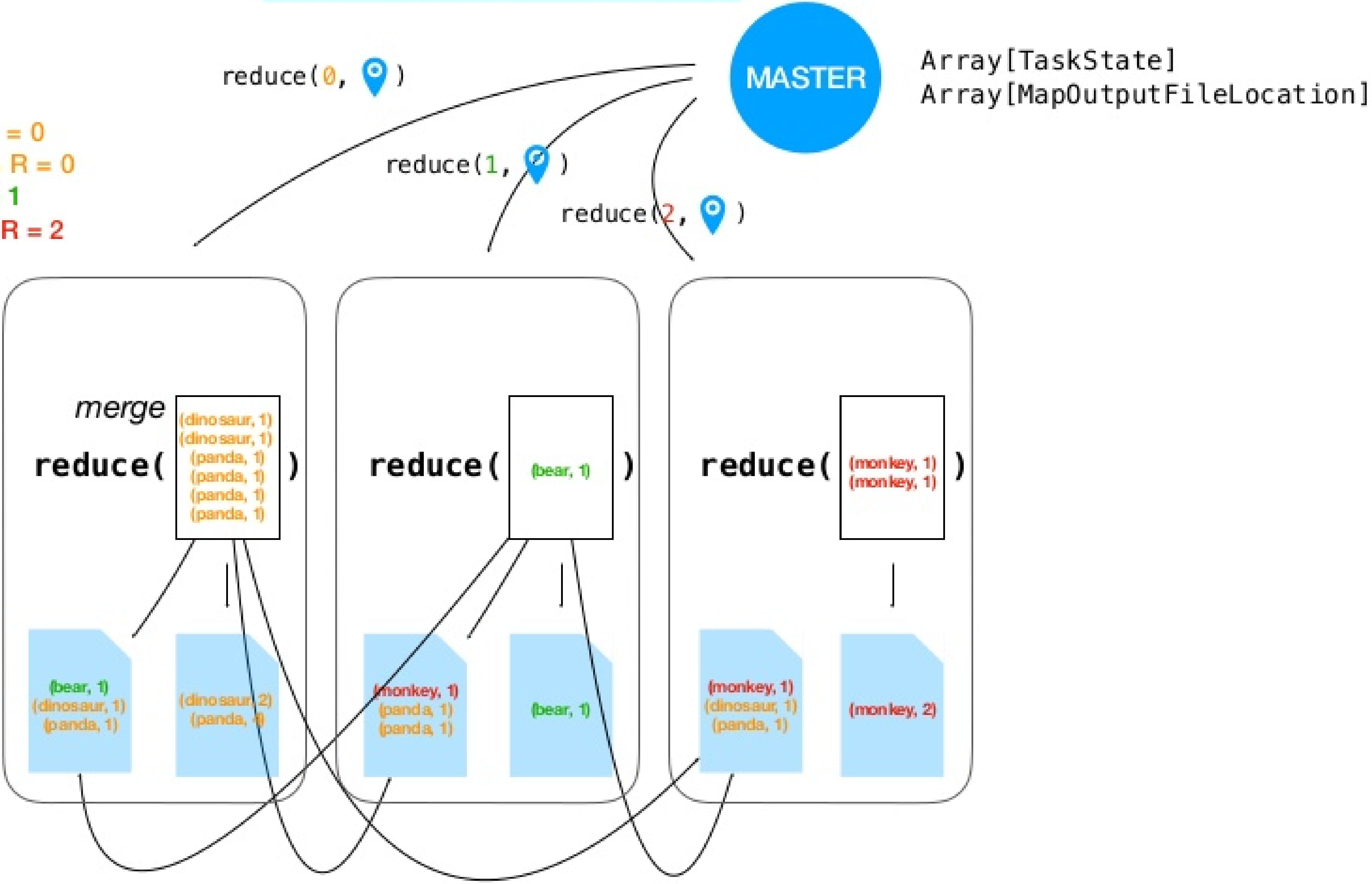
(monkey, 1)
(panda, 1)
(dinosaur, 1)

(monkey, 1)
(dinosaur, 1)
(panda, 1)

reduce(k2, iterator(v2))

R=3

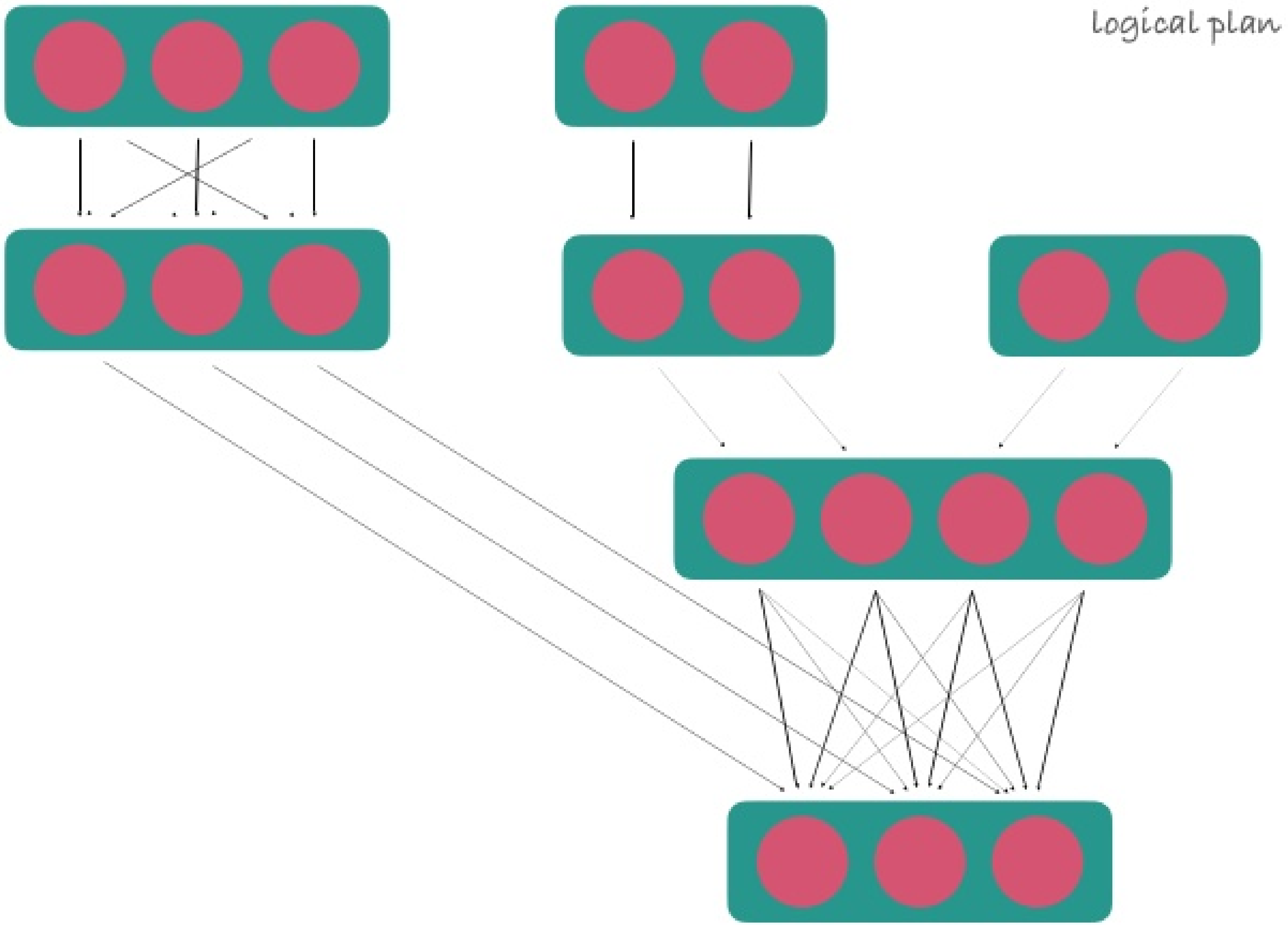
hash(panda) % R = 0
hash(dinosaur) % R = 0
hash(bear) % R = 1
hash(monkey) % R = 2


MASTER
Array[TaskState]
Array[MapOutputFileLocation]
reduce(0,)
reduce(1,)
reduce(2,)
merge
reduce((dinosaur, 1) (dinosaur, 1) (panda, 1) (panda, 1) (panda, 1) (panda, 1))
reduce((bear, 1))
reduce((monkey, 1) (monkey, 1))
(bear, 1) (dinosaur, 1) (panda, 1)
(dinosaur, 2) (panda, 4)
(monkey, 1) (panda, 1) (panda, 1)
(bear, 1)
(monkey, 1) (dinosaur, 1) (panda, 1)
(monkey, 2)

User Program

```
spark
  .load()
  .withColumn()
  .groupBy()
  .count()
  .show()
```

```
rdd1 = sparkContext
      .parallelize(…)
      .repartition(3)
rdd2 = sparkContext
      .parallelize(…)
      .map(…)
rdd3 = sparkContext
      .parallelize(…)
rdd2.union(rdd2)
      .union(rdd)
      .join(rdd1)
      .count()
```



logical plan

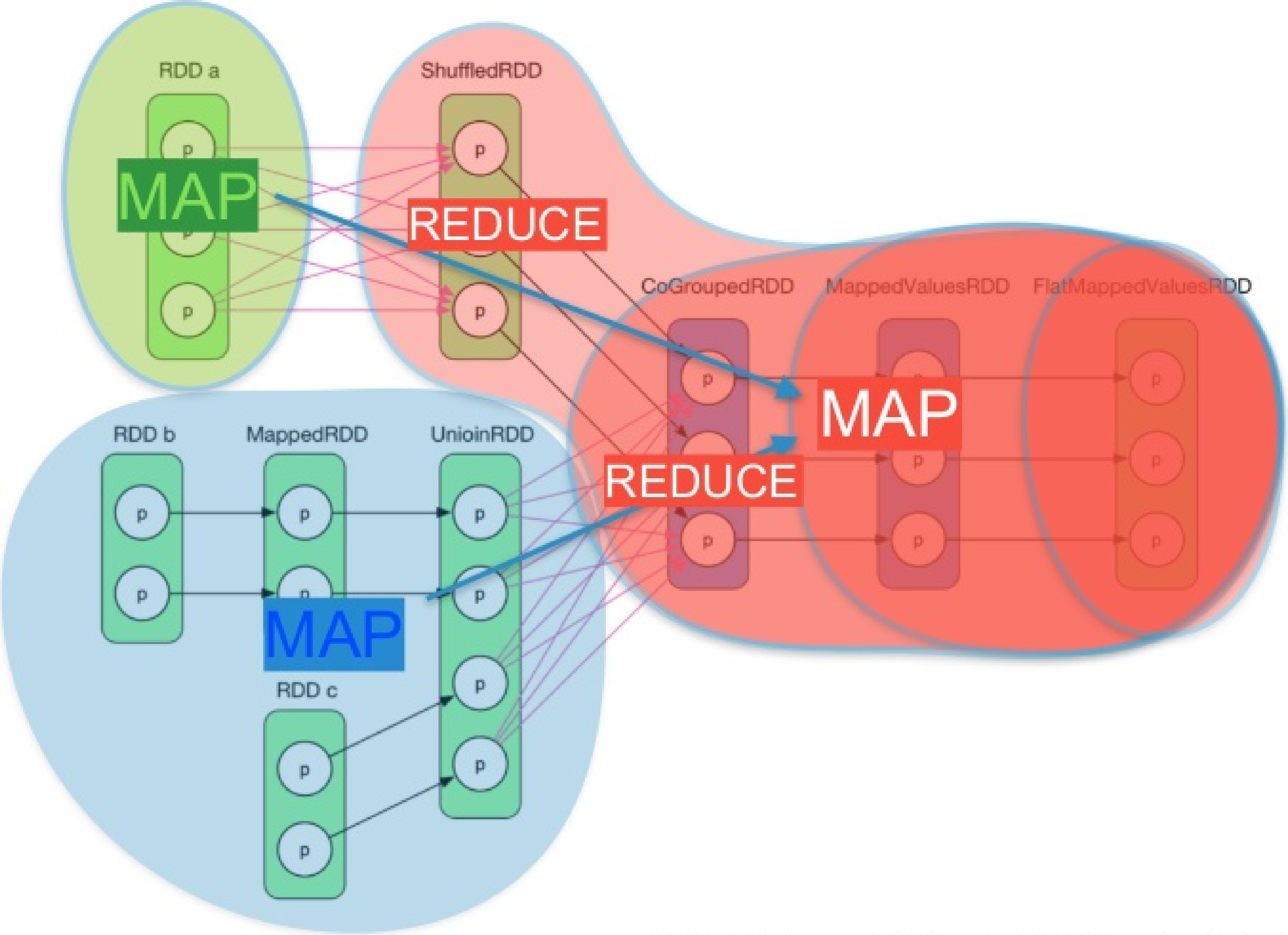
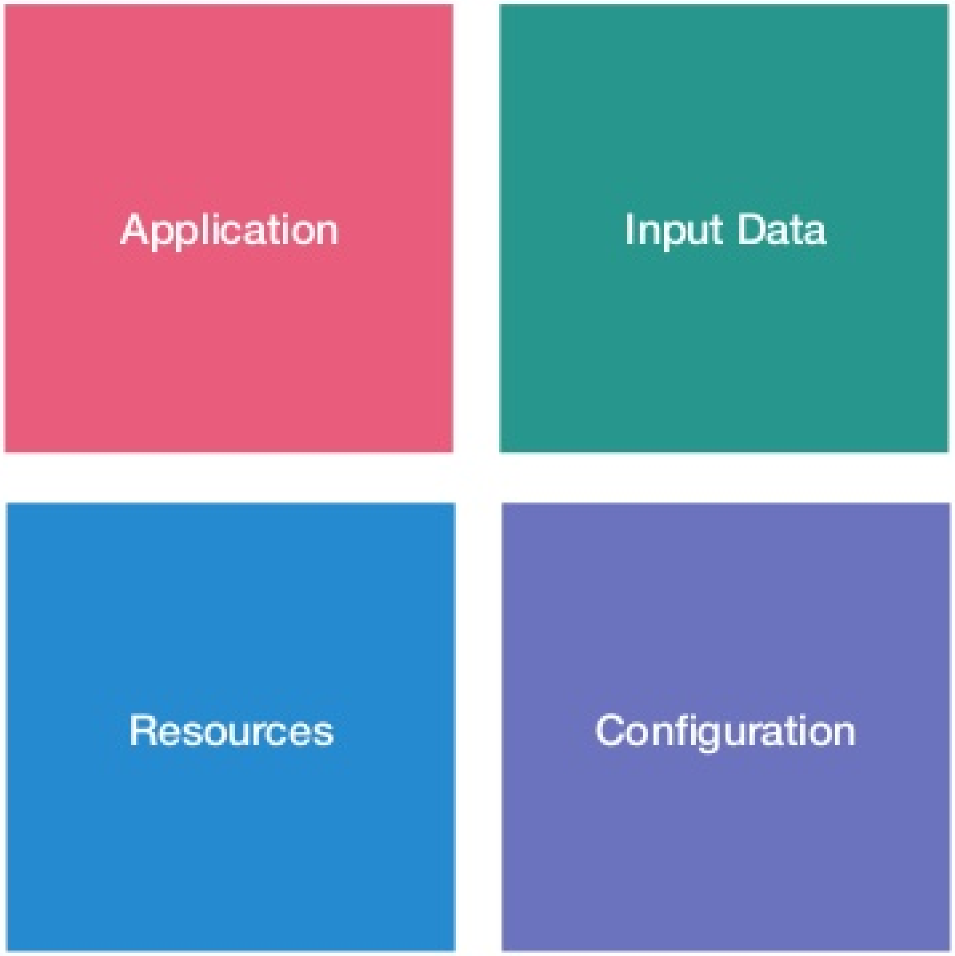


Image from: https://github.com/JerryLead/SparkInternals/blob/master/PNGfigures/ComplexJob.png

Performance Bottlenecks

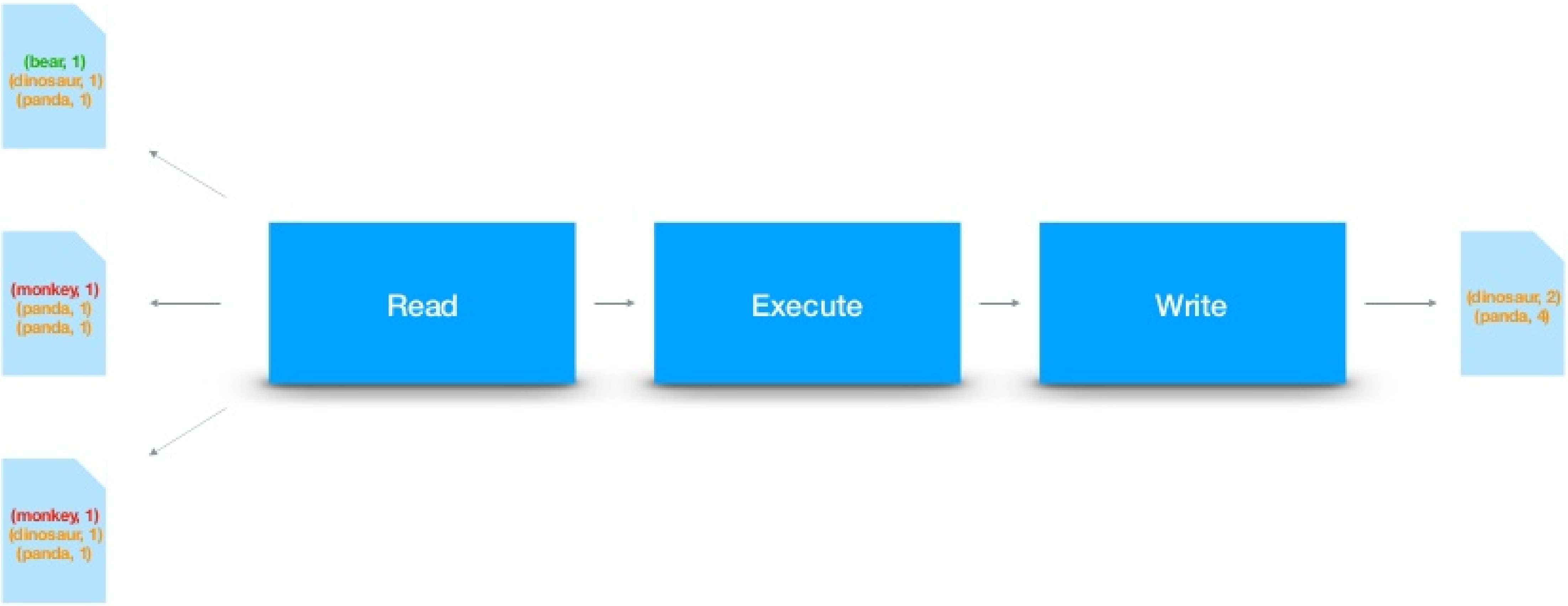


Application

Input Data

Resources

Configuration



(bear, 1)
(dinosaur, 1)
(panda, 1)

(monkey, 1)
(panda, 1)
(panda, 1)

(monkey, 1)
(dinosaur, 1)
(panda, 1)

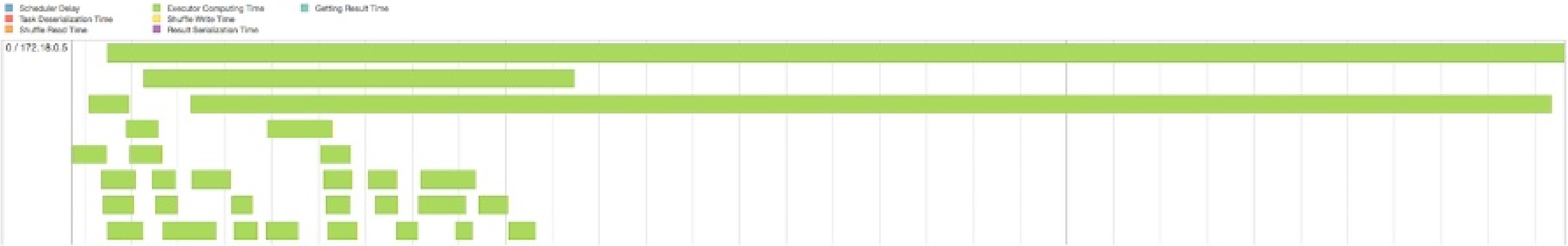
Read

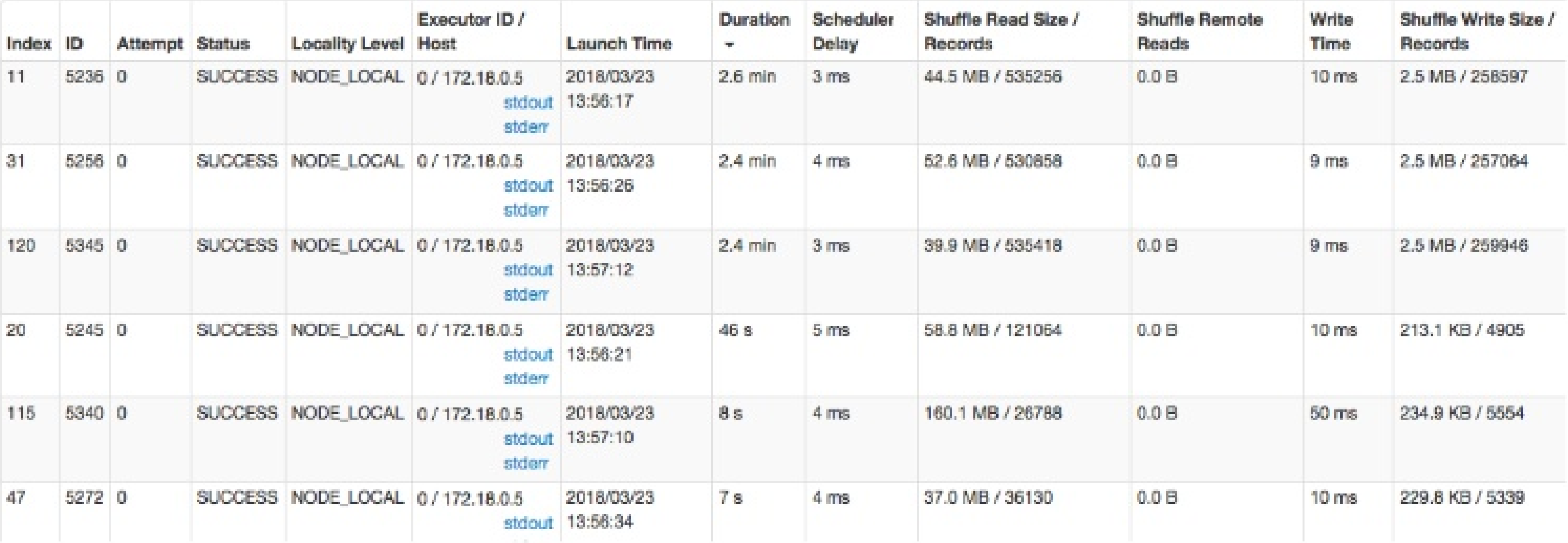
Execute

Write

(dinosaur, 2)
(panda, 4)

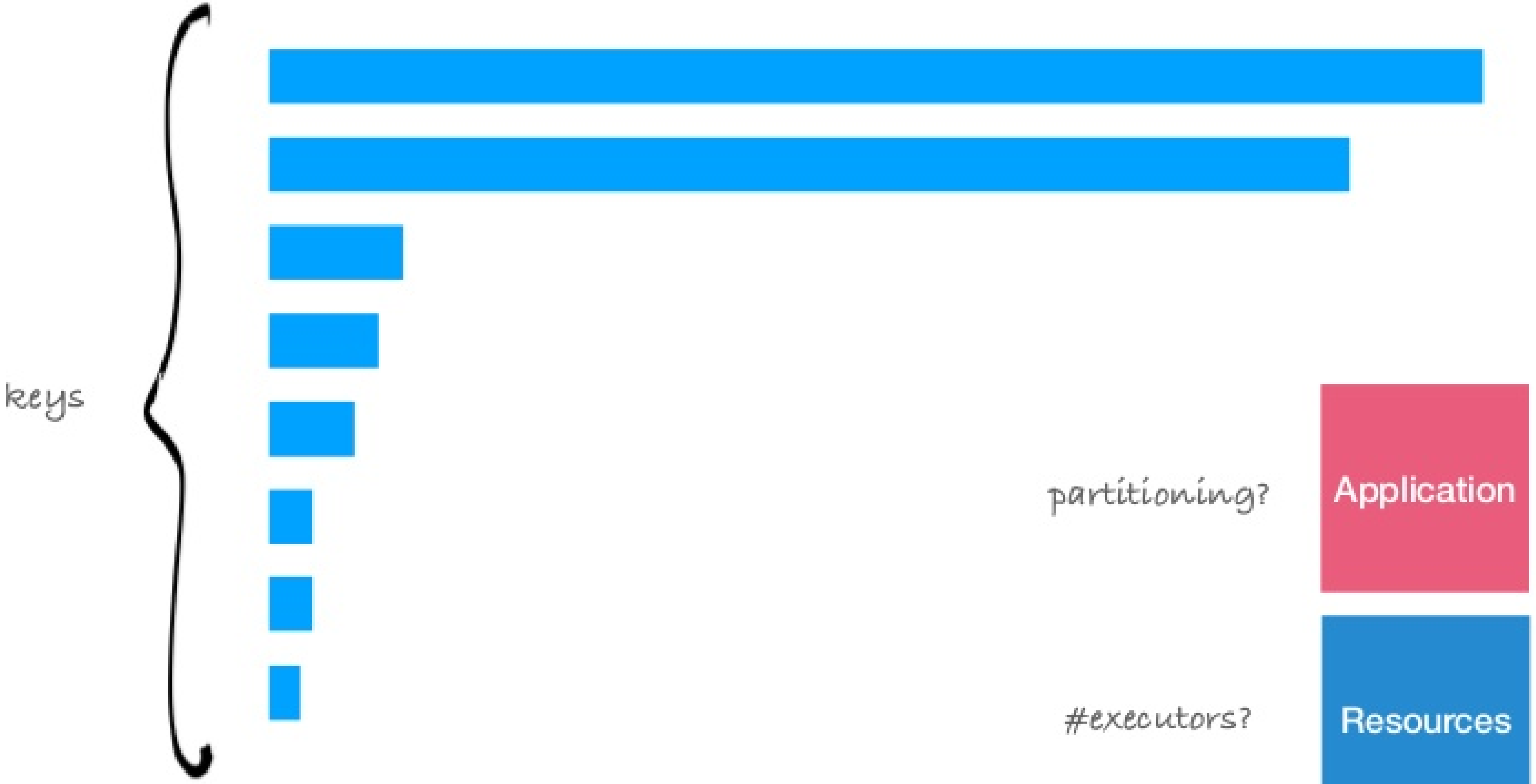
Stragglers

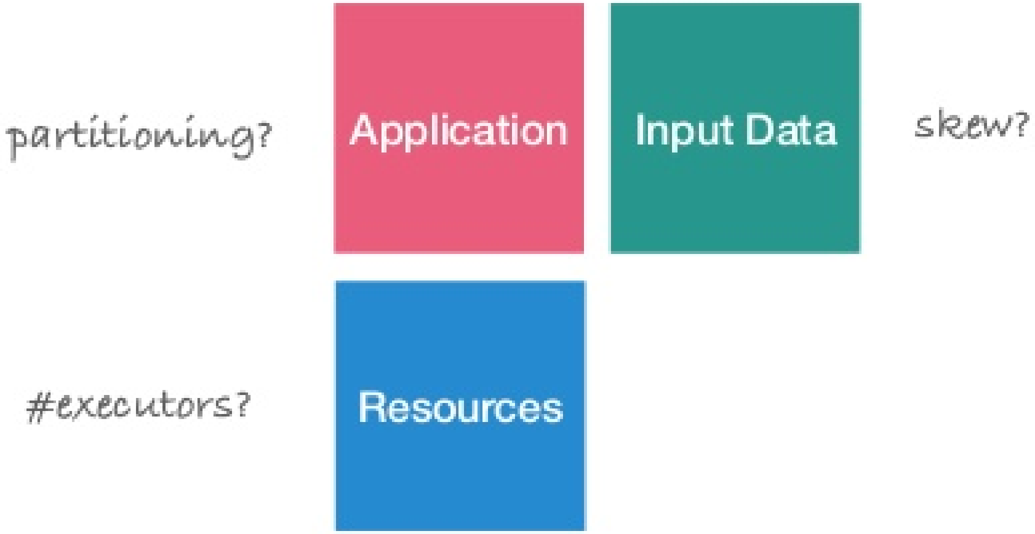




Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration ▾	Scheduler Delay	Shuffle Read Size / Records	Shuffle Remote Reads	Write Time	Shuffle Write Size / Records
11	5236	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout stderr	2018/03/23 13:56:17	2.6 min	3 ms	44.5 MB / 535256	0.0 B	10 ms	2.5 MB / 258597
31	5256	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout stderr	2018/03/23 13:56:26	2.4 min	4 ms	52.6 MB / 530858	0.0 B	9 ms	2.5 MB / 257064
120	5345	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout stderr	2018/03/23 13:57:12	2.4 min	3 ms	39.9 MB / 535418	0.0 B	9 ms	2.5 MB / 259946
20	5245	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout stderr	2018/03/23 13:56:21	46 s	5 ms	58.8 MB / 121064	0.0 B	10 ms	213.1 KB / 4905
115	5340	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout stderr	2018/03/23 13:57:10	8 s	4 ms	160.1 MB / 26788	0.0 B	50 ms	234.9 KB / 5554
47	5272	0	SUCCESS	NODE_LOCAL	0 / 172.18.0.5 stdout	2018/03/23 13:56:34	7 s	4 ms	37.0 MB / 36130	0.0 B	10 ms	229.8 KB / 5339

Stragglers





partitioning?

Application

Input Data

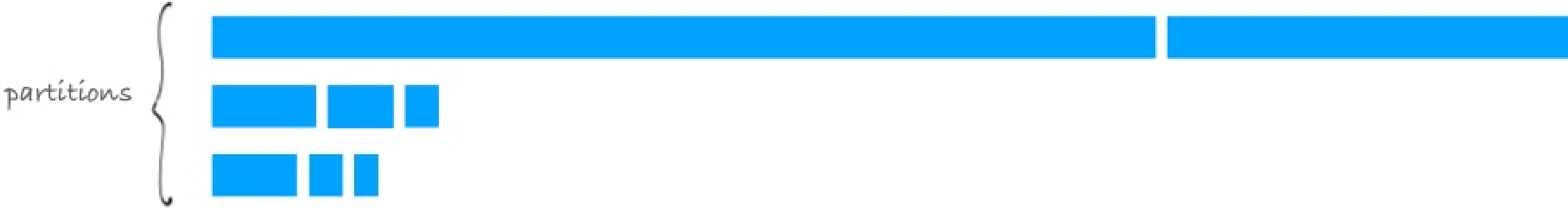
skew?

#executors?

Resources

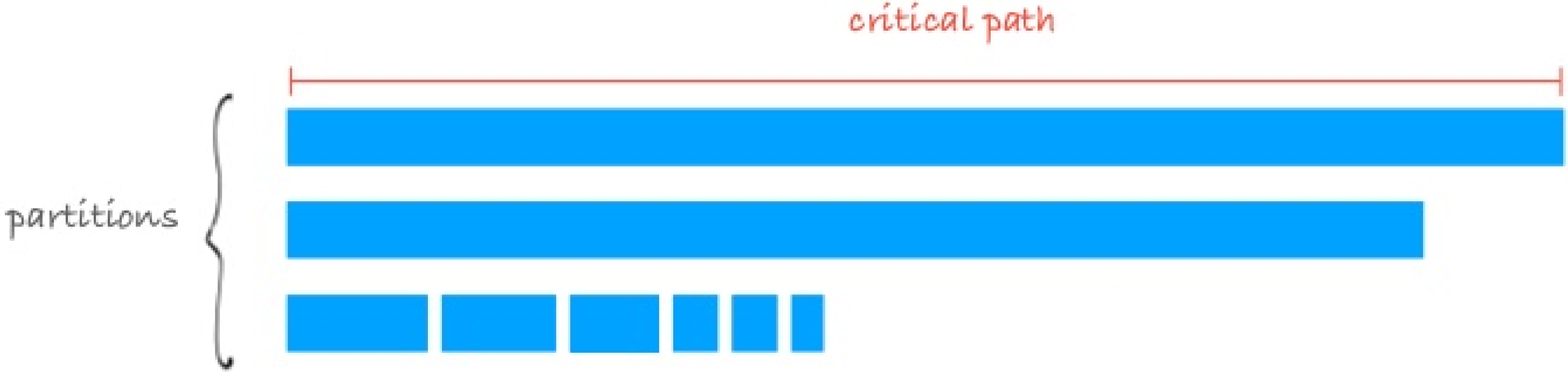
Stragglers

Partitioning - worst case

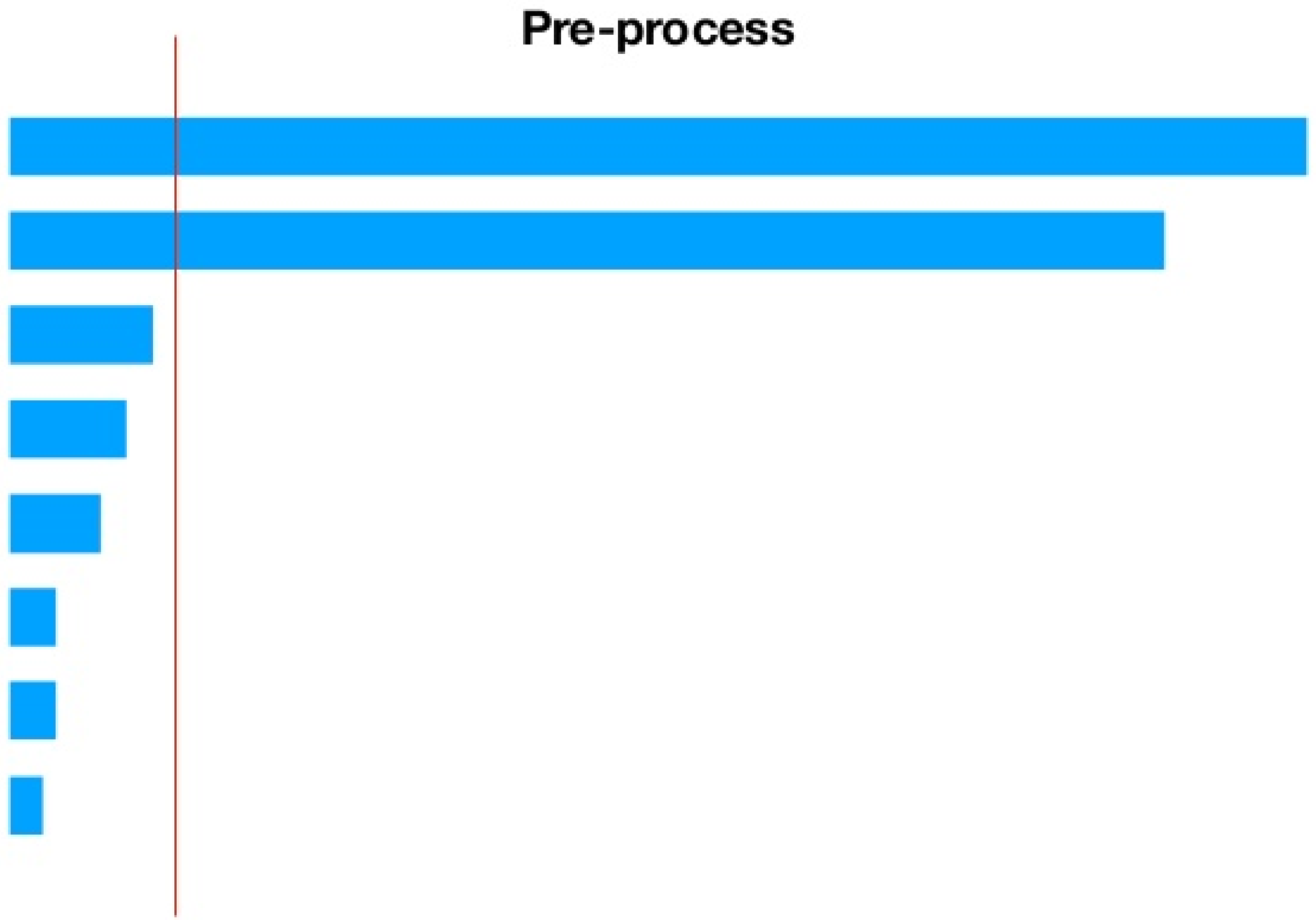


Stragglers

Partitioning - optimal case

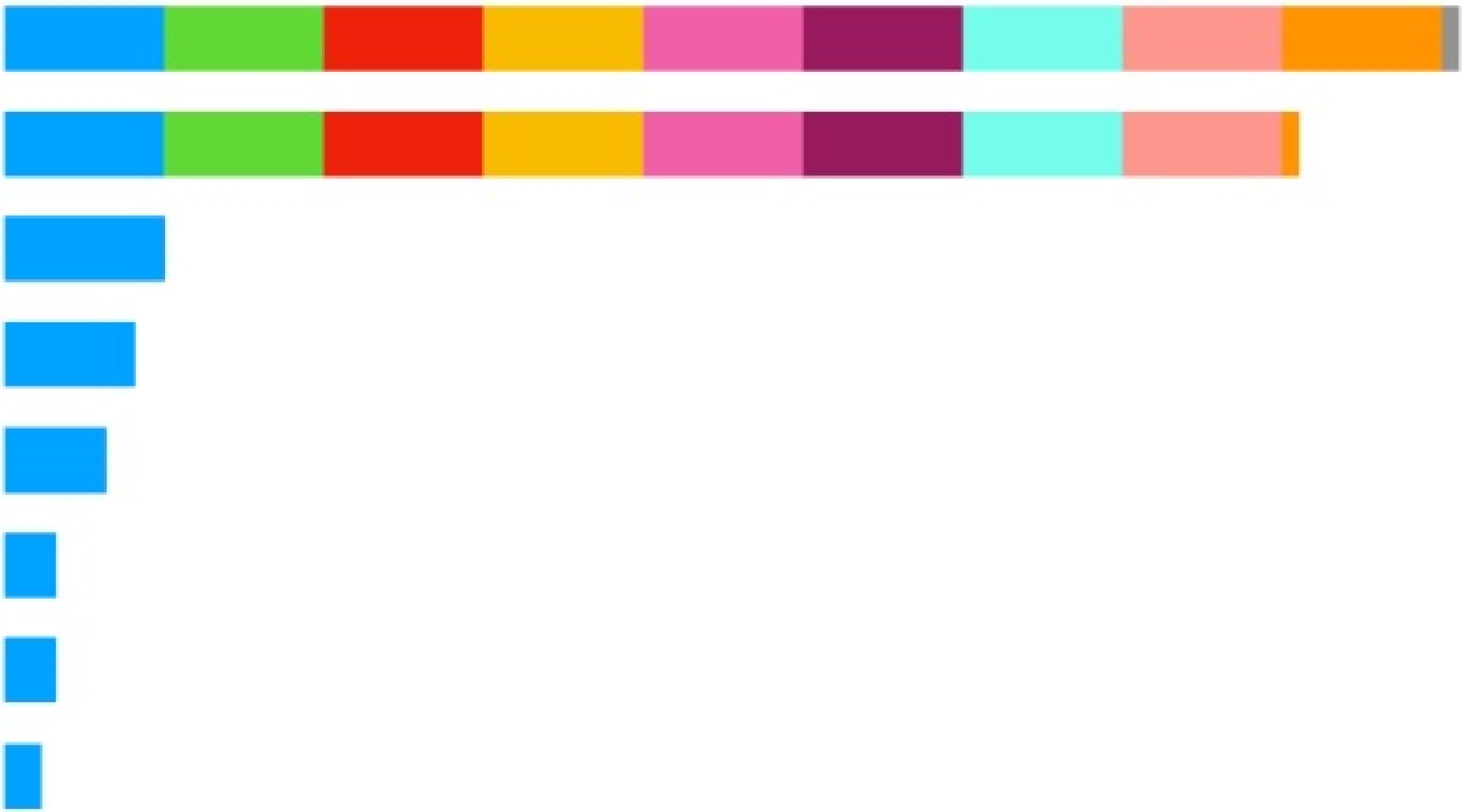


Stragglers



Stragglers

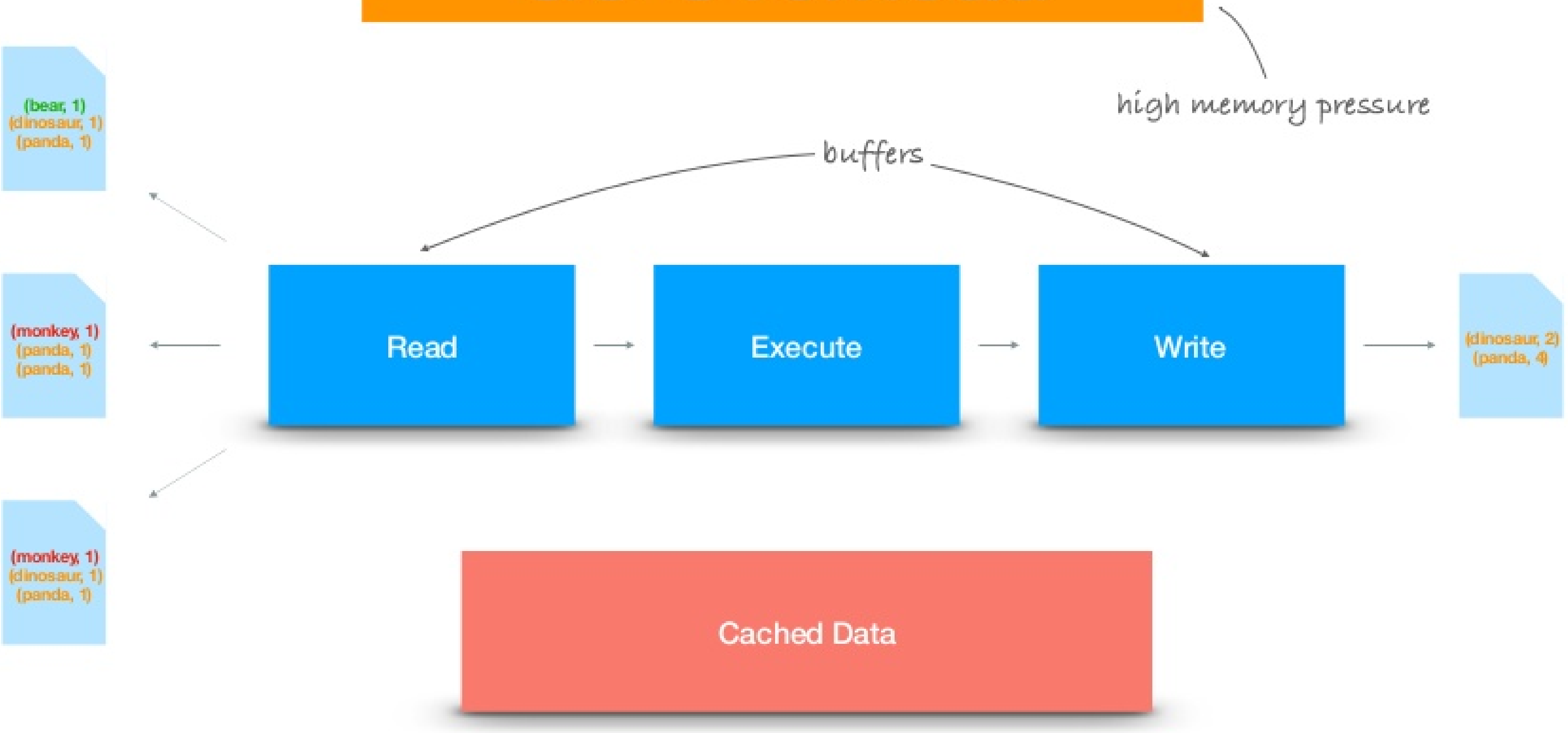
Salting



no idea how large
this will become

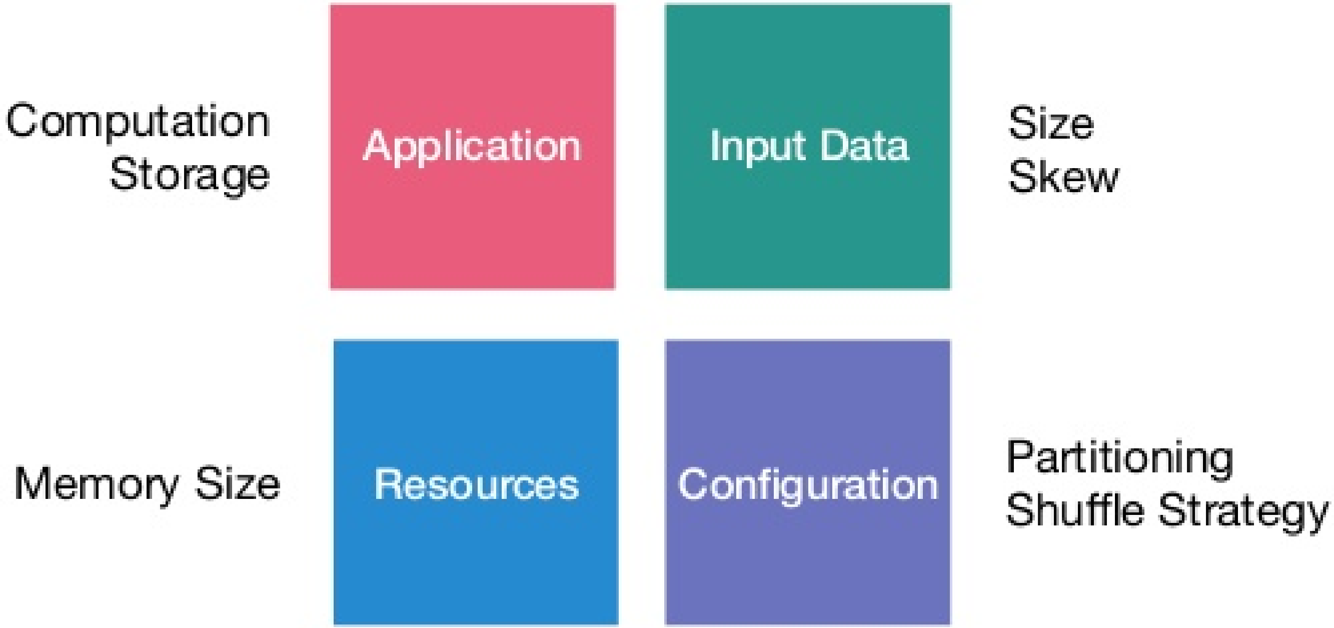
```
case class DuplicateClass(id: Long, docs: List[Document])
```

GC Overhead


high memory pressure
buffers
(bear, 1)
(dinosaur, 1)
(panda, 1)
(monkey, 1)
(panda, 1)
(panda, 1)
(monkey, 1)
(dinosaur, 1)
(panda, 1)
Read
Execute
Write
(dinosaur, 2)
(panda, 4)
Cached Data

GC Overhead

high memory pressure



Computation
Storage

Application

Input Data

Size
Skew

Memory Size

Resources

Configuration

Partitioning
Shuffle Strategy

Stragglers!

SCAN
ME

https://goo.gl/kRLy1t

Questions?

Follow me for updates and more resources.

@p_brunenberg

Philipp Brunenberg

www.philipp-brunenberg.de