APACHE Spark

LOAD BIG DATA EFFICIENTLY

PART 8: HOW SPARK PARTITIONS INFLUENCE LOADING SPEEDS

DATA WITH NIKK THE GREEK

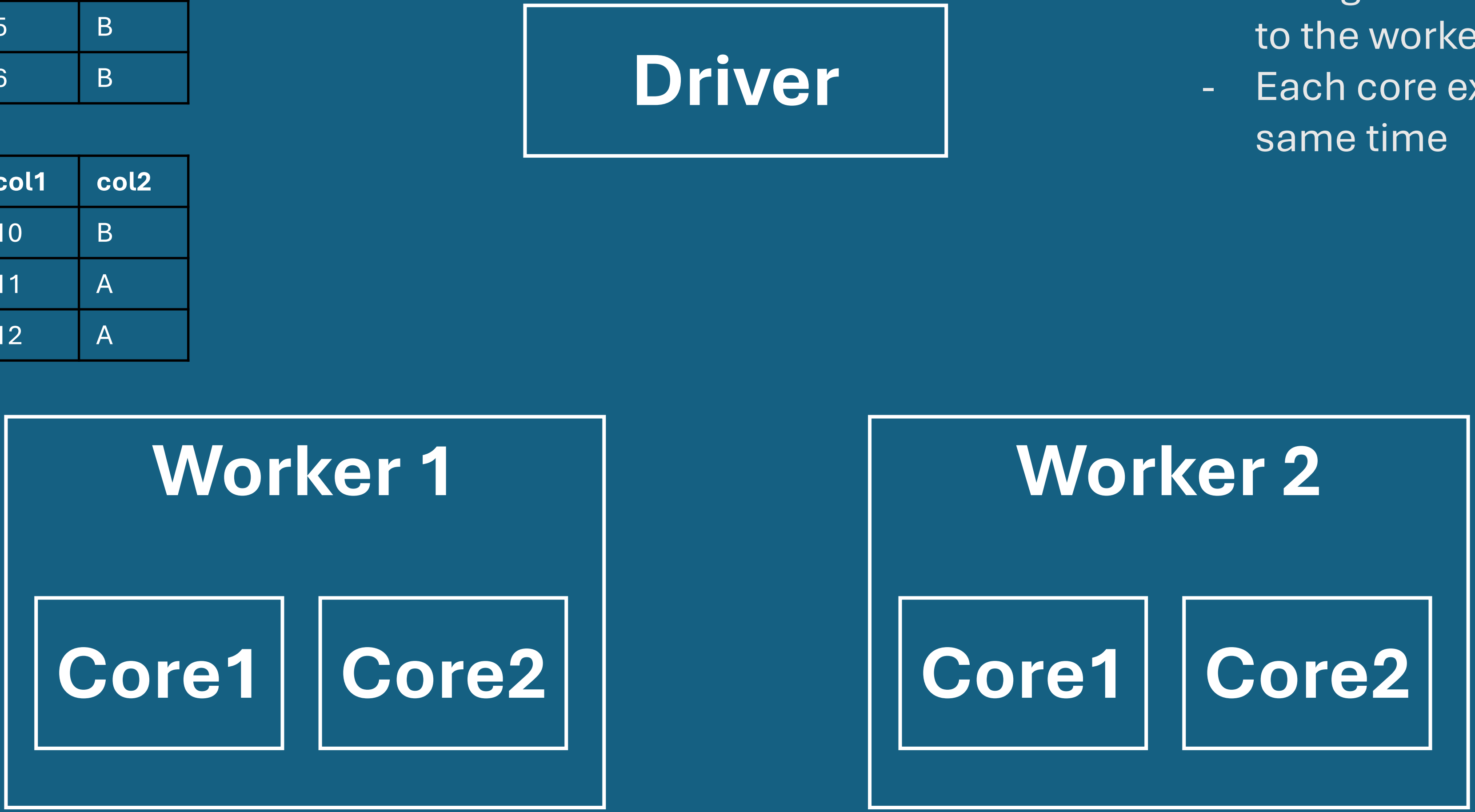
Spark Architecture

col1	col2
1	A
2	B
3	A

col1	col2
4	B
5	B
6	B

col1	col2
7	A
8	A
9	B

col1	col2
10	B
11	A
12	A



Driver

Worker 1

Core1

Core2

Worker 2

Core1

Core2

- Assigns files to partitions
- Delegates the partitions as tasks to the worker
- Each core executes one task at the same time

Reference Architecture Basics: https://youtu.be/kCydZHkqXc0

What determines the number of partitions?

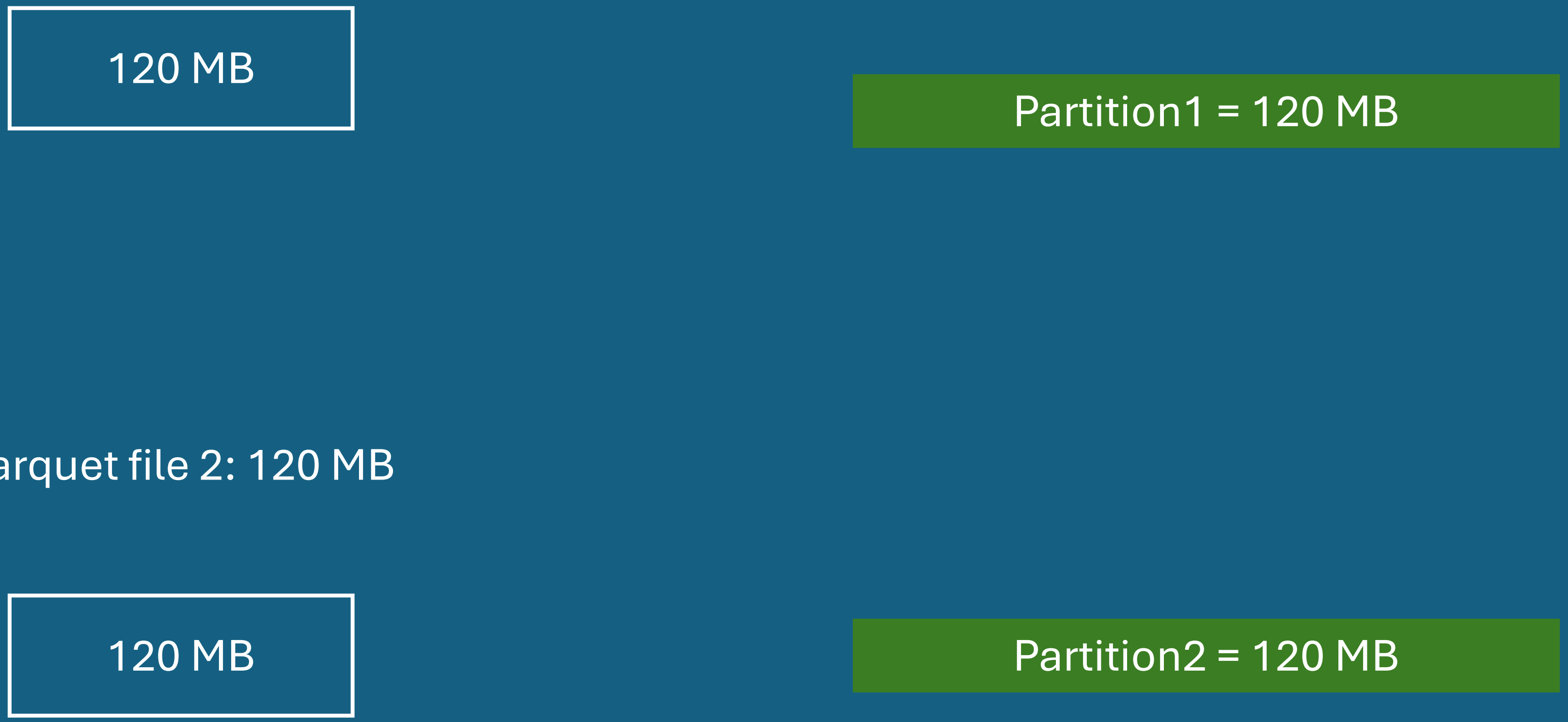
- Num of cores:
  - Spark tries to create at least the number of partitions equal to your number of cores
  - Can be changed with conf `spark.sql.files.minPartitionNum`
- Num of parquet files and its row groups: Parquet is only splitable on Row group level for partitioning
- Max Partition size:
  - Default 128 MB as the default row group size
  - Can be changed with conf `spark.sql.files.maxPartitionBytes`
- Max Cost per Bytes:
  - Represents the cost of creating a new partition, defaulting to 4 MB
  - Can be changed with conf `spark.sql.files.openCostInBytes`

Reference Influence on Partitions: https://youtu.be/vkOxEHEKYhA

How files are assigned to Partitions

Parquet file 1: 120 MB

```
spark.sql.files.maxPartitionBytes = 128 MB
spark.sql.files.maxPartitionBytes = 100 MB
```



120 MB

Partition1 = 120 MB

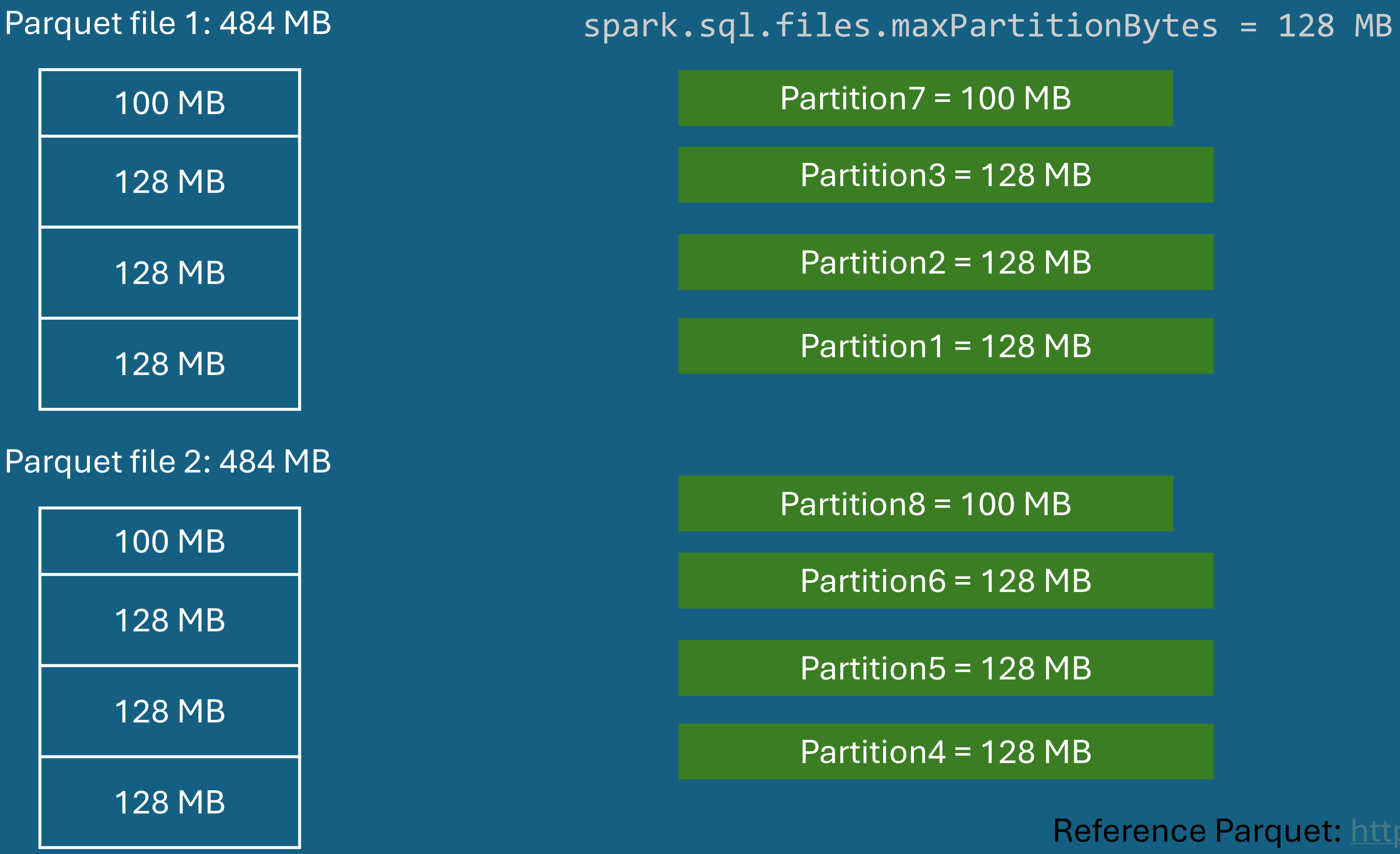
Parquet file 2: 120 MB

120 MB

Partition2 = 120 MB

Reference MaxPartitions Bytes: https://youtu.be/Inr0vH9EsEY

How files are assigned to Partitions



Parquet file 1: 484 MB

- 100 MB
- 128 MB
- 128 MB
- 128 MB

Parquet file 2: 484 MB

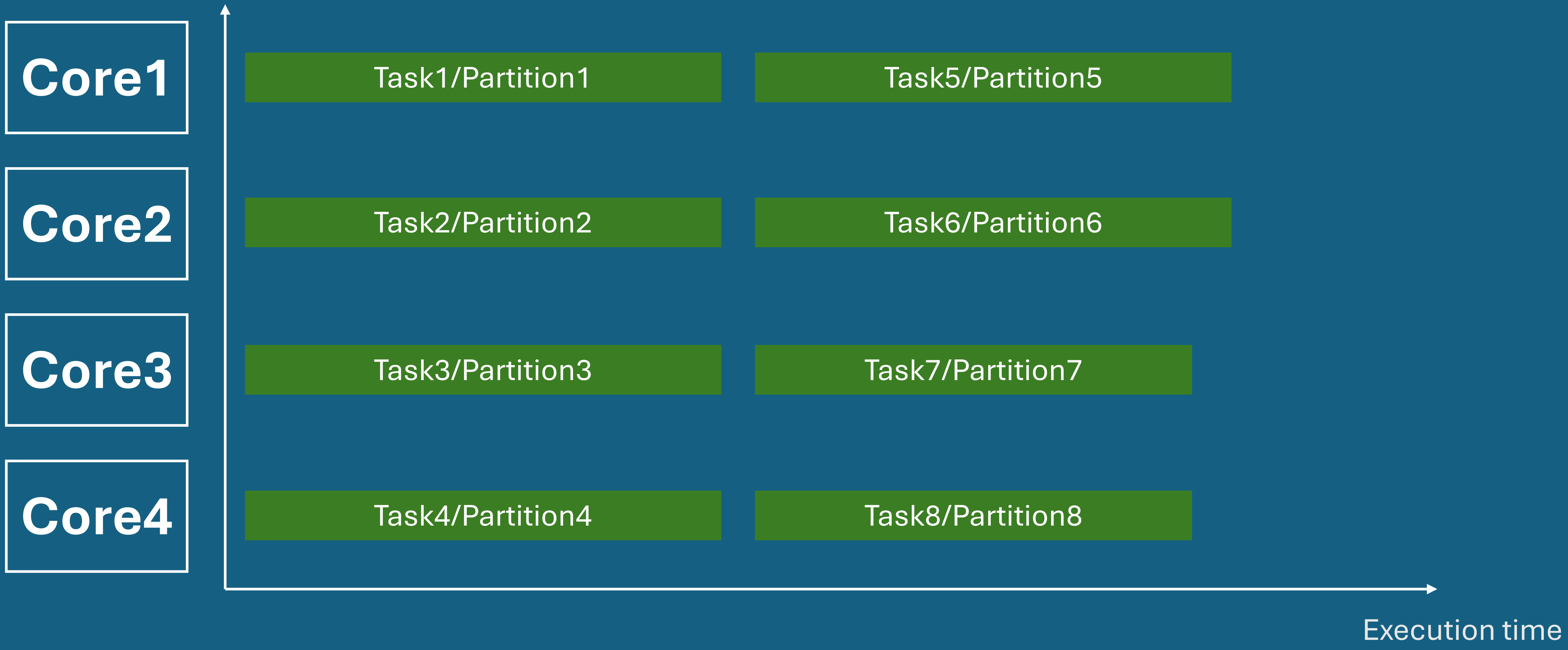
- 100 MB
- 128 MB
- 128 MB
- 128 MB

```
spark.sql.files.maxPartitionBytes = 128 MB
```

- Partition7 = 100 MB
- Partition3 = 128 MB
- Partition2 = 128 MB
- Partition1 = 128 MB
- Partition8 = 100 MB
- Partition6 = 128 MB
- Partition5 = 128 MB
- Partition4 = 128 MB

Reference Parquet: https://youtu.be/Lq6OnSakDrg

Perfect distributions of Partitions



Core1

Task1/Partition1

Task5/Partition5

Core2

Task2/Partition2

Task6/Partition6

Core3

Task3/Partition3

Task7/Partition7

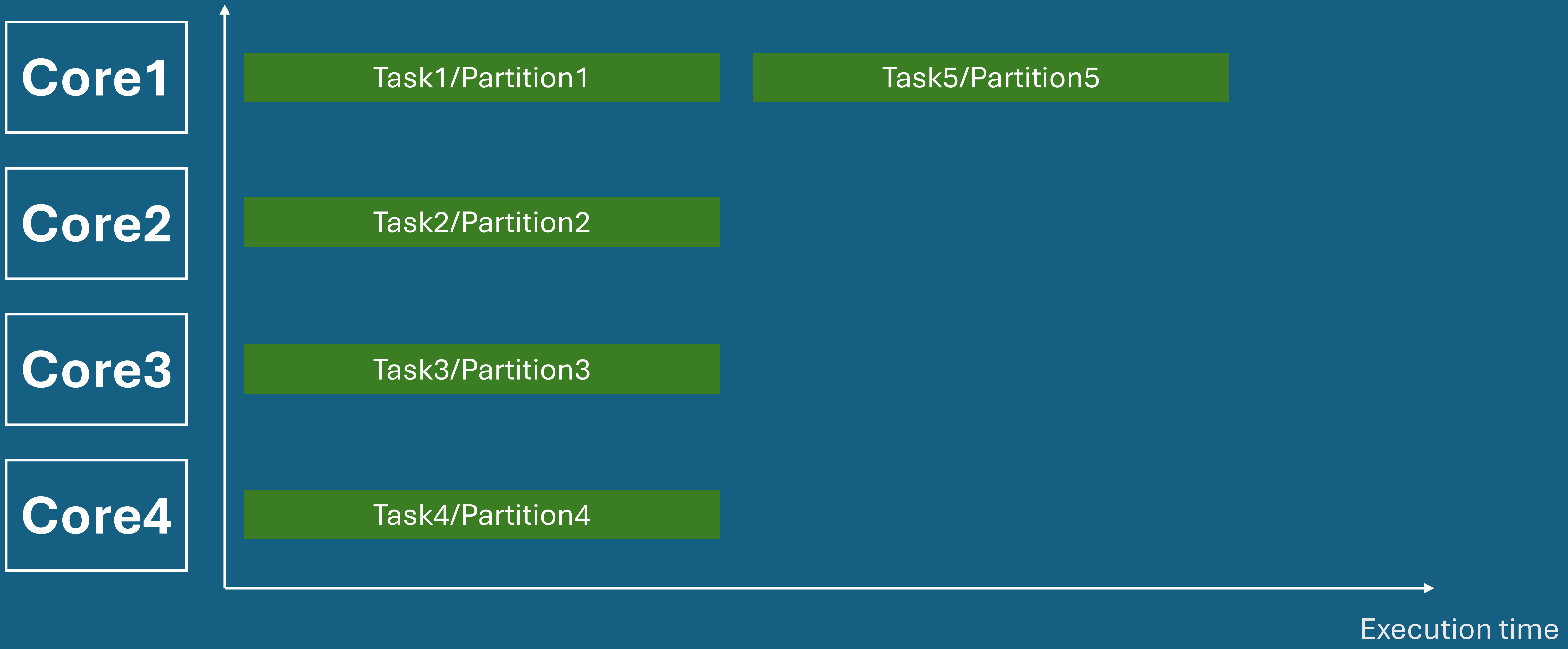
Core4

Task4/Partition4

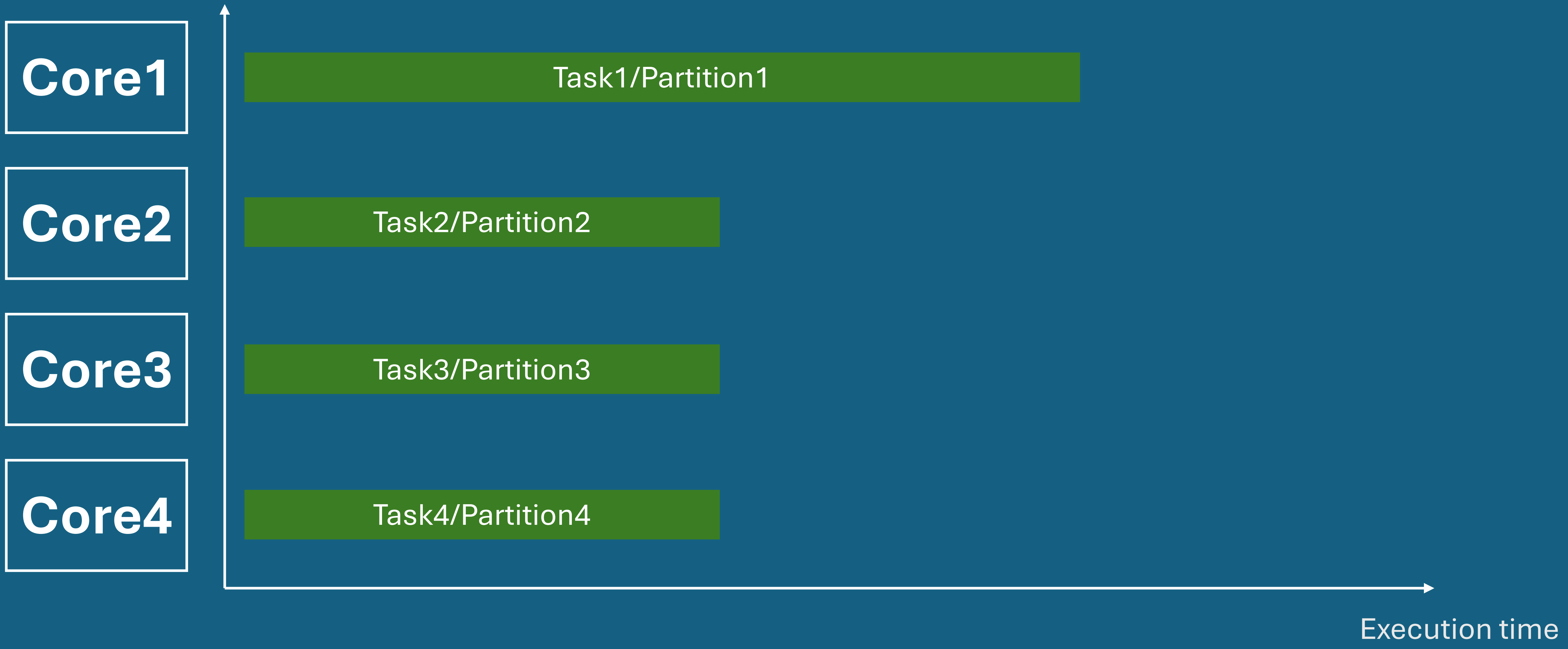
Task8/Partition8

Execution time

Bad example


Core1
Core2
Core3
Core4
Task1/Partition1
Task5/Partition5
Task2/Partition2
Task3/Partition3
Task4/Partition4
Execution time

Bad example


Core1
Core2
Core3
Core4
Task1/Partition1
Task2/Partition2
Task3/Partition3
Task4/Partition4
Execution time

Basic rules of good partitions

- Good parallelisation:
  - Factor 2-4 of your number of cores (exceptions for smaller files)
  - Uniform datasets generate also uniform partitions
- Partition size:
  - To big partitions can lead to out of memory issues
  - Max partition size is at 128 MB, 100 MB to 1 GB is recommended
  - It depends of course on your machine and your other operations
- Distribution overhead:
  - A high number of partitions can create a distribution overhead
  - Execution time should make 90 % of the whole execution time
  - Exception: Small file problem where the distribution overhead is ok

Partition behaviour: https://youtu.be/QrHtWvPwgS8

Summary

- We saw how you can use to check the number of partitions `sdf_parquet.rdd.getNumPartitions()`before execution
- We saw how the number of files, file size and row groups influence the partitions
- We saw how we can improve performance and partition distribution with conf `spark.sql.files.maxPartitionBytes`