Stage & Task Decomposition

Objective

Understand how Spark splits jobs into computational chunks

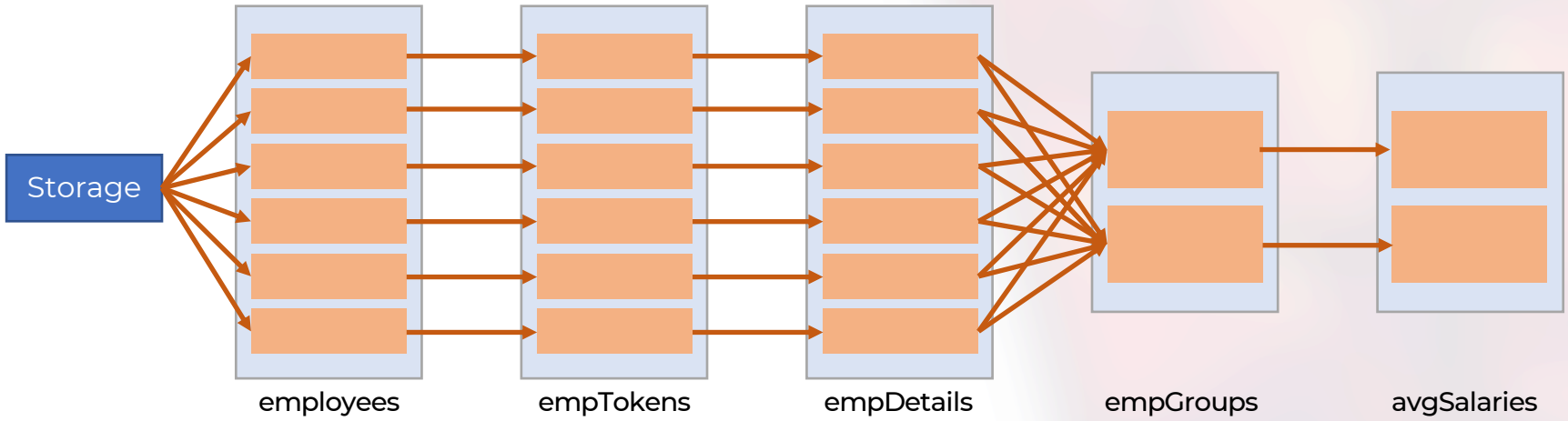
Make the distinction between narrow and wide transformations

Understand shuffles

Stages & Tasks

Example: compute average salaries of employees by department

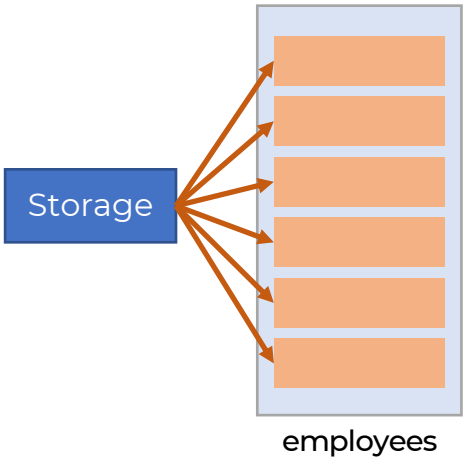
```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
val empDetails = empTokens.map(tokens => (tokens(4), tokens(7)))
val empGroups = empDetails.groupByKey(2)
val avgSalaries = empGroups.mapValues(salaries => salaries.map(_.toInt).sum / salaries.size)
```



Overview

Step 1: read the text file as a DF, into 6 partitions

```
val employees = sc.textFile("/tmp/employees.csv")
```

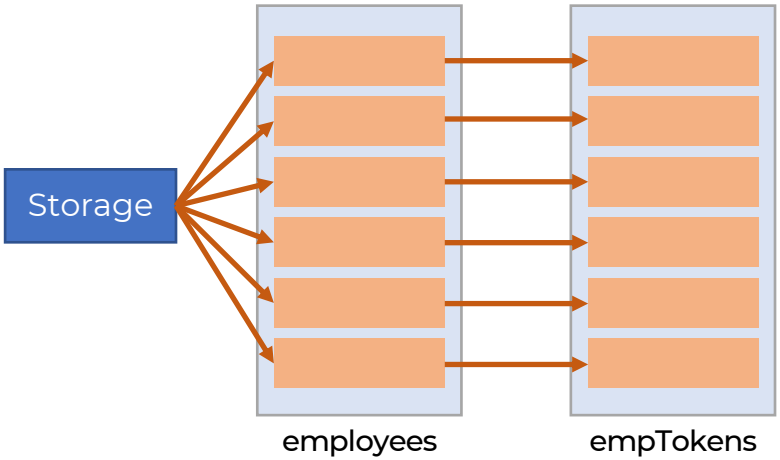


Overview

Step 2: split the records into lines

```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
```

Narrow transformation: partitions don't need to know about each other

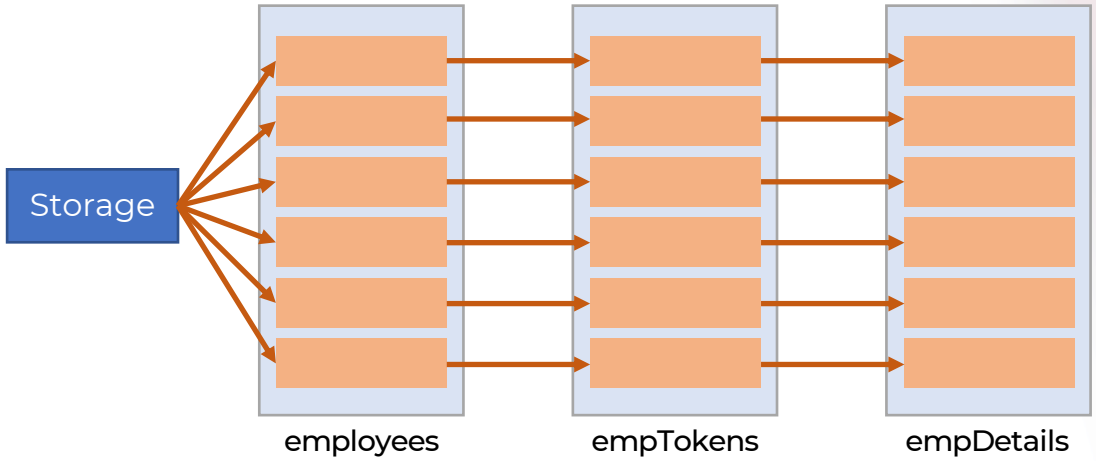


Stages & Tasks

Step 3: tuple the relevant information

```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
val empDetails = empTokens.map(tokens => (tokens(4), tokens(7)))
```

maps are narrow transformations

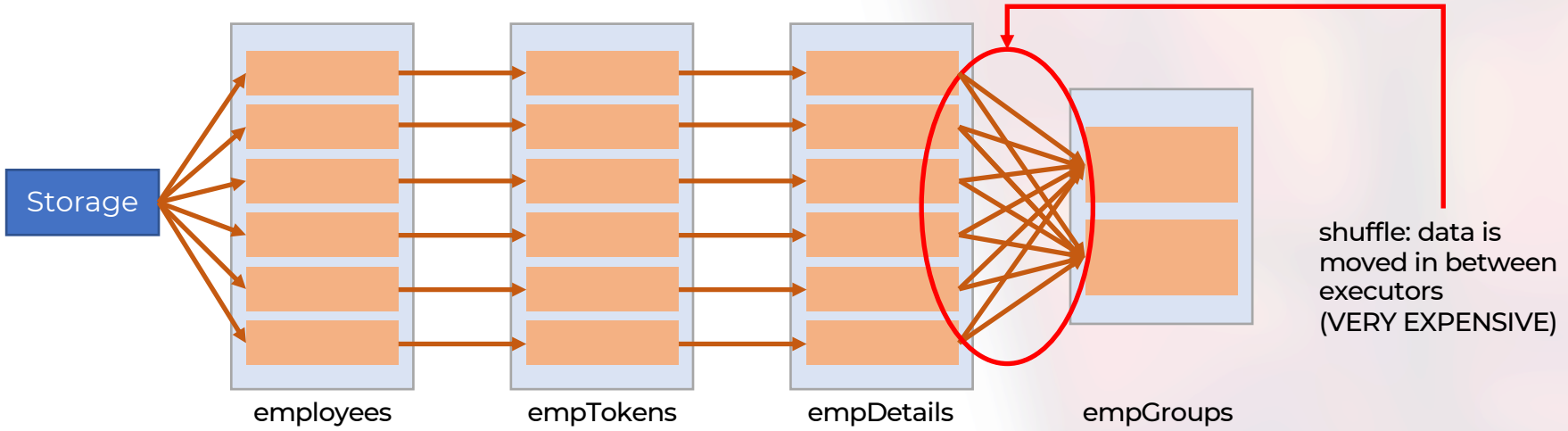


Stages & Tasks

Step 4: group the data by department

```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
val empDetails = empTokens.map(tokens => (tokens(4), tokens(7)))
val empGroups = empDetails.groupByKey(2)
```

wide transformation: all partitions need to be considered

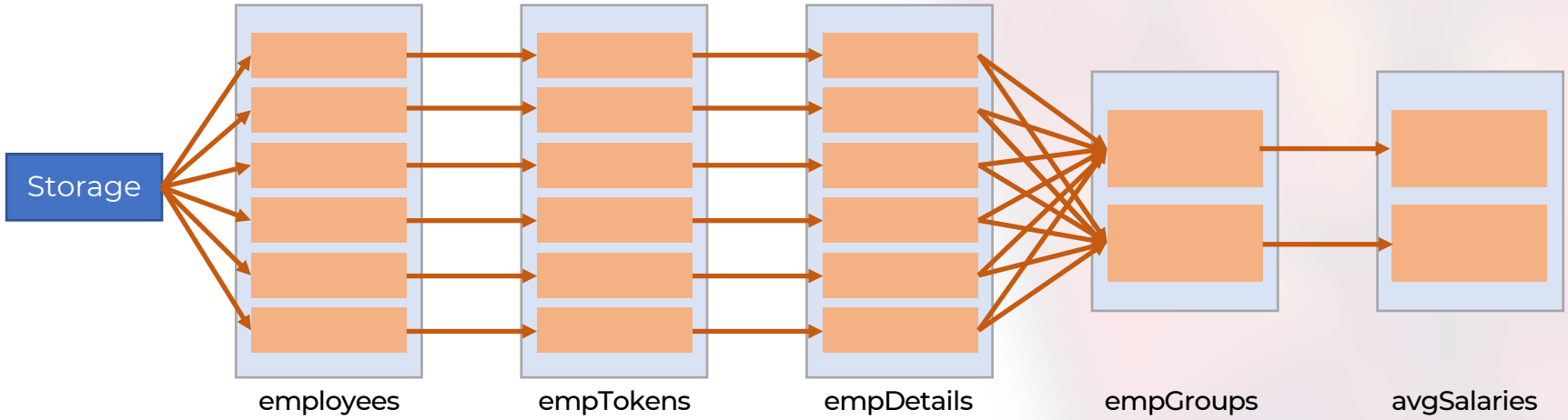


Stages & Tasks

Step 5: average the values in each group

```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
val empDetails = empTokens.map(tokens => (tokens(4), tokens(7)))
val empGroups = empDetails.groupByKey(2)
val avgSalaries = empGroups.mapValues(salaries => salaries.map(_.toInt).sum / salaries.size)
```

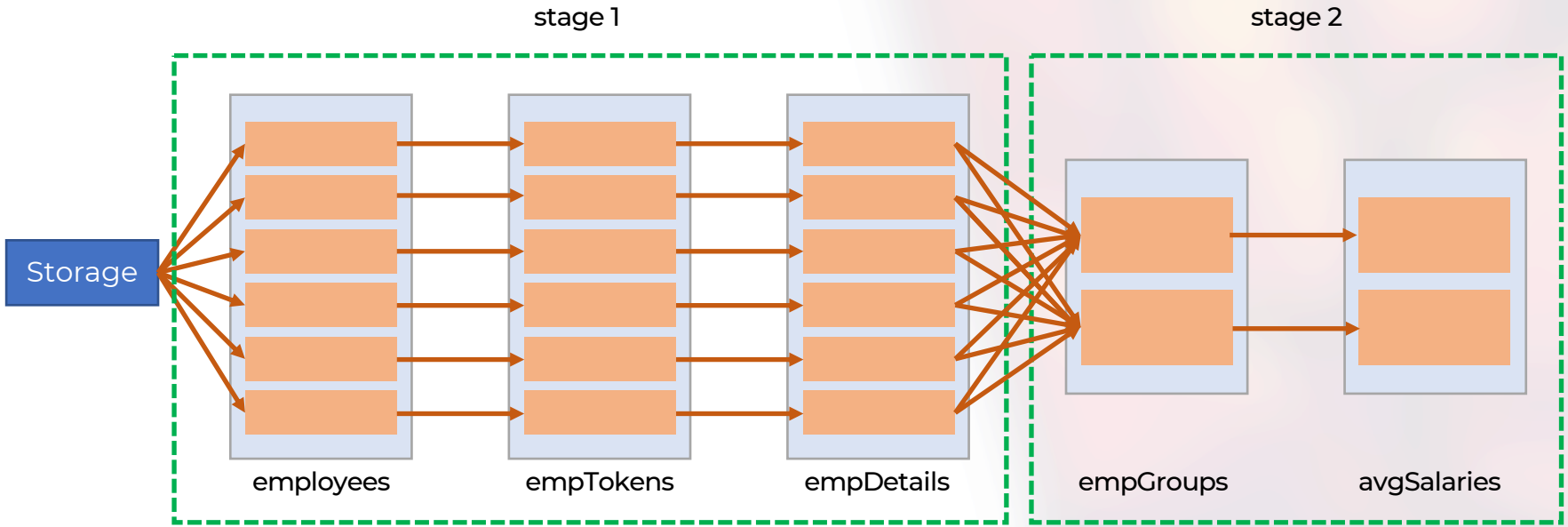
narrow transformation



Stages & Tasks

```
val employees = sc.textFile("/tmp/employees.csv")
val empTokens = employees.map(line => line.split(","))
val empDetails = empTokens.map(tokens => (tokens(4), tokens(7)))

val empGroups = empDetails.groupByKey(2)
val avgSalaries = empGroups.mapValues(salaries => salaries.map(_.toInt).sum / salaries.size)
```



Stages & Tasks

Task

- the smallest unit of computation
- executed once, for one partition, by one executor

Stage

- contains tasks
- enforces no exchange of data = no partitions need data from other partitions
- depends on the previous stage = previous stage must complete before this one starts

Shuffle

- exchange of data between executors
- happens in between stages
- must complete before next stage starts

An application contains jobs

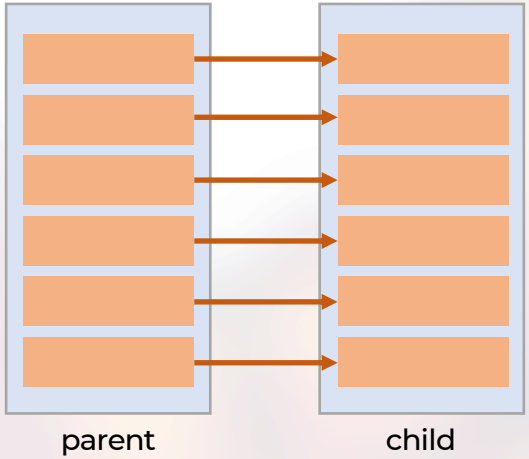
A job contains stages

A stage contains tasks

Dependencies

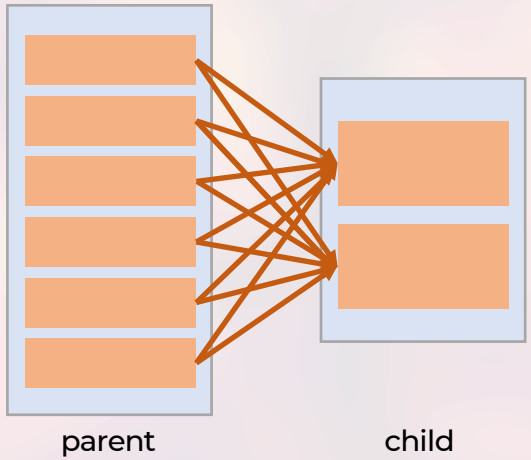
Narrow dependencies

- one input (parent) partition influences a single output (child) partition
- fast to compute
- examples: map, flatMap, filter, projections



Wide dependencies

- one input partition influences more than one output partitions
- involve a shuffle = data transfer between Spark executors
- are costly to compute
- examples: grouping, joining, sorting

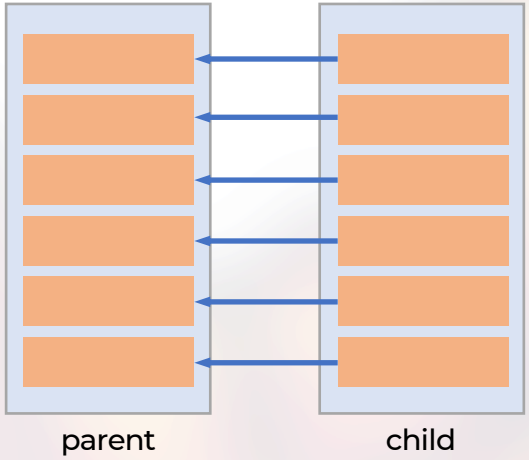


Dependencies

Expressed differently in terms of "depends on":

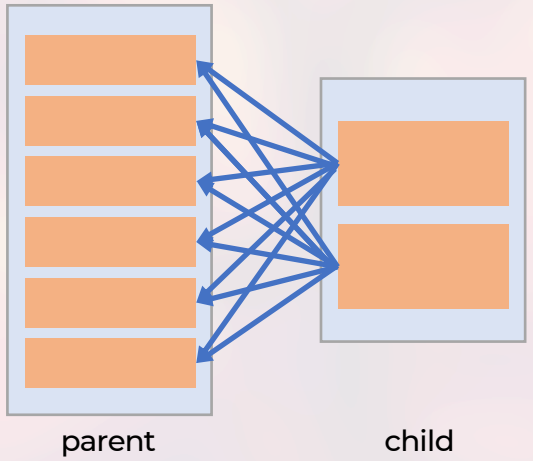
Narrow dependencies

- given a parent partition, a single child partition depends on it
- fast to compute
- examples: map, flatMap, filter, projections



Wide dependencies

- given a parent partition, more than one child partitions depend on it
- involve a shuffle = data transfer between Spark executors
- are costly to compute
- examples: grouping, joining, sorting



Shuffles

Data exchanges between executors in the cluster

Expensive because of

- transferring data
- serialization/deserialization
- loading new data from shuffle files

Shuffles are performance bottlenecks because

- exchanging data takes time
- they need to be fully completed before next computations start

Shuffles limit parallelization

Spark rocks