Yandex

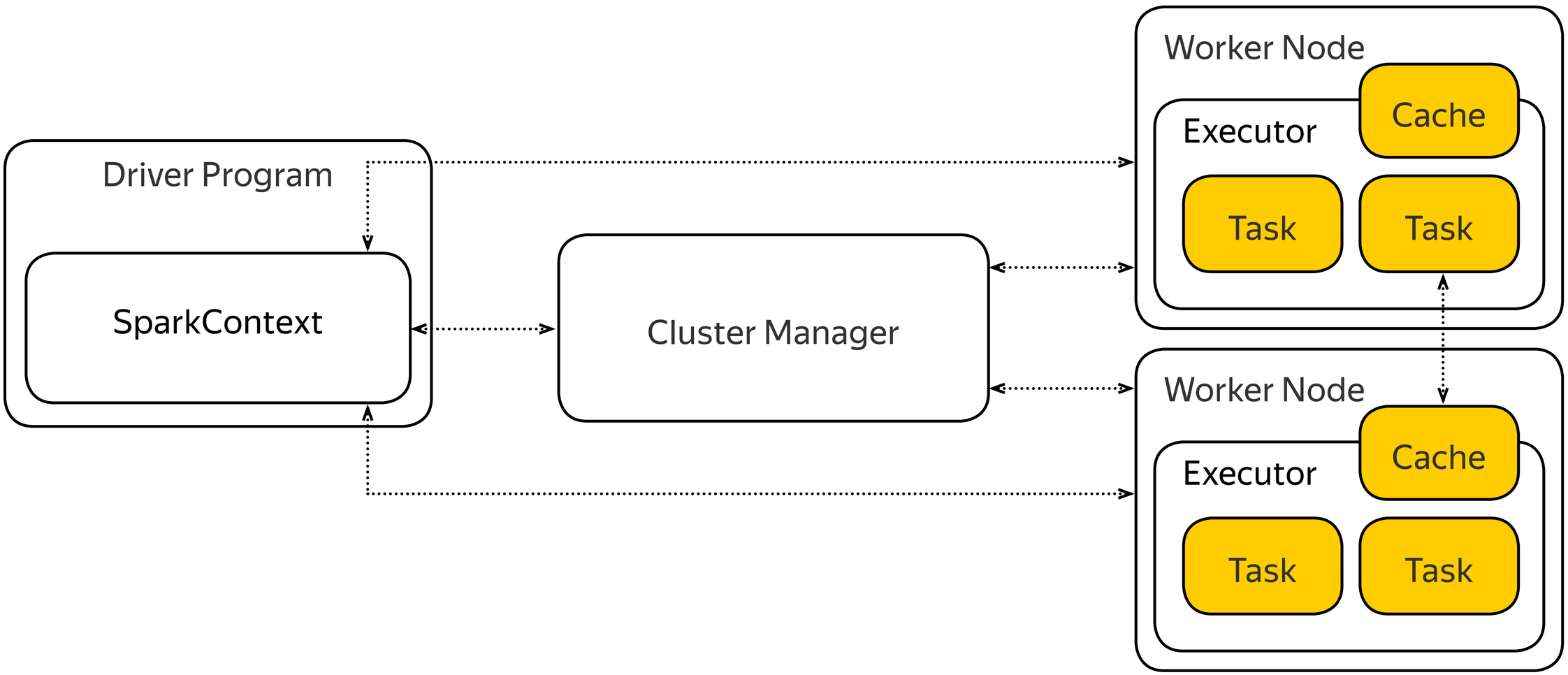
Execution & scheduling

SparkContext

- Tells your application how to access a cluster
- Coordinates processes on the cluster to run your application

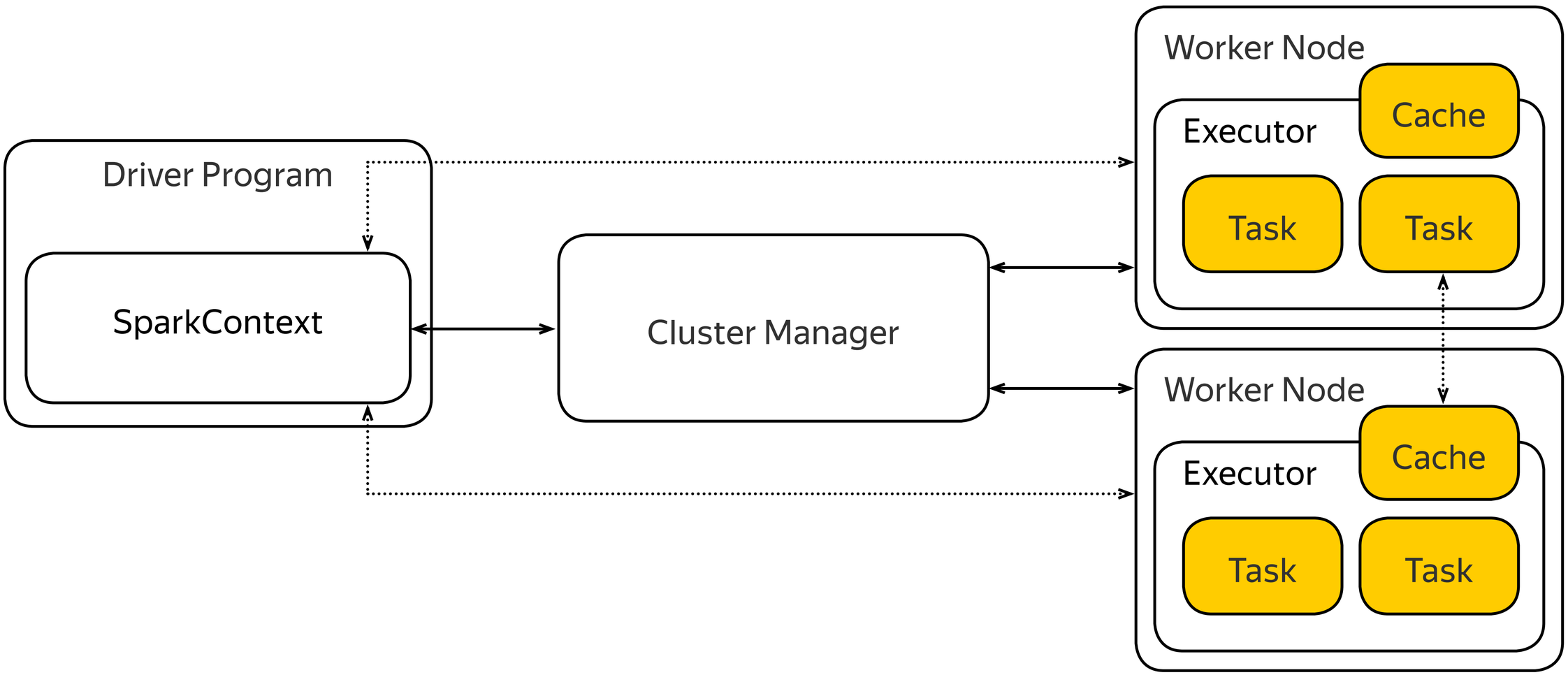
SparkContext

› Tells your application how to access a cluster
› Coordinates processes on the cluster to run your application



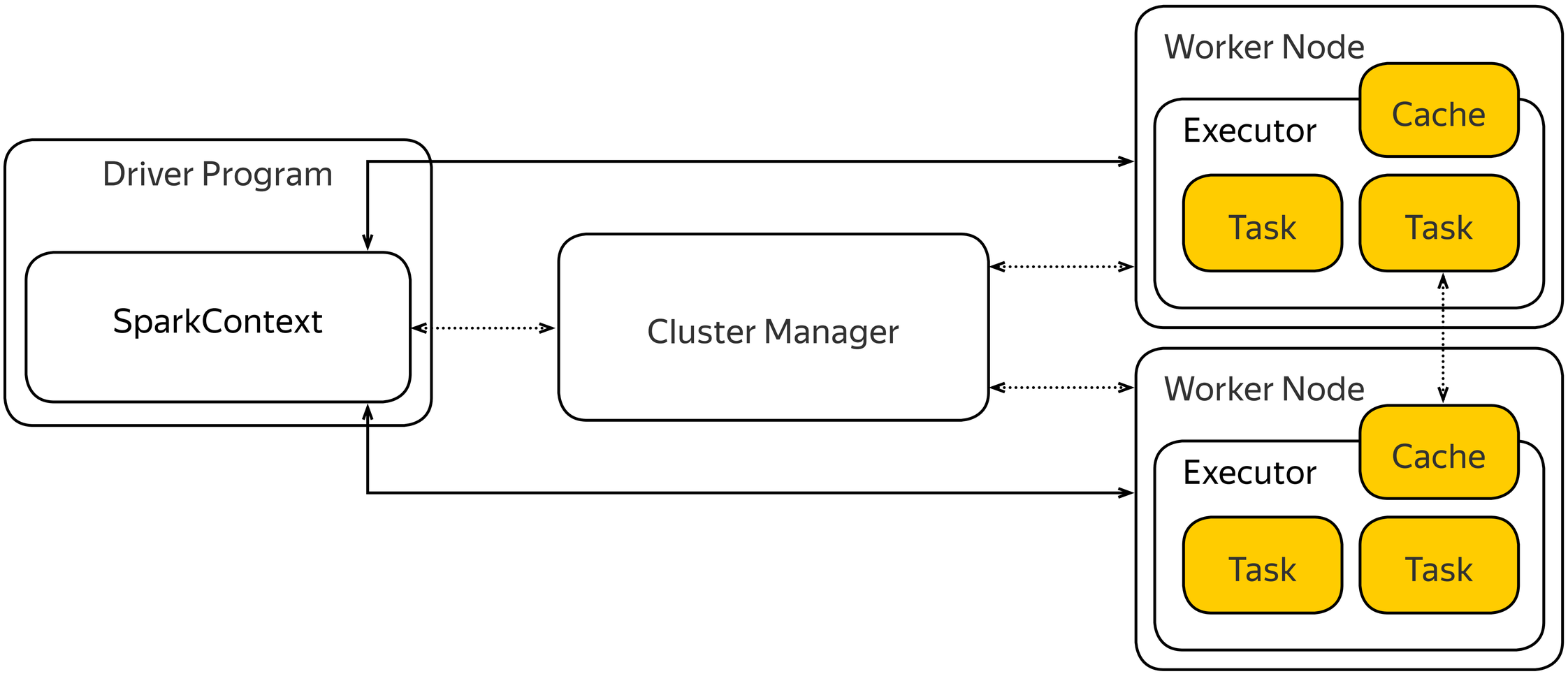
SparkContext

› Tells your application how to access a cluster
› Coordinates processes on the cluster to run your application



SparkContext

› Tells your application how to access a cluster
› Coordinates processes on the cluster to run your application



Jobs, stages, tasks

- Task is a unit of work to be done
- Tasks are created by a job scheduler for every job stage
- Job is spawned in response to a Spark action
- Job is divided in smaller sets of tasks called stages

Jobs, stages, tasks (example)

› Z = X
.map(lambda x: (x % 10, x / 10))
.reduceByKey(lambda x, y: x + y)

Jobs, stages, tasks (example)

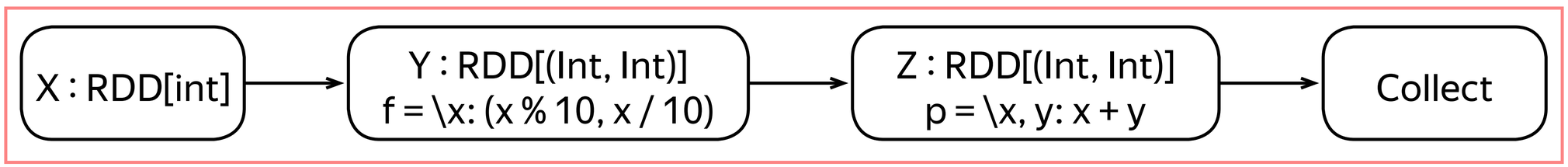
```
› Z = X
    .map(lambda x: (x % 10, x / 10))
  .reduceByKey(lambda x, y: x + y)
  .collect()
```

1. Invoking an action...

Jobs, stages, tasks (example)

› Z = X
  .map(lambda x: (x % 10, x / 10))
  .reduceByKey(lambda x, y: x + y)
  .collect()

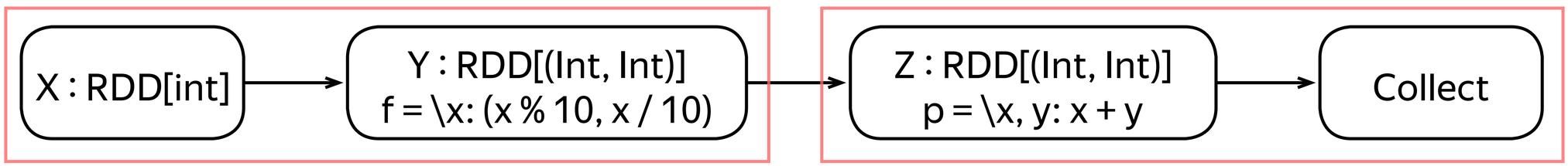
1. Invoking an action...
2. ...spawns the job...



Jobs, stages, tasks (example)

› Z = X
  .map(lambda x: (x % 10, x / 10))
  .reduceByKey(lambda x, y: x + y)
  .collect()

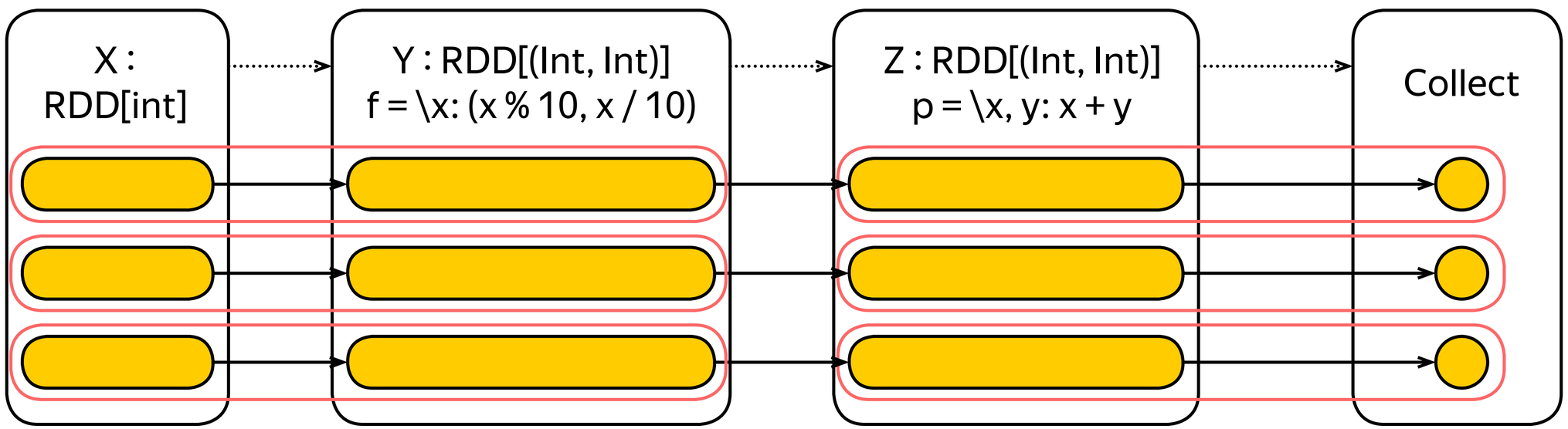
1. Invoking an action...
2. ...spawns the job...
3. ...that gets divided into
   the stages by the job scheduler...



Jobs, stages, tasks (example)

```
› Z = X
    .map(lambda x: (x % 10, x / 10))
    .reduceByKey(lambda x, y: x + y)
    .collect()
```

1. Invoking an action...
2. ...spawns the job...
3. ...that gets divided into the stages by the job scheduler...
4. ...and tasks are created for every job stage.



Jobs, stages, tasks

- Job stage is a pipelined computation spanning between materialization boundaries
  - not immediately executable
- Task is a job stage bound to particular partitions
  - immediately executable

Jobs, stages, tasks

- Job stage is a pipelined computation spanning between materialization boundaries
- Task is a job stage bound to particular partitions
- Materialization happens when reading, shuffling or passing data to an action
  - narrow dependencies allow pipelining
  - wide dependencies forbid it

SparkContext – other functions

- Tracks liveness of the executors
  - required to provide fault-tolerance
- Schedules multiple concurrent jobs
  - to control the resource allocation within the application
- Performs dynamic resource allocation
  - to control the resource allocation between different applications

Summary

- The SparkContext is the core of your application
- The driver communicates directly with the executors
- Execution goes as follows:
  Action ➞ Job ➞ Job Stages ➞ Tasks
- Transformations with narrow dependencies allow pipelining

BigDATAteam