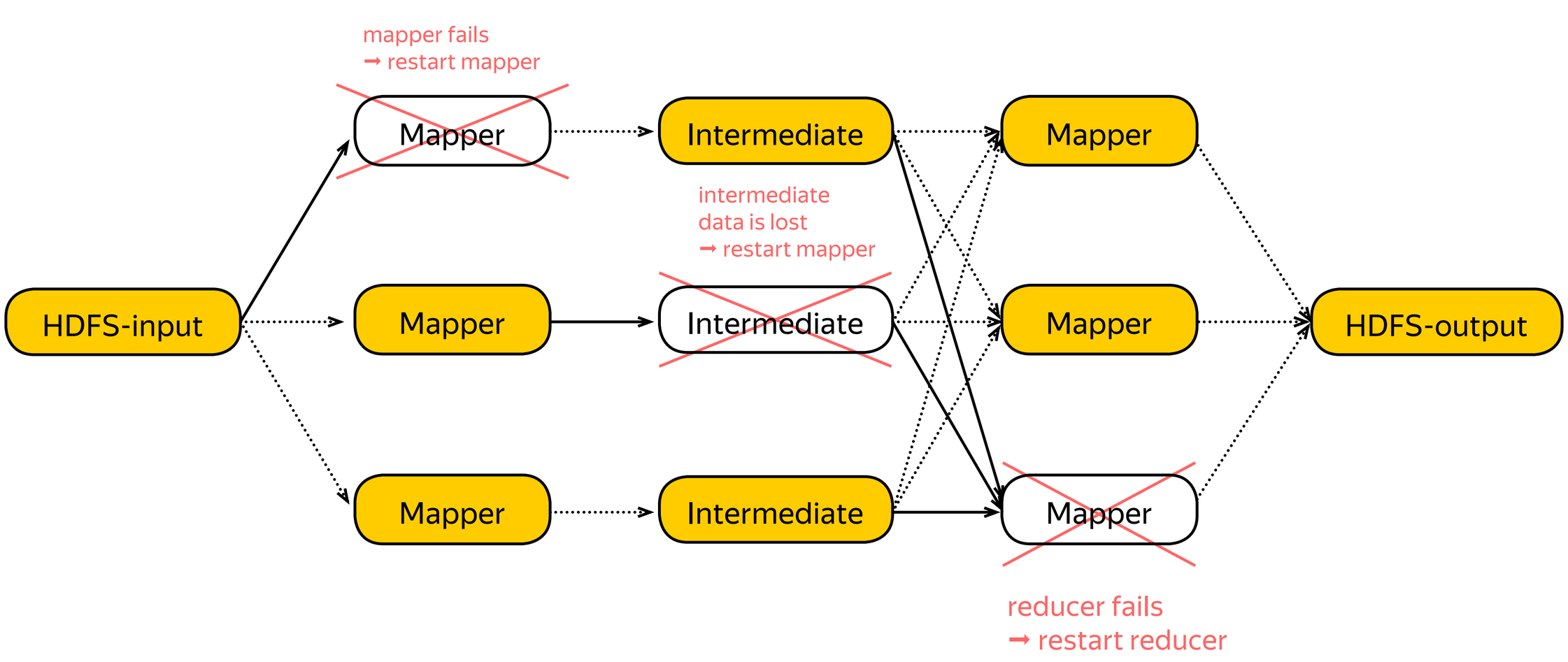
Yandex

Resiliency

Fault-tolerance in MapReduce

- Two key aspects
  - reliable storage for input and output data
  - deterministic and side-effect free execution of mappers and reducers



Fault-tolerance in Spark

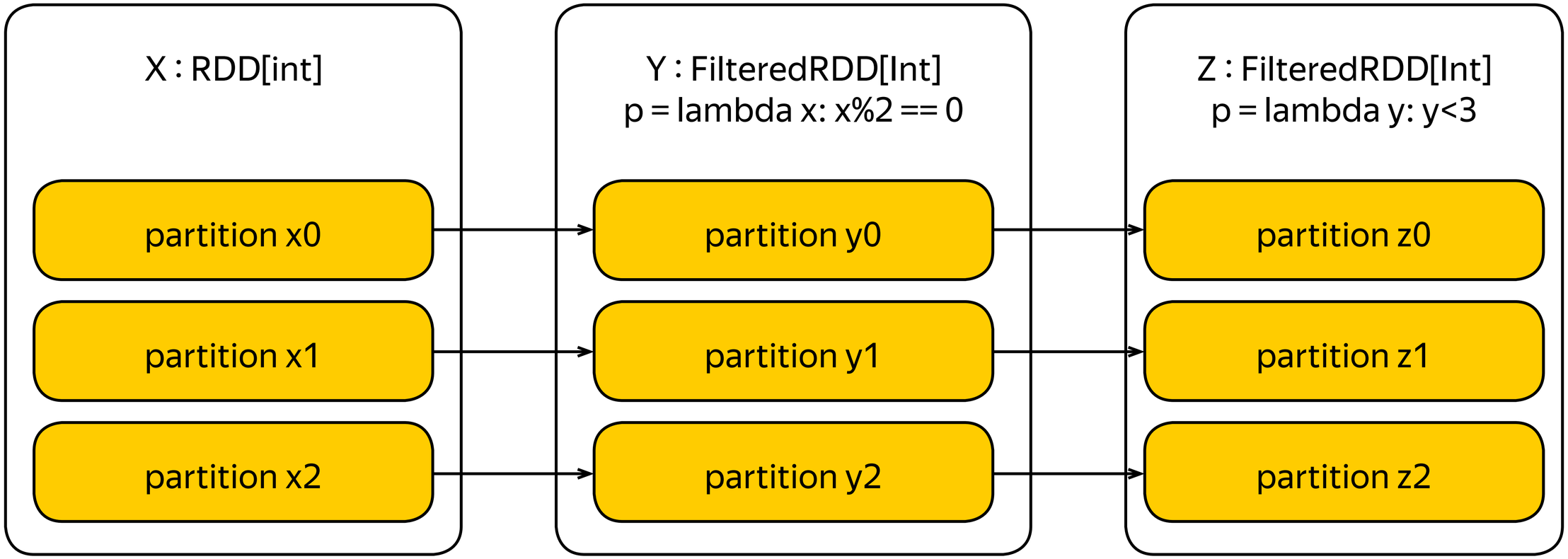
- Same two key aspects
  - reliable storage for input and output data
  - deterministic and side-effect free execution of transformations(including closures)

- Determinism — every invocation of the function results in the same returned value
  - e. g. do not use random numbers, do not depend on a hash value order

- Freedom of side-effects — an invocation of the function does not change anything in the external world
  - e. g. do not commit to a database, do not rely on global variables

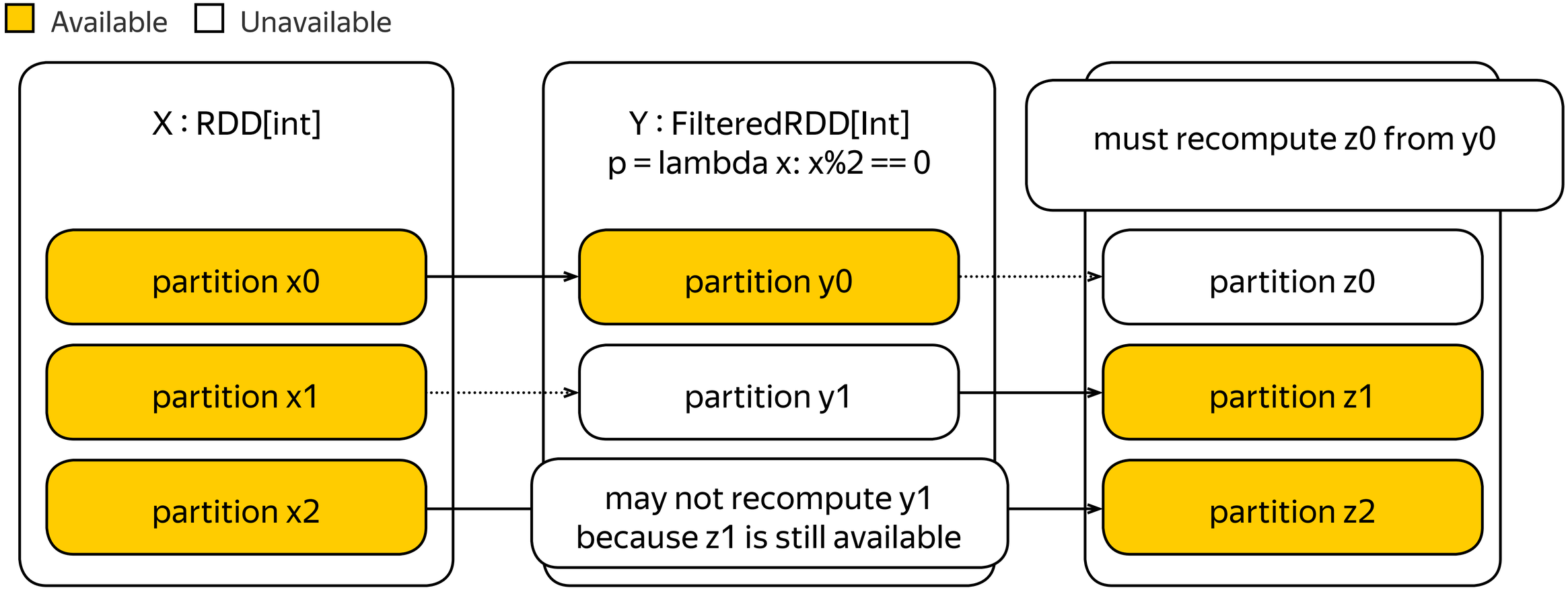
Fault-tolerance & transformations

› Lineage — a dependency graph for all partitions of all RDDs involved in a computation up to the data source



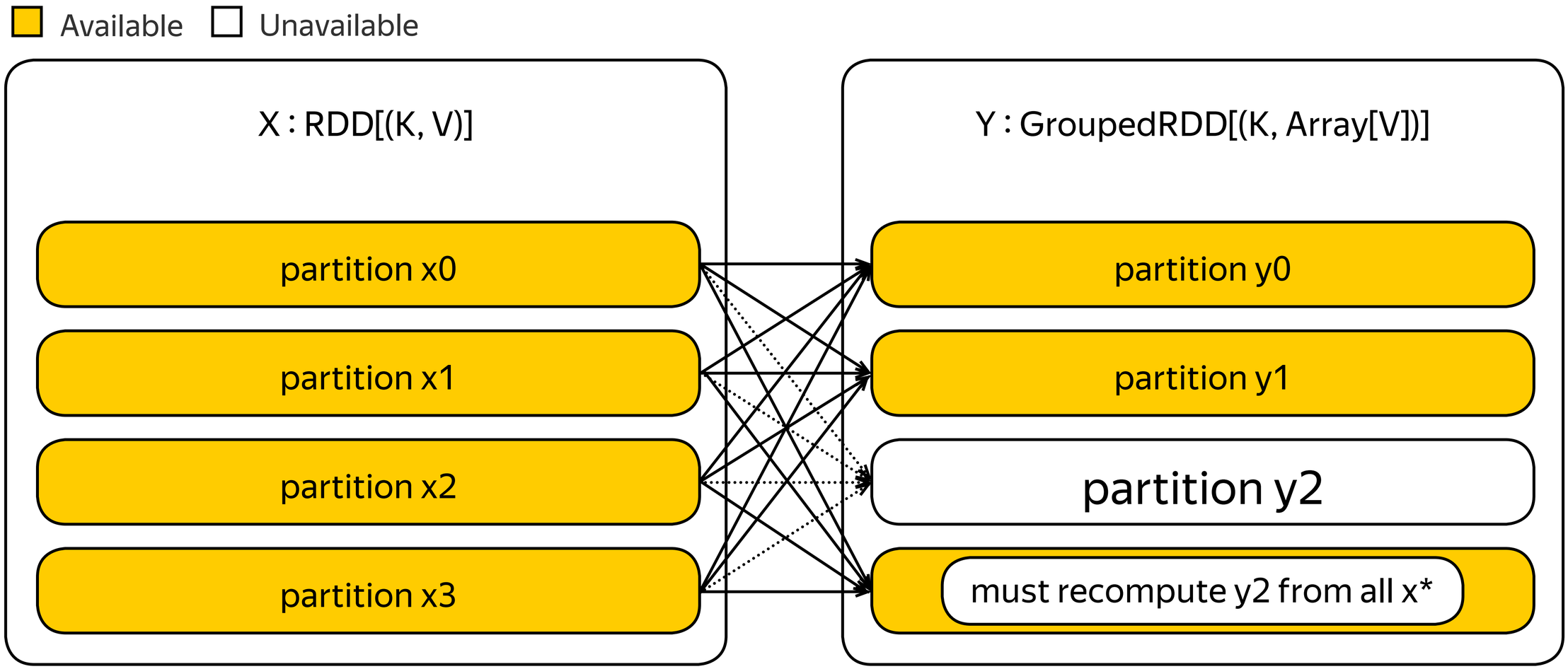
Fault-tolerance & transformations

› Lineage — a dependency graph for all partitions of all RDDs involved in a computation up to the data source



Fault-tolerance & transformations

› Lineage — a dependency graph for all partitions of all RDDs involved in a computation up to the data source



Fault-tolerance & actions

› Actions are side-effects in Spark

› Actions have to be idempotent that is safe to be re-executed multiple times given the same input

Fault-tolerance & actions

- Actions are side-effects in Spark
- Actions have to be idempotent that is safe to be re-executed multiple times given the same input
  - Example: collect()
    - the dataset is immutable; thus reading it multiple times is safe
  - Example: saveAsTextFile()
    - the dataset is immutable; thus file would be the same after every write

Quiz

Summary

- Resiliency is implemented by
  - tracking lineage
  - assuming deterministic & side-effect free execution of transformations(including closures)
  - assuming idempotency for actions

- May improve resiliency by increasing durability of RDDs
  - in the next lesson!

BigDATAteam