

Apache Spark

Lecture by: Faria Kalim (lead TA)

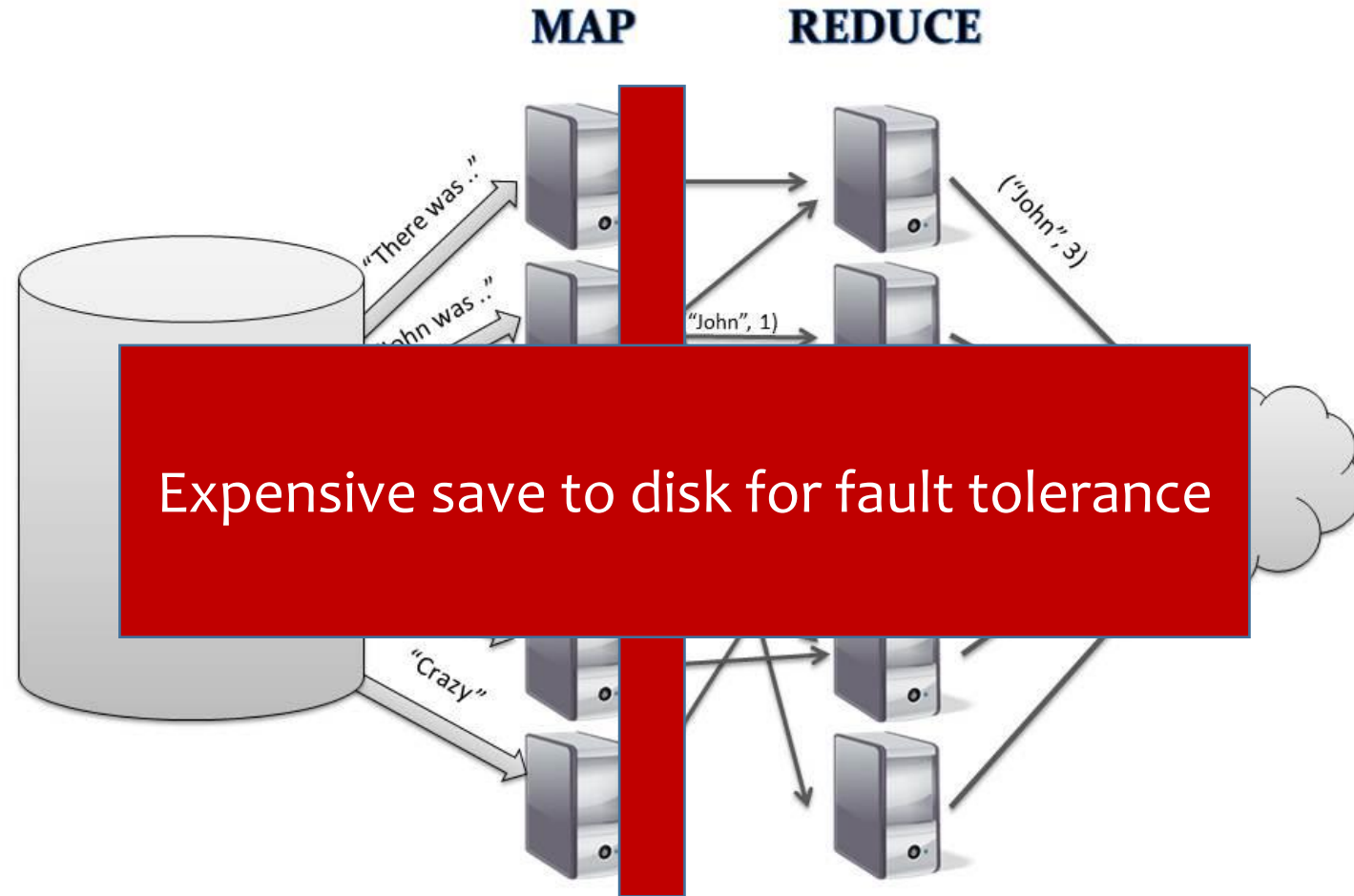
CS425

UIUC

(See Video)

Why Spark?

- Another system for big data analytics
- Isn't MapReduce good enough?
 - Simplifies batch processing on large commodity clusters

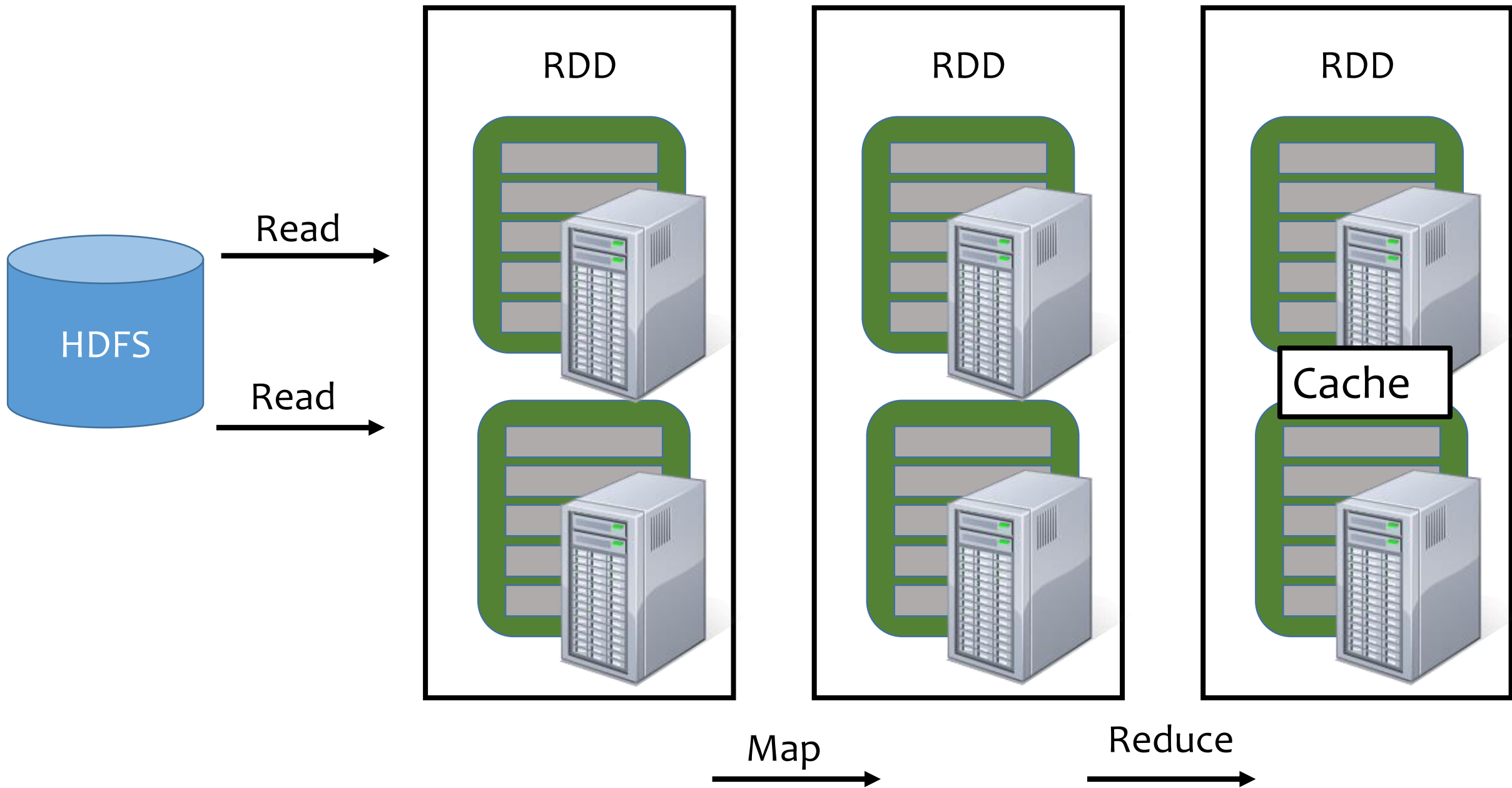


Why Spark?

- MapReduce can be expensive for some applications e.g.,
 - Iterative
 - Interactive
- Lacks efficient data sharing
- Specialized frameworks did evolve for different programming models
 - Bulk Synchronous Processing (Pregel)
 - Iterative MapReduce (Haloop)

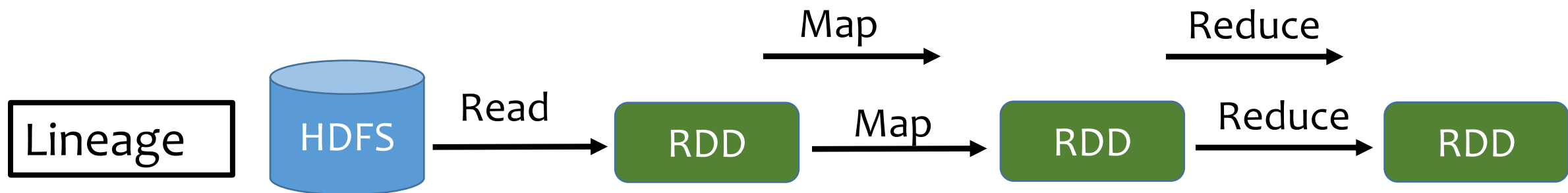
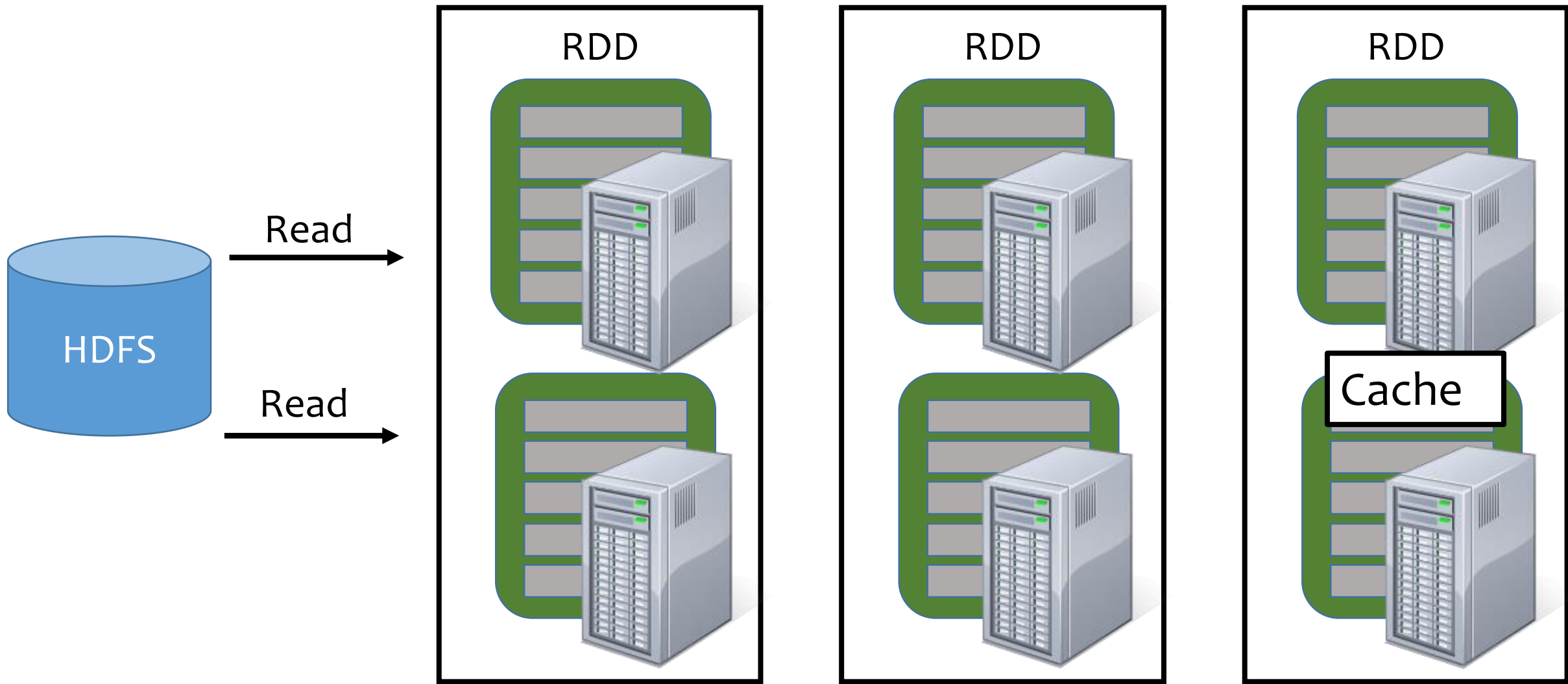
Solution: Resilient Distributed Datasets (RDDs)

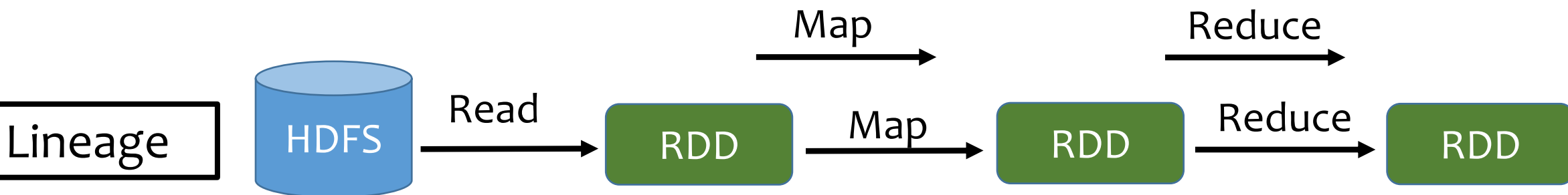
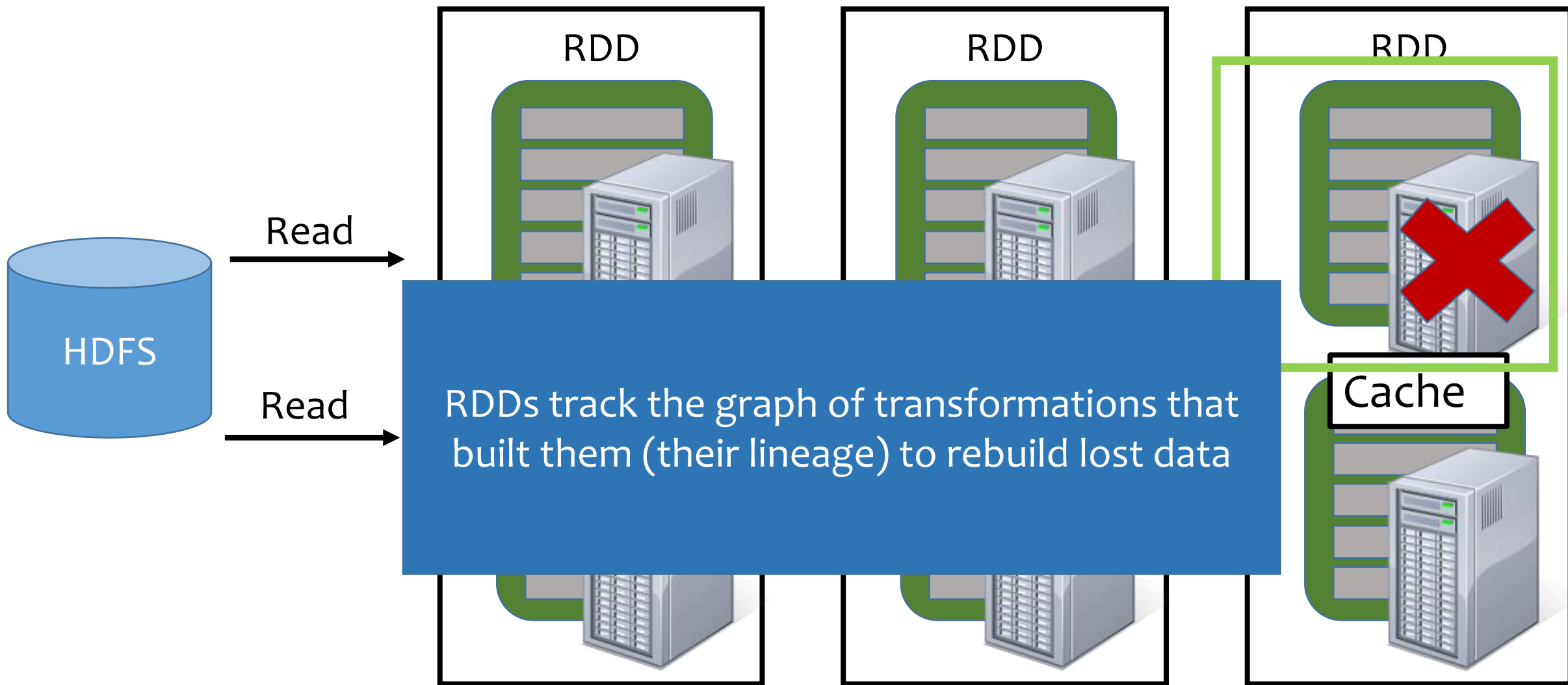
- RDDs
 - Immutable, partitioned collection of records
 - Built through coarse grained transformations (map, join ...)
 - Can be cached for efficient reuse



Solution: Resilient Distributed Datasets (RDDs)

- RDDs
 - Immutable, partitioned collection of records
 - Built through coarse grained, ordered transformations (map, join ...)
- Fault Recovery?
 - Lineage!
 - Log the coarse grained operation applied to a partitioned dataset
 - Simply recompute the lost partition if failure occurs!
 - No cost if no failure



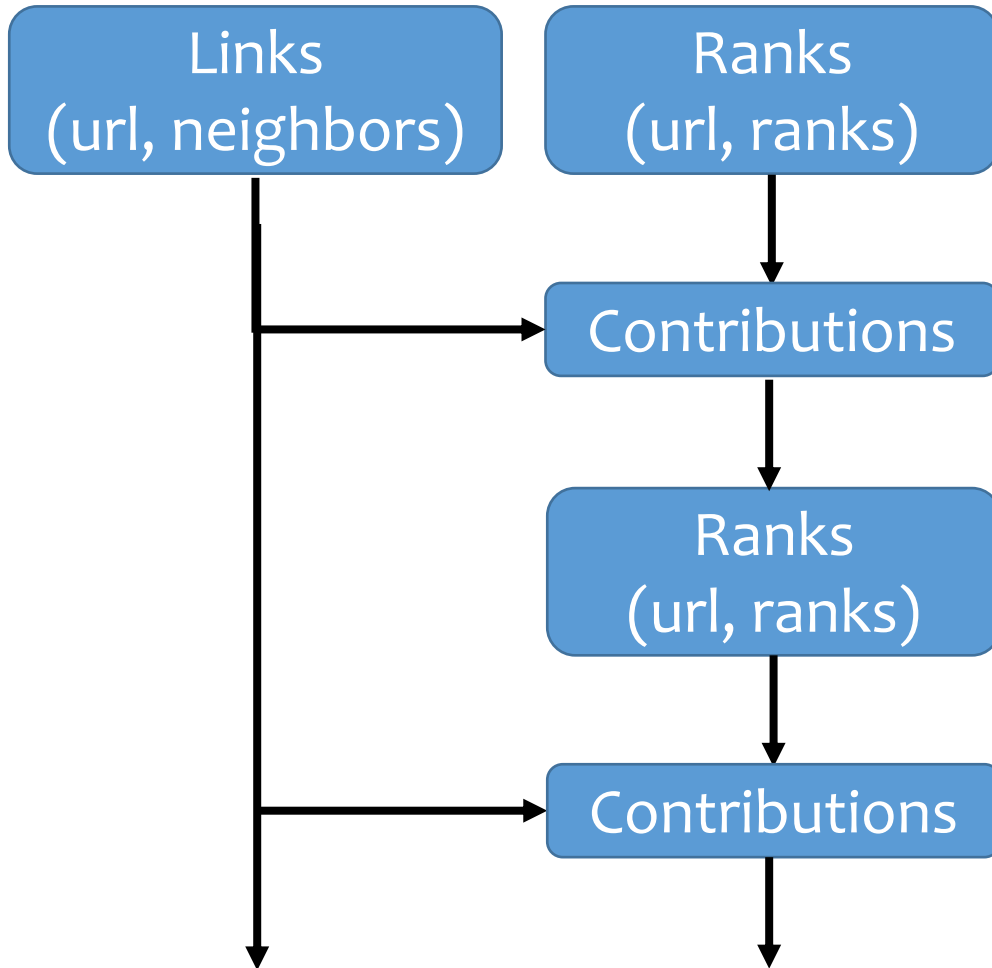


What can you do with Spark?

- RDD operations
 - Transformations e.g., filter, join, map, group-by ...
 - Actions e.g., count, print ...
- Control
 - Partitioning
 - Persistence

Partitioning

- PageRank

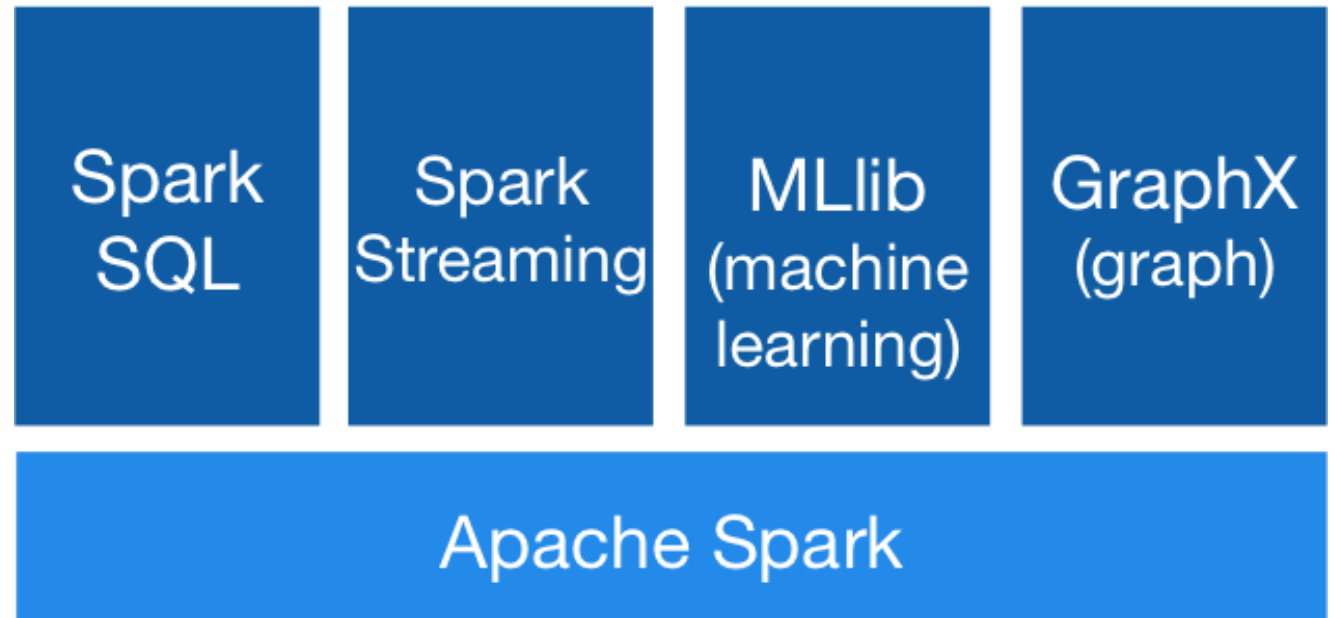


Joins take place repeatedly

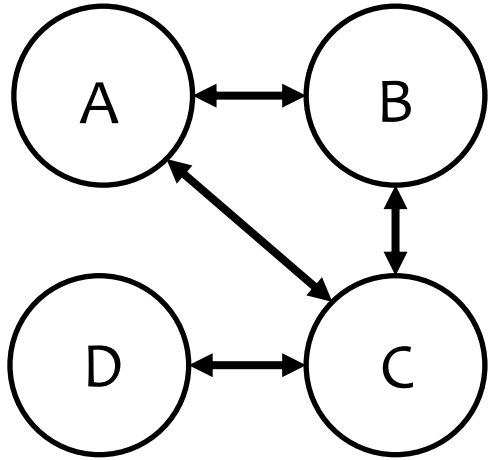
Good partitioning reduces shuffles

Generality

- RDDs allow unification of different programming models
 - Stream Processing
 - Graph Processing
 - Machine Learning

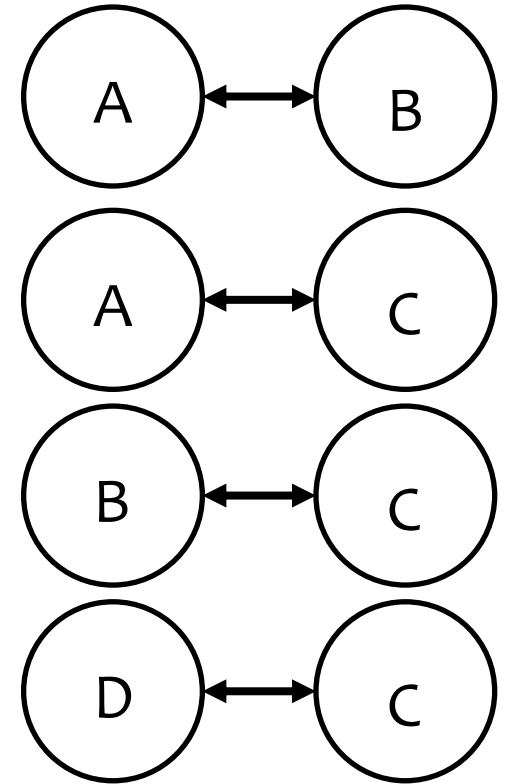
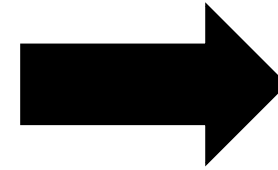


Gather-Apply-Scatter on GraphX



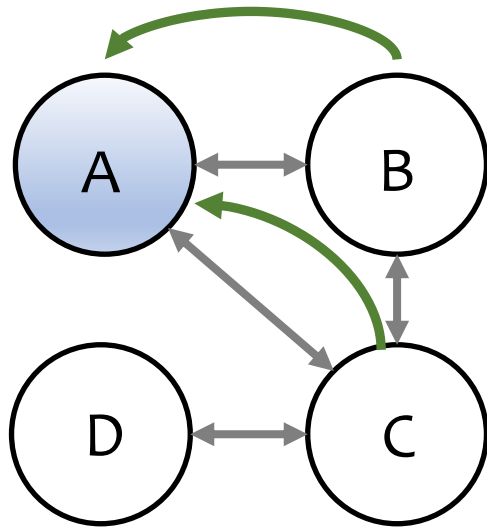
Vertices	Neighbors
A	B
A	C
B	C
D	C

Graph Represented In a Table

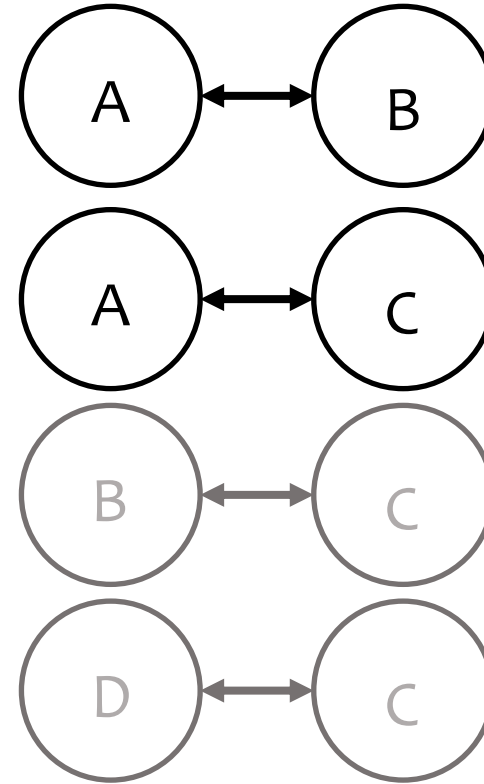


Triplets

Gather-Apply-Scatter on GraphX

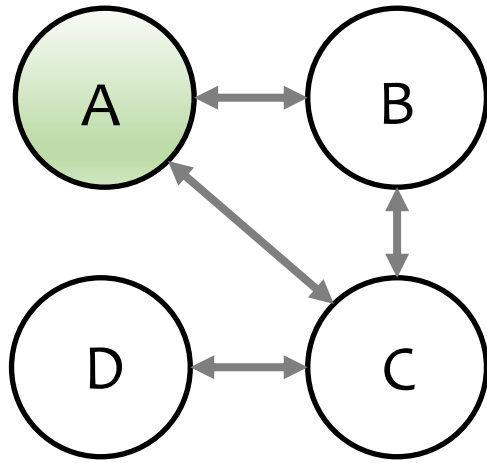


Gather at A

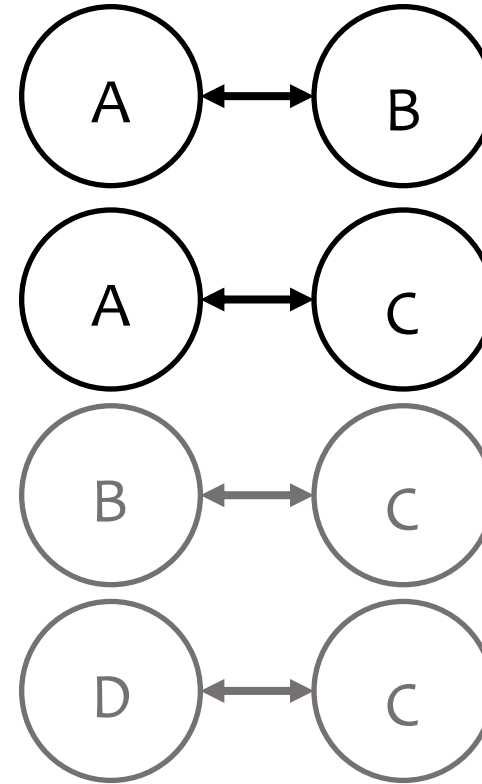


Group-By A

Gather-Apply-Scatter on GraphX

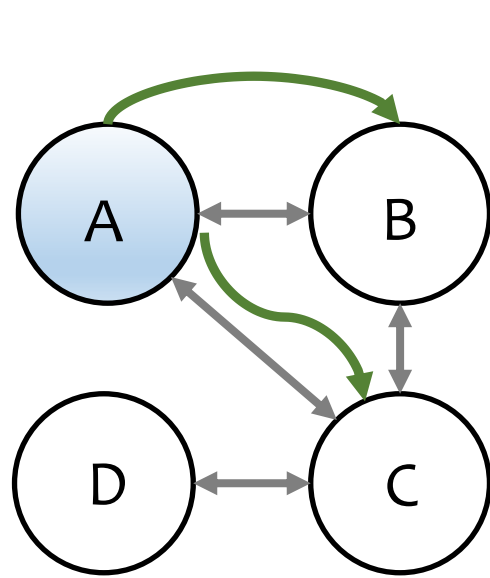


Apply

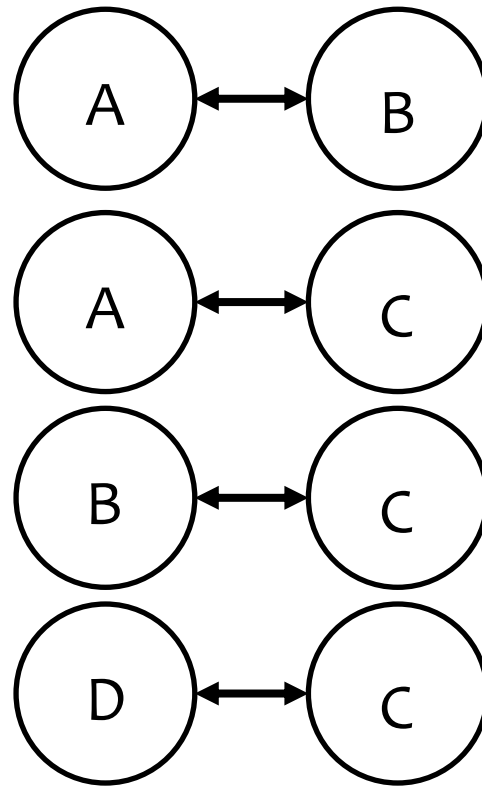


Map

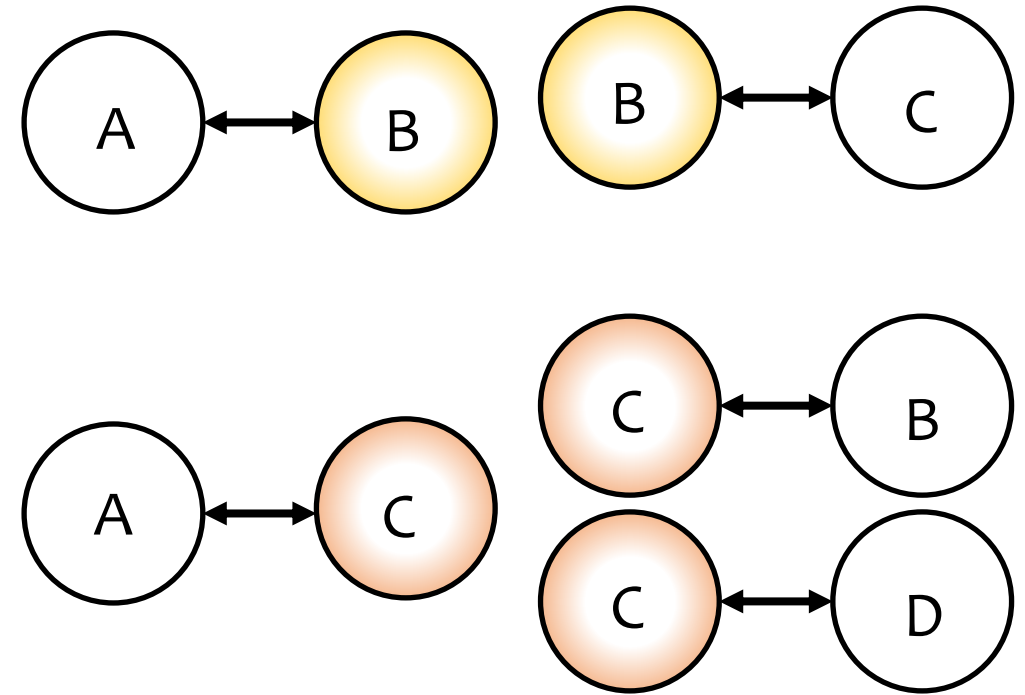
Gather-Apply-Scatter on GraphX



Scatter



Triplets



Join

Summary

- RDDs provide a simple and efficient programming model
- Generalized to a broad set of applications
- Leverages coarse-grained nature of parallel algorithms for failure recovery