Streaming Systems

Spark & Discretized Streams

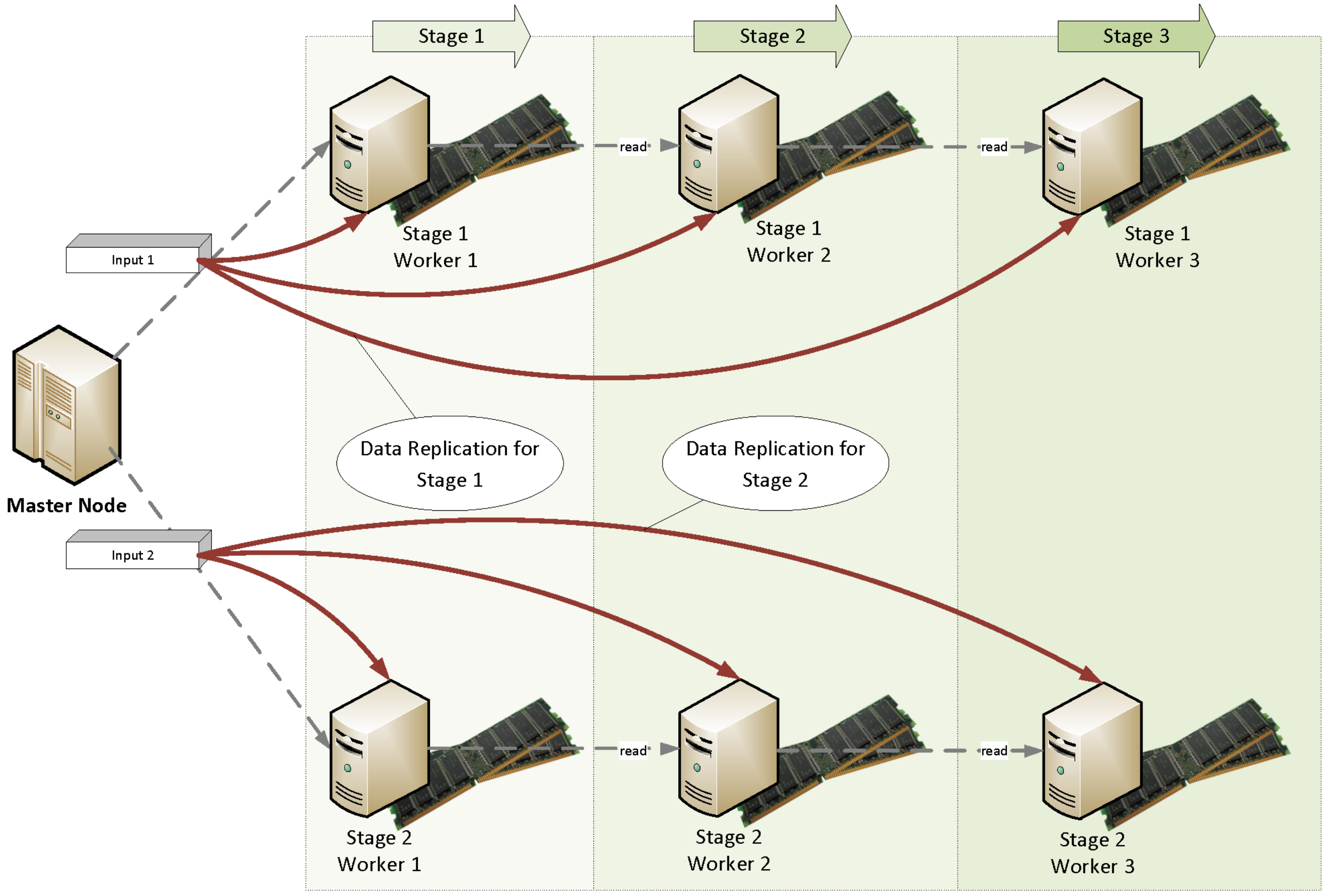
An introduction to Spark

How does Spark‘s pipelining approach differs from MapReduce?

Spark

How does Spark work?

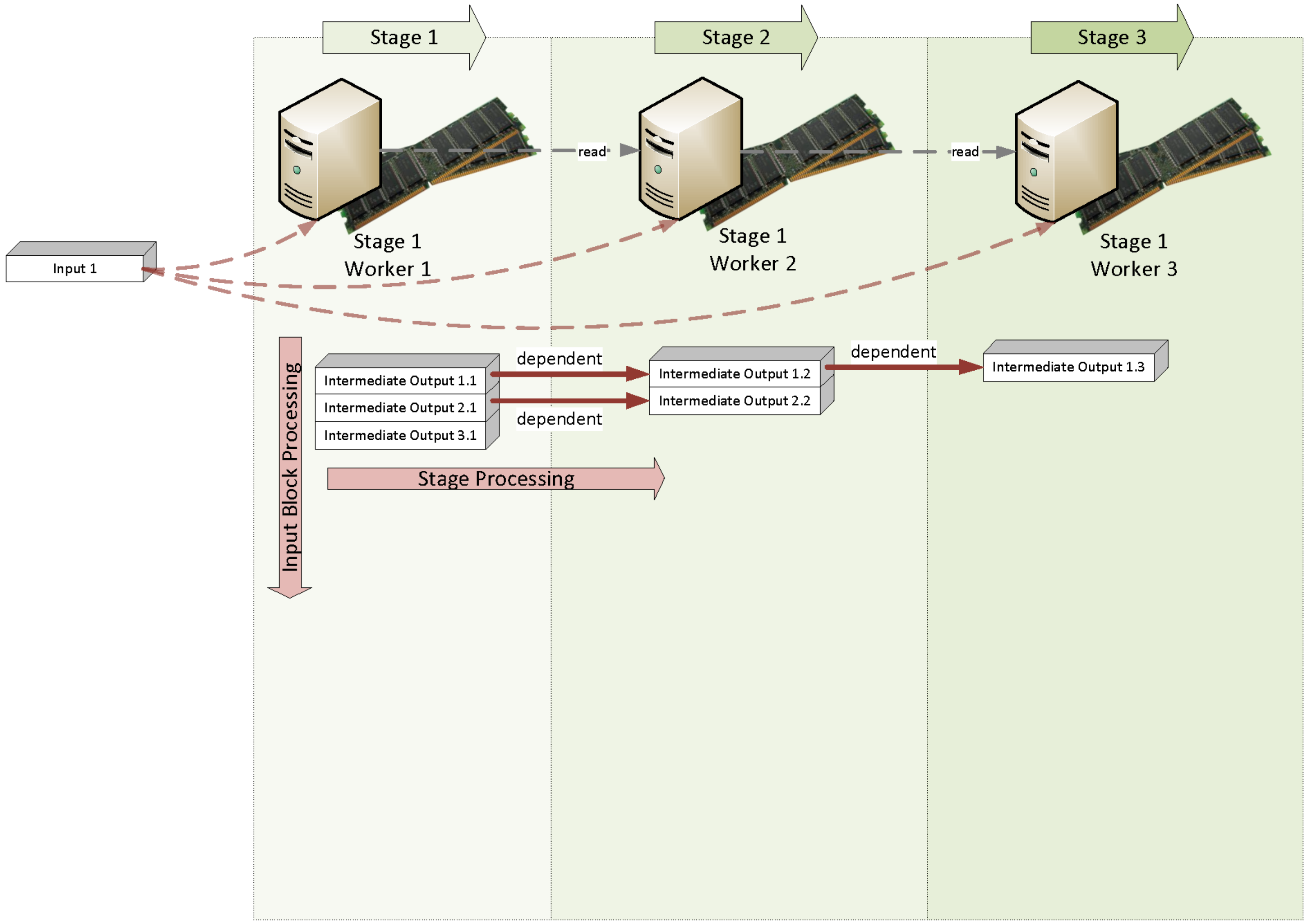
- Initialization
- Pipelined Processing
- Lineage (Checkpointing)
- Recovery Mechanisms
- RDD Dependencies



Spark

How does Spark work?

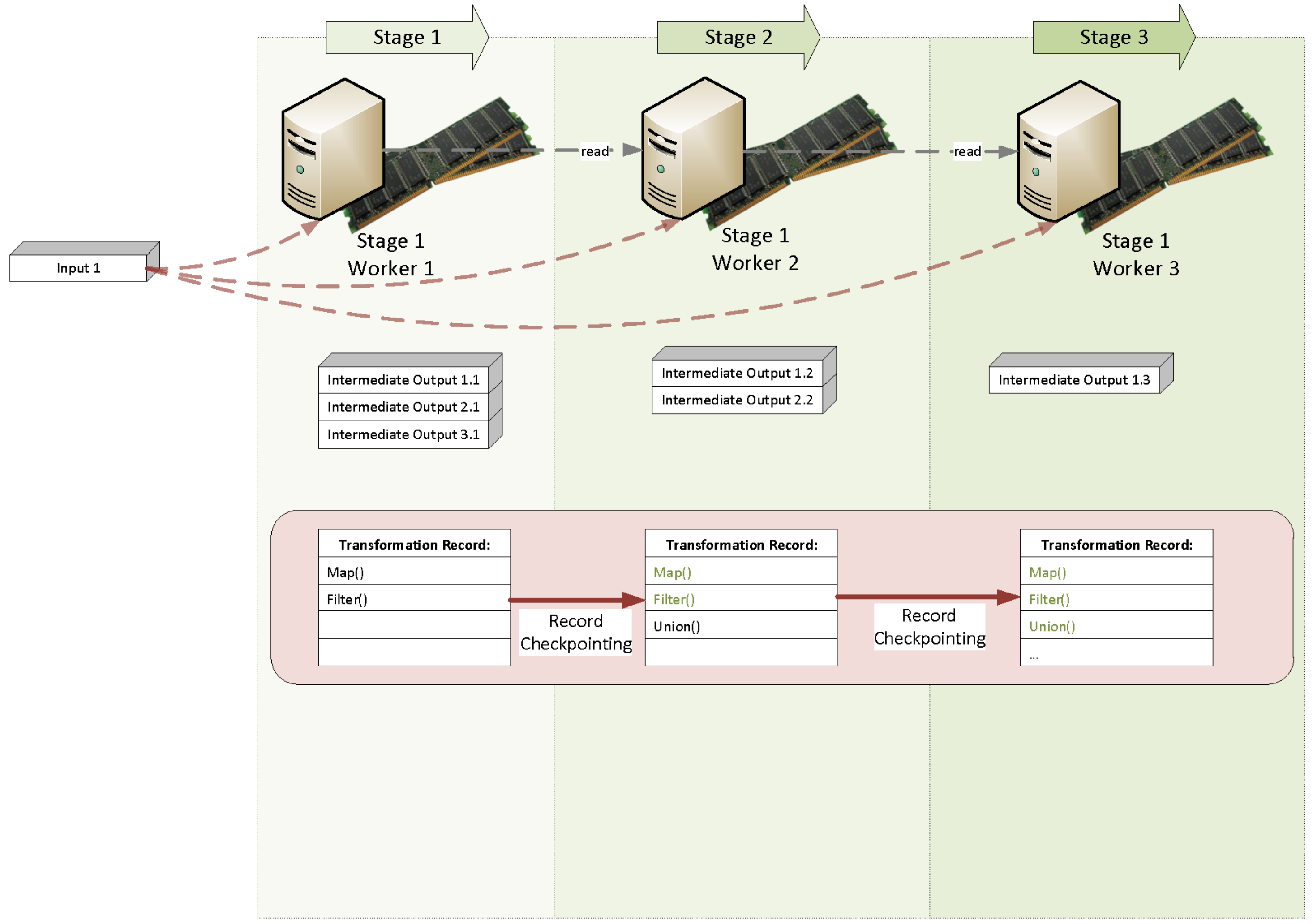
- Initialization
- Pipelined Processing
- Lineage (Checkpointing)
- Recovery Mechanisms
- RDD Dependencies



Spark

How does Spark work?

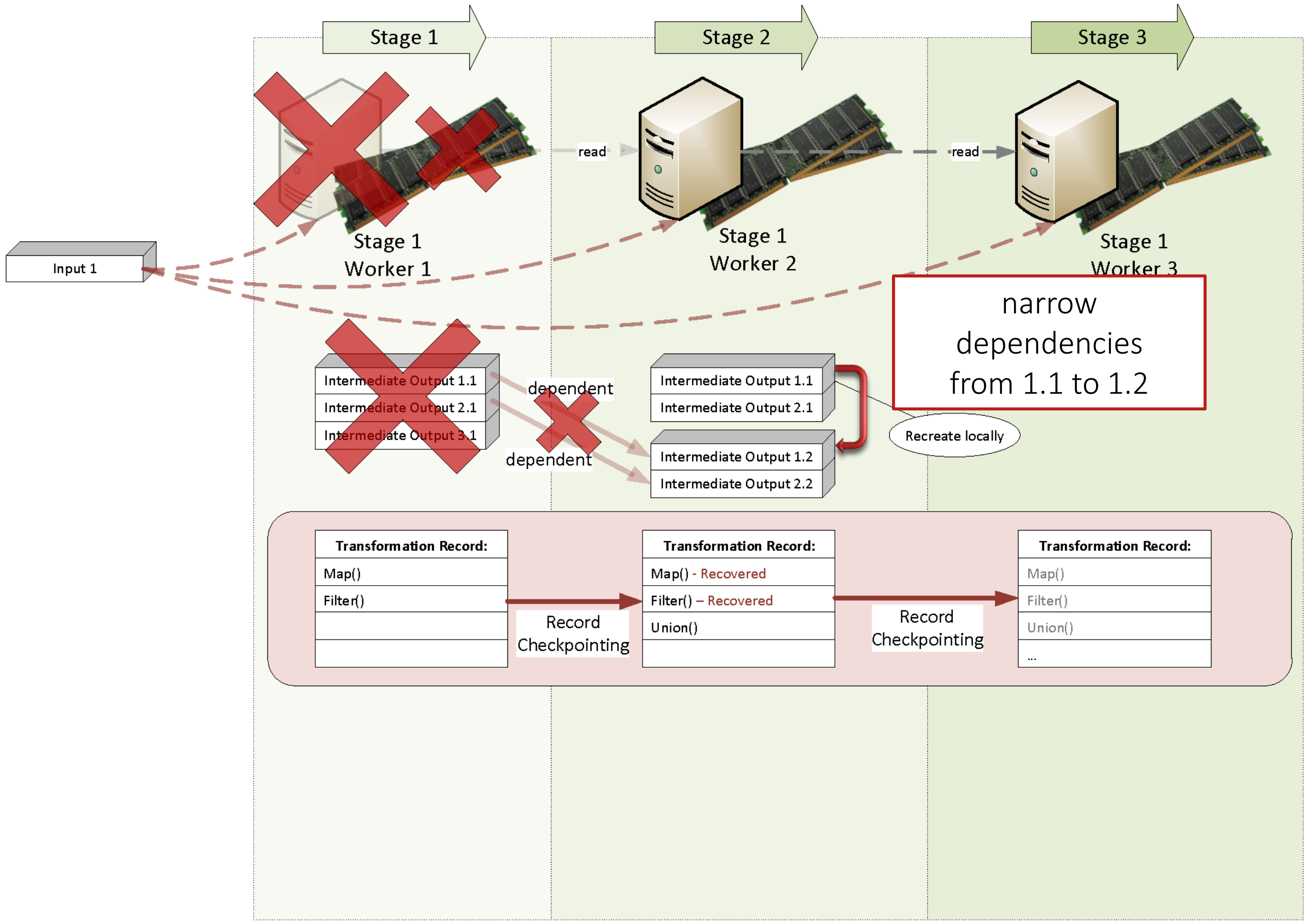
- Initialization
- Pipelined Processing
- Lineage (Checkpointing)
- Recovery Mechanisms
- RDD Dependencies



Spark

How does Spark work?

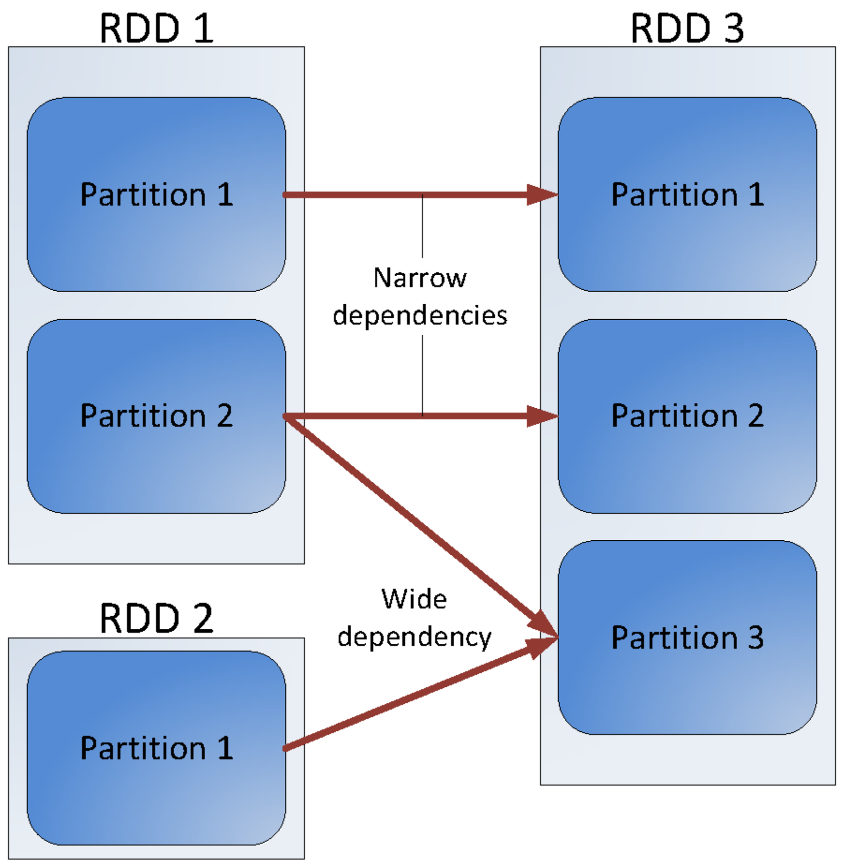
- Initialization
- Pipelined Processing
- Lineage (Checkpointing)
- Recovery Mechanisms
  - *only easy with narrow dependecies*
- RDD Dependecies



Spark

How does Spark work?

- Initialization
- Pipelined Processing
- Lineage (Checkpointing)
- Recovery Mechanisms
- RDD Dependencies



Streaming in Spark

Streaming on the example of Discretized Streams in Spark

Streaming in Spark

Discretized Streams (D-Streams)

- Each D-Stream is a *Series of RDDs*
    - Batch computations, grouped on small time interval
- Wait and Store
- Once time Interval completed:
    - execute parallel operations (map, reduce, groupBy)
    - distinguish between *stateless & stateful* operators

Streaming in Spark: Operator Comparison

Stateless Operators

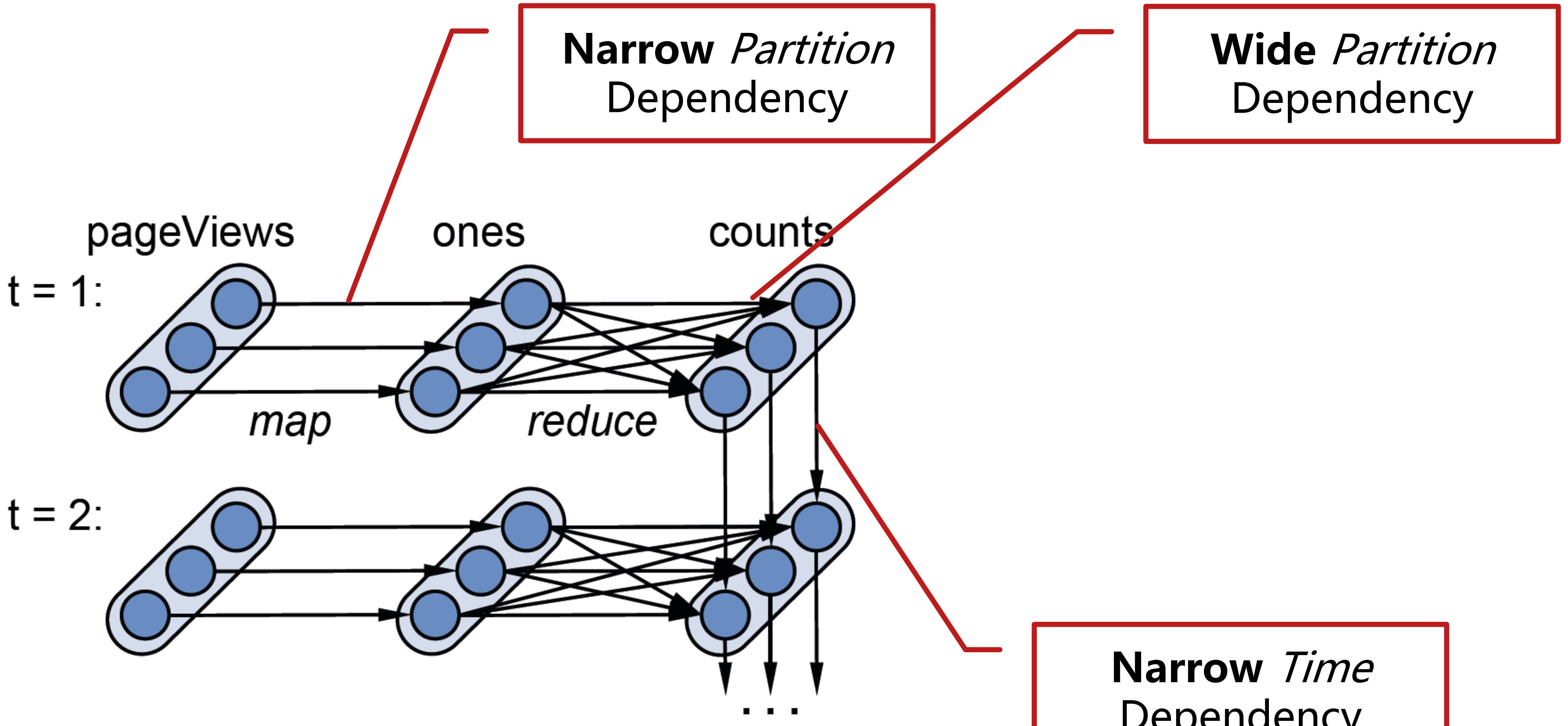
for example: **Map()**

- stateless operators are applied on each D-Stream on its own
- Stateless Operators have no dependency in time
  - act independently on each time interval

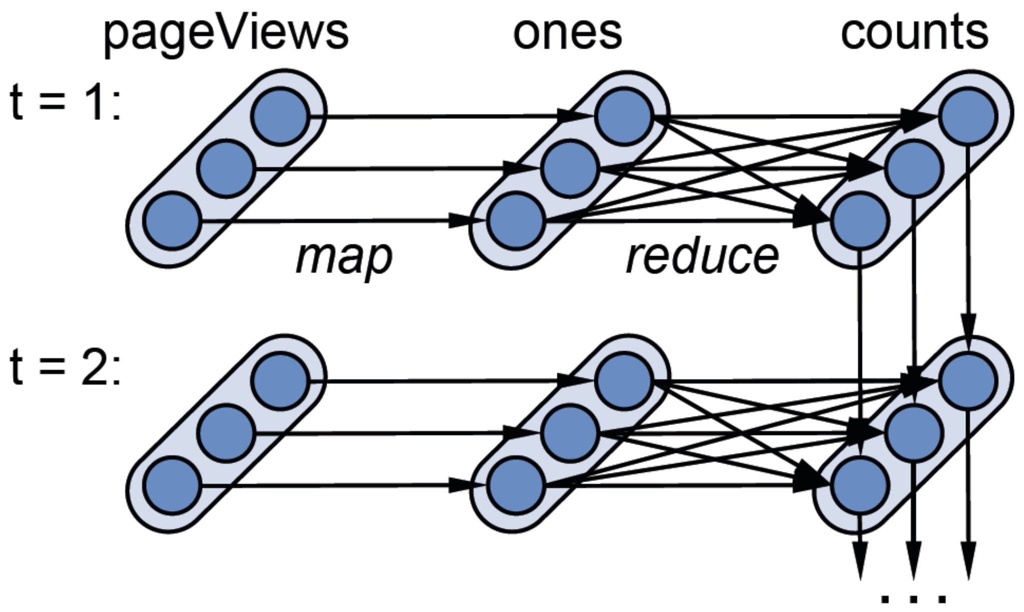
Stateful Operators

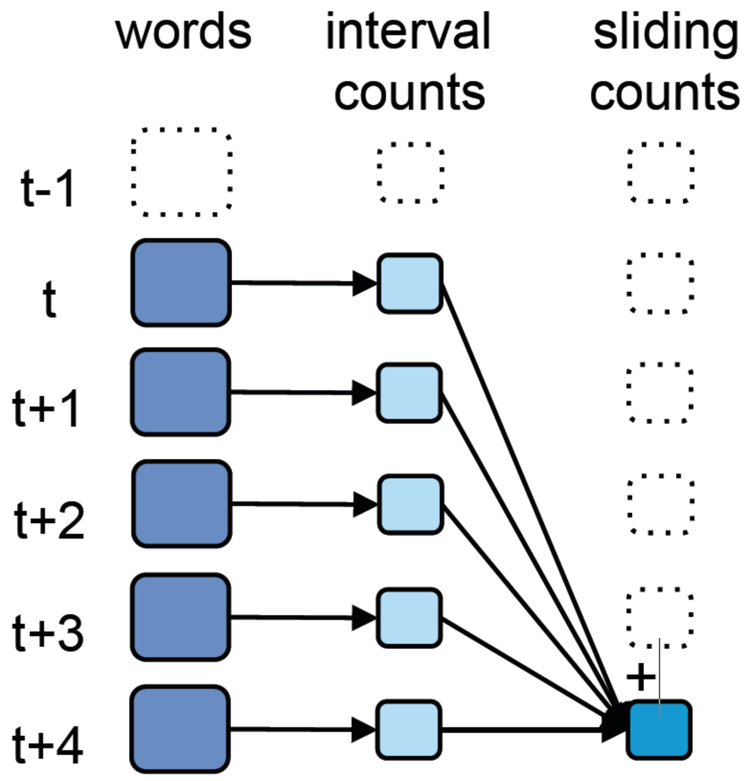
- for example: **Reduce()** over time period
- operates on multiple time intervals (D-Streams)
  - may produces intermediate RDDs as state
- Several stateful Operator-Types:
  - Windowing
  - Incremental aggregation
  - Time-Skewed Joins
- Stateful Operators have a *wide dependency* in time, when calculated over several time-intervals
- Stateful Operators have a *narrow dependency* in time, when calculated over one time-interval only

Example of a Word-Count Stream


Narrow *Partition* Dependency
Wide *Partition* Dependency
pageViews
ones
counts
t = 1:
map
reduce
t = 2:
. . .
Narrow *Time* Dependency

Example of a Word-Count Stream





Stateful Streaming Operator-Types

Windowing

- Groups all records from a range of time-intervals (D-Streams)
  - creates an intermediate RDD for the whole range
- Most general, but slow

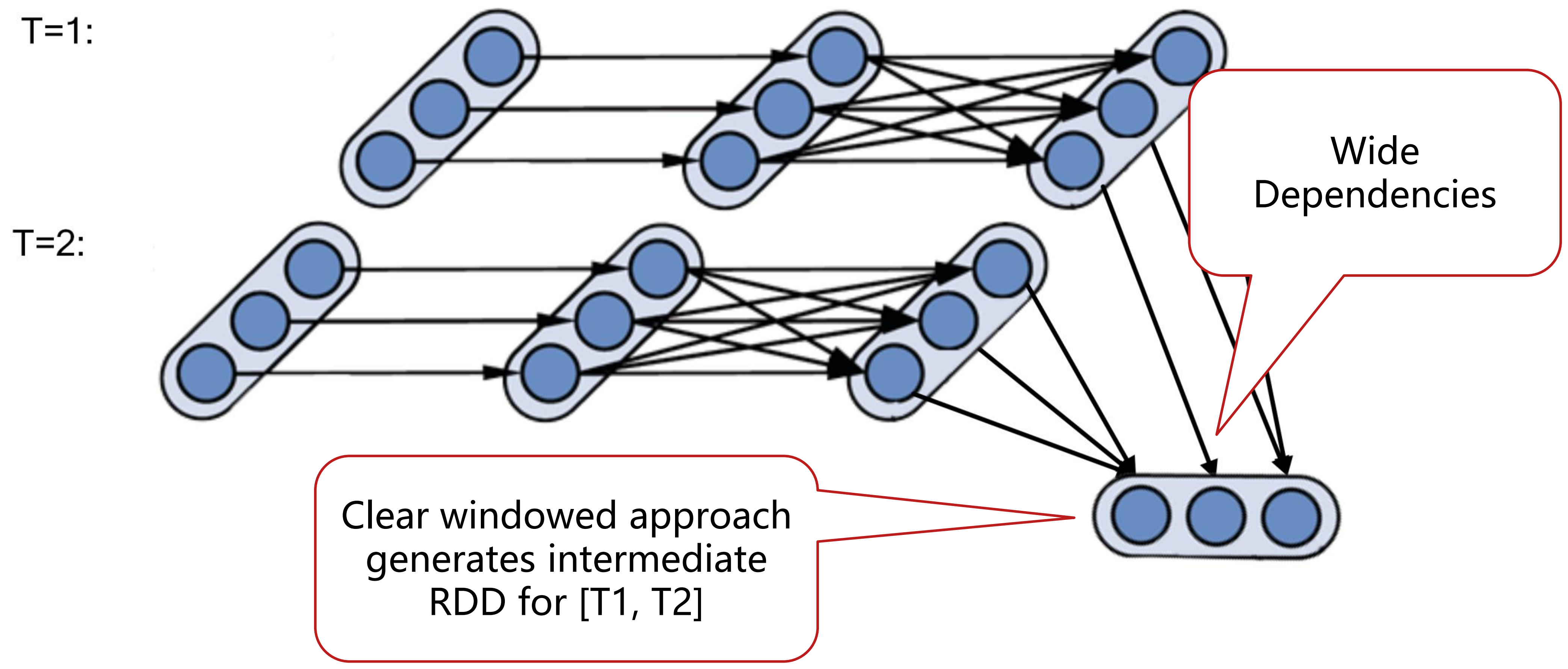
Time-Skewed Joins

- join a current D-stream against an RDD from some time ago
- Example:
  - Compare current Page view counts to page views five minutes ago

Incremental Aggregation

- computing aggregate value is a common use case (count, sum)
- sliding Window
- combines values with a merge operation

Stateful Example with wide dependencies


T=1:
T=2:
Wide Dependencies
Clear windowed approach generates intermediate RDD for [T1, T2]

Fault Tolerance

- Parallel Recovery
  - periodically checkpointing of a D-Stream‘s RDD states
    - for example: Every minute
  - async replicating checkpoint to other N nodes
  - When node failes, system detects missing RDD partitions
    - calls backup Nodes to recover latest checkpoint
    - re-calculate from there