CS 378 – Big Data Programming

Lecture 27

Aggregation and Broadcast Variables

Working with Partitions

Review

- Assignment 12
  - Create user sessions
  - Order events by timestamp
  - Order sessions by user ID
  - Partition sessions by referring domain
  - Sample OTHER sessions (1 in 1,000)

Accumulators

- In our session generator app,
- Suppose we wanted to count the number of sessions that are sampled (OTHER, 1 in 1000)

- How would we do this?

- How did we do this using Hadoop map-reduce?

Accumulators

- An accumulator provides a means for aggregating values from worker nodes back to the driver node.

- Create an accumulator from the context

- Increment the accumulator in functions passed to worker nodes

Accumulators

- For failures or re-evaluation, what happens?
- Actions:
  - Each task's update applied only once
- Transformations:
  - No guarantee that task updates applied only once
  - Re-evaluation will update accumulator each time

Broadcast Variables

- If you want to access a read-only data structure from multiple transformations
  - It will be wrapped into each closure
  - Wasteful if the data is large
- A broadcast variable addresses this issue
  - Sent to each worker node only once
  - Accessible from closures sent to the workers
  - Data must be serializable

Broadcast Variables

- Example use of broadcast variable
- In user sessions, we have:
  - VIN – vehicle identification number
  - Make, model, trim, …
- A VIN prefix (characters 1-8, 10) specifies some of this info (make, model, trim, …)
- Pass a table that maps VIN prefix to this info
- We can then verify that the info is correct

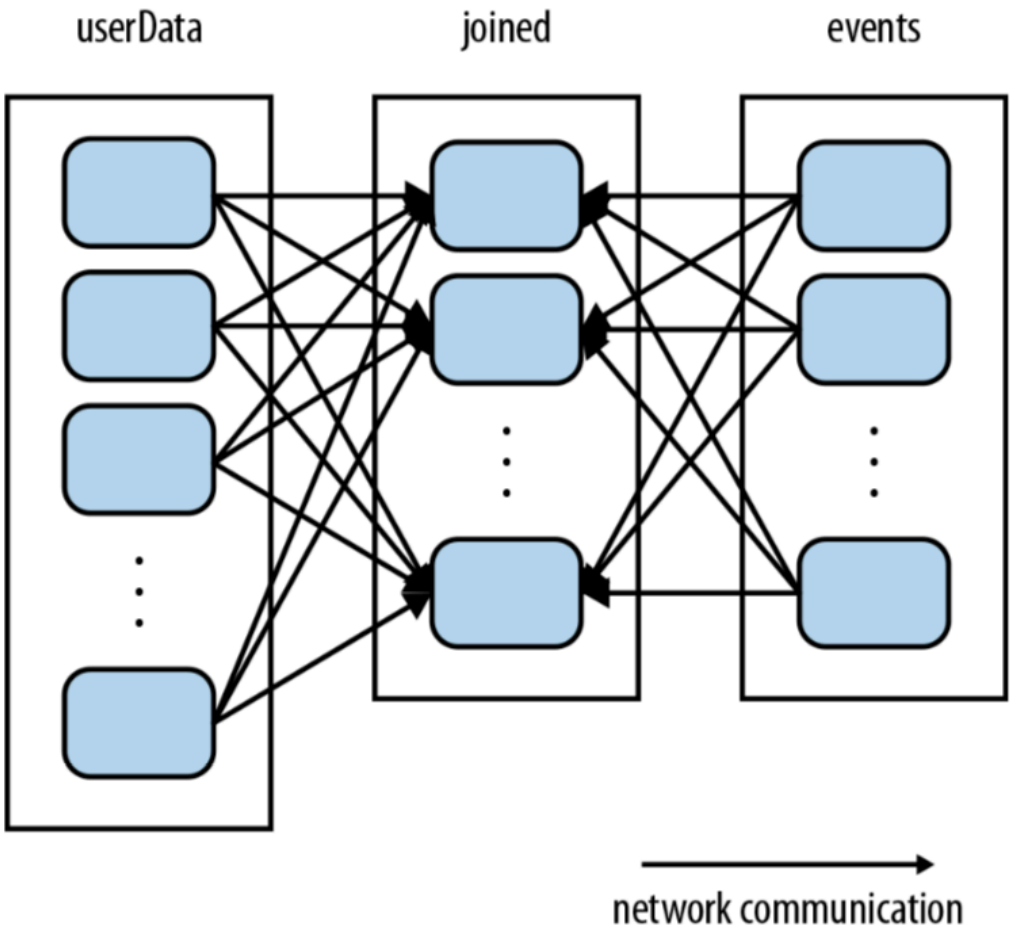
Partitioning - Review

- Prudent partitioning can greatly reduce the amount of communication (shuffle)

- If an RDD is scanned only once, no need
- If an RDD is reused multiple times in key-oriented operations
  - Partitioning can improve performance significantly

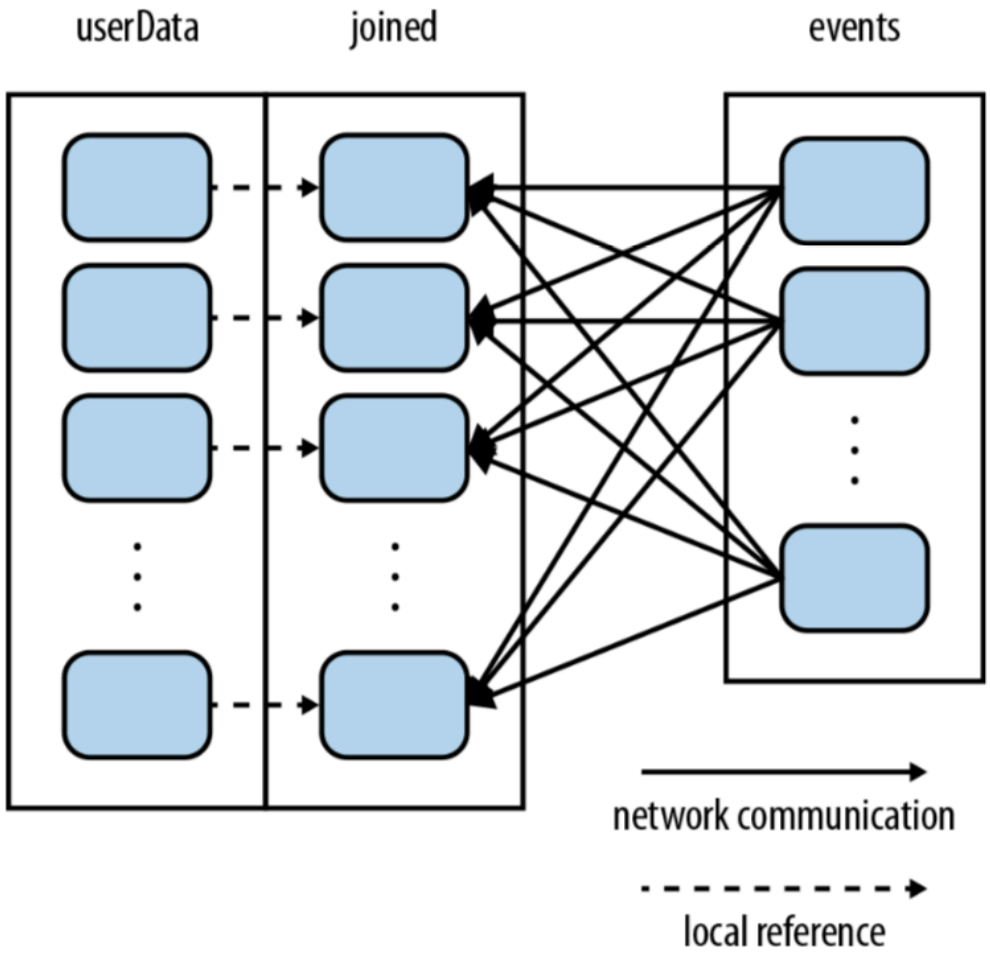
Partitioning

Figure 4-4, from Learning Spark



Partitioning

Figure 4-5, from Learning Spark



Working Per-Partition

- There are sometimes operations that we want to do once in each partition of an RDD,
- Versus once for each element in the RDD
  - Open a database connection
  - Create a complex object like a parser (XML, JSON)
- Spark has a means to do this
  - `mapPartitions()`
  - `mapPartitionsToPair()`
  - `foreachPartition()`

Working Per-Partition

- The `mapPartitions()` method takes a
  - `FlatMapFunction`
  - The `call()` method takes an iterator
  - The `call()` method is invoked once per partition

- In the `call()` method
  - Do work that should be done once (open database)
  - Iterate through the elements of the RDD partition
  - Cleanup (close database connection)
  - Returns an iterable over the results