Lecture 16.2
Spark Map and Partitioning

EN 600.320/420

Instructor: Randal Burns

9 April 2018

Department of Computer Science, *Johns Hopkins University*

Map/Reduce in Spark

- The following steps would be equivalent:
  - spark.textfile(...).flatMap(...).reduceByKey(...).save()
  - Doesn't use RDD pipelining. flatMap produces a sequence.
  - Doesn't use memory abstraction

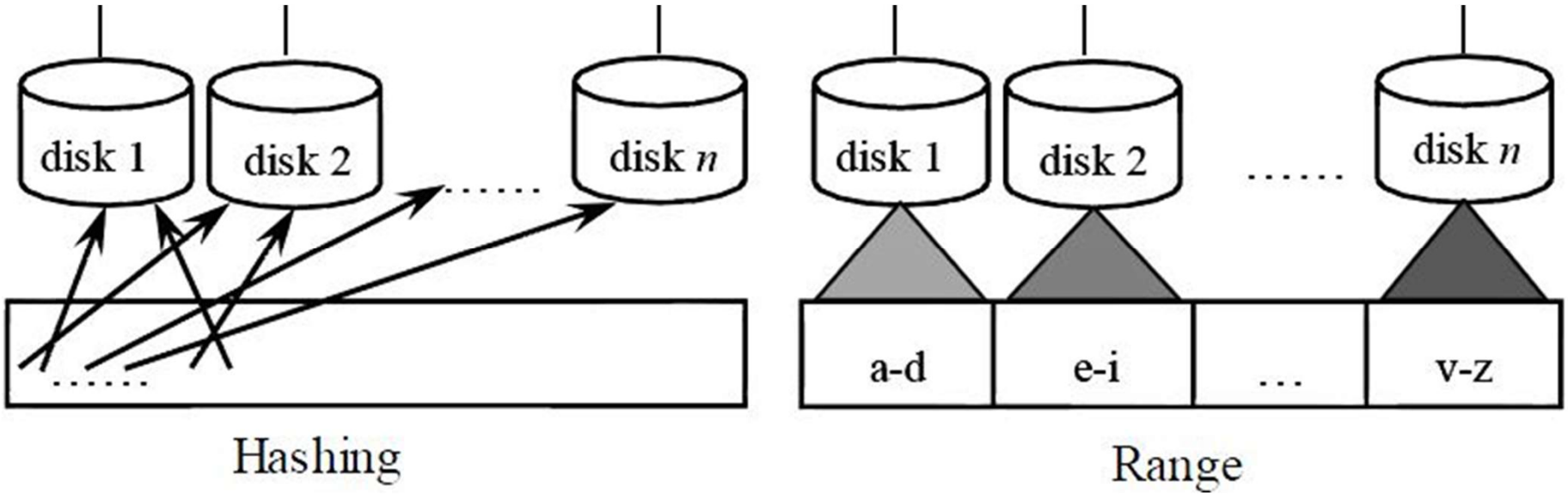
Comments about Transformations

- map() is one-to-one consistent w/ scala semantics
  - flatMap is many-to-one like Map in M/R
  - mapValues: does not transform key (important for partitioning)
- Most transformations are one-to-one
  - Important for partitioning
- Others are not one-to-one: called “wide-dependencies”
  - Bad for partitioning
  - Will discuss later

Aside: Range and Hash Partitioning

- Distribution of keys from an RDD to partitions
  - Range builds on sorting keys
  - Hash randomizes based on keys
  - For figure replace “disk” with “partition”

http://api.ning.com/files/.../5.JPG



Partitioning and Spark

- Allows fine-grained user control of partitioning
  - Can query (to discover partitioning)
  - Can specify partitions

- Range queries (block size of B)
  - Range: fetch c contiguous items in O(log n + c/B) ~ O (log n)
  - Hash: fetch any c items in O(c) time
- Stabbing/equality queries (fetch single item)
  - Range = O(log n), Hash = O(1)

- Hash partitioning is the default and is better for load balancing.