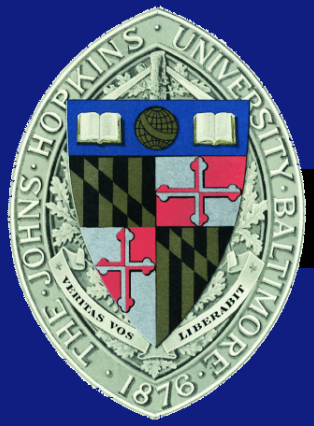
Lecture 16.4
Spark Performance

EN 600.320/420

Instructor: Randal Burns

10 April 2017



Department of Computer Science, *Johns Hopkins University*

Spark Concepts Review

- Lazy evaluation and pipelining
- Distributed memory and reuse (persist)
  - Iterative algorithms
- Data parallelism through partitioning
  - Hash and range partitions
- Fault tolerance
  - Lineage
  - Checkpoints: wide-dependencies and long lineages

How much better?

- Lots: for iterative algorithms
  - Hadoop is the WRONG tool, but people have used it anyway because of the easy access to massive parallelism

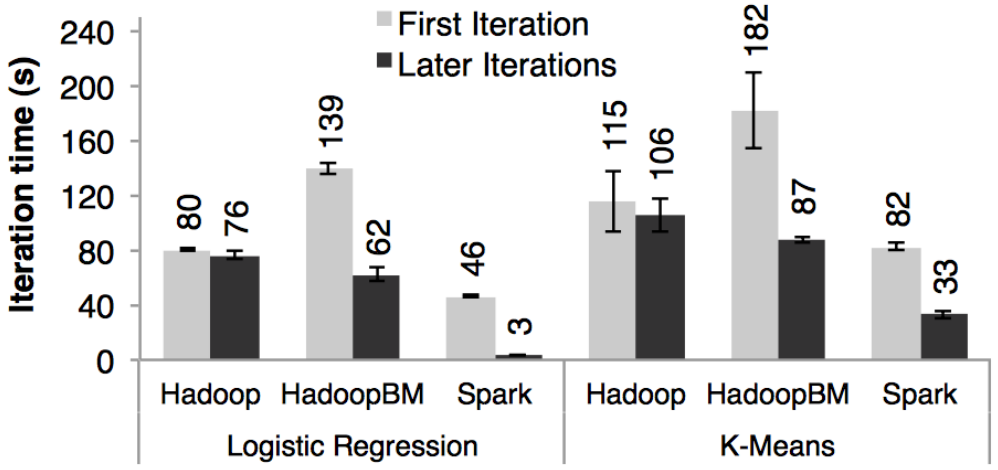


Figure 7: Duration of the first and later iterations in Hadoop, HadoopBinMem and Spark for logistic regression and k-means using 100 GB of data on a 100-node cluster.

Spark/DSM/MPI

- I can do all of these things in distributed shared memory and the message passing interface
  - Yes, but must manage distribution and recovery
  - Spark provides transparent parallelism for a broader class of applications than MR and easier than other frameworks

Spark/DSM/MPI

- Chart is a little unfair, but makes the same point

Aspect	RDDs	Distr. Shared Mem.
Reads	Coarse- or fine-grained	Fine-grained
Writes	Coarse-grained	Fine-grained
Consistency	Trivial (immutable)	Up to app / runtime
Fault recovery	Fine-grained and low-overhead using lineage	Requires checkpoints and program rollback
Straggler mitigation	Possible using backup tasks	Difficult
Work placement	Automatic based on data locality	Up to app (runtimes aim for transparency)
Behavior if not enough RAM	Similar to existing data flow systems	Poor performance (swapping?)

Table 1: Comparison of RDDs with distributed shared memory.