Concepts and Models of Parallel and Data-centric Programming

Apache Spark – Streaming and Applications

Lecture, Summer 2020

Simon Schwitanski
Dr. Christian Terboven

i12 High Performance Computing
RWTH AACHEN UNIVERSITY

Outline

0. Organization
1. Foundations
2. Shared Memory
3. GPU Programming
4. Bulk-Synchronous Parallelism
5. Message Passing
6. Distributed Shared Memory
7. Parallel Algorithms
8. Parallel I/O
9. MapReduce
10. **Apache Spark**

a. Spark Programming Model
b. Resilient Distributed Datasets (RDDs)
c. Job Scheduling and Fault Tolerance
d. Streaming and Applications
e. Concluding Remarks

Spark Streaming

- Extension of core Spark API
- Apply Spark functions on streamed data
- Data flow
  - Live input stream divided into batches
  - Batches processed by Spark engine to generate final stream of results in batches
- High-level abstraction of continuous data stream: *discretized stream (DStream)*
- DStream internally represented as sequence of RDDs
  - API similar to RDDs

Image Source: https://spark.apache.org/docs/latest/streaming-programming-guide.html

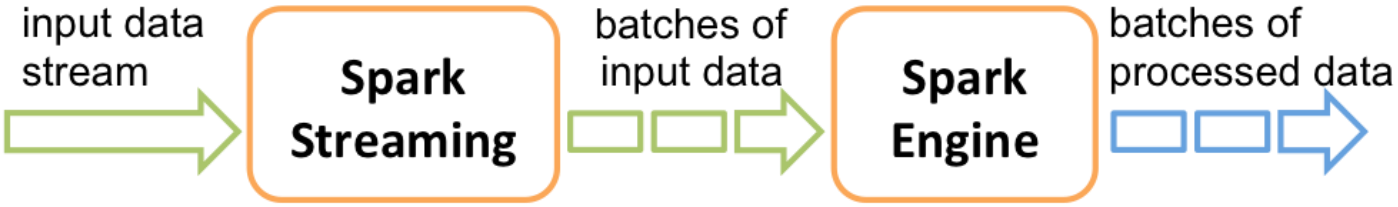

Image Source: https://spark.apache.org/docs/latest/streaming-programming-guide.html

Example Application – Traffic Modeling

- Inferring road traffic congestion from automobile GPS measurements using parallelized learning algorithm
- “Mobile Millennium” project 2011 at Berkeley
- Source data: 10,000 link road network for metropolitan area, 600,000 samples of point-to-point trip times for automobiles with GPS
- Traffic model allows time estimation for travel across individual road links
- Model trained using an EM algorithm (expectation maximization) with *map()* and *reduceByKey()* steps
- Figure: Application scales nearly linearly from 20 to 80 nodes with 4 cores

Image Source: Timothy Hunter et al. “Scaling the mobile millennium system in the cloud”. SoCC 2011: 28

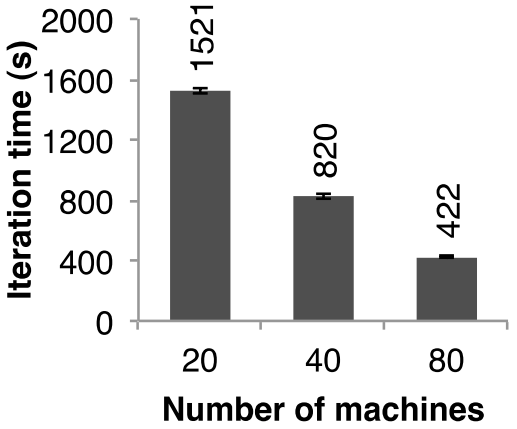

Image Source: Matei Zaharia et al. “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing.” NSDI2012: 15-28

Example Application – Twitter Spam Classification

- Use Spark to identify link spam in Twitter messages
- "Monarch" project at Berkeley
- Logistic regression classifier on top of Spark
- Iteratively calling *reduceByKey()*
- Figure: 50 GB training data
  - 250,000 URLs
  - 10^7 features / dimensions
  - 4 cores per machine
- Scaling behavior slightly worse → Higher fixed communication cost per iteration

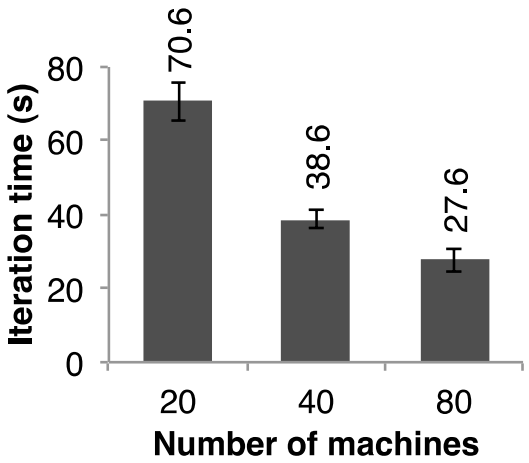

Image Source: Matei Zaharia et al. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing." NSDI2012: 15-28