15-440: Distributed Systems

Rubrics of Project 4

School of Computer Science
Carnegie Mellon University, Qatar

MapReduce K-Means (Points: 50%)

- K-Means using MapReduce for 2D data points (25%)
- K-Means using MapReduce for DNA strands (25%)

  Details (these are same for DNA and 2D data points versions):

  - ✓ The code compiles and runs correctly (1%)
  - ✓ Create map and reduce classes, extend MapReduceBase and implement Mapper and Reducer (1%)
  - ✓ Configuring jobs correctly including setting the mapper and reducer classes, the input and output formats, the file input and output formats, launching a job, and looping over for new rounds (3%)
  - ✓ Read the value of centroids at each mapper (3%)
  - ✓ Create the map function with correct input and output key-value pairs (i.e., correct input and output formats) (3%)
  - ✓ Apply K-Means at the mapper (2%)
  - ✓ Create the reduce function with correct input and output key-value pairs (i.e., correct input and output formats) (3%)
  - ✓ Compute new means at the reducer (3%)
  - ✓ Write the value of centroids at each reducer (3%)
  - ✓ Outputting final results and aborting cleanly (3%)

Write-up (Points: 45%)

- ✓ Performance comparison between sequential, MPI, and MapReduce with a fixed dataset size (10%)
- ✓ Development effort comparison between sequential, MPI and MapReduce (2%)
- ✓ MapReduce scalability w.r.t 1, 2, and, 4 map slots (10%)
- ✓ MapReduce scalability w.r.t 32MB, 64MB, and 128MB block sizes (10%)
- ✓ MapReduce scalability w.r.t 1, 2, 3 VMs (10%)
- ✓ Performance trade-offs between MPI and MapReduce (2%)
- ✓ Thoughts on the applicability of K-Means to MapReduce (2%)
- ✓ Recommendations regarding the usage of MapReduce for algorithms similar to K-Means (2%)
- ✓ Paper structure, level of writing, and language (2%)

Code Style (Points 5%)

- Method Comments, Block comments, Readability, Dead code, Code design