Dense Matrix Multiplication

Sreenivasan Ramesh
sreenivasan.ramesh@asu.edu

Hardware Configuration

Processor	Intel Core i5 (8259U)
Base Clock Frequency	2.3 GHz
Core Count	4 cores
Memory	8 GB

Software Configuration

Operating System	MacOS Catalina 10.15.4
Scala	2.11.12
Spark-core	2.4.5
Spark-mllib	2.4.5
Netlib	1.1.2
Breeze	0.11.2
Sbt	1.3.10

Performance Comparison

100 iterations

	Hand-coded Scala	Spark	Breeze
Iterations	100	100	100
Time Taken	233.3 seconds	1.7 seconds	1.7 seconds

1000 iterations

	Hand-coded Scala	Spark	Breeze
Iterations	1000	1000	1000
Time Taken	42 minutes	16.7 seconds	17.7 seconds

Hand-coded Scala is the slowest with 1000 iterations taking over 40 minutes. Both Spark mllib's and Breeze's DenseMatrix provide much better results, both taking about 17 seconds for 1000 iterations. The one second difference between Spark and Breeze for 1000 iterations is within the margin of error, and the performance difference between using Spark and Breeze is negligible.

Things Learned:

Implementations of numerical operations on large size dense vectors are expensive. Libraries like Breeze and Spark, and their implementations for linear algebra operations provide optimized implementations for such operations offering huge performance gains.

We also got some hands-on experience with Scala and the sbt build tool. I also played around with compiling the code with sbt to create a fat jar as spark-submit accepts only fat jars. Creating thin jars on sbt is easy, but sbt is a pain for creating fat jars as you have to figure out merge strategies for conflicting packages during assembly and there are a lot of versioning dependencies. Maven and Gradle are much easier to deal with for building packages.