Different file formats

Understanding different file formats is very important.

* Different file formats using SparkContext – textFile, sequenceFile
* Different file formats using SQLContext – parquet, orc, avro, json
* Understand different compression algorithms – deflate, gzip, bzip2, snappy
* One should be able to read and write all the formats
* Compression need not be specified while reading the data, we can use appropriate API to read respective file format which is compressed
* We can also use SQLContext to read data from remote database using JDBC

Quick glance of supported file formats in Spark

|  |  |  |
| --- | --- | --- |
| **File Format** | **Action** | **Procedure and points to remember** |
| TEXT FILE | READ | sparkContext.textFile(<path to file>); |
| WRITE | sparkContext.saveAsTextFile(<path to file>,classOf[**compressionCodecClass**]); //use any codec here org.apache.hadoop.io.compress.([BZip2Codec](https://hadoop.apache.org/docs/r3.0.0-alpha2/api/org/apache/hadoop/io/compress/BZip2Codec.html) or GZipCodec or SnappyCodec) |
| SEQUENCE FILE | READ | sparkContext.sequenceFile(<path location>,”KEY\_TYPE”,”VALUE\_TYPE”); //read the head of sequence file to understand what two class names need to be used here |
| WRITE | rdd.saveAsSequenceFile(<path location>, “COMPRESSION\_CODEC”) //use any codec here ([BZip2Codec](https://hadoop.apache.org/docs/r3.0.0-alpha2/api/org/apache/hadoop/io/compress/BZip2Codec.html),GZipCodec,SnappyCodec) //here rdd is MapPartitionRDD and not the regular pair RDD. |
| PARQUET FILE | READ | //use data frame to load the file. sqlContext.read.parquet(<path to location>); //this results in a data frame object. |
| WRITE | sqlContext.setConf(“spark.sql.parquet.compression.codec”,”gzip”) //use gzip, snappy, lzo or uncompressed here dataFrame.write.parquet(<path to location>); |
| ORC FILE | READ | sqlContext.read.orc(<path to location>); //this results in a dataframe |
| WRITE | df.write.mode(SaveMode.Overwrite).format(“orc”) .save(<path to location>) |
| AVRO FILE | READ | import com.databricks.spark.avro.\_; sqlContext.read.avro(<path to location>); // this results in a data frame object |
| WRITE | sqlContext.setConf(“spark.sql.avro.compression.codec”,”snappy”) //use snappy, deflate, uncompressed; dataFrame.write.avro(<path to location>); |
| JSON FILE | READ | sqlContext.read.json(); |
| WRITE | dataFrame.toJSON().saveAsTextFile(<path to location>,”COMPRESSION\_CODEC”) |