PA-2b Report

Yunjie Xu

A20303205

Methodology:

In HadoopSort program designed: I created a new Mapper class, and in map function, it gets first 10 chars as KEY and rest of the line as Value; in new Reducer class, I convert Text structure to string and write back to context. In main function, I first set configuration and set class, set Mapper, set Combiner, set Reducer, and set outputKey as Text and set outputVaule as Text and also set MapoutputKey and MapoutputValue as Text. And using input and output passed by interactive to open the file and save final file.
In SparkSort program design: in main function, set configuration first, and create JavaRDD object to read file it passes; create JavaPairRDD and JavaRDD using maptoPair to store key as each line and store empty value to Tuple2 structure; and sort JavaRDD object by function sortbykey, and then use map to convert tuple type to string; and last store the string using saveAsTextFile function.

Environment:

In Hadoop setting, I used default mapper task(2) and set reducer tasks to 200. In Spark, drive memory is 1g, and executing memory 7G, executing core 4, number of executing is 4.

Problem:

in Hadoopsort, I can't finish my sort 80GB file in hour, and I tried increasing reducer tasks from 1 to 20, but it still can't finish job in hour. And Xuzhu Chen who told me that I should increase reducer tasks number to 200, and I tried to increase reducer task to 200 and success finished job in hour.

In Spark, there were two problems put me stuck in few days, one is when I used tuple2 data frame in sortbykey function, it is very hard to remove parenthesis in each line when I output to file, I tried different way to remove, like convert to string and used substring to get rid of head and tail, but it's useless, and I think this data specific data frame and I can't simply used this way to change data frame. And another problem was when I validate my data, even it seems sorted, it showed not right type of data, this problem was coming from each line in raw data having \r\n in the end, and when I sort each line, Spark remove \r\n and when I save back to file, it just adds \n. I simply added \r in each value.

Result:

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Hadoop Sort (4VM 8GB)	Spark Sort (4VM 8GB)
Computation Time (sec)	29	26	255	232
Data Read (GB)	2GB	2GB	16GB	16GB
Data Write (GB)	2GB	2GB	16GB	16GB
I/O Throughput (MB/sec)	141	157	128	141
Speedup	N/A	N/A	-226/-219	-212/-209
Efficiency	N/A	N/A	11%/10%	12.5%/11.2%

Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 20GB)	Spark Sort (4VM 20GB)
Computation Time (sec)	542	480	463	740
Data Read (GB)	60	40GB	40GB	40GB
Data Write (GB)	60	40GB	40GB	40GB
I/O Throughput (MB/sec)	226	170	176	
Speedup	N/A	N/A	79/17	-198/-260
Efficiency	N/A	N/A	117%/103%	73%/64%

Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 80GB)	Spark Sort (4VM 80GB)
Computation Time (sec)	542	480	1804	1254
Data Read (GB)	60	40	160	160
Data Write (GB)	60	40	160	160
I/O Throughput (MB/sec)	226	170	181	261
Speedup	N/A	N/A	-1262/-1324	-710/-770
Efficiency	N/A	N/A	30%/26%	43%/38.2%

Discussion:

It seems most case, spark is fast than Hadoop, but data is 20GB, spark is slower. I think I didn't use maximum memory to reduce RDD process, maybe this is key point of speed.

Conclusion:

Weak scaling part, Hadoop and spark is slower than share memory and linux sort, and I think it is very make sense, because Hadoop need manage task and start and spark need write back to disk when memory is full. Spark is fast than Hadoop, but it is slower than share memory and linsort.

Strong scaling part, Hadoop and spark is slightly fast than Linsort and shared memory, and I think it can further increase speedup if tuning parameters. While Spark is slower than both linsort, share memory and Hadoop.