BWT by Spark

Xiang Niu
Johns Hopkins University
xniu7@jhu.edu

1 Introduction

1.1 BWT

The BurrowsWheeler transform (BWT) is an algorithm used in data compression techniques. When a character string is transformed by the BWT, none of its characters change value. The transformation permutes the order of the characters. For example, the BWT of "abaaba$" is "abba$aa".

1.2 Spark

In this project, we found spark is a good parallel tool and decide to use it to construct the BWT. Spark contains two kinds of operations, transformations and actions. Data will be processed only after an action is finished. In our project, we use several transfromations such as map, flatMap, filter, groupByKey, reduceByKey, sortByKey; and several actions such as reduce, collect, count, saveAsFile.

2 Parrallel Sort

We tried two parallel sorting methods, which is called default sort and partition sort.

2.1 Default Sort

- parallel load text files.
- map each read to multiple suffixes by flatMap.
- sort suffixes by sortBykey.
- store the sorted value by saveAsFile.

Table 1: running time of 1M reads by default sort

5 machines	m1.large	m1.xlarge	c1.xlarge	c3.4xlarge	c3.8xlarge
sort	360s	102s	75s	43s	29s
write	336s	120s	66s	28s	19s
total	696s	222s	141s	71s	48s
10 machines	m1.large	m1.xlarge	c1.xlarge	c3.4xlarge	c3.8xlarge
sort	132s	48s	46s	26s	7s
write	192s	60s	30s	15s	11s
total	324s	108s	76s	41s	18s

2.2 Partition Sort

- parallel load text files.
- map each read to multiple suffixes by flatMap.
- partition suffixes by groupByKey.
- sort partitions by sortByKey.
- sort suffixes in each partition by radix sort.
- store the sorted value by saveAsFile.

3 Result

We test and compare two kinds of sorting on amazon EC2. Our dataset includes 1M, 10M, 100M reads which stored in multiple text files in amazon S3.

3.1 1M reads

At first, we sort 1M reads by default sort and test results on different machines. Table 1 shows that sorting 1M reads needs only several minutes. Besides, running on 10 machines needs almost the half time compared to 5 machines. Table 2 shows that sorting 1M reads by default sort needs almost the half time compared to partition sort.

Table 2: running time of 1M reads by default sort and partition sort

	default	partition
5 machines	11m	20m
10 machines	5m	11m

Table 3: running time of reads by default sort and partition

	1M	10M	100M
instance	10 m1.xlarge	10 m1.xlarge	10 m2.2xlarge
cpu	10x4	10x4	10x4
memory	10x15G	10x15G	10x35G
price	10x0.35	10x0.35	10x0.49
launch time	15m	15m	15m
default sort	4m	2h	10h
partition sort	8m	>2h	>10h
cost of default sort	$2.00	$8.00	$50.00

3.2 100M reads

Table 3 shows that sorting 100M reads by default sort needs only 50.00$. Besides, there are two important things that we should consider during the sorting.

- The memory of each instance. Unlike hadoop, spark use memory to run a program. In this case, we should make sure the program will not exceed memory in each instance. That's why we choose m2.2xlarge instance instead of m1.xlarge instance when dealing with 100M reads.
- The cpu of each instance. We could save more time by a faster cpu. However, there is a trade-off between speed and price. Pick a suitable cpu is also importance.

4 Conclusion

Constructing BWT by spark is feasible nowadays. For any individual researchers, they could pay for less than 100$ to construct a BWT even for 100M reads. This method is economic and practical.