Large-scale Workload Characterization in Apache Spark Framework

Sebastian Cousins, Debzani Deb
Department of Computer Science
Winston-Salem State University

Purpose

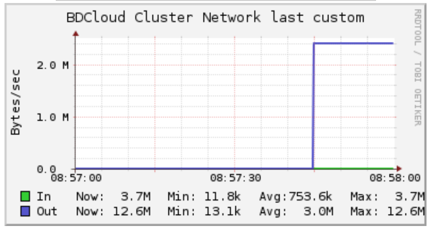
- Apache Spark platform [1] has grown both in popularity and complexity and has been increasingly adopted by a wide spectrum of areas for big data analytics and many of them use cloud environments to deploy their applications.
- The goal of this study is to investigate the characteristic of spark application performance through running them with various configuration settings and to gain deep insight about the resource provisioning, workload management etc.
- We implemented a Spark on YARN cluster on the Chameleon Cloud [2], a large-scale and open cloud research platform funded by the NSF that allows for greater computational power. Our cluster has master and 8 slave nodes (each with 48 cores, 128 GB memory) running Ubuntu 16.04 OS with Hadoop 2.7.4, Spark 2.2.0, and Scala 2.13.8.

Methodology

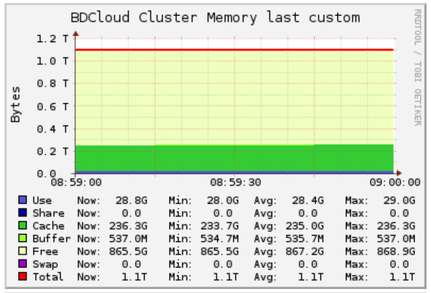
- We gathered our experimental data based on the logged information available through Spark's history server and by utilizing cluster-wide monitoring tool Ganglia, that provides us important insight about the overall cluster resource utilization (such as CPU and memory).
- We used two representative Spark-SQL applications from TPC-H [3] benchmark and executed them with 50GB and 100GB datasets. We choose Query 1 (Q1) and Query 5 (Q5) because of their unique characteristics. Q1 has minimal join operation (therefore least shuffling data), which makes it to be CPU bound, on the other hand, Q5 is an I/O bound job that contains multiple join operations (significant shuffling during stages).

SQL Query Resource Utilization (single configuration)

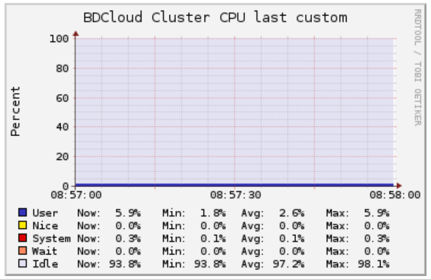
Q1



```
🟩 In    Now:  3.7M   Min: 11.8k   Avg:753.6k   Max:  3.7M
🟦 Out   Now: 12.6M   Min: 13.1k   Avg:  3.0M   Max: 12.6M
```

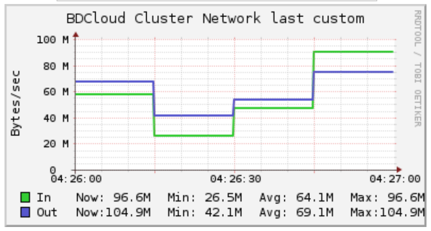


```
🟦 Use    Now:  28.8G   Min:  28.0G   Avg:  28.4G   Max:  29.0G
🟦 Share  Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
🟩 Cache  Now: 236.3G   Min: 233.7G   Avg: 235.0G   Max: 236.3G
🟩 Buffer Now: 537.0M   Min: 534.7M   Avg: 535.7M   Max: 537.0M
🟨 Free   Now: 865.5G   Min: 865.5G   Avg: 867.2G   Max: 868.9G
🟪 Swap   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
🟥 Total  Now:   1.1T   Min:   1.1T   Avg:   1.1T   Max:   1.1T
```

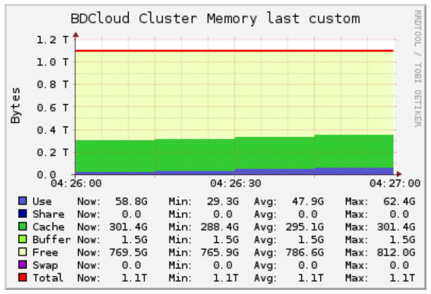


```
🟦 User   Now:  5.9%   Min:  1.8%   Avg:  2.6%   Max:  5.9%
🟨 Nice   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
🟥 System Now:  0.3%   Min:  0.1%   Avg:  0.1%   Max:  0.3%
🟧 Wait   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
⬜ Idle   Now: 93.8%   Min: 93.8%   Avg: 97.2%   Max: 98.1%
```

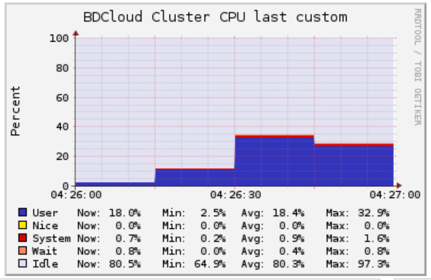
Q5



```
🟩 In    Now: 96.6M   Min: 26.5M   Avg: 64.1M   Max: 96.6M
🟦 Out   Now:104.9M   Min: 42.1M   Avg: 69.1M   Max:104.9M
```



```
🟦 Use    Now:  58.8G   Min:  29.3G   Avg:  47.9G   Max:  62.4G
🟦 Share  Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
🟩 Cache  Now: 301.4G   Min: 288.4G   Avg: 295.1G   Max: 301.4G
🟩 Buffer Now:   1.5G   Min:   1.5G   Avg:   1.5G   Max:   1.5G
🟨 Free   Now: 769.5G   Min: 765.9G   Avg: 786.6G   Max: 812.0G
🟪 Swap   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
🟥 Total  Now:   1.1T   Min:   1.1T   Avg:   1.1T   Max:   1.1T
```



```
🟦 User   Now: 18.0%   Min:  2.5%   Avg: 18.4%   Max: 32.9%
🟨 Nice   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
🟥 System Now:  0.7%   Min:  0.2%   Avg:  0.9%   Max:  1.6%
🟧 Wait   Now:  0.8%   Min:  0.0%   Avg:  0.4%   Max:  0.8%
⬜ Idle   Now: 80.5%   Min: 64.9%   Avg: 80.3%   Max: 97.3%
```

Job Runtimes

	Exec	Cores	Mem	Runtime
Q1 50GB	8	46	112G	36 s
	16	23	50G	34 s
	24	15	36G	33 s
	36	11	18G	35 s
	48	5	8G	37 s
Q5 50GB	8	46	112G	42 s
	16	23	50G	44 s
	24	15	36G	43 s
	36	11	18G	45 s
	48	5	8G	44 s
Q1 100GB	8	46	112G	45 s
	16	23	50G	40 s
	24	15	36G	41 s
	36	11	18G	43 s
	48	5	8G	41 s
Q5 100GB	8	46	112G	63 s
	16	23	50G	64 s
	24	15	36G	67 s
	36	11	18G	65 s
	48	5	8G	63 s

Conclusions

- The two different applications exhibits different resource consumptions and their execution times are significantly impacted by the settings of the configuration parameters such as the number of concurrent tasks in each executor, executor configuration (core and memory per executor) etc.
- Configuring a remote Spark on Yarn cluster from scratch with all the abovementioned tools is a significantly challenging task and we learned a lot throughout the process.

References

1. Apache Spark, https://spark.apache.org
2. Chameleon cloud, https://www.chameleoncloud.org
3. TPC-H is a Decision Support Benchmark, http://www.tpc.org/tpch/

Acknowledgment

NSF RIA Award #1600864