Google Cluster Trace Analysis with Apache Spark

Kadir Korkmaz

19/01/2019

Contents

Abstract

In this worked, I have analyzed the Google Cluster trace using Apache Spark. I have used the Data Frame API of Spark. I have conducted proposed analyses. Google Cluster trace contains data for 29 days. In the cluster, there are 12,583 machines. In the cluster, there is only 10 different kind of machine when we consider the CPU and memory. More than 90% of the machines have 0.5 CPU capacity. Only 6.32 percent of the machines have the highest CPU and memory capacity. Machines have three different kinds of hardware platforms. During the trace period, 672.004 Job is scheduled. Every job has one or more task. The total number of tasks is 25.4 million. The average number of task per job is 37.83. Because of eviction and killing scheduled task count 47.3 million. There is no linear relation between the priority of a task and probability of eviction but it is clear that some priority levels have a higher evicted and killed task percentage. There is no relation between the priority of a task and resources of the machines which task is scheduled. Tasks are almost uniformly distributed on machines. There is tasks which consume significantly fewer resources than what they requested. In the cluster, there is a maintenance or machine failure for every 279 seconds. There are 36 machines which are underutilized.

The number of worker threads has a big effect on performance if there is enough input file to process for every worker thread. JVM memory size could affect performance in a different way. Caching has a big impact on performance. If it is possible to reuse an RDD, caching increasing performance linearly.

Introduction

In this work, I analysed Google cluster-usage traces using Apache Spark and I created the document using R Studio.

You can access the subject of this work from this link. You can find the detailed documantatin of traces from this link.

This version of the code does not contains the Spark codes which are used to extract information from the traces also this is not the final version. There is no guarante related to correctness of the results.

Trace Analysis of the Cluster

1. Machine Distibution

In this section, I have showed the distribution of the machines according to CPU and memory capacity. Also, I have showed machine distribution according to paltform type but this information is not significant because machine distirbution according to platform type is same with according to CPU

Machine Distibution According to CPU and Memory Capacity

I have used machine events table to calculate machine distribution. In this table we have CPU and memory information of the machines. I have ordered data in decending order according to time then I have grouped by according to machine id. I get the first cpu and memory information from every group. So this machine distribution shows the final state of the cluster. It is possible to get slightly different machine distirbution according to ordering because cluster is not something static. It is changing with time. Graph 1 shows the machibe distribution according CPU and memory. From this graph it is easy to observe that more than 90% of the machines have 0.5 CPU and very few machines have 0.25 CPU.

Graph 1: Machine distribution according to CPU and Memory

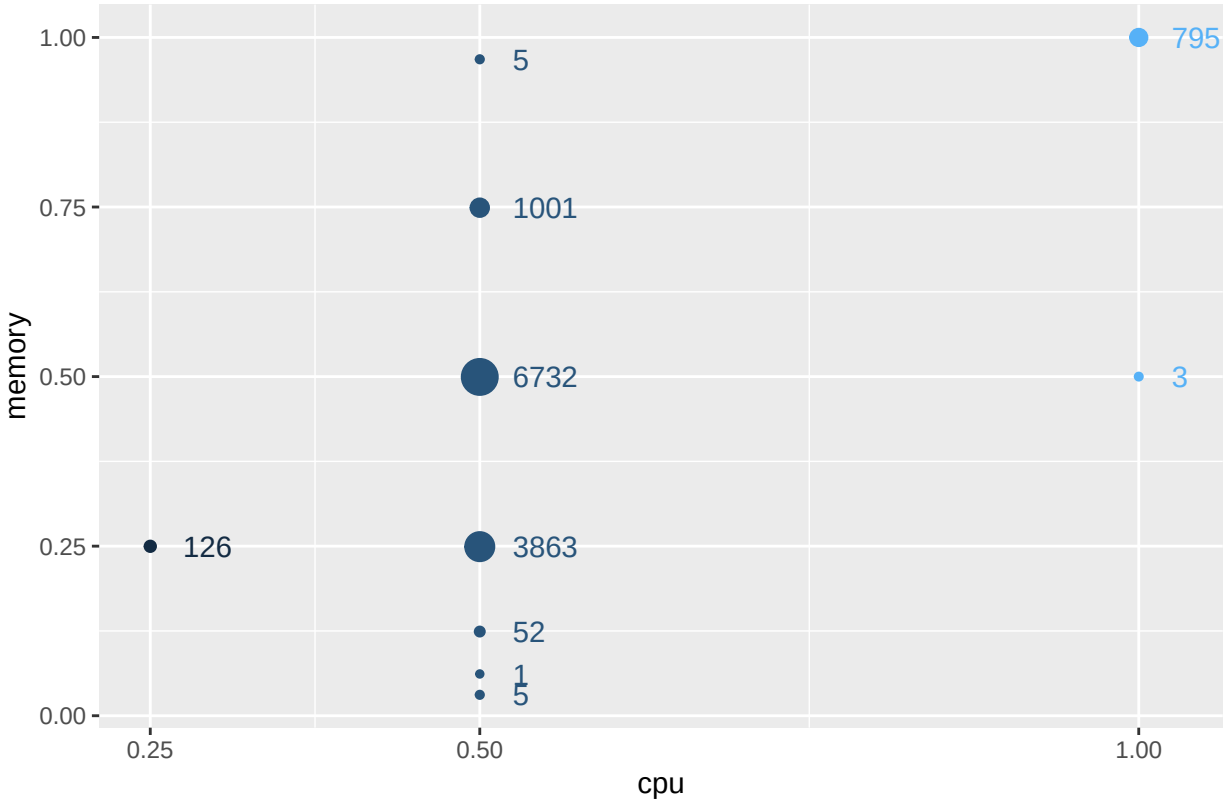


Table 1 show the same information with Graph 1 additionally it shows **capacity** metric and percentages of the machines in every capacity. I have created capacity metric. Capacity is average value of CPU and Memory capacity(Capacity = (CPU + Memory) / 2). This table shows the percentages of machines for every capacity. In next sections I will use this metric on plots.

Table 1: Machine distribution according to capacity

cpu	memory	capacity	number_of_machines	machine_percentage
1.00	1.00000	1.000000	795	6.32
1.00	0.50000	0.750000	3	0.02
0.50	0.96780	0.733900	5	0.04
0.50	0.74900	0.624500	1001	7.96
0.50	0.49950	0.499750	6732	53.50
0.50	0.24930	0.374650	3863	30.70
0.50	0.12410	0.312050	52	0.41
0.50	0.06158	0.280790	1	0.01
0.50	0.03085	0.265425	5	0.04
0.25	0.24980	0.249900	126	1.00

Machine Distibution According to Platform

Machine distribution acording to platform is not significant because it is same with CPU distirbution. On the documentation it is stated that two machines with same platform id can have very different CPU and memory resources. Table 2 shows the machine distribution acccording to platform.

Table 2: Machine distribution according to platform id

platform_id	number_of_machines
HofLGzk1Or/8Ildj2+Lqv0UGGvY82NLoni8+J/Yy0RU=	11659
GtXakjpd0CD41brK7k/27s3Eby3RpJKy7taB9S8UQRA=	798
70ZOvysYGtB6j9MUHMPzA2Iy7GRzWeJTdX0YCLRKGVg=	126

2. On average, how many tasks compose a job?s

I have used task events table to count the number of jobs and number of job per task. I have select distinct rows according to job_id and task_index then I grouped by according to job_id to count the number of task per job. Table 3 shows the total number of jobs, total number of tasks and average number of task per job.

Table 3: Job and task counts

number_of_jobs	number_of_tasks	avg_number_of_task_count
672,004	25,424,731	37.8342

Table 4 shows the statistics about number of tasks per job. From this statistics we can conclude that big part of the jobs contains only 1 task. Average number of task per job is **37.8342**. The job with biggest number of task has **90050** task.

Table 4: Summary statistics of the number of task per job

	statistics
Min.	1.0000
1st Qu.	1.0000
Median	1.0000
Mean	37.8342
3rd Qu.	1.0000
Max.	90050.0000

3. Job/Task that got killed or evicted

I have used job table and task event tables to count the number of evicted and killed jobs and tasks. Table 5 shows the figures related to job/task schedule, evicted and killed events.

Table 5: Job/task evicted killed count

type	scheduled_count	evicted_count	killed_count
Job	672,075	22	272,341
Task	47,351,173	5,864,353	103,496,805

4. Eviction Probability of Tasks with Respect to Priority

I have used task events table to calculate efiction percentages of tasks accordign to priority. I have count the number of events according to priority and than I have calculated percentages of events on every priority. Table 5 shows the percentages of events according to priority. In this table, percentages are calculated according to task counts. *task_count* column shows the number of **unique** task on every priority level.

When we analyse the data on the table we can not see any linear relation between priority of task and eviction counts however some priority levels have higher eviction percentages. On the documentation it had been stated that there are some special priority ranges : "free", "production" and "monitoring".

Table 6: Task event percentages according to priority of tasks

priority	killed_percent	evicted_percent	failed_percent	lost_percent	task_count
11	25.6	0.0	0.0	0.0	7,538
10	668.8	35.9	1,526.9	0.1	1,403
9	79.6	5.9	457.6	0.0	286,269
8	54.2	1.2	6.7	0.0	254,680
7	199.0	0.0	0.0	0.0	400
6	21.2	0.1	8.4	0.0	639,784
5	0.0	0.0	0.0	0.0	104
4	23.7	0.3	4.5	0.0	14,197,733
3	6.0	14.9	0.1	0.0	1,027
2	14.8	4.3	2.4	0.0	1,111,810
1	37.6	18.8	29.4	0.0	2,453,482
0	83.2	81.7	170.5	0.1	6,472,128

5. Relation between the Priority of a Task and Machine Resources(CPU, Memory)

Is there any relation between priority of tasks and machine resources which tasks are scheduled? I have used task events and machine events tables to answer this question. I have calculated distinct machine table from machine events and I have joined it with task events table than I have grouped by priority and *capacity* then I have count the number of tasked scheduled.

Graph 2 show a heat map show the relation. In y axis we have the priorities of the tasks and on the x axis we have the capacity of the machines which are tasked scheduled. On the cell's we have the percentages of tasks which are schedulet on that knd of machines. Table 6 shows number of tasks according to priority of task. Also Table 1 shows the percentages of the machines according to capacity. **When we consider all the information on the heat map and tables we can conclude that tasks are uniformly distributed on the machines. There is no relation between priority of task and capacity of the machines.**

For example, In the cluster, 53.5% of the machines have capacity of 0.49975(from Table 1). When we look at the values of the cell's on the heat map with capacit 0.49975 we can see that approximatly 55% of the tasks on all categories are scheduled on this kind of machines. We can do same observation for other capacities also.

Graph 2: Machine capacity and priority heat map

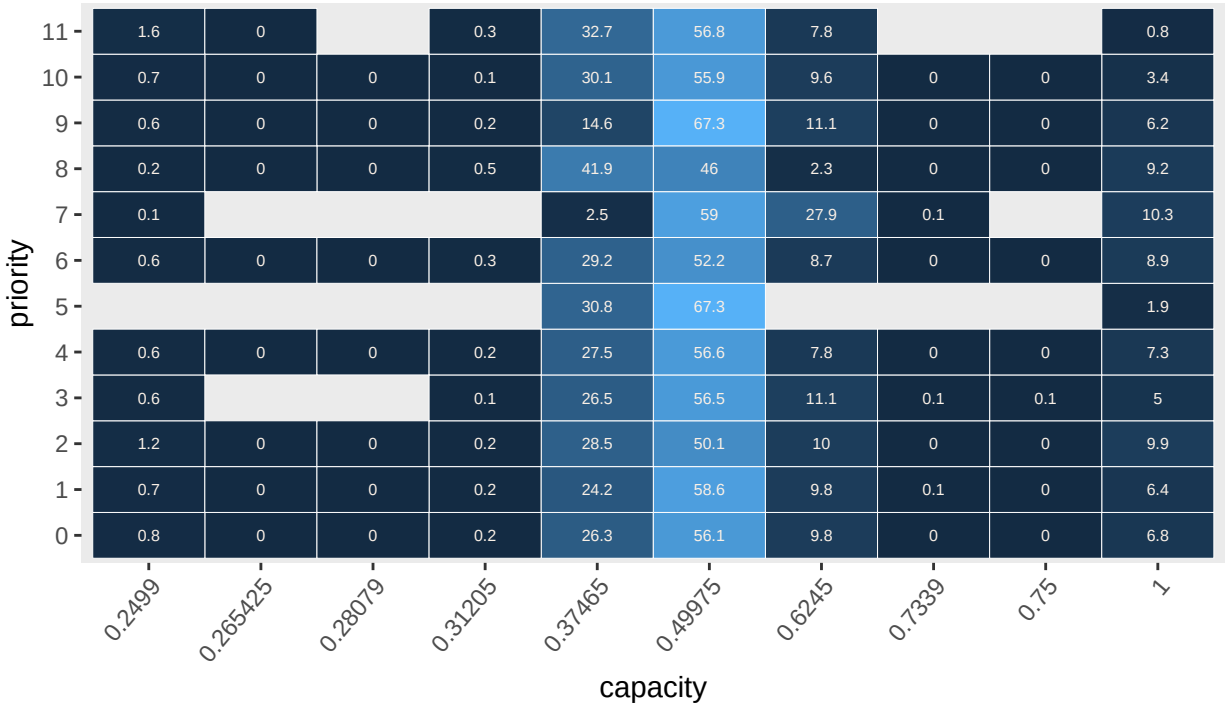


Table 6 shows the task distribution according to piority. In this table, task counts are not unique task counts because some tasks are rescheduled because of eviction or failure.

Table 7: Task counts according to priority of the task

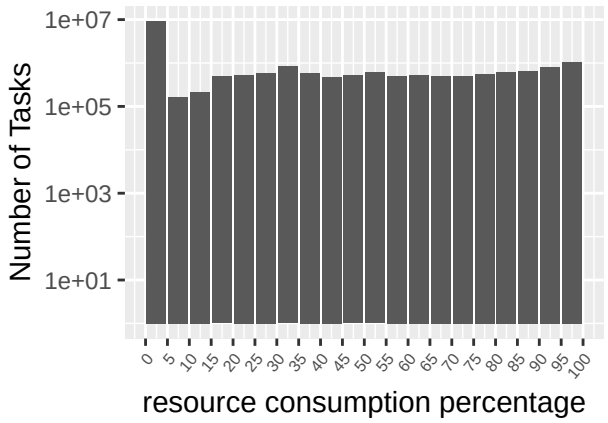
priority	total_task_count
0	24,640,306
1	3,745,699
2	1,200,766
3	1,203
4	15,099,442
5	104
6	689,599
7	796
8	274,962
9	1,656,760
10	32,128
11	9,405

6. Tasks consumes significantly less resource than what they requested

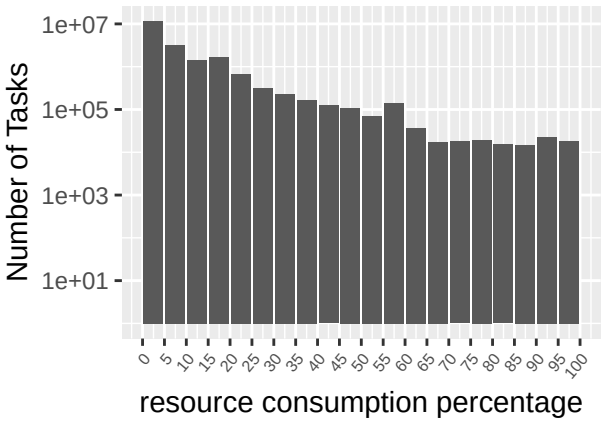
I have used special method to decide whether or not a task is using significantly less resource from what it requested. To decide this, I have looked at the tasks's maximum CPU and memory resource consumption. If both of the maximum values never reaches what it requested than I assumed that this task is using significantly less resource from what it requested. I have used **task events** and **task usage** tables to find the specific tasks. I have joined this two table then I have group by according to job_id and task_index and I have looked at the max velues of maximum cpu usage and maximum memory usage. Then I have decided whether or not a task using significantly less resource.

I have decided to create a histogram to have better understanding the resoruce consumption behavior of tasks which are using significantly less recoruces from what they requested. I have created two histogram. On the **X** axis I showed the resource consumption percentage according to what it requested on the **Y** I have showed the number of tasks. Graph 3 shows the memory consumption behavior and Graph 4 shows the CPU consumpsion behavior of the tasks which are usign significantly less resoruces from what they requested.

Graph 3: Tasks consumes significantly less memory



Graph 4: Tasks consumes significantly less CPU



7. Machine availability

I have conducted some analysis to have idea related to machine availibility. I think this analysis can be useful to answer the next question because

1- What is the average time between two consecutive machine remove event.

Average time between two consecutive remove event is *279.7* Seconds.

According to the documentation, machine remove events can occure because of failures or maintenance so we can conclude that every *279.7* second there is a failure or maintanence in the cluster.

2- Are there any machine which is removed from cluster and did not added to back.

97 of the machines are removed from cluster and they are not added back. This number could be helpful to make following analysis.

8. Machines which resources are largely underused

In the cluster, there are 36 machines wich highly underused. I have query all the machine usage data to find the machines which are under used.

Performance Analyses of Apache Spark

1. Efect of Number of Worker Thread

If there is enough input file to process than in this case number of worker threads effecting performance. If there is only one input file to process than number of worker threads has small performance effect.This Effect is not linear. For example to count all the distinct jobs in job tables with 8 worker thread takes 43 seconds. Same process takes 120 seconds with only one thread. For only one input file with single worker thread same job takes 15 seconds and with 8 worker thread it takes 14 seconds.

2. Efect of JVM Memory Size

Normally in the spark cluster there is two kind of process driver and executor.There is two different parameter to control the JVM memory size. These parameters are “spark.driver.memory” and “spark.executor.memory”. Normall this two parameter have different effect but in our case “spark.executor.memory” parameter is not useful because we are running Spark in local mode which means that driver process spawning executor threads to do the processing so there is only one JVM which is dirver JVM. Because of this reason I can only assess the effect of “spark.driver.memory” parameter.

When we increase the memory size of the driver process, we can easily observe via using system tools(system monitor, top, htop) driver JVM is instantiated with specified memory and it is using the reserved memory. Also for caching spark is using memory on default so if there is enough memory caching effect could be big incase of all the cached data small enoug to fit in memory.

During my study, I have java.lang.OutOfMemoryError incase of collecting huge amount of data at driver process. Arranging JVM memory size could solve this problem.

3. Effect of Lazy Evaluation

4. Effect of Cache

Cache has a big impect to performance. In normal conditions spark is not caching RDD's. Assume that you are making a calculation and you are counting the number of elements in the RDD later you tried to save the RDD to a csv file. Without caching Spark will calculate same RDD two times, once for counting once for writing to csv file. With cachign it will only calculate once and it will use the result of the same calculation for counting and writing to csv file.

At teh beginning of this study I was showing result of the RDD's and later I was tring to do further processing without caching. In this case, it was taking two times more time to finih the job. Later, I realized the effect of caching and I tried to used caching when it is necessary.

Conclution