SECOND GROUP WORK IN THE COURSE BIG DATA / ANALYSIS BIG DATA

Team Members:

Zervas Michalis (ics20015)

Drikos Christos(ics20046)

Steps - Problems - Workarounds

The steps we followed were first to better understand how Spark works and the commands we would need. Then relying mainly on the material we were given, we proceeded to compile the code. In the code, the problems we encountered were related to how the threshold would be calculated and how to store the results.

Regarding the threshold, at first we ran FP-growth with support=0 and then with the filter we left the results with frequency>=threshold. But because this implementation would create delays due to additional calculations and because it was not a good approach programmatically, we left it and set the support to be calculated based on the threshold and the total number of transactions.

Regarding how to store the results, the dataframe had to be transformed into a single column string so that it could be written to a csv file. Then, because the results were written in many small parts, we chose, to be more clear, to merge them all into one csv file. Also, when we tried to run it on 2 workers, some errors appeared in the spark output, which were ultimately due to the fact that the 2 vms had different python versions.

Finally, after solving these problems and arriving at the final form of the code, we proceeded to collect the results and all other necessary information requested for each threshold.

Times (http://83.212.80.243:8080/)

2 workers:

threshold: 5,000 :**min_time**=3.2 min,**max_time**=3.6 min,**avg_time**=3.4 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration
app-20230106170426-0008	FP-Growth approach	16	1024.0 MiB		2023/01/06 17:04:26	user	FINISHHED	3.2 min
app-20230106170820-0009	FP-Growth approach	16	1024.0 MiB		2023/01/06 17:08:20	user	FINISHHED	3.6 min
app-20230106171346-0010	FP-Growth approach	16	1024.0 MiB		2023/01/06 17:13:46	user	FINISHHED	3.3 min

threshold: 10,000 :**min_time**=3.1 min,**max_time**=3.5 minutes,**avg_time**=3.3 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration
app-20230106164453-0004	FP-Growth approach	16	1024.0 MiB		2023/01/06 16:44:53	user	FINISHHED	3.4 min
app-20230106165022-0005	FP-Growth approach	16	1024.0 MiB		2023/01/06 16:50:22	user	FINISHHED	3.1 min
app-20230106165812-0007	FP-Growth approach	16	1024.0 MiB		2023/01/06 16:58:12	user	FINISHHED	3.5 min

threshold: 50,000 :**min_time**=3.2 min,**max_time**=3.3 min,**avg_time**=3.2 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration
app-20230106152748-0000	FP-Growth approach	16	1024.0 MiB		2023/01/06 15:27:48	user	FINISHHED	3.2 min
app-20230106154131-0001	FP-Growth approach	16	1024.0 MiB		2023/01/06 15:41:31	user	FINISHHED	3.2 min
app-20230106163215-0002	FP-Growth approach	16	1024.0 MiB		2023/01/06 16:32:15	user	FINISHHED	3.3 min

1 worker:

threshold: 5,000 :**min_time**=1.4 min,**max_time**=1.5 minutes,**avg_time**=1.4 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration

app-20230106172016-0011	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:20:16	user	FINISHED	1.4 min
app-20230106172300-0012	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:23:00	user	FINISHED	1.4 min
app-20230106172454-0013	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:24:54	user	FINISHED	1.5 min

threshold: 10,000 :**min_time**=1.4 min,**max_time**=1.5 minutes,**avg_time**=1.4 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration
app-20230106172728-0014	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:27:28	user	FINISHED	1.4 min
app-20230106173114-0015	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:31:14	user	FINISHED	1.5 min
app-20230106173309-0016	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:33:09	user	FINISHED	1.4 min

threshold: 50,000 :**min_time**=1.3 min,**max_time**=1.4 min,**avg_time**=1.3 min

Application ID	Name	Cores	Memory per Executor	Resourcees Per Executor	Submitted Time	User	State	Duration
app-20230106173624-0017	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:36:24	user	FINISHED	1.3 min
app-20230106174117-0018	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:41:17	user	FINISHED	1.4 min
app-20230106174318-0019	FP-Growth approach	8	1024.0 MiB		2023/01/06 17:43:18	user	FINISHED	1.4 min

Commenting on years:

We notice that executions with 1 worker require about half the time of those with 2 workers. This is not something expected, as we would expect with 2 workers to share the computing load and have faster executions. This may be due to the network overhead or the coalesce(1) command which collects all the results of each worker in a file, so as the number of workers increases, the time to collect them may also increase.