General Description:

The following instructions are to reproduce the results from ***COMPASS: Online Sketch-based Query Optimization for In-Memory Databases***
https://dl.acm.org/doi/10.1145/3448016.3452840

NOTE #1, the experiments have been moved to docker image to ease reproducibility. Therefore, you may expect similar (not exact) results. One could improve the results further by reproducing in more powerful environment.

NOTE #2: The provided results and instructions were run on Join Order Benchmark (JOB). However, one could also repeat the experiments on JOB-light workload with a bit more changes in the bash and python scripts.

----------DOCKER SETUP----------

Enabling Non-root Users to Run Docker Commands: *sudo usermod -aG docker $USER*

1. **COMPASS/MAPD docker container setup:**

- Pull docker image: *docker pull yizenov/reproduced_compass*
- Create docker container from image: *docker run -it --name compass_docker --gpus device=0 --runtime=nvidia --shm-size 64gb -w="/home/" reproduced_compass:latest*
- From inside the docker container run the command: *exit*
- Start docker container: *docker start compass_docker*
- Access docker container: *docker exec -it compass_docker bash*
- Start MapD: *will be started in the following scripts. No action is needed.*

2. **PostgreSQL docker container setup:**

- Pull docker image: *docker pull adatta2/pg_docker*
- Create and run docker container from image: *docker run --name pg_docker --shm-size 64gb -w="/home/pg_experiments" -d adatta2/pg_docker*
- Access docker container: *docker exec -it pg_docker bash*
- Start PostgreSQL: *Will automatically start within 20-30 Seconds. No action is needed.*

3. **MonetDB docker container setup:**

- Pull docker image: *docker pull adatta2/monetdb_docker:latest*
- Create and run docker container from image: *docker run --name monetdb_docker --shm-size 64gb -w="/home/monetdb/monetdb_experiments" -d adatta2/monetdb_docker*
- Access docker container: *docker exec -it monetdb_docker bash*
- Start MonetDB: *monetdbd start /home/monetdb/DB_FARM/*

----------METHODOLOGY----------

Experiments on the JOB workload (i.e. 113 queries):

1. run all 113 queries and collect join order plans for the following systems:
    a. COMPASS (PART #1, PART #2, PART #3, PART #5)
    b. MAPD (PART #1, PART #3, PART #6)
    c. PostgreSQL (PART #20)
    d. MonetDB (PART #8)
2. run all 113 queries and collect runtimes for the following systems:
    a. COMPASS (PART #1, PART #2, PART #3, PART #9)
    b. MAPD (PART #1, PART #3, PART #10)

3. compute costs for all plans collected from all systems. This step is performed inside PostgreSQL's container (PART #21).
4. parse runtimes for all plans collected from all systems. This step is performed inside PostgreSQL's container (PART #22).

5. generate Figures 6 and 7. (PART #23).
6. Table #2 is updated after STEP #5.
   See *runtime_cardinality_figures.ods* → Runtime tab → Scroll to right [Table 'Total Runtime (Minutes)']

7. combine runtime results to generate Figures 8 (a) and (b). The result is stored in python_results/collected_all_runtime_data.csv. (PART #11)
8. the results from step #5 is copied into collected_job_runtime_figure.ods to update the Figures.
9. Table #3 is updated after STEP #8.
   See *collected_job_runtime_figure.ods* → Scroll to right [Table 3 'JOB benchmark execution in MapD']

10. run all 70,407 subqueries for the following systems:
    a. COMPASS (estimates only, PART #12)
    b. PostgreSQL (estimates and true cardinalities are already pre-computed)
11. parse estimates for all 70407 subqueries for the following systems:
    a. COMPASS (PART #14)
    b. PostgreSQL (PART #13, PART #15)

12. generate Figure 5 (a). The results are stored in python_results/*.pdf files. (PART #16, PART #17)

13. (Optional). to update sketch templates and seeds (PART #4, PART #3)
14. (Optional). to compare cardinalities of base tables with selection predicates. (PART #18)

----------DETAILS----------

PART #1. To run queries just collect plans or/and execute queries:
plan only: change '*ONLY_GET_PLAN=true*' in /home/mapd-core/Catalog/Catalog.h (line 641)
execution: change '*ONLY_GET_PLAN=false*' in /home/mapd-core/Catalog/Catalog.h (line 641)
compile the server by running the following command: *(cd /home/mapd-core/build/ && make -j $(nproc))*

PART #2. To change between Search Algorithms (Greedy=1 or Exhaustive=500):
change '*NODE_TRAVERSE_BOUND=value*' accordingly in /home/mapd-core/Catalog/Catalog.h (line 639)
compile the server by running the following command: *(cd /home/mapd-core/build/ && make -j $(nproc))*
NOTE: The results of Greedy Search Algorithm have been reported in the paper.

PART #3. To run workload queries from /home/:
NOTE: pre-computed results have already been provided in the docker image
Please provide all 5 argument values:
-i : Name of folder in /home/QUERIES/ where input queries are located.
-o : Name of folder in /home/RESULTS_COMPASS/ or /home/RESULTS_MAPD/ where input queries are located.
-p : Choose processor type (1=GPU, 2=CPU).
-c : Choose optimizer (1=COMPASS, 2=MapD).
-t : Set timeout time (e.g. 10s or 5m or other). Otherwise, default timeout 5 minutes is applied.

```
screen -S mapd_run -dm -L sh -c 'time /home/mapd-core/my_scripts/./benchmark_compass_timed.sh -i input_folder -o output_folder -p value1 -c value2 -t 5m'
```

NOTE: the script logs are recorded in /home/screenlog.0.
one may want to clean the logs between script runs.

NOTE: a query is stopped if the query doesn't finish within 5 minutes (adjustable parameter).

PART #4. To run specific queries to update the template sketches and seeds:

change 'ONLY_GET_PLAN' accordingly in /home/mapd-core/Catalog/Catalog.h (line 641)
change 'PRE_PROCESSING' accordingly in /home/mapd-core/Catalog/Catalog.h (line 638)
change 'CAT_SKETCH_BUCKETS' accordingly in /home/mapd-core/Catalog/Catalog.h (line 630)
change 'CAT_SKETCH_ROWS' accordingly in /home/mapd-core/Catalog/Catalog.h (line 631)
compile the server by running the following command: (cd /home/mapd-core/build/ && make -j $(nproc))

```
-i  Input_folder = QUERIES/SKETCH_TEMPLATE_QUERIES
-o Output_folder = RESULTS_COMPASS/SKETCH_TEMPLATE_QUERIES_RESULTS
-p Processor type = CPU
-c Optimier type = COMPASS
-t Timeout = 5m
```

PART #5. To parse COMPASS' plan results:

Run the following command: /usr/bin/python3 /home/python_parsers/job_plans_compass_parser.py arg1
Script requires one argument:
a) Plan Search Algorithm: (1 = Greedy, 0 = Exhaustive)

PART #6. To parse MAPD's plan results:

Run the following command: /usr/bin/python3 /home/python_parsers/job_plans_mapd_parser.py
The script requires no arguments.

PART #8. To parse MonetDB's plan results:

Follow Docker #3 to access MonetDB container. Make sure $pwd is "/home/monetdb/monetdb_experiments"
a) Generate execution plans for 113 JOB queries: *python3 export_plans.py*

NOTE: All plans will be in the /home/monetdb/monetdb_experiments/PLANS folder.
NOTE: We don't have a script to extract join orders from MonetDB logs.
We manually collected plan /home/monetdb/monetdb_experiments/monetdb_plans.txt.

PART #9. To parse COMPASS' runtime results:

Run the following command: /usr/bin/python3 /home/python_parsers/job_runtime_compass_parser.py arg1 arg2
Script requires two arguments:
a) Plan Search Algorithm: (1 = Greedy, 0 = Exhaustive)
b) Processor type: (1 = CPU, 0 = GPU)

PART #10. To parse MAPD's runtime results:

Run the following command: /usr/bin/python3 /home/python_parsers/job_runtime_mapd_parser.py arg1
Script requires one argument:
a) Processor type: (1 = CPU, 0 = GPU)

PART #11. To collect data for Figures 8 (a) and (b):

Run the following command: /usr/bin/python3 /home/python_parsers/collect_runtime_figures.py
Script requires no arguments.

PART #12. To run all 70,407 subqueries to collect estimated and true cardinalities:

NOTE: pre-computed results have already been provided in the docker image

Follow PART #3 with the following parameters:

-i Input_folder = QUERIES/JOB_SUBPLANS_QUERIES
-o Output_folder = RESULTS_COMPASS/JOB_SUBPLAN_RESULTS_EST
-p Processor type = GPU
-c Optimier type = COMPASS
-t Timeout = 5m

PART #13. To parse PSQL's estimated cardinalities:

NOTE: pre-computed results have already been provided in the docker image

Run the following command: /usr/bin/python3 /home/python_parsers/job_subplans_psql_est_parser.py arg1

Script requires one argument:

a) Wokrload Type: (1 = JOB-light, 0 = JOB)

NOTE: JOB-light workload is not supported.

PART #14. To parse COMPASS' estimated cardinalities:

Run the following command: /usr/bin/python3 /home/python_parsers/job_subplans_compass_est_parser.py arg1

Script requires one argument:

a) Wokrload Type: (1 = JOB-light, 0 = JOB)

NOTE: JOB-light workload is not supported.

PART #15. To parse PSQL's true cardinalities:

Run the following command: /usr/bin/python3 /home/python_parsers/job_subplans_psql_true_parser.py arg1

Script requires one argument:

a) Wokrload Type: (1 = JOB-light, 0 = JOB)

NOTE: JOB-light workload is not supported.

PART #16. To generate Figure 5 (a):

Run the following command: /usr/bin/python3 /home/python_parsers/estimations_parser.py arg1 arg2

Script requires two arguments:

a) Optimizer type: (1 = COMPASS, 0 = PostgreSQL)
b) Include all joins: (1 = only up to 10-way joins, 0 = all joins)

PART #18. To compare true cardinalities of base tables with/without selection predicates:

-i Input_folder = QUERIES/JOB_SELECTION_QUERIES
-o Output_folder = RESULTS_COMPASS/JOB_SELECTION_QUERIES_RESULTS
-p Processor type = GPU
-c Optimier type = COMPASS
-t Timeout = 5m

PART #20. Parse PostgreSQL join order:

Follow **Docker #2** to access PostgreSQL container. Make sure $pwd is "/home/pg_experiments"

a) Generate execution plans for 113 JOB queries: *python3 scripts/export_plans.py*
   All plans will be in results/PLANS/ folder.
b) Parse join orders from the log files generated at step (a): *python3 scripts/parse_psql_join_order.py*
   All parsed join orders will be saved in /home/pg_experiments/results/ALL_PLANS/postgresql_plans.csv

PART #21. Calculate plan costs for all the systems:

a) Copy join orders from COMPASS, MapD, MonetDB, and PostgreSQL to "/home/pg_experiments/results/ALL_PLANS" inside PostgreSQL's container.

Example:
COMPASS:
Docker to host: docker cp compass_docker:/home/python_results/collected_plans-job-compass-greedy.csv .
Host to pg_docker: docker cp collected_plans-job-compass-greedy.csv pg_docker:/home/pg_experiments/results/ALL_PLANS
MapD:
Docker to host: docker cp compass_docker:/home/python_results/collected_plans-job-mapd.csv .
Host to pg_docker: docker cp collected_plans-job-mapd.csv pg_docker:/home/pg_experiments/results/ALL_PLANS
MonetDB:
Manually collected plan is available inside /home/pg_experiments/results/ALL_PLANS

b) Calculate plan costs (i.e. sum of intermediate subqueries cardinalities) for COMPASS, MAPD, MonetDB, and PostgreSQL.
python3 scripts/get_cardinality.py

The result will be located in **results/cadinlities.csv**.

PART #22. Parse runtimes for all 113 queries:

a) Generate queries with EXPLICIT join order for COMPASS, MAPD, PostgreSQL, and MonetDB.
python3 scripts/sql_generator.py (Output Folder: queries/JOB_EXPLICIT_QUERIES/)
b) Collect runtime for all queries (TOTAL: 113 queries * 4 systems). Each query is run 5 times, and the timeout is 5 minutes.
python3 scripts/collect_runtime.py (Output folder: results/RUNTIME/)

PART #23. To generate Figures 6 and 7:

Follow **Docker #2** to access the PostgreSQL container. Make sure $pwd is "/home/pg_experiments"

a) Create Figure 6 (Cardinality ratio) and 7(a), 7(b)
Copy results from PART #21 in runtime_cardinality_figures.ods file (cardinalities tab) to generate the figure.

NOTE: Failed queries (because of join orders involving cross joins, nested bushy trees, etc.) will be replaced with 'inf'. Plan costs for queries with nested bushy trees can be calculated using *scripts/get_cardinality_manual.py* which runs in two stages.

- 1st stage - takes query name (1a, 10c, 29b, etc.) as input argument.
- 2nd stage - takes join orders as input and adds them incrementally. For example, to calculate *t (mc ci) n* cost- we need to enter join orders in the following sequence: 1) mc ci 2) t mc ci 3) t mc ci n

b) Create Figures 6 (Runtime ratio) and 7(c), 7(d)
Copy results from PART #22 in runtime_cardinality_figures.ods file (runtime tab) to generate the figure.

NOTE: Copy the last column, which is the median of five runs, and paste them to the runtime_cardinality_figures.ods file.