Ira A. Fulton Schools of Engineering
Arizona State University
Engineering | School of Computing, Informatics, and Decision Systems Engineering

CSE 511: Data Processing at Scale

Parallel Sort and Parallel Join

Introduction:

The required task is to build a generic parallel sort and parallel join algorithm.

1. Implement a Python function ParallelSort() that takes as input: (1) InputTable stored in a PostgreSQL database, (2) SortingColumnName the name of the column used to order the tuples by. ParallelSort() then sorts all tuples (using five parallelized threads) and stores the sorted tuples in a table named OutputTable (the output table name is passed to the function). The OutputTable contains all the tuple present in InputTable sorted in ascending order.

For more information on Psycopg, click here.

Function Interface

ParallelSort (InputTable, SortingColumnName, OutputTable, openconnection)

- InputTable – Name of the table on which sorting needs to be done.
- SortingColumnName – Name of the column on which sorting needs to be done, would be either of type integer or real or float. Basically numeric format. Will be sorted in ascending order.
- OutputTable – Name of the table where the output needs to be stored.
- openconnection – Connection to the database.

2. Implement a Python function ParallelJoin() that takes as input: (1) InputTable1 and InputTable2 table stored in a PostgreSQL database, (2) Table1JoinColumn and Table2JoinColumn that represent the join key in each input table respectively. ParallelJoin() then joins both InputTable1 and InputTable2 (using five parallelized threads) and stored the resulting joined tuples in a table named OutputTable (the output table name is passed to the function). The schema of OutputTable should be similar to the schema of both InputTable1 and InputTable2 combined.

Function Interface:

ParallelJoin (InputTable1, InputTable2, Table1JoinColumn, Table2JoinColumn, OutputTable, openconnection)

- InputTable1 – Name of the first table on which you need to perform join.
- InputTable2 – Name of the second table on which you need to perform join.
- Table1JoinColumn – Name of the column from first table (i.e. join key for first table).
- Table2JoinColumn – Name of the column from second table (i.e. join key for second table).
- OutputTable – Name of the table where the output needs to be stored.
- openconnection – Connection to the database.

Please use the function signature exactly same as mentioned in Assignment3_Interface.py. You will notice that in top of Assignment3_Interface.py, the following things are declared:

- FIRST_TABLE_NAME = 'table1'
- SECOND_TABLE_NAME = 'table2'
- SORT_COLUMN_NAME_FIRST_TABLE = 'column1'
- SORT_COLUMN_NAME_SECOND_TABLE = 'column2'
- JOIN_COLUMN_NAME_FIRST_TABLE = 'column1'
- JOIN_COLUMN_NAME_SECOND_TABLE = 'column2'

Example:

Let's use an example to understand these fields and their usage.

Example: Once your implementation is complete, you will have to create two tables in database manually. Name them *MovieRating* and *MovieBoxOfficeCollection*. Suppose you want to sort MovieRating by column Rating and MovieBoxOfficeCollection by column Collection. You also want to join MovieRating and MovieBoxOfficeCollection by column MovieID. Then, you would define the variables mentioned above as:

- FIRST_TABLE_NAME = 'MovieRating'
- SECOND_TABLE_NAME = 'MovieBoxOfficeCollection'
- SORT_COLUMN_NAME_FIRST_TABLE = 'Rating'
- SORT_COLUMN_NAME_SECOND_TABLE = 'Collection'
- JOIN_COLUMN_NAME_FIRST_TABLE = 'MovieID'
- JOIN_COLUMN_NAME_SECOND_TABLE = 'MovieID'

Naming Conventions:

The naming convention is to be strictly followed:

- Database name: dds_assignment
- Postgres user name: postgres

- Postgres password: 1234

Instructions on How Assignment will be Tested:

Please follow these instructions closely.

1. Two tables will be created in the database manually.
2. The created tables will contain at least an integer field, which would be used for both Parallel Sorting and Parallel Joining.
3. Then, the ParallelSort() and ParallelJoin() Function will be called to check the correctness of the program.
4. Your code should use 5 threads for both ParallelSort() as well as ParallelJoin().
5. Your code should be able to handle table irrespective of its schema.
6. Do not make your code dependent on any particular table; it should be able to work on any table.

Assignment Tips:

Please follow these instructions closely or else points will be deducted.

1. Please follow the function signature as provided in the Interfacy.py.
2. Please use the same database name, table name, username and password as provided in the assignment to keep it consistent.
3. Please make sure to run the file before submitting and make sure there is no indentation error. In case of any compilation error, 0 marks will be given.
4. Do not modify any function signature in Interface.py. In case any modification is needed, please post the same on discussion board.
5. Any print function you add to your Interface.py will be suppressed in the grader feedback.

Submission:

Upload the completed Interface.py to Coursera. Do not upload *.zip files.

Note: Failure to follow the instructions provided in the document will result in the loss of points.

We provide a virtual machine that has the testing environment and an installed PostgreSQL.

- OS username: user
- OS password: user
- Postgres username: postgres
- Postgres password: 1234

You can download it and use any VM software such as VirtualBox to import it:
https://drive.google.com/file/d/1EBImGZmqDQ8XGTuiHPoqP7XE2ZykAXkX/view?usp=sharing

You will need the following files in your assignment:

1. Interface.py: You need to implement your code inside this file.
   Interface.py
2. tester.py: You can use this tester to test your code.
   tester.py
3. movies.txt and ratings.txt: Some test data.
   movies.txt
   ratings.txt
4. Assignment1.py: The correct answer of Assignment 1 with slight modifications.
   Assignment1.py

Feedback:

There are a total of two test cases for a total of 20 points, so each test case is worth 10 points. We will run one test case for "ParallelSort()" and one test case for "ParallelJoin()". **If one of the functions fails, you will see the corresponding error logs that indicate *where* the error occurred.**

Test cases are not executed in parallel. You have to unlock all previous test cases to execute the current test case. For example, to execute test case 2 your code must be able to successfully execute test case 1. The test case number in the feedback will provide you an indication of which part of the submission caused the error.