CSE603 Assignment3

Ruhan Sa 50060400

November 30, 2012

Aim: Compute the product of matrix A and matrix B. Result stored into C using Netezza.
Verification: This is done using matrix with unit elements and the result is verified by checking wether elements of C equals to the matrix size or not. If so the implementation is verified, if not the implementation is wrong.
Script: All the compilations and SQL scripts that may have used is attached in source code as comment.

1 Source code explanation

1.1 Input function

Both A and B are generated using UDTF functions Ain(int, int) and Bin(int, int), where matrix size and input routines (0 or 1) are taken as first and second input parameters respectively. 0 indicates the verification input routine, where the elements are chosen as unity, and 1 indicates the computational experimental input routine, where the matrix element is chosen as random numbers ranges from 0 to 2. SPU distribution on row numbers and column numbers are used for A and B to speed up the result. Figure 1 shows the table layout of A and B.

1.2 Output function

The time taken to do multiplication is done using SQL command line – time. The multiplication of the matrix is done by UDTF function rout(int, int, varchar, varchar), where first two parameters denote the row and column number, last two parameter denote the row and column element needed from A and B. The element of A and B are passed into UDTF function parameters.

2 Experimental result

2.1 Time consumption with the increase of matrix size

Figure 2 shows the time consumption of NETEZZA with the increase of matrix size. The time consumption is computed as the sum of matrix input time and

```
RUHANSA_DB(RUHANSA)=> CREATE TABLE A1 AS SELECT * FROM TABLE(Ain(0, 10));
INSERT 0 10
RUHANSA_DB(RUHANSA)=> CREATE TABLE B1 AS SELECT * FROM TABLE(Bin(0, 10));
INSERT 0 10
RUHANSA_DB(RUHANSA)=> SELECT * FROM A1;
 ROWNUM |                    AVAL
--------+-----------------------------------------
      7 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      3 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      9 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      2 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      5 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      4 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      8 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      6 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
     10 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      1 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
(10 rows)

RUHANSA_DB(RUHANSA)=> SELECT * FROM B1;
 COLNUM |                    BVAL
--------+-----------------------------------------
      2 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      7 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      6 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      9 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      1 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      3 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      5 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      4 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
     10 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
      8 | 1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0;1.0
(10 rows)
```

Figure 1: Table layout

matrix output time. Matrix input time indicates the time of creating matrix A, B and the time for creating Tables for each of the matrix in database. Matrix output time indicates the time taken to do matrix multiplication and create table C to store the result into database.

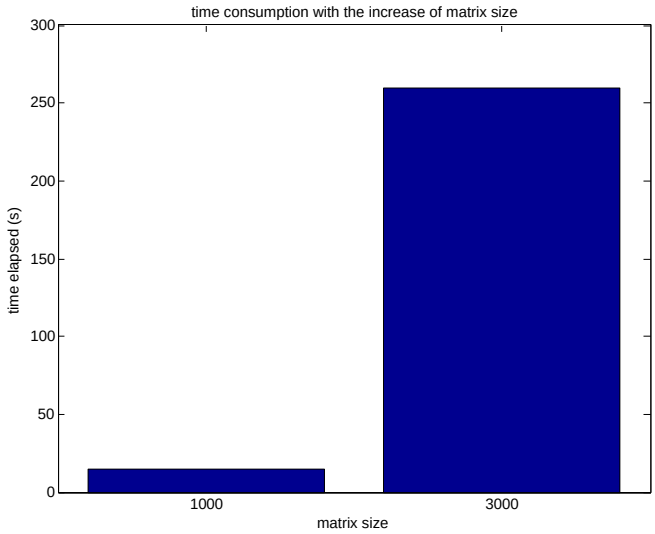

Figure 2: Time consumption with two different matrix size.

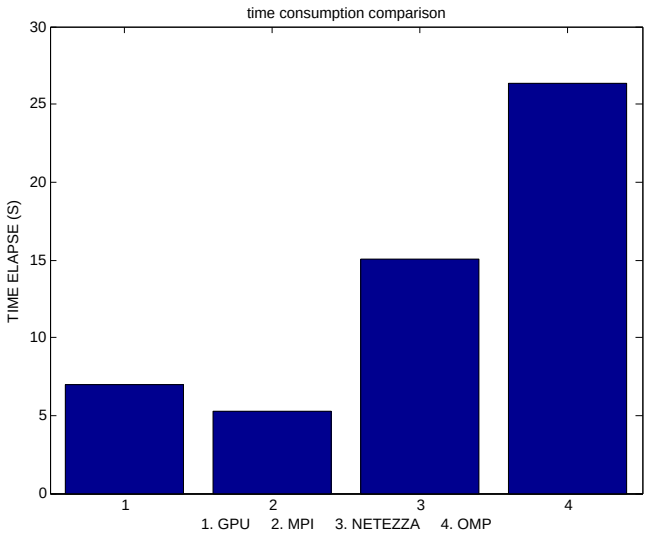

Figure 3: Time consumption comparison with MPI, OMP and GPU.

2.2 Comparison with MPI, OpenMP and GPU

Figure 3 shows the performance comparison with OpenMP, MPI and GPU when computing matrix size of 1000×1000. Here the MPI performance was observed when using 20 processors, OpenMP performance was observed using 20 processors, the *gpu* bar indicates the best performance chosen from last assignment.