**Cloud Computing Assignment-2(Design Doc)**

**Distributed Database on MPI:**

We implemented select and join commands of sql database on top of a flat file on MPI with int and string datatypes allowed( as mentioned in the problem statement).

Constraints:

1. Only one condition is allowed in select query. Two or more conditions combined with logical operators is not supported.
2. Number of columns are restricted to 5 in the table
3. Currently only three tables are supported (student,department,person\_rec).
4. Metadata file containing mapping from column name to column index should be present in the shared directory.
5. Queries are case-sensitive.
6. For join query the column on which join is applied should be first column of both the tables.(should be integer).

|  |  |
| --- | --- |
| Operator | Function |
| = | Integer equal to |
| > | Integer greater than |
| & | String not equal to |
| < | Integer smaller than |

Syntax for queries:

1. Select query:

Select <col\_name> from <table> where <condition>

1. Join query:

Join <table1>,<table2> on <common\_attribute>

Method Used:

**Select :**

According to number of processors available a metadata is created dynamically which has three attributes:

<Rank, Offset, No\_records\_to\_read>

All processors use this metadata to find the offset in the file from where they have to read data and no of records tell them how much data they have to read.

So each processor reads data from this metadata in parallel and reads the part of file assigned to it.

Then based on the where condition each processor filters out the tuples which do not satisfies the where condition.

Then finally a common view is created and presented to the user which has all the tuples satisfying the query.

**Join:**

In join program will create metadata file for that table which has more size among the two. Then based on this file (as done in select) pieces of file is distributed across multiple process. For the other file, each process reads the complete file and based upon where condition records are filtered. Then finally a common view is created and presented to the user which has all tuples satisfying the query.

**Hadoop:**

**Assumptions:**

1. **Query is in standard Sql syntax:**
   1. **Syntax:**

**Select <col\_name> from table where <condition>.**

1. **In condition operator allowed for integer data type is “=”**

**And operator for string type is “&”.**

1. **Table is assumed to be of 5 columns.**
2. **The table name specified in the select query is assumed to be present in a directory in hdfs by the same name as specified in the query**

In Hadoop mapper is passed the flat file in form of key, value pair.

Key is the offset in the file and value is a tuple from the database storage.

In mapper key emitted to intermediate stage is taken as row number of tuple. And individual field values are emitted as keys if they satisfy the where condition.

In reducer data comes after filtering.

Corresponding to key values(individual field values) are collected and then written on output file in hdfs.

**Commands to run MPI and Hadoop programs:**

1. **MPI**
2. (Assuming .out file)

Mpirun –n <no\_of\_processes> --hosts <ip\_addresses> ./<>.out “<query>”

**2.Hadoop**

**Assuming jar of class files and input file present in hdfs**

Hadoop jar <jar\_name>.jar /path\_to \_outdir\_hdfs “<query>”

P.S: Assuming queries according to the constraints mentioned above.

**Data Set used:**

We generated the dataset randomly with number of tuples varying from 10K to 1000K.

The schema of the tables is:

1. student (id,name,branch,subject,marks)
2. department(id, name, branch, subject, marks**)**

**Submitted by:**

**Pooja garg(2014H103100P)**

**Shashank Gupta(2014H112198P)**