**PARALLEL AND DISTRIBUTED COMPUTING**

***PROJECT\_REVIEW-2***

**QR FACTORIZATION OF LARGE-LEAST SQUARE NUMBERS WITH HADOOP**

**TEAM MEMBERS:**

Himansh Jain(15BCB0016)

Aaditya Vaidya(15BCB0032)

Dev Kumar(15BCB0060)

Yash Shah(15BCB0123)

**GUIDED BY:**

Prof. Manoov R

**PROJECT DETAILS**

**Abstract**

A new algorithm is presented to provide an efficient solution to most of the square problems, in which the linear mechanism's multiplication of matrix is ​​two small dimensions Matroso's CraneCare product. The solution algorithm is based on the QR terminology of the small dimension matrix. Full load equilibrium is achieved by absorbing the property of 'Crookcrake' property, and the communication needs can be reduced by using the binary exchange algorithm for matrix transposition. Parallel algorithms are presented, and timing results are shown on test runs on the Intel i860 computer.

**Introduction**

QR Factorization is used n to solve linear equations ,least squares problems

And constrained least squares problems

In this project we will try to implement a parallel algorithm to solve the least square problem (full order)

(AQB) x = t, Where AE Mora, Page and B E MN, Q, Sequence (A) = P, Sequence (B) = q, and xE ~ RM, TE ~

According to the pre-existing research a new method for halting based on A and B curare disintegration. Here we describe an improved algorithm which needs to be compiled in one of the R-matrices and discusses its implementation on the Intel i860 computer. Improved algorithm uses Crankker's product 'Comutability' property. A pair of high triangular systems is produced, from which at least squares can be solved by backside of the two systems in parallel. It has significantly less communication overhead than the version of algorithm given in a better computational load equilibration and.

**Plan**

We are going to use Kronecker product large-least squares algorithm for parallelization to improvise the QR-factorization problem in Hadoop.

**Algorithm**

I. Compute the vector sr according to Sr = (QIIt)~ ~ 13)tr;

2. Compute the vector hr according to hr = (I~ ® QI2)' )s7;

3. Block back substitute to obtain the solution vector zr to the system (I] ® RC2))Zr = hr;

4. Form the vector z from the vector ZT, i.e., perform a matrix transpose operation on zT;

5. Block back substitute to obtain the solution vector y to the system (12 ® RC~))y = z;

6. Repermute the vector yr according to .qr= (1~ ® P2)y7;

7. Compute the vector dr according to d7.--(I~ ® A),qT.;

8. Form the vector d from the vector dr, i.e., perform a matrix transpose operation on DT;

9. Repermute the vector d according to f = (13 ® Pj )d;

10. Compute the residual vector r according to r = t - (13 ® B)f.

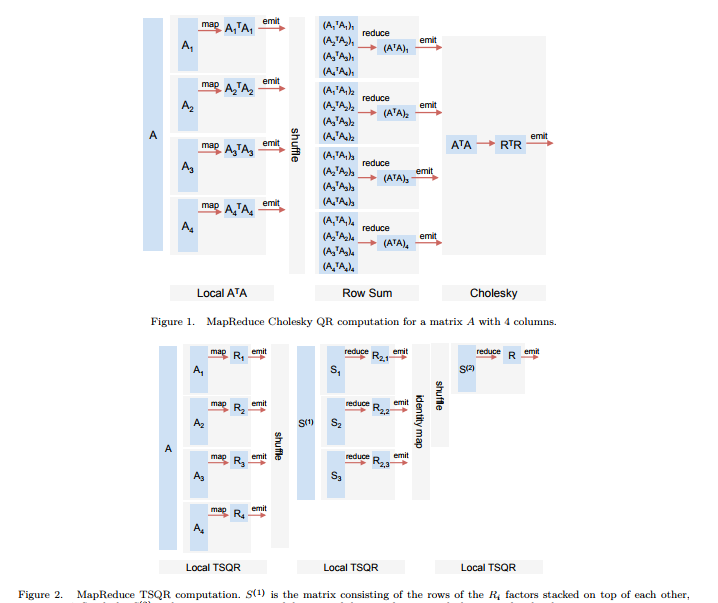
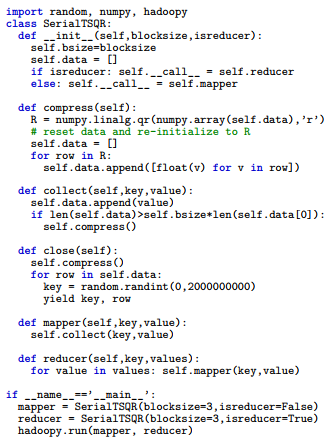
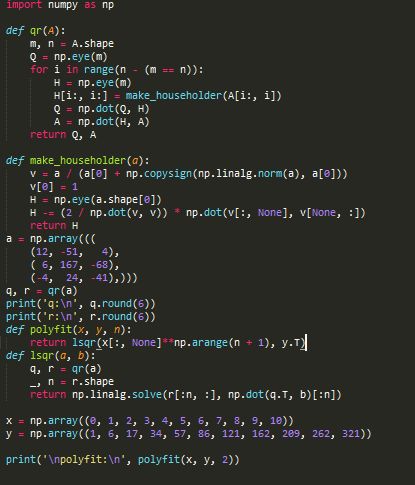
****

FIG: TSQR MapReduce

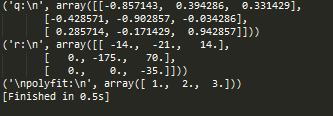
**Psedo Code Snippet**



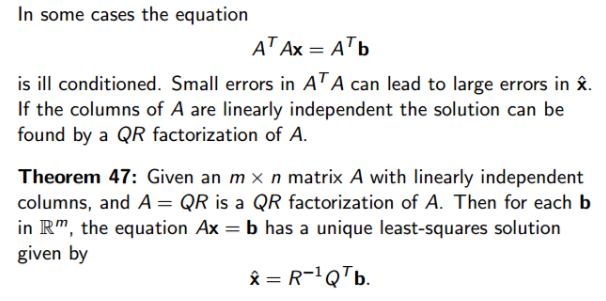
**Code** *(python implementation for testing data sets)*



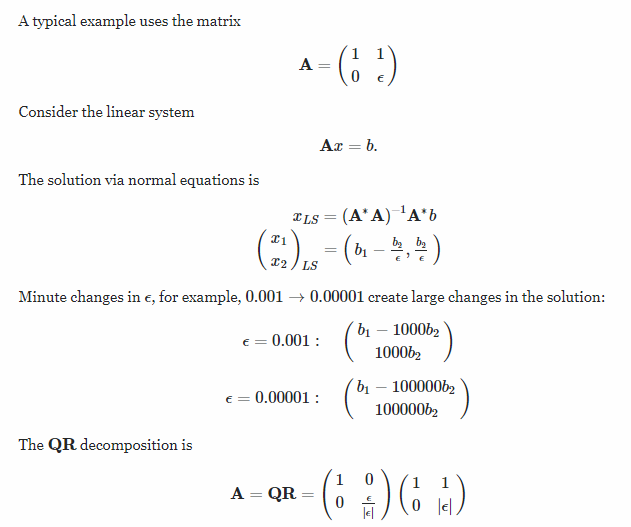
**Output**

****

**Theorem** *(For simplification used in code.)*

****

**Example** *--for the theorm--*

****