DISTRIBUTED SYSTEMS ARCHITECTURE

Lab Assignment 1: Distributed Matrix Inversion

Javier Angulo Lucerón
<javier.angulo1@alu.uclm.es>

César Mora Castro
<cesar.mora@alu.uclm.es>

Curso 2010 - 2011

Ingeniería Superior en Informática - Escuela Superior de Informática de Ciudad Real

Universidad de Castilla la Mancha

Contents

1 Problem

Design and implement a distributed system to do matrix inversion. Focus on large-scale matrix.

2 Algorithm

Having matrix inversion in mind, a few algorithms to obtain the inverse of a matrix appear. Considering that our system must be parallelizable, an algorithm with several parallel parts should be used. The best algorithm at this point is the one who uses determinants and the matrix of cofactors. However, if the matrix has long width, time becomes an unsolvable problem.

Despite of the parallel grade of Gauss-Jordan algorithm, we consider that the best solution to be used. As can be read in any algebra book, this algorithm works in three steps:

1. Extend the original matrix, appending an identity one on the right.
2. Obtain 0's below the main diagonal.
3. Obtain 0's above the main diagonal.

After that, the inverse matrix appears on the extended part. As can be noticed, some problems may appear. For example, one of the members or the main diagonal could be 0. In that case, the original matrix rows must be ordered. However, while running the algorithm, 0's can also appear. To solve this, a previous step should be carried in order to get the exact combination of rows that generates no problem. Nevertheless, some matrix are not invertible.

Our system would be designed to solve simple problems where the source matrix is invertible without ordering its rows. If, suddenly, a 0 appear, the process will abort.

3 Components

The system will be composed of the following components:

- **M**aster: prepares each pivot row and generates the list of works to be done.
- **A**lus: represent the several worker nodes that make the arithmetic operations.
- **S**tore: stores the matrix. Both Master and Alu nodes read and write the source matrix using this component. Every work done modifies the matrix. So, when all the works are done, the resulting matrix is in the store, and the Master can take it from it.

4 Performance

As can be noticed, the nodes which have the most network and processor load are Master and Store nodes. To improve the performance, they should be deployed in different physical nodes. As many alu nodes as desired can be added. They can also be added in the middle of the algorithm.

An important limitation of the system is that alu nodes cannot be removed while the algorithm is running. This problem could be solved storing, in the master node, which works are asigned to each agent. Alu's state should be also monitorized to detect a failure.

5 Reference

See README file located at the main project directory for instructions to compile, deploy and use the software.