Matrix Multiplication in Memphis platform

geaninne.mnl@gmail.com

June 2019

1 Introduction

This document presents an implementation of a matrix multiplication algorithm for the Memphis platform. The most common method to perform matrix multiplication, is to multiply the elements of the lines of matrix A by the elements of the columns of matrix B, followed by the sum of the resulting products. An algorithm for the multiplication of two square matrices is described is this paper.

This implementation uses the master-slave paradigm. The master task fills the arrays, performs the load distribution, and transmits the pertinent data to each slave task. The slave tasks receive the messages, perform the multiplication, and send the result to the master task that constructs the resulting array.

2 How Execute

A shell script file was created, so that the user can parameterize the number of slaves and the size of the array. In addition, the script itself performs the simulation of the application.To execute the script you must use the following command: ./script.sh $SLAVES\ MATRIX\_ORDER\ TIME\ APPLICATION\ TESTCASE\ SCENARIO$ At where:

- SLAVES: Number of slave tasks;
- MATRIX_ORDER: Array order;
- TIME: Simulation time;
- TESTCASE: Tescase file that will be simulated;
- SCENARIO: Scenario file that will be simulated;

For example, to simulate the multiplication of two matrices of order 10, with 5 slaves, using mult_testcase and mult_scenario as hardware and software description respectively, the following command should be used:

./script.sh 5 10 30 *mult_testcase.yaml mult_scenario.yaml*

If the platform has already been generated, it should comment line 53 that contains the following command $memphis - gen\ \$TESTCASE$. The testcase and scenario files should be in sandbox-memphis directory.

3 The implementation

The Figure 1 presents an overview of the application implementation, where the communication between the master task and the slave tasks are represented by dashed lines and the main functions are represented by the colorful boxes. In this implementation the matrices are represented by one-dimensional arrays.

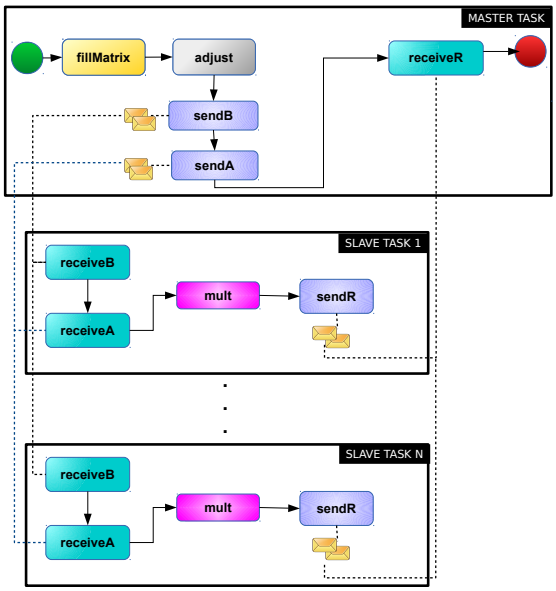

Figure 1: Application Modeling Multiplication of Matrices

The functions are described below:

- **fillMatrix**: the arrays A and B are filled with integer numbers;
- **adjust**: Determines, in a circular way, the lines of the matrix A that should be assigned to each slave task. An example is presented in Figure 2;
- **sendA**: Calculates the number of messages needed to transmit each parcel of A and 4 more elements that indicates: initial position, final position, number of messages and amount of elements of matrix A. Then store the values in the messages that will go to the pipe.
- **sendB**: Calculates the number of messages needed to transmit B and 1 more element that indicates the number of messages. Then store the values in the messages that will go to the pipe.

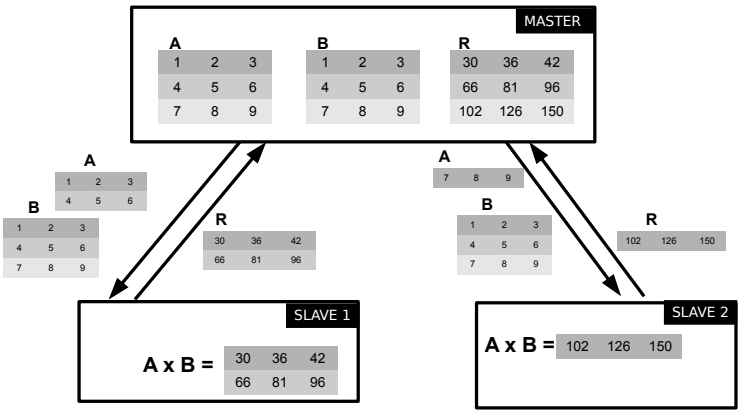

Figure 2: Load Distribution

- **receiveR**: Receive the values of the slaves and built the resulting array;
- **receiveA and receiveB**: Receives the messages and stores the values in A and B arrays;
- **mult**: Performs the multiplication between part of A and B.