CSCI 6401 Programming Project #3 Dr. Tu

Pipeline in Networks: Matrix Multiplication by Blocks

Due Date Monday, November 10, 2014

Problem Description

In Figure 9.7, our textbook gives a distributed algorithm to compute matrix multiplication by blocks, which has two phases. In Phase 1, each row of matrixes ***a*** is shifted toward left for a different distance; each column of matrix ***b*** is shifted upward for a different distance.

```
# shift values in aij circularly left i columns
send left[i,LEFTI](aij); receive left[i,j](aij);
# shift values in bij circularly up j rows
send up[UPJ,j](bij); receive up[i,j](bij);
```

In this complex initialization, matrixes ***a*** and ***b*** are converted into shifted matrixes ***a′*** and ***b′*** as shown in Page 10 of Chap9.pdf.

In Phase 2, the matrix product ***a*** X ***b*** is actually computed. In every iteration of the for loop, matrixes ***a*** and ***b*** are shifted in a much simpler way: the entire matrix ***a*** is shifted toward left for one step, and the entire matrix ***b*** is shifted upward one step.

```
cij = aij * bij;

for [k = 1 to n-1] {
  # shift aij left 1, bij up 1, then multiply and add
  send left[i,LEFT1](aij); receive left[i,j](aij);
  send up[UP1,j](bij); receive up[i,j](bij);
  cij = cij + aij*bij;
}
```

Tasks

Your task is to compute the product of two *n* by *n* matrixes ***a*** and ***b***, ***a*** X ***b***, by implementing the above algorithm using *p* computers to in a structure of circular pipeline parallelization (see Figure 9.5). In your implementation, the computation of Phase 1 can be carried out by an initiator process that accepts two matrixes ***a*** and ***b*** and convert them into ***a′*** and ***b′*** locally. Process Initiator will then send the blocks of ***a′*** and ***b′*** to *p* computers. Every computer hosts process Worker that has three *m* by *m* matrixes $\boldsymbol{a}_k'$, $\boldsymbol{b}_k'$ and $\mathbf{c}_k$, where $m = \frac{n}{\sqrt{p}}$, $k = 1, 2, \ldots, p$. In Phase 2, processes Worker$_k$ will have to communicate with each other and carry out matrix shifting. You are recommended to use Java socket. But other communication mechanisms can be accepted upon approval.

Your program should be configurable with respect to *n* and *p*.

Enhanced Features for Bonus

In the basic version, p is a perfect square number so that $\sqrt{p}$ is an integer, and n is a multiple of $\sqrt{p}$. Therefore, $m = \frac{n}{\sqrt{p}}$ is an integer, too. In reality, n is determined by the actual problem and very likely n is *not* a multiple of $\sqrt{p}$. Your program should be able to wisely distribute blocks with variable sizes to p computers. Furthermore, p is determined by the availability of computers; it is possible that p is not a perfect square number but $p = r \bullet c$. Your program should be able to form a r by c grid using $r \bullet c$ computers.

Test and Performance Measuring

Adequate test cases should be prepared. At least, you should prepare testing matrix and show the correctness of your initialization, the correctness of data distribution, and correctness of matrix shifting (rotation). You should find a local matrix multiplication program and produce the correct result by the Initiator process which will be used to verify the correctness of your distributed program. Be sure to execute your program in lab computers. I will inspect your programs running in lab computers.