Uppsala University
Department of Information Technology
Scientific Computing

Programming of Parallel Computers
2015-01-29

Compulsory Assignment 1

Implementing dense matrix multiplication in parallel distributed memory environment (Fox algorithm + MPI)

1 Problem setting

Let two dense matrices be given, $A = \{a_{i,j}\}$ and $B = \{a_{i,j}\}$, $A, B \in \mathbf{R^{n \times n}}$, i.e., $i, j = 1, 2, \cdots, n$. Let $C = AB$. The elements of C are given by

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}, \quad i, j = 1, 2, \cdots, n.$$

The task is to design a matrix-matrix multiplication procedure for performing the multiplication in a distributed memory parallel computer environment and implement the algorithm using Fortran/C/C++ and MPI. The implementation must be independent of the size of the matrices n and the number of processors p to be used, thus, n and p must be input parameters. For example, your c version code should be run as following: mpirun -np 16 100000, with $n = 100000$, and $p = 16$.

2 Data generation and partitioning strategies

Create your matrices with entries initialized as random double precision numbers on a single processor and distribute the elements according to the given partitioning strategy, see below.

$A^{0,0}$	$A^{0,1}$	...	$A^{0,p2-1}$
$A^{1,0}$	$A^{1,1}$	...	$A^{1,p2-1}$
...	...	...	...
$A^{p1-1,0}$	$A^{p1-1,1}$	...	$A^{p1-1,p2-1}$

Figur 1: Two dimensional data partitioning.

As shown in Figure 1 the processors are assumed to form a 2D Cartesian grid where $p = p1 \times p2$. It is also assumed that n is divisible by both $p1$ and $p2$, i.e., $r1 = n/p1$ and $r2 = n/p2$. For simplicity

you can also assume or restrict to the case $p1 = p2 = \sqrt{p}$. Assume that the matrices A, B and C are partitioned in blocks, as shown in Figure 1. Then, the blocks $A^{(s,r)}, B^{(s,r)}, C^{(s,r)}$ are local for processor $P^{(s,r)}$, $s = 0, 1, \cdots, p1 - 1$, $r = 0, 1, \cdots, p2 - 1$.

3 Fox's algorithm

Processor $P^{(s,r)}$ must compute, in block form,

$$C^{(s,r)} = \sum_{k=0}^{\sqrt{p}-1} A^{(s,k)} B^{(k,r)}$$

where $A^{(s,k)}B^{(k,r)}$ is a matrix-matrix multiplication of two blocks. Processor $P^{(s,r)}$ needs to have all blocks $A^{(s,:)}$ (in row s) and all blocks $B^{(:,r)}$ (in column r). This requires interprocessor communication. The communication and computations can be organized as follows (Fox's algorithm).

For each step k ($k = 0, ..., \sqrt{p} - 1$):
1. Broadcast block $A^{(i,m)}$ within each block-row i ($i = 0, ..., \sqrt{p} - 1$) where $m = (i + k) mod \sqrt{p}$.
2. Multiply the broadcasted block with the B-block in each processor, $C^{(s,r)} = C^{(s,r)} + A^{(s,m)} B^{(m,r)}$.
3. Shift the blocks of B one step cyclicly upwards.

4 MPI implementation

To get an efficient MPI implementation you are required to use a cartesian 2D processor topology and create separate communicators for the rows and the columns. In phase 1, utilize the row-communicators by using collective communication, i.e. `MPI_Bcast`. Then, overlap the communication in phase 3 with the computations in phase 2 by using non-blocking send and receive within the column-communicators. (Start up the communication of the B-blocks before phase 2 and then in phase 3 wait for these communication operations to finish.)

5 Numerical experiments

Check the correctness of your algorithm by applying it to two simple matrices. Then, use your algorithm to compute $C = AB$ for matrices of different sizes. Go up in matrix size from 100×100 to 1440×1440 with a few intervals. For each matrix size vary the number of processors and measure speedup. Pay attention to your results and note any abnormalities. Make several runs for each problem size and processor configuration and report the lowest run-time.

6 Coaching session

You are welcome anytime to ask about the assignment but we will also like to meet all groups in coaching sessions. The intention is to see that you are on the right track and to correct any missunderstandings

in an early stage. Please, contact us for booking a time for the coaching session. Before the coaching session you should have started the implementation of the assignment but there is no requirement that the program can run (or even compile correctly).

7 Writing a report on the results

The report can be written in English and submitted as a PDF file. The report should cover the following issues:

Report requirements:

0. *Author*, write your names, email addresses, and personnummer (civil registration number).
1. *Problem description*, presenting the task.
2. *Solution method*, i.e. a description of the parallel implementation.
3. *Results*, presenting plots of speedup. The plots can be done using Matlab, Maple or any other graphical tool.
4. *Discussion*, with observations and comments on the results (can also be included in 3 above).
5. *Conclusions*, with explanations of the results and with ideas for possible optimizations or improvements.
6. *Appendix*, with a list of filenames and descriptions of your code and tables of the numerical results.
7. *Code*. Don't copy and paste your code in side the report. Instead, upload them as separated files. If you have 2 or more code files, then compress them as a .tar or .zip file (no .rar, I cannot open it).

The assignments are a part of the examination, this means that you should protect your source code from others and you can't copy others solutions. (See `http://www.it.uu.se/internt/policy_rapport`). The last day to hand in the report is **February 13, 2015**.

If you would like to use the University Linux/Unix system at home, please read the file "ssh_instruction.rtf". You can also find demonstration code and lectures at git hub: `https://github.com/JinLi971/MPI_DEMO`.