ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Advanced Parallel Computing for Scientific Applications
Autumn Term 2009

Prof. I. F. Sbalzarini
ETH Zentrum, CAB G34
CH-8092 Zürich

Prof. P. Arbenz
ETH Zentrum, CAB G69.3
CH-8092 Zürich

Exercise 8

Release: 09 Nov 2009
Due: 16 Nov 2009

Question 1: Fox's algorithm I: Introduction

Consider a the matrix multiplication $C = A \cdot B$. One intuitive way for parallel processing of this task with p processes is to subdivide the matrices into equally-sized submatrices A_{ij} and B_{ij} with $i, j = 1 \dots q$. For example the matrix A can be subdivided as follows:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}$$

where A_{11}, for example, corresponds to $(a_{22}\ a_{23}; a_{32}\ a_{33})$. This is called a *Checkerboard* distribution of the submatrices.

Assume that all the submatrix-multiplications can be computed with different processes. A specific element of the result C can be calculated as follows:

$$C_{ij} = A_{i0}B_{0j} + A_{i1}B_{1j} + \dots + A_{iq}B_{qj}$$

a) **(Paper and Pencil)** For two arbitrary (4×4) matrices A and B and $q = 2$, please calculate the element $C_{23} = (A \cdot B)_{23}$ using the scheme above.

b) **(Paper and Pencil)** Suppose that we have 4 processors available connected by a network for this task. Each process is responsible for the calculation of a submatrix of C. Suppose further that the matrices are too large to store all data from A and B at once, so each process holds data according to the checkerboard distribution (Process $\mathcal{P}_{11}$ that treats submatrix C_{11} holds A_{11} and B_{11}). What information needs to be sent to the process $\mathcal{P}_{11}$?

Now think of the calculation for all elements of C: For large matrices, a lot of data needs to be transmitted. Fox's algorithm enables good networking and load balance for this task:

Before the algorithm starts, each process holds the submatrices according to the checkerboard distribution. At iteration $k = 0 \dots n-1$, the process at $\mathcal{P}_{ij}$ calculates:

$$C_{ij} = C_{ij} + A_{i\bar{k}} \cdot B_{\bar{k}j},\ \bar{k} = (i+k) \mod n.$$

For example, at the first iteration $k = 0$, C is calculated as follows

$$\begin{array}{lll} C_{00} = A_{00} * B_{00} & C_{01} = A_{00} * B_{01} & C_{02} = A_{00} * B_{02} \\ C_{10} = A_{11} * B_{10} & C_{11} = A_{11} * B_{11} & C_{12} = A_{11} * B_{12} \\ C_{20} = A_{22} * B_{20} & C_{21} = A_{22} * B_{21} & C_{22} = A_{22} * B_{22} \end{array} \tag{1}$$

c) **(Paper and Pencil)** For two arbitrary 3×3 matrices, simulate the Fox algorithm with 9 processes. For the first iteration, you may use the formulas in the table 1 above. What elements are now transmitted at each step of the algorithm, and more important, where? Please draw a sketch to illustrate.

Question 2: Fox's Algorithm II: Implementation

For simplicity, we restrict ourselves to square matrices of size $(m \cdot \sqrt{p}) \times (m \cdot \sqrt{p})$; $m, \sqrt{p} \in \mathbb{N}$.

The implementation for p processors contains the following tasks:

1. Determine the number of submatrices per row and column, respectively: $q = \sqrt{p}$
2. Assign each processor its coordinates i and j.
3. Determine the coordinates of the processor to send to ($\mathcal{P}_{\text{dest}}$): $((i-1) \mod q, j)$.
4. Determine the coordinates of the processor from which data is received: $((i+1) \mod q, j)$.
5. Iterate: $k = 0 \ldots n-1$
   (a) - Calculate $\bar{k} = (i+k) \mod q$
   (b) - Broadcast $A(i, \bar{k})$ to the processors on the same row i.
   (c) - Calculate $C_{ij} = C_{ij} + A_{i\bar{k}} \cdot B_{\bar{k}j}$ locally.
   (d) - Send $B_{\bar{k},j}$ to $\mathcal{P}_{\text{dest}}$ and receive $B_{(\bar{k}+1 \mod q),j}$ from the source.

Please download the code skeleton from the course web-site.

a) Please implement the C-function `generate_communicators(...)` that creates the communicators needed: one communicator per row and one communicator per column. There are several ways to do that. One possible way is using `MPI_Comm_Split(...)`. Another more elegant possibility is to use a cartesian topology. The appropriate MPI function to invoke in this case is `MPI_Cart_create(...)`.

b) Please implement the C-function `generate_sub_matrix(...)` that takes a `COMM_INFO`-struct as an argument and generates the submatrix for this particular process. The element (i, j) of the global matrix reads: $\cos(i \cdot dim + j)$; $i, j = 0..dim - 1$.

c) Write a C-function `multiply_local(...)` that does a local matrix multiplication.

d) Please implement the C-function `collect_and_print(...)` where the master process gets the element C_{ij}, from the process that owns this entry, and prints it to the console.

e) In your main function, implement the steps 1-5 outlined above. Calculate the product $C = A \cdot B$ where A and B are both generated with `generate_sub_matrix(...)`.

The element $C_{98,99}$ is equal to -0.37962 for $dim = 100$. Test your program for several numbers of processors.