# Day 7:
# Worksheet Questions

David W. Walker

Professor of High Performance Computing

Cardiff University

http://www.cardiff.ac.uk/people/view/118172-walker-david

# Q1: Network Metrics

*Evaluate the bisection width, expansion increment, and network narrowness for a cubic mesh of $k^3$ nodes.*

The bisection width of a k×k×k mesh is the number of links between adjacent 2D slabs, which is $k^2$.

The expansion increment is:

$$(k+1)^3 - k^3 = 3k^2 + 3k + 1$$

Partition the mesh into m slabs and call this group A, so NA = $mk^2$ and NB = $(k-m)k^2$. The number of links between groups A and B is I = $k^2$, so NB/I = k-m. This is maximized when m is as small as possible, but since NA≥NB the smallest value of m is k/2. Thus, the network narrowness for a k×k×k mesh is k/2.

# Q2: Non-blocking and Asynchronous Send

*Explain carefully the difference between a non-blocking send and an asynchronous send in MPI.*

On return from a non-blocking send it is not guaranteed that the data has “left” the calling application, so any subsequent change to the data may affect the message sent. The contents of the message buffer should not be overwritten until a subsequent MPI call has ensured that the data has “left” the calling application. An asynchronous send may return even if the matching receive operation has not started on the destination process, thus the send and receive operations may not overlap in time.

# Q3: Application Topologies

*What is meant by an application topology, and how are Cartesian mesh application topologies supported in MPI?*

An application topology is the topology of the communication between processes in the application. This can be viewed as a graph in which the nodes represent the processes and the arcs between nodes represent communication. MPI provides routines for determining the process rank given a location in the mesh, and vice versa. MPI also provides a routine to find the source and destination process ranks for shift communication operations

# Q4: Hypercube Node Numbers

*In a 6-dimensional hypercube, which nodes are directly connected to node number 20?*

20 as a 6-bit binary number is 010100, so node 20 is connected to nodes:

$(010101)_2 = 21$, $(010110)_2 = 22$,

$(010000)_2 = 16$, $(011100)_2 = 28$,

$(000100)_2 = 4$, $(110100)_2 = 52$.

# Q5: Mapping Meshes to Hypercubes

*A regular 4x4 mesh is mapped onto a 4-dimensional hypercube so that neighbouring nodes in the mesh are also neighbours in the hypercube. Determine the node number of the hypercube node at each location in the mesh.*

# Q5: Answer

| Location | Split | Gray | Node |
|---|---|---|---|
| (0,0) | (00,00) | (00,00) | 0 |
| (0,1) | (00,01) | (00,01) | 1 |
| (0.2) | (00,10) | (00,11) | 3 |
| (0,3) | (00,11) | (00,10) | 2 |
| (1,0) | (01,00) | (01,00) | 4 |
| (1,1) | (01,01) | (01,01) | 5 |
| (1,2) | (01,10) | (01,11) | 7 |
| (1,3) | (01,11) | (01,10) | 6 |
| (2,0) | (10,00) | (11,00) | 12 |
| (2,1) | (10,01) | (11,01) | 13 |
| (2,2) | (10,10) | (11,11) | 15 |
| (2,3) | (10,11) | (11,10) | 14 |
| (3,0) | (11,00) | (10,00) | 8 |
| (3,1) | (11,01) | (10,01) | 9 |
| (3,2) | (11,10) | (10,11) | 11 |
| (3,3) | (11,11) | (10,10) | 10 |

| | | | |
|---|---|---|---|
| 0 | 1 | 3 | 2 |
| 4 | 5 | 7 | 6 |
| 12 | 13 | 15 | 14 |
| 8 | 9 | 11 | 10 |

# Q6: OpenMP For Loop

*Consider the following piece of OpenMP code (the line numbers are not part of the code):*

```
 1   int n=12, chunk=10, i, j;
 2   float x[n][n], y[n][n], xji;
 3   #pragma omp parallel shared(default) private(xji,i,j) num_threads(5)
 4   {
 5   #pragma omp for collapse(2) schedule(static,chunk)
 6   for(j=0;j<n;j++)
 7       for(i=0;i<n;i++){
 8           xji = x[j][i];
 9           y[j][i] = xji*xji + 4.0*xji + 7.0;
10       }
11   }
```

*a) Which elements of the array y will be evaluated by thread 4?*

*b) If the collapse clause is removed from line 5, which thread will evaluate y[5][10]?*

# Q6: Answer

In linearized form, chunk $c$ is evaluated by thread 4 if $c$ mod 5 = 4, so chunks 4, 9, and 14 (there are only 15 chunks) are evaluated by thread 4. Chunk $c$ starts at ($10c$/12; ($10c$)%12) and ends at (($10c$ + 9)/12; ($10c$ + 9)%12) (except when the last chunk is not full). So the elements evaluated by thread 4 are:

Chunk 4: (3,4), (3,5), (3,6), (3,7), (3,8) (3,9), (3,10), (3,11), (4,0), (4,1)
Chunk 9: (7,6), (7,7), (7,8), (7,9), (7,10), (7,11), (8,0), (8,1), (8,2), (8,3)
Chunk 14: (11,8), (11,9), (11,10), (11, 11)

If the collapse clause is removed then only the outer loop will be parallelized, so the first 10 iterations of the outer loop will be done by thread 0. Thus, y[5][10] will be evaluated by thread 0.

# Q7a: Cyclic Data Distribution

*Suppose we have a (cyclic[4],cyclic[5]) data distribution that is used to distribute a matrix, A, over a 3x4 process mesh. What is the position in the process mesh of the process holding matrix element (21,36)? Note: all indexing starts at 0.*

# Q7a: Answer

We treat each dimension independently.

First the row direction: if the block size is 4 then element 21 is in global block 21/4 = 5. If there are 3 processes per row then this block will be in process row 5mod(3) = 2.

Next the column direction: if the block size is 5 then element 36 is in global block 36/5 = 7. If there are 4 processes per column then this block will be in process column 7mod(4) = 3. Thus, the processor position is (2,3).

# Q7b: Cyclic Data Distribution

*On a distributed memory computer each process holds its part of the matrix A in a local array, L. If the same data distribution is used as in Q7a, what is the global index (m,n) of the element of the matrix A stored at location (3,3) of L in the process at location (2,3) of the processor mesh?*

# Q7b: Answer

The element at (3,3) in the processor at (2,3) is in local block ($b_r$,$b_c$) = (0,0), where it is stored at local index ($i_r$,$i_c$) = (3,3). The global index is:

$$(m, n)$$
$$= \big((b_r P + p)k_r + i_r\big), (b_c Q + q)k_c + i_c)$$
$$= (24 + 3, 35 + 3) = (11,18)$$

# Q8: Parallel Efficiency

*An identical parallel application is run on two parallel computers, A and B. A and B both have the same number of processors, but the processors on A and B are of different types, with those on B having twice the serial speed as those of A. On A the parallel efficiency is 0.7, and on B it is 0.5, when running on all the processors. What is the ratio of the parallel execution times for the application on A and B?*

# Q8: Answer

Let N be the number of processors:

$$\varepsilon_A = \frac{1}{N}\frac{T_{seq,A}}{T_{par,A}(N)} \quad \text{and} \quad \varepsilon_B = \frac{1}{N}\frac{T_{seq,B}}{T_{par,B}(N)}$$

So,

$$\frac{\varepsilon_A}{\varepsilon_B} = \frac{T_{seq,A}}{T_{seq,B}}\frac{T_{par,B}(N)}{T_{par,A}(N)} \Rightarrow \frac{T_{par,A}(N)}{T_{par,B}(N)} = \frac{\varepsilon_B}{\varepsilon_A}\frac{T_{seq,A}}{T_{seq,B}}$$

So the ratio of parallel execution times is (0.5/0.7)*2 = 1.4285.

# Q9: CUDA Thread Block Sizes

*Two matrices of size 8200x8200 are multiplied together to produce a third matrix. CUDA is used to perform this operation on a GPU by using one thread to calculate each element of the output matrix. A square layout of thread blocks is used, and each thread block is 32x32.*

*a) What are the grid dimensions in blocks?*

*b) How many threads are in the grid?*

# Q9: Answers

a) ceil(8200/32) = 257 so the grid will be of size 257x257 blocks.

b) The number of threads is 257x257x32x32 = 67634176.

# Q10: Compute To Global Memory Access Ratio

*The bandwidth for global memory accesses on a GeForce GTX 960 GPU is 112GByte/s. Assuming a compute to global memory access ratio of 8.0, and that each floating-point value is of size 4 bytes, what is the maximum execution speed in Gflop/s?*

For each global memory access 8 floating-point operations can be performed. $112 \times 10^9/4 = 28 \times 10^9$ floats can be accessed per second, so the maximum execution speed is $8 \times 28 \times 10^9$ flop/s, or 224 Gflop/s.

# Q11: Performance Analysis

A data parallel algorithm running on N processes operates on a nxnxn mesh of points in a series of time steps. Each point requires 12 floating-point operations per time step. In addition, each time step of the parallel algorithm involves 6 data shift operations, in each of which a process communicates $n^2$ floating-point numbers. Determine the efficiency of the parallel algorithm, assuming that the best sequential algorithm is the same as the parallel algorithm running on one process.

# Q11: Answer

The time per time step for sequential execution on one processor is $T_{seq} = 12n^3t_{calc}$. The parallel execution time is:

$$T_{par} = \frac{12n^3}{N} t_{calc} + 6n^2 t_{shift}$$

So the speedup is:

$$S(n,N) = \frac{T_{seq}}{T_{par}} = \frac{12n^3 t_{calc}}{12\left({}^{n^3}/_{N}\right) t_{calc} + 6n^2 t_{shift}}$$

$$= \frac{N}{1 + \left({}^{N}/_{n}\right)\left({}^{\tau}/_{2}\right)}$$

where $\tau = t_{shift}/t_{calc}$

# Q12: Performance Analysis

*A sequential algorithm requires $2n^3$ floating-point operations to multiply two nxn matrices together. The corresponding parallel algorithm on a square mesh of PxP processors assigns a (n/P)x(n/P) submatrix to each processor, and consists of P stages. In each stage each processor does the following:*

A. *Performs $2(n/P)^3$ floating-point operations.*
B. *Is involved in a broadcast that takes time $((n/P)^2 \log_2 P)t_{comm}$, where $t_{comm}$ is the time to communicate one floating-point number between two processors.*
C. *Except for the first stage, is involved in a point-to-point communication that takes time $(n/P)^2 t_{comm}$.*

*What is the parallel efficiency of the algorithm?*

# Q12: Answer

$$T_{par}(n,P) = 2P\left({}^{n}/_{P}\right)^{3} t_{calc} + \left({}^{n}/_{P}\right)^{2}(Plog_2P + P - 1)t_{comm}$$

$$S(n,P) = \frac{2n^3 t_{calc}}{2P\left({}^{n}/_{P}\right)^{3} t_{calc} + \left({}^{n}/_{P}\right)^{2}(Plog_2P + P - 1)t_{comm}}$$

$$= \frac{P^2}{1 + \left({}^{\tau}/_{2}\right)\left({}^{P}/_{n}\right)\left(log_2P + 1 - {}^{1}/_{P}\right)}$$

$$\varepsilon(n,P) = \frac{1}{1 + \left({}^{\tau}/_{2}\right)\left({}^{P}/_{n}\right)\left(log_2P + 1 - {}^{1}/_{P}\right)}$$

# Q13: Cyclic Data Distribution

*Suppose we have a block-cyclic data distribution with block size 3x4 that is used to distribute a matrix, A, over a 3x6 processor mesh. On a distributed memory computer each processor holds its part of the matrix A in a local array, L. What is the global index (m,n) of the element of the matrix A stored at location (2,4) of L in the processor at location (1,4) of the processor mesh?*

Use: m = (bN+p)k + i

# Q13: Answer

We treat each dimension independently.

First the row direction: the item stored locally in row 2 is in local row block 2/3=0 at row index 2%3=2 within the block. If this is in the processor in row 1 of the processor mesh, then the global row block is 1, so the global row is 1*3+2 = 5.

Next the column direction: the item stored locally in column 4 is in local column block 4/4=1 at column index 4%4=0 within the block. If this is in the processor in column 4 of the processor mesh, then the global column block is 1*6+4=10, so the global column is 10*4+0 = 40. Thus, the item is at global location (5,40).

# Q14: OpenMP For Loop

*In the following fragment of OpenMP code, a, b, and c are two-dimensional arrays:*

```
int N = 15;
int i, j, chunk = 5;
#pragma omp parallel shared(a,b,c,chunk) private(i,j)
{
#pragma omp for collapse(2) schedule(static,chunk)
for (i=0; i < N; i++)
        for (j=0; j<N; j++)
                c[i][j] = a[i][j] + b[i][j];
}
```

*Draw a diagram showing how the first 3 iterations of the outer loop are assigned to threads in the above code fragment, assuming there are 6 threads.*

# Q14: Answer

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |

# Q15: Performance Analysis

An ocean model on an n x n numerical grid performs $10000n^3T$ floating-point operations to simulate T seconds of global ocean circulation.

a) Give an formula for the execution rate in floating-point operations per second that must be achieved if a 100-year simulation is to complete in one hour. You should assume there are 365.25 days in a year.

b) If the current fastest supercomputer can compute at a rate of $15x10^{15}$ floating-point operations per second (15 Petaflop/s), what is the largest problem that can be simulated for 100 years in one hour on this machine?

c) If supercomputers keep increasing in speed by a factor of 100 every 5 years:

   i. In 15 years from now what will be the largest problem that can be simulated for 100 years in one hour?

   ii. How long will it be before a problem of size n=20000 can be simulated for 100 years in one hour?

# Q15: Answer

a) Number of operations for 100 year simulation = $100 \times 365.25 \times 24 \times 3600 \times 10000 n^3 = 3.15576 \times 10^{13} n^3$

To complete in one hour the execution rate must be:

$3.15576 \times 10^{13} n^3 / 3600 = 8.766 \times 10^9 \, n^3$ floating-point operations per second

$= 8.766 n^3$ Gflop/s.

b) The largest problem is given by $8.766 \times 10^9 n^3 = 15 \times 10^{15}$, so $n^3 = 1.7112 \times 10^6$, and n = 119 is the largest problem.

c)

i. Speed = $15 \times 10^{15} \times 10^{2T/5}$, so speed in 15 years = $15 \times 10^{21}$ flop/s. So the largest problem in 15 years is given by $8.766 \times 10^9 n^3 = 15 \times 10^{21}$, so $n^3 = 1.7112 \times 10^{12}$, and n = 11960.

ii. Speed needed to solve n=20000 problem in one hour = $8.766 \times 10^9 \times 20000^3 = 7.013 \times 10^{22}$. So the time until this speed is achieved is given by $15 \times 10^{15} \times 10^{2T/5} = 7.013 \times 10^{22}$, so $10^{2T/5} = 4.675 \times 10^6$. The n = 20000 problem can therefore be solved in T = 16.67 years from now.