

HPC Fall 2007 – Homework 3

Robert van Engelen

Due date: October 9, 2007

1. The standard, non-optimized matrix multiply (DGEMM) operation has an FP:M ratio of 1.00 when $K \rightarrow \infty$. Two memory references are made to distinct locations in A and B (while location C remains the same in the inner loop) and two floating point operations are performed per inner loop iteration:

```
DO J = 1,N
  DO I = 1,M
    DO L = 1,K
      C(I,J) = C(I,J) + A(L,I) * B(L,J)
    ENDDO
  ENDDO
ENDDO
```

The performance of this loop nest is determined by the performance of the innermost loop. Thus, the FP:M ratio is a good indicator of expected performance¹

Determine the FP:M ratio for the following kernels, assuming large iteration spaces for the (innermost) loops:

- (a) The 2-norm of a vector

$$\|x\|_2 = \sqrt{x^T x} = \sqrt{\sum_{i=1}^n x_i^2}$$

- (b) Scalar vector accumulation (DAXPY)

$$y_i = y_i + ax_i \quad \forall i = 1, \dots, n$$

¹Theoretically, we have a total of $2NMK$ operations and $NM + MK + NK$ distinct memory references to A , B , and C . However, the memory references to A and B are repeated in the innermost loop. When K is relatively large the performance of the innermost loop determines the overall performance.

- (c) DGEMM with hoist/sink of matrix C using unroll and jam of the J and K loops by a factor of 2:

```

DO J = 1,N,2
  DO I = 1,M,2
    C11 = C(I,J)
    C12 = C(I+1,J)
    C21 = C(I,J+1)
    C22 = C(I+1,J+1)
    DO L = 1,K
      C11 = C11 + A(L,I) * B(L,J)
      C21 = C21 + A(L,I+1) * B(L,J)
      C12 = C12 + A(L,I) * B(L,J+1)
      C22 = C22 + A(L,I+1) * B(L,J+1)
    ENDDO
    C(I,J)      = C11
    C(I+1,J)    = C12
    C(I,J+1)    = C21
    C(I+1,J+1) = C22
  ENDDO
ENDDO

```

2. Suppose we have an $N \times N$ computational grid that is shaped as a 2D torus. We apply a finite difference operation on this grid with the following stencil:

$$Du_{i,j} = \frac{u_{i+1,j} - u_{i-1,j} + u_{i,j+1} - u_{i,j-1}}{4}$$

Suppose we divide the grid into P equal-sized $B \times B$ blocks, where P is the number of processors and each processor executes the finite difference operation on its local $B \times B$ grid block. For the grid points on the edges of its block the processor must communicate with another processor to obtain the value of u . Processors are logically arranged in a $\frac{N}{B} \times \frac{N}{B}$ two-dimensional torus of P processors, i.e. $\sqrt{P} = \frac{N}{B}$, where we assume that N is a multiple of B .

This is implemented as follows:

```

for (i = 0; i < B; i++)
{ for (j = 0; j < B; j++)
  { if (i == 0)  u1 = recv(Pnorth); else u1 = u[i-1][j];
    if (i == B-1) u2 = recv(Psouth); else u2 = u[i+1][j];
    if (j == 0)  u3 = recv(Pwest);  else u3 = u[i][j-1];
    if (j == B-1) u4 = recv(Peast); else u4 = u[i][j+1];
    Du[i][j] = (u2-u1+u4-u3)/4.0;
  }
}

```

Note that for sake of simplicity the send operations are not shown. Processors are *logically* connected to the North, South, East, and West. However, the *physical* topology of the interconnect network may vary when it is not a 2D torus, which means that communication from P to P_{north}, P_{south}, P_{west}, and P_{east} is not necessarily one hop in that case and messages may have to travel through multiple nodes and/or network latencies increase by distance from P.

The communication time of one floating point value versus arithmetic is $\gamma = 8$, i.e. we can perform 8 arithmetic operations in the same time it takes one communication step to complete (a “hop” between two processors).

- Assume that the physical processor interconnect network is a 2D torus, so the send/rcv operations take only one hop between two processors (one send/rcv step for P to communicate with P_{north}, P_{south}, P_{east}, and P_{west}, and there is no additional latency due to contention on the communication links). For $P = 4^k$ with $k = 0, \dots, 9$ and $N = 512$ plot the log-log relative performance prediction graph of t_{comp} and t_{comm} . Express t_{comp} and t_{comm} in terms of B and $\gamma = 8$. Use the fact that $\sqrt{P} = \frac{N}{B}$.
- From the plot, determine when $t_{\text{comp}} = t_{\text{comm}}$.
- Plot the relative speedup $S_P^1 = \frac{t_1}{t_P}$ for $P = 4^k$ with $k = 0, \dots, 9$ and $N = 512$. Note that for $P = 1$ there is no communication, so you should use $t_1 = t_{\text{comp}}$ for $P = 1$.
- Assume that the physical processor interconnect network is a linear torus. That is, processors are arranged in a network where each processor has only two links, one to its left and one to its right. We will map the 2D torus on the 1D torus by laying out the 2D torus row-by-row on the 1D torus. That is, P has direct links to P_{west} and P_{east} (ignoring the links of processors on the edges of the grid). Messages from P to P_{north} and P_{south} take \sqrt{P} hops to send via all the intermediate processors that must send the message from P_{north} (or P_{south}) to P (draw a picture if you are not sure, where you map processor $P_{i,j}$ on the logical 2D torus to processor P_{j+ki} with $k = \frac{N}{B}$). Given this physical topology, plot the log-log relative performance prediction graph of t_{comp} and t_{comm} for $P = 4^k$ with $k = 0, \dots, 9$ and $N = 512$.
- From the plot, determine when $t_{\text{comp}} = t_{\text{comm}}$.
- Plot the relative speedup for $P = 4^k$ with $k = 0, \dots, 9$ and $N = 512$.