CS 590V: Parallel Computing

Spring 1999

Problem Set 1: Parallelism in Microprocessors,

Elements of Parallel Computing Platforms

Due Date: Feb 16, 1999 in class

No late submissions will be accepted

1. Consider a memory system with a level 1 cache of 32 KB and DRAM of 64 MB with the processor operating at 500 MHz. The latency to L1 cache is 1 cycle and the latency to DRAM is 50 cycles. In each memory cycle, the processor fetches 4 words. What is the peak achievable performance of a dot product of two vectors?

```
/* dot product loop */
for (i = 0; i < dim; i++)
        dot_prod += a[i] * b[i];
```

2. Now consider the problem of multiplying a dense matrix with a vector using a two-loop dot-product formulation. The matrix is of dimension $4K \times 4K$. (Each row of the matrix takes 16KB of storage.) What is the peak achievable performance of this technique using a two-loop dot-product based matrix-vector product?

```
/* matrix-vector product loop */
for (i = 0; i < dim; i++)
        for (j = 0; i < dim; j++)
                c[i] += a[i][j] * b[j];
```

3. Extending this further, consider the problem of multiplying two dense matrices of dimension $4K \times 4K$. What is the peak achievable performance using a three-loop dot-product based formulation? (Assume that matrices are laid out in a row-major fashion).

```
/* matrix-matrix product loop */
for (i = 0; i < dim; i++)
        for (j = 0; i < dim; j++)
                for (k = 0; k < dim; k++)
                        c[i][j] += a[i][k] * b[k][j];
```

4. Restructure the matrix multiplication algorithm to achieve better cache performance.

   The most obvious cause of the poor performance of matrix multiplication was the absence of spatial locality. We were wasting three of the four words fetched from memory. To fix this problem, we compute the elements of the result matrix four at a time. Using this approach, we can increase our FLOP count to 80 MFLOPS with a simple restructuring of the program.

   However, it is possible to achieve much higher performance from this problem. This is possible by viewing the matrix multiplication problem as a cube in which each internal lattice point corresponds to a multiply-add operation. Matrix multiplication algorithms traverse this cube in different ways; which induce different partitioning of the cube. The data required for computing a partition grows as the surface area of the input faces of the partition and the computation as the volume of the partition. For the algorithms discussed above, we were slicing thin partitions of the cube for which the area and volume were comparable (thus achieving poor cache performance). To remedy this, we restructure the computation by partitioning the cube into subcubes of size $k \times k \times k$. The data associated with this is $3 \times k^2$ (k^2 data for each of the three matrices) and the computation is k^3. To maximize performance, we would like $3 \times k^2$ to be equal to $8K$ since thats the amount of cache available. This corresponds to $k = 51$. The computation associated with a cube of this dimension is 132651 multiply-add operations or 265302 FLOPs. To perform this computation, we needed to fetch two submatrices of size 51×51. This corresponds to 5202 words or 1301 cache lines. Accessing these cache lines takes 65050 cycles or 130100 ns. Since 265302 FLOPs are performed in 130100 ns, the peak computation rate of this formulation is 2.04 GFLOPS. Since this exceeds the processor performance, the computation rate is infact bound by the number processor performance of 1 GFLOPS. Code this example and plot the performance as a function of k. (Code on any conventional microprocessor. Make sure you note the clock speed, the microprocessor and the cache available at each level).

5. Consider an SMP with a distributed shared address space. Consider a simple cost model in which it takes 10 ns to access local cache, 100 ns to access local memory, and 400 ns to access remote memory. A parallel program is running on this machine. The program is perfectly load balanced with 80% of all accesses going to local cache, 10% to local memory, and 10% to remote memory. What is the effective memory access time for this computation? If the computation is memory bounded, what is the peak computation rate?

   Now consider the same computation running on 1 processor. Here, the processor hits the cache 70% of the times and local memory 30% of the time. What is the effective peak computation rate for 1 processor? What is the efficiency is this computation?

   Note: Notice that the cache hit for multiple processors is higher than that for 1 processor. This is typically because the cache available on multiprocessors is larger than single processor systems. For example, 2 R10000 processors will have 64KB L1 cache and a single R10000 only has 32KB of cache. As we shall subsequently see (and you may extrapolate from this question), this is often a source of superlinearity (speedup of greater than p on p processors).

6. The concept of a minimum congestion mapping is very useful. Consider two parallel computers with different interconnection networks such that a congestion-r mapping of the first into the second exists. Ignoring the dilation of the mapping (i.e. the number of links of the second architecture that each link in the first is mapped to), if each communication link in the second computer is more than r times faster than the first computer, the second computer is strictly superior to the first.

   Now consider mapping a d-dimensional hypercube onto a 2^d-processor mesh. Ignoring the dilation of the mapping, what is the minimum-congestion mapping of the hypercube onto the mesh? Use this result to determine whether a 1024-processor mesh with communication links operating at 25 million bytes per second is strictly better than a

1024-processor hypercube (whose processors are identical to those used in the mesh) with communication links operating at two million bytes per second.

Hint: Recursively map halves of machines and compute the congestion as ratios of bisection width.

7. Derive the diameter, bisection width, and cost (in terms of the number of links) for the following networks.

   (a) Completely connected network (every node is connected to every other node).

   (b) Complete binary tree.

   (c) 2-D wraparound mesh.

   (d) Hypercube.

   (e) wraparound k-ary d-cube.

8. Consider a hypercube network of p processors. Assume that the channel width of each communication link is one. The channel width of the links in a k-ary d-cube (for $d < \log p$) can be increased by equating the cost of the this network with that of a hypercube. Two distinct measures can be used to evaluate the cost of a network.

   (a) The cost can expressed in terms of the total number of wires in the network (the total number of wires is a product of the number of communication links and the channel width).

   (b) The bisection bandwidth can be used as a measure of cost.

   Using each of these cost metrics and equating the cost of a k-ary d-cube with a hypercube, what is the channel width of a k-ary d-cube with an identical number of processors, channel rate, and cost?