CS 740, Fall 1997
Assignment 4:
Memory Systems and I/O

Assigned: Wed., Oct. 22
Due: Mon., Nov. 3

Policy

You may work in a group of up to 3 people in solving the problems for this assignment. You should turn in a single report for your entire group, identifying all of the group members.

Logistics

Any clarifications and revisions to the assignment will be posted on the class bboard and Web page. There is no electronic handin for this assignment. Turn in the two parts as separate documents, so that they can be graded independently.

1. Memory System Performance

For this part of the assignment you will characterize the different levels of your computer's memory system. You will then use this information to optimize the performance of a memory-intensive application.

1.1. Platform Choice

A first step is to choose the platform you will measure and optimize. The machine must have a cache. It would be more interesting if it has multiple levels of cache, but this is not essential. Also, since caching is very sensitive to the effects of context switching, you should use a very lightly loaded machine.

Document your platform type and its clock rate.

1.2. Cache Parameters

Determine how many levels and what types of data caches you have. In particular, determine for each level:

1. The total size
2. The block size
3. Is it write-through or write-back?

You can do this by either tracking down the literature on your processor, using any available online information, or writing simple routines that, by measuring running times, allow you to determine these parameters.

1.3. Memory System Performance

For each level of cache *plus main memory*, determine the maximum achievable bandwidths (measured in MB/second) for the following integer operations:

1. Reading, e.g., when summing an array of integers.
2. Writing, e.g., when setting all elements of an array to a constant value.
3. Reading and writing, e.g., when copying the contents of one array to another.

Here are some tips for performing this analysis:

- Make your measurements by repetitively performing the operations on one or more arrays that fit within the target memory level but not in the next level higher.
- Do whatever unrolling and software pipelining is required to reach the point where your code is bandwidth limited.
- Be very careful about getting the appropriate cache "warmed up" before you begin your measurement.

1.4. Application

Consider the problem of transposing an $n \times n$ array, converting it from column-major to row-major order. The naive code for this would look something like this:

```
void transpose1(int src[], int dest[], int n)
{
   int i, j, srcpos, destpos;
   for (i = 0; i < n; i++)
     for (j = 0; j < n; j++) {
        srcpos = i + j*n;
        destpos = i*n + j;
        dest[destpos] = src[srcpos];
     }
}
```

Doing strength reductions and pointer coding gives us the following:

```
void transpose2(int src[], int dest[], int n)
{
   int i, j;
   int *next_src;
   int *next_dest = dest;
   for (i = 0; i < n; i++) {
     next_src = src+i;
     for (j = 0; j < n; j++) {
        *next_dest = *next_src;
        next_dest++;
        next_src += n;
```

```
        }
    }
}
```

Test the performance of `transpose2` on your platform for matrices sized to fit within the different levels of your memory hierarchy. How close do these come to the optimal copy bandwidth you measured earlier?

1.5. Optimization

Modify this code to take reduce overhead and to take better advantage of your memory system characteristics. In particular, you should be able to take better advantage of the cache block size effects. You may assume that n is a multiple of 8. You should be able to come close to the optimal copy bandwidth.

Explain the optimizations you make and document the performance you obtain at all levels of the hierarchy.

2. I/O System Performance

This part of the assignment will be ready by Friday, October 24.