CS 740, Fall 1998
Homework Assignment 3 Part B
Memory System Performance
Assigned: Oct. 8, Due: Thurs., Oct. 22, 1:20 PM

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

Policy

You may work in a group of up to 3 people in solving the problems for this assignment. There will be just hard copy hand-in for this part. Turn in separate documents for part A and part B. Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory.

Platform Choice

A first step is to choose the platform you will measure and optimize. We would like you to use a machine with an "interesting" cache hierarchy—i.e., with multiple levels of cache. As a rule of thumb, we would like you to use a machine with a high-end processor that came out within the past three years. For example, you can use the class Alpha machines, a machine with a PentiumPro or Pentium II processor, a machine with a recent PowerPC processor, etc. If in doubt, please send us email to confirm whether the machine you have in mind is interesting enough. Also, keep in mind that since caching is very sensitive to the effects of contention and context switching, you should use a very lightly loaded machine.

Document your platform type and its clock rate.

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 associativity;
3. the block size; and
4. whether it is it write-through or write-back.

You can do this by either tracking down the literature on your processor (most of this information is on the web), or by writing simple routines that allow you to measure these parameters by measuring running times.

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 data (e.g., when summing an array of (long) integers);
2. writing data (e.g., when setting all elements of an array to a constant value); and
3. reading *and* writing data (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.

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(long int src[], long int dest[], long int n)
{
   long 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];
     }
}
```

After performing strength reduction and some other optimizations, we get the following:

```
void transpose2(long int src[], long int dest[], long int n)
{
   long int i, j;
   long int *next_src;
   long 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?

Optimization

Modify this code to 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.