# Homework 6

Josh Bradt

February 29, 2016

I modified the code from the second homework assignment such that the loop being timed was this:

```
for (size_t iter = 0; iter < numIters; iter++) {
#pragma omp parallel for default(none) shared(c,a,N) schedule(static)
    for (size_t i = 0; i < N; i++) {
        c[i] = a[i];
    }
    dummy(a, c);
}
```

I compiled the code using the Intel compiler with no optimization. (I had to do this since the optimizer was replacing the element-by-element copy with a single call to an optimized version of `memcpy`.) I then ran it for an array size of $N = 50\,000\,000$ with 20 iterations. For comparison, I ran the code both on my quad-core Mac Mini and on the HPCC's `dev-intel14` node, which has two 10-core processors. The results are listed in Table 1 and plotted in Figure 1.

In both cases, the performance seems to increase nearly linearly until the number of threads approaches the number of physical cores in the system.[1] After that, the performance levels off. This matches well with the suggested model of $R = \min(N_t R_1, R_{\max})$, where $N_t$ is the number of threads and $R_1$ and $R_{\max}$ are the rate with one thread and the maximum observed rate, respectively.

[1] Though in the case of the HPCC, this limit is instead at the number of cores in *one* of the two processors. This may reflect limitations of memory sharing between the two processors, or it may indicate that all of the threads were scheduled on one processor each time.

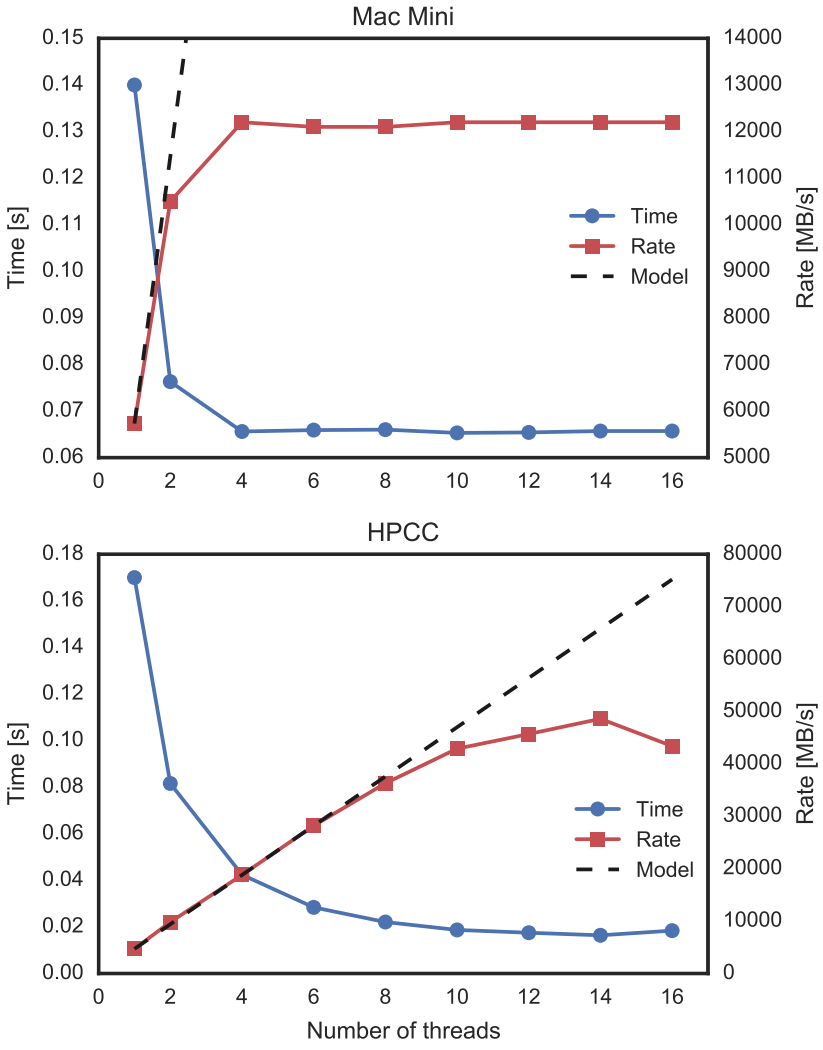


Figure 1: The results of running on the two computers. In each case, the performance levels off around when the number of threads equals the number of physical cores in one processor (4 for the Mac Mini, and 10 for the HPCC). The models shown are $R = N_t R_1$ where $N_t$ is the number of threads and $R_1$ is the rate for one thread.

| Num Threads | Mac Time [s] | Mac Rate [MB/s] | HPCC Time [s] | HPCC Rate [MB/s] |
|---|---|---|---|---|
| 1 | 0.1400 | 5730 | 0.1700 | 4700 |
| 2 | 0.0763 | 10 500 | 0.0816 | 9810 |
| 4 | 0.0656 | 12 200 | 0.0424 | 18 800 |
| 6 | 0.0659 | 12 100 | 0.0284 | 28 200 |
| 8 | 0.0660 | 12 100 | 0.0221 | 36 300 |
| 10 | 0.0653 | 12 200 | 0.0187 | 42 900 |
| 12 | 0.0654 | 12 200 | 0.0175 | 45 700 |
| 14 | 0.0657 | 12 200 | 0.0164 | 48 600 |
| 16 | 0.0657 | 12 200 | 0.0184 | 43 400 |

Table 1: Result of the measurement on both a quad-core Mac desktop and the $2 \times 10$-core `dev-intel14` node.