Report for assignment 5

cs20b018

We study and compare the cache effects of Integer Matrix multiplication using the common naive algorithm and a blocked matrix multiplication algorithm. In blocking, instead of operating on entire rows or columns, we operate on submatrices known as blocks. This is for reusing the data loaded into the faster memory level.

The following piece of code is used for the blocking algorithm:

```
for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            for(int k=0; k<n; k++)
                c[i][j] += a[i][k]*b[k][j];
```

The following piece of code is used for the blocking algorithm:

```
for(int kk = 0; kk < n; kk+= block_size){
        for(int jj = 0; jj < n; jj+= block_size){
            for(int i = 0; i < n; i++){
                for(int k = kk; k < min(kk + block_size - 1, n); k++){
                    for(int j = jj; j < min(jj + block_size - 1, n); j++){
                        c[i][j] += a[i][k]*b[k][j];
                    }
                }
            }
        }
    }
```

We use the perf tool in linux to measure the various metrics while running the matrix multiplication using either the naive or the blocked algorithm. We measure the counts for the following hardware events:

- L1-dache-loads number of loads into l1 cache
- L1-dcache-load-misses number of load misses in l1 cache
- L1-icache-load-misses number of load misses in l1 instruction cache
- l2_rqsts.references number of references in l2
- l2_rqsts.miss number of misses in l2

The data obtained is as follows

L1-dcache-loads

	0	8	16	32	64	128
128	5,32,33,619	5,85,26,610	6,66,09,195	6,82,20,406	6,90,22,514	6,51,46,871
256	34,95,83,290	44,04,70,947	47,35,22,739	49,37,19,192	50,19,24,755	50,40,10,341
512	2,68,15,02,021	3,38,82,45,306	3,61,40,28,960	3,74,74,88,955	3,81,10,19,219	3,83,94,03,042

L1-dcache-load-misses

	0	8	16	32	64	128
128	15,29,372	1,68,954	1,08,134	96,659	1,22,020	1,93,470
256	1,92,64,142	14,06,501	4,09,823	7,86,118	14,63,308	12,14,299
512	13,50,38,115	1,14,72,322	61,55,131	1,35,03,058	91,36,436	1,02,66,424

L1 data cache miss rate

	0	8	16	32	64	128
128	2.87%	0.29%	0.16%	0.14%	0.18%	0.30%
256	5.51%	0.32%	0.09%	0.16%	0.29%	0.24%
512	5.04%	0.34%	0.17%	0.36%	0.24%	0.27%

The results obtained from the l2 cache instructions from my pc were inconsistent and did not show any patterns.Hence nothing could be inferred from that.

Observations:-

First, we can see that, regardless of block size, the blocked algorithm performs better than the naive algorithm. This is because the blocked algorithm uses the cache more efficiently than the naive algorithm.

The best block size overall appears to be 16, since for most matrices the cache-misses are least when block size is picked to be 16.

A graph plotted with the cache miss rate on the y-axis and block size on the x-axis will be concave upwards from the above table.