Code Performance and Caches

Tabish Syed

COMP 273, Winter 2020

Cache Size

Loop Time

```
1         int[] A = new int[128 * 1024*1024];
2         double total = 0,start,stop;
3         int N = 8;
4         // Loop 1
5         for (int j = 0 ; j < N; ++j){
6             start = System.nanoTime();
7             for (int i = 0; i < A.length; ++i) A[i] *= 3;
8             stop = System.nanoTime();
9             double loop1Time = stop - start;
10            total += loop1Time;
11        }
12        double averageLoop1Time = total / N;
13        System.out.println( "Average time for loop 1 = " + averageLoop1Time);
14        // Loop 2
15        total = 0;
16        for (int j = 0 ; j < N; ++j){
17            start = System.nanoTime();
18            for (int i = 0; i < A.length; i+=32) A[i] *= 3;
19            stop = System.nanoTime();
20            double loop2Time = stop - start;
21            total += loop2Time;
22        }
23        double averageLoop2Time = total / N;
24        System.out.println("Average Time for loop 2 = " + averageLoop2Time);
25        System.out.println("Ratio of times = " + averageLoop1Time/averageLoop2Time);
26        System.out.println("But first loop does 32 times more work !!");
```

Loop Time

```
1         int[] A = new int[128 * 1024*1024];
2         double total = 0,start,stop;
3         int N = 8;
4         // Loop 1
5         for (int j = 0 ; j < N; ++j){
6             start = System.nanoTime();
7             for (int i = 0; i < A.length; ++i) A[i] *= 3;
8             stop = System.nanoTime();
9             double loop1Time = stop - start;
10            total += loop1Time;
11        }
12        double averageLoop1Time = total / N;
13        System.out.println( "Average time for loop 1 = " + averageLoop1Time);
14        // Loop 2
15        total = 0;
16        for (int j = 0 ; j < N; ++j){
17            start = System.nanoTime();
18            for (int i = 0; i < A.length; i+=32) A[i] *= 3;
19            stop = System.nanoTime();
20            double loop2Time = stop - start;
21            total += loop2Time;
22        }
23        double averageLoop2Time = total / N;
24        System.out.println("Average Time for loop 2 = " + averageLoop2Time);
25        System.out.println("Ratio of times = " + averageLoop1Time/averageLoop2Time);
26        System.out.println("But first loop does 32 times more work !!");
```

```
Average time for loop 1 = 1.3477324915E9
Average Time for loop 2 = 1.0673333525E8
Ratio of times = 12.627099943454638
But first loop does 32 times more work !!
```

Source: http://igoro.com/archive/gallery-of-processor-cache-effects, Prof. Paul Kry

Cache Block Size

```
1         System.out.println("A=[");
2         String xticklabels = "{";
3         int [] A = new int [128 *1024 *1024];
4         long start, stop;
5         int K = 1;
6         for ( int k = 0; k < 11; ++k){
7             start = System.nanoTime();
8
9             for(int i = 0; i < A.length; i += K)
10                A[i] *= 3;
11
12            stop = System.nanoTime();
13            System.out.println( k + " " + K + " " + (stop - start));
14            xticklabels += "\'2^{"+  k  + "}\',";
15            K *=2;
16        }
17        xticklabels += "}";
18        System.out.println("];");
19        System.out.println("plot(A(:,1), A(:,3));");
20        System.out.println("hold on;\nplot(A(:,1), A(:,3),'r*');");
21        System.out.println("xticklabels(" + xticklabels + ");");
22        System.out.println("xticks(A(:,1));");
23        System.out.println("title('Size of cache block');");
24        System.out.println("ylabel('Time ->');");
25        System.out.println("xlabel('Step size-> ');");
```

Cache Block Size

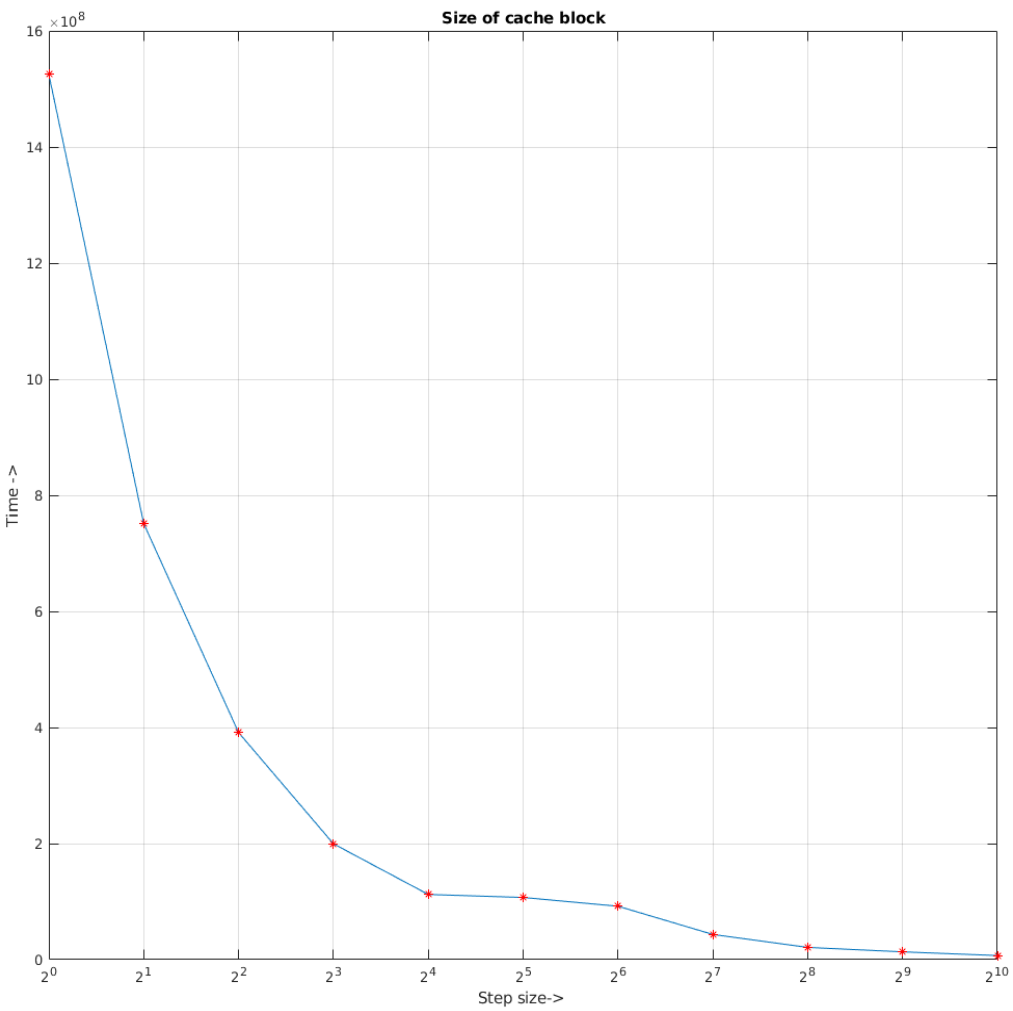
```
1         System.out.println("A=[");
2         String xticklabels = "{";
3         int [] A = new int [128 *1024 *1024];
4         long start, stop;
5         int K = 1;
6         for ( int k = 0; k < 11; ++k){
7             start = System.nanoTime();
8
9             for(int i = 0; i < A.length; i += K)
10                A[i] *= 3;
11
12            stop = System.nanoTime();
13            System.out.println( k + " " + K + " " + (stop - start));
14            xticklabels += "\'2^{"+  k  + "}\',";
15            K *=2;
16        }
17        xticklabels += "}";
18        System.out.println("];");
19        System.out.println("plot(A(:,1), A(:,3));");
20        System.out.println("hold on;\nplot(A(:,1), A(:,3),'r*');");
21        System.out.println("xticklabels(" + xticklabels + ");");
22        System.out.println("xticks(A(:,1));");
23        System.out.println("title('Size of cache block');");
24        System.out.println("ylabel('Time ->');");
25        System.out.println("xlabel('Step size-> ');");
```

```
A=[
0 1 1524888723
1 2 751189497
2 4 391276992
3 8 199589491
4 16 112349443
5 32 107168869
6 64 92080344
7 128 43472331
8 256 21159764
9 512 13482912
10 1024 7101547
];
plot(A(:,1), A(:,3));
hold on;
plot(A(:,1), A(:,3),'r*');
xticklabels({'2^{0}','2^{1}'...
xticks(A(:,1));
title('Size of cache block');
ylabel('Time ->');
xlabel('Step size-> ');
```

Source: http://igoro.com/archive/gallery-of-processor-cache-effects, Prof. Paul Kry

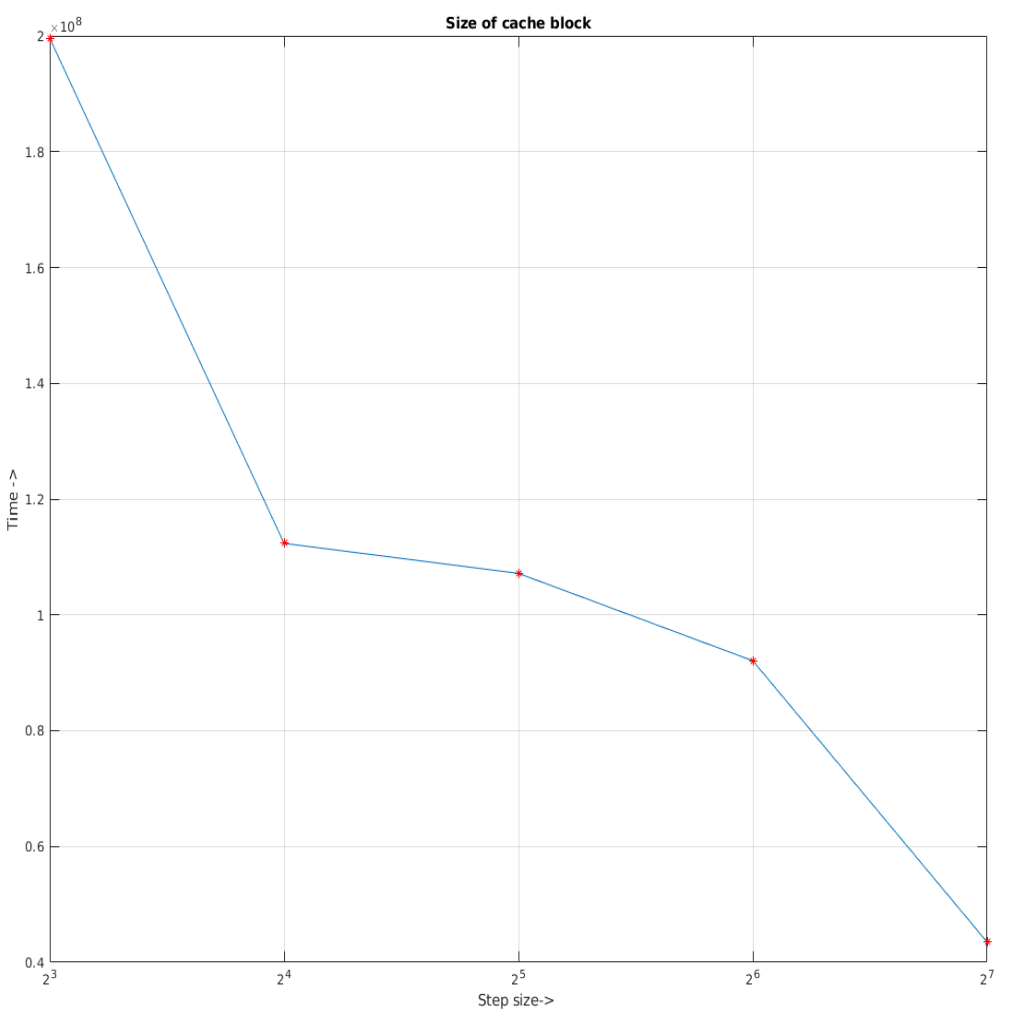
Cache Block Size

```
1        System.out.println("A=[");
2        String xticklabels = "{";
3        int [] A = new int [128 *1024 *1024];
4        long start, stop;
5        int K = 1;
6        for ( int k = 0; k < 11; ++k){
7            start = System.nanoTime();
8
9            for(int i = 0; i < A.length; i += K)
10               A[i] *= 3;
11
12           stop = System.nanoTime();
13           System.out.println( k + " " + K + " " +
14           xticklabels += "\'2^{"+  k  + "}\',";
15           K *=2;
16       }
17       xticklabels += "}";
18       System.out.println("];");
19       System.out.println("plot(A(:,1), A(:,3));");
20       System.out.println("hold on;\nplot(A(:,1), A
21       System.out.println("xticklabels(" + xticklab
22       System.out.println("xticks(A(:,1));");
23       System.out.println("title('Size of cache blo
24       System.out.println("ylabel('Time ->');");
25       System.out.println("xlabel('Step size-> ');"
```



Cache Block Size

```
1        System.out.println("A=[");
2        String xticklabels = "{";
3        int [] A = new int [128 *1024 *1024];
4        long start, stop;
5        int K = 1;
6        for ( int k = 0; k < 11; ++k){
7            start = System.nanoTime();
8
9            for(int i = 0; i < A.length; i += K)
10               A[i] *= 3;
11
12           stop = System.nanoTime();
13           System.out.println( k + " " + K + " " +
14           xticklabels += "\'2^{"+  k  + "}\',";
15           K *=2;
16       }
17       xticklabels += "}";
18       System.out.println("];");
19       System.out.println("plot(A(:,1), A(:,3));");
20       System.out.println("hold on;\nplot(A(:,1), A
21       System.out.println("xticklabels(" + xticklab
22       System.out.println("xticks(A(:,1));");
23       System.out.println("title('Size of cache blo
24       System.out.println("ylabel('Time ->');");
25       System.out.println("xlabel('Step size-> ');"
```



Cache Size

```
1          int steps = 64*1024*1024;   //Arbitrary large number
2          long start,stop;
3
4          System.out.println("B = [");
5          int size = 1024; // initial size = 2^10 * 2^2 = 4KB
6          String xticklabels = "{";
7          for (int j = 0; j < 20; ++j){
8              int[] A = new int[size];
9              start = System.nanoTime();
10
11             int lengthMod = A.length - 1;
12             for (int i = 0; i < steps; ++i)
13                 A[((i*32) & lengthMod)]++;
14
15             stop = System.nanoTime();
16             System.out.println(j + " " + size + " " + (stop - start));
17
18             System.gc(); // garbage collection
19             xticklabels += "\'2^{" + (j+12) + "}\',";
20             size *= 2;
21         }
22         xticklabels += "}";
23         System.out.println("];");
24         System.out.println("plot(B(:,1)+10, B(:,3));");
25         System.out.println("hold on;");
26         System.out.println("plot(B(:,1)+10, B(:,3),'r*');");
27         System.out.println("xticklabels(" + xticklabels + ");");
28         System.out.println("xticks(B(:,1)+10)");
29         System.out.println("title('Size of cache');");
30         System.out.println("ylabel('time ->');");
31         System.out.println("xlabel('Array Size (Bytes) -> ');");
```

Cache Size

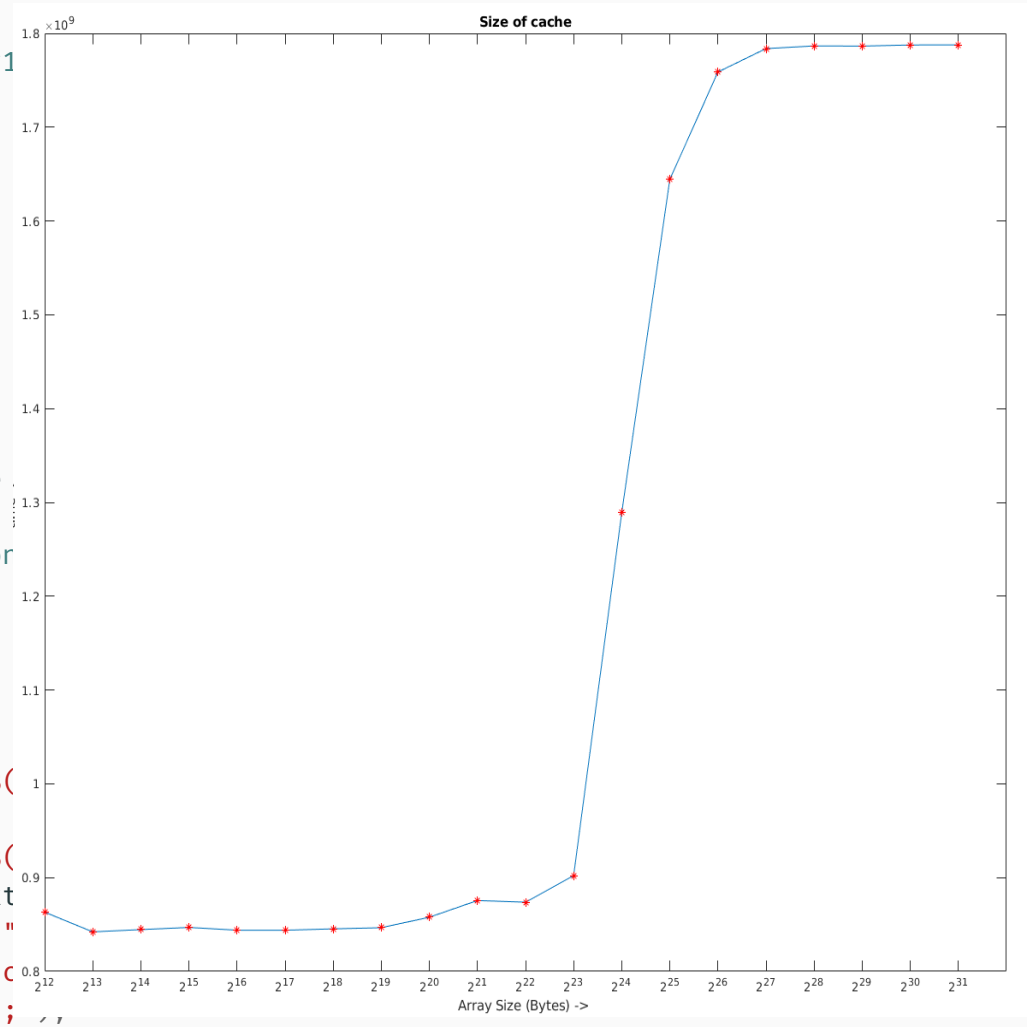
```
1         int steps = 64*1024*1024;   //Arbitrary large number
2         long start,stop;
3
4         System.out.println("B = [");
5         int size = 1024; // initial size = 2^10 * 2^2 = 4KB
6         String xticklabels = "{";
7         for (int j = 0; j < 20; ++j){
8             int[] A = new int[size];
9             start = System.nanoTime();
10
11            int lengthMod = A.length - 1;
12            for (int i = 0; i < steps; ++i)
13                A[((i*32) & lengthMod)]++;
14
15            stop = System.nanoTime();
16            System.out.println(j + " " + size + " " + (stop - start));
17
18            System.gc(); // garbage collection
19            xticklabels += "\'2^{" + (j+12) + "}\',";
20            size *= 2;
21        }
22        xticklabels += "}";
23        System.out.println("];");
24        System.out.println("plot(B(:,1)+10, B(:,3));");
25        System.out.println("hold on;");
26        System.out.println("plot(B(:,1)+10, B(:,3),'r*');");
27        System.out.println("xticklabels(" + xticklabels + ");");
28        System.out.println("xticks(B(:,1)+10)");
29        System.out.println("title('Size of cache');");
30        System.out.println("ylabel('time ->');");
31        System.out.println("xlabel('Array Size (Bytes) -> ');");
```

```
B = [
0 1024 863180725
1 2048 842193845
2 4096 844589679
3 8192 846904579
4 16384 843970741
5 32768 843925466
6 65536 845388826
7 131072 846607085
8 262144 857787294
9 524288 875579717
10 1048576 873786842
11 2097152 902210755
12 4194304 1288983422
13 8388608 1644425253
14 16777216 1758837615
15 33554432 1783776047
16 67108864 1786640664
17 134217728 1786447414
18 268435456 1787642556
19 536870912 1787765700
];
plot(B(:,1)+10, B(:,3));
hold on;
plot(B(:,1)+10, B(:,3),'r*');
xticklabels({'2^{12}','2^{13}',...
xticks(B(:,1)+10)
title('Size of cache');
ylabel('time ->');
xlabel('Array Size (Bytes) -> ');
```

Source: http://igoro.com/archive/gallery-of-processor-cache-effects, Prof. Paul Kry

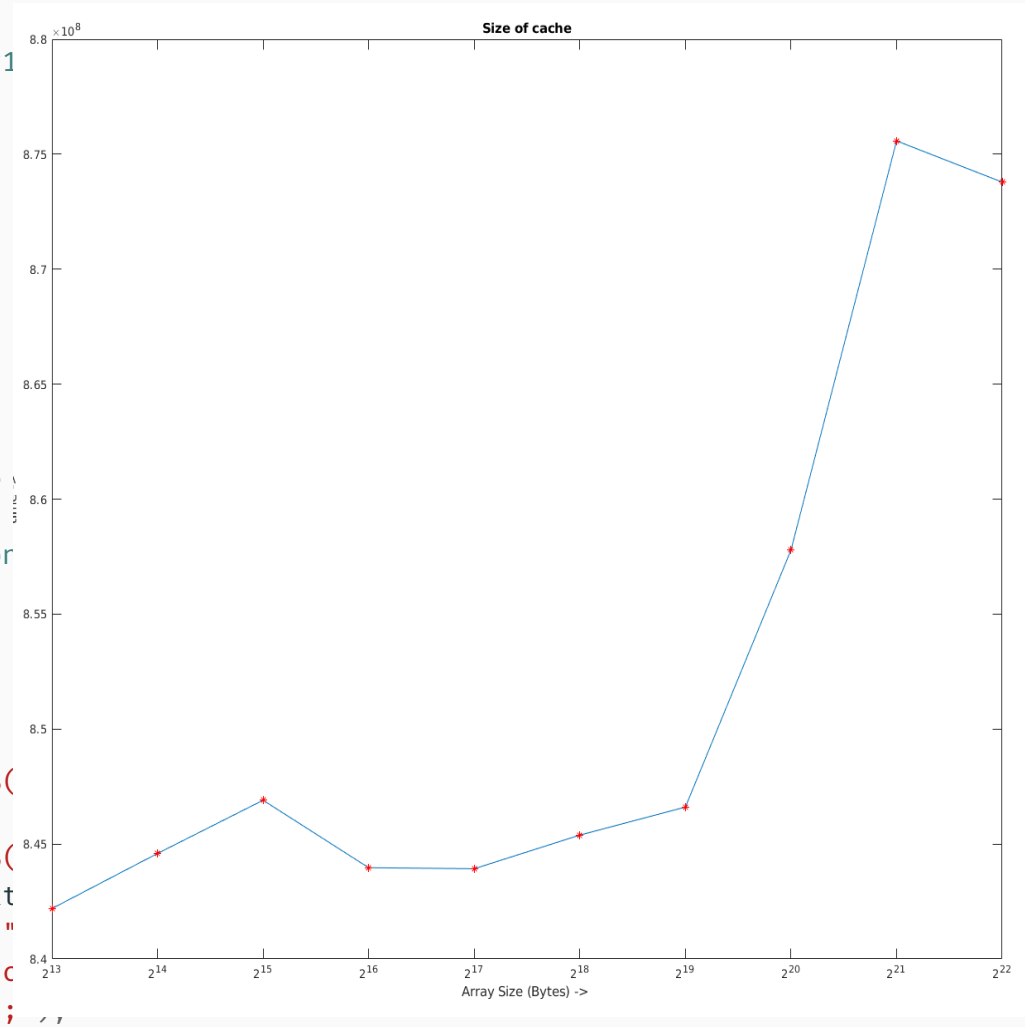
Cache Size

```
1         int steps = 64*1024*1024;   //Arbitrary large number
2         long start,stop;
3
4         System.out.println("B = [");
5         int size = 1024; // initial size = 2^1
6         String xticklabels = "{";
7         for (int j = 0; j < 20; ++j){
8             int[] A = new int[size];
9             start = System.nanoTime();
10
11            int lengthMod = A.length - 1;
12            for (int i = 0; i < steps; ++i)
13                A[((i*32) & lengthMod)]++;
14
15            stop = System.nanoTime();
16            System.out.println(j + " " + size
17
18            System.gc(); // garbage collectior
19            xticklabels += "\'2^{" + (j+12) +
20            size *= 2;
21        }
22        xticklabels += "}";
23        System.out.println("];");
24        System.out.println("plot(B(:,1)+10, B(
25        System.out.println("hold on;");
26        System.out.println("plot(B(:,1)+10, B(
27        System.out.println("xticklabels(" + xt
28        System.out.println("xticks(B(:,1)+10)"
29        System.out.println("title('Size of cac
30        System.out.println("ylabel('time ->');
31        System.out.println("xlabel('Array Size (Bytes) -> ');");
```



Cache Size

```
1         int steps = 64*1024*1024;   //Arbitrary large number
2         long start,stop;
3
4         System.out.println("B = [");
5         int size = 1024; // initial size = 2^1
6         String xticklabels = "{";
7         for (int j = 0; j < 20; ++j){
8             int[] A = new int[size];
9             start = System.nanoTime();
10
11            int lengthMod = A.length - 1;
12            for (int i = 0; i < steps; ++i)
13                A[((i*32) & lengthMod)]++;
14
15            stop = System.nanoTime();
16            System.out.println(j + " " + size
17
18            System.gc(); // garbage collectior
19            xticklabels += "\'2^{" + (j+12) +
20            size *= 2;
21        }
22        xticklabels += "}";
23        System.out.println("];");
24        System.out.println("plot(B(:,1)+10, B(
25        System.out.println("hold on;");
26        System.out.println("plot(B(:,1)+10, B(
27        System.out.println("xticklabels(" + xt
28        System.out.println("xticks(B(:,1)+10)"
29        System.out.println("title('Size of cac
30        System.out.println("ylabel('time ->');
31        System.out.println("xlabel('Array Size (Bytes) -> ');");
```



Machine Configuration

```
$ lscpu
...
L1d cache:                       32K
L1i cache:                       32K
L2 cache:                        1024K
L3 cache:                        14080K
...
```

Instruction Parallelism

Instruction Level Parallelism

```
1         int steps = 256 * 1024*1024;
2         int[] A = new int[8];
3         double start,stop;
4         int N = 8;
5         // Loop 1
6         start = System.nanoTime();
7         for (int i = 0; i < steps; ++i){
8             A[0]++; A[1]++; A[2]++; A[3]++;
9             A[4]++; A[5]++; A[6]++; A[7]++;
10        }
11        stop = System.nanoTime();
12        double loop1Time = (stop - start) / steps;
13        System.out.println( "Average time for loop 1 = " + loop1Time);
14        // Loop 2
15        start = System.nanoTime();
16        for (int i = 0; i < steps; ++i) {
17            A[0]++; A[0]++; A[0]++; A[0]++;
18            A[7]++; A[7]++; A[7]++; A[7]++;
19        }
20        stop = System.nanoTime();
21        double loop2Time = (stop - start)/steps;
22        System.out.println("Average Time for loop 2 = " + loop2Time);
23        System.out.println("Ratio of times = " + loop1Time/loop2Time);
24        System.out.println("But the loops do the same amount work !!");
```

Instruction Level Parallelism

```
1         int steps = 256 * 1024*1024;
2         int[] A = new int[8];
3         double start,stop;
4         int N = 8;
5         // Loop 1
6         start = System.nanoTime();
7         for (int i = 0; i < steps; ++i){
8             A[0]++; A[1]++; A[2]++; A[3]++;
9             A[4]++; A[5]++; A[6]++; A[7]++;
10        }
11        stop = System.nanoTime();
12        double loop1Time = (stop - start) / steps;
13        System.out.println( "Average time for loop 1 = " + loop1Time);
14        // Loop 2
15        start = System.nanoTime();
16        for (int i = 0; i < steps; ++i) {
17            A[0]++; A[0]++; A[0]++; A[0]++;
18            A[7]++; A[7]++; A[7]++; A[7]++;
19        }
20        stop = System.nanoTime();
21        double loop2Time = (stop - start)/steps;
22        System.out.println("Average Time for loop 2 = " + loop2Time);
23        System.out.println("Ratio of times = " + loop1Time/loop2Time);
24        System.out.println("But the loops do the same amount work !!");
```

```
Average time for loop 1 = 48.02225795388222
Average Time for loop 2 = 77.47476292029023
Ratio of times = 0.6198438839146863
But the loops do the same amount work !!
```

Source: http://igoro.com/archive/gallery-of-processor-cache-effects, Prof. Paul Kry