



JavaOne

JUNE 2-5



Benchmarks
Gone
Wild !!!



www.kodewerk.com



Me

- 👤 Work as independent (a.k.a. freelancer)
 - performance tuning services
 - benchmarking
 - Java performance course and seminars

👤 Co-author:
www.javaperformancetuning.com

👤 Nominated Sun Java Champion

👤 Blah blah blah



www.kodewerk.com





Java Performance Tuning
Chania Crete
Sept 21-24, 2009



Disclaimer

The resemblance of any opinion, recommendation or comment made during this presentation to performance tuning advice is merely coincidental.



www.kodewerk.com



I've decided to start my talks with this disclaimer because of all of the bad performance tuning advice that I've seen being given with the best of intentions. I've also seen a lot of good advice being misused or misunderstood and then misapplied. So with all other advice it has to be evaluated critically to see if it really applies to you

Stutter

Measure cost of executing some algorithm



www.kodewerk.com



This one is not much different than many others.

```
public int factorial(int n) {
    if ( n == 0) return 1;
    if (n <= 1) return 1;
    else return n * factorial (n - 1);
}

public void execute() {
    long total = 0;
    long time;
    for ( int i = 0; i < 20; i++) {
        long start = System.nanoTime();
        this.factorial( 100);
        total += (System.nanoTime() - start);
    }
    System.out.println( total / 20);
}
```



www.kodewerk.com



Results

👤 Average reported as 8950 nanoseconds

👤 Multiple choice question

A. cost of calculating $100!$ is almost 9 ms

B. something doesn't look right



www.kodewerk.com



Individual Data Points

1.	11000	11.	8000
2.	8000	12.	9000
3.	9000	13.	8000
4.	8000	14.	9000
5.	8000	15.	16000
6.	9000	16.	9000
7.	9000	17.	8000
8.	8000	18.	8000
9.	9000	19.	9000
10.	8000	20.	8000



www.kodewerk.com



Samples	20
Average	8950.0
Variance	3313158
Std. Dev.	1820
Median	9000.0
Minimum	8000
Maximum	16000



www.kodewerk.com



🚀 Larger times suggest interference with benchmark

- JIT and OSR
- eliminate possible interference
- warmup to put bench in steady state

🚀 All data values end with X000

- resolution of nanoseconds look suspect
- duration benchmark is too short
 - increase size of the unit of work to at least 100ms



www.kodewerk.com



```
public int factorial(int n) {
    if ( n == 0) return 1;
    if (n <= 1) return 1;
    else return n * factorial( n - 1);
}

public void execute() {
    long total = 0;
    long time;
    for ( int i = 0; i < 20; i++) {
        long start = System.nanoTime();
        this.factorial( 100);
        total += (System.nanoTime() - start);
    }
    System.out.println( total / 20);
}
```



www.kodewerk.com



Harness is bad

A Hoisting We Will Go

👤 What we know

- repeating operations in a loop is bad for performance

👤 How we will prove it

- perform an operation that contains some redundant calculation in a loop. Repeat with the redundant calculation moved to outside of the loop.



www.kodewerk.com



```
public void noHoist( int a, int b) {
    int total = Integer.MIN_VALUE;
    for ( int i = 0; i < LOOP_COUNT; i++)
        total += (a + b);
}

public void hoist( int a, int b) {
    int total = Integer.MIN_VALUE;
    int hoisted = a + b;
    for ( int i = 0; i < LOOP_COUNT; i++)
        total += hoisted;
}
```



www.kodewerk.com



```
public Population hoistTest() {
    this.hoist( 1, 1);
    this.hoist( 1, 1);
    System.out.println("Hoisting");
    for ( int i = 0; i < TRIALS; i++) {
        long start =
            System.currentTimeMillis();
        this.hoist( 1, 1);
        hoisted.add(
            System.currentTimeMillis() -
            start);
    }
    return hoisted;
}
```



www.kodewerk.com



🚀 Warmup to settle JIT

🚀 Monitor GC

- reset heap to eliminate GC pause during run

🚀 Monitor System

- eliminate any other interfering factors

🚀 Switch settings

- `-server -Xmx512m`



www.kodewerk.com



	non-hoisted	hoisted
Samples (n)	10	10
Average	0.0	0.0
Variance	0.0	0.0
Std. Dev.	0.0	0.0
Median	0.0	0.0
Minimum	0.0	0.0
Maximum	0.0	0.0



www.kodewerk.com



We just figured out that Java is really the fastest runtime on the planet

```
public void noHoist( int a, int b) {  
    int total = Integer.MIN_VALUE;  
    for ( int i = 0; i < LOOP_COUNT; i++)  
        total += (a + b);  
}
```

🦋 Observations

- no references outside scope
- nothing returned

🦋 Conclusion

- code is dead



www.kodewerk.com



Fragile Fix

```
public int noHoist( int a, int b) {  
    int total = Integer.MIN_VALUE;  
    for ( int i = 0; i < LOOP_COUNT; i++)  
        total += (a + b);  
    return total;  
}
```



www.kodewerk.com



	non-hoisted	hoisted
Samples (n)	10	10
Average	443.6	443.1
Variance	2.7	0.7
Std. Dev.	1.6	0.9
Median	443	443
Minimum	442	442
Maximum	448	445



www.kodewerk.com



Adding a return value breaks the possibility of optimizing away the call

The Memory Hog

Measure the memory requirements of two different algorithms



www.kodewerk.com



Local

```
public void execute1( int size, int increment) {
    long count = 0;
    try {
        while (true) {
            byte[] array = new byte[1024 * size];
            size += increment;
            count++;
        }
    } catch( Throwable t) {
        array = null;
        System.out.println( t.toString());
        System.out.println("Incremented: "+ count);
    }
}
```



www.kodewerk.com



Instance

```
private byte[] array;
public void execute2( int size, int increment) {
    long count = 0;
    try {
        while (true) {
            array = new byte[1024 * size];
            size += increment;
            count++;
        }
    } catch( Throwable t) {
        array = null;
        System.out.println( t.toString());
        System.out.println("Incremented: "+ count);
    }
}
```



www.kodewerk.com



Results

- 👤 start with 100k and increment by 100k
- 👤 execute1 runs out of memory on the 1163 iteration
- 👤 execute2 runs out of memory on the 582 iteration



www.kodewerk.com



clue, $582 * 2 = 1164$

Local

```
public void execute1( int size, int increment) {
    long count = 0;
    try {
        while (true) {
            byte[] array = new byte[1024 * size];
            size += increment;
            count++;
        }
    } catch( Throwable t) {
        array = null;
        System.out.println( t.toString());
        System.out.println("Incremented: "+ count);
    }
}
```



www.kodewerk.com



Instance

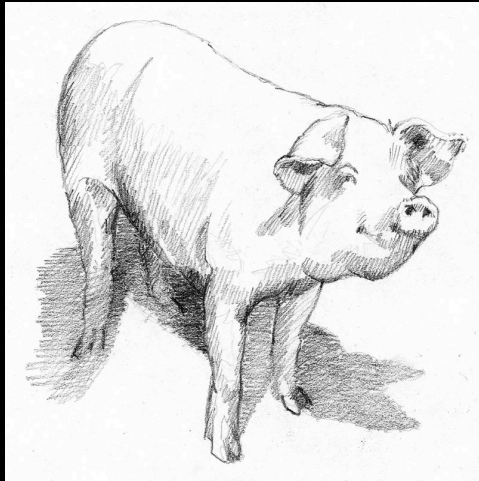
```
private byte[] array;
public void execute2( int size, int increment) {
    long count = 0;
    try {
        while (true) {
            array = new byte[1024 * size];
            size += increment;
            count++;
        }
    } catch( Throwable t) {
        array = null;
        System.out.println( t.toString());
        System.out.println("Incremented: "+ count);
    }
}
```



www.kodewerk.com



This may seem obvious but it is something that we rarely think of



www.kodewerk.com

Kodewerk
Java™ Performance Services

The Matrix

A simple matrix operation



www.kodewerk.com



```
public int[] mul( int a[][], int b[][]) {
    int rows = a.length;
    int columns = (a[0]).length;
    int[] result = new int[rows];
    for ( int i = 0; i < rows; i++)
        for ( int j = 0; j < columns; j++)
            result[i] += a[i][j]*b[j][i];
    return result;
}

public void execute() {
    int[][] a={{1,1},{2,2},{3,3},{4,4},{5,5},{6,6},{7,7},{8,8}};
    int[][] b={{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8}};
    warmup( a,b);
    long start = System.currentTimeMillis();
    for (int i = 0; i < 500000; i++) mul( b, a);
    System.out.println( System.currentTimeMillis() - start);
    start = System.currentTimeMillis();
    for (int i = 0; i < 500000; i++) mul( a,b);
    System.out.println( System.currentTimeMillis() - start);
}
```



www.kodewerk.com

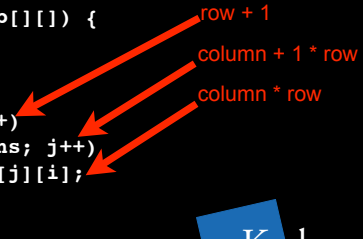


🏠 8x2 runs in 102ms

🏠 2x8 runs in 76ms

🏠 Puzzle: why?

```
public int[] mul( int a[][], int b[][]) {  
    int rows = a.length;  
    int columns = (a[0]).length;  
    int[] result = new int[rows];  
    for ( int i = 0; i < rows; i++)  
        for ( int j = 0; j < columns; j++)  
            result[i] += a[i][j]*b[j][i];  
    return result;  
}
```



www.kodewerk.com



Counting Statements

Size	8x2	2x8	Difference
rows + 1	9	3	6
columns * 1 + rows	24	18	6
columns * rows	16	16	0
Total	49	37	~75%
Runtime	102	75	~75%



www.kodewerk.com

Kodewerk
Java™ Performance Services

a lot of things go into the runtime number so this could still be spurious

Algorithms often have cut off points where they become less efficient than an alternative implementation. If you understand where that point is, you can test for it and switch implementations.
bubble sort, quicksort vs. list size.

Matrix Reloaded

👤 Objective: yet another matrix multiply to puzzle over



www.kodewerk.com



```
public long execute1() {
    matrix = new long[ 1000][1000];
    long start = System.currentTimeMillis();
    for ( int index = 0; index < 100; index++)
        for ( int i = 0; i < 1000; i++)
            for ( int j = 0; j < 1000; j++)
                matrix[i][j] *= 2L;
    return System.currentTimeMillis() - start;
}

public long execute2() {
    matrix = new long[ 1000][1000];
    long start = System.currentTimeMillis();
    for ( int index = 0; index < 100; index++)
        for ( int i = 0; i < 1000; i++)
            for ( int j = 0; j < 1000; j++)
                matrix[j][i] *= 2L;
    return System.currentTimeMillis() - start;
}
```



www.kodewerk.com



square matrix

Results

🏠 execute1 runs in 509 ms

🏠 execute2 runs in 1597 ms

```
public void init() {  
    matrix = new long[ 1000][1000];  
    for ( int index = 0; index < 100; index++)  
        for ( int i = 0; i < 1000; i++)  
            for ( int j = 0; j < 1000; j++)  
                matrix[i][j] = 100L;  
}
```



www.kodewerk.com



The Code

```
public long execute1() { // 509ms
    matrix = new long[ 1000][1000];
    long start = System.currentTimeMillis();
    for ( int index = 0; index < 100; index++)
        for ( int i = 0; i < 1000; i++)
            for ( int j = 0; j < 1000; j++)
                matrix[i][j] *= 2L;
    return System.currentTimeMillis() - start;
}

public long execute2() { // 1597ms
    matrix = new long[ 1000][1000];
    long start = System.currentTimeMillis();
    for ( int index = 0; index < 100; index++)
        for ( int i = 0; i < 1000; i++)
            for ( int j = 0; j < 1000; j++)
                matrix[j][i] *= 2L;
    return System.currentTimeMillis() - start;
}
```



www.kodewerk.com



thrashing L1/L2 cache, moral, hardware matters, WORA is a bit of a marketing pitch

Accidental Throttle

Low pause concurrent collector will yield better user response times over a full stop the world collector



www.kodewerk.com



Simulation

```
public void run() {  
    this.setRunning( true);  
    while ( isRunning()) {  
        long start = System.currentTimeMillis(  
            workspace.request());  
        doUnitOfWork();  
        population.add( System.currentTimeMillis() - start);  
    }  
}
```



www.kodewerk.com



External Resource Simulation

```
public void request() {
    synchronized( lock) {
        while ( threadCount >= maxThreadCount )
            try {
                lock.wait();
            } catch (InterruptedException e) {}
        threadCount++;
        lock.notify();
    }

    try {
        Thread.sleep( SERVICE_TIME);
    } catch (InterruptedException e) {
        synchronized ( lock) {
            threadCount--;
            lock.notify();
        }
    }
}
```



www.kodewerk.com



Unit of Work

```
public int doUnitOfWork() {  
    int length = 0;  
    String buffer = "a";  
    for ( int i = 0; i < 2000000; i++)  
        pinned[ i % 10000] = buffer.concat( "John");  
    return buffer.length();  
}
```




www.kodewerk.com



Results

 -server -XX:+UseConcMarkSweepGC

● 8781ms

 -server

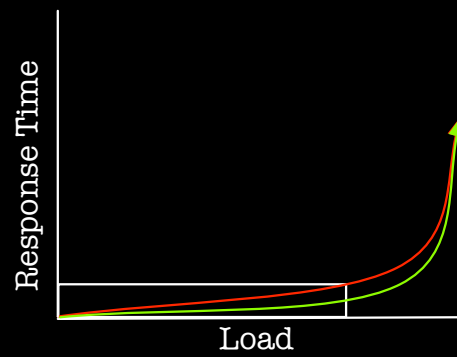
● 7342ms



www.kodewerk.com



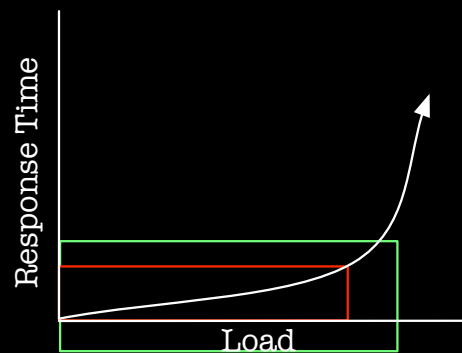
GC Savings



www.kodewerk.com



External System Loss



www.kodewerk.com





www.kodewerk.com



