JPMorganChase presents
Java Debugging and Troubleshooting with No Source Code in Sight
By Minoy Mathew
The proliferation of the black and the gray boxes

- Buy - Build - Reuse - Assemble
- Vendor products (black box)
- Open source products (gray box)
- Reusable components (gray box)
- Dependencies with other systems (black box)
Topics

- Information Visualization
- Information Dumps
- Debugging without Source
- Correlating Information
Thinking like the programmer who wrote it!

- Patterns
- Globalized Industry
- Common mistakes and best practices
- She would have had to make it work with another black box too
- Some rules are ubiquitous
What’s in a name - Major Major Major
Leaking Map

```java
package test;

public class LeakTester implements Runnable{

    public static void main(String[] args) throws Exception{
        Thread test1 = new Thread(new LeakTester());
        test1.setName("leaking Map populator");
        test1.start();
    }

    public void run(){
        LeakingMapHolder leakingMapHolder = new LeakingMapHolder();
        for(int i=0;true ;i++){
            try{
                Thread.currentThread().sleep(50);
                leakingMapHolder.leakingMap.put(new String("Leaking map Key" + i),
                new String("Leaking map Value" + i));
            }
            catch(Exception e){}
        }
    }
}
```
Tools: Visual VM
# Heap Dump

**VisualVM Milestone 3**

File Edit View Profile Tools Window Help

- **Applications**
  - Local
  - Test.LeakTester (pid 4524)
    - heapdump-120031019503788.hprof

**test.LeakTester (pid 4524)**

Heap Dump

**Classes**

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Instances [%]</th>
<th>Instances</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>15620</td>
<td>26%</td>
<td>374880</td>
</tr>
<tr>
<td>char[]</td>
<td>15411</td>
<td>24%</td>
<td>826539</td>
</tr>
<tr>
<td>int[]</td>
<td>12125</td>
<td>20%</td>
<td>1058290</td>
</tr>
<tr>
<td>java.util.HashMap.Entry</td>
<td>8589</td>
<td>13%</td>
<td>165536</td>
</tr>
<tr>
<td>byte[]</td>
<td>6889</td>
<td>11%</td>
<td>159982</td>
</tr>
<tr>
<td>short[]</td>
<td>1322</td>
<td>2%</td>
<td>159982</td>
</tr>
<tr>
<td>java.lang.Object[]</td>
<td>1038</td>
<td>1%</td>
<td>23666</td>
</tr>
<tr>
<td>java.lang.reflect.Method</td>
<td>417</td>
<td>1%</td>
<td>32159</td>
</tr>
<tr>
<td>java.lang.Class[]</td>
<td>360</td>
<td>1%</td>
<td>4002</td>
</tr>
<tr>
<td>java.lang.String[]</td>
<td>338</td>
<td>1%</td>
<td>4996</td>
</tr>
<tr>
<td>java.util.SoftReference</td>
<td>330</td>
<td>1%</td>
<td>9600</td>
</tr>
<tr>
<td>java.util.HashMap.Entry[]</td>
<td>285</td>
<td>1%</td>
<td>86416</td>
</tr>
<tr>
<td>java.lang.Integer</td>
<td>285</td>
<td>1%</td>
<td>34036</td>
</tr>
<tr>
<td>java.util.HashMap</td>
<td>276</td>
<td>1%</td>
<td>11346</td>
</tr>
<tr>
<td>java.lang.String[]</td>
<td>266</td>
<td>1%</td>
<td>9588</td>
</tr>
<tr>
<td>java.util.HashMap.Entry[]</td>
<td>195</td>
<td>0%</td>
<td>4690</td>
</tr>
<tr>
<td>java.lang.Object[]</td>
<td>183</td>
<td>0%</td>
<td>6036</td>
</tr>
<tr>
<td>java.util.List</td>
<td>180</td>
<td>1%</td>
<td>4092</td>
</tr>
<tr>
<td>java.util.Map</td>
<td>176</td>
<td>1%</td>
<td>5663</td>
</tr>
<tr>
<td>java.util.List</td>
<td>156</td>
<td>1%</td>
<td>2628</td>
</tr>
</tbody>
</table>
Homing in on the leak
JMX: Introspection into the JVM -- JConsole
Named your fear must be before banish it you can
- Master Yoda (Star Wars)
Information Visualization: Squiggly Lines - Profession - Neurologist - EEG
Squiggly Lines - Profession - Java Developer

~ 100MB lost tenured floor

Normal activity after tenured lost space
Garbage - When Apps Panic

Temporary GC Panic

Hourly cycle of tenured memory growth

Growth in Tenured Space
Garbage - Finding Patterns

Full GC activity from Weblogic Console

Definite Tenured Space Growth

Manual clearing of benchmarks

Flush point
Garbage - Findings

GC panic was recoverable

Indicative of an expiring cache usage?

Activity persisted even with no load

A background thread or process?

Cyclical nature to recovery (~ 1 hour)

Indicative that cache and background thread are related?

Root Cause - What Really Happened:

A culprit background thread refreshing lookup data into an UNBOUNDED cache that had a TIMED EXPIRATION of an hour.
Garbage - Problem Solved
Garbage and Squiggly Lines - Tools and Learnings

Know your GC algorithms and their repercussions

java -Xms600m -Xmx600m -server -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSClassUnloadingEnabled -XX:PermSize=64m -XX:MaxPermSize=256m -XX:NewSize=96m

References:

http://java.sun.com/developer/technicalArticles/Programming/GCPortal/

http://www.tagtraum.com/gcviewer.html

Increase your GC verbosity!

Force the GCs and look for patterns

Load Testing

http://jakarta.apache.org/jmeter/

http://grinder.sourceforge.net/

Association, correlation and causal analysis

THE POWER OF EVENTS
AN INTRODUCTION TO COMPLEX EVENT PROCESSING IN DISTRIBUTED ENTERPRISE SYSTEMS

DAVID LUCKHAM
package test.sql;
import java.lang.management.ManagementFactory;

public class DBTest {
    private Connection connection = null;
    private OracleConnection oracleConnection = null;
    public DBTest(){
        try {
            String processName = getClientInfo();
            Class.forName("oracle.jdbc.driver.OracleDriver");
            connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl10", "hr", "orcl");
            //connection.set
            oracleConnection = (OracleConnection) connection;
            oracleConnection.setClientIdentifier(processName);
        } catch (Exception e) {} 
    }
    
    public void executeSQL(int employee_id) throws Exception{
        PreparedStatement statement = connection.prepareStatement("select * from employees where employee_id = " + employee_id);
        statement.executeQuery();
    }
    
    private String getClientInfo(){
        Thread.currentThread().setName("@DBTest");
        RuntimeMXBean rt = ManagementFactory.getRuntimeMXBean();
        return rt.getName() + Thread.currentThread().getName();
    }
    
    public static void main(String[] args) throws Exception{
        Thread dbTester = new Thread(new DBTester());
        dbTester.start();
    }
    
    public void run(){
        DBTest dbtest = new DBTest();
        for(int i=0;true ;i++)
            try {
                Thread.currentThread().sleep(500);
                dbtest.executeQuery(i);
            } catch (Exception e) {} 
    }
}
Correlating it to DB info

![Image of Oracle Enterprise Manager interface](image.png)

### Additional Monitoring Links
- Top Activity
- Top Consumers
- Duplicate SQL
- Blocking Sessions
- Hang Analysis
- Instance Locks
- Instance Activity
- Baseline Normalized Metrics
- Snapshots
- SQL Tuning Sets
Correlating it to DB info
Correlating it to DB info

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>SQL Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2a80860372</td>
<td><code>select order# from access3 where d_objf# = 1</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select object# from access3 where d_objf# = 1</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 108</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 160</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 205</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 213</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 246</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 14</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 159</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 188</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 191</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 249</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 150</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 223</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 52</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 224</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 276</code></td>
</tr>
<tr>
<td>0x2a80860377</td>
<td><code>select * from employees where employee_id = 151</code></td>
</tr>
<tr>
<td>0x2a860377</td>
<td><code>select * from employees where employee_id = 54</code></td>
</tr>
<tr>
<td>0x2a860377</td>
<td><code>select * from employees where employee_id = 236</code></td>
</tr>
</tbody>
</table>
More profiling

![VisualVM Milestone 3](image)

**test.DBTester (pid 4184)**

**Profiler**

- Profile: CPU
- Memory
- Stop

**Status:** profiling running (Analyze Performance)

**Profiling results**

<table>
<thead>
<tr>
<th>Hot Spots - Method</th>
<th>Self time (%)</th>
<th>Self time</th>
<th>Invocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.net.SocketInputStream.read (byte[], int, int)</td>
<td></td>
<td>1541 ms</td>
<td>11</td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.writeObject (Object, boolean)</td>
<td>3.4 ms</td>
<td>4396</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectStreamClass.getFields (boolean)</td>
<td>2.85 ms</td>
<td>353</td>
<td></td>
</tr>
<tr>
<td>java.net.SocketOutputStream.socketWrite (byte[], int, int)</td>
<td>1.75 ms</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.flushDataOutputStream.setBlockDataMode (boolean)</td>
<td>1.67 ms</td>
<td>2206</td>
<td></td>
</tr>
<tr>
<td>java.util.TreeMap.putAfterInsertion (java.util.TreeMap.Entry)</td>
<td>1.32 ms</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.DataOutputStream.drain ()</td>
<td>1.11 ms</td>
<td>1999</td>
<td></td>
</tr>
<tr>
<td>java.lang.String.compareTo (String)</td>
<td>1.05 ms</td>
<td>1911</td>
<td></td>
</tr>
<tr>
<td>java.lang.String.compareTo (Object)</td>
<td>1.7 ms</td>
<td>1911</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectStreamClass.lookup (Class, boolean)</td>
<td>1.7 ms</td>
<td>383</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.Handshake.lookup (Object)</td>
<td>1.5 ms</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.Handshake.hash (Object)</td>
<td>1.5 ms</td>
<td>1984</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.defaultWriteFields (Object, java.io.ObjectStreamClass)</td>
<td>0.542 ms</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>java.io.ObjectOutputStream.writeHandle (int)</td>
<td>0.525 ms</td>
<td>699</td>
<td></td>
</tr>
<tr>
<td>java.lang.Math.min (int, int)</td>
<td>0.827 ms</td>
<td>2512</td>
<td></td>
</tr>
<tr>
<td>java.util.TreeMap.entry (java.util.TreeMap.Entry)</td>
<td>0.620 ms</td>
<td>3152</td>
<td></td>
</tr>
<tr>
<td>java.lang.String.equals (Object)</td>
<td>0.785 ms</td>
<td>2639</td>
<td></td>
</tr>
</tbody>
</table>

[HJUG Mar 2008]

[JPMorganChase]
Always in motion is the future

Master Yoda (Star Wars)
package test;

public class LockTest implements Runnable{
    static String lock1 = "lock1";
    static String lock2 = "lock2";
    boolean clockwise = false;
    LockTest(boolean clockwise){
        this.clockwise = clockwise;
    }

    public static void main(String[] args) throws Exception{
        Thread lockTest1 = new Thread(new LockTest(true));
        lockTest1.setName("Test Thread 1");
        lockTest1.start();
        Thread lockTest2 = new Thread(new LockTest(false));
        lockTest2.setName("Test Thread 2");
        lockTest2.start();
    }

    public void run(){
        if(clockwise){clockwise();
        }else{ anticlockwise();
        }
    }

    public void clockwise(){
        synchronized(lock1){
            try{
                Thread.currentThread().sleep(1000);
            }catch(Exception e){}
            synchronized(lock2){
            }
        }
    }

    public void anticlockwise(){
        synchronized(lock2){
            try{
                Thread.currentThread().sleep(1000);
            }catch(Exception e){}
            synchronized(lock1){
            }
        }
    }
}

DEADLOCK!
Thread dump: kill -3

VisualVM Milestone 3

File Edit View Profile Tools Window Help

Applications

<Unknown> (pid 4588)
<Unknown> (rcb)
<Unknown> (gc)
VisualVM
Remote
VM Core Dumps

Thread Dump

Test Thread 2:
- waiting to lock monitor 0x02a5e624 (object 0x26a94f48, a java.lang.String),
  which is held by 'Test Thread 1'

Test Thread 1:
- waiting to lock monitor 0x02a5e75c (object 0x26a94f70, a java.lang.String),
  which is held by 'Test Thread 2'

Java stack information for the threads listed above:

Test Thread 2:
  at test.LockTest.acquireLock(LockTest.java:36)
  - waiting to lock 0x26a94f48 (a java.lang.String)
  - locked 0x26a94f70 (a java.lang.String)
  at test.LockTest.run(LockTest.java:19)
  at java.lang.Thread.run(Thread.java:613)

Test Thread 1:
  at test.LockTest.lockTest(LockTest.java:27)
Java Thread Dumps - Stuck in a Service Call

"Portlet Runner Thread Pool Thread-1" daemon prio=5 tid=0x01263680 nid=0x109 runnable
[96ffe000..96fffc28]
  at java.net.SocketInputStream.socketRead0(Native Method)
  at java.net.SocketInputStream.read(SocketInputStream.java:129)
  at java.net.SocketInputStream.read(SocketInputStream.java:182)
  at java.io.FilterInputStream.read(FilterInputStream.java:66)
  at java.io.PushbackInputStream.read(PushbackInputStream.java:120)
  at org.apache.commons.httpclient.HttpParser.readRawLine(HttpParser.java:109)
  at org.apache.commons.httpclient(HttpParser.java:135)
  at org.apache.commons.httpclient.HttpConnection.readLine(HttpConnection.java:1086)
  at org.apache.commons.httpclient.HttpMethodBase.readLine(HttpMethodBase.java:2188)
  at org.apache.commons.httpclient.HttpMethodBase.readLine(HttpMethodBase.java:1949)
  at org.apache.commons.httpclient.HttpMethodBase.readLine(HttpMethodBase.java:1949)
  at org.apache.commons.httpclient.HttpMethodBase.readLine(HttpMethodBase.java:1093)
  at com.mycompany.utils.GZIPAwarePostMethod.execute(GZIPAwarePostMethod.java:81)
  at org.apache.commons.httpclient.HttpClient.executeMethod(HttpClient.java:675)
  at org.apache.commons.httpclient.HttpClient.executeMethod(HttpClient.java:529)

Then use Netstat or some other tool to find where the connection is being made to
Java Thread Dumps - Waiting is a Good Thing, But Beware of the Sleepers!

"Entitlement Thread Pool Thread-2" daemon prio=1 tid=0x00b08a28 nid=0xed in Object.wait()
   at java.lang.Object.wait(Native Method)
   - waiting on <0xbc0236a0> (a com.mycompany.utils.ThreadPoolImpl$PoolThread)
   at com.mycompany.utils.ThreadPoolImpl$PoolThread.run(ThreadPoolImpl.java:122)
   - locked <0xbc0236a0> (a com.mycompany.utils.ThreadPoolImpl$PoolThread)

"Cache (LookupCache) Maintenance Thread" daemon prio=5 tid=0x0045e820 nid=0xa9 waiting on condition [986ff000..986ffc28]
   at java.lang.Thread.sleep(Native Method)
   at com.mycompany.cache.JBossCache$MaintenanceThread.run(JBossCache.java:120)
Java Thread Dumps - Not dependable when JVM in GC Panic

"ExecuteThread: '2' for queue: 'weblogic.admin.RMI'" daemon prio=5 tid=0x00200038 nid=0x49 runnable [9db7d000..9db7fc28]

    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:92)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:136)
    at com.sybase.jdbc2.tds.PduOutputFormatter.doFlush(PduOutputFormatter.java:212)
    at com.sybase.jdbc2.tds.PduOutputFormatter.flush(PduOutputFormatter.java:155)
    at java.io.FilterOutputStream.flush(FilterOutputStream.java:123)
    at com.sybase.jdbc2.tds.Tds.language(Tds.java:698)
    at com.sybase.jdbc2.jdbc.SybStatement.sendQuery(SybStatement.java:1451)
    at com.sybase.jdbc2.jdbc.SybPreparedStatement.sendQuery(SybPreparedStatement.java:1158)
    at com.sybase.jdbc2.jdbc.SybStatement.executeUpdate(SybStatement.java:1621)
    at com.sybase.jdbc2.jdbc.SybPreparedStatement.executeUpdate(SybPreparedStatement.java:91)
Remote Debugging - Inspection of all Variables Possible
Remote Debugging - Don’t Need Source to Set a Breakpoint
Remote Debugging - Tools and Learnings

Need to enable remote debugging at the command line:

```
java -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7010
```

Most modern IDEs have remote debugging capabilities built in:

- [http://www.jetbrains.com/idea/docs/help/project/rundebugremote.html](http://www.jetbrains.com/idea/docs/help/project/rundebugremote.html)

Fairly safe to use in disparate environments - WHEN NEEDED!

Targetted use of breakpoints can home in on race condition and deadlock errors.
Fallacies: Post hoc ergo propter hoc -- Correlation is not causation

Judge me by my size, do you?

Master Yoda (Star Wars)
Lots of Stuff Not Covered

Java Scripting - BeanShell, Java 6 scripting - Open the JVM so that you can connect to it and inspect anything!

OS level tracing: prstat, pstack, pmap, truss, netstat

Log Scraping

Decompilation - JAD - injection of decompiled classes into an app

http://www.kpdus.com/jad.html

Proxies - Monitor the communication to and from an app

http://ws.apache.org/commons/tcpmon/