Memory Management for Android Apps
Patrick Dubroy (dubroy.com · @dubroy)
May 11, 2011
Software

Work expands to fill the time available.
Overview

• Changes in Gingerbread and Honeycomb
  – heap size
  – GC
  – bitmaps
• Understanding heap usage
  – logs
  – memory leaks
  – Eclipse Memory Analyzer (MAT)
Expectations

• Android
• Dalvik heap
• Garbage collection
• OutOfMemoryError
Heap Size

• Heap size limits
  – G1: 16MB
  – Droid: 24MB
  – Nexus One: 32MB
  – Xoom: 48MB

• ActivityManager.getMemoryClass()
Large Heaps

• Honeycomb adds “largeHeap” option in AndroidManifest.xml:
  – **Degrades performance!** Use only if you understand why you need it.

```
<application
    android:name="com.example.foobar"
    android:largeHeap="true"
...
</application>
```

ActivityManager.getLargeMemoryClass()
Garbage Collection
Garbage Collection

GC Roots
Garbage Collection

GC Roots

Diagram showing the relationship between GC Roots and other objects in the memory system.
Garbage Collection
Garbage Collection

• Bigger heaps = longer pauses?
• Pre-Gingerbread GC:
  – Stop-the-world
  – Full heap collection
  – Pause times often > 100ms
• Gingerbread and beyond:
  – Concurrent (mostly)
  – Partial collections
  – Pause times usually < 5ms
Bitmaps

Old way (pre-Honeycomb):
— freed via recycle() or finalizer
— hard to debug
— full, stop-the-world GCs
Bitmaps

Old way (pre-Honeycomb):
— freed via recycle() or finalizer
— hard to debug
— full, stop-the-world GCs

New way:
— freed synchronously by GC
— easier to debug
— concurrent & partial GCs
Overview

• Changes in Gingerbread and Honeycomb
  – heap size
  – GC
  – bitmaps
• Understanding heap usage
  – logs
  – memory leaks
  – Eclipse Memory Analyzer (MAT)
Overview

• Changes in Gingerbread and Honeycomb
  – heap size
  – GC
  – bitmaps

• Understanding heap usage
  – logs
  – memory leaks
  – Eclipse Memory Analyzer (MAT)
Interpreting Log Messages

D/dalvikvm(9050): GC_CONCURRENT freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms
Interpreting Log Messages

D/dalvikvm(9050): **GC_CONCURRENT** freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms

• Reason for GC
  – GC_CONCURRENT
  – GC_FOR_MALLOC
  – GC_EXTERNAL_ALLOC
  – GC_HPROF_DUMP_HEAP
  – GC_EXPLICIT
Interpreting Log Messages

D/dalvikvm( 9050): GC_CONCURRENT freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms

• Reason for GC
• Amount freed
Interpreting Log Messages

D/dalvikvm( 9050): GC_CONCURRENT freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms

• Reason for GC
• Amount freed
• Heap statistics
**Interpreting Log Messages**

<table>
<thead>
<tr>
<th>Reason for GC</th>
<th>Amount freed</th>
<th>Heap statistics</th>
<th>External memory statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• D/dalvikvm( 9050): GC_CONCURRENT freed 2049K, 65% free 3571K/9991K, <strong>external 4703K/5261K</strong>, paused 2ms+2ms</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

• Reason for GC
• Amount freed
• Heap statistics
• External memory statistics
Interpreting Log Messages

D/dalvikvm( 9050): GC_CONCURRENT freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms

• Reason for GC
• Amount freed
• Heap statistics
• External memory statistics
• Pause time
Heap Dumps

• Binary dump of all objects
• Create with:
  – DDMS
  – android.os.Debug.dumpHprofData()
• Convert to standard HPROF format:
  hprof-conv orig.hprof converted.hprof
• Analyze with MAT, jhat, etc.
Memory Leaks

• GC does not prevent leaks!
• Leak: ref to an unused object preventing GC
• References to Activity (Context)
  – View, Drawable, ...
Memory Leaks
Eclipse Memory Analyzer (MAT)

• Download from http://eclipse.org/mat/
• “Shallow heap” and “retained heap”
Eclipse Memory Analyzer (MAT)

• Download from http://eclipse.org/mat/
• “Shallow heap” and “retained heap”
Eclipse Memory Analyzer (MAT)

- Download from [http://eclipse.org/mat/](http://eclipse.org/mat/)
- “Shallow heap” and “retained heap”
Eclipse Memory Analyzer (MAT)

• Download from [http://eclipse.org/mat/](http://eclipse.org/mat/)
• “Shallow heap” and “retained heap”
Dominator Tree

• Dominator: closest object on every path to node
Demo: Debugging a memory leak with MAT
public class MainActivity extends Activity implements ActionBar.TabListener {

    static Leaky leak = null;

    class Leaky {
        void doSomething() {
            System.out.println("Wheee!!!");
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (leak == null) {
            leak = new Leaky();
        }
    ...
}
public class MainActivity extends Activity implements ActionBar.TabListener {

    static Leaky leak = null;

    class Leaky {
        void doSomething() {
            System.out.println("Wheee!!!");
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (leak == null) {
            leak = new Leaky();
        }

        ...
    }
}
public class MainActivity extends Activity implements ActionBar.TabListener {

    static Leaky leak = null;

    class Leaky {
        void doSomething() {
            System.out.println("Whee!!!");
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (leak == null) {
            leak = new Leaky();
        }
        ...
    }
}
public class MainActivity extends Activity implements ActionBar.TabListener {

    static Leaky leak = null;

    class Leaky {
        void doSomething() {
            System.out.println("Wheee!!!");
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (leak == null) {
            leak = new Leaky();
        }

        ...
    }
}
Demo: Debugging a memory leak with MAT
Memory Leaks

• References to Activity, Context, View, Drawable, ...
• Non-static inner classes (e.g. Runnable)
• Caches
Links

• Articles on Android Developers Blog
  – Memory Analysis for Android Applications
  – Avoiding Memory Leaks by Romain Guy

• Eclipse Memory Analyzer: http://www.eclipse.org/mat/

• Markus Kohler’s Java Performance Blog: http://kohlerm.blogspot.com/

• Feedback on this talk:  
  http://speakermeter.com/talks/memory-management-android