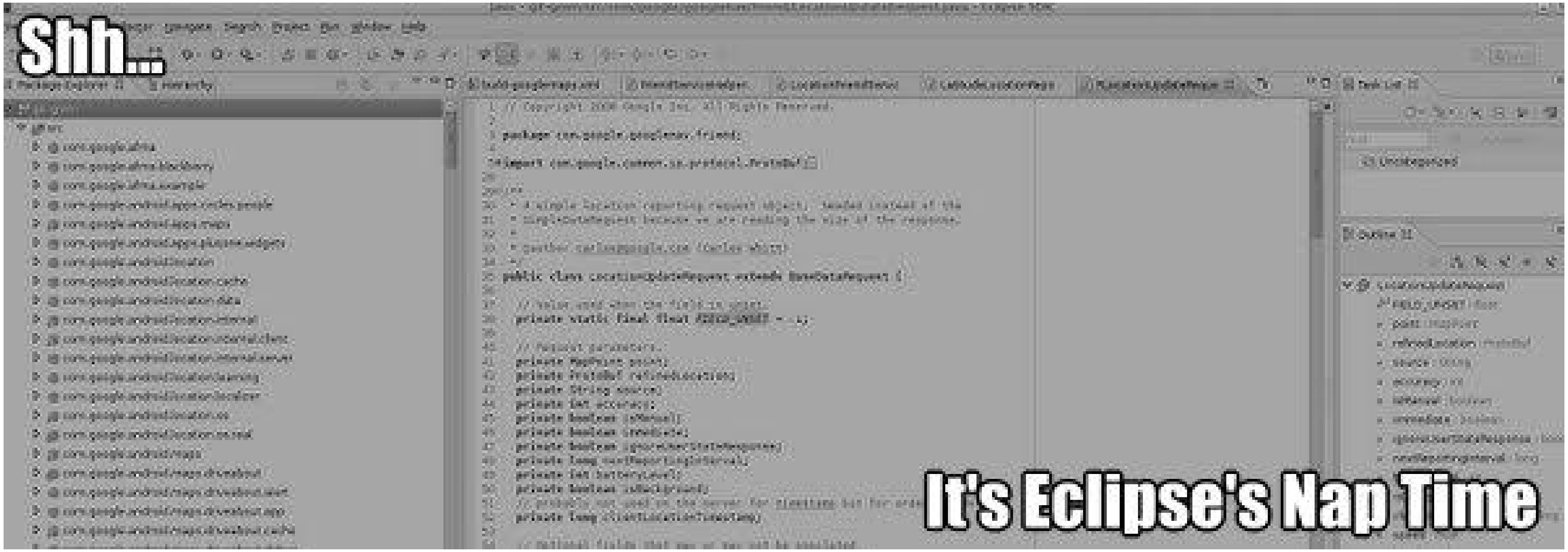
Tuning Eclipse for Better Performance

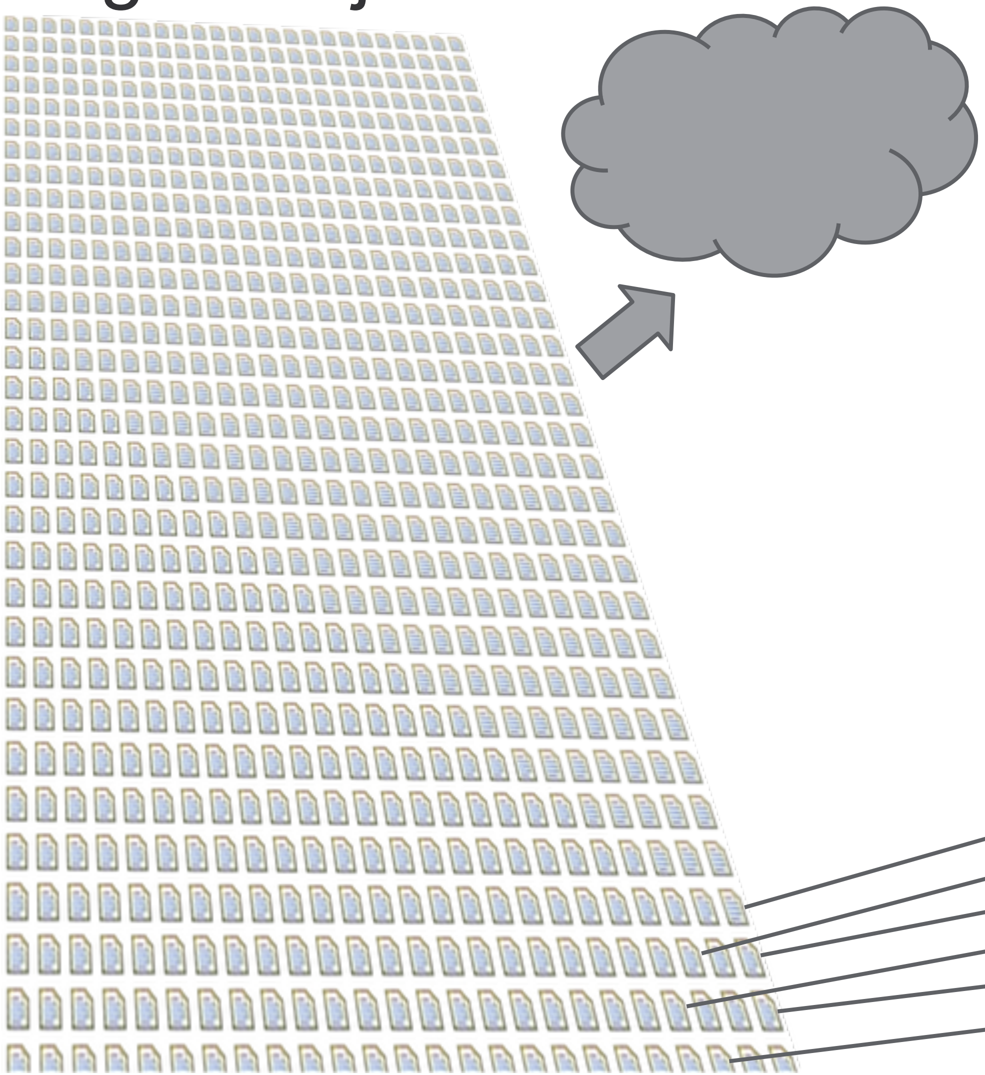
What is Performance?

UI Responsiveness!

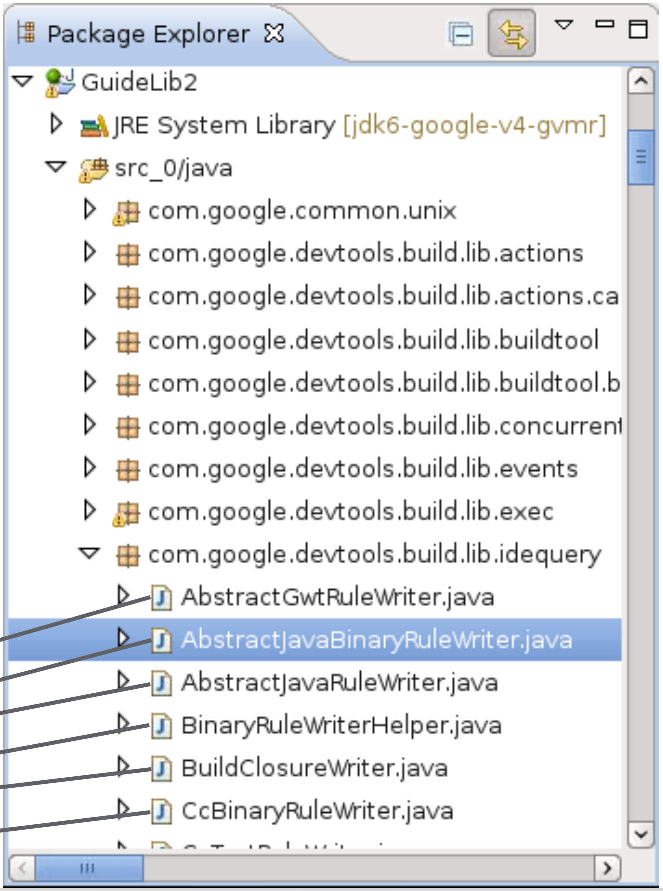


```
// Copyright 2008 Google Inc. All Rights Reserved.

package com.google.googlenav.friend;

import com.google.common.io.protocol.ProtoBuf;

/**
 * A simple location reporting request object. Needed instead of the
 * SimpleDataRequest because we are reading the size of the response.
 *
 * @author [illegible] ([illegible])
 */
public class LocationUpdateRequest extends BaseDataRequest {

  // Value used when the field is unset.
  private static final float FIELD_UNSET = -1;

  // Request parameters.
  private MapPoint point;
  private ProtoBuf refinedLocation;
  private String source;
  private int accuracy;
  private boolean isManual;
  private boolean immediate;
  private boolean ignoreUserStateResponse;
  private long nextReportingInterval;
  private int batteryLevel;
  private boolean isBackground;
  // probably not used on the server for [illegible] but for [illegible]
  private long clientLocationTimestamp;

  // Optional fields that may or may not be populated
```

Google Project Structure



.classpath
(8k archives)



Package Explorer

GuideLib2
- JRE System Library [jdk6-google-v4-gvmr]
- src_0/java
  - com.google.common.unix
  - com.google.devtools.build.lib.actions
  - com.google.devtools.build.lib.actions.ca
  - com.google.devtools.build.lib.buildtool
  - com.google.devtools.build.lib.buildtool.b
  - com.google.devtools.build.lib.concurrent
  - com.google.devtools.build.lib.events
  - com.google.devtools.build.lib.exec
  - com.google.devtools.build.lib.idequery
    - AbstractGwtRuleWriter.java
    - AbstractJavaBinaryRuleWriter.java
    - AbstractJavaRuleWriter.java
    - BinaryRuleWriterHelper.java
    - BuildClosureWriter.java
    - CcBinaryRuleWriter.java

How to Diagnose?

1. User reports/stack traces

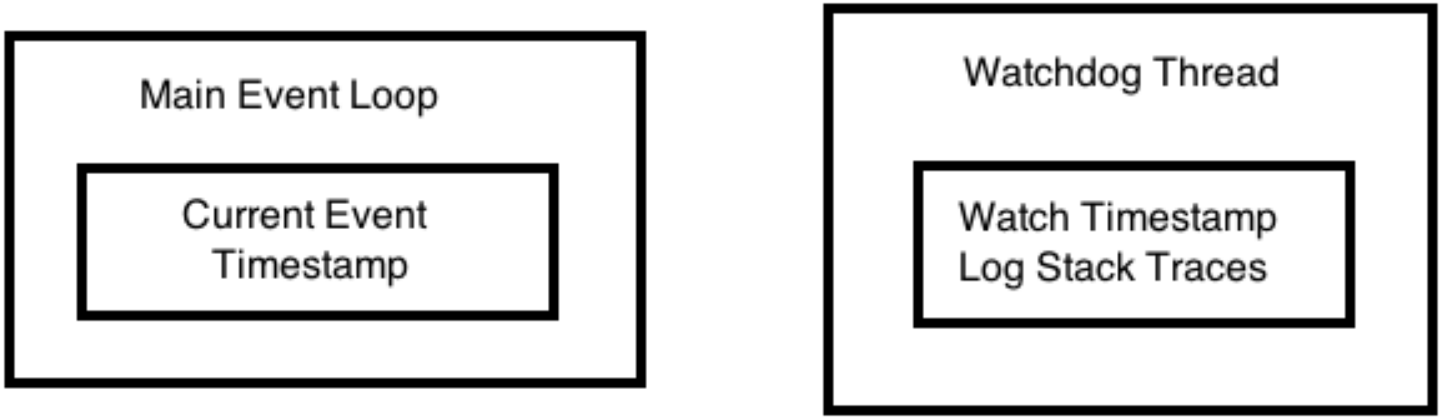
2. Profile using Eclipse debug options:

```
org.eclipse.jdt.core/debug=true
org.eclipse.jdt.core/debug/javamodel/cache=true
```

3. Collect massive amounts of data!

Monitoring API

Logging of blockages



Main Event Loop

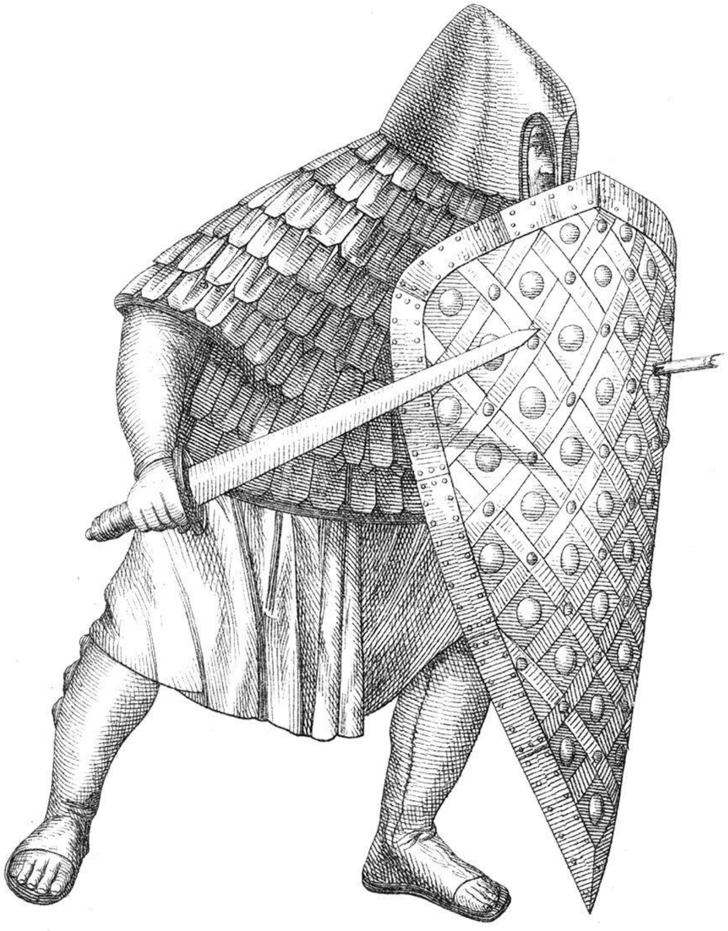
Current Event Timestamp

Watchdog Thread

Watch Timestamp
Log Stack Traces

https://bugs.eclipse.org/360052

War Stories



{{PD-old-70}}

Simple mistakes

- font handle leak

Memory tuning

- cache sizes

Unneeded work

- editor instantiation

Expensive/Sensitive APIs

- Pydev markers
- Launch config validation

Simple Mistakes

A plug-in leaked handles
    (1 each time a timer fired)

After 3 days, FindReplace dialog took 10+ seconds

Details: https://bugs.eclipse.org/333286
- SWT is JDK 1.1 compliant, no HashSets
- O(N^2) lookup using Array
- In 3.7, switched it to O(N) algorithm, effectively O(1)
- BUT we also fixed our leaked handles

Lesson: Be an^L^L^L^L^LEmphasize correctness!

Memory Tuning

JavaModelCache: in-memory caches
LRU caches (if undersized, flushed by classpath walk)
Defaults to percentage of memory given to Eclipse

To adjust using Java system properties:

```
-Dorg.eclipse.jdt.core.javamodelcache.ratio=1.5
-Dorg.eclipse.jdt.core.javamodelcache.jartyperatio=3.0
```

To view tracing information on caches:

```
org.eclipse.jdt.core/debug=true
org.eclipse.jdt.core/debug/javamodel/cache=true
```

http://bugs/eclipse.org/365499 allowed finer control of JarTypeCache

Unnecessary Work

Checkstyle plug-in extends IStartup

1. Creates a registry of IFiles retrieved from editor parts
2. Runs a job to execute the CheckStyle Builder

Forces instantiation of all editor tabs at Eclipse startup!

(median 8.6 second delay)

Work is only used if CheckStyle Builder is enabled

AND the Builder works fine even when IStartup is disabled!

Expensive and Sensitive APIs

PyDev does Marker manipulation on the UI thread

GPE did classpath access in Launcher validation callback

My team used JavaCore.create() in a PropertyTester
(should have used IAdaptable)

Lessons:

- Learn what executes on the UI thread
- Be hyper-sensitive about I/O and expensive API access

Tuning Eclipse

1. Local disk for all metadata (SSDs even better?)
2. Memory tuning (including Eclipse's internal caches)
3. IWYU (Install What You Use)
4. In your code, use APIs correctly (look at Platform code for examples)
5. Disable auto-triggering of potentially expensive operations (autocompletion, hyperlink detection)