Chronicler: Lightweight Recording to Reproduce Field Failures

Jonathan Bell, Nikhil Sarda and Gail Kaiser
Columbia University, New York, NY USA
What happens when software crashes?
A problem has been detected and Windows has been shut down to prevent damage to your computer.

**DRIVER_IRQL_NOT_LESS_OR_EQUAL**

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup options, and then select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C, 0x00000002, 0x00000000, 0xF86B5A89)

** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further assistance.
You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。
Your computer was restarted because of a problem.

This report will be sent to Apple automatically.

Comments

Problem Details and System Configuration

Interval Since Last Panic Report: 10078 sec
Pancics Since Last Report: 4
Anonymous UUID:

Wed Aug 15 11:34:03 2012

Backtrace (CPU 5), Frame : Return Address
0xffffffff80c50abb0 : 0xffffffff800e21d5f6
0xffffffff80c50abc20 : 0xffffffff7f90000f1f
0xffffffff80c50abc30 : 0xffffffff800e5f9fae
0xffffffff80c50abc90 : 0xffffffff800e5f809d
0xffffffff80c50abc0 : 0xffffffff800e5f67d
0xffffffff80c50abd40 : 0xffffffff800e5f694d
0xffffffff80c50abd0 : 0xffffffff800e607c33
We're Sorry

Firefox had a problem and crashed. We'll try to restore your tabs and windows when it restarts.

To help us diagnose and fix the problem, you can send us a crash report.

- [ ] Tell Mozilla about this crash so they can fix it

Details...

Add a comment (comments are publicly visible)

- [ ] Include the address of the page I was on
- [ ] Allow Mozilla to contact me about this report

Enter your email address here

Your crash report will be submitted before you quit or restart.

Restart Firefox  Quit Firefox
<table>
<thead>
<tr>
<th>Frame</th>
<th>Module</th>
<th>Signature</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XUL</td>
<td>js::MaybeGC</td>
<td>js/src/jsgc.cpp:2654</td>
</tr>
<tr>
<td>1</td>
<td>XUL</td>
<td>nsJSContext::ScriptEvaluated</td>
<td>dom/base/nsJSEnvironment.cpp:2806</td>
</tr>
<tr>
<td>2</td>
<td>XUL</td>
<td>nsCxPusher::nsCxPusher</td>
<td>content/base/src/nsContentUtils.cpp:3127</td>
</tr>
<tr>
<td>3</td>
<td>XUL</td>
<td>nsGlobalWindow::SetNewDocument</td>
<td>dom/base/nsGlobalWindow.cpp:2150</td>
</tr>
<tr>
<td>4</td>
<td>XUL</td>
<td>DocumentViewerImpl::InitInternal</td>
<td>layout/base/nsDocumentViewer.cpp:926</td>
</tr>
<tr>
<td>5</td>
<td>XUL</td>
<td>DocumentViewerImpl::Init</td>
<td>layout/base/nsDocumentViewer.cpp:676</td>
</tr>
<tr>
<td>6</td>
<td>XUL</td>
<td>nsDocShell::SetupNewViewer</td>
<td>docshell/base/nsDocShell.cpp:8023</td>
</tr>
<tr>
<td>7</td>
<td>XUL</td>
<td>nsDocShell::Embed</td>
<td>docshell/base/nsDocShell.cpp:6075</td>
</tr>
<tr>
<td>8</td>
<td>XUL</td>
<td>nsDocShell::CreateAboutBlankContentViewer</td>
<td>docshell/base/nsDocShell.cpp:6814</td>
</tr>
<tr>
<td>9</td>
<td>XUL</td>
<td>nsDocShell::EnsureContentViewer</td>
<td>docshell/base/nsDocShell.cpp:6697</td>
</tr>
<tr>
<td>10</td>
<td>XUL</td>
<td>nsDocShell::GetInterface</td>
<td>docshell/base/nsDocShell.cpp:951</td>
</tr>
<tr>
<td>11</td>
<td>XUL</td>
<td>nsGetInterface::operator</td>
<td>obj-firefox/x86_64/xpcom/build/nsIInterfaceRequestorUtils.cpp:19</td>
</tr>
<tr>
<td>12</td>
<td>XUL</td>
<td>nsCOMPtr_base::assign_from_helper</td>
<td>obj-firefox/x86_64/xpcom/build/nsCOMPtr.cpp:110</td>
</tr>
<tr>
<td>13</td>
<td>XUL</td>
<td>nsGlobalWindow::GetDocument</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>XUL</td>
<td>XUL@0x673d6d</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>XUL</td>
<td>XPCConvert::NativeInterface2JSONObject</td>
<td>js/xpconnect/src/XPCConvert.cpp:837</td>
</tr>
<tr>
<td>16</td>
<td>XUL</td>
<td>XPCConvert::NativeData2JS</td>
<td>js/xpconnect/src/XPCConvert.cpp:319</td>
</tr>
<tr>
<td>17</td>
<td>XUL</td>
<td>XPCWrappedNative::CallMethod</td>
<td>js/xpconnect/src/xpcprivate.h:3312</td>
</tr>
<tr>
<td>18</td>
<td>XUL</td>
<td>XPC_WN_GetterSetter</td>
<td>js/xpconnect/src/xpcprivate.h:2823</td>
</tr>
<tr>
<td>19</td>
<td>XUL</td>
<td>js::InvokeKernel</td>
<td>js/src/jsctxtinlines.h:372</td>
</tr>
<tr>
<td>20</td>
<td>XUL</td>
<td>js::Invoke</td>
<td>js/src/jsinterp.h:119</td>
</tr>
<tr>
<td>21</td>
<td>XUL</td>
<td>js::InvokeGetterOrSetter</td>
<td>js/src/jsinterp.cpp:469</td>
</tr>
<tr>
<td>22</td>
<td>XUL</td>
<td>js::Shape::get</td>
<td>js/src/jsscopeinlines.h:296</td>
</tr>
</tbody>
</table>
Stack Traces Aren’t Enough

Method 1

Stack Trace
Method 1
Stack Traces Aren’t Enough

Method 1

Method 2

Writes

Stack Trace
Method 1
Method 2

@jon_bell
Stack Traces Aren’t Enough

Method 1

Method 2

Writes

Stack Trace

Method 1
Stack Traces Aren’t Enough

Method 1

Writes

Method 2

Method 3

Stack Trace
Method 1
Method 3

@_jon_bell_

5/23/13
Stack Traces Aren’t Enough

Method 1

Method 2

Method 3

Stack Trace
Method 1
Method 3

*Crashes*
Potential Solutions

- Frequent Snapshots [Artzi et al]
- Guided symbolic execution [Cheung et al], [Crameri et al], [Jin and Orso], [Zamfir and Candea]
Recording External Inputs

[Geels et al], [Clause and Orso], [Mickens et al], this work
Recording External Inputs

- Other applications on the Network
- Files on the user's disk
- Inputs from the user (e.g. keyboard)

Our application
Recording External Inputs

Records all external interaction

Other applications on the Network

Files on the user's disk

Recording System

Our application

Inputs from the user (e.g. keyboard)
Record and Replay: Workflow

Application

Instrumented for log

R+R System

Instrumented for replay

Used in the field

R+R system generates test case

Bug successfully reproduced in the lab

Crashes
Previous work looks at logging system calls
- e.g. read, write, open
- Can lead to very high overhead in VM-based languages, which perform many such operations just to setup the VM!
Our insight: API Level Logging
Reading a File: Log System Calls

```java
public class ReadFile {
    public static void main(String[] args) throws FileNotFoundException {
        String text = new Scanner(new File("testFile")).useDelimiter("\A").next();
    }
}
```

$ java ReadFile

684 System Calls

@_jon_bell_
Reading a File: API Logging

```java
public class ReadFile {
    public static void main(String[] args) throws FileNotFoundException {
        String text = new Scanner(new File("testFile")).useDelimiter("\A").next();
    }
}
```

$ java ReadFile

I “External” API call
API Level Logging: The Challenge

Language VM (.NET CLR, JVM, etc)

Application

Language API

Normal API

External-input API

Chronicler

Outside world (sources of external input)

@__jon_bell__
API Level Logging: The Challenge

Language VM (.NET CLR, JVM, etc)

Application

Language API

Normal API

External-input API

Outside world (sources of external input)

Must determine reliably
Identifying External Inputs

Language VM (.NET CLR, JVM, etc)

"Native" interface

Outside world (sources of external input)
External Inputs in Java

Language VM (.NET CLR, JVM, etc)

Language API

Native Methods in API

Outside world (sources of external input)
External Inputs in Java

Language VM (.NET CLR, JVM, etc)

Language API

API Calling Native Methods

Native Methods in API

Outside world (sources of external input)
Problem: A lot is done outside of the VM
Log explosion

Not all native methods return external input

```java
public static native void arraycopy ...
```

1984 direct callers
Log explosion: System.arraycopy

```java
void getChars(char dst[], int dstBegin) {
    System.arraycopy(...);
}

public String concat(String str) {
    ...
    getChars(0, count, buf, 0);
    ...
}

125,714 indirect callers
```
Managing Log Explosion

Language VM (.NET CLR, JVM, etc)
Language API

API calling external input methods

Native methods in API receiving external input

Outside world (external input)
Outside world (not input-providing)
Managing Log Explosion

- Stop list: approx 70 native methods that do NOT gather external input
- Also includes StrictMath, some image processing
- Not necessary to be complete in the stop list
Managing Log Explosion

• Stop list: approx 70 native methods that do NOT gather external input

• Also includes StrictMath, some image processing

• Not necessary to be complete in the stop list

• Result: decrease log sites by ~25%
Implementation

• Targets Java
• Bytecode transformation (no source!)
• Logs all external inputs from the API
• Plenty of clever tricks: details in the paper (fast logging, callbacks, finalizers, and more!)
Evaluation
Functionality Evaluation

Find bug in bug tracker

Instrument with Chronicler

Chronicler Reproduces it Again

Manually Reproduce Bug
Functionality Evaluation

• Attempted to reproduce 11 real bugs in:
  • Jetty
  • Apache Commons-Math
  • Apache Commons-Lang
  • Groovy
• Succeeded in all cases
Performance: DaCapo

DaCapo Absolute Performance

<5% overhead (6)
Performance: DaCapo

DaCapo Absolute Performance

<10% overhead (4)
Performance: DaCapo

DaCapo Absolute Performance

<30% overhead (2)
Performance: DaCapo

DaCapo Absolute Performance

<40% overhead (2; nearly worst case)

Baseline
Chronicler
Performance: DaCapo

DaCapo Absolute Performance

Baseline vs Chronicler

avrora  batik  eclipse  fop  h2  jython  luindex  lusearch  pmd  sunflow  tomcat  tradebeans  tradesoap  xalan

Average benchmark time (ms)

0  5000  10000  15000  20000  25000  30000  35000  40000
Performance: Best Case

SciMark Absolute Performance

- Composite: 0.23%
- FFT: 1.15%
- SOR: 0.29%
- Monte Carlo: 0.19%
- MatMult: 0.27%
- LU: 0%

Performance (Megafllops)
Performance: Worst Case

I/O Microbenchmark Relative Performance

Worst case performance

Logging overhead masked by JVM setup/teardown

Input File Size (MB)
Limitations and Future Work
Limitations and Future Work

- Assumes VM correctness
- Thread Interleavings
  - Employ developer-side race detectors [e.g. Naik]
  - Integrate with existing thread access loggers [e.g. Huang]
- Privacy
  - Symbolic execution on the client to generate new inputs [e.g. Castro, Clause]
Chronicler: Lightweight Recording to Reproduce Field Failures

Jonathan Bell, Nikhil Sarda and Gail Kaiser
Columbia University, New York, NY USA

https://github.com/Programming-Systems-Lab/chroniclerj

jbell@cs.columbia.edu

@_jon_bell_