EventRacer: Finding Concurrency Errors in Event-Driven Applications

Pavol Bielik
Android Errors Caused by Concurrency

Display article twice

Display wrong directory

Display wrong order
Web Page Errors Caused by Concurrency

- Incomplete form submitted
- jQuery version used non-deterministically
- Non-operational menu
Event-Driven Applications

designed to hide latency, various asynchronous APIs
    network, disk, database, timers, UI events

highly asynchronous and complex control flow
    scheduling non-determinism

asynchrony is not intuitive
Trouble with Asynchrony

- Background task, progress dialog, orientation change - is there any 100% working solution?
- JavaScript function sometimes called, sometimes not
- Avoiding race conditions in Google Analytics asynchronous tracking
- Ajax Call Sometimes Works, Sometime works and refreshes, Sometimes refreshes and fails ...?
- Is AsyncTask really conceptually flawed or am I just missing something?
“Hello World” of web page concurrency

```html
<html><body>
<script>
var v=undefined;
</script>

<img src="img1.png" onload="v='Hi!';">  
<img src="img2.png" onload="alert(v);"> 

</body></html>
```
Bad interleaving

```html
<html><body>
<script>
var v=undefined;
</script>

<img src="img1.png" onload="v='Hi!';">
<img src="img2.png" onload="alert(v);">
</body></html>
```
Understanding the problem

```html
<html><body>
<script>
var v=undefined;
</script>

<img src="img1.png" onload="v='Hi!';">
<img src="img2.png" onload="alert(v);">

</body></html>
```
Understanding the problem

```
<html><body>
<script>
    var v=undefined;
</script>

<img src="img1.png" onload="v='Hi!';">

<img src="img2.png" onload="alert(v);">
</body></html>
```
Understanding the problem

```html
<html><body>
<script>
  var v=undefined;
</script>

<img src="img1.png" onload="v='Hi!';">
<script>
  alert(v);
</script>
</body></html>

Event Actions

Happens-before

Race

write v

read v

6
Online Analysis

http://www.eventracer.org

Memory Locations

Search by name:  

Shown memory locations:  

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Num. races</th>
<th>Num. uncovered races</th>
<th>Race classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS Variable</td>
<td>JSActivation[138074].c</td>
<td>2</td>
<td>1</td>
<td>readyStateChange race</td>
</tr>
<tr>
<td>JS Variable</td>
<td>Function[29].bind</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>Window[26].gapi</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>Window[26].__jsl</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>[0x76f7ec7700].message</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>Object[106535].ker</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>Object[40173].pnone</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>JSActivation[107403].lp</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>Object[34661].isframeResized</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JS Variable</td>
<td>[0x76f7ec7700].DOMContentLoaded</td>
<td>1</td>
<td>1</td>
<td>NO_EVENT_ATTACHED</td>
</tr>
<tr>
<td>JS Variable</td>
<td>document[0x76f7ec36400].DOMContentLoaded</td>
<td>3</td>
<td>1</td>
<td>ONLY_LOCAL_WRITE</td>
</tr>
<tr>
<td>DOM Attribute</td>
<td>DOMNode[0x76f7ebbe626].cookie</td>
<td>2</td>
<td>1</td>
<td>MAYBE_LAZY_INIT COOKIE_OR_CLASSNAME</td>
</tr>
<tr>
<td>JS Array</td>
<td>Array[25294][SLFN]</td>
<td>1</td>
<td>1</td>
<td>ONLY_LOCAL_WRITE</td>
</tr>
<tr>
<td>JS Variable</td>
<td>JSActivation[27873].g</td>
<td>7</td>
<td>1</td>
<td>MAYBE_LAZY_INIT initialize</td>
</tr>
</tbody>
</table>

14 rows
EventRacer end-to-end System

Android App, Web Page → Instrumented System → Execution Trace → Happens-before Graph → Race Detector → Race Filtering and Grouping → Race Explorer
What are the **memory locations** on which asynchronous events can race?

**JS variables, functions, arrays**

```javascript
v='Hi!';  // write(v)
function f() {}  // write(f)
messages[2] = 42;  // write(messages[2])
```

**DOM nodes and attributes**

```html
<img id="img1" src="img1.png" onlload="v='Hi!';">  // write(#img1)
```

```javascript
document.getElementById("img1").addEventListener("click", f);  // write(#img1.click)
```
EventRacer end-to-end System

What are the atomic events used in event-driven applications?

Web
- parsing an HTML element
- executing a script
- handling user input
- ...

```html
<script>
 var v=undefined;
</script>

```
What is the event **happens-before**?

**Web**
- setInterval, setTimeout, AJAX, ...

**Android**
- postDelayed, postAtFront, postIdle, ...

```html
<script>
var v=undefined;
</script>
```

**Happens-before**

- `onload='v='Hi!';'`
- `onload='alert(v);'`
How to make **scalable race detection** in event-based setting?

(Naive algorithms have asymptotic complexity $O(N^3)$ and require $O(N^2)$ space)

<table>
<thead>
<tr>
<th>State of the art</th>
<th>EventRacer</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtime</td>
<td>TIMEOUT</td>
</tr>
<tr>
<td>memory</td>
<td>25181MB</td>
</tr>
</tbody>
</table>
EventRacer end-to-end System

Is the system effective at finding harmful races while reporting few benign races?

We filter common classes of benign races:
commutative operations, recycled objects, lazy initialization, local reads, ...

<table>
<thead>
<tr>
<th></th>
<th>Web</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td># races found</td>
<td>646</td>
<td>1328</td>
</tr>
<tr>
<td># races reported</td>
<td>17.3</td>
<td>13</td>
</tr>
<tr>
<td>reduction</td>
<td>37x</td>
<td>100x</td>
</tr>
</tbody>
</table>
Manual evaluation

Web (314 reports)
Fortune 100 Web Pages

- 24.6% Harmful bugs
  - unhandled exceptions
  - UI glitches
  - broken analytics
  - page needs refresh to work normally

- 58.4% synchronization races
  - various idioms:
    - if (ready) ...
    - try { ... } catch { retry }
    - array of callbacks
    - etc.

- 17% harmless races
  - commutative operations
  - benign races
  - framework related

Android (104 reports)
8 Play Store Applications

- 25% Harmful bugs
  - unhandled exceptions
  - UI glitches
  - broken analytics
  - page needs refresh to work normally

- 17.3% synchronization races
  - various idioms:
    - if (ready) ...
    - try { ... } catch { retry }
    - array of callbacks
    - etc.

- 57.7% harmless races
  - commutative operations
  - benign races
  - framework related

Web (314 reports)
Fortune 100 Web Pages

- 24.6% Harmful bugs
  - unhandled exceptions
  - UI glitches
  - broken analytics
  - page needs refresh to work normally

- 58.4% synchronization races
  - various idioms:
    - if (ready) ...
    - try { ... } catch { retry }
    - array of callbacks
    - etc.

- 17% harmless races
  - commutative operations
  - benign races
  - framework related

Android (104 reports)
8 Play Store Applications

- 25% Harmful bugs
  - unhandled exceptions
  - UI glitches
  - broken analytics
  - page needs refresh to work normally

- 17.3% synchronization races
  - various idioms:
    - if (ready) ...
    - try { ... } catch { retry }
    - array of callbacks
    - etc.

- 57.7% harmless races
  - commutative operations
  - benign races
  - framework related
protected void onCreate() {
    locationManager.requestLocationUpdates(GPS_PROVIDER, 0, 0, mListener);
    mDbHelper = new SQLiteOpenHelper(this, DB_NAME, DB_VERSION);
}

LocationListener mListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        //show location on map
        mDbHelper.getWritableDatabase().insert(loc);
    }
};

protected void onStop() {
    locationManager.removeUpdates(mListener);
    mDbHelper.close();
}
protected void onCreate() {
    locationManager.requestLocationUpdates(GPS_PROVIDER, 0, 0, mListener);
    mDbHelper = new SQLiteOpenHelper(this, DB_NAME, DB_VERSION);
}

LocationListener mListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // show location on map
        mDbHelper.getWritableDatabase().insert(loc);
    }
};

protected void onStop() {
    locationManager.removeUpdates(mListener);
    mDbHelper.close();
}

public void removeUpdates (LocationListener listener)

Added in API level 1

Removes all location updates for the specified LocationListener. Following this call, updates will no longer occur for this listener.
http://www.eventracer.org/android

A RACE DETECTOR FOR ANDROID

TRY ONLINE

Select Android application APK file for analysis

Find Races

Note: To perform a more thorough analysis of your application, please download the tool.
onLocationChanged and onStop are reported as not ordered

---

**event 1 source**
async IPC, interface(android.location.ILocationListener), code(onLocationChanged))

**event 2 source**
async IPC, interface(android.app.IApplicationThread), code(SCHEDULE_STOP_ACTIVITY))

→ **calling context**
Landroid/database/sqlite/SQLiteDatabase;.insert(...)

→ **calling context**
Landroid/database/sqlite/SQLiteClosable;.close(...)

→ UPDATE-UPDATE - 63568 Landroid/database/sqlite/SQLiteConnectionPool;.mAvailablePrimaryConnection
→ READ-UPDATE - 63568 Landroid/database/sqlite/SQLiteConnectionPool;.mIsOpen
→ READ-UPDATE - 63576 Landroid/database/sqlite/SQLiteConnection;.mConnectionPtr
→ READ-UPDATE - 63576 Landroid/database/sqlite/SQLiteConnection;.mPreparedStatementPool
Is the Alternative Interleaving Feasible?

D/GPS: onCreate
D/GPS: insert: Location[gps 47.284646,8.632389 acc=10 et=0 vel=2.0 mock]
D/GPS: insert: Location[gps 47.284656,8.632598 acc=10 et=0 vel=2.0 mock]
D/GPS: insert: Location[gps 47.284712,8.632722 acc=10 et=0 vel=2.0 mock]
D/GPS: insert: Location[gps 47.284832,8.632837 acc=10 et=0 vel=2.0 mock]
D/GPS: onStop
D/GPS: insert: Location[gps 47.285022,8.633205 acc=10 et=0 vel=2.0 mock]

E/AndroidRuntime: FATAL EXCEPTION: main
E/AndroidRuntime: Process: com.example.gps, PID: 2249
E/AndroidRuntime: java.lang.IllegalStateException: attempt to re-open an already-closed object: SQLiteDatabase: /data/data/com.example.gps/test.db
Current Directions

Google Chromium port
  V8 javascript engine instrumentation

Testing tools based on EventRacer
  Integration with Selenium
  PhantomJS

Application for Parallelization

Other Application Domains (beyond Web Pages, Android)
  Node.js
Instrumented System → Execution Trace → Happens-before Graph → Race Detector → Race Filtering and Grouping → Race Explorer

www.eventracer.org

Martin Vechev, Veselin Raychev, Pavol Bielik
Anders Møller, Casper Jensen
Manu Sridharan
Boris Petrov, Yasen Trifonov
Julian Dolby