Characterizing and Detecting Performance Bugs for Smartphone Applications

Yepang Liu
Dept. of Comp. Sci. and Engr.
The Hong Kong Univ. of Sci. and Tech.
Hong Kong, China
andrewust@cse.ust.hk

Chang Xu*
State Key Lab for Novel Soft. Tech.
Dept. of Comp. Sci. and Tech.
Nanjing University,
Nanjing, China
changxu@nju.edu.cn

Shing-Chi Cheung
Dept. of Comp. Sci. and Engr.
The Hong Kong Univ. of Sci. and Tech.
Hong Kong, China
scc@cse.ust.hk

* Corresponding author.

Yepang Liu (劉燁龐)

PhD Student, ACM & IEEE Student Member
Department of Computer Science and Engineering
The Hong Kong Univeristy of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Office: Room 3661 (RFID Lab, near Lift 31/32)
Email: andrewust@cse.ust.hk

http://www.cse.ust.hk/~andrewust/

XU, Chang (许畅)

Ph.D. (HKUST), CCF member, ACM member, IEEE member

Associate Professor with:

- State Key Laboratory for Novel Software Technology
- Department of Computer Science and Technology
- Institute of Computer Software

Mailing address:

- Building of Computer Science and Technology
- Nanjing University (Xianlin Campus)
- 163 Xianlin Avenue, Qixia District, Nanjing, Jiangsu, China (210023)

http://cs.nju.edu.cn/changxu/

S.C. CHEUNG (張成志)

Professor
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

http://www.cse.ust.hk/~scc/

Research Questions

RQ1 (Bug types and impacts):
What are common types of performance bugs in Android applications? What impacts do they have on user experience?

RQ2 (Bug manifestation):
How do performance bugs manifest themselves? Does their manifestation need special inputs?

RQ3 (Debugging and bug-fixing effort):
Are performance bugs more difficult to debug and fix than non-performance bugs? What information or tools can help with this?

RQ4 (Common bug patterns):
Are there common causes of performance bugs? Can we distill common bug patterns to facilitate performance analysis and bug detection?

RQ1: Bug Types and Impacts

GUI lagging. (53 / 70 = 75.7%)

- In Firefox browser, tab switching could take up to ten seconds in certain scenarios (Firefox bug 7194932)
- May trigger the infamous “Application Not Responding (ANR)” error

Energy leak. (10 / 70 = 14.3%)

- Zmanim (bug 50) My Tracks (bug 520)
- if an application drains battery quickly, users may switch to other applications that offer similar functionalities but are more energy-efficient.

Memory bloat. (8 / 70 = 11.4%)

- Can cause “Out of Memory (OOM)” errors and application crashes.
- Even mild memory bloat ——garbage collection would be frequently invoked, leading to degraded application performance.

RQ2: Bug Manifestation

Small-scale inputs suffice to manifest performance bugs.

Android applications can be susceptible to performance bugs.

Special user interactions needed to manifest performance bugs.

- Effectively testing performance bugs requires coverage criteria that explicitly consider sequences of user interactions in assessing the testing adequacy.
- Test input generation should construct effective user interaction sequences to systematically explore an application's state space.

Automated performance oracle needed.

Now:

- Human oracle.
- Product comparison.
- Developers' consensus.

Performance bugs can be platform-dependent.

RQ3: Debugging and Bug-fixing Effort

Quantify debugging and bug-fixing effort

- bug open duration
- number of bug comments
- patch size

Table 2. Performance bug debugging and fixing effort

Metric	Min.	Median	Max.	Mean
Bug open duration (days)	1	47	378	59.2
Number of bug comments	1	14	71	16.7
Patch size (LOC)	2	72	2,104	182.3

RQ3: Debugging and Bug-fixing Effort

Debugging and fixing performance bugs requires more effort

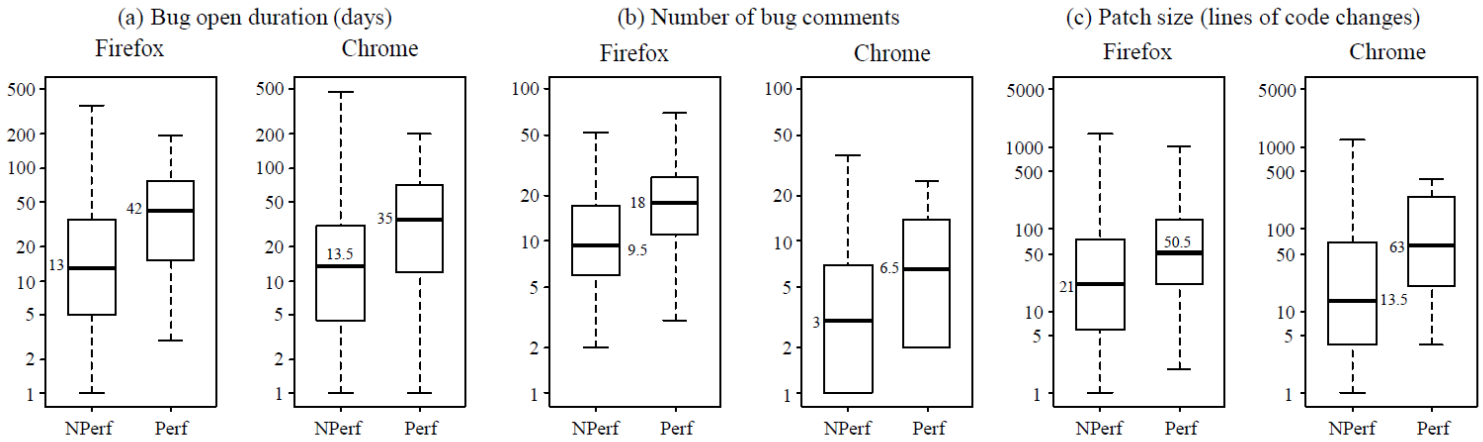

Figure 3. Comparison of debugging and bug-fixing effort ("Perf" = "performance bug", "NPerf" = "non-performance bug")

Mann-Whitney U-test

RQ3: Debugging and Bug-fixing Effort

Two kinds of useful tools

1. **Profiling tools.**

   Profiling tools (or profilers) monitor an application's execution, record its runtime information (e.g., execution time of a code unit), and trace details of its resource consumption (e.g., memory).

2. **Performance measurement tools.**

   Performance measurement can directly report performance for a selected code unit in an application.

RQ4: Common Bug Patterns

Lengthy operations in main threads.

```
1.     public void refreshThumbnails() {
2.       //generate a thumbnail for each browser tab
3.  -    Iterator<Tab> iter = tabs.values().iterator();
4.  -    while (iter.hasNext())
5.  -      GeckoApp.mAppContext.genThumbnailForTab(iter.next());
6.  +    GeckoAppShell.getHandler().post(new Runnable() {
7.  +      public void run() {
8.  +        Iterator<Tab> iter = tabs.values().iterator();
9.  +        while (iter.hasNext())
10. +          GeckoApp.mAppContext.genThumbnailForTab(iter.next());
11. +      }
12. +    });
13.    }
```

Note: the method genThumbnailForTab() compresses a bitmap to produce a thumbnail for a browser tab.

Figure 4. Firefox bug 721216 (simplified)

RQ4: Common Bug Patterns

Wasted computation for invisible GUI.

```
1.    public class ZmanimActivity extends Activity {
2.      private ZmanimLocationManager lm;
3.      private ZmanimLocationManager.Listener locListener;
4.      public void onCreate() {
5.        //get a reference to system location manager
6.        lm = new ZmanimLocationManager(ZmanimActivity.this);
7.        locListener = new ZmanimLocationManager.Listener() {
8.          public void onLocationChanged(ZmanimLocation newLoc) {
9.            //build UI using obtained location in a new thread
10.           rebuildUI(newLoc);
11.         }
12.       };
13.       //register location listener
14.       lm.requestLocationUpdates(GPS, 0, 0, locListener);
15.     }
16.     public void onResume() {
17. +     //register location listener if UI still needs update
18. +     if(buildingUINotFinished)
19. +       lm.requestLocationUpdates(GPS, 0, 0, locListener);
20.     }
21.     public void onPause() {
22. +     //unregister location listener
23. +     lm.removeListener(locListener);
24.     }
25.     public void onDestory() {
26. -     //unregister location listener
27. -     lm.removeListener(locListener);
28.     }
29.   }
```

Figure 5. Zmanim bug 50 (simplified)

RQ4: Common Bug Patterns

Frequently invoked heavy-weight callbacks.

```
    //inefficient version
 1. public View getView(int pos, View recycledView, ViewGroup parent) {
 2.   //list item layout inflation
 3.   View item = mInflater.inflate(R.layout.listItem, null);
 4.   //find inner views
 5.   TextView txtView = (TextView) item.findViewById(R.id.text);
 6.   ImageView imgView = (ImageView) item.findViewById(R.id.icon);
 7.   //update inner views
 8.   txtView.setText(DATA[pos]);
 9.   imgView.setImageBitmap((pos % 2) == 1 ? mIcon1 : mIcon2);
10.   return item;
11. }
```

RQ4: Common Bug Patterns

Frequently invoked heavy-weight callbacks.

```
12. //apply view holder pattern
13. public View getView(int pos, View recycledView, ViewGroup parent) {
14.   ViewHolder holder;
15.   if(recycledView == null) { //no recycled view to reuse
16.     //list item layout inflation
17.     recycledView = mInflater.inflate(R.layout.listItem, null);
18.     holder = new ViewHolder();
19.     //find inner views and cache their references
20.     holder.text = (TextView) recycledView.findViewById(R.id.text);
21.     holder.icon = (ImageView) recycledView.findViewById(R.id.icon);
22.     recycledView.setTag(holder);
23.   } else {
24.     //reuse the recycled view, retrieve the inner view references
25.     holder = (ViewHolder) recycledView.getTag();
26.   }
27.   //update inner view contents
28.   holder.text.setText(DATA[pos]);
29.   holder.icon.setImageBitmap((pos % 2) == 1 ? mIcon1 : mIcon2);
30.   return recycledView;
31. }

32. //view holder class for caching inner view references
33. public class ViewHolder {
34.   TextView text;
35.   ImageView icon;
36. }
```

Figure 7. View holder pattern

PerfChecker

Detecting lengthy operations in main threads.

1. **conducts a class hierarchy analysis to identify a set of check-points**

   These checkpoints include those lifecycle handlers defined in application component classes (e.g., those extending the Activity class) and GUI event handlers defined in GUI widget classes.

2. **constructs a call graph for each checkpoint, and traverses this graph to check whether the checkpoint transitively invokes any heavy APIs**

   heavy APIs, e.g., networking, database query, file IO, or other expensive APIs like those for Bitmap resizing.

PerfChecker

Detecting violation of the view holder pattern.

1. **conducts a class hierarchy analysis to identify a set of check-points**

   all getView() callbacks in concerned list views' adapter classes.

2. **constructs a program dependency graph for each checkpoint, and traverses this graph to check whether the following rule is violated:**

   list item layout inflation and inner view retrieval operations should be conditionally conducted based on whether there are reusable list items.

PerfChecker

Table 4. Subjects and the detected bug pattern instances in them

Application name	Application category	Revision no.	Size (LOC)	Downloads	Availability	Bug pattern instances		Bug ID(s)
						VH (69)	LM (57)	
Ushahidi	Communication	59fbb533d0	43.3K	10K ~ 50K	GitHub [48]	**9***	**2**	146, 159
c:geo	Entertainment	6e4a8d4ba8	37.7K	1M ~ 5M	GitHub [8]	0	**5**	3054
Omnidroid	Productivity	865	12.4K	1K ~ 5K	Google Code [38]	9	8	182, 183
Open GPS Tracker	Travel & Local	14ef48c15d	18.1K	100K ~ 500K	Google Code [39]	1	0	390
Geohash Droid	Entertainment	65bfe32755	7.0K	10K ~ 50K	Google Code [16]	0	1	48
Android Wifi Tether	Communication	570	9.2K	1M ~ 5M	Google Code [4]	1	3	1829, 1856
Osmand	Travel & Local	8a25c617b1	77.4K	500K ~ 1M	Google Code [40]	**18**	**17**	1977, 2025
My Tracks	Health & Fitness	e6b9c6652f	27.1K	10M ~ 50M	Google Code [34]	**2***	0	1327
WebSMS	Communication	1f596fbd29	7.9K	100K ~ 500K	Google Code [49]	0	**1**	801
XBMC Remote	Media & Video	594e4e5c98	53.3K	1M ~ 5M	Google Code [50]	1	0	714
ConnectBot	Communication	716cdaa484	33.7K	1M ~ 5M	Google Code [10]	0	6	658
Firefox	Communication	895a9905dd	122.9K	10M ~ 50M	Mozilla Repositories [33]	**1**	0	899416
APG	Communication	a6a371024b	98.2K	50K ~ 100K	Google Code [6]	4	8	140, 144
FBReaderJ	Books & References	0f02d4e923	103.4K	5M ~ 10M	GitHub [13]	**6***	6	148, 151
Bitcoin Wallet	Finance	12ca4c71ac	35.1K	100K ~ 500K	Google Code [7]	**4**	0	190
AnySoftKeyboard	Tools	04bf623ec1	26.0K	500K ~ 1M	GitHub [5]	**2***	0	190
OI File Manager	Productivity	f513b4d0b6	7.8K	5M ~ 10M	GitHub [37]	**1***	0	39
IMSDroid	Media & Video	553	21.9K	100K ~ 500K	Google Code [24]	10	0	457

1. "VH" means "Violation of the view Holder pattern", and "LM" means "Lengthy operations in Main threads".

2. Underlined bug pattern instances have been confirmed by developers as real performance issues, and "*" marked instances have been fixed by developers accordingly. For more details, readers can visit corresponding subject's source repositories and bug tracking systems by our provided links and bug IDs.

Related Work

Detecting performance bugs.

- Resource leaks(memory, sensors..)
- Heavy call or loop

Testing for application performance.

- Many performance bugs in smartphone applications need certain user interaction sequences to manifest.
- Smartphone applications also have some unique features, e.g., long GUI lagging can force an Android application to close.

Debugging and optimization for application performance.

- Estimates performance
- Uses profiling to log performance-related information

Understanding performance bugs.

- There is little evidence showing that fixing performance bugs has a high chance of introducing new bugs.

Conclusion and Future Work

We discussed several characteristics of performance bug

Performance bug detection tools are helpful to developers

Future work on improving PerfChecker

- More bug patterns to boost its detection capability
- Improve the effectiveness of bug detection algorithms

Paper:
http://sccpu2.cse.ust.hk/andrewust/files/ICSE2014.pdf

Talk Slide:
http://sccpu2.cse.ust.hk/andrewust/files/ICSE2014_presentation.pdf

For empirical study data and tool runnable, please visit:
http://sccpu2.cse.ust.hk/perfchecker/

Related Paper:
http://sccpu2.cse.ust.hk/andrewust/files/percom13.pdf