A Characterization of State Spill in Modern OSes

Kevin Boos  Emilio Del Vecchio  Lin Zhong

ECE Department, Rice University

EuroSys 2017
How do we deal with complexity?
Modularization

complex system
Modularization

1

2

3

4
Reducing complexity *should* make things easier…

- Process migration
- Fault isolation & fault tolerance
- Live update, hot-swapping, software virtualization
- Maintainability
- Security and more
Modularization is not enough!

Effects of interactions:

- Propagation of data and control
- Changes to the state of each entity

interactions have complex effects!

state spill
State spill in a nutshell

*a new term* to describe the phenomenon when:

A software entity’s state undergoes **lastling change** as a result of an interaction with another entity.
Outline of contributions

1. Define and identify state spill as a root cause of challenging problems in computing

2. Classify state spill examples collected from real OSes

3. Automate state spill detection with STATESPY

4. Results from Android system services
Definition of State Spill
State spill definition by example

**Source** (application)

```java
public void main() {
    int id = 5;
    byte cfg = 1;
    fn cb = handleCb;

    service.addCallback(id, cfg, cb);

    log("added cb!");
}

void handleCb() {
    // do something
}
```

**Destination** (system service)

```java
public class SystemService {
    static int sCount;
    byte mConfig;
    List<Callback> mCallbacks;

    int unrelated;

    public void addCallback(int id, byte cfg, Callback cb) {
        int b = id;
        Log.print("id=" + b);
        mConfig = cfg;
        mCallbacks.add(cb);
        sCount++;
    }
}
```

Before (empty)

During

After

Source

- int id
- byte cfg
- fn cb
- int unrelated

Destination

- static int sCount
- byte mConfig
- List<Callback> mCallbacks
- int unrelated

Temporary variables:

- int b = id
- Log.print("id=" + b)
- mConfig = cfg
- mCallbacks.add(cb)
- sCount++
STATESpy: Automated State Spill Detection
STATESPY: runtime + static analysis

- Goal: help developers understand how state spill occurs in their entities
State Spill in Android Services
Evaluating Android system services

- StateSpy monitors service stub boundary (onTransact)
- `monkey` induces real apps to invoke various transactions

Found state spill in 94% of service stubs analyzed.
Secondary state spill

Hinders fault tolerance, hot-swapping, maintainability
Case study: Flux  [EuroSys’15]

- Android app migration

Case study: Flux

- Android app migration

Case study: Flux [EuroSys’15]

- Android app migration via record & replay
- Manually handles residual dependencies with *decorator methods* for each service transaction
  - Significant effort to overcome state spill
Comparison with Flux

- Using Flux apps, we reproduced 113 unique transactions for analysis with STATESPY
High correlation with state spill

- State spill identifies problematic service transactions
  - and which states need special handling
**STATESPY catches what’s missing**

- Found state spill in 18 (21%) undecorated methods, each is potentially dangerous
- Easy detection demonstrates `STATESPY`’s utility
Parting Thoughts
Designs to avoid state spill

• Client-provided resources
• Stateless communication
  RESTful principle
• Separation of multiplexing from indirection
• Hardening of entity state
• Modularity without interdependence
Related work

- Coupling\textsuperscript{[1]}/modularity\textsuperscript{[2]} as a \textit{necessary} condition
- Info-flow analysis\textsuperscript{[3,4]}
- Designs that partially reduce state spill
  - Compartmentalizing important states
    - Barrelfish/DC\textsuperscript{[5]}, Microreboot\textsuperscript{[6]}, CuriOS\textsuperscript{[7]}
- RESTful architectures (web)\textsuperscript{[11,12]}
Conclusion

• State spill is an underlying problem that hinders many computing goals
• Prevalent and deeply ingrained in many OSes
• Reducing state spill will lead to better designs
  • More so than minimizing coupling, etc.
• Next steps: redesign OS to minimize state spill

STATESPY & more:  http://download.recg.org
References


