Static Spotter for Scalability
Anti-Patterns Detection

Jinying Yu and Goran Piskachev
**Motivation**

[Image of a cloud with a BookStore icon and user icons pointing to it.]

**Graphs:**
- **Response time [s]** vs. **Number of Users**
  - Expected: Blue triangles
  - Measured: Orange line

- **CPU Utilization [%]** vs. **Number of Users**
  - Expected: Blue triangles
  - Measured: Orange line

---

**BookStore System architect**

December 2, 2014

Static Spotter for Scalability Anti-Patterns Detection
Problem

- **Response time [s]**
  - Number of Users: 1 6 11 16 21 26 31
  - Response time increase with number of users.

- **CPU Utilization [%]**
  - Number of Users: 1 6 11 16 21 26 31
  - CPU utilization remains relatively stable.

**Current system**
- Legacy code
- Only source code available
- Does not scale!

**Modernization**
- Detect problems
- Re-design (re-implment)

**Goal**
- Scalable system

**How?**
- System architect

---

CloudScale

December 2, 2014

Static Spotter for Scalability Anti-Patterns Detection
Solution

Problem: Scalability Problems → Static Spotter

Outline

- 1-lane-bridge
- Static Spotter: Overview
- Static Spotter: Specification
- Static Spotter: Detection
1-lane-bridge

Static Spotter: Overview

Static Spotter: Specification

Static Spotter: Detection

1-lane-bridge in traffic

In Software:
- passive resource limits the concurrency
- e.g. synchronization points, database locks
- may cause congestion

Response time [s]

Number of Users

CPU Utilization [%]

Number of Users

CloudScale

December 2, 2014

Static Spotter for Scalability Anti-Patterns Detection
1-lane-bridge

Static Spotter: Overview

Static Spotter: Specification

Static Spotter: Detection

Model of the System

- abstract syntax graph (ASG)

Patterns

- graph transformation rules wrt ASG

Static Spotter Match Engine

- isomorphic graph matching

List of Candidates

- annotated model
class Book {
    synchronized void modify() {
        ....
    }
    ...
}
How to detect candidates for 1-lane-bridge? e.g. synchronization spots
- synchronized method
- synchronized block

Graphical DSL for patterns specification
Graph matching algorithm
- single root object
- the graph transformation rules are leveled
- combined a bottom-up strategy and a top-down strategy

Results
- list of annotations

Static Spotter: Overview
1-lane-bridge

Static Spotter: Specification

Static Spotter: Detection

CloudScale December 2, 2014
Static Spotter for Scalability Anti-Patterns Detection 9
Tool Demo!
Conclusion

Model of the System

Patterns

Static Spotter Match Engine

List of Detected Pattern Candidates

Dynamic Spotter

Affecting Anti-patterns

Catalog of (Semi)-automatic solutions
Links

- CloudScale project: http://www.cloudscale-project.eu/
- Tool installation: https://code.google.com/p/reclipse-emf/
Back-up slides
Rule leveling:
A rule depending only on objects in the initial ASG gets number 1. A rule depending on other rules, i.e., whose definition includes annotations created by other rules, gets a higher number consistent with the natural topological order of the rules. Rules included in cycles concerning their dependencies get the same level number and are marked as recursive.

Bottom-up then top-down strategy:
After parsing the source code to create the ASG, the analysis starts in bottom-up mode. Initially, all ASG objects schedule level 1 rules, i.e., those depending on ASG objects only. When a rule that depends on other rules cannot be applied in bottom-up mode, the algorithm switches to top-down mode, which uses a separate top-down priority queue. The top-down strategy tries to make the other rules create the missing annotations based on currently available information. The rule/context pair at the front of the top-down mode queue is not dequeued if the rule involved schedules other lower-level rules. Instead, pairs added to the top-down queue are queued in ascending order of their level number. This means that the higher-level rule will be reconsidered after the lower-level rules on which it depends (if these succeed). Using a priority queue rather than a stack means that the top-down algorithm goes as far down the ASG as quickly as possible.
The algorithm runs in top-down mode until the top-down queue is empty or a rule in the queue fails with no alternative contexts left to explore.
The overall analysis finishes when the bottom-up queue is empty. In this case the algorithm has analysed all ASG objects and created annotations on the objects for all rules that could be applied.
1-lane-bridge

CloudScale Method

Static Spotter: Overview

Static Spotter: Specification

Static Spotter: Detection

(1) Requirements Identification

(3) Realization

(4) Deployment

(5) Operation

(2) System Construction and Analysis

(6) Monitoring
(1) Requirements Identification

(2) System Construction and Analysis

Extractor

Reverse Engineering

(Re)design system

Solution available?

(3) Realization

Apply Anti-pattern solution

(4) Deployment

Requirements met?

(5) Operation

Analyser and Static Spotter

yes

yes

no

no

Dynamic Spotter

(6) Monitoring

Legend

decision

Process step

System Construction and Analysis

Static Spotter: Overview

Static Spotter: Specification

Static Spotter: Detection

CloudScale Method

1-lane-bridge

Static Spotter for Scalability Anti-Patterns Detection