Feedback-Driven Concurrency Refinement and Tuning for Performance Models

Diploma Thesis Proposal Talk
Misha Strittmatter, Lucia Happe, Philipp Merkle | July 16, 2012
Motivation

Where to Optimize?

Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012

1/10
Motivation

Find Hot-Spots

Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Motivation

Find Hot-Path

Introduction
Approach
Related Work
Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012

1/10
Problem:
- Concurrency Flaws in SW
- Fix Flaws as Early as Possible
- Model Lacks Necessary Details
- Model Auto-Tuning very limited

Idea:
- Analyze Performance Model
- Involve Model Developer
- Advice & Feedback Loop

Benefit:
- Flaws Fixed Early
- Developer Feedback Tames Abstraction
- Refinement Enables Tuning

Action:
- Create Analysis & Feedback Process
- Expand on Bottleneck Analysis
Process

PCM Model

Analysis of Structure and Simulation Results

Consult or Advice Model Developer
Analysis Results

- Load Test: Active/Passive Resource with Highest Utilization
- Usage per Component/Internal Action, Hot-Spots
- “Hot-path” Through the System
- Peak/Burst Test: “Slowest” Part of the System
- Imbalances Between Resources of the Same Type
- Anti-Patterns: OLB, Architecture Bottleneck, Extensive Processing, More is Less?
- ...

Results can be aggregated over a range of workloads, where resources did not change.
Refine Critical Section

10 s → 10 s → 10 s → 50 s
20 s
...
Redeployment

PC1 (U=10%)  PC2 (U=95%)  LAN (U=80%)

Redeploy?

Introduction  Approach  Related Work  Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning  July 16, 2012
Redeployment

PC1 (U=10%)

LAN (U=80%)

PC2 (U=95%)

Redeploy?

Introduction
Approach
Related Work
Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Redeployment

PC1 (U=40%) → LAN (U=100%) → PC2 (U=10%)
Challenges

- Which concurrency flaws can be detected?
- How can they be detected?
- How to solve flaws on design level?
- How to give appropriate feedback?
- How to analyze component cohesion?
- Automate horizontal scaling?
Validation iterations:

1. Insert concurrency flaws into model, can we fix them using presented approach?
2. Analyze real world model, use presented approach, is there improvement?
3. Iter 2 + refactor system implementation, is there improvement?
RW & Delineation
Foundations

Performance Antipatterns
AP Detection
Monitoring
Systematic Experimenting
Software Analysis
Simulation Result Feedback

My Work
Bottleneck Analysis

Model Analysis
Results
Structure

Introduction
Approach
Related Work
Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning
July 16, 2012
Summary

Where to Optimize?

DB

PCM Model
Analysis of Structure and Simulation Results

Consult or Advice Model Developer

Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012

11/10
References I


References II

[Par07] T. Parsons.  
*Automatic detection of performance design and deployment antipatterns in component based enterprise systems.*  

Parallelizing bzip2: A case study in multicore software engineering.  

Concurrency by modularity: Design patterns, a case in point.  

*Automatische Performanzoptimierung paralleler Architekturen.*  

Engineering parallel applications with tunable architectures.  

Software performance antipatterns.  

Software performance antipatterns; common performance problems and their solutions.  
References III

More new software performance antipatterns: Even more ways to shoot yourself in the foot. 

New software performance antipatterns: More ways to shoot yourself in the foot. 

Detection and solution of software performance antipatterns in palladio architectural models. 

*Automated generation of architectural feedback from software performance analysis results.* 

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. 
Kieker: A framework for application performance monitoring and dynamic software analysis. 

Uncovering performance antipatterns by systematic experiments. 

Balancing & Horizontal Scaling

Webserver

U = 40%

Appserver

U = 100%
Balancing & Horizontal Scaling

Webserver
U = 40%

Appserver
U = 33%

Introduction
Approach
Related Work
Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning
July 16, 2012 16/10
Balancing & Horizontal Scaling

- Webserver: U = 40%
- Load Balancer: U = 20%
- Appserver: U = 33%

Introduction
Approach
Related Work
Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Example: Pipeline

5 s

5 s

20 s
Example: Pipeline
Example: Pipeline
Hot-Path Influence

Indirect Impact (2. Grade)

Asynch Fork

Direct Impact

Indirect Impact

No Impact

Asynch Fork

Indirect Impact

Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Risks

- Detection of Certain Concurrency Flaws not Feasible
- Technical Problems Implementing the Analysis Framework
- Technical Problems with Validation
- Changes in PCM Tool-Chain
Possible Refactoring Actions

- Insert Load Balancer & Replicate Node
- Soften Hot-Spot (Refine, Task-/Data-Parallelism, Florian’s Skeletons, ...)
- Try to Resolve OLB (Reduce CS Size, Increase Cap, Replicate Resources, Shared Access, ...)
- Redeploy Component
- Change Scheduling
- ...

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Classical Bottleneck Analysis

- Increase Load
- Run Test
- Find Bottleneck
- Add Resources

- System Saturated?
  - Yes: Find Bottleneck
  - No: Can Resources be Added to Remove the Bottleneck?
    - Yes: Add Resources
    - No: Not Scalable
Motivation: Analysis
Motivation: Analysis

Introduction

Approach

Related Work

Summary

Misha Strittmatter – Feedback-Driven Concurrency Refinement and Tuning

July 16, 2012
Motivation: Analysis
Motivation: Analysis

- Bottleneck X
- Antipattern Y
Refs from Foundations/RW

- Anti-Patterns [SW00, SW02, SW03a, SW03b]
- Patterns [ZH10, RKR10]
- Autotuning [Sch10, SPT10, PJT09]
- Monitoring [Par05, Par07, MCC1995, BPSH05]
- Anne & Catia [TK11]
- Structure [CDME10, CMRT10]
- Results: [Erd12]
- Catia AP Feedback Diss: [Tru11]
- Feedback: [KSB10]
- Alex: [Wer12]
- SoPeCo: [WHHH10]
- Kieker: [vWHH12]
Copyright to the title image ("Ready, Set, Go 2") is owned by the user bjarwicke of http://www.sxc.hu
Duplicating the image from this presentation is prohibited by its license. Instead you can download it there:
http://www.sxc.hu/photo/789596

Cliparts from:
http://office.microsoft.com/de-de/images/

All other images by me.