Performance Monitoring of Database Operations

Christian Zirkelbach

July 29, 2015
1. Introduction

2. Approach

3. Implementation

4. Evaluation

5. Related Work

6. Conclusions & Future Work
Performance of new or legacy software systems is insufficient
Performance problems or bottlenecks are supposed or detected
Often related to database operations
Figure: Enriched performance issue detection workflow
Requirements

Introduction ▶ Requirements

- Monitoring Component
  - executed SQL and prepared SQL statements
  - their call parameters
  - execution times

- Analysis & Visualization Component
  - analyzing recorded data
  - filtering, sorting, ...
  - grouping prepared statements
1. Identification of Performance Analysis Methods and Tools
2. Implementation of a Tool for Database Performance Analysis
3. Generic Monitoring Approach
4. Evaluation of the Developed Tool
Approach
Figure: Architecture of our software system as component diagram
Figure: Relationships between major classes and interfaces in *java.sql*
Figure: Monitoring components
Monitoring Record Structure

Approach ▶ Monitoring Component

- record type: differs between before or after event
- timestamp: represents date and time of the record
- operation name: full Java class name
- return type: e.g., ResultSet, boolean or int
- return value: e.g., number of affected database records
- operation arguments: e.g., SQL statement
Figure: Generic Monitoring Record Processing based on P&F architecture
Specific Database Call Handling

Approach ▶ Analysis Component

Figure: Specific Database Call Handling based on P&F architecture
Mock-up: Prepared Statements

Approach → Visualization Component

Figure: Mock-up view based on *Call Tree Views* [De Pauw et al. (2002)]
Implementation
Figure: Statement view screenshot
### Aggregated Statements View

**Implementation ▶ Visualization Component**

![Aggregated statement view screenshot](image)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Count</th>
<th>Total</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT itemid, productid, listprice FROM item</td>
<td>2</td>
<td>300008 ns</td>
<td>150004 ns</td>
<td>100004 ns</td>
<td>200004 ns</td>
</tr>
<tr>
<td>DROP INDEX productName</td>
<td>1</td>
<td>3011932 ns</td>
<td>3011932 ns</td>
<td>3011932 ns</td>
<td>3011932 ns</td>
</tr>
</tbody>
</table>

**Operation:**

```java
boolean java.sql.Statement.executeUpdate(String)
```

**Statement:**

```java
SELECT itemid, productid, listprice FROM item
```

**Number of Calls:** 2

<table>
<thead>
<tr>
<th>Total</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>300008 ns</td>
<td>150004 ns</td>
<td>100004 ns</td>
<td>200004 ns</td>
</tr>
</tbody>
</table>

**116 Aggregated Statement(s)**

**Figure:** Aggregated statement view screenshot
# Prepared Statements View

**Implementation ➤ Visualization Component**

![Prepared statement view screenshot](image)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Count</th>
<th>Total</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = ?</td>
<td>7</td>
<td>5445597 ns</td>
<td>777942 ns</td>
<td>501614 ns</td>
<td>1296373 ns</td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT QTY AS VALUE FROM INVENTORY WHERE ITEMID = EST-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation:** PreparedStatement java.sql.Connection.prepareStatement(String)

**Abstract PreparedStatement:**

```
SELECT PRODUCTID, NAME, DESCN AS DESCRIPTION, CATEGORY AS CATEGORYID FROMPRODUCT
WHERE CATEGORY = ?
```

13 PreparedStatement(s)

**Figure:** Prepared statement view screenshot
Evaluation
Usability Test Experiment

Based on a questionnaire, 36 participants

Questionnaires have been used a long time to evaluate user interfaces [Root and Draper(1983)]
3 Statements

3.1 Name the Trace ID and response time (in ms) of the statement, that has the highest response time.

3.2 What is its underlying calling Java operation?

3.3 What kind of SQL statement took the lowest amount of time?
## Evaluation

<table>
<thead>
<tr>
<th>Question</th>
<th>Average Correctness (Percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3.1</td>
<td>67%</td>
</tr>
<tr>
<td>Q3.2</td>
<td>97%</td>
</tr>
<tr>
<td>Q3.3</td>
<td>97%</td>
</tr>
</tbody>
</table>

**Figure:** Average correctness per question within the statements part
6 Debriefing Questions

6.1 How difficult was it to **navigate** through the program?

- very difficult [ ] [ ] [ ] [ ] [ ] very easy

6.2 How difficult was it to **filter** and **sort** database statements for specific problems?

- very bad [ ] [ ] [ ] [ ] [ ] very good

6.3 Was the program **easy to use**?

- very difficult [ ] [ ] [ ] [ ] [ ] very easy

6.4 How was your **overall impression** of the tool?

- very bad [ ] [ ] [ ] [ ] [ ] very good
Debriefing Questions

Evaluation > Results

Figure: Average rating of easy or good per question within the debriefing questions

- Q6.1: 84%
- Q6.2: 79%
- Q6.3: 82%
- Q6.4: 75%
Evaluation ▶ Results

- Positive feedback from participants
- High average correctness rates
- Positive usability rating
Related Work

▶ Cloud Monitoring [Ma et al. (2012)]
Conclusions & Future Work

- Software supports performance analysis on database operations
- Monitoring component versatile (Kieker Trace Diagnosis and ExplorViz [Fittkau 2015])
- Approach validated through conducted experiment
Future Work

Conclusions & Future Work

- Easier filtering, e.g., based on substrings
- More visualization options, e.g., 3D visualizations
- Further experiments, e.g., controlled experiments similar to [Fittkau et al., May 2015a]
AspectJ language extension.
URL http://www.eclipse.org/aspectj/.

Understanding Web applications through dynamic analysis.

C. Artho and A. Biere.
Combined Static and Dynamic Analysis.

C. Artho, V. Schuppan, A. Biere, P. Eugster, M. Baur, and B. Zweimüller.

T. Ball.
The Concept of Dynamic Analysis.

Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications.
M. Bastian, S. Heymann, M. Jacomy, et al.

S. Becker, W. Hasselbring, A. van Hoorn, S. Kounev, and R. Reussner.

G. Canfora Harman and M. Di Penta.
New Frontiers of Reverse Engineering.

A. Chawla and A. Orso.
A Generic Instrumentation Framework for Collecting Dynamic Information.

S. Chiba.
Javassist - a reflection-based programming wizard for Java.

E. Chikofsky and I. Cross, J.H.
Reverse engineering and design recovery: a taxonomy.

J. M. Coble, J. Karat, M. J. Orland, and M. G. Kahn.
Iterative usability testing: ensuring a usable clinical workstation.
B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke.  
A Systematic Survey of Program Comprehension through Dynamic Analysis.  

Visualizing the execution of Java programs.  

S. Ducasse and D. Pollet.  
Software Architecture Reconstruction: A Process-Oriented Taxonomy.  

J. Ehlers, A. van Hoorn, J. Waller, and W. Hasselbring.  
Self-adaptive Software System Monitoring for Performance Anomaly Localization.  

H. Eichelberger and K. Schmid.  
Flexible resource monitoring of Java programs.  

T. Eisenbarth, R. Koschke, and D. Simon.  
Aiding program comprehension by static and dynamic feature analysis.  

M. D. Ernst.  
Static and dynamic analysis: Synergy and duality.  
References

F. Fittkau.
Live Trace Visualization for System and Program Comprehension in Large Software Landscapes.
In PhD Topic Presentation, Juli 2015.

F. Fittkau, J. Waller, P. C. Brauer, and W. Hasselbring.
Scalable and Live Trace Processing with Kieker Utilizing Cloud Computing.

F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring.
Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach.
In 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013), pages 1–4, September 2013b.

F. Fittkau, S. Finke, W. Hasselbring, and J. Waller.
Comparing Trace Visualizations for Program Comprehension through Controlled Experiments.
In 23rd IEEE International Conference on Program Comprehension (ICPC 2015), Mai 2015a.

F. Fittkau, S. Roth, and W. Hasselbring.
ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes.
In 23rd European Conference on Information Systems (ECIS 2015), Mai 2015b.

Florian Fittkau.
URL http://www.explorviz.net/.

M. Fowler.
Patterns of enterprise application architecture.
References V

L. Froihofer, G. Glos, J. Osrael, and K. M. Goeschka.
Overview and evaluation of constraint validation approaches in Java.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides.
Design patterns: elements of reusable object-oriented software.

JDBC Database Access with Java, volume 7.

W. Hasselbring.
Reverse Engineering of Dependency Graphs via Dynamic Analysis.

R. Jung, R. Heinrich, and E. Schmieders.
Model-driven Instrumentation with Kieker and Palladio to forecast Dynamic Applications.

J. Karat.
Evolving the scope of user-centered design.
Aspect-oriented programming.

G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold.
An Overview of AspectJ.

C. Knight and M. Munro.
Virtual but visible software.

R. Likert.
A technique for the measurement of attitudes.
*Archives of psychology*, 1932.

A. M. Lund.
Measuring usability with the USE questionnaire.

K. Ma, R. Sun, and A. Abraham.
Toward a lightweight framework for monitoring public clouds.
Reverse Engineering: A Roadmap.

Software bottlenecking in client-server systems and rendezvous networks.

A mathematical model of the finding of usability problems.

Oracle.
JDBC™ 4.1 Specification, 2011.

R. Pooley.
Software Engineering and Performance: A Roadmap.

K. Project.
URL http://kieker-monitoring.net/documentation/.
Trace-Context Sensitive Performance Profiling for Enterprise Software Applications.

R. W. Root and S. Draper.
Questionnaires as a software evaluation tool.

A. Sabetta and H. Koziolek.

J. Sauro and J. R. Lewis.
*Quantifying the user experience: Practical statistics for user research.*

E. Stroulia and T. Systä.
Dynamic Analysis for Reverse Engineering and Program Understanding.

L. Titchkosky, M. Arlitt, and C. Williamson.
A Performance Comparison of Dynamic Web Technologies.
ISSN 0163-5999.
P. Tonella and M. Ceccato.
Aspect mining through the formal concept analysis of execution traces.

T. Tullis and W. Albert.

T. S. Tullis and J. N. Stetson.
A comparison of questionnaires for assessing website usability.

A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst.
Continuous Monitoring of Software Services: Design and Application of the Kieker Framework.
Technical Report TR-0921, Department of Computer Science, Kiel University, Germany, Nov. 2009.

DynaMod Project: Dynamic Analysis for Model-Driven Software Modernization.
Invited paper.

A. van Hoorn, H. Knoche, W. Goerigk, and W. Hasselbring.
Model-Driven Instrumentation for Dynamic Analysis of Legacy Software Systems.
A. van Hoorn, J. Waller, and W. Hasselbring.
Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis.

A. van Hoorn, J. Waller, and W. Hasselbring.
Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis.

A. Wert.
Performance Problem Diagnostics by Systematic Experimentation.
In *Proceedings of the 18th International Doctoral Symposium on Components and Architecture, WCOP ’13*, pages 1–6, New York, NY, USA, 2013. ACM.

R. Wettel and M. Lanza.
CodeCity: 3D Visualization of Large-Scale Software.

The Future of Software Performance Engineering.

C. Wulf, N. C. Ehmke, and W. Hasselbring.
Toward a Generic and Concurrency-Aware Pipes & Filters Framework.
In *Symposium on Software Performance 2014: Joint Descartes/Kieker/Palladio Days*, November 2014.
