Combining Kieker with Gephi for Performance Analysis and Interactive Trace Visualization

Christian Zirkelbach, Wilhelm Hasselbring, and Leslie Carr

2015-11-06
Performance issues regarding established Perl software

- Often difficult to identify potential bottlenecks
- Architectural discovery
- Program comprehension
- Performance analysis

→ Visualization of results

→ Kieker Monitoring Framework
Approach
Perl Instrumentation using Kieker

Approach

```perl
use Sub :: WrapPackages
packages = > [qw(EPrints EPrints ::*)],
pre = > sub {
  use Kieker;
  my $kieker = Kieker -> new ();
  my $packageName = $__ [0];
  $packageName =~ s /:://g ;
  $packageName =~ /^(.*) \..*? $ /;
  $kieker -> EntryEvent ($packageName,$1);
},
post = > sub {
  use Kieker;
  my $kieker = Kieker -> new ();
  my $packageName = $__ [0];
  $packageName =~ s /:://g ;
  $packageName =~ /^(.*) \..*? $ /;
  $kieker -> ExitEvent ($packageName,$1);
};
```
Instrumentation Components

Approach

Perl Instrumentation

- Monitoring Writer
- Monitoring Controller
- Monitoring Probe

Kieker Data Bridge

- JMS
- JMS Client
- Service Container
- Monitoring Controller

Kieker.Monitoring

- Monitoring Writer
- Monitoring Controller

Kieker.Analysis

- Monitoring Reader
- Analysis Controller
- Analysis / Visualization Plugin
Performance Analysis Workflow

Approach

1. Perl Instrumentation
2. Monitoring via Kieker
   - Appropriate coverage?
   - no
   - yes
   - finished?
   - no
   - yes

- Batch Visualization via Graphviz
- Interactive Visualization via Gephi
Further analysis and visualization via Gephi

Approach

1. Manipulating the generated dependency call graph (Kieker)
   - Architecture Discovery & Program Comprehension
     - Aggregation based on the package hierarchy
   - Performance Analysis
     - Number of edges (calls) in combination with response times

2. Visualizing the manipulated graph
Further analysis and visualization via Gephi

## Approach

**Data Table**

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>wishes_to_export() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_117</td>
</tr>
<tr>
<td>who_filter() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_128</td>
</tr>
<tr>
<td>show_columns() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_106</td>
</tr>
<tr>
<td>render_title() min: 3ms, avg: 3.00ms, max: 3ms</td>
<td>depNode_169</td>
</tr>
<tr>
<td>render_title() min: 1ms, avg: 2.50ms, max: 4ms</td>
<td>depNode_149</td>
</tr>
<tr>
<td>render_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_150</td>
</tr>
<tr>
<td>render_tab_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_36</td>
</tr>
<tr>
<td>render_tab_title() min: 0ms, avg: 0.00ms, max: 0ms</td>
<td>depNode_64</td>
</tr>
<tr>
<td>render_items() min: 505ms, avg: 505.00ms, max: 505ms</td>
<td>depNode_135</td>
</tr>
<tr>
<td>render_import_bar() min: 21ms, avg: 21.00ms, max: 21ms</td>
<td>depNode_132</td>
</tr>
</tbody>
</table>
Further analysis and visualization via Gephi

Approach

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Type</th>
<th>Id</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>depNode_1</td>
<td>depNode_11/</td>
<td>Directed</td>
<td>1/5</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_125</td>
<td>Directed</td>
<td>1789</td>
<td>2</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_111</td>
<td>Directed</td>
<td>1792</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_116</td>
<td>Directed</td>
<td>1804</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_170</td>
<td>Directed</td>
<td>1807</td>
<td>6</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_101</td>
<td>Directed</td>
<td>1818</td>
<td>131</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_118</td>
<td>Directed</td>
<td>1826</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_2</td>
<td>Directed</td>
<td>1838</td>
<td>10</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_115</td>
<td>Directed</td>
<td>1886</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_155</td>
<td>Directed</td>
<td>1870</td>
<td>1</td>
</tr>
<tr>
<td>depNode_1</td>
<td>depNode_173</td>
<td>Directed</td>
<td>1797</td>
<td>1</td>
</tr>
</tbody>
</table>
Case Study

Let’s evaluate our approach!
Performance analysis of EPrints 3.3.12 using Kieker

- Potential bottleneck detection for upcoming release version 4
- Evaluation in collaboration with the development team
- Software maintenance & modernization
- Different instrumentation levels are possible
- Combined visualization: GraphViz and Gephi
Evaluation Goals

Case Study

- Is it possible to identify an architecture? (Architectural Discovery)
- Can we support other developers by providing information about the software? (Program Comprehension)
- Can we detect potential bottlenecks? (Performance Analysis)
### Manage deposits

<table>
<thead>
<tr>
<th>Last Modified</th>
<th>Title</th>
<th>Item Type</th>
<th>Item Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 Apr 2014 13:56</td>
<td>Thomas Ker of Reeder's trip to the Low Countries, 1620</td>
<td>Book Section</td>
<td>User Workarea</td>
</tr>
<tr>
<td>30 Apr 2014 13:50</td>
<td>Playing with Haddon's string figures</td>
<td>Article</td>
<td>User Workarea</td>
</tr>
<tr>
<td>30 Apr 2014 13:50</td>
<td>Outside in: making sense of the deliberate concealment of garments within buildings</td>
<td>Article</td>
<td>User Workarea</td>
</tr>
<tr>
<td>30 Apr 2014 13:50</td>
<td>[Response to Philip Whalen's article 'Burgundian regionalism and French Republican commercial culture at the 1837 Paris International Exposition']</td>
<td>Article</td>
<td>User Workarea</td>
</tr>
</tbody>
</table>

---

Default VM is powered by EPrints which is developed by the School of Electronics and Computer Science at the University of Southampton. More information and software credit.
Case Study

Architecture Discovery
Batch Visualization via Graphviz

Case Study ▶ Architecture Discovery
Gephi: System Architecture Level

Case Study ▶ Architecture Discovery

- avg: 333.73ms, max: 7239ms

EPrints.Language
- avg: 6.64ms, max: 429ms

EPrints.Repository
- avg: 1.89ms, max: 841ms

EPrints.Database
- avg: 36.50ms, max: 658ms

EPrints.Plugin
- avg: 0.73ms, max: 44ms

EPrints.DataObj
- avg: 3.66ms, max: 218ms

EPrints.DataObj.User
- avg: 5.74ms, max: 156ms

EPrints.Dataset
- avg: 0.59ms, max: 106ms

EPrints.Database.mysql
- avg: 0.63ms, max: 78ms

EPrints.PluginFactory
- avg: 11.49ms, max: 472ms

EPrints.ScreenProcessor
- avg: 1283.63ms, max: 9376ms
Gephi: System Architecture Level

Case Study ▶ Architecture Discovery

EPrints.Repository

EPrints.XML

EPrints.DataObj

EPrints.DataSet

EPrints.Database

EPrints.Language

EPrints.ScreenProcessor

EPrints.Plugin

EPrints.MetaField

EPrints_Utils

Zirkelbach, Hasselbring, and Carr

Combining Kieker with Gephi

2015-11-06 17 / 28
Case Study

Performance Bottleneck Identification
Case Study ▶ Performance Bottleneck Identification

'Entry'

<<assembly component>>
@2:.Instrumentation
<<assembly component>>
@1:EPrints.Plugin.Screen.Items

Request_1()

min: 559ms, avg: 559.00ms, max: 559ms

1

render()

min: 390ms, avg: 390.00ms, max: 390ms
1

can_be_viewed()

min: 0ms, avg: 0.00ms, max: 0ms
3

new()

min: 0ms, avg: 0.38ms, max: 5ms
13

render_title()

min: 0ms, avg: 0.50ms, max: 1ms
2

properties_from()

min: 7ms, avg: 7.00ms, max: 7ms
1

perform_search()

min: 3ms, avg: 3.00ms, max: 3ms
1

render_items()

min: 347ms, avg: 347.00ms, max: 347ms
1

get_filters()

min: 0ms, avg: 0.00ms, max: 0ms
1
GraphViz: Operations - EPrints.Screen.Items

Case Study ➤ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations

Zirkelbach, Hasselbring, and Carr
Combining Kieker with Gephi
2015-11-06 20 / 28
Case Study ▶ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations
Case Study ▶ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations


Figure 11: Reduced operation dependency graph using gephi [8]
Case Study ▶ Performance Bottleneck Identification

Potential bottleneck: database-oriented operations
Related Work
Summary

▶ Well-structured workflow for a performance analysis
▶ Verification through a case study of an established application
▶ Detection of performance issues → basis for recommendations

→ Continuous integration
Infinite Loop Detection in Version 4

Outlook

- 25460
- <<deployment component>> @18:..MetaField
- <<deployment component>> @21:..Itemref
- <<deployment component>> @20:..Counter
- <<deployment component>> @27:..State
What happened after our Case Study?
Application-level Perspective in ExplorViz

Outlook
Physical 3D Model based on ExplorViz

Outlook
Physical 3D Model based on ExplorViz

Outlook