Monitoring of Perl-based web applications using Kieker

Nis Wechselberg

Institut für Informatik
Christian-Albrechts-Universität zu Kiel

2013-06-28
1 Motivation
   • System
   • Problems with Kielprints
Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis


Abstract
Kieker is an extensible framework for monitoring and analyzing the runtime behavior of concurrent or distributed software...
Project summary

- Open access system for scientific research data and publications (*self archiving*)
- Expanded and heavily modified Version of EPrints
- Operated at GEOMAR | Helmholtz Zentrum für Ozeanforschung Kiel
- Expanded the original System OceanRep GEOMAR
- At time of writing about 15,000 documents from more than 1,000 Authors
User Perspective

- Searching for documents on the platform:
  - High Latency, slow responses
  - Slow processing of searches by division
Extreme latency when changing pages in backend

Example: Generation of drop-down menus for authors takes more than 10 seconds
2 Technologies and methods
- Kieker-Framework
- The programming language Perl
- Performance monitoring
Kieker Monitoring Framework

- Tool for dynamic software analysis, created at the research group for Software Engineering
- Recommended tool at SPEC RG Software Repository
Two-part structure: *Kieker.Monitoring* and *Kieker.Analysis*

*Kieker.Monitoring* for monitoring and data generation

*Kieker.Analysis* for analysis and presentation

Loosely coupled by monitoring log or stream

At time of writing no perl support
The programming language Perl

- imperative, platform independent, interpreted
- First published 1987, recent version 5.16 available
- Support for object oriented programming via CPAN available
- Very free syntax, "There’s always one more way to do it"
- Strong support for string manipulation and regular expressions
- Direct support in webservers with mod_perl
Performance monitoring step-by-step

1. Instrumentation of the code with *probes*
2. Execution of the system under observation
3. Logging of monitoring data
4. Analysis of generated data
Gathered data

- Application specific data
  - Function calls and dependencies
  - Frequency of calls
  - Response times

- System data
  - CPU usage
  - RAM usage
  - active processes
  - ...

Nis Wechselberg (CAU Kiel)
Perl.Kieker implementation
- Architecture
- Base module
- Kieker-Data-Bridge
Support for monitoring records, monitoring writer and controlling

Minimal subset of records: OperationEntryEvent, OperationExitEvent, Trace

Analysis with *Kieker.Analysis*
Kieker in perl

- **Kieker**
  - Encapsulation of the functions. Interface
  - Manages Kieker::Writer and Kieker::Controlling instances

- **Kieker::Controlling**
  - Manages Trace-IDs and Order-Indices
  - Creates new traces when requested

- **Kieker::Util**
  - Currently only manages time sampling
  - Converts Perl-microseconds to nanoseconds
Kieker in Perl - Writer

- Kieker::Writer::FileWriter
  - Writes generated records to file system
  - Only used for early testing
- Kieker::Writer::JMSWriter
  - Uses STOMP for Java Message Service
  - Connection to Kieker via Kieker-Data-Bridge
Kieker in Perl - Records

- Kieker::Record::Trace
- Kieker::Record::OperationEntryEvent // OperationExitEvent
  - Event based records for begin and end of execution
  - Record data: function, package, timestamp, trace, orderIndex
  - Trace reconstruction in Kieker.Analysis
Kieker-Data-Bridge I

Kieker Data Bridge

Service Container

TCP Server

Service Connector

Kieker Monitoring

Monitoring Controller

JMS Client

Monitoring Writer

Logging

Periodic Sampling

JMX Interface

Nis Wechselberg (CAU Kiel)

Perl-Monitoring

2013-06-28 20 / 37
By-product at MENGES-project
Universal interface for future expansions
Multiple connectors
  (TCP Client, TCP Server, JMS Client, JMS Embedded)
Accepts both binary data or text messages
Text messages are separated by semicolons
Records identification via mapping file
Amount and type of parameter determined by record type
Example: 1;1362747533540734000;6889;
5;EPrints.current_repository;EPrints
4 Instrumentation
Using CPAN module Sub::WrapPackages

AOP based creation of wrappers

```perl
use Sub::WrapPackages
packages => [qw(EPrints EPrints::*),],
pre => sub {
    use Kieker;
    my $kieker = Kieker->new();
    \$_[0] =~ s/::/./g;
    \$_[0] =~ /^(.*)\..*?$/;
    $kieker->EntryEvent($\_[0],$1);
}
```
5 Testing
- System
- Requests
Comparison between Systems

- Virtual machine using Ubuntu 12.04 LTS
- Database dump from GEOMAR dated 2013-02-07
- Comparison between EPrints 3.2 and Kielprints
Components

VirtualBox

- PostgreSQL
- Apache
- mod_perl
- HTTP
- SQL
- EPrints/Kielprints
5 selected requests at admin interface

Scetching Use-Case *New document*

4 "normal" requests, 1 AJAX request
Results

- Dependencies
- Timing
- Function calls
- Most active modules
## Timings

<table>
<thead>
<tr>
<th>Request</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EP norm.</strong></td>
<td>402 ms</td>
<td>220 ms</td>
<td>136 ms</td>
<td>413 ms</td>
<td>348 ms</td>
</tr>
<tr>
<td><strong>EP inst.</strong></td>
<td>15389 ms</td>
<td>15043 ms</td>
<td>18408 ms</td>
<td><strong>23430 ms</strong></td>
<td><strong>7066 ms</strong></td>
</tr>
<tr>
<td><strong>KP norm.</strong></td>
<td>10270 ms</td>
<td>227 ms</td>
<td>166 ms</td>
<td>13420 ms</td>
<td>18890 ms</td>
</tr>
<tr>
<td><strong>KP inst.</strong></td>
<td>28505 ms</td>
<td>16623 ms</td>
<td>17414 ms</td>
<td><strong>280927 ms</strong></td>
<td><strong>342662 ms</strong></td>
</tr>
<tr>
<td>F1</td>
<td>25,5</td>
<td>1,0</td>
<td>1,2</td>
<td>32,5</td>
<td>54,3</td>
</tr>
<tr>
<td>F2</td>
<td>1,8</td>
<td>1,1</td>
<td>0,9</td>
<td>11,9</td>
<td>48,5</td>
</tr>
</tbody>
</table>

**Table:** Timings of Eprints and Kielprints before and after instrumentation

### Requests

1. Admin homepage after login
2. New Entry
3. File upload (no file provided)
4. Metadata
5. AJAX: More authors
### Function calls

<table>
<thead>
<tr>
<th>Request</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eprints</td>
<td>20875</td>
<td>28590</td>
<td>33171</td>
<td>43056</td>
<td>16980</td>
<td>142672</td>
</tr>
<tr>
<td>Kielprints</td>
<td>31742</td>
<td>36681</td>
<td>40165</td>
<td>905580</td>
<td>934760</td>
<td>1948928</td>
</tr>
<tr>
<td>Factor</td>
<td>1,5</td>
<td>1,3</td>
<td>1,2</td>
<td>21,0</td>
<td>55,1</td>
<td>13,7</td>
</tr>
</tbody>
</table>

**Table:** Function calls at EPrints and Kielprints
# Most active modules

<table>
<thead>
<tr>
<th>EPrints</th>
<th>Aufrufe</th>
<th>Kielprints</th>
<th>Aufrufe</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPrints.XML</td>
<td>12486</td>
<td>EPrints.Database</td>
<td>189624</td>
</tr>
<tr>
<td>EPrints.DataSet</td>
<td>7488</td>
<td>EPrints.DataObj</td>
<td>156391</td>
</tr>
<tr>
<td>EPrints.DataObj</td>
<td>5439</td>
<td>EPrints.XML</td>
<td>107939</td>
</tr>
</tbody>
</table>

**Table:** The 10 most active modules
7 Conclusion
Conclusion

- An monitoring system for Perl based web applications has been created
- Comparison between EPrints and Kielprints
- Discovery: High amount of database queries at Kielprints
- Discovered optimization options both at EPrints and Kielprints.