

Monitoring of Perl-based webapplications using Kieker

Nis Wechselberg

Institut für Informatik
Christian-Albrechts-Universität zu Kiel

2013-06-28

- 1 Motivation
- 2 Technologies and methods
- 3 Perl.Kieker implementation
- 4 Instrumentation
- 5 Testing
- 6 Results
- 7 Conclusion

1 Motivation

- System
- Problems with Kielprints

Geben Sie Ihren Befehl ein

© VirtualDex Project

Wicket & Framework for Apps

C A U

Christian-Albrechts-Universität zu Kiel

ePrints

Startseite Suche Kontakt Sitemap Impressum Datenschutz English

Sie sind hier: Startseite >

Informationen über Informationen für

Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis

Anmelden

van Hoorn, André, Waller, Jan und Hasselbring, Wilhelm (2012) *Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis [Paper]* In: 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE'12), April 22-25, 2012, Boston, Massachusetts, USA.

Tools

RDF+XML

Export

Text
Kieker/CPE2012-camera-ready.pdf - angenommene Version
Download (692Kb) | Vorschau

Image
Kieker/CPE2012-poster.pdf - Begleitmaterial
Download (659Kb) | Vorschau

Slideshow
120424-ICPE-slides-final.pdf - Präsentation
Download (1718Kb) | Vorschau

Offizielle URL: <http://dx.doi.org/10.1145/2188286.2188326>

Abstract

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

- 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

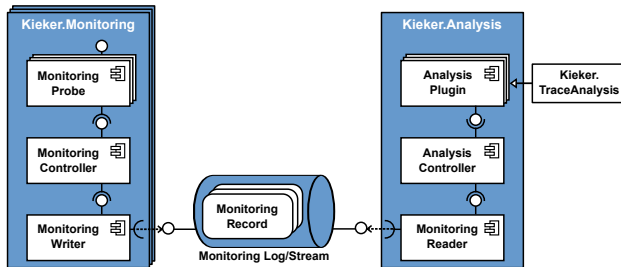
- Searching for documents on the platform:
- High Latency, slow responses
- Slow processing of searches by division

- 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 objectoriented 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 webserver with mod_perl

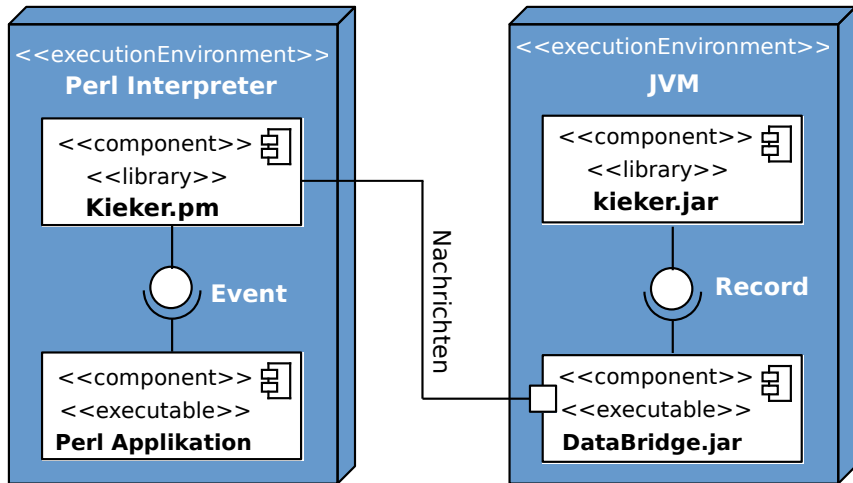
- 1 Instrumentation of the code with *probes*
- 2 Execution of the system under observation
- 3 Logging of monitoring data
- 4 Analysis of generated data

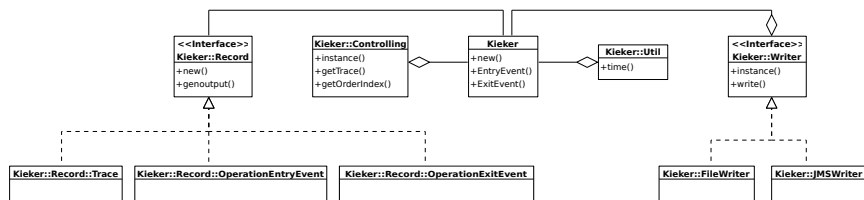
- Application specific data
 - Function calls and dependencies
 - Frequency of calls
 - Response times
- System data
 - CPU usage
 - RAM usage
 - active processes
 - ...

3 Perl.Kieker implementation

- Architecture
- Base module
- Kieker-Data-Bridge

Architectural view





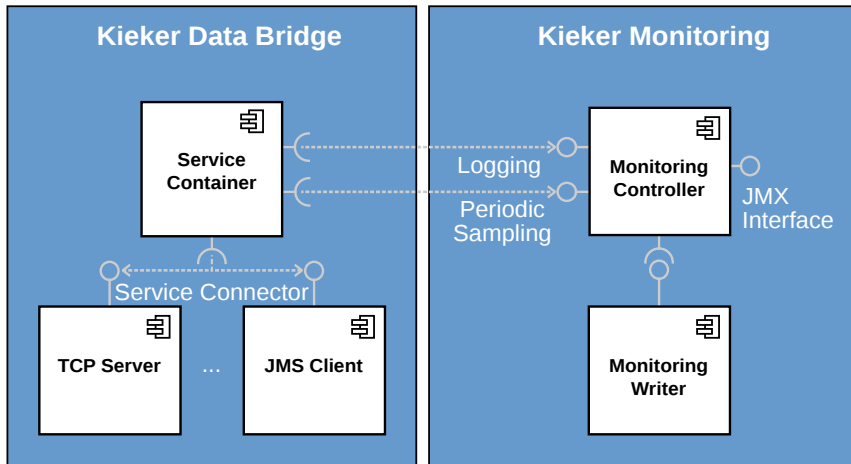
- Support for monitoring records, monitoring writer and controlling
- Minimal subset of records: OperationEntryEvent, OperationExitEvent, Trace
- Analysis with *Kieker.Analysis*

- 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::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::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



- 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

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

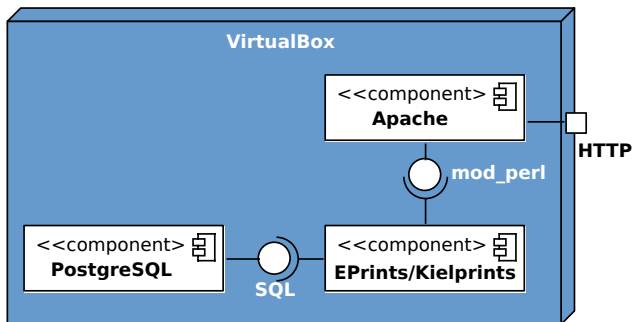

5 Testing

- System
- Requests

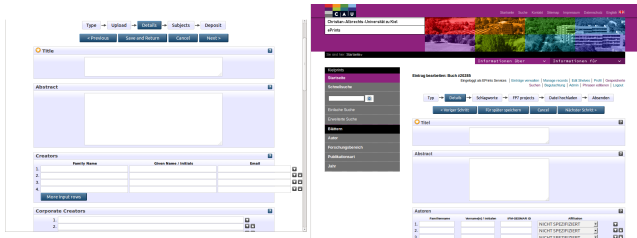
Comparison between Systems

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

Components



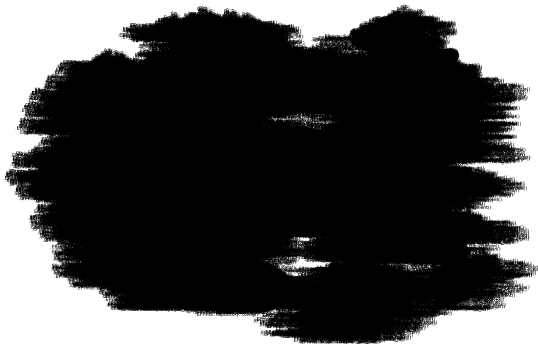
- 5 selected requests at admin interface
- Scetching Use-Case *New document*
- 4 "normal" requests, 1 AJAX request



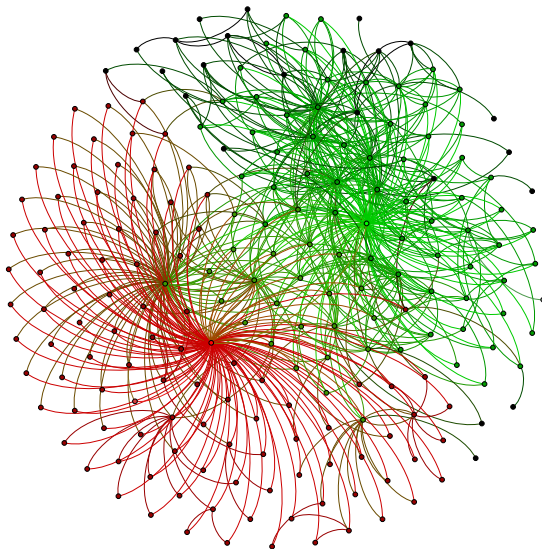
6 Results

- Dependencies
- Timing
- Function calls
- Most active modules

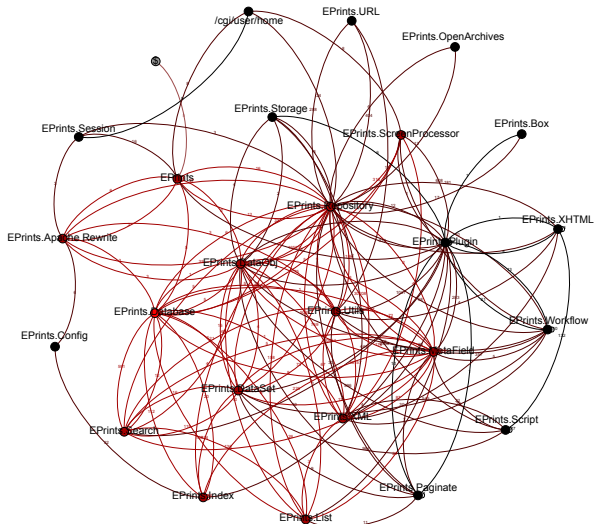
"Call Tree" EPrints



Dependency graph EPrints



Condensed dependency graph



Timings

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

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

Request	1	2	3	4	5	Sum
Eprints	20875	28590	33171	43056	16980	142672
Kielprints	31742	36681	40165	905580	934760	1948928
Factor	1,5	1,3	1,2	21,0	55,1	13,7

Table: Function calls at EPrints and Kielprints

Most active modules

EPrints	Aufrufe	Kielprints	Aufrufe
EPrints.Script.Compiler	30304	EPrints.MetaField	501872
EPrints.Repository	25980	EPrints.DataSet	253493
EPrints.MetaField	18374	EPrints.Repository	243699
EPrints.XML	12486	EPrints.Database	189624
EPrints.DataSet	7488	EPrints.DataObj	156391
EPrints.Utils	7220	EPrints.MetaField.Id	155978
EPrints.XML.EPC	5703	EPrints.Utils	118036
EPrints.DataObj	5439	EPrints.XML	107939
EPrints.Database	3466	EPrints.MetaField.Multilang	45312
EPrints.Script.Compiled	3404	EPrints.Script.Compiler	39550

Table: The 10 most active modules

7 Conclusion

Conclusion

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