Monitoring von Perl-basierten Web Anwendungen mit Kieker

Bachelor-Abschlusskolloquium

Nis Wechselberg

Institut für Informatik
Christian-Albrechts-Universität zu Kiel

26. März 2013

Gliederung

1. Motivation
2. Technologien und Methoden
3. Perl.Kieker Implementierung
4. Instrumentierung
5. Testdurchführung
6. Ergebnisse
7. Fazit

Gliederung

1. Motivation
   - System
   - Beobachtete Performanceprobleme

Kielprints


Christian-Albrechts-Universität zu Kiel
ePrints
Startseite Suche Kontakt Sitemap Impressum Datenschutz English
Sie sind hier: Startseite»
Informationen über
Informationen für
Kielprints
Startseite
Schnellsuche
Einfache Suche
Erweiterte Suche
Blättern
Autor
Forschungsbereich
Publikationsart
Jahr
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
KiekerICPE2012-camera-ready.pdf - angenommene Version
Download (692Kb) | Vorschau
Image
KiekerICPE2012-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

Projektbeschreibung

- Plattform zur Veröffentlichung von wissenschaftlichen Dokumenten (*self archiving*)
- Ergänzte und modifizierte Version von EPrints
- Betrieben durch das GEOMAR | Helmholtz Zentrum für Ozeanforschung Kiel
- Erweiterung des OceanRep GEOMAR
- Über 15000 Veröffentlichungen von mehr als 1000 Autoren

Benutzerperspektive

- Abrufen von Einträgen über die Plattform:

- Erhöhte Antwortzeiten
- Langsame Suche in den Fachbereichen

Administration/Betrieb

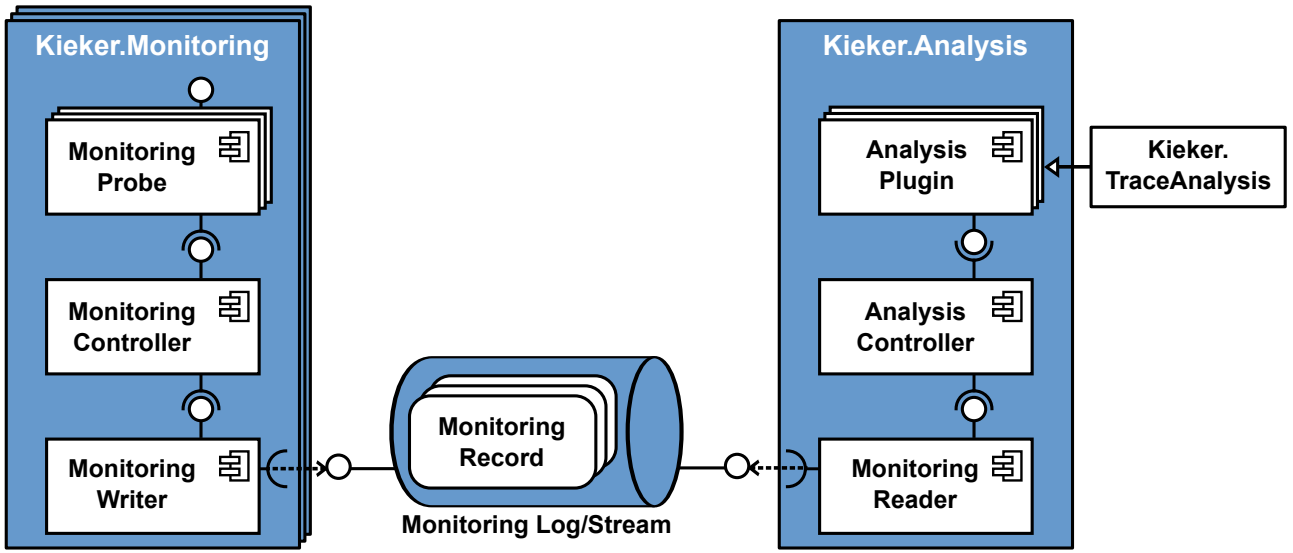
- Hohe Wartezeiten bei Seitenwechseln im Backend
- Beispiel: Beim Eintragen von neuen Publikationen über das Backend und hinzufügen von Autoren dauert die Generierung von Dropdown-Menüs etwa 10 Sekunden.

Gliederung

2. Technologien und Methoden
   - Kieker-Framework
   - Programmiersprache Perl
   - Performance-Monitoring

Kieker Monitoring Framework

- Monitoring Tool aus diesem Lehrstuhl
- Empfohlenes Tool im SPEC RG Software Repository



Überblick über Kieker

- Aufteilung in *Kieker.Monitoring* und *Kieker.Analysis*
- *Kieker.Monitoring* zur Instrumentierung und Überwachung
- *Kieker.Analysis* zur Auswertung und Veranschaulichung
- Kommunikation über Monitoring Log oder Stream

- Bisher keine Unterstützung für Perl

Programmiersprache Perl

- imperative, plattformunabhängige, interpretierte Sprache
- Entwickelt 1987, heute in Version 5.16 verfügbar
- Module für Objektorientierung über CPAN verfügbar
- Sehr viele Freiheiten für Programmierer (freie Syntax)
- Starke Funktionen für String-Manipulationen und Reguläre Ausdrücke
- Direkte Integration in Webserver mittels mod_perl

Vorgehen bei Performance-Monitoring

1. Instrumentierung des Codes mit *Probes*
2. Ausführung des instrumentierten Codes
3. Protokollierung von Monitoring Daten
4. Auswertung der Daten mittels geeigneter Tools

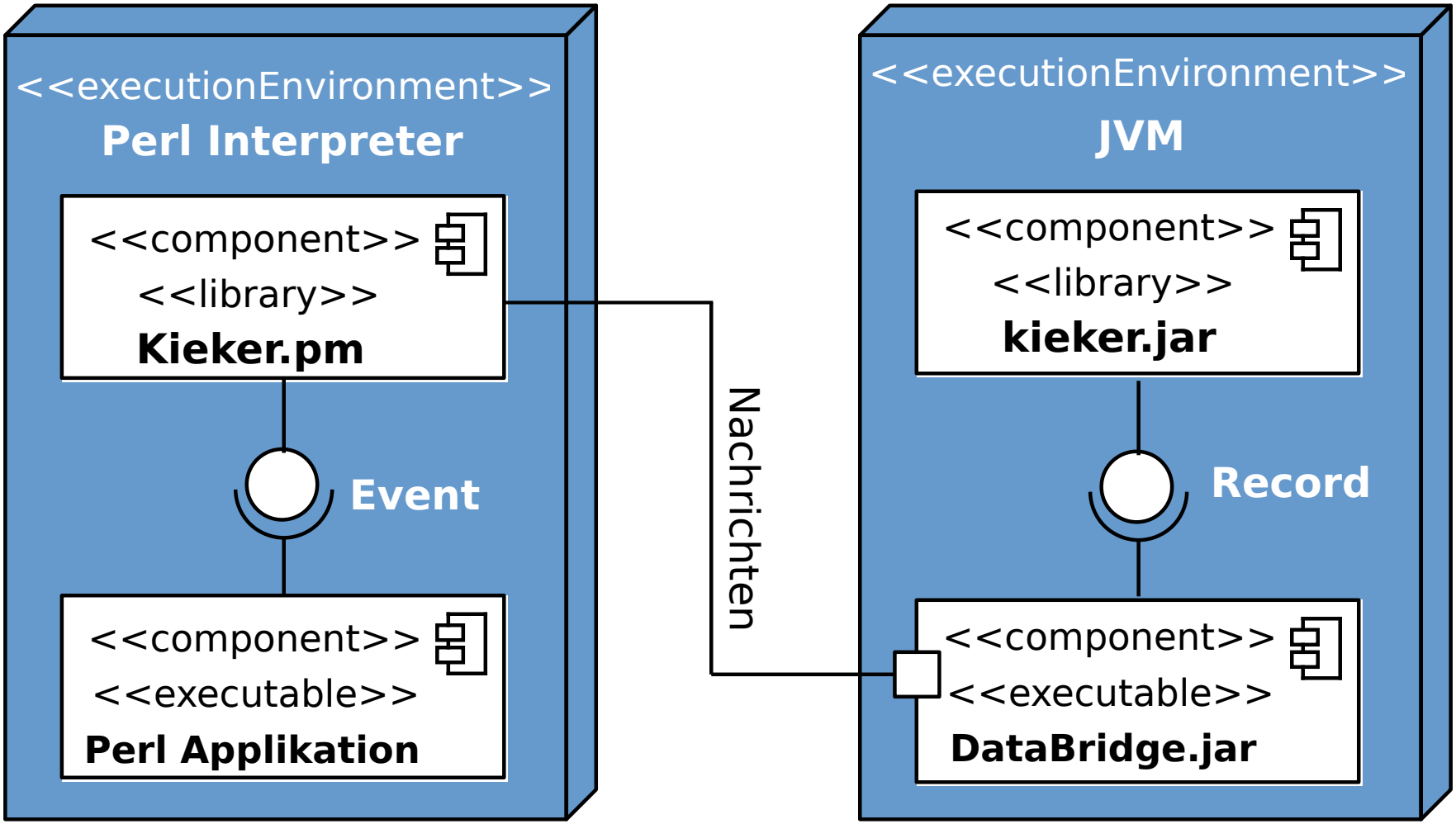
Erfasste Daten

- Anwendungsdaten
  - Aufrufreihenfolge
  - Aufrufhäufigkeiten
  - Ausführungszeiten
- Systemdaten
  - CPU-Auslastung
  - Arbeitsspeicher
  - aktive Prozesse
  - ...

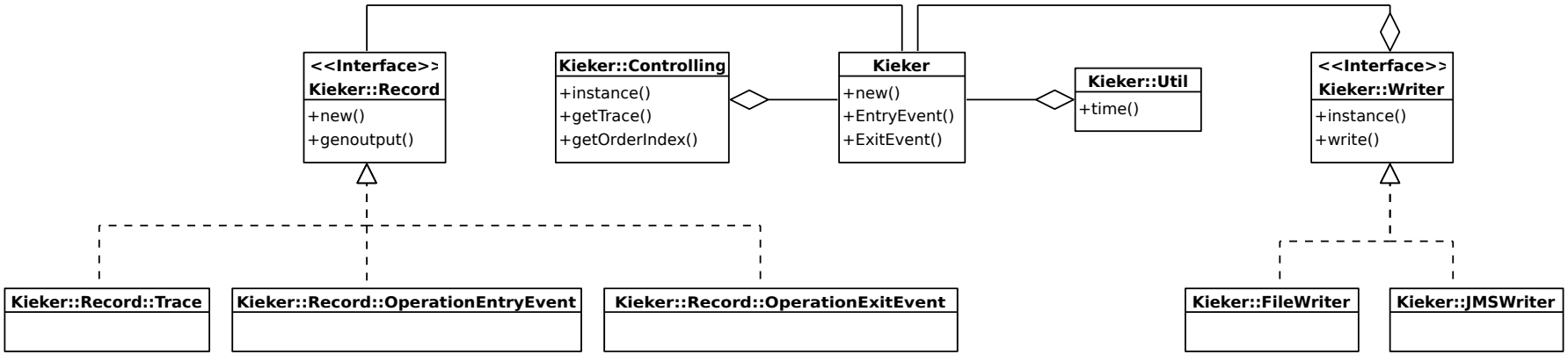
Gliederung

3 Perl.Kieker Implementierung
- Architektur
- Basismodule
- Kieker-Data-Bridge

Architekturentwurf



Neuimplementierungen in Perl



- Implementierung von Monitoring Records, Monitoring Writer und Kontrollmechanismen
- Unterstützte Records: OperationEntryEvent, OperationExitEvent, Trace
- Auswertung mit *Kieker.Analysis*

Kieker in Perl

- Kieker
  - Kapselt die weiteren Module zur einfacheren Verwendung
  - Hält die Instanzen von Kieker::Writer und Kieker::Controlling
- Kieker::Controlling
  - Verwaltet Trace-IDs und Order-Indices
  - Erzeugt bei Bedarf neue Traces
- Kieker::Util
  - Verwaltet Zeitmessung
  - Rechnet Perl-Mikrosekunden in Nanosekunden um

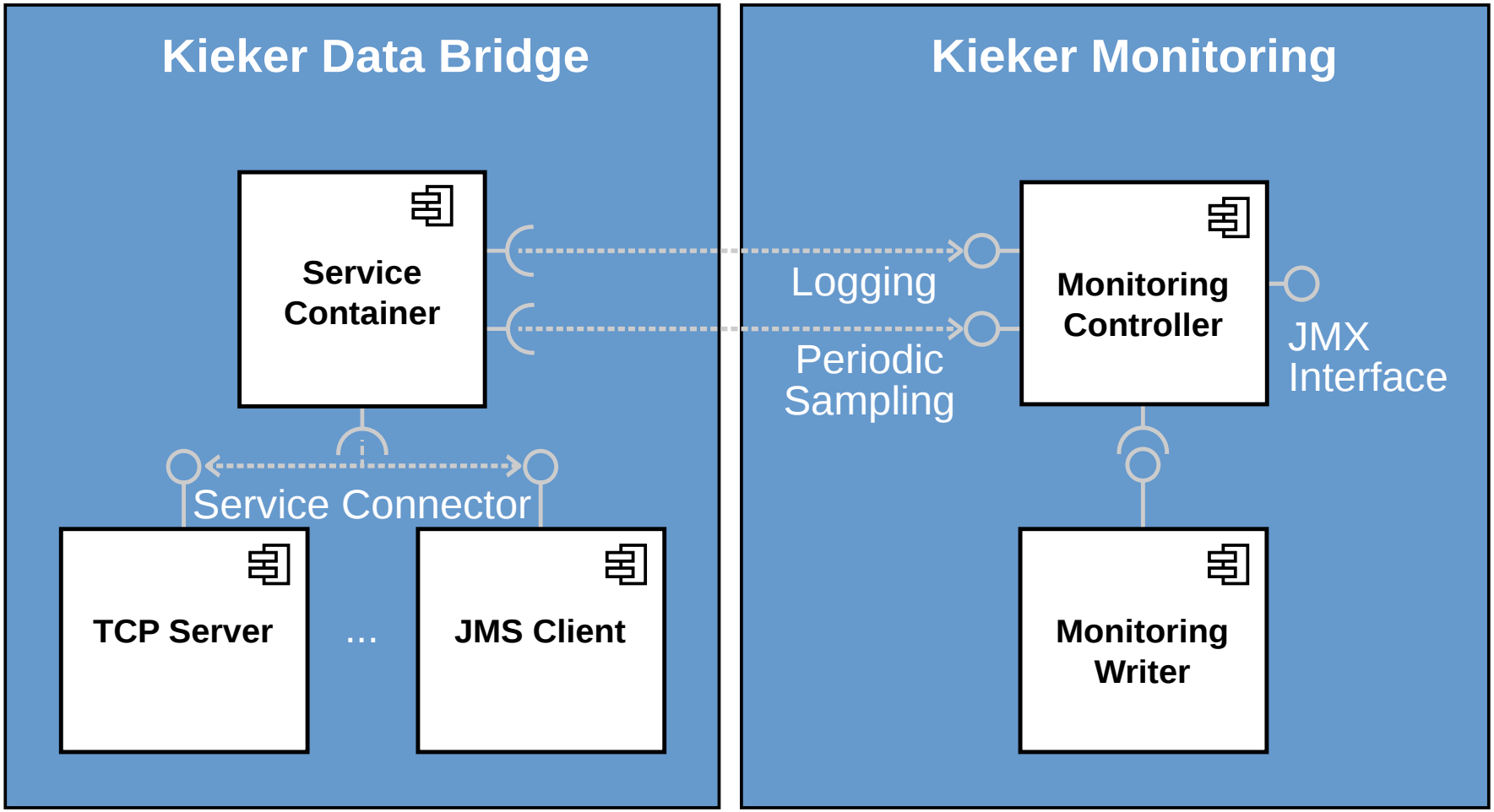
Kieker in Perl - Writer

- Kieker::Writer::FileWriter
  - Schreibt serialisierte Records in angelegte Datei
  - Wurde für frühe Test verwendet
- Kieker::Writer::JMSWriter
  - Verwendet den Java Message Service zur Übertragung der Nachrichten
  - Anbindung an Kieker mittels Kieker-Data-Bridge

Kieker in Perl - Records

- Kieker::Record::Trace
- Kieker::Record::OperationEntryEvent // OperationExitEvent
  - Zeigen Start und Ende einer Funktion an
  - Benötigte Daten: Funktionsname, Paketname, Zeitstempel, Trace, OrderIndex
  - Rekonstruktion durch Kieker.Analysis

Kieker-Data-Bridge I



Kieker-Data-Bridge II

- Entwickelt aus dem MENGES-Projekt heraus
- Stellt eine einheitliche Schnittstelle für neue Erweiterungen dar
- Verschiedene Konnektoren
  (TCP Client, TCP Server, JMS Client, JMS Embedded)
- Empfängt Binärdaten und Textnachrichten

Kieker-Data-Bridge III

- Textnachrichten mit Semikolon getrennt
- Records werden mit Mapping-Datei identifiziert
- Parameter hängen von Record-Typ ab
- Beispiel: `1;1362747533540734000;6889;5;EPrints.current_repository;EPrints`

Gliederung

4 Instrumentierung

Automatische Instrumentierung

- Verwendung von CPAN-Modul Sub::WrapPackages
- Automatische Erstellung von Wrappern

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

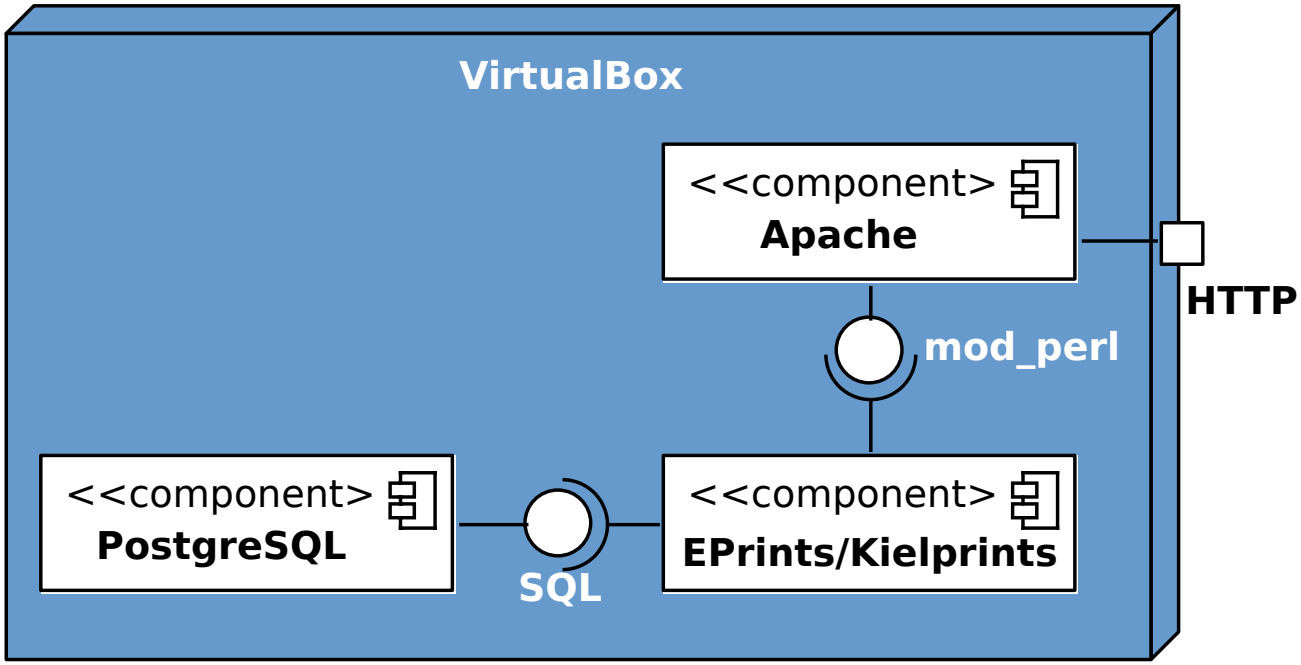
Gliederung

5 Testdurchführung
- System
- Requests

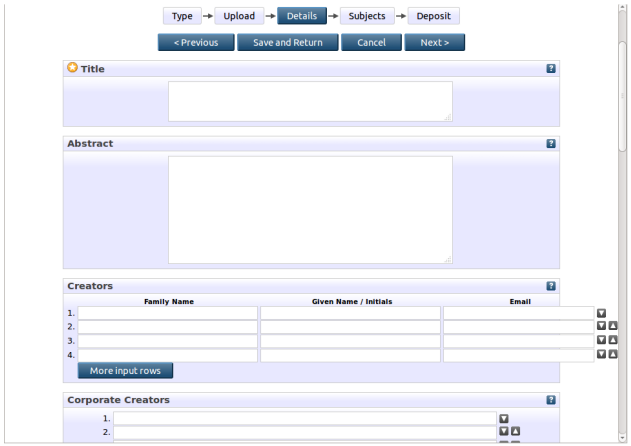
Vergleichstest

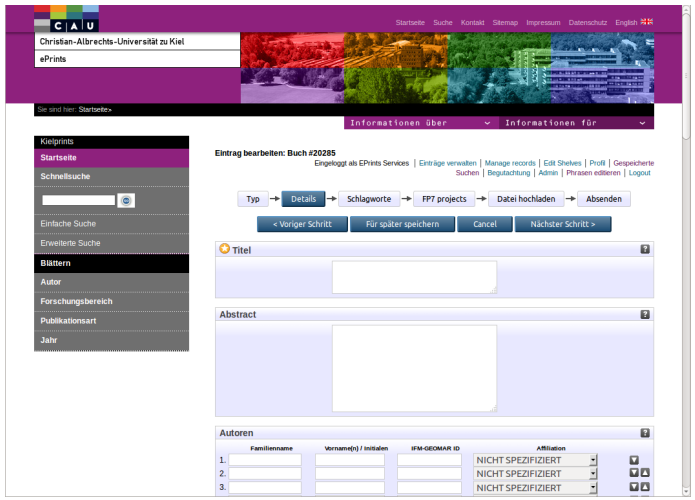
- Virtuelle Maschine unter Ubuntu 12.04 LTS
- Datenbasis vom GEOMAR vom 7. Februar 2013
- Vergleich von EPrints 3.2 und Kielprints

Komponenten



- 5 beispielhafte Anfragen auf der Admin-Oberfläche
- Anwendungsfall *Neue Publikation eintragen* (Skizze)
- 4 normale Requests, 1 AJAX Request





Gliederung

6 Ergebnisse
- Zeitliches Verhalten
- Funktionsaufrufe
- Aktive Pakete
- Abhängigkeiten

Zeitmessungen

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

Tabelle: Zeitliches Verhalten von Eprints und Kielprints vor und nach Instrumentierung

Requests

1. Admin Homepage
2. Neue Publikation eintragen
3. Dateiupload
4. Metadaten
5. AJAX: Mehr Autorenfelder

Funktionsaufrufe

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

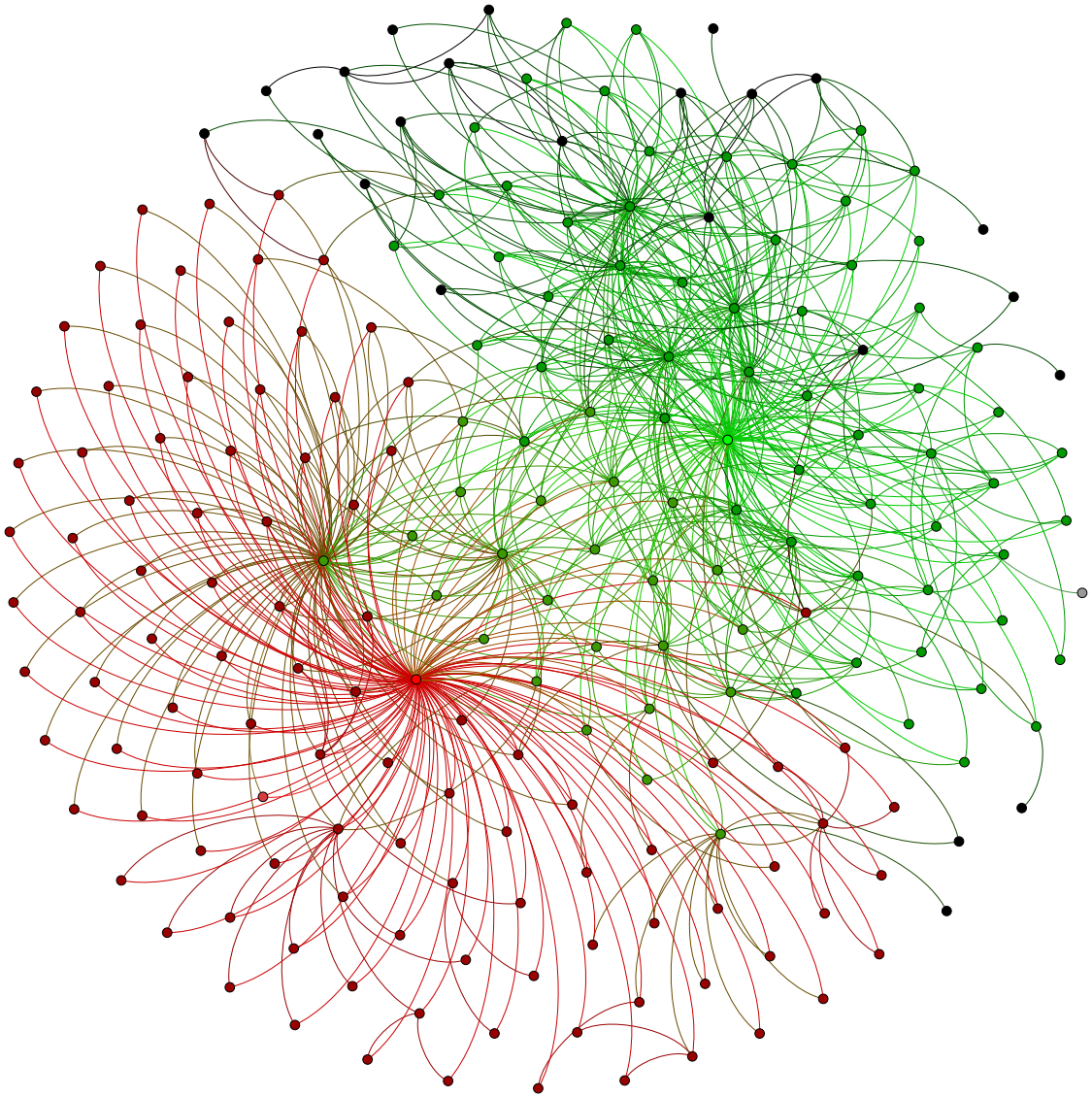
Tabelle: Funktionsaufrufe von EPrints und Kielprints

Aktive Pakete

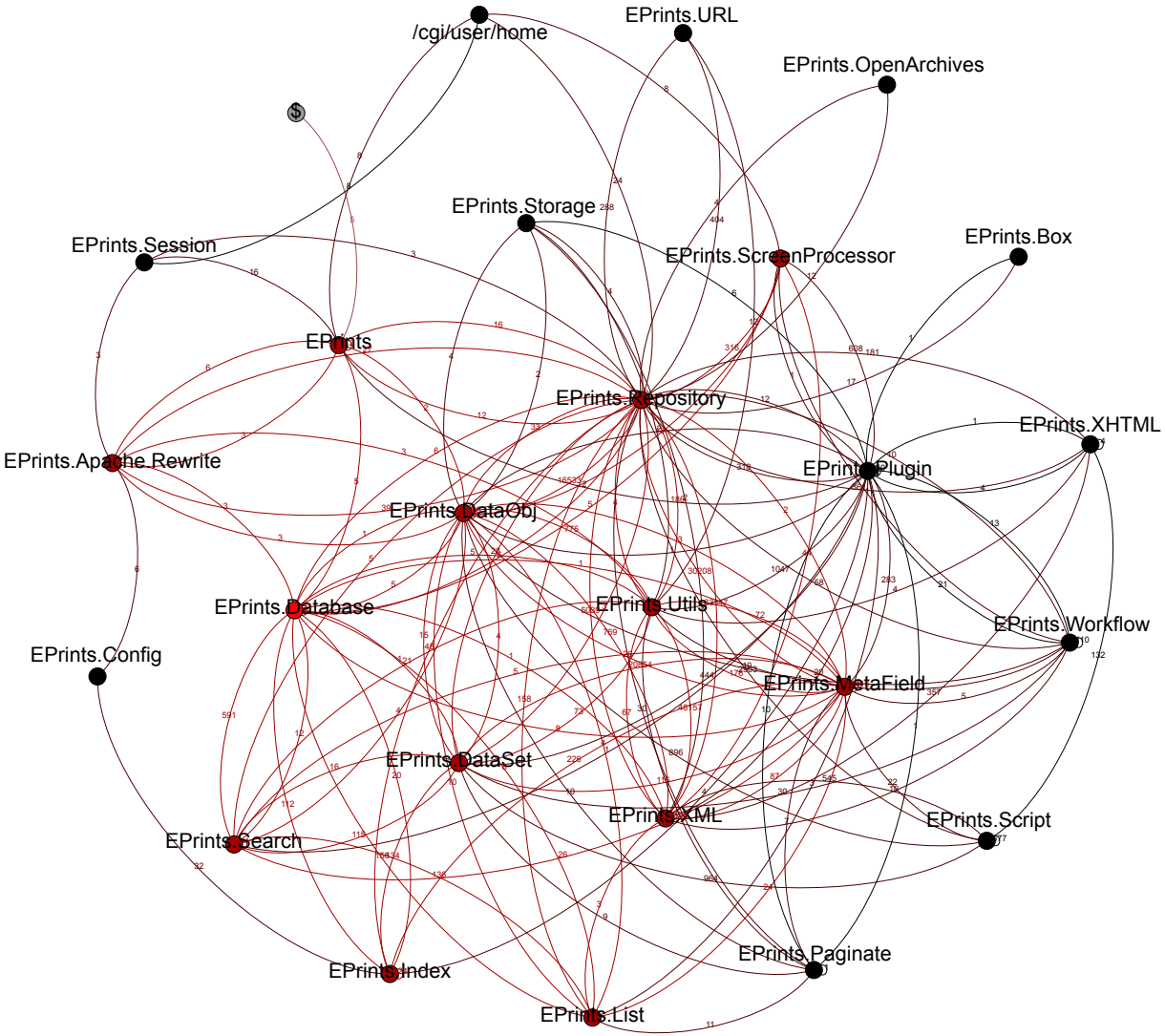
EPrints	Aufrufe	Kielprints	Aufrufe
EPrints.Script.Compiler	30304	EPrints.MetaField	501872
EPrints.Repository	25980	EPrints.DataSet	253493
EPrints.MetaField	18374	EPrints.Repository	243699
EPrints.XML	12486	EPrints.Database	189624
EPrints.DataSet	7488	EPrints.DataObj	156391
EPrints.Utils	7220	EPrints.MetaField.Id	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

Tabelle: Die 10 aktivsten Pakete

Abhängigkeitsgraph EPrints



Reduzierter Abhängigkeitsgraph



Gliederung

7 Fazit

Fazit

- Es wurde ein erstes Monitoringverfahren für Perl-Anwendungen etabliert.
- Die Perl-Data-Bridge wurde das erste Mal *produktiv* eingesetzt.
- Es wurde ein Vergleichstest von EPrints und Kielprints durchgeführt.
- Es wurden häufige Datenbankanfragen in Kielprints festgestellt.
- Es wurden Schwächen in der Architektur sowohl von EPrints und Kielprints erkannt.