Diplomarbeit:
Open Source Rapid Web Development Frameworks - Eine Untersuchung der Skalierungsstrategien

Ergebnispräsentation Kolloquium

Ralf Geschke
FOM Köln

27.04.2009

Gliederung

- Einleitung
- Vorgehensweise
- Ergebnisse
- Ausblick

Einleitung Motivation und Zielsetzung

- **„Web 2.0“**
  - Technologien
  - Konzepte
  - Aktive Teilnahme
- **Software**
  - Entwicklungsprozess
  - Dienstleistung
  - „beta“
  - Flexibilität
  - Wachstum

- **Erstellung von skalierbaren und flexiblen Web-Anwendungen mit Open Source Frameworks**

Einleitung Skalierbarkeit

- **Definition: Hinzufügen von Ressourcen führt zu einer proportionalen Leistungssteigerung des Systems**
- **vertikale Skalierbarkeit**
  - Einsatz leistungsfähigerer Komponenten
  - Vorteil: schnell, einfach
  - Nachteil: Kosten, Leistungsgrenzen
- **horizontale Skalierbarkeit**
  - Hinzufügen weiterer Einheiten
  - Auslagerung Dienste
  - Nachteil: höherer Aufwand in Applikationsebene
  - Vorteil: nahezu unbegrenzt möglich, einzig „echtes“ Verfahren

Vorgehensweise

- **Auswahl Frameworks**
  - Ruby on Rails (Ruby)
  - Django (Python)
  - Symfony (PHP)
  - Grails (Groovy / Java)
- **Anwendungsfall News-System**
- **Erstellung mit PHP und Frameworks**
  - ohne Optimierungen
  - mit Optimierungen
- **Messung der Performance**
- **Nutzwertanalyse**

Vorgehensweise Anwendung Screenshot

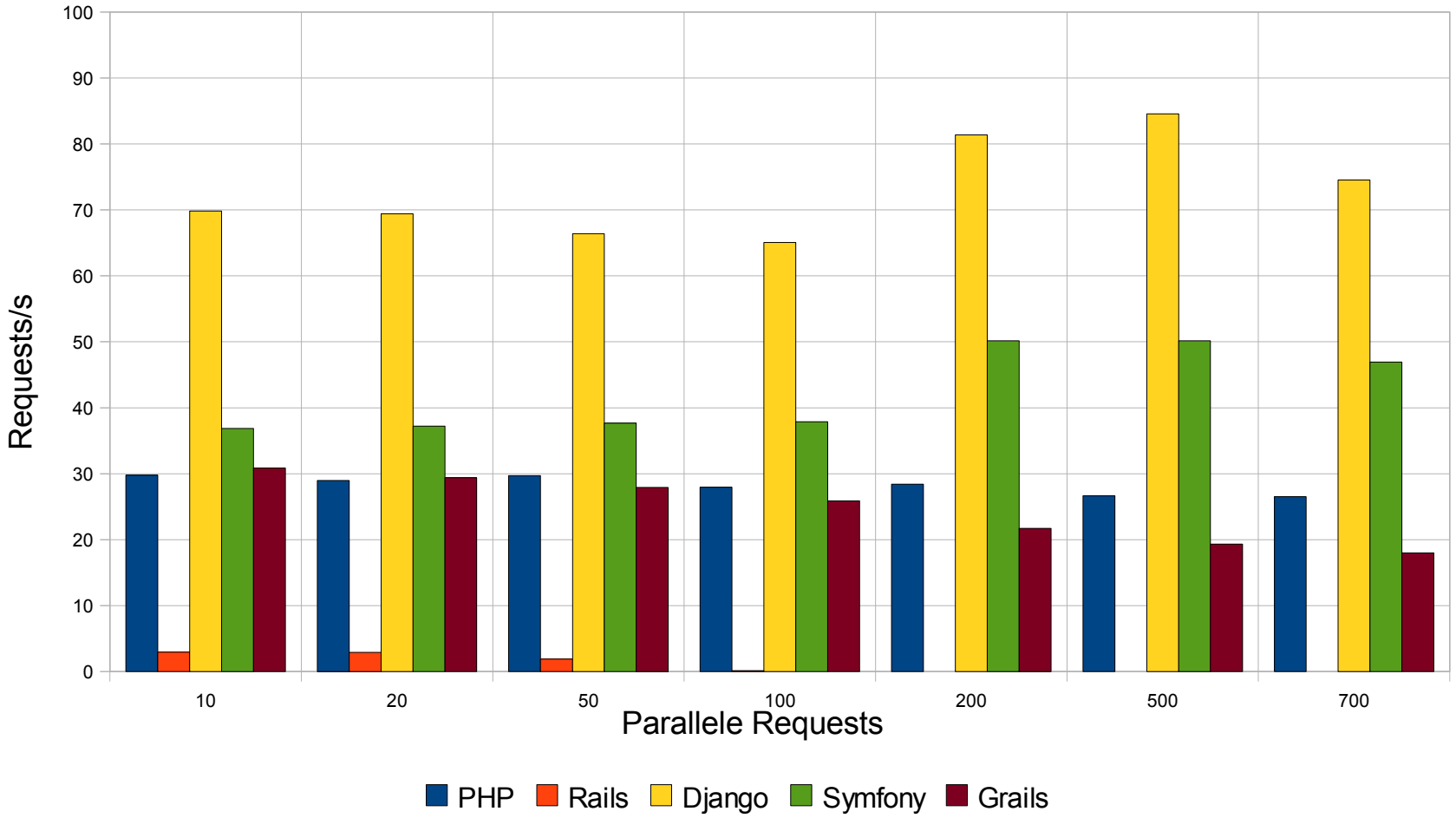
Vorgehensweise Optimierung und Messung

- **Optimierung**
  - PHP: Memcache für Listen und Datensätze, File-Cache für Seiten
  - Frameworks
    - Rails: Action-Caching
    - Django: Site-Cache
    - Symfony: Action-Caching
    - Grails: kein Caching, Optimierung ORM
- **Benchmark-Test**
  - Tool „Siege“
  - Zugriff auf Übersichtsseiten, Detailseiten, Kommentarabgabe
  - parallele Requests
  - Lastverhalten

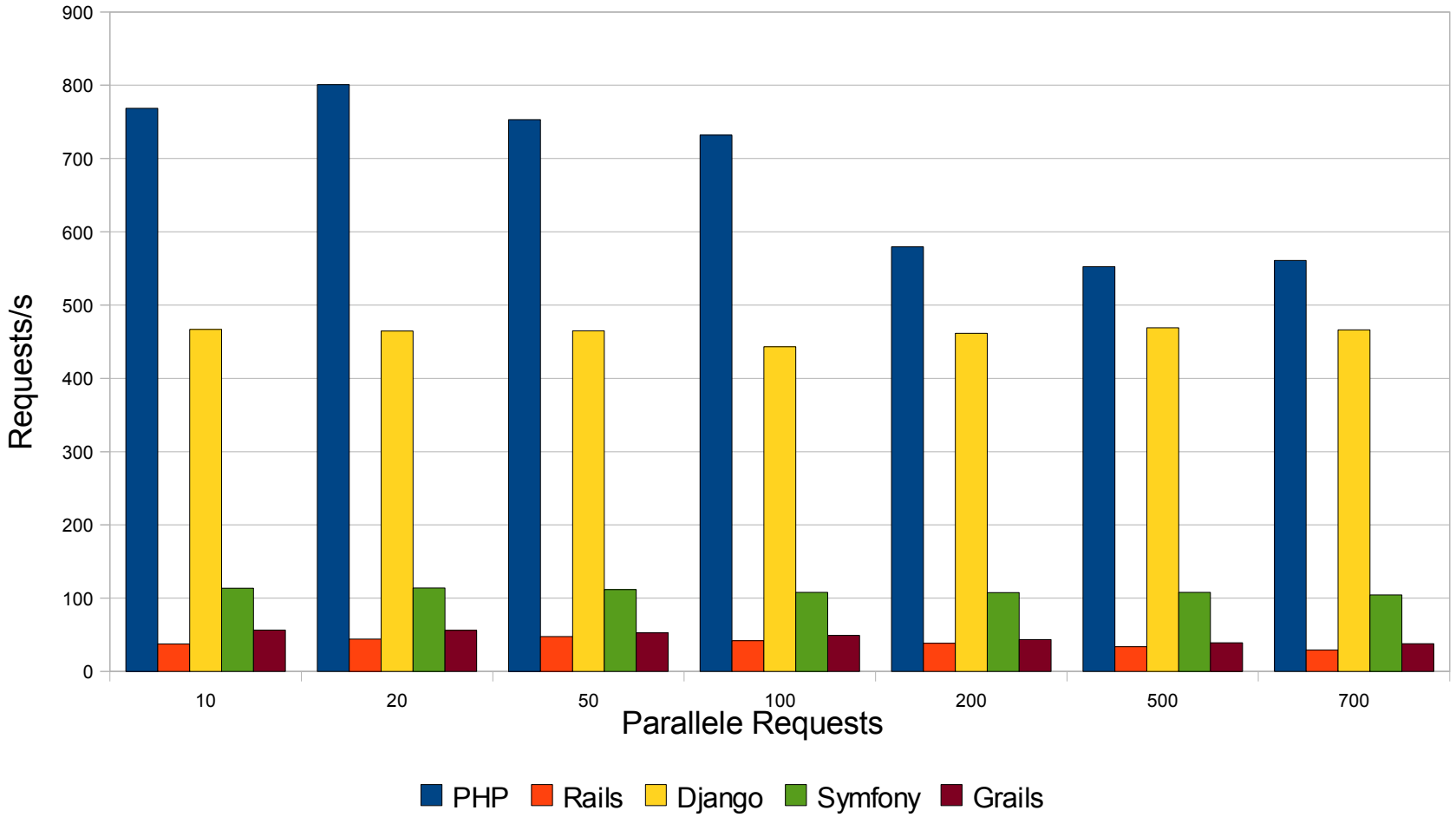
Vorgehensweise Nutzwertanalyse

- **Zielkriterien:**
  - Entwicklung
  - Funktionalität
  - Geschwindigkeit
  - Support
- **Zwei Szenarien**
  - Hohe Geschwindigkeit
  - Geringer Entwicklungsaufwand

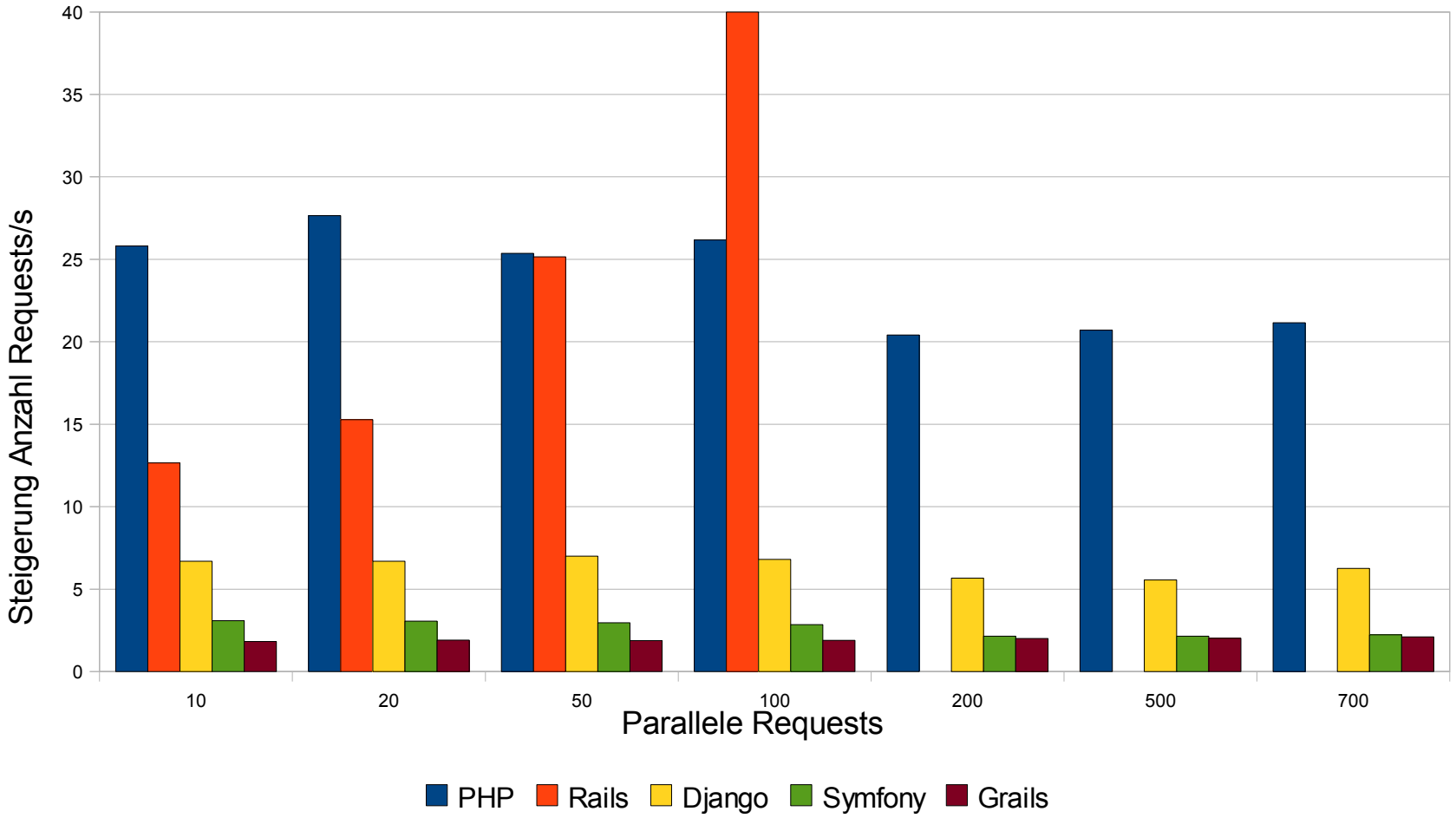
Ergebnisse nicht optimierte Anwendung



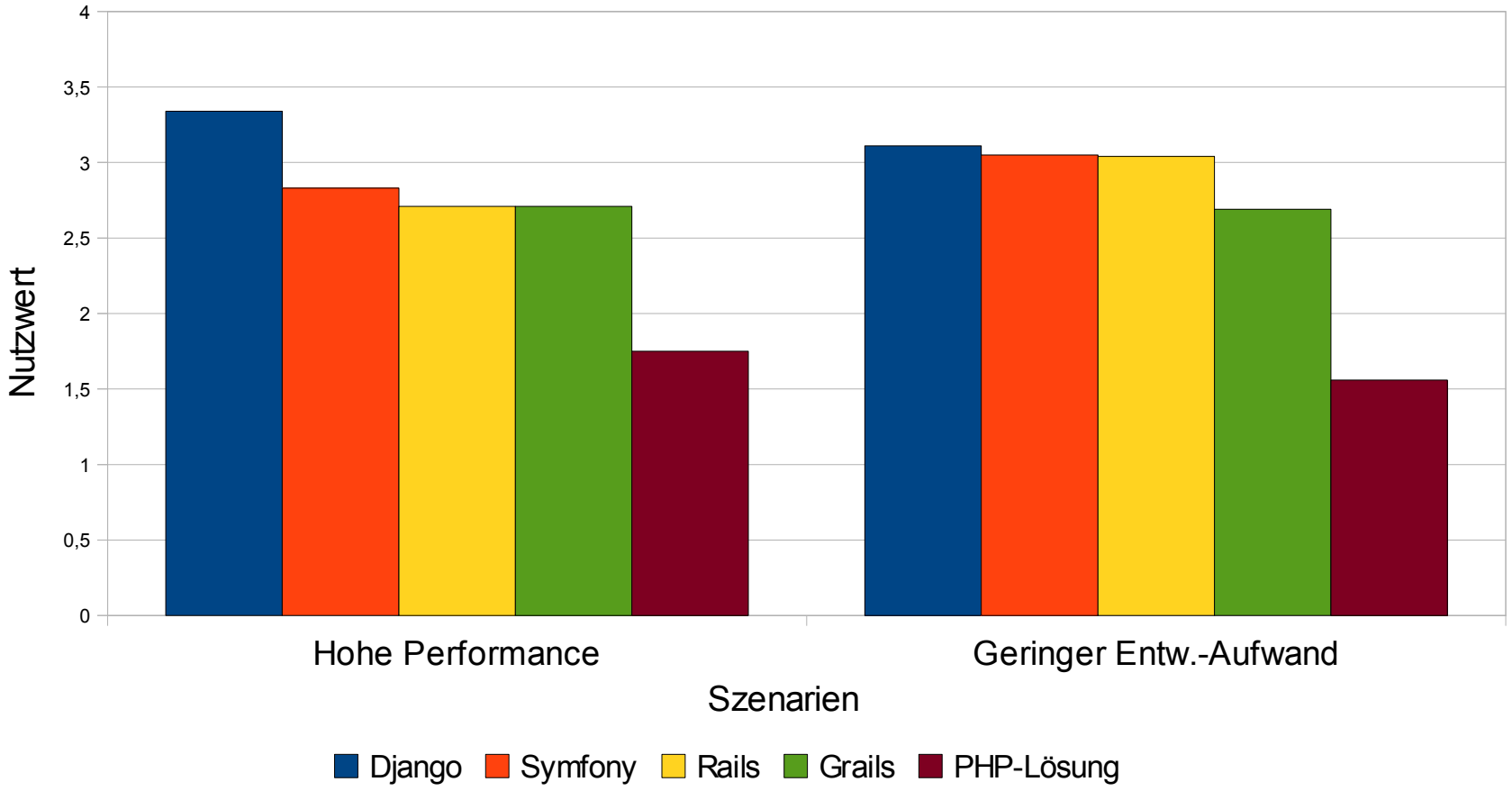
Ergebnisse optimierte Anwendung



Ergebnisse Geschwindigkeitssteigerung



Ergebnisse Nutzwertanalyse



Ergebnisse und Beurteilung

- **PHP**
  - schnellste Anwendung nach Optimierung
  - gute Strukturierung möglich
- **Rails**
  - sehr langsam
  - Entwicklung in Ordnung
- **Django**
  - schnellstes Framework
  - subjektiv: funktioniert!
- **Symfony**
  - Geschwindigkeit in Ordnung
  - Entwicklung subjektiv „holprig“
- **Grails**
  - wenige Optimierungsmöglichkeiten
  - Entwicklung gut

Ausblick Frameworks

- **Weiterentwicklung Frameworks**
  - Verschmelzung Rails und Merb: Rails 3 = Merb 2
  - PHP-Frameworks, z.B.
    - „Standard“ Zend Framework
    - SilverStripe CMS Framework
    - viele weitere...
  - Django
    - Conditional View Processing
    - Multiple Database Support (GSOC 2009)
  - Grails 1.1: Batch Fetching (Optimierung Lazy Loading)

Ausblick „Hot Topics“

- **(verteilte) Key-Value-Speichersysteme**
  - Google BigTable
  - Amazon SimpleDB
  - Open Source: CouchDB, MongoDB, Project Voldemort...
- **Asynchrone Systeme**
  - Vor-Generierung von Daten (Cronjob)
  - Message-Queues
- **Google App Engine**
  - Web Applications innerhalb Google Infrastruktur
  - Python / Java
  - Entwicklungs- und Laufzeitzumgebung
- **Web-Server: Nginx, Lighttpd**

Vielen Dank...

...für Ihre Aufmerksamkeit!