FirstSpirit™
Your Content Integration Platform

UX-Bridge Technisches Datenblatt

Version	**1.0**
State	**RELEASED**
Date	**2012-09-17**
Department	Product Management
Author/ Authors	C. Feddersen
Copyright	2012 e-Spirit AG
File name	UX-Bridge_Technical_Datasheet_DE

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-Spirit.com
www.e-Spirit.com

Inhaltsverzeichnis

1 Einführung

Das vorliegende Technische Datenblatt gilt für die aktuell freigegebene UX-Bridge Release-Version 1.0.

Es beschreibt in Kapitel 2 die benötigten Komponenten zum Betrieb des UX-Bridge-Moduls und deren Systemvoraussetzungen. Kapitel 3 gibt einige Hinweise zum Sizing, die aus den Ergebnissen der Lasttests resultieren. Diese finden Sie in Kapitel 4.

2 Systemvoraussetzungen

2.1 FirstSpirit

Das UX-Bridge-Modul wurde auf FirstSpirit 5.0 entwickelt und getestet. Somit wird FirstSpirit in der Version 5.0 empfohlen. Allerdings besteht auch Kompatibilität zu FirstSpirit 4.2R4, mit folgender Ausnahme:

- Unter FirstSpirit 4.2R4 ist der UX-Bus nicht auf dem InternalJetty des FirstSpirit-Servers lauffähig. Weitere Informationen zur Installation des UX-Bus auf einem FirstSpirit-Server finden Sie in Kapitel 2.1.2 der UX-Bridge-Installationsanleitung.

Systemvoraussetzungen zur Installation von FirstSpirit können dem Technischen Datenblatt zu FirstSpirit entnommen werden.

2.2 UX-Bus

Zum Betrieb des UX-Bus werden folgende Anforderungen an das System gestellt.

2.2.1 Hardware

- 100 MB freier Plattenplatz.
  Je nach Konfiguration (Persistenz) und Betrieb (Master Slave) kann deutlich mehr Speicherplatz benötigt werden.
- 2GB RAM

2.2.2 Betriebssystem

- Windows Server 2008 R2
- AIX V6.1+7.1
- Solaris 10+11
- Red Hat Enterprise Linux 5+6
- Debian Linux 5+6

2.2.3 Java-Umgebung

- Oracle Java 1.6.0_18 oder höher

3 Sizing

Die oben genannten Mindestanforderungen sollten für alle Szenarien ausreichend sein, in denen der UX-Bus primär für das Schreiben der Daten von FirstSpirit in ein oder mehrere Content-Repositories verwendet wird. Ein solcher Anwendungsfall wurde im Rahmen der Lasttests geprüft. Die Ergebnisse finden Sie im nächsten Kapitel.

Soll der UX-Bus als zentrale Nachrichtenkomponente innerhalb der Webseiten-Infrastruktur dienen, so sind ggf. mehr Ressourcen notwendig. Kandidaten sind Szenarien, in denen pro Request der Webapplikation eine Nachricht auf den UX-Bus geschickt wird, beispielsweise um das Userverhalten zu analysieren. Details zur Skalierung des UX-Bus finden Sie im UX-Bridge Installationshandbuch Kapitel 3.3 Hochverfügbarkeit. Für solche Szenarien sollte innerhalb des Projektes immer ein entsprechender Lasttest durchgeführt werden, um sicherzustellen, dass die Anforderung an Performance und Skalierbarkeit auch erfüllt werden.

4 Ergebnisse der Lasttests

Im Rahmen der Entwicklung der UX-Bridge wurden Lasttests durchgeführt. Zielsetzung war ein möglichst realistisches Szenario zu entwickeln, welches eine konkrete Nutzung der UX-Bridge im Rahmen einer Webseite simuliert, die sowohl dynamische als auch vorgenerierte Inhalte enthält.

4.1 Testszenario

4.1.1 Redaktionssystem

Es wird davon ausgegangen, dass mit dem FirstSpirit Client 100 Redakteure gleichzeitig arbeiten. Dabei entstehen pro Tag 1.000 Änderungen im System (neue Inhalte oder Bearbeitung von bestehenden Inhalten). Zusätzlich gibt es einen automatischen Importmechanismus von Inhalten, der ebenfalls 1.000 Änderungen pro Tag erzeugt. Alle Änderungen werden über einen Arbeitsablauf freigegeben und deployed. Die Generierung umfasst sowohl vorgenerierte HTML-Dateien, die notwendigen Assets (Grafiken, CSS, JS) als auch die notwendigen Nachrichten, um die Daten im Content-Repository zu aktualisieren.

4.1.2 Webseite

Als Webseite wurde das im Tutorial „News-Szenario“ entwickelte Projekt verwendet. Details dazu finden Sie in der Entwicklerdokumentation, Kapitel 4.2. Es handelt sich dabei um einen hybriden Internetauftritt mit dynamischen und vorgenerierten Anteilen. Im Newsbereich kommt eine dynamische Webapplikation zum Einsatz, die dem Benutzer die aktuellen Pressemitteilungen anzeigt und eine dynamische Filterung nach Kategorien ermöglicht. Die restlichen Seiten enthalten vorgeneriertes HTML.

Es wird angenommen, dass 30% der Seitenaufrufe auf die dynamische Webapplikation für die Pressemitteilungen entfallen. Die restlichen 70% sind Aufrufe von vorgenerierten Seiten. Dies erscheint realistisch, da die eigentliche Pressemitteilung eine vorgenerierte Seite ist und nur die Übersicht der Pressemitteilung dynamisch ausgeliefert wird.

Um das Benutzerverhalten so exakt wie möglich abzubilden, enthalten die Testszenarien entspreche „think times“. Diese wurden aus realen Surfverhalten extrapoliert. Die Verweildauer auf einer Seite variiert zwischen 1-4 Sekunden.

Ein simulierter Benutzer surft pro Besuch immer mehrere Seiten ab. Während eines Besuches wird ein Browsercache simuliert, so dass Designelemente, JavaScript und CSS nicht erneut angefordert werden müssen. Um das Browserverhalten nachzubilden, werden maximal 5 gleichzeitige Anfragen pro Benutzer erzeugt, um Ressourcen nachzuladen. Nach Beendigung einer solchen Sitzung wird der Cache wieder geleert.

Eine aufgerufene Webseite führt dabei im Durchschnitt zu 44 HTTP-Requests mit

einem durchschnittlichen Übertragungsvolumen von 180kb. Befinden sich die Dateien im Browsercache so reduziert sich das Übertragungsvolumen auf durchschnittlich 25kb.

Im Laufe des Testes werden bis zu 5000 gleichzeitige Benutzer auf der Webseite simuliert. Die Content-Änderungen durch das Redaktionssystem erfolgen im Test zeitgleich zu den Lasttests auf dem Webserver.

4.2 Testinfrastruktur

Um die Zahl der Benutzer einfach skalieren zu können wurde der komplette Test auf der Amazon Cloudinfrastruktur durchgeführt. Dabei wurden bewusst alle Komponenten auf dedizierte Instanzen installiert, um etwaige Flaschenhälse einfacher identifizieren zu können.

Details zu den Amazon EC2 Instantypen finden Sie unter http://aws.amazon.com/de/ec2/instance-types/. Dort finden Sie auch Hinweise welcher Hardware die Instanzen in etwa entsprechen. Eine EC2 Compute Unit bietet die entsprechende CPU-Kapazität eines 1,0-1,2 GHz Opteron oder Xeon Prozessors von 2007.

Die beteiligten Systeme waren:

FirstSpirit-Server: m1.small mit FirstSpirit 5.0.102.53034

UX-Bus: m1.small mit ActiveMQ 5.6.0

Webserver: m2.4xlarge mit nginx 1.0.15

Tomcat für Webapplikation: c1.xlarge

Tomcat für Adapter: m1.small

jMeter (für Benutzersimulation): m1.xlarge

MySQL für FirstSpirit Datenquellen: Amazon RDS Instanz vom Typ db.m1.small mit MySQL 5.1.61

MySQL Instanz als Content-Repository: Amazon RDS Instanz vom Typ db.m1.small mit MySQL 5.1.61

Jedes System (bis auf die jMeter Instanzen für die Client-Simulation) war nur einmal vorhanden, d.h. es wurde hier noch keinerlei Clustering zur Performance-

Verbesserung betrieben.

Der Webserver wurde als Proxy für den Tomcat konfiguriert. Da die dynamische Webapplikation keine personalisierten Inhalte ausliefert, kann der Webserver die vom Tomcat erhaltenen Inhalte für maximal eine Minute cachen. Findet während dieser Zeit eine Inhaltsänderung durch das Redaktionssystem statt, so wird der Cache entsprechend vorher invalidiert, so dass die Änderung sofort auf der Webseite sichtbar wird.

In einer Ramp-Up-Phase von 30 Minuten wurde die Zahl der Webseitenbesucher bis auf 5000 gleichzeitige Benutzer gesteigert. Diese Last wurde über 1 Stunde gehalten, gefolgt von einer Ramp-Down-Phase von 10 Minuten. Für Contentänderungen im Redaktionssystem gab es keine Ramp-Up- bzw. Ramp-Down-Phasen. Hier wurden eine konstante Erstellung von Inhalten sowie ein anschließendes Deployment durchgeführt.

Die Erfassung aller relevanten Messdaten erfolgte über eine Reihe von Werkzeugen und wurde zentral in einer Graphite-Instanz (siehe http://graphite.wikidot.com/) gesammelt und visualisiert.

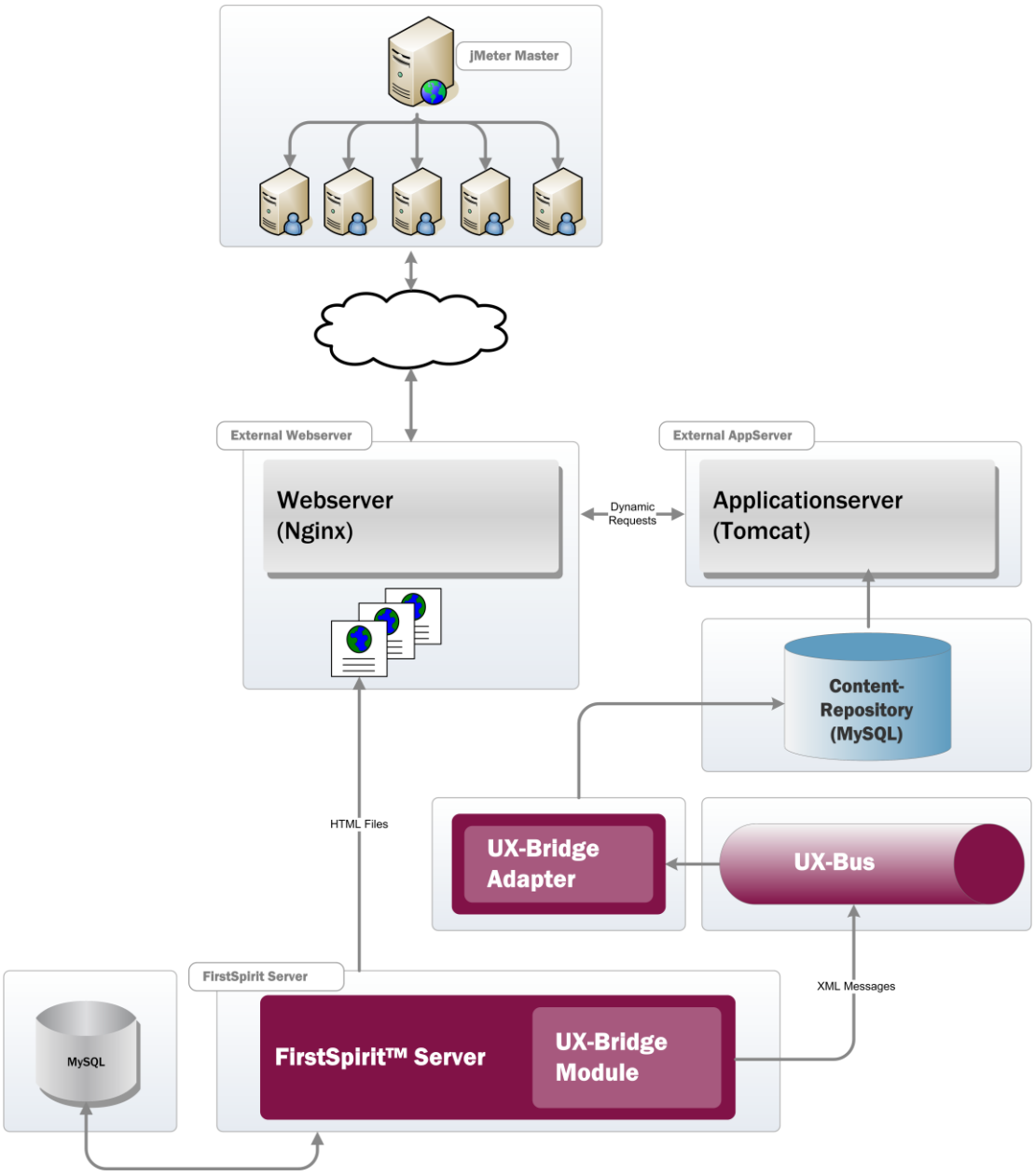


Abbildung: Infrastruktur für Lasttests

4.3 Ergebnisse

Die Auswertung der gewonnen Messdaten führte zu folgenden Erkenntnissen:

- Unter Volllast von 5000 Benutzern beantwortete der Webserver ca. 13.900 Anfragen pro Sekunde. Rechnet man diese Zahl auf einen Tag hoch, so entspricht das einem Traffic von rund 27 Millionen Seitenaufrufen pro Tag. Die Seiten, inkl. aller Ressourcen, waren dabei durchschnittlich innerhalb von 117ms auf dem Client verfügbar. Die UX-Bridge kann somit auch in

Szenarien mit hohen Zugriffszahlen eingesetzt werden.

- Auch bei einer sehr hohen Last auf der Webseite gibt es keinerlei Beeinträchtigung für die Redakteure. Die Veröffentlichungszeiten von Inhalten sind über den gesamten Testlauf hinweg konstant. Dies ist auf die strikte Trennung des Redaktionssystems von den Livesystemen zurückzuführen. Umgekehrt wirken sich häufige Content-Änderungen nicht auf die Performance der Website aus.

- Die hohe Zahl an redaktionellen Inhaltsänderungen und Veröffentlichungen stellt für die UX-Bridge, selbst in der Standardkonfiguration, kein Problem dar. Alle Änderungen waren durchschnittlich innerhalb von 357ms auf der Webseite und im Content-Repository der UX-Bridge verfügbar. In vielen Kundenprojekten werden deutlich weniger Änderungen und Veröffentlichungen pro Tag durchgeführt.

- Ist in Projekten ein Caching der Applikationsdaten nicht möglich, so empfiehlt sich eine horizontale Skalierung der Applikationsserver, da diese erfahrungsgemäß den ersten Flaschenhals bilden. Dies gilt umso mehr, wenn der Trafficanteil auf dem dynamischen Teil der Webseite höher ist als im Lasttest angenommen.

Im Anhang finden Sie die wesentlichen Kennzahlen in grafischer Form.

Y-Achse links: Einheit sind Millisekunden

Y-Achse rechts: Anfragen pro Sekunde bzw. Zahl der simulierten Benutzer

X-Achse: Zeit

4.3.1 UX-Bridge deploy time

Dieser Graph stellt die benötige Zeit dar, um eine inhaltliche Änderung in das Content-Repository zu schreiben. Gemessen wurde die Zeit zwischen der Freigabe und dem Schreiben in das Content-Repository. Es gilt die Y-Achse links.

Die Spitze ganz zu Beginn lässt sich durch den Overhead des initialen Verbindungsaufbaus der einzelnen Komponenten erklären. Es ist deutlich zu sehen, dass die Last auf den Webservern keine Auswirkung auf diese Messgröße hat.

4.3.2 Average response time (jMeter)

Durchschnittliche Antwortzeit aller Aufrufe von Webseiten (30% dynamisch, 70%

statisch). Der Wert bezieht sich auf die komplette Seite inkl. Javascript, Stylesheets und Grafiken. Es gilt die Y-Achse links.

Erst ab ca. 4000 gleichzeitigen Benutzern ist ein kleiner Anstieg der Antwortzeiten zu erkennen. Gleichzeitig ist ein Anstieg der Load auf den jMeter-Instanzen und dem Webserver zu beobachten. Dabei ist die Load bei allen Maschinen noch im akzeptablen Bereich. Zudem deuten die Zahlen des Webservers darauf hin, dass ab dieser Zahl das Connection-Handling des Webservers zu einer Verschlechterung der Antwortzeiten führt.

4.3.3 Webserver req/sec:

Zahl der HTTP-Anfragen pro Sekunde die auf dem Webserver ankommen. Es gilt die rechte Y-Achse.

Die Ausschläge unter Volllast noch oben korrelieren mit einer erhöhten Last auf den jMeter. Diese wurde durch die variablen „think times“ verursacht. Die Messdaten zeigen, dass zu diesem Zeitpunkt deutlich mehr Requests von den jMeter-Instanzen erzeugt wurden.

4.3.4 Concurrent users (jMeter)

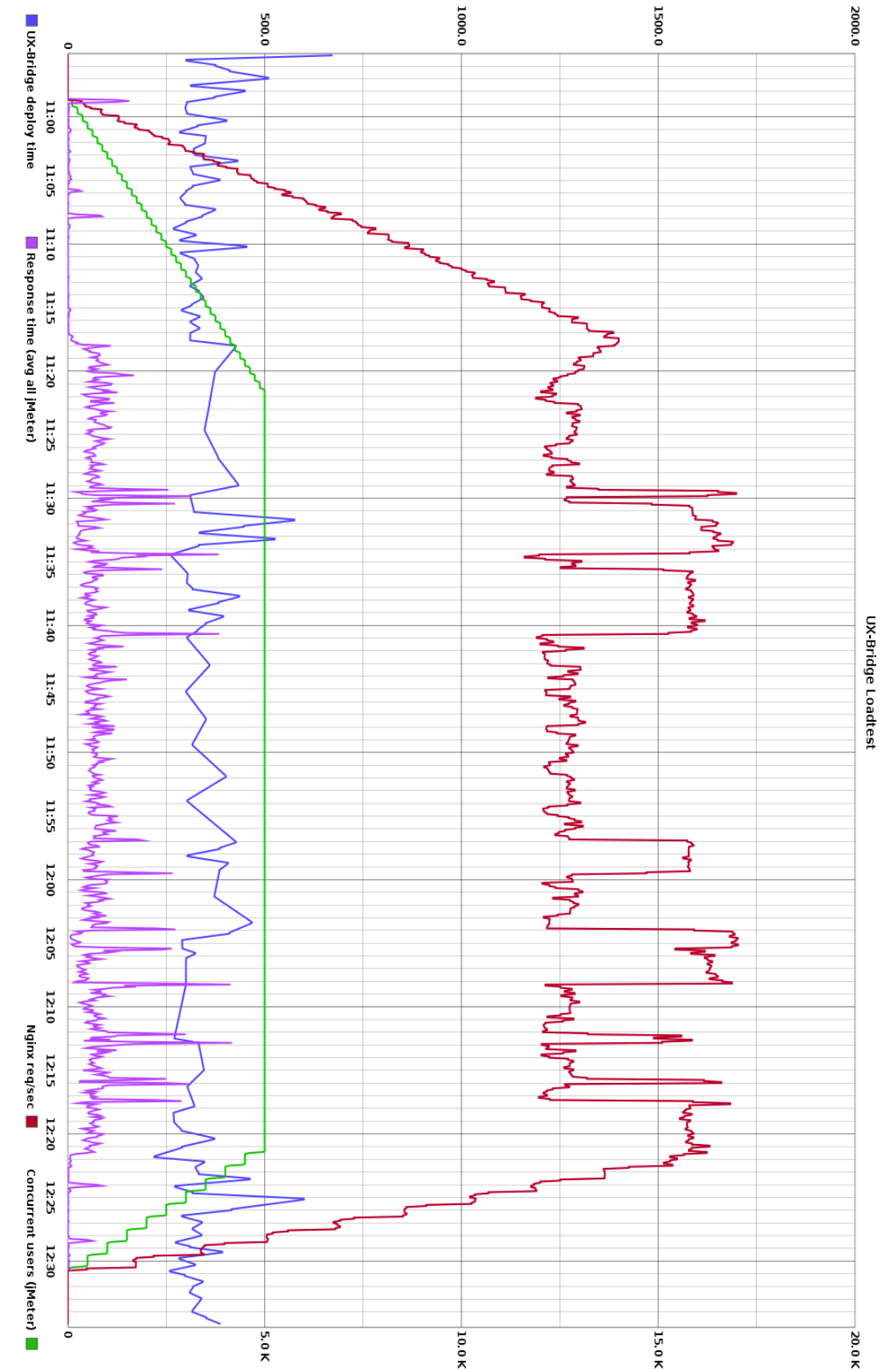
Zahl der gleichzeitigen Benutzer auf der Webseite. Es gilt die rechte Y-Achse.

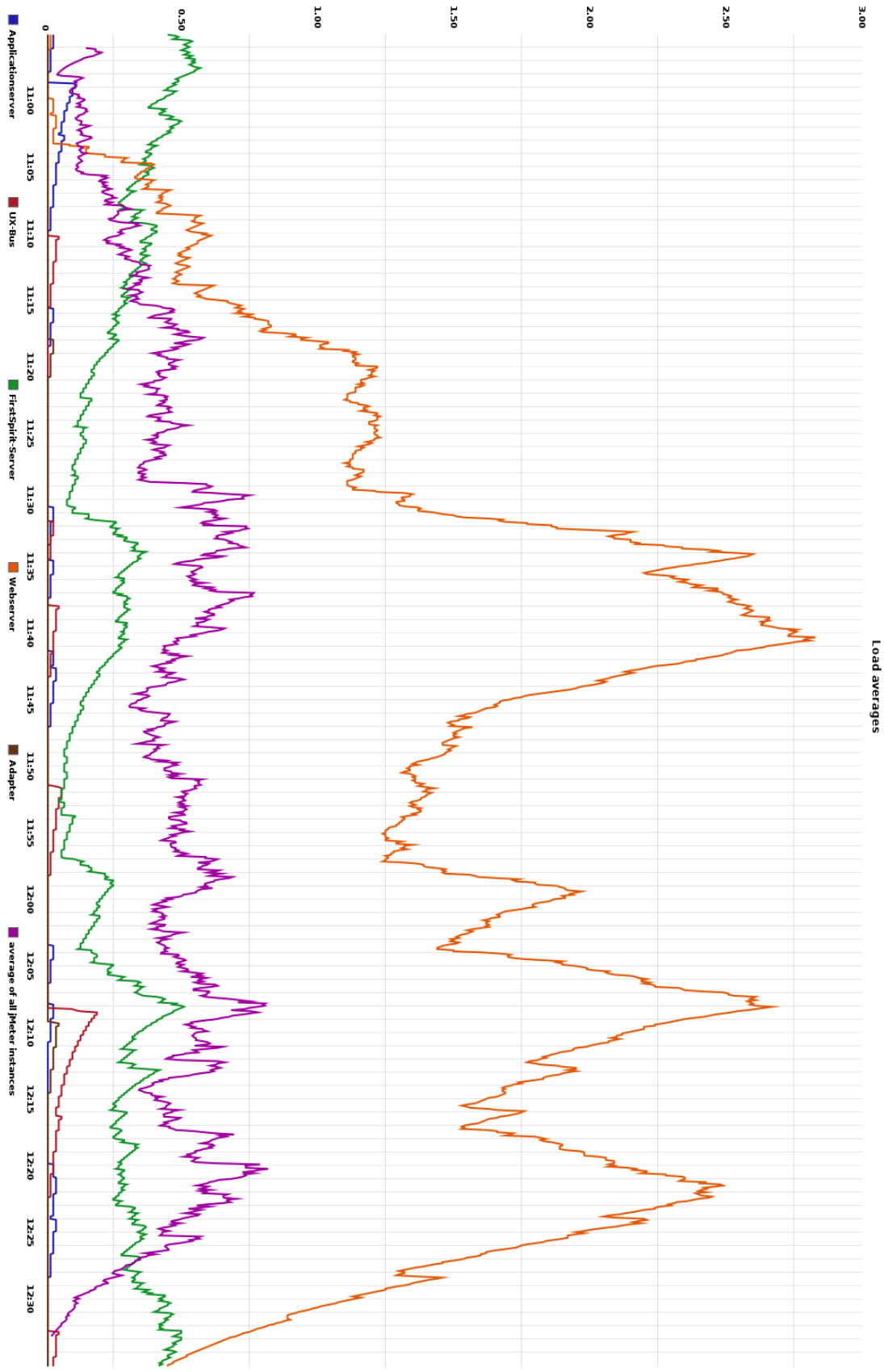
Der Graph entspricht genau dem erwarteten Verlauf. Die Ramp-Up- und Ramp-Down-Phase sind deutlich zu erkennen.

4.3.5 Last auf den beteiligten Maschinen

In der zweiten Grafik ist die Last der beteiligten Maschinen abgebildet. Keine der Maschinen wurde während des Testlaufes an seine Belastungsgrenzen gebracht. Für den FirstSpirit-Server, den Adapter und den UX-Bus stellten die Inhaltsänderungen keinerlei Problem dar. Der Applikationsserver wurde durch die Verwendung des Webservers als Proxy nur zeitweise belastet, so dass die durchschnittliche Last ebenfalls sehr gering ausfällt.

5 Anhang


UX-Bridge deploy time
Response time (avg all jMeter)
Nginx req/sec
Concurrent users (jMeter)
UX-Bridge Loadtest
0
500.0
1000.0
1500.0
2000.0
11:00
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
12:25
12:30
0
5.0 K
10.0 K
15.0 K
20.0 K


Applicationserver
UX-Bus
FirstSpirit-Server
Webserver
Adapter
average of all jMeter instances
0
0.50
1.00
1.50
2.00
2.50
3.00
11:00
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
12:25
12:30
Load averages