Universität Potsdam

Mathematisch-Naturwissenschaftliche Fakultät

Institut für Informatik

Seminararbeit

Praktikum Paralleles Rechnen

Einrichten eines Clusters zum Einspielen von Wikipedia-Traces

Sebastian Menski (734272)

menski@uni-potsdam.de

Betreuer: Prof. Dr. Bettina Schnor

M.Sc. Jörg Zinke

Potsdam, 26. Juli 2011

Inhaltsverzeichnis

1 Einleitung

Um Webserver und Server Load Balancer zu testen werden verschiedenste Benchmarks verwendet. Oft nutzen diese Benchmarks synthetisch erzeugte Requests, im Falle von Webservern auch synthetische Websites. Dies hat zum Nachteil, dass kein realistisches Nutzerverhalten simuliert werden kann. Somit fehlt die Verbindung zu realistischen Lastsituationen. Zum Testen von existieren Webservern können realistische, aufgezeichnete Traces genutzt und modifiziert werden um realistischere Lasttests zu generieren. Für das Testen von Server Load Balancer bleibt weiterhin das Problem, dass auf existierende Benchmarks und Webanwendungen zurückgegriffen werden muss.

Am Lehrstuhl Betriebssysteme und Verteilte Systeme von Frau Prof. Dr. Schnor schreibt M.Sc. Jörg Zinke seine Doktorarbeit zum Thema Server Load Balancing. Um die entwickelten Algorithmen anhand eines realistischen Benchmarks bzw. einer realistischen Webanwendung zu testen, soll die englische Wikipedia simuliert werden. Zum Testen stehen außerdem reale Webserver-Traces der Wikipedia zur Verfügung.

Als nächstes wird die Aufgabenstellung beschrieben. In Abschnitt 2 werden die nötigen Grundlagen für die Lösung der Aufgabe diskutiert und erläutert. Daraus ergibt sich das Konzept, welches in Abschnitt 3 vorgestellt wird. Dessen Implementation wird in Abschnitt 4 erläutert. Die in Abschnitt 5 präsentierten Daten, sollen die Zusammensetzung der Traces und die Wichtigkeit der Bild-Dateien, für das erfolgreiche Einspielen von Traces, verdeutlichen. Abschließend wird in Abschnitt 6 das Ergebnis dieser Arbeit zusammengefasst und in Abschnitt 7 ein Ausblick auf die Weiterentwicklung der hier vorgestellten Lösung präsentiert.

1.1 Aufgabenstellung

Ziel dieses Praktikums war es, eine Anwendung bzw. ein Skript zu entwickeln, welches es ermöglicht ein Webserver-Cluster mit einer Wikipedia-Umgebung einzurichten. Dazu wird ein gegebener Wikipedia-Dump genutzt und ein bestimmter Teil eines Wikipedia-Traces. Nach dem Einrichten sollen alle Server, die Artikel der englischen Wikipedia (entsprechende des Dumps) anbieten. Hinzu kommen die im benutzten Trace-Abschnitt verwendeten Bilddateien. Anschließend soll bei einem Wiedereinspielen des Trace-Abschnitts der Anteil der nicht gefundenen Ressourcen minimal sein. Daher ergeben sich folgende Unteraufgaben:

1. Filtern des Traces
2. Ermitteln und Bereitstellen der benötigten Ressourcen
3. Verteilen und Einrichten der präparierten Umgebung auf dem Cluster

2 Grundlagen

Um eine Instanz der Wikipedia bereitzustellen benötigt man mehrere Komponenten. Als Basis benötigt man einen Webserver (empfohlen Apache httpd[1]), PHP (empfohlen ≥ 5.0) und MySQL (empfohlen ≥ 4.0). Als weitere Software wird Mediawiki[2] benötigt, welches die Wikisoftware der Wikipedia ist. Nach der Installation dieser Komponenten ist ein leeres Wiki eingerichtet. Um daraus eine Instanz der Wikipedia zu machen, benötigt man eine Dump der Wikipedia. Ein Dump bezeichnet hier die Sicherung der Datenbanken der Mediawiki-Installation. Dabei ist zu beachten das jede Sprache in der Wikipedia, wiederum ein eigenes Wiki ist (z.B. `http://en.wikipedia.org/` oder `http://de.wikipedia.org/`). Somit benötigt man den Dump einer bestimmten Wikipedia, diese Dumps werden regelmäßig von der Wikimedia Foundation erstellt und zum Download angeboten[3]. Leider enthalten diese Dumps wie bereits erwähnt nur die Sicherungen der Datenbanken und aus rechtlichen Gründen nicht die in den Wikipedia-Artikeln genutzten Bilder. Aber mit Hilfe eines Dumps und den vom Mediawiki bereitgestellten Skripten können zu mindestens die Wikipedia-Artikel in die Datenbank eingelesen werden.

2.1 WikiBench

Die für diese Arbeit genutzten Dumps und Traces stammen von der Internetseite `http://www.wikibench.eu/`. Bei WikiBench handelt es sich um ein Webanwendungs-Benchmark, welcher einen Dump der englischen Wikipedia und reale Traces der Wikipedia nutzt. WikiBench wurde im Laufe einer Abschlussarbeit [1] an der VU Universität Amsterdam entwickelt. Die genutzten Traces wurden zudem in dem Paper [3] analysiert.

Der Dump enthält über 6 Millionen Artikel der englischen Wikipedia. Die Traces enthalten 25.6 Milliarden HTTP-Requests in der Zeit vom 19. September 2007 bis zum 2. Januar 2008. Jeder Request besteht aus einer eindeutigen ID, einem Zeitstempel, einer URL und einem Feld, welches anzeigt ob es sich um eine Operation zum Speichern von Daten gehandelt hat (entweder „save“ oder „-“). Beispiele (zur besseren Darstellung teilweise gekürzt):

```
5399461277 1194892944.297 http://de.wikipedia.org/wiki/Potsdam -
5[...] 1194892392.420 http://en.wikipedia.org/w/index.php?[...]&action=submit save
```

Die Requests sind vollständig anonymisiert und entsprechen nur 10% der tatsächlichen Requests in diesem Zeitraum. Rund 43% der Seitenanfragen richten sich an die englische Wikipedia, wobei insgesamt 32% der Requests auf Bilder oder Thumbnails verweisen (vgl.

[1] `http://httpd.apache.org/`
[2] `http://www.mediawiki.org/`
[3] `http://dumps.wikimedia.org/`

[3]). Dieser Anteil zeigt, dass möglichst viele Bilder und Thumbnails bereitgestellt werden müssen, um ein sinnvolles Wiedereinspielen zu ermöglichen.

2.2 Leistungsanalyse: Messen, Modellieren und Simulation

In der Lehrveranstalltung „Leistungsanalyse: Messen, Modellieren und Simulation“ (Sommersemester 2010) hatte Fabian Hahn, das Thema „Server load balancing: Wikipedia“ [2]. Dabei war es seine Aufgabe den Server Load Balancer Perlbal mit Hilfe des Benchmarks httperf zu testen. Dazu nutzte er den Wikipedia-Dump und die Wikipedia-Traces von WikiBench. Die für diese Arbeit wichtigen Schlussfolgerungen aus seiner Arbeit waren, dass das Einspielen des Dumps sehr zeitintensiv und instabil ist, die Bilder nicht im Dump enthalten sind und nachgeladen werden müssen und die in den Wikipedia-Artikeln enthaltenen Thumbnails dynamisch generiert werden. Zum Nachladen der Bilder existiert im Internet ein Crawler[4], welcher den Dump nach Bilderverweisen durchsucht und diese versucht herunterzuladen. Dabei ist zu beachten, dass er keine Thumbnails herunterlädt. Trotzdem ergibt sich das Problem, dass die Thumbnails direkt in den Traces aufgerufen werden und es nicht garantiert ist, dass der entsprechende Wikipedia-Artikel vorher im Trace-Verlauf enthalten ist. Somit müssten zu erst alle Thumbnails generiert werden, bevor der Trace eingespielt werden kann. Eine simple Strategie dafür wäre ein Aufruf aller enthaltenen Wikipedia-Artikel, was aber sequentiell nicht effektiv handhabbar ist.

2.3 Erfahrungen Praktikum Paralleles Rechnen

Zu Beginn des Praktikums lauteten die Aufgabenpunkte, für diese Arbeit:

1. Die Wikipedia Instanz sollte auf mehrere Server eines Clusters verteilt werden.
2. Die Server sollten durch einen Load Balancer verbunden werden.
3. Es sollte eine parallele Anwendung entwickelt werden, welche alle Wikipedia-Artikel aufruft.

Durch dieses Vorgehen sollten alle nötigen Thumbnails für die Traces generiert werden. Nach dem erfolgreichen Aufsetzen aller Wikipedia-Instanzen und der Entwicklung einer parallelen Anwendung zum Abrufen der Wikipedia-Artikel, scheiterte diese Vorhaben jedoch an mehreren Faktoren. In erster Linie scheiterte es an der riesigen Datenmenge, welche durch die große Anzahl an Wikipedia-Artikel und Bildern erzeugt wurde. Die zu dieser Zeit genutzte Hardware stieß dabei an ihre Speichergrenzen. Hinzu kamen Probleme bei der Thumbnail-Generierung und Hardware-Defekte. Als Lösung des Speicherplatz-Problems bot sich an, nicht die gesamten Bilder und Thumbnails zu verwenden, sondern

[4] `http://meta.wikimedia.org/wiki/Wikix`

nur die benötigten Ressourcen für einen Abschnitt aus den Traces. Weiterhin bot es sich an gezielt die in diesem Trace-Abschnitt verwendeten Bilder und Thumbnails direkt von der aktuellen Wikipedia herunterzuladen. Durch diese neue Herangehensweise entstand die im Abschnitt 1.1 beschriebene Aufgabenstellung.

3 Konzept

Das Konzept der hier vorgestellten Lösung besteht aus 4 Teilschritten. Als Eingabe wird ein Trace, ein Zeitabschnitt und eine Liste an zu filternden URLs benötigt.

1. **Analyse (optional):** Der erste Schritt ist optional und dient nur der Informationsgewinnung. Hierbei werden die Request des angegeben Traces nach Host-Adressen und Bild- bzw. Thumbnail-Vorkommen analysiert. Außerdem werden optional die Requests pro Sekunde grafisch dargestellt.

2. **Filterung:** Im zweiten Schritt, wird der gegeben Trace gefiltert. Dazu werden nur Requests weiterverarbeitet, die in dem angegeben Zeitabschnitt liegen. Außerdem werden nur Request beachtet bei denen das letzte Feld der Requestzeile (Speicherung von Daten; siehe 2.1) ein „-“ enthält. Also keine Speicherung von Daten durchgeführt wurde. Da es sich sonst vermutlich um `POST` Request handelte, welche sich nicht zum Wiedereinspielen eignen, da der Request-Body fehlt. Abschließend werden nur die Requests gespeichert, welche den zu filternden URLs entsprechen. Damit können die Traces auf den existieren Dump angepasst werden. So werden in diesem konkreten Beispiel nur Requests auf die englische Wikipedia und die damit verbunden Bilder-Ressourcen gefiltert.

3. **Download:** In diesem Schritt wird nun der gefilterte Trace ausgewertet und enthaltene Request für Bilder oder Thumbnails verarbeitet. Dazu wird zuerst geprüft ob die entsprechende Ressource bereits in einem früheren Durchlauf heruntergeladen wurden, wenn nicht wird versucht sie herunterzuladen. Dazu wird die im Trace angegebene URL genutzt. Ist die Ressource unter dieser URL nicht mehr verfügbar wurde sie seit Erstellung des Traces, entweder verschoben bzw. umbenannt oder gelöscht. Leider ist es dann nicht möglich, diese Ressource lokal bereitzustellen. Nachdem alle verfügbaren Ressourcen heruntergeladen wurden, müssen die Bilder der Wikipedia Datenbank bekannt gemacht werden. Dafür existiert ein PHP Skript im Mediawiki, welches Metadaten der Bilder in der Datenbank abspeichert. Dieses Skript ist sehr langsam und speicherintensiv, allerdings existiert leider keine Alternative.

4. **Installation:** Nachdem alle Ressourcen heruntergeladen wurden und die Datenbank aktualisiert wurde, können diese auf die anderen Server im Cluster transferiert werden und dort lokal eingerichtet werden. Anschließend sind alle Server des Clusters mit der selben Wikipedia Instanz ausgestattet.

Bis auf Schritt 1 sind alle Schritte erforderlich und benötigen mindestens einen Durchlauf des vorherigen Schrittes. Das heißt Schritt 3 kann erst ausgeführt werden, wenn Schritt

2 mindestens einmal ausgeführt wurde. Dann kann Schritt 3 aber beliebig wiederholt werden, ohne das Schritt 2 noch einmal ausgeführt werden muss. Das gleiche gilt für Schritt 4 bezüglich Schritt 3.

Ein weitere Punkt des Konzepts ist es, alle Schritte in einer Anwendung zu vereinen und diese Anwendung über eine Konfigurationsdatei zu steuern. Die Verwendung von Kommandozeilenoptionen schien nicht ratsam auf Grund der Fülle an Einstellungsmöglichkeiten und benötigten Parametern. Des weiteren, sollte die verfügbare CPU möglichst effizient ausgenutzt werden, weshalb eine Parallelisierung der Anwendung sinnvoll ist.

4 Implementierung

In diesem Abschnitt wird die Implementierung des in Abschnitt 3 vorgestellten Konzepts besprochen. Dazu werden als erstes die Voraussetzungen für das Ausführen der Anwendung und die Einrichtung der Server beschrieben (siehe Abschnitt 4.1). Anschließend wird das Entwickelte Skript vorgestellt (siehe Abschnitt 4.2) und die dazu gehörige Konfigurations-Datei erläutert (siehe Abschnitt 4.2.2). Zum Abschluss wird der Ablauf des Skripts erläutert (siehe Abschnitt 4.2.3).

4.1 Voraussetzungen

Wie bereits teilweise in Abschnitt 2 beschrieben, benötigt man auf einem Webserver, der eine Wikipedia-Instanz bereitstellen soll, folgende Software:

- Webserver (empfohlen Apache httpd[5])
- PHP (empfohlen ≥ 5.0)
- MySQL (empfohlen ≥ 4.0)
- ImageMagick (empfohlen zur Generierung von Thumbnails)

Außerdem wird auf dem Rechner, auf dem das Setup-Skript ausgeführt wird, mindestens Python 2.6 benötigt und gegebenenfalls `gnuplot`. Zudem muss ein Mediawiki[6] installiert und der Wikipedia Dump in die Datenbank eingespielt sein. Auf den Servern, auf denen die Wikipedia Umgebung installiert werden soll, wird mindestens Python 2.4 benötigt.

In diesem konkreten Fall wurde die Mediawiki Version 1.16.5 genutzt und der Dump der Wikipedia Artikel zuerst mit dem Programm `xml2sql`[7] in SQL transformiert und dann mit dem Programm `mysqlimport` in die Datenbank importiert. Dieses Verfahren wird offiziell nicht empfohlen, aber es stellte sich als schneller und sicherer heraus als die Nutzung des PHP Skripts vom Mediawiki.

Anschließend sollte die Datenbank in diesem Zustand gesichert werden, da dieser Stand der Datenbank wieder benötigt wird, falls ein anderer Abschnitt des Traces vorbereitet werden soll. Dazu sollte das MySQL Verzeichnis (z.B. `/var/lib/mysql/`) per `tar` gepackt werden und gegebenenfalls noch komprimiert werden per `gzip` oder `bzip2`. Dieses Archiv wird später bei der Konfiguration des Setup-Skripts benötigt.

Nach der Installation vom Mediawiki (der Ordner des Mediawiki sollte „w/“ genannt werden) muss die Datei `LocalSettings.php` noch angepasst werden. Dazu sollten folgende Zeilen ergänzt werden:

[5] `http://httpd.apache.org/`
[6] `http://www.mediawiki.org/`
[7] `http://meta.wikimedia.org/wiki/Xml2sql/`

```
$wgArticlePath      = '/wiki/$1';
$wgScriptPath       = "/w";

$wgMaxShellMemory   = "unlimited";
$wgMaxShellFileSize = "unlimited";

require_once( "$IP/extensions/Cite/Cite.php" );
require_once( "$IP/extensions/ImageMap/ImageMap.php" );
require_once( "$IP/extensions/OggHandler/OggHandler.php" );
require_once( "$IP/extensions/ParserFunctions/ParserFunctions.php" );
```

Die angegeben Extensions müssen dann noch heruntergeladen werden und in dem `extensions` Ordner hinterlegt werden. Dies ist z.B. über das SVN-Repository[8] des Mediwiki möglich. Damit der Apache httpd Server mit den verschieden URL-Formen der Wikipedia zurecht kommt, sollte noch folgende Eintrag in der httpd-Konfigurationsdatei (welche z.B. unter `/etc/httpd/conf/httpd.conf` zu finden ist) ergänzt werden (wobei der Pfad zur `index.php` anzupassen ist):

```
Alias /wiki "/var/www/html/w/index.php"
```

Es gibt noch weitere Möglichkeiten die Unterstützung der Short-URLs von Wikipedia zu erreichen, diese sind in dem Mediawiki Wiki[9] nachzulesen.

Bei der Verwendung von RedHat basierten Systemen (wie z.B. CentOS) ist zu beachten, dass in dem Paket `gnome-vfs2` bis zur Version 2.16.2-8.el5 ein Bug[10] die korrekte Ausführung von ImageMagick verhindert. Dadurch konnten keine Thumbnails für SVG Dateien erstellt werden. Daher ist es empfehlenswerte mindestens die genannte Version des Pakets zu installieren.

4.2 Setup-Skript

Die entwickelte Anwendung ist ein Python 2.6 Skript `setup_env.py` und nutzt das ebenfalls selbst entwickelte Modul `ppr`. Der Sourcecode liegt der Ausarbeitung bei oder kann aus einem GIT-Repository[11] heruntergeladen werden.

In Abbildung 1 wird das Klassendiagramm des `ppr` Moduls gezeigt. Die Basisklasse `Process` erbt von der Klasse `multiprocessing.Process`. Daran ist zu erkennen das fast jede Klasse ebenfalls ein eigener Prozess ist. Somit können verschieden Aufgaben in dem Skript parallel durchgeführt werden. Die Kommunikation zwischen den Prozessen wird

[8] `http://svn.wikimedia.org/svnroot/mediawiki/tags/REL1_16_5/extensions/`
[9] `http://www.mediawiki.org/wiki/Manual:Short_URL`
[10] `http://rhn.redhat.com/errata/RHBA-2011-0441.html`
[11] `https://github.com/menski/ppr-s11`

über sogenannte `Pipes` realisiert, wodurch Daten nicht lange im Speicher gehalten werden müssen.

4.2.1 Klassen

Process: Basisklasse, welche von `multiprocessing.Process` erbt und eine Logger bereitstellt.

SyncClient: Ist eine Klasse, die genutzt wird um eine Wikipedia Instanz auf einen anderen Rechner zu kopieren.

FileReader: Diese Klasse liest eine Datei ein und sendet den Inhalt zur Verarbeitung an eine Menge von `Pipes`. Dabei ist es möglich die Funktion zum Öffnen der Datei anzugeben, um z.B. komprimierte Dateien zu öffnen.

PipeReader: Eine Klasse, welche eine `Pipe` nach Außen anbietet, über die sie mit Daten versorgt werden kann.

FileWriter: Erbt von der Klasse `PipeReader` und speichert die übergeben Daten in einer Datei. Wiederum ist es möglich eine Funktion zum Öffnen der Datei anzugeben, um z.B. komprimierten Dateien zu schreiben.

TraceAnalyser/WikiAnalyser: Spezielle `PipeReader` zur Analyse von Traces und zum Erstellen von Statistiken und Grafiken.

TraceFilter/WikiFilter: Klassen zum Filtern von Traces, welche ebenfalls von `PipeReader` erben.

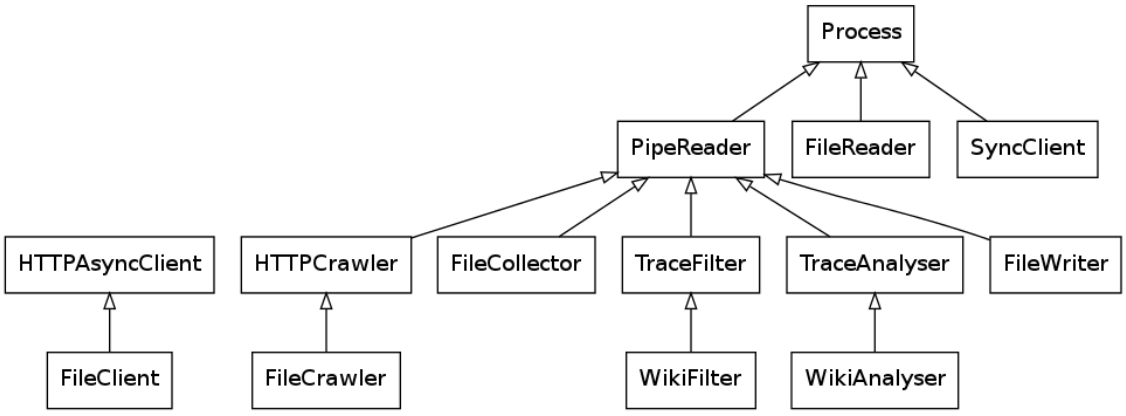

Abbildung 1: Klassendiagramm des `ppr`-Moduls

HTTPAsyncClient/FileClient: Klassen, welche das Python Modul `asynchat` nutzen um HTTP1.1 Requests zu versenden. Die Klasse `FileClient` kann außerdem den Response in einer Datei speichern.

HTTPCrawler/FileCrawler: Klassen, die unter Nutzung der `HTTPAsyncClient/FileClient` Klassen und dem Python Modul `asyncore`, asynchrone HTTP Request versenden.

FileCollector: Eine Unterklasse von `PipeReader`, welche überprüft ob eine Datei bereits im Dateisystem existiert und falls nicht diese mit Hilfe der `FileCrawler` Klasse versucht herunterzuladen.

4.2.2 Konfigurations-Datei

Ein vollständige Konfigurations-Datei ist im Anhang A aufgeführt. In diesem Abschnitt sollen die einzelnen Bereiche der Konfigurations-Datei erläutert werden.

[general] In dem `general` Abschnitt können die einzelnen Schritte des Skritps aktiviert bzw. deaktiviert werden (siehe dazu Abschnitt 3). Außerdem kann das Log-Level gewählt werden und festgelegt werden ob bei der Analyse eine Grafik für die Requests pro Sekunde erstellt werden soll (dazu ist `gnuplot` nötig).

[trace] Der folgende `trace` Abschnitt gibt den Pfad zur Trace-Datei an und ob sie mit `gzip` komprimiert ist.

[filter] Im `filter` Abschnitt wird das Zeitintervall definiert, nach dem der Trace gefiltert wird. Dabei wird das Format $a : b$ genutzt und auf die folgende Weise interpretiert: a ist ein UNIX-Zeitstempel, wenn $a > b$ gilt wird b als Anzahl von Sekunden interpretiert und auf a addiert, sonst wird b ebenfalls als UNIX-Zeitstempel interpretiert. Als nächstes wird eine Hostname bestimmt, der genutzt wird um einen modifizierten Trace zu erstellen, welcher später zum Wiedereinspielen genutzt werden kann. Um nach bestimmten URLs zu filtern kann ein regulärer Ausdruck angegeben werden. Abschließend kann bestimmt werden, ob die gefilterten Traces mit `gzip` komprimiert gespeichert werden.

[download] Der `download` Abschnitt definiert das Verhalten beim Herunterladen von Bilder. Dazu wird ein Pfad angegeben, an dem nach bereits heruntergeladen Bilder gesucht wird und ebenfalls die neue heruntergeladenen Bilder gespeichert werden. Weiterhin kann der Port angegeben werden, über den die Requests versendet werden, um neue Bilder herunterzuladen. Die `async` Option gibt die parallel aufgebauten, asynchronen Verbindungen an. Des weiteren, werden die Pfade zu der Mediawiki- und MySQL-Installation, sowie der Name des lokal laufenden MySQL-Services benötigt. Mit den Optionen `clean_images`

und `clean_mysql` kann bestimmt werden ob die aktuell in der Mediawiki-Installation vorhanden Bilder anfangs gelöscht werden sollen, bzw. ob die Datenbank vor dem einlesen der Bilder gelöscht werden soll und durch eine „saubere“ Datenbank ersetzt werden soll. Diese „saubere“ Datenbank wird über die Option `mysql_archive` bestimmt. Die nach dem erfolgreichen Einrichten gepackten Mediawiki- und MySQL-Archive werden an dem Pfad gespeichert, der mit der Option `output_dir` bestimmt wird.

[install] Der `install` Abschnitt ist der letzte notwendige Abschnitt. Er besteht allein aus einem String, welcher die Server angibt auf denen die Wikipedia-Instanz installiert werden soll und die dafür zu verwendende Konfiguration. Dabei ist der String eine durch „:“ getrennte Liste. Jedes Element besteht aus einem Konfigurations-Namen, welcher durch ein „@“ von einer einer Liste (getrennt durch Kommata) von IP-Adressen oder Hostname getrennt ist. Für jeden angegebenen Konfigurations-Namen muss wiederum ein Abschnitt in der Konfigurations-Datei angelegt werden. Diese Abschnitte bestehen dann aus dem Usernamen auf dem Server, dem Verzeichnis in, das die zu übertragenden Daten kopiert werden, dem Verzeichnis in dem das Mediawiki bzw. die MySQL-Datenbank installiert werden soll und dem Namen des MySQL-Service auf dem Server.

4.2.3 Ablauf

In diesem Abschnitt soll der Ablauf des Skripts dargestellt werden und beschrieben werden, welche Dateien erstellt werden.

1. Die Konfigurations-Datei wird eingelesen und auf Korrektheit geprüft.

2. Sofern der Analyse-Schritt oder Filter-Schritt aktiviert wurde, wird ein `FileReader` für das Tracefile erzeugt. Ist eine Analyse gefordert wird ein `WikiAnalyser` erzeugt bzw. ist die Filterung gefordert ein `WikiFilter`. Der `FileReader` sendet nun die gelesen Daten an die erzeugten Instanzen. Wurde eine `WikiAnalyser` erzeugt, erstellt dieser für eine Trace-Datei `trace.gz`, die folgenden Dateien:

   - `trace.gz.stats` (Statistiken zum Trace)
   - `trace.image.gz` (Alle Requests, die ein Bild anfordern)
   - `trace.thumb.gz` (Alle Requests, die ein Thumbnail anfordern)
   - `trace.page.gz` (Alle restlichen Requests)
   - `trace.gz.eps` und `trace.gz.log` (falls die `plot` Option aktiviert wurde; Requests pro Sekunde Graph und gnuplot Ausgabe)

   Eine `WikiFilter` Instanz erzeugt die Dateien `trace.timestampA-timestampB.gz` (gefilterter Trace) sowie `trace.timestampA-timestampB.rewrite.gz` (Trace zum

Wiedereinspielen). Außerdem erzeugt sie eine `WikiAnalyser` Instanz, womit ebenfalls für den gefilterten Trace, die oben beschrieben Datei erzeugt werden.

3. Im nächsten Schritt wird, sofern gewünscht, zuerst das Bilder-Verzeichnis von der Mediawiki-Installation geleert. Dann wird jeweils ein `FileReader` und ein `FileCollector` für die benötigten Bilder bzw. Thumbnails erzeugt. Nachdem der Download abgeschlossen ist wird die Mediawiki-Installation gepackt und parallel die Bilder in die MySQL-Datenbank eingelesen und die MySQL-Datenbank ebenfalls gepackt.

4. Im letzten Schritt wird nun für jeden Server auf dem die Wikipedia-Instanz eingespielt werden soll, ein `SyncClient` erzeugt, der die gepackte MediaWiki-Installation, die gepackte MySQL-Datenbank und die Datei `ppr/server.py`, mittels `scp`, auf den entsprechenden Server kopiert. Und anschließend die kopierte Datei `server.py` mit entsprechenden Parametern über `ssh` aufruft. Das `server.py` Skript, welches wie bereits erwähnt Python 2.4 benötigt, entpackt die kopierten Archive auf dem Server und richtet die Umgebung ein. Anschließend ist auf dem Server die gewünschte Wikipedia-Instanz eingerichtet.

5 Beispiel: Trace-Analyse und -Filterung

Die Trace-Analyse und -Filterung soll in diesem Kapitel anhand eines Beispiel Traces gezeigt werden. Dazu wurde die Trace-Datei `wiki.1194899823.gz`[12] gewählt. Die Analyse dieses Traces ergibt die in Tabelle 1 dargestellten Werte und der Verlauf der Requests pro Sekunde wird in Abbildung 2 gezeigten. Filtert man diesen Trace nun für das Intervall 1194892290-1194894090 und nach den englischen Artikeln, erhält meinen einen Trace der durch die Tabelle 2 und die Abbildung 3 beschrieben wird. Es ist zu erkennen das der größte Teil der Requests Medien-Dateien abruft, welche über den Host `upload.wikimedia.org` erreichbar sind. Bei dem Abruf von Wikipedia Artikeln dominiert klar die englische Wikipedia und die Requests, welche mit „save“ gekennzeichnet sind, können vernachlässigt werden.

[12] `http://www.wikibench.eu/wiki/2007-11/wiki.1194899823.gz`

General	
tracefile:	wiki.1194899823.gz
start time:	Mon, 12 Nov 2007 18:31:30 +0000
end time:	Mon, 12 Nov 2007 19:31:35 +0000
duration:	3604.986 sec
lines:	17.518.902
requests:	17.518.817
page requests:	7969566 (45,5%)
media requests:	9.549.251 (54,5%)
errors:	85
save operations:	3.309 (0,02%)
Hosts	
upload.wikimedia.org:	9.549.251 (54,5%)
en.wikipedia.org:	3.228.167 (18,4%)
meta.wikimedia.org:	977.302 (5,6%)
de.wikipedia.org:	747.068 (4,2%)

Tabelle 1: Trace-Analyse der Datei `wiki.1194899823.gz`

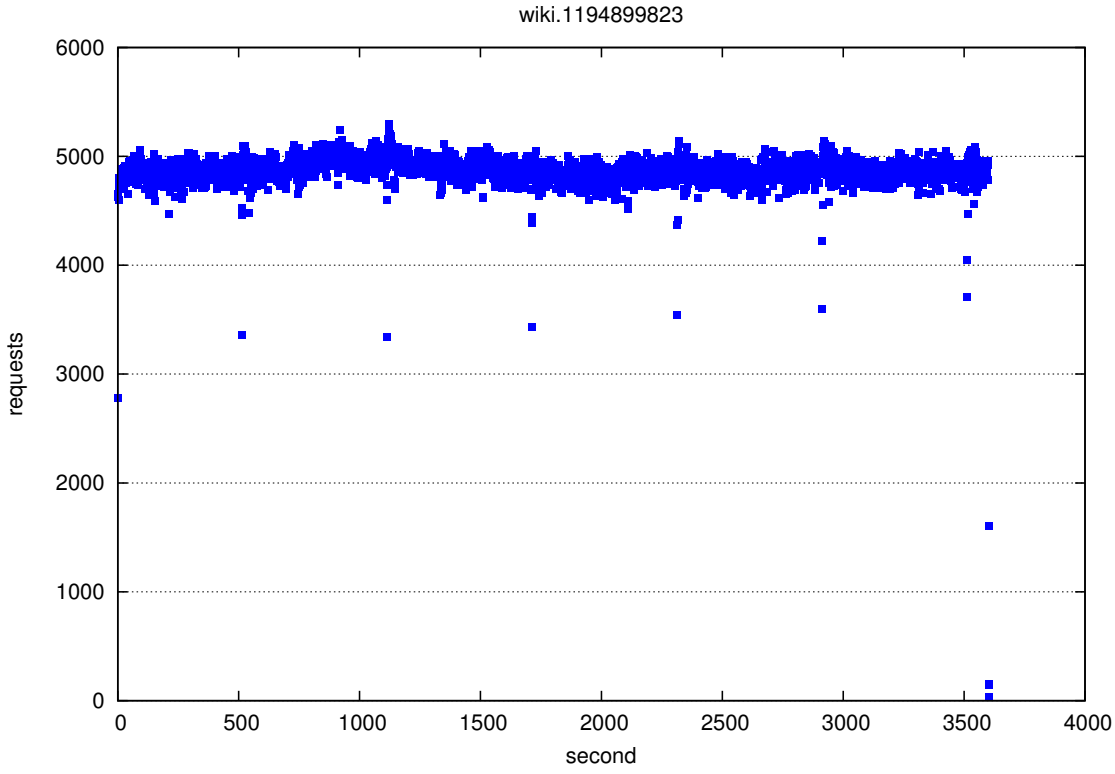

Abbildung 2: Requests pro Sekunde der Datei `wiki.1194899823.gz`

General	
tracefile:	wiki.1194899823.1194892290-1194894090.gz
start time:	Mon, 12 Nov 2007 18:31:30 +0000
end time:	Mon, 12 Nov 2007 19:01:30 +0000
duration:	1800.788 sec
lines:	4795845
requests:	4.795.845
page requests:	1.599.427 (33,3%)
media requests:	3.196.418 (66,6%)
errors:	0

Hosts	
upload.wikimedia.org:	3.196.418 (66,6%)
en.wikipedia.org:	1.599.427 (33,3%)

Tabelle 2: Trace-Analyse der Datei `wiki.1194892290-1194894090.1194899823.gz`

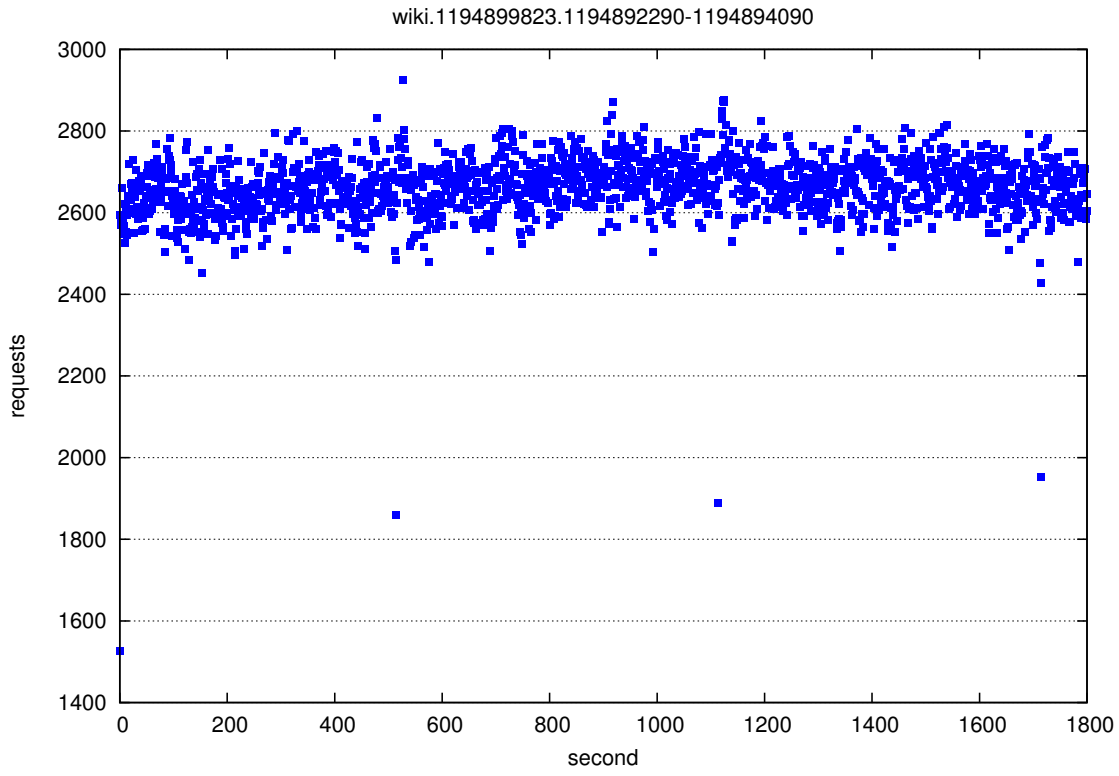

Abbildung 3: Requests pro Sekunde der Datei `wiki.1194892290-1194894090.1194899823.gz`

6 Zusammenfassung

Das in dieser Arbeit vorgestellt Skript ermöglicht es eine lokale Installation der Wikipedia mit Bildern anzureichern. Außerdem kann die lokale Installation anschließend auf weitere Server übertragen und dort eingerichtet werden. Damit ist die in Abschnitt 1.1 beschriebene Aufgabenstellung erfüllt. Das Skript ermöglicht durch die Parallelisierung anhand von Prozessen, eine optimale Auslastung der zu Verfügung stehenden Rechenleistung. Außerdem ist das Skript durch die parallel Abarbeitung sehr Speicher schonend, was sich besonders bei großen Traces auszahlt. Trotzdem ist die Ausführung des Skripts mit allen Schritten immer noch sehr zeitintensiv. Was an den großen Datenmengen liegt, welche verarbeitet, kopiert und verschoben werden müssen.

Zur Stabilität des Skriptes kann keine klare Aussage getroffen werden. Aufgrund einer sehr kurzen Entwicklungszeit und mehrerer Hardwaredefekte konnte es kaum auf dem Produktiv-System getestet werden. Allerdings konnten mit Hilfe des Skriptes 4 Server, für ein 30 minütigen Trace-Ausschnitt, erfolgreich eingerichtet werden. Es liegen jedoch keine Messungen vor, wie hoch der aktuelle Anteil von nicht erreichbaren Ressourcen ist.

7 Ausblick

Wie bereits in der Zusammenfassung (siehe Abschnitt 6) erwähnt, konnte das Skript kaum getestet werden. Hier bedarf es weitere Praxis-Tests und gegebenenfalls Fehlerbeseitigung. Die Phasen während des Downloads und der Installation auf den Servern, benötigt am meisten Zeit und Ressourcen. Diese Phasen könnten Möglicherweise optimiert oder anders konzipiert werden. Außerdem wäre es möglich nach dem Download der Bilder, den gefilterten Trace ein weiteres mal zu filtern und die nicht verfügbaren Bilder und Thumbnails zu entfernen.

Literatur

[1] Baaren, Erik-Jan van: *WikiBench: A distributed, Wikipedia based web application benchmark.*
Masterarbeit, VU University Amsterdam, Mai 2009.
`http://www.wikibench.eu/wp-content/uploads/2010/10/van-baaren-thesis.pdf`.

[2] Hahn, Fabian: *Server load balancing: Wikipedia.*
Seminararbeit, Univeristät Potsdam, 2010.

[3] Urdaneta, Guido, Guillaume Pierre und Maarten van Steen: *Wikipedia Workload Analysis for Decentralized Hosting.*
Elsevier Computer Networks, 53(11):1830–1845, Juli 2009.
`http://www.globule.org/publi/WWADH_comnet2009.html`.

A Beispielkonfiguration

```
1  # EXAMPLE CONFIG
2
3  [general]
4  # Analyse trace file and write statistics
5  analyse=true
6
7  # Filter trace file and analyse filtered trace file
8  filter=true
9
10 # Download images and thumbs from filtered trace file
11 download=true
12
13 # Pack images and database and install them on other server
14 install=true
15
16 # Logging level (optional)
17 # values: notset, debug, info, warning, error, critical
18 # default: debug
19 logging=info
20
21 # Plot request per second during trace analyse (optional - gnuplot required)
22 # values: true, false
23 # default: false
24 plot=true
25
26
27 [trace]
28 # Path of trace file
29 file=traces/wiki.1194899823.gz
30
31 # Is the trace file gzip commpressed? (optional)
32 # values: true, false
33 # default: false
34 gzip=true
35
36
37 # The filter section is read, if in the general section the filter or
38 # download option is true
39 [filter]
40 # Time interval to filter trace (format a:b)
41 # values: timestamp:timestamp or timestamp:seconds
42 # timestamp and seconds are float values
43 # if a > b then b is interpreted as seconds, else as another timestamp
44 interval=1194892290:1800
```

```
45
46 # Host address for rewrite trace (name or IP)
47 # During trace filtering a new trace file is created, which rewrites the
48 # original urls for this host. This trace file is used to replay the trace
49 # an a local system.
50 host=ib1
51
52 # Regular expression to filter urls (optional)
53 # value: regex string
54 # default: http://en.wikipedia.org|http://upload.wikimedia.org/wikipedia/
       commons/|http://upload.wikimedia.org/wikipedia/en/
55 #regex=
56
57 # Save fitler trace gzip commpressed? (optional)
58 # values: true, false
59 # default: false
60 gzip=true
61
62
63 # The download section is read, if in the general section the download or
64 # install option is true
65 [download]
66 # Directory to download images and thumbs from filter trace, also test
67 # if they already exist
68 download_dir=images
69
70 # Port used to send http request for downloading files (optional)
71 # default: 80
72 port=80
73
74 # Number of asynchronously requests during download (optional)
75 # default: 25
76 async=25
77
78 # Mediawiki root directory
79 wiki_dir=/var/www/html/w
80
81 # Remove mediawiki image directory befor copying new images (optional)
82 # value: true, false
83 # default: true
84 clean_images=true
85
86 # MySQL service name on localhost, used to stop and start the database
87 # during packing the content
88 mysqld=mysqld
89
```

```
# MySQL directory, packed for installation on other server
mysql_dir=/var/lib/mysql

# Clean MySQL database before importing new images (optional)
# value: true, false
# default: true
clean_mysql=false

# Tar file containing the clean MySQL database (without imported images) used
# to clean MySQL database if required (optional if clean_mysql is false)
# default: ""
mysql_archive=etc/mysql-clean.tar.bz

# Directory to save packed images and database for exchange
output_dir=etc


# The install section is read, if in the general section the install option
# is true
[install]
# List of configurations and server used to exchange and install the packed
# images and database
# value: config@ip_list:[config@ips_list]
#        config - section in this configfile
#        ip_list - comma seperated list of hostnames or IPs
# defualt: ""
server=centos@192.168.1.104,192.168.1.105:ubuntu@localhost


# Install configurations
# Values:
#   user: Username on server (required)
#   copy_dir: Directory to copy data by scp (optional - default: ~/")
#   wiki_dir: Mediawiki root directory on server (optional - default: "None")
#   mysqld: MySQL service name on server (optional - default: mysqld)
#   mysql_dir: MySQL directory on server (optional - default: "None")
# Remarks:
#   If no wiki_dir is given, the images are not unpacked. If no mysql_dir is
#   given, the database is not unpacked.

# Example configuration for CentOS system
[centos]
user=root
copy_dir=/data/
```

```
134 wiki_dir=/var/www/html/w
135 mysqld=mysqld
136 mysql_dir=/var/lib/mysql
137
138 # Example configuration for Ubuntu system
139 [ubuntu]
140 user=root
141 wiki_dir=/var/www/w
142 mysqld=mysql
143 mysql_dir=/var/lib/mysql
```

Listing 1: Beispiel Konfigurations-Datei