kopal
Daten für die Zukunft

Projekt *kopal*
Kooperativer Aufbau eines Langzeitarchivs digitaler Informationen

Evaluierung von Strategien für lokales Entpacken und Übertragen komprimierter Objekte eines digitalen Archivs

Von

Dipl.-Inf. Kadir Karaca Koçer

Dr. Thomas Wollschläger

Frankfurt am Main 2005

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Vorbemerkung

Innerhalb der Entwicklung des Lösungsdesigns für das Zugriffsmodul des Archivsystems von *kopal* gab es frühzeitig Überlegungen, ob und wenn ja wie ein existierender Web-Cache-Proxy – z.B. Squid - das benötigte Zwischenspeichern häufig angeforderter Dateien übernehmen kann. Die Feststellung der durch das Entpacken verursachten Performanzverluste wurde als ein Aufgabenpaket definiert, in dessen Rahmen mehrere Tests durchgeführt werden sollten. Ziel dieser Tests war, eine Art Entscheidungshilfe für die zukünftige Caching-Strategie und die Anforderungen an die dafür benötigte Hardware zu formulieren.

1. Testdaten

Für die Tests wurde die größte in Der Deutschen Bibliothek gespeicherte Dissertation als Referenzobjekt gewählt. Diese Entscheidung hatte zwei Gründe:

1. Bezug zur Realität: Künstlich erzeugte Daten könnten falsche Rückschlüsse bezüglich später in Frage kommender realer Daten verursachen.
2. Schlimmster Fall (Worst-Case): Die größte Datei als Testmuster zu wählen garantiert, dass bei allen anderen 32.500 Dissertationen eine bessere Performanz realisiert wird.

Insgesamt bestand das Paket aus 15 Dateien und zwei Unterordnern mit 359.913.237 Byte Gesamtgröße.

2. Hardware

Auf drei verschiedenen Rechnern wurden die Tests durchgeführt. Auf einem Server wurden mehrere andere Tests durchgeführt (s.u.).

Die Hardwarekonfigurationen waren wie folgt:

Rechner A:
Pentium III Prozessor, 866 MHz/ 256 MB RAM / 20 GB HDD /Windows XP

Rechner B: (Notebook)
Mobile Celeron Prozessor, 1700 MHz/ 256 MB RAM / 20 GB HDD /Gentoo Linux

Rechner C:
Pentium 4 Prozessor, 3000 MHz/ 504 MB RAM (Shared Graphic)/ 20 GB HDD /Windows XP

Rechner D: (Testserver)
Xeon Prozessor 2800 MHz mit Hyperthreading/ 2GB RAM /SuSE Linux

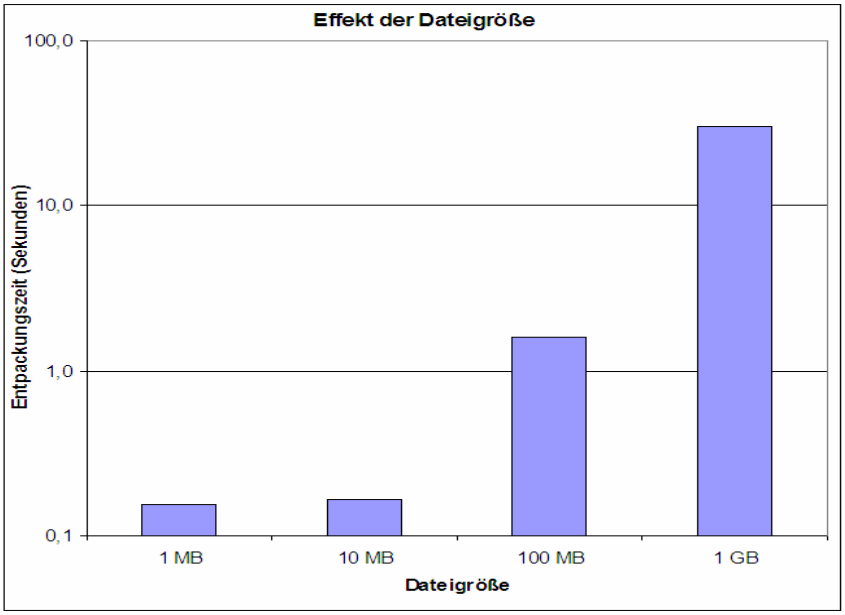
3. Testverfahren

Eine erste Überlegung war, ob und wenn ja, wie die Komprimierungsstärke die Geschwindigkeit des Entpackens beeinflusst. Dafür sind zwei verschiedene ZIP-Dateien von dem Dissertationsverzeichnis mit Hilfe von WinZIP v.9.0 erzeugt worden; eine mit maximaler Komprimierungsstufe, MAX genannt, und eine ohne Komprimierung, MIN genannt.

Das Entpacken der MAX war auf Rechner A fast fünf Prozent langsamer als das Entpacken der MIN. Daher wurde für weitere Tests MAX ausgewählt, um beim Worst-Case-Szenario konsequent zu bleiben.[1]

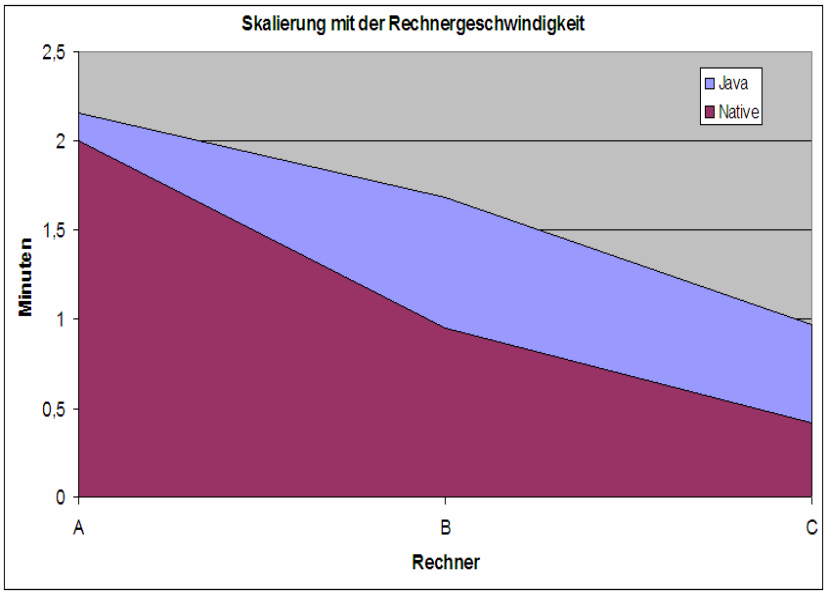
Die zweite Überlegung war, das Verhältnis der Entpackroutine in Java gegenüber den nativen Entpackern auf Geschwindigkeitsunterschiede zu untersuchen. Es wurde die Javaklasse **`TestZip`** implementiert und kompiliert, die beliebige ZIP-Dateien entpackt. Erste Ergebnisse auf Rechner A waren viel versprechend für Java: Der Geschwindigkeitsnachteil lag bei unter neun Prozent. Dieses Verhältnis hat sich aber interessanterweise mit der Geschwindigkeit des Rechners schnell geändert: schon bei Rechner B war TestZIP gegenüber UnZIP v. 5.50 signifikant langsamer (100 Sekunden zu 57 Sekunden). Auf Rechner C war der Unterschied mit 58 zu 25 Sekunden noch größer.

Der Einfluss der Dateigröße war der dritte Untersuchungsgegenstand. Dafür wurden mit Hilfe eines Shell-Skripts vier ZIP-Dateien aus Zufallswerten mit Größen von 1 MB, 10 MB, 100 MB und 1 GB erzeugt und wieder entpackt - alles auf Rechner D. Wie erwartet, erhöht sich die Entpackungszeit entsprechend der Dateigröße. Interessant ist der Zusammenhang: Während die Zeiten für 10 MB und 100 MB miteinander in einem idealen linearen Verhältnis stehen (eine zehnfach größere Datei braucht die zehnfache Zeit), folgen ganz kleine und ganz große Dateien dieser Linearität nicht und brauchen mehr Zeit als erwartet. Die Tatsache, dass der Server für eine 10 MB Datei nur 8,5% mehr Zeit braucht als für eine zehnfach kleinere 1 MB Datei, deutet darauf hin, dass unter einer bestimmten Dateigröße die Zeit, die vom Betriebsystem für Suchen, Laden, Initialisieren u.ä. des Entpackungsprogramms benötigt wurde, viel größer ist, als die Zeit zum Entpacken selber. Die Konsequenz für uns ist: Egal wie viele und wie schnelle Prozessoren zur Verfügung stehen, muss sogar für die kleinste Datei mit einem Minimum an Verzögerung gerechnet werden (Grafik 1a). Da diese Verzögerung unter 0,1 Sekunden beträgt, ist sie hier jedoch nicht relevant.



Grafik 1a: Entpackungszeiten gegen Dateigröße (alle Werte auf Rechner D)

[1] Die Benchmarks in einige Internetquellen behaupten, dass stärkere Kompression schneller sei. Angeblich hängt das Ergebnis vom Leistungsverhältnis Prozessor/Festplatte und der Komprimierbarkeit des Archivs ab. Das Thema ist hier nicht weiter untersucht worden, da die Unterschiede von ca. 5% in diesem Fall irrelevant waren.



Grafik 1b: Entpackungszeiten gegen Rechnergeschwindigkeit (alle Werte auf Rechner D)

Die Probleme mit einer 1GB-Datei lassen sich mit Engpässen im Hauptspeicher oder mit langsamen Festplatten-Zugriffen erklären und sind viel wichtiger, da sie mögliche Schwierigkeiten mit DVD-ROM-Images signalisieren (momentan bis zu 8,5 GB, in ein paar Jahren bis zu 30 GB). Typische CD-ROM-Images dagegen bereiten keine Probleme (unter 20 Sekunden).

Als viertes wurde die Relation zwischen Rechnergeschwindigkeit und Entpackzeit untersucht. Die Ergebnisse sind in Grafik 1b zu sehen. Wie erwartet gibt es einen direkten Zusammenhang – das seltsam anmutende Verhalten des Rechners B lässt sich mit unterschiedlichen Betriebsystem- und Java Virtual Machine - Versionen erklären. Da für die Rechnergeschwindigkeit kein absolutes Maß existiert, kann keine direkte Aussage aus den Messergebnissen formuliert werden.

Interessant verlief auch die Untersuchung, wie ein Server bei verschiedener gleichzeitiger Last reagiert. Dafür wurde folgendes Skript programmiert und auf dem Server (Rechner D) laufen lassen:

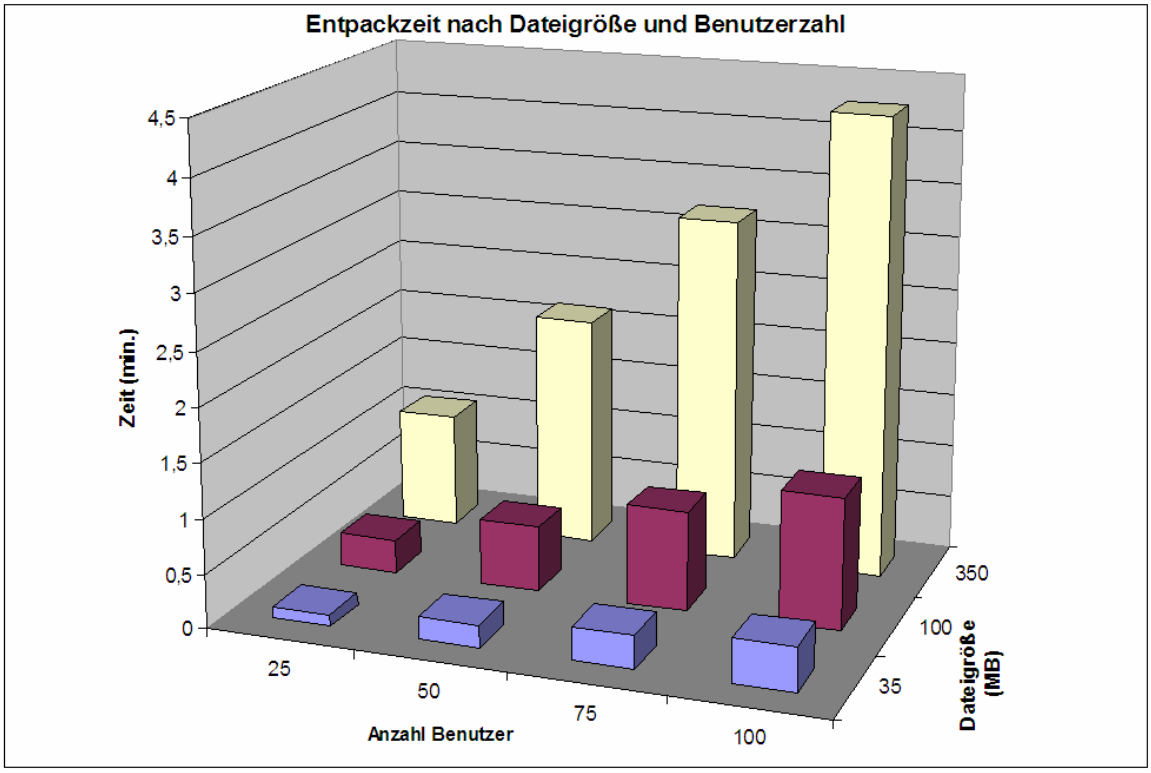
```
#!/bin/bash

for file in *.gz
do
        for user in 25 50 75 100
        do
                echo ------------------------------
                echo File: $file
                echo User: $user
                time {
                        for (( task=$user ; $task > 0 ; task-- ))
                        do
                                echo -n .
                                gunzip -c $file >/dev/null &
                        done
                        wait
                        echo
                }
```

done
done

Die Ergebnisse sind in Grafik 2 zu sehen. Die Benutzung von **`/dev/null`** als Entpackungziel hat die Folgen:

a) Ohne bremsende Festplattenschreibzugriffe konnte der Xeon Prozessor seine volle Kapazität nutzen und skalierte in beiden Richtungen fast linear.
b) Genau deswegen sind diese Werte aber mit Vorsicht zu bewerten. Bei größeren Dateien, die weder in den Festplattencache noch in den Betriebssystembuffer passen, werden die Entpackungszeiten einige Prozent höher sein.

Wichtigste Erkenntnis ist, trotz Worst-Case-Szenario von 100 gleichzeitigen Entpackensvorgängen mit 350MB und mit großzügigem Addieren von 3,5 Minuten für fehlende Festplattenschreibzeiten, dass alle 100 Dateien in weniger als acht Minuten fertig entpackt worden sind, was völlig akzeptabel ist. Falls diese acht Minuten den Anforderungen nicht entsprächen, könnte der neue Server entsprechend dimensioniert werden. Da diese 100 Entpackungsprozesse voneinander völlig unabhängig sind, werden sie mit der Anzahl der Prozessoren linear skalieren. D.h., auf einem neuen Server mit acht schnelleren Prozessoren und einer 4GB Ramdisk sind alle 100 innerhalb von 30 Sekunden fertig.



Grafik 2: Entpackzeiten auf Rechner D mit Abhängigkeiten von der Anzahl gleichzeitiger Benutzer und von der Dateigröße

Das letzte Experiment sollte klären, wie die Konkurrenz verschiedener (De-) Komprimierungsalgorithmen in Punkto Geschwindigkeit aussieht. Internetrecherchen ergaben, dass unter den URLs:
http://www.elis.ugent.be/~wheirman/compression/index.php?network=10000&decomp=10#ranking
http://datacompression.info/SourceCode.shtml
http://www.maximumcompression.com/
schon sehr ausführliche Informationen und Benchmarkergebnisse, die eigene Tests überflüssig machen, gelistet sind.

Da im Test nur die Zeiten während des Dekomprimierens entscheidend waren, ergab sich die folgende Reihenfolge:

#	PROGRAMM (+ Parameter)	ZEIT (Sekunden)	LEISTUNG (Kleiner ist besser)	KOSTEN	URL
1.	LZOP -7	1,0	%26,6	GPL	http://www.lzop.org/
2.	RAR -m3	1,1	%13,4	18€[2]	http://www.rarlabs.com/index.htm
3.	ZIP -6	1,2	%23,4	BSD	http://www.info-zip.org/pub/infozip/
4.	GZIP -5	1,9	%23,8	GPL	http://www.gzip.org/

Es überrascht, dass das – außer vielleicht in Fachkreisen – völlig unbekannte LZOP der schnellste Entpacker ist. Nach weiterer Recherche wird deutlich, dass die Entwickler (http://www.oberhumer.com/) sich ganz professionell mit dem Thema beschäftigen (unter anderem im Zusammenhang mit der NASA Mars Mission) und LZOP von Anfang an für Geschwindigkeit optimiert wurde. Diese Optimierungen gehen leider auf Kosten der Kompressionsleistung und dies erklärt, warum LZOP im Vergleich so wenig komprimiert. Die offenen Quellcodes unter GPL sind ebenfalls interessant.

RAR ist zwar 10% langsamer, aber komprimiert fast doppelt so gut. Außerdem hat das **`*.rar`** Format einige wichtige Vorteile bezüglich Archivintegrität und Recovery, was für *kopal* sehr nützlich sein könnte:
http://www.winrar-rog.de/html-ger/info/compare3.htm

ZIP ist nicht so leistungsfähig und schnell und hat keine Extras wie RAR, aber es ist das meist verbreiteste und bekannteste Format und hat die liberalste Lizenz von allen.

GZIP gewinnt in keine der Disziplinen gegen ZIP – daher ist es uninteressant.
Außer diesem Geschwindigkeitsranking gibt es auch ein Ranking nach Kompressionsleistungen:
http://studwww.ugent.be/~jdebock/gimp_source_compression_test.htm

4. Übertragungsgeschwindigkeit

Während der Messungen dauerte der Transfer der oben genannten Dissertation als gesamtes Verzeichnis in unkomprimierter Form, wie es später der Fall sein wird, 70 Sekunden. Damit schafft das hausinterne Netzwerk Der Deutschen Bibliothek (100 MBit-Ethernet) einen guten Wert von 4,9 MB/Sek. In realen Benutzungsfällen mit mehreren gleichzeitigen Zugriffen im Multimedialesesaal dürfte die Leistung entsprechend geringer ausfallen.

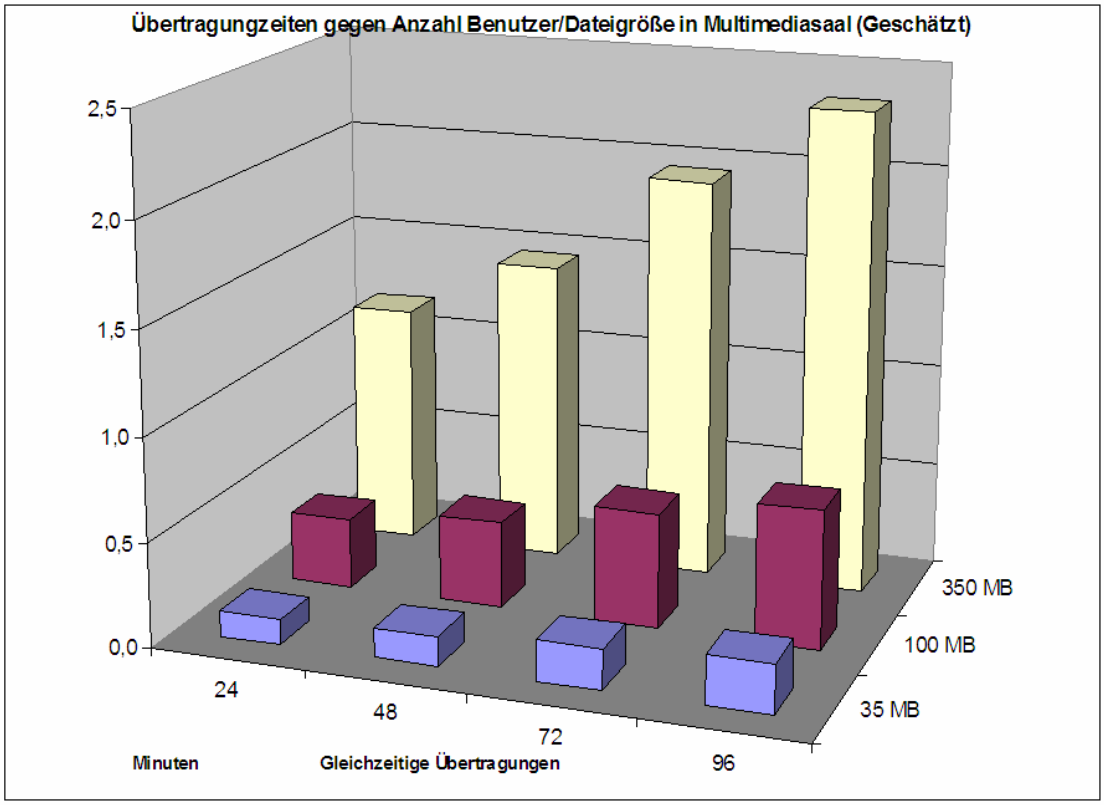
Die Ergebnisse der Analyse von Übertragungszeiten bei internen und externen Zugriffen sind in Grafik 3 zu sehen. Es gibt grundsätzlich zwei völlig verschiedene Ansätze:

1. **Multimedia Lesesaal in der Deutschen Bibliothek Frankfurt:** Dieser genießt den Vorteil, in demselben Gebäude mit dem Server zu sein. Die Verbindung ist sehr schnell und vor allem so intelligent realisiert, dass die Leistung mit der Anzahl der Benutzer nicht linear, sondern treppenförmig in jeweils in 24er Schritten sinkt. D.h., 1 bis 24 Benutzer sind an einem Switch mit 2 GBit Anbindung angeschlossen und können so alle jeweils die oben genannten 4,9 Mbyte/Sek. komplett nutzen, weil sie sich gegenseitig nicht blockieren. Die 25. bis 48. Benutzer schließen

[2] Der Entpacker selbst ist als Sourcecode und kostenfrei erhältlich. Kostenpflichtig ist der Packer.

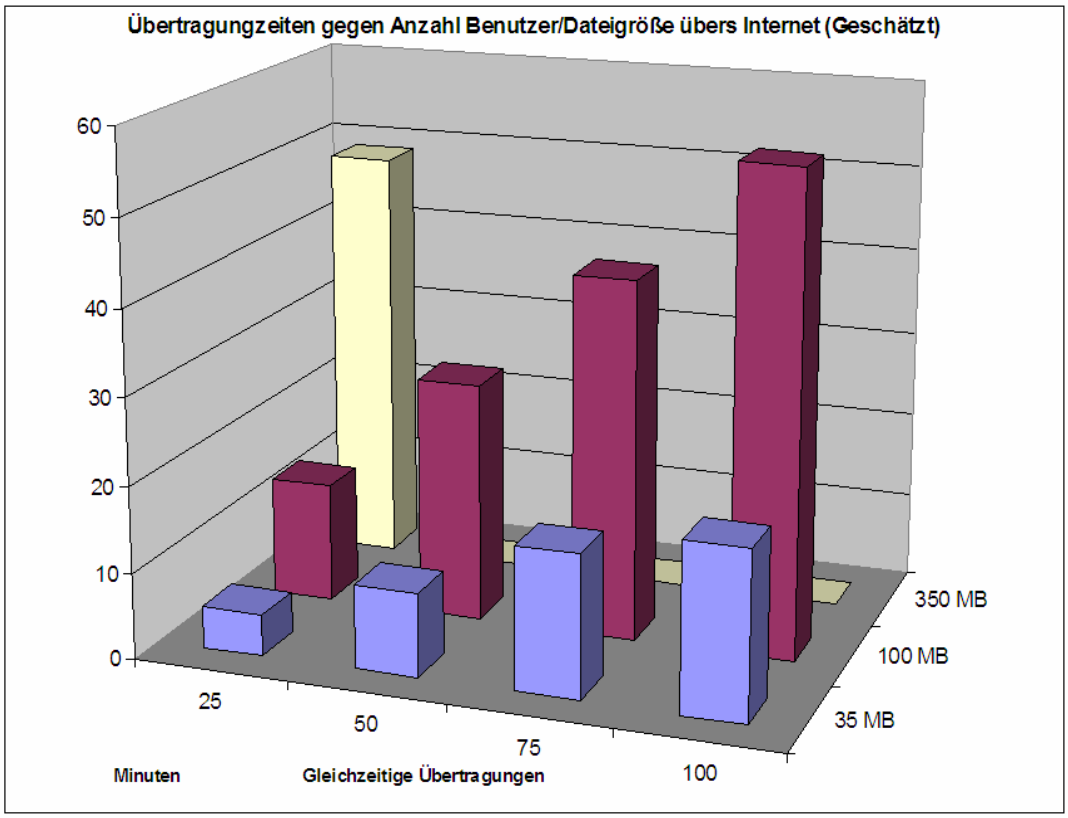
sich an einem zweiten, die 49. bis 72. Benutzer an einem dritten Switch an, usw. Wegen anderer Engpässe, z.B. bei der Netzwerkanbindung der Server oder beim RAID-Array, sinkt die Leistung natürlich mit jedem neuen Benutzer. Aber der Leistungseinbruch ist so gering und das Netz ist so schnell, dass im Endeffekt 15 Sekunden mehr Warten für 350 Megabyte Daten nicht so schwer fällt.

2. **Benutzer außerhalb des internen Netzes:** Im Gegensatz zum internen Netz, wo jeder allein eine 100MBit Verbindung hat, ist die Anbindung nach draußen mit 34MBit realisiert und zwar als Summe aller Verbindungen. Als Folge ist es insgesamt viel langsamer und die Leistung sinkt linear mit jedem neuen Benutzer. Wenn, optimistisch angenommen, 25MBit nur für *kopal* zur Verfügung stehen würden, würde bei 25 gleichzeitigen Übertragungen von außen 1MBit pro Benutzer übrig bleiben. Bei 200 Verbindungen bleibt sogar nur Dual-ISDN Niveau übrig, was für Wissenschaftler und Studierende mit DFN- und Großstädter mit 5MBit DSL Anschluss eindeutig zu gering wäre. Die schlechtesten drei Werte in Grafik 3b (jeweils 1,5-2,5 und 3,25 Stunden) sind nicht abgebildet, um eine bessere Skalierung für die anderen Werte zu ermöglichen.



Grafik 3a: Übertragungszeiten in Minuten in Abhängigkeit von gleichzeitigen Nutzern und Dateigröße im Multimedia Lesesaal (DDB internes Netzwerk),

Eine Ausnahme bilden die Arbeitsplätze im Multimedia-Lesesaal in der Deutschen Bücherei Leipzig. Sie sind mit einer extra Leitung an die Server in Frankfurt angebunden. So gesehen sind sie „im" Frankfurter Netz. Andererseits ist diese Verbindung mit 34MBit im Vergleich zu den 2GBit des Frankfurter Multimedia-Lesesaals sehr gering, so dass die Benutzer schlimmstenfalls dort mit ca. 12-fachen Übertragungszeiten rechnen müssen.



Grafik 3b: Übertragungszeiten in Minuten in Abhängigkeit von gleichzeitigen Nutzern und Dateigröße, Zugriff übers Internet (DDB externes Netzwerk)

Als Lösung für Kapazitätsengpässe wurde ein Thin-Client-Ansatz vorgeschlagen. So muss nicht jedes Mal die ganze entpackte Datei, sondern nur die aktuelle Bildschirmausgabe übertragen werden. Nähere Informationen finden sich unter:
http://www.sun.com/sunray/sunray1/index.xml
http://www.sun.com/sunray/techinfo/New_SR_WP_12_04.pdf

Diese Lösung hätte gleich mehrere Vorteile:

- Viel bessere Reaktionszeiten für die Benutzer wegen der Übertragung sehr geringer Datenmengen und das auch nur auf Anforderung. Die erste Seite ist immer in unter zwei Minuten da!
- Der Unterschied zwischen dem Frankfurter und dem Leipziger Multimedia-Lesesaal verschwindet praktisch. Für Leipzig muss keine extra Lösung entwickelt werden.
- Das Problem „Emulation älterer Systeme“ reduziert sich auf ein Emulationsfenster auf dem Server und ist damit einfacher handhabbar.
- Das bisher eingesetzte Multimediabereitstellungssystem könnte damit abgeschafft werden.
- Kein Betriebsystem, Anwendungsprogramme u.s.w. sind auf dem Client:
    - kein Installationsaufwand
    - kein Administrationsaufwand
    - der Client ist damit viel weniger störungsanfällig
    - der Benutzer kann - wissend oder unwissend - nichts kaputtmachen
- Authentifizierung durch Chipkarte
    - es können/dürfen nur berechtigte Benutzer arbeiten
    - je nach Dienstleistung kann einzeln abgerechnet werden
    - die Session des Benutzers existiert über Rechnergrenzen hinweg

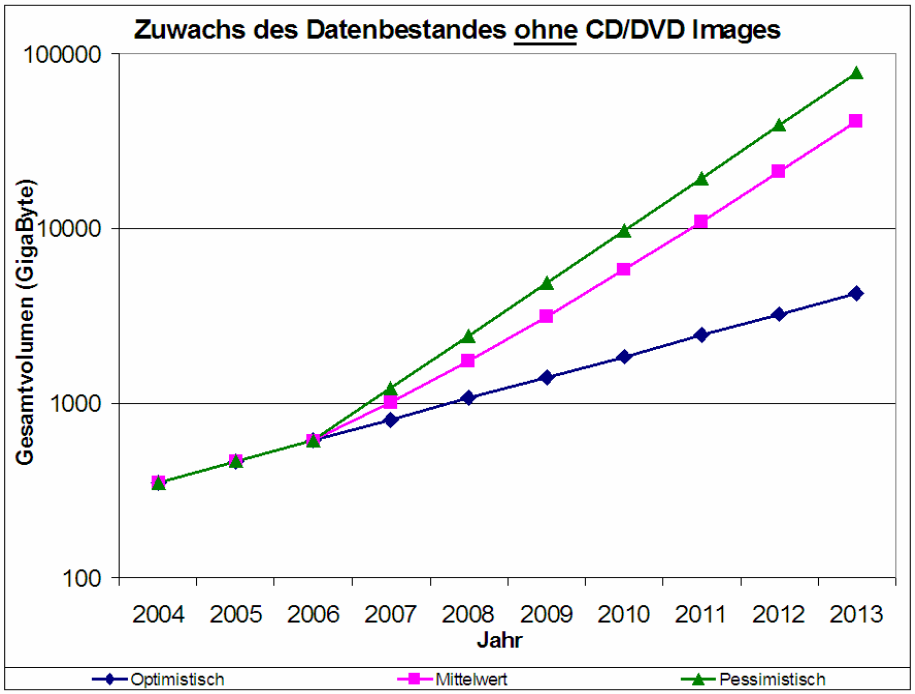
Leider bringt dieser Ansatz für die externen Benutzern nichts und die Kosten sind unbekannt.

5. Datenvolumen

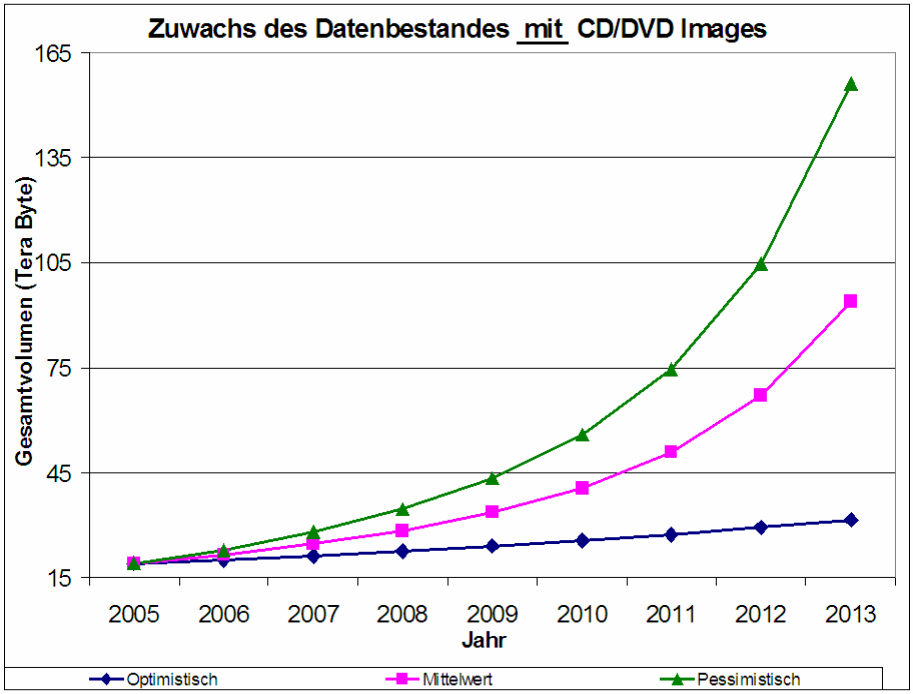
Eine Entscheidung über mögliche Caching-Strategien und die dafür benötigte Hardware kann nur getroffen werden, wenn - unter anderem - die Größe der zu verarbeitenden Datenmenge bekannt ist. Genau diese Menge ist aber nicht bekannt, und ihre Entwicklung in der Zukunft ist auch ungewiss. Als einzige Möglichkeit blieb daher, die bekannten Zahlen aus den Jahren 2004 und 2005 mit verschiedenen Annahmen zu extrapolieren, um mindestens die grobe Bandbreite zu bestimmen.

In Grafik 4a sind drei mögliche Entwicklungen zu sehen. Optimistische Annahme ist: Der Datenbestand (ohne CD/DVD-Images) wird sich in Zukunft prozentual so ändern, wie er sich zwischen 2004 und 2005 geändert hat. Pessimistische Annahme ist: Nach Inkrafttreten des neuen Gesetzes über Die Deutsche Bibliothek, welches auch die Pflichtablieferung von Netzpublikationen beinhaltet, wird sich ab 2007 die Datenmenge jedes Jahr verdoppeln. Deren Mittelwert sieht zwar optisch nicht mittig aus, was aber nur an der logarithmischen Skalierung liegt.

Grafik 4b dagegen zeigt einen „imaginären" Zustand, in dem in 2005 alle in Der Deutschen Bibliothek befindlichen ca. 30.000 CD- und über 1.000 DVD-ROMs als Images in *kopal* gespeichert würden. Die optimistische Schätzung nimmt an, dass ab 2006 das Datenvolumen der Images jedes Jahr um 5% zunimmt und addiert sie mit der ebenfalls optimistischen Schätzung von Online-Dokumenten (blaue Linie in Grafik 4a).
Die pessimistische Schätzung dagegen geht von einem Zuwachs von 20% aus und addiert die Summe mit der ebenfalls pessimistischen Schätzung von Grafik 4a (grüne Linie).
Da die Wahrheit meistens irgendwo zwischen beiden Extremen liegt, markiert die Mittelwertlinie den Durchschnitt beider Werte.



Grafik 4a: Entwicklung des Datenbestandes **a)** ohne (logarithmisch) CD/DVD Images (linear gezeichnet). Optimistische und pessimistische Prognosen und deren Mittelwert.



Grafik 4b: Entwicklung des Datenbestandes **mit** CD/DVD Images (linear gezeichnet). Optimistische und pessimistische Prognosen und deren Mittelwert.

Egal welches Szenario realistisch erscheint, muss *kopal* für mindestens 40 Terabyte konzipiert werden. Ob dies Anfang 2009 oder Ende 2010 erreicht wird, ist im Endeffekt nicht so wichtig. An die - für heutige Verhältnisse unerreichbar erscheinenden - Größenordnungen von 300 Terabyte und mehr muss man sich ebenfalls gewöhnen. Schließlich werden sie nur zehn Jahre nach Projektende allein für Die Deutsche Bibliothek benötigt. Von anderen Institutionen, die DIAS-Core in Zukunft mitbenutzen werden, ganz zu schweigen.

Wie groß der Cache sein soll, muss diskutiert werden. Z.B. braucht die Staats- und Universitätsbibliothek Göttingen als zweiter Projektpartner fast keinen Cache, weil sie mit der Gesellschaft für Wissenschaftliche Datenverarbeitung Göttingen als Systemdienstleister ohnehin über eine Gigabit-Leitung verbunden ist. Die Deutsche Bibliothek benötigt möglichst viel Cache, weil ihr nur ein Dreißigstel dieser Verbindungskapazität zur Verfügung steht. Andererseits, je größer der Cache ist, desto teurer ist er und desto aufwendiger ist es auch, ihn ständig in einem konsistenten Zustand zu halten.
Je nach finanziellem Aufwand kann man ihn auf 5 bis 15% des Gesamtvolumens schätzen.

6. Fazit

Die gute Nachricht zuerst: Bei der durchschnittlichen Dissertationsgröße (momentan knapp unter 7 MB, bezogen auf Dokumente, die bei Der Deutschen Bibliothek archiviert sind) könnte sogar der bestehende Server und das interne Netzwerk leicht Hunderte gleichzeitiger Benutzer bedienen – wobei gleichzeitig tatsächlich gleichzeitiges Entpacken/Übertragen bedeutet und die gerade lesenden / suchenden usw. Benutzer nicht beinhaltet. Diese positiven Zahlen setzen sich fort bis zu 100 gleichzeitigen Benutzern mit jeweils 350 MB Dateiumfang (größte existierende Dissertation). Die Wartezeit würde im Frankfurter Lesesaal unter 8 Minuten und im Leipziger Lesesaal unter 22 Minuten betragen und wäre damit zumutbar.

Die schlechte Nachricht ist: Diese Zahlen basieren auf der Annahme, dass die angeforderten Dokumente sich im Cache befinden, die Benutzer im Lesesaal sitzen und mehr als hundert gleichzeitige Zugriffe unwahrscheinlich sind. Falls sehr viele Anfragen außerhalb des Netzes Der Deutschen Bibliothek kommen würden und viele von den angeforderten Dateien nicht im Cache sind, wird das System gleich an vielen Fronten Engpässe haben.

Es ist zu hoffen, dass mit dem Ergebnis dieser Tests eine Entscheidungshilfe für die zukünftige Caching-Strategie zur Verfügung gestellt werden konnte. Die Anforderungen an die dafür benötigte Hardware hängt stark von den künftigen Zielvorgaben bzw. -vorstellungen für die Menge der eingehenden digitalen Archivobjekte, die Anzahl erwartete Besucher, die gewünschten Zugriffszeiten und von den finanziellen Rahmenbedingungen. Hier sind Entscheidungen zu treffen, welche die Struktur und Performanz des Zugriffssystems stark beeinflussen werden.

Das Projekt *kopal* stellt diese Ergebnisse der interessierten Öffentlichkeit zur Verfügung in der Hoffnung, dass andere Projekte / Entwicklungen von diesen Ergebnissen profitieren können. Als von öffentlichen Mitteln gefördertes Projekt und um den Geist des Open Access zu unterstützen, ist es ausdrücklich erwünscht, unter Angabe der Quelle die Ergebnisse und Aussagen weiterzugeben und weiterzuverwenden.