Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Evaluation von alternativen Speicherszenarien für hierarchische Speichersysteme

vorgelegt von

Marc Perzborn

(Matrikel-Nr.: 6554028)

Bachelorarbeit im Studiengang Lehramt an Beruflichen Schulen

Eingereicht in dem Unterrichtsfach Berufliche Informatik

Universität Hamburg

2016

Betreuer: **Dr. Julian Kunkel, Jakob Lüttgau**

Erstgutachter: **Dr. Julian Kunkel**

Zweitgutachter: **Prof. Dr. Thomas Ludwig**

Abgabedatum: **31.10.2016**

Abstract

Ziel der vorliegenden Bachelorarbeit war es, das Simulationsprogramm FeO auf seine Korrektheit zu überprüfen und zu verbessern. Dazu wurden verschiedene Szenarien simuliert. Die Ergebnisse bestätigen zum großen Teil die Annahmen. Im Cache gespeicherte Informationen können schneller Ausgegeben werden, als nicht im Cache gespeicherte. Bei wenig verbauten Laufwerken müssen lesende Anfragen auf nicht gecachte Informationen warten, wenn jedes Laufwerk belegt ist. Das Speichermanagement eines vollen Cache funktioniert einwandfrei. Bei einem Cache mit freiem Speicherplatz wird nicht wie in einem realen System reagiert. Die Verarbeitungszeiten für Anfragen auf nicht gecachte Informationen variiert, wenn verschiedene Komponenten des Bandarchives, beispielsweise die Generation der Laufwerke, die Anzahl der Laufwerke des Bandarchives oder die Bandbreite von Komponenten, verändert werden.

Danksagung

An dieser Stelle möchte ich Dr. Julian Kunkel für seine aufgebrachte Zeit danken, in welcher er mit mir über die Erstellung diverser Grafiken und Inhalte gesprochen hat. Jakob Lüttgau hat sehr viel Zeit damit verbracht mit mir über sein Programm und die gesamte Thematik zu sprechen, wofür ich ihm danken möchte. Meiner guten Freundin Victoria Bieder möchte ich für ihr Korrekturlesen danken.

Inhaltsverzeichnis

1. Einleitung 1

1.1. Motivation 1

1.2. Ziele der Arbeit 2

1.3. Gliederung der Arbeit 2

2. Theoretischer Hintergrund 3

2.1. Hierarchische Speichersysteme 3

2.1.1. Bandarchive 4

2.1.2. StorageTek SL8500 5

2.1.3. Virtuelle Bandarchive 6

2.1.4. High Performance Storage System 7

2.2. Simulationsprogramm FeO 9

2.2.1. Diskrete ereignisorientierte Simulation 9

2.2.2. Programmiersprache 12

2.2.3. Architektur 13

2.2.4. Nutzlast 15

2.2.5. Simulation 16

2.2.6. Datenausgabe 18

3. Evaluation 20

3.1. Grundlegender Versuchsaufbau 20

3.2. Cacheverhalten 21

3.2.1. Gecachte und nicht gecachte Informationen 21

3.2.2. Datenmanagement eines vollen Cache 26

3.3. Verhalten bei Veränderung verschiedener Komponenten 29

3.3.1. Verbesserung der Laufwerke 29

3.3.2. Einbau mehrerer Laufwerke 33

3.3.3. Verbesserung der Bandbreite des Klienten 36

4. Schlussbetrachtung 40

4.1. Zusammenfassung 40

4.2. Ausblick 42

Literaturverzeichnis 43

Abbildungsverzeichnis 45

Tabellenverzeichnis 45

1. Einleitung

In diesem Kapitel wird ein Überblick über diese Arbeit und ihre Bedeutung gegeben. Abschnitt 1.1 gibt eine thematische Einführung. Die Ziele der Arbeit werden in Abschnitt 1.2 formuliert. Zum Schluss wird in Abschnitt 1.3 die Gliederung der Arbeit erläutert.

1.1. Motivation

Heutzutage gibt es viele verschiedene Speichermedien. Dazu gehören zum Beispiel USB-Laufwerke, magnetische Bänder, optische Speichermedien (Jones & Bartlett, 2014). Durch die Vielfalt an verfügbaren Speichermedien lassen sich diese in Hierarchien aufteilen. Kombiniert werden verschiedene Speichermedien beispielsweise in Bandarchive. Diese werden häufig in Einrichtungen verwendet, welche einen großen Speicherbedarf besitzen. Besonders Wissenschaftler, die Großrechner verwenden, benötigen Speichersysteme, welche eine hohe Speicherkapazität und einen langlebigen Speicher besitzen. Bänder erfüllen besonders die Anforderung an die Langlebigkeit und besitzen in der Masse eine hohe Speicherkapazität. Im Vergleich zu anderen Speichermedien gehören die Bänder zwar zu den langsameren Medien, sind im Gegenzug aber sehr kostengünstig (Jones & Bartlett, 2014).
Die Speichergröße von Großrechnern wächst geometrisch, deshalb steigen die Anforderungen an die Speichersysteme ebenso (Inman, Grider & Chen, 2014, Abstract). Aus diesem Grund ist es wichtig, die Technologien im Bereich der magnetischen Bänder weiter zu erforschen und zu verbessern. Um bereits vor der Invention neuer Technologien überprüfen zu können, ob die geplanten Neuerungen lohnenswert sind, ist es hilfreich Bandarchive gut zu kennen und simulieren zu können. Aktuell werden Simulationsprogramme entwickelt, mit denen Bandarchive simuliert werden können. Die Verbesserung derer ist wichtig, um eine effektive Forschung im Bereich der hierarchischen Speichersysteme sicherzustellen.

1.2. Ziele der Arbeit

In dieser Arbeit wurde mit einem Simulationsprogramm gearbeitet, um die Vorgänge der hierarchischen Speichersysteme abzubilden. Da dieses Programm erst vor wenigen Monaten fertiggestellt wurde, gibt es mit dieser Arbeit auch die Möglichkeit das Programm weiter auf seine Korrektheit zu überprüfen und zu verbessern, sollten einige Prozesse nicht korrekt simuliert werden. Auf diese Weise wird ein besseres Verständnis über die hierarchischen Speichersysteme und deren innere Vorgänge erlangt und die zukünftigen Simulationsprozesse können verbessert werden.

1.3. Gliederung der Arbeit

Nach der Einführung in die Arbeit werden in Kapitel 2 Hintergrundinformationen zu der Thematik gegeben. Es wird anhand des praxisnahen Beispiels der Bandarchive beschrieben was hierarchische Speichersysteme sind. Im Anschluss wird das in der Arbeit verwendete Simulationsprogramm und seine Funktionsweise vorgestellt. In Kapitel 3 werden ausgewählte Szenarien vorgestellt, welche mit Hilfe des Simulationsprogrammes simuliert und ausgewertet wurden. Eine Zusammenfassung über die gesamte Arbeit wird in Kapitel 4 vorgenommen. Zum Schluss wird ein Ausblick über das mögliche weitere Vorgehen zu diesem Themenbereich gegeben.

Zusammenfassung

Auf Grund der hohen Nachfrage nach magnetischen Bändern ist es sinnvoll hier neue Technologien zu entwickeln. Hierfür ist es hilfreich Bandarchive simulieren zu können. Für diesen Zweck werden aktuell Simulationsprogramme entwickelt. Simulationen sind unter Anderem hilfreich, um kostengünstig Daten für die Forschung zu erheben. Diese Arbeit soll zum besseren Verständnis über die hierarchischen Speichersysteme beitragen und künftige Simulationsprozesse verbessern.

2. Theoretischer Hintergrund

In diesem Kapitel werden die theoretischen Aspekte dieser Arbeit erläutert. In Abschnitt 2.1 werden zunächst die hierarchischen Speichersysteme erläutert. In Abschnitt 2.2 wird das Programm FeO beschrieben, welches für die Simulation von hierarchischen Speichersystemen entworfen wurde.

2.1. Hierarchische Speichersysteme

Die verschiedenen Arten von Speichermedien haben je ihre Vor- und Nachteile, besonders im Hinblick auf die Zugriffszeit, die Speicherkapazität und die Kosten für das Medium. Die verschiedenen Speichermedien lassen sich somit in einer Hierarchie ordnen, welche in Abbildung 1 in Form einer Pyramide dargestellt ist.

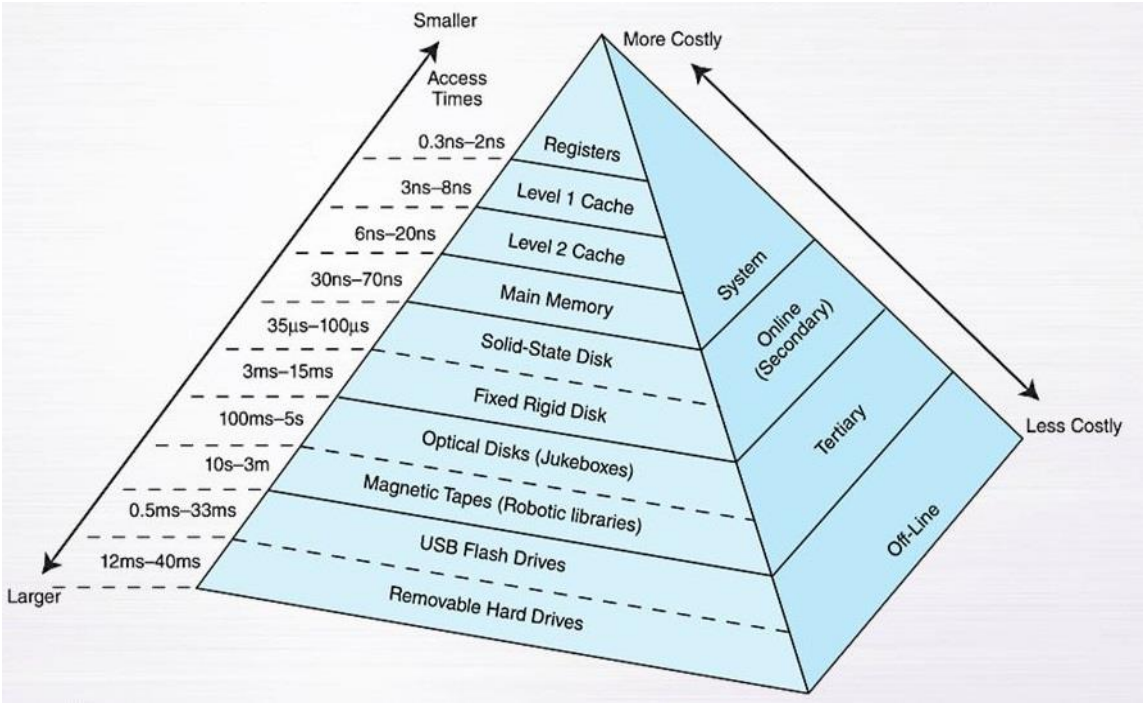


Abb. 1: Die Hierarchie der Speichermedien als Pyramide (Jones & Bartlett, 2014).

In Abbildung 1 ist zu erkennen, dass die Zugriffszeit zur Spitze der Pyramide geringer wird, die Kosten pro GB steigen dafür. Die Speicherkapazität steigt hingegen in Richtung der günstigeren Speichermedien. Bei diesen steigt die Zugriffszeit stark an.

Es bietet sich für große Speichersysteme an, mehrere Arten an Speichermedien zu kombinieren. Häufig benötigte Daten werden auf einem Speichermedium geschrieben, welches in der Hierarchie höher angesiedelt ist, zum Beispiel im Cache. Werden Daten selten benötigt, können diese auf Speichermedien gespeichert werden, welche in der Hierarchie weiter unten angesiedelt sind, zum Beispiel auf Bändern. Auf diese Weise wird das Speichersystem Kosten / Nutzen effizienter aufgebaut.

2.1.1. Bandarchive

Bandarchive sind Systeme in denen magnetische Bänder aufbewahrt werden (Nelson, 2011, S. 17). In den Archiven gibt es ein regalartiges System, in welchem die Bänder und Laufwerke, mit welchen auf die Bänder geschrieben und von ihnen gelesen werden kann, gelagert werden. Über Roboter werden die Bänder in den Archiven sortiert, verschoben und in die Laufwerke eingelegt (Lüttgau, 2016, S. 18). Die Bänder gibt es in verschiedenen Ausführungen. Die gebräuchlichsten Ausführungen sind die folgenden (Nelson, 2011, S. 37-38):

- Digital Linear Tape (DLT)
- Linear Tape Open (LTO)
- Advanced Intelligent Tape (AIT) (Sony, 2016)
- StorageTek (STK) Txxxxx
- IBM 3490/3590

Der verwendete Standard ist heutzutage das LTO. Da die Laufwerke standardisiert sind, können in der Regel auch Drives von verschiedenen Herstellern in einem Archiv verwendet werden (Quantum Corporation, 2016).
Viele Archive besitzen horizontale Slots, in welchen die Bänder platziert werden. Eine platzsparendere Alternative wird beispielsweise von der Firma Spectralogic verwendet. Hier werden Regale verwendet, in welchen die Bänder horizontal eingeordnet werden. Dadurch wird im Vergleich zu konkurrierenden Firmen 50% des Platzes eingespart (Spectralogic, 2015, S. 3).

2.1.2. StorageTek SL8500

Für die Simulation mit dem Programm FeO, welches in Abschnitt 2.2 erläutert wird, wird angenommen, dass ein StorageTek SL8500 Bandarchiv verwendet wird. Im realen Leben findet sie in zahlreichen Einrichtungen Platz, wie zum Beispiel im Deutschen Klimarechenzentrum (DKRZ), wo insgesamt acht Bandarchive dieser Ausfertigung zu finden sind (DKRZ, 2016).
In Abbildung 2 ist eine schematische Darstellung eines einzelnen StorageTek SL8500 Bandarchives mit Beschriftung der wichtigen Elemente dargestellt.

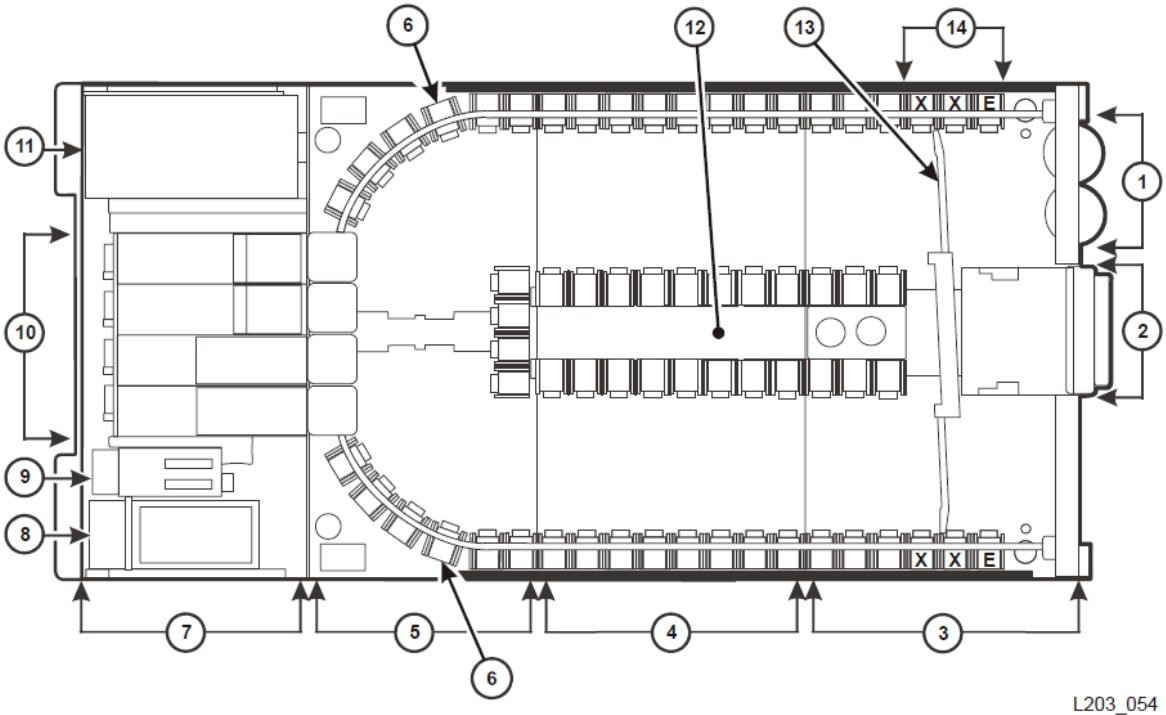


Abb. 2: Schematische Darstellung des StorageTek SL8500 Bandarchives mit Beschriftung (Oracle, 2010, S. 4).

Das StorageTek SL8500 bietet Platz für bis zu 6.632 Slots und bis zu 64 Laufwerke. Es können weitere Erweiterungen eingebaut werden, wodurch bis zu 10.088 Slots in einem Archiv beherbergt werden können (Oracle, 2010, S. 2). Ein Archiv kann bis zu acht Roboter beinhalten (Oracle, 2015, S. 1). An einem Archiv können weitere SL8500 Archive über Pass-thru Ports (PTPs) Verbindungen gekoppelt werden, um einen

Archivkomplex zu bilden. Über eine PTP Verbindung können die Roboter archivübergreifend Bänder bewegen. Dies kann nützlich sein, falls in einem Archiv alle Laufwerke belegt sind, aber ein Band in diesem Teil des Archivkomplexes ausgelesen werden soll (Oracle, 2010, S. 19). Insgesamt können bis zu 32 weitere Archive verbunden werden, welche insgesamt über 320.000 Slots und 2.048 Laufwerke beinhalten können (Oracle, 2010, S. 1).
In einem Archiv gibt es zwei Aufzüge, welche es den Robotern ermöglichen Bänder auf verschiedenen Höhen zu bewegen. Die Roboter selbst bewegen sich zwischen den Slots, den Laufwerken und den Bandausgabeslots (CAP) (Oracle, 2010, S. 6). Die Roboter besitzen zwei Motoren, so dass beim Ausfall eines Motors der zweite Motor den Roboter in den vorderen Bereich des Archives zur Entnahme bewegen kann. Sollten beide Motoren ausfallen, schiebt ein zweiter Roboter den defekten Roboter in den vorderen Bereich, wo er für die Reparatur entnommen werden kann. Mit Hilfe von zwei Kameras kann über eine Monitoring Software auf einem Computer das Innere des Bandarchives betrachtet werden (Oracle, 2010, S. 8).

2.1.3. Virtuelle Bandarchive

Bänder haben eine Lebenszeit von über 30 Jahren und sind damit an der Spitze, wenn es um die dauerhafte Datenspeicherung geht. In den vergangenen Jahren sind die Preise für Festplatten stark gesunken, so dass einige Unternehmen begannen ihre Speichersysteme auf Festplatten basieren zu lassen. Der Vorteil gegenüber Bandsystemen ist die geringere Zugriffszeit. Um die auf Festplatten basierenden Systeme kompatibel mit ansonsten veralteter Technologie zu machen, wurden die virtuellen Bandarchive (im Folgenden VTL, vom englischen Virtual Tape Library) entwickelt. Diese emulieren ein physisches Bandarchiv, so dass Hardwareprobleme im Zusammenhang mit Bandarchiven umgangen werden (QUADStor, 2016). Bei einer VTL werden, ebenso wie bei einem physischen Bandarchiv, die Daten über ein Fibre Channel Interface übertragen (EMC Corporation, 2016). Ein virtuelles Bandarchiv verhält sich in seinen Aktionen wie ein physisches Bandarchiv. Die VTL emuliert die Komponenten eines physischen Archives über die Ein- und Ausgabe-Ports so, dass sie nach außen genau wie ihre physischen Gegenstücke wirken. Beispielsweise können virtuelle Bänder in virtuelle Laufwerke geladen oder entnommen werden. Über ein physisches VTL-System können mehrere Archive erzeugt werden. Auf diese

Weise wird der Austausch von Bändern zwischen mehreren Archiven möglich (Nelson, 2011, S. 67). In Abbildung 3 wird dieses Verhalten schematisch dargestellt.

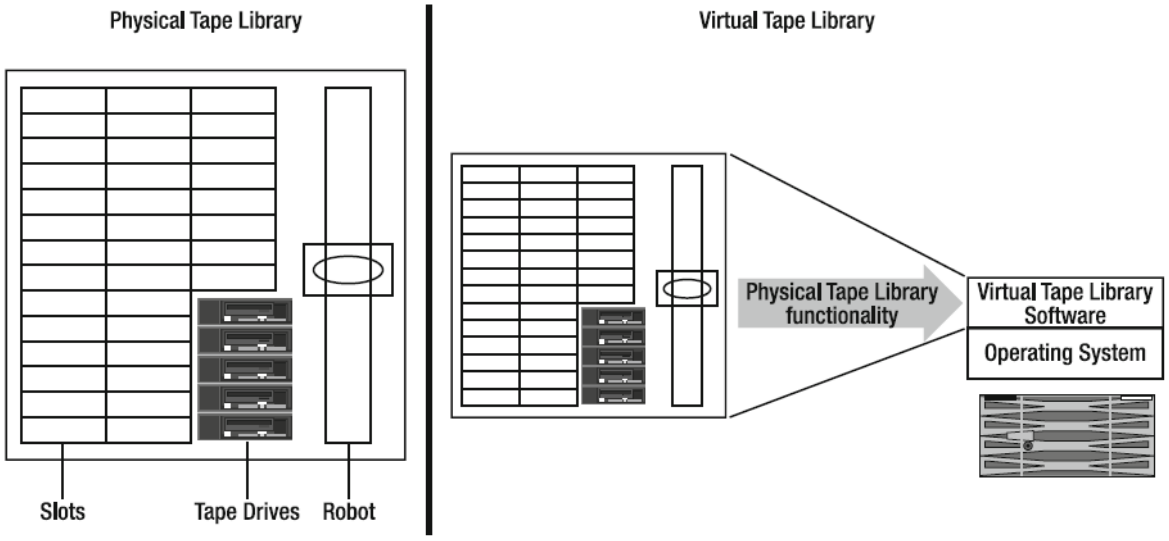


Abb. 3: Funktioneller Vergleich zwischen physischem und virtuellem Bandarchiv (Nelson, 2011, S. 68).

In Abbildung 3 ist zu erkennen, dass die Software der VTL physische Bandarchive nach außen so emuliert, wie sie als physisches Gerät aussehen würde. In der Software selbst sind die Komponenten lediglich Verbindungspunkte, welche auf die Anfragen von den I/O-Ports antworten. Diese Anfragen beinhalten einen Hardwareidentifikations-String, welcher bei den physischen Komponenten verwendet werden würde. Über diesen wird die richtige, von der Software erstellte, Komponente angesprochen. Die VTL antwortet mit denselben Antworten, welche die richtige Hardware verwenden würde. Durch diese detailgetreue Emulation, nimmt der Klient ein physisches Bandarchiv wahr. Auch die Kapazitäten des Archives werden standardmäßig wiedergegeben. Wenn ein Bandarchiv, welches lediglich 100 Slots besitzt, emuliert wird, hat auch die VTL in der Regel lediglich 100 Slots. Hier gibt es allerdings die Möglichkeit derartige Limitationen beliebig zu erweitern (Nelson, 2011, S. 69).

2.1.4. High Performance Storage System

Das High Performance Storage System (HPSS) ist eine von IBM entwickelte Software für hierarchische Speichersysteme, mit welcher einige Petabytes an Daten bei einer schnellen Datenübertragung, von einigen Gigabyte pro Sekunde, gemanagt werden können (Watson, 2005, S. 1). HPSS regelt den Lebenszyklus an Daten, indem lange nicht mehr verwendete Daten auf Tapes verschoben werden. Bei dem nächsten Aufruf

der Daten werden sie wieder neu erfasst (IBM, 2011, S. 1). In Abbildung 4 ist der schematische Aufbau des HPSS nach IBM dargestellt.

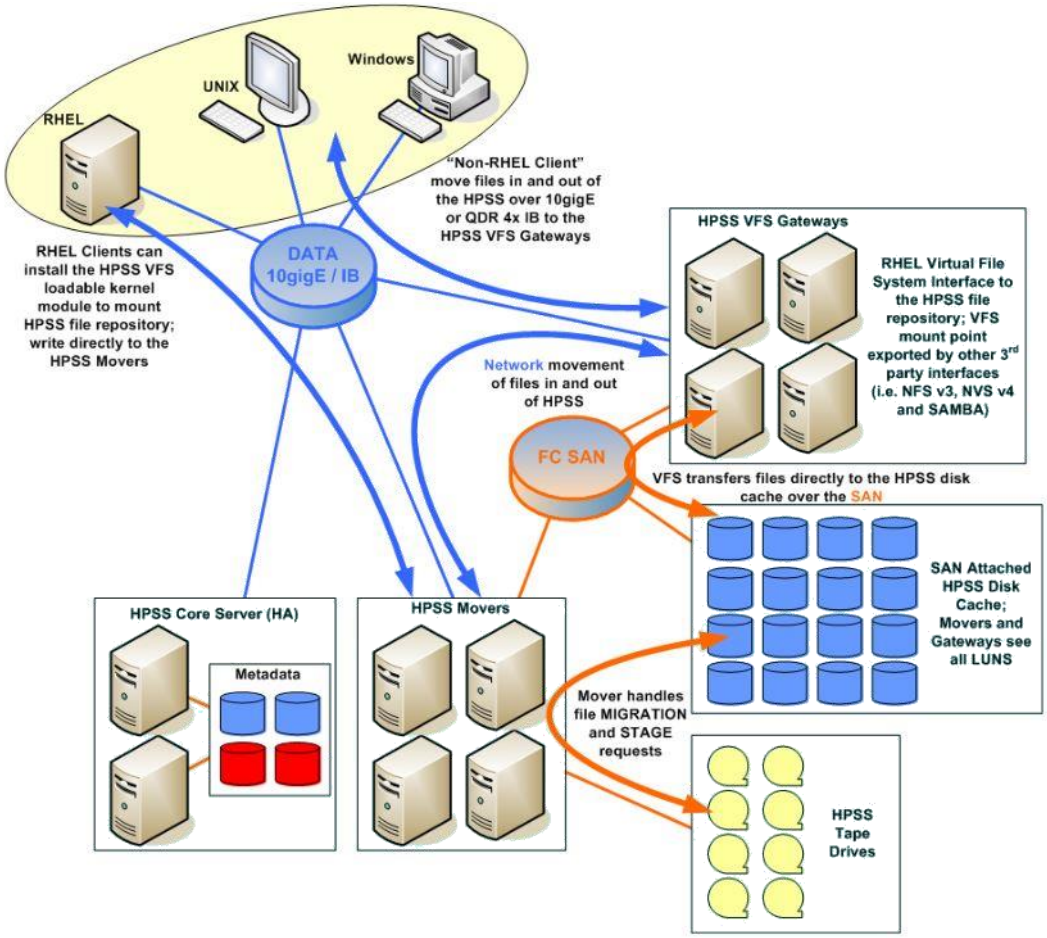


Abb. 4: Schematischer Aufbau des High Performance Storage Systems (IBM, 2011, S. 1).

HPSS vereint durch sein Cluster Design die Kapazitäten von mehreren Computerknoten in einem einzigen Speichersystem. Das Cluster kann beliebig groß sein, denn dem Klienten wird das Speichersystem immer als ein einzelnes Speichersystem angezeigt. Die Daten werden über die sogenannten *Movers* bereitgestellt. Sie werden dort in einem festplattenbasierenden Dateisystem gecached, um von Laufwerken und I/O Servern abgerufen zu werden. (IBM, 2011, S. 1).

Für Linux gibt es ein VFS Interface, welches POSIX read/write Operationen unterstützt. Dadurch kann HPSS als ein Linux Dateisystem verwendet werden. HPSS unterstützt die Verwendung von Hardware von Drittanbietern. So werden neben Systemen von IBM auch Systeme von Oracle StorageTek, Spectralogic, oder Quantum unterstützt (IBM, 2011, S. 2).

2.2. Simulationsprogramm FeO

Das Programm FeO[1] wurde von Jakob Lüttgau im Rahmen einer Masterarbeit geschrieben, um die hierarchischen Speichersysteme, speziell Bandarchive, moderner Datencenter zu simulieren. In diesem Unterkapitel wird das Simulationsprogramm beschrieben. Dafür wird zunächst die diskrete ereignisorientierte Simulation beschrieben, gefolgt von der verwendeten Programmiersprache, der Architektur, der Rolle der Nutzlast, der Simulation als solches und den Möglichkeiten für die Ausgabe diverser Daten.

2.2.1. Diskrete ereignisorientierte Simulation

Das Programm FeO führt die Simulationen nach dem Vorgehen einer diskreten ereignisorientierten Simulation (im Folgenden DES) aus. Die DES ist gut erforscht und wird häufig für die Simulation von technischen Systemen und Prozessen genutzt. In Abschnitt 2.2.2.1 ist das Prinzip der DES beschrieben. In Abschnitt 2.2.2.2 wird näher auf den Ablaufalgorithmus einer Simulation eingegangen.

2.2.1.1. Das Prinzip der diskreten ereignisorientierten Simulation

Bei der DES springt der Simulator von einem Ereignis am Zeitpunkt t_i zu dem nächsten Ereignis am Zeitpunkt t_{i+1}. Ein Ereignis kann dabei den aktuellen Zustand der Simulation verändern, sowie auch neue Ereignisse in der Zukunft erzeugen. In Abbildung 5 ist das Prinzip einer DES über die Zeit dargestellt.

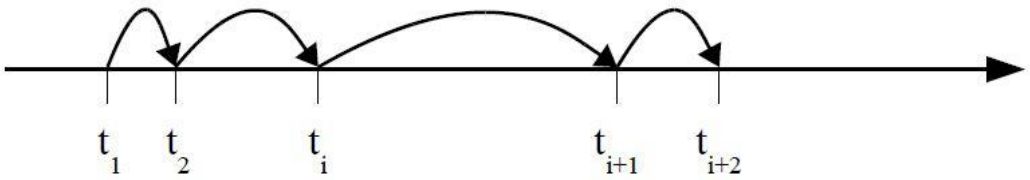


Abb. 5: Prinzip der diskreten ereignisorientierten Simulation (Wehrle, Günes & Gross, 2010, S. 3).

[1] Für die Beschichtung von magnetischen Tape Medien wird häufig Eisen(III)-oxid verwendet (Dee, 2008, S. 1777). Als Anspielung darauf erhielt das Programm seinen Namen FeO.

Die Ereignisse werden als Vormerkungen in einer Liste festgehalten, um die zukünftigen Ereignisse strukturiert auftreten zu lassen. Eine Vormerkung muss dabei mindestens aus den zwei Werten der Zeit, welche den Zeitpunkt festlegt, an welchem das Ereignis erzeugt werden soll, sowie dem Typ des Ereignisses bestehen (Wehrle, Günes & Gross, 2010, S. 3). Die Liste der zukünftigen Ereignisse ist dabei nach dem vorgesehenen Zeitpunkt für die Ereignisse sortiert. In Python wird dabei standardmäßig ein stabiler Algorithmus für die Sortierung verwendet. Sollten zwei Ereignisse planmäßig zu demselben Zeitpunkt initialisiert werden, wird nach dem First In First Out / FIFO-Prinzip (Platz & Østerdal, 2012, S. 1) als Erstes das Ereignis behandelt, welches zuerst in die Liste eingefügt wurde (Lüttgau, 2016, S. 61). Im Allgemeinen besitzt ein Simulator der DES die folgenden Komponenten (Wehrle, Günes & Gross, 2010, S. 3):

- Der *Zustand des Systems* besteht aus Variablen, welche den aktuellen Systemzustand beschreiben.
- Die *Uhr* gibt die aktuelle Uhrzeit der Simulation wieder.
- Eine *Ereignisliste* verwaltet alle zukünftigen Ereignisse.
- *Statistische Werte* geben Auskunft über die Performance des Systems.
- Eine *Initialisierungsfunktion* startet die Simulation und setzt die Uhrzeit auf 0.
- Eine *Ereignisfunktion* wird bei einem bestimmten Ereignis aufgerufen. In der Regel ist für jeden Typ von Ereignissen eine Ereignisfunktion definiert.

2.2.1.2. Der Simulationsalgorithmus

Die diskrete ereignisorientierte Simulation verläuft nach einem bestimmten Algorithmus, über welchen bestimmt wird, ob die Simulation beendet wird. Der Algorithmus ist in Abbildung 6 dargestellt.

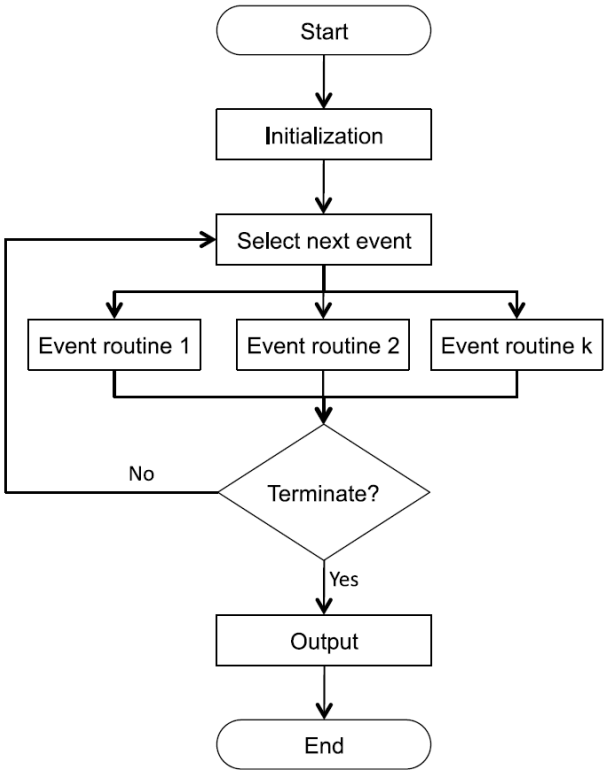


Abb. 6: Ablaufalgorithmus der diskreten ereignisorientierten Simulation (Wehrle, Günes & Gross, 2010, S. 4).

In Abbildung 6 ist zu erkennen, dass der Ablaufalgorithmus aus der Initialisierung, dem Ereigniskreislauf und der Ausgabe besteht. Wenn ein Ereignis auftritt, wird die jeweilige Ereignisfunktion aufgerufen, um das Ereignis entsprechend zu handhaben. Anschließend wird überprüft, ob eine Abbruchbedingung für die Simulation erfüllt ist. Wenn eine Abbruchbedingung erfüllt ist, wird die Ausgabe generiert und die Simulation wird beendet. Falls keine Abbruchbedingung erfüllt ist, wird das nächste Ereignis der Ereignisliste aufgerufen. Eine Simulation wird beendet, wenn die Simulation einen bestimmten Zeitpunkt erreicht hat, ein unerwartetes Verhalten bzw. ein Fehler auftritt oder wenn die Liste für die zukünftigen Ereignisse leer ist (Wehrle, Günes & Gross, 2010, S. 5). Bei dem Programm FeO werden laufende Statistiken schon während der Simulation gespeichert. Die Simulation wird hier normalerweise beendet, wenn die Ereignisliste leer ist. Die Ereignisse werden hier durch Anfragen an das Archiv repräsentiert (Lüttgau, 2016, S. 61).

2.2.2. Programmiersprache

Das Programm FeO wurde in der höheren Programmiersprache Python geschrieben (Lüttgau, 2016, S. 58). Verwendet wurde die Version Python 3.4. Python ist eine objektorientierte, frei zugängliche Programmiersprache. Sie wurde für die Softwarequalität, Entwicklerproduktivität, Verwendung in verschiedenen Entwicklungsumgebungen und Integration verschiedener Skripte, wie z. B. C / C++ – Skripte, optimiert. Python wird weltweit von vielen Entwicklern, unter anderem auch von Google und der NASA, für die Systemprogrammierung, Entwicklung von User Interfaces, die Webprogrammierung und mehr verwendet (Lutz, 2010, S. xxxix).

Bei der Programmierung von FeO wurde Python verwendet, da es sich als objektorientierte Sprache dafür eignet die physischen und logischen Komponenten in einzelnen Klassen zu modellieren. Python besitzt zudem eine breite Standardbibliothek und unterstützt eine große Bandbreite an Bibliotheken von Drittanbietern. Für FeO wurden beispielsweise die Bibliotheken NumPy und SciPy verwendet (Lüttgau, 2016, S. 58). Die Bibliothek NumPy verfügt über Operationen für die numerische Programmierung und fügt zu Python z. B. Vektorobjekte und weitere komplexe mathematische Operationen hinzu (Lutz, 2010, S. 750). Die Bibliothek SciPy baut auf NumPy auf. SciPy verfügt über Operationen um Daten zu manipulieren und zu visualisieren. Durch diese mathematische Bibliothek soll Python im Hinblick auf die Möglichkeiten der vergleichbar mit MATLAB, oder R-Lab sein (SciPy, 2016, S. 107).

Während der Ausführung des Programmes gehört Python zu den langsameren Sprachen. Dafür ist mit Python eine schnelle Softwareentwicklung möglich. Da Python die Einbindung von C / C++ – Skripten unterstützt, können Performanceprobleme an wichtigen Stellen durch deren Verwendung negiert werden. (Lüttgau, 2016, S. 58).

2.2.3. Architektur

FeO wurde als ein objektorientiertes Programm geschrieben. Für die Hauptkomponenten von hierarchischen Speichersystemen wurden Klassen modelliert. In Abbildung 7 ist ein UML Klassendiagramm des Programmes dargestellt.

Der *Simulation Kernel* führt die diskrete ereignisorientierte Simulation aus. Die Hauptkomponenten erben von der *Interface Komponente,* welches von dem Simulations Kernel verwendet wird. Auf Grund des Interfaces müssen sich die Komponenten in der Simulation registrieren und die Methode *next_action()* implementieren. Der Aufbau des Netzwerks und die Verknüpfungen der einzelnen Komponenten untereinander werden durch die *Netzwerk Topologie* festgelegt. Die in der Netzwerk Topologie registrierten Komponenten werden *Netzwerk Komponenten* genannt. Zu den Netzwerk Komponenten gehören der *Klient, IOServer* und das *Laufwerk.* Die Netzwerk Topologie kann durch eine XML-Datei geladen werden, dazu mehr in Kapitel 2.2.5.2. Während die Anfragen verarbeitet werden, werden die Teile des Netzwerkes miteinander verknüpft und anschließend wieder getrennt. Komponenten, welche in der *Archiv Topologie* registriert werden, werden *Archiv Komponenten* genannt. Diese werden verwendet, sobald ein Band in dem Archiv positioniert oder geladen werden soll. Zu den Archiv Komponenten gehört ebenfalls das *Laufwerk,* aber auch das *Band* und der *Roboter.*

Um die Emulation der Hardware möglichst gut von der Software zu trennen, wurden die *Software Komponenten* modelliert. Zu den Software Komponenten gehören der *Datei Manager, Band Manager* und der *CacheManager,* sowie der *IO Planer, Laufwerk Planer* und der *Roboter Planer.*

Mit dem *Persistenz Manager* können die mit der Simulation zusammenhängenden Daten gespeichert werden. Dadurch ist es möglich einzelne Simulationsdurchläufe zu wiederholen oder Daten zur Auswertung für bestimmte Problematiken geordnet abzuspeichern. Außerdem können die einzelnen Komponenten der Simulation mit Hilfe des *Berichte Managers* CSV-Dateien anlegen, in welchen die Datenwerte gespeichert werden. Die Daten einer Simulation werden separat in einem für diese Simulation eindeutigen Ordner gespeichert (Lüttgau, 2016, S. 59).

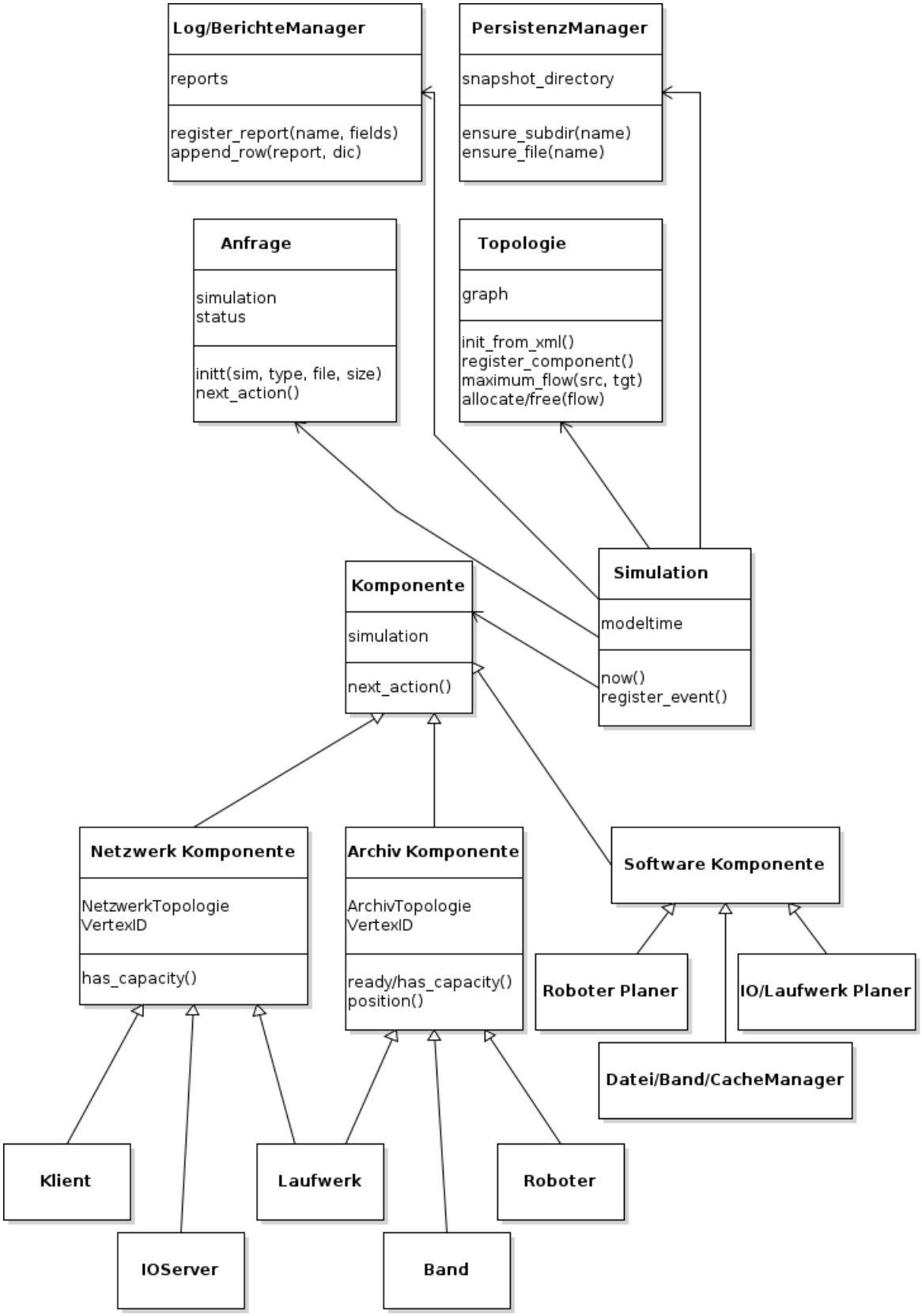


Abb. 7: UML Klassendiagramm des Programmes FeO (Lüttgau, 2016, S. 60).

2.2.4. Nutzlast

Es können Anfrage-Objekte initialisiert und anschließend der Simulation hinzugefügt werden. Auf diese Weise verrichtet das virtuelle Bandarchiv Arbeit. Mit Hilfe von Ereignis Providern werden die geplanten Ereignisse, bzw. Anfragen, erzeugt und der Simulation hinzugefügt. Dies geschieht so lange, bis alle geplanten Anfragen ausgeführt wurden und die Liste der zukünftigen Ereignisse somit leer ist (Lüttgau, 2016, S. 62). FeO unterstützt auch die Verwendung von echten Nutzlasten. Hierfür müssen von einem Administrator die auf einem System durchgeführten Anfragen in sogenannten Spurdaten[2] aufgezeichnet werden. Dabei können Spurdaten im xferlog Format verwendet werden. Hierfür wird der implementierte *Xferlog Provider* verwendet. Um Fehler zu vermeiden, erstellt dieser nur bekannte Arten von Anfragen, alle anderen in der xferlog-Datei auftretenden Arten werden verworfen. Es werden insgesamt vier Arten von Anfragen unterstützt, welche in *lesende* und *schreibende* Anfragen aufgeteilt werden. Die für die *lesenden* Anfragen zulässigen Typen sind *PST_Cmd* und *STOR_Cmd*, die für die *schreibenden* Anfragen sind *PRTR_Cmd* und *RETR_Cmd* (Lüttgau, 2016, S. 62).

Beim Start einer Simulation werden zunächst die Spurdaten analysiert. Auf dessen Basis wird anschließend das Dateisystem und das Bandarchiv erstellt. Der *Xferlog Provider* fügt dabei unbekannte Komponenten als Netzwerk Komponente zu einem bestimmten Knoten der Netzwerk Topologie hinzu. Dabei werden die Verbindungen automatisch beidseitig angelegt. Den Komponenten können in der XML-Datei verschiedene Bandbreiten zugewiesen werden. Basierend auf dem jeweiligen Stand der Technik, wird in der Regel ein Standardwert gesetzt (Lüttgau, 2016, S. 62).

[2] In diesem Fall repräsentieren die Spurdaten die Details zu den Anfragen, welche an das System gesendet werden oder wurden.

2.2.5. Simulation

Um eine Simulation zu konfigurieren wurden die in Abschnitt 2.2.5.1 behandelten Kommandozeilen Argumente implementiert. In Abschnitt 2.2.5.2 wird die Möglichkeit beschrieben die verwendete Netzwerk Topologie als XML-Datei zu speichern oder diese auch aus einer XML-Datei zu laden.

2.2.5.1. Kommandozeilen Argumente

Mit Hilfe von den Kommandozeilen Argumenten können schnelle Änderungen der Konfiguration vorgenommen werden. Um den Spurdaten Provider zu konfigurieren können folgende Argumente verwendet werden (Lüttgau, 2016, S. 63):

-- tracefile <Dateiname>	gibt die zum Laden der Spurdaten zu verwendende Datei an
-- limit <Zahl>	legt die maximale Größe der Spurdaten fest

Um die Topologie der Simulation manuell festzulegen können folgende Argumente verwendet werden (Lüttgau, 2016, S. 63):

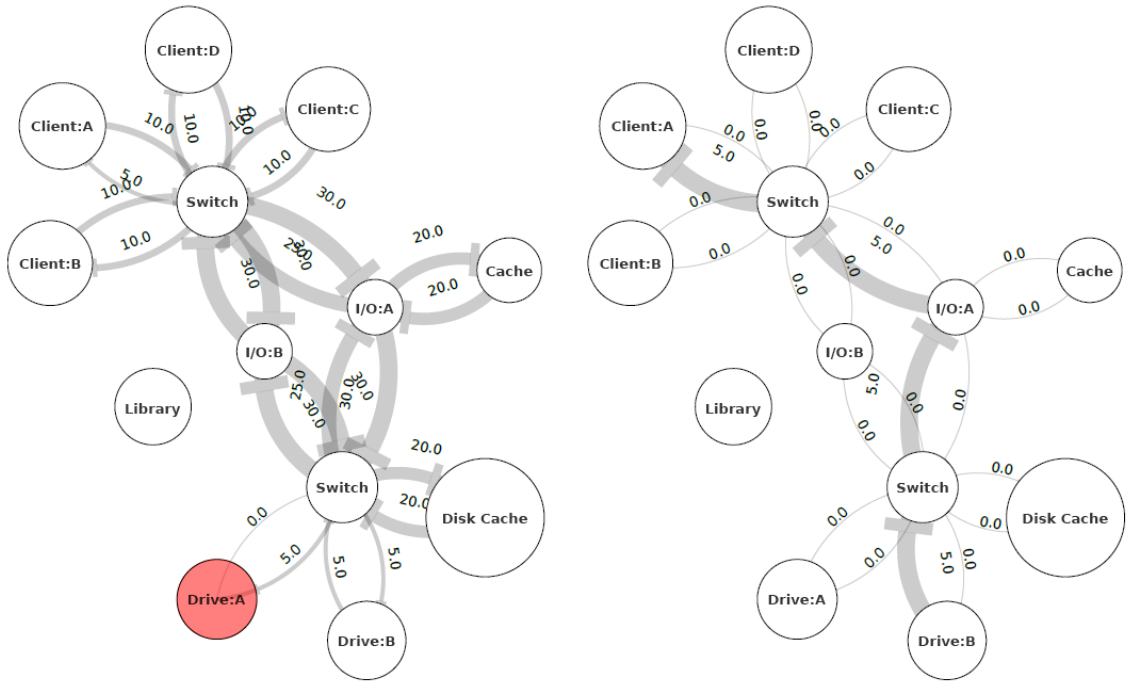
-- networktopology <Dateiname>	gibt die Datei zum Laden der Netzwerk Topologie an
-- librarytopology <Dateiname>	gibt die Datei zum Laden der Archiv Topologie an
-- drives <Zahl>	legt die Anzahl der Laufwerke fest

Zusätzlich kann mit dem folgenden Argument manuell durch die einzelnen Simulationsschritte gesprungen werden (Lüttgau, 2016, S. 63):

-- confirm-step

2.2.5.2. XML-Dateien als Ausgangspunkt der Netzwerk Topologie

In FeO können die Topologien über eine XML-Datei initialisiert werden. Das Netzwerk wird von einem Graphen repräsentiert, welcher die Verknüpfung der physischen Komponenten darstellt. Jede Komponente verfügt über Methoden, um die Kapazität einer Komponente abzufragen. Einzelne Teile der Netzwerk Kapazität können reserviert und wieder freigegeben werden. Die Verknüpfung wird über einen Pfad von der Quelle zum Ziel dargestellt. In Abbildung 8 ist dieses Vorgehen anhand einer einfachen Konfiguration mit einem verwendeten Laufwerk dargestellt. Zusätzlich wird der Datenstrom von einem freigegebenen Laufwerk zu einem Klienten dargestellt. Diese Art wurde als Annäherung an das paketorientierte Arbeiten eines Netzwerkes gewählt (Lüttgau, 2016, S. 63).



(a) Der aktuelle Zustand des Systems, Drive : A ist in Verwendung, dargestellt durch die Färbung.

(b) Ein Datenstrom von Drive : B zu Client : A für den links dargestellten Zustand.

Abb. 8: Beispiel einer Netzwerk Topologie (Lüttgau, 2016, S. 64).

FeO verwendet die externe Bibliothek *Graph-tool*. Graph-tool ist eine Bibliothek für Python für die Manipulation und die statistische Analyse von Graphen, insbesondere für Netzwerke. Die Kerndatenstrukturen und Algorithmen wurden in C++ geschrieben. Auf diese Weise ist die Performance vergleichbar mit einer reinen C/C++ Bibliothek (Peixoto, 2015).

2.2.6. Datenausgabe

Bei dem Programm FeO gibt es die Möglichkeit für jeden Simulationsdurchlauf diverse Ausgabewerte zu speichern. Für jeden Simulationsdurchlauf wird ein neuer Ordner erstellt, welcher mit einem Zeitstempel signiert ist. In diesem werden die Daten gespeichert, welche es erleichtern sollen die Simulation zu analysieren (Lüttgau, 2016, S. 65).
Zunächst gibt es die Möglichkeit Daten über Anfragen zu speichern. Diese werden als CSV-Dateien gespeichert und geben unter Anderem Auskunft über den Durchsatz an Anfragen, der Größe der angefragten Informationen, Wartezeiten oder die Stages. Da es sich hier um ein Simulationsprogramm handelt, können auch Systemdaten gespeichert werden, welche von einem echten System nicht geliefert werden können. Dafür müssten zum Beispiel weitere, unter Umständen teure, Sensoren verbaut werden. Dadurch ist es dem Nutzer möglich, eine ungefähre Vorstellung von speziellen Vorgängen zu erhalten, welche er auf andere Weise nicht bekommen kann.
Um den Status der Simulation zu analysieren, gibt es Daten zum Status des Dateisystems, des Bandsystems und des globalen Cachesystems. Um die Konfiguration des HSM/Band Systems einzusehen, werden die Netzwerk- und Archiv Topologien als XML-Dateien abgespeichert. Wenn aktiviert, kann auch ein Log des Simulationsprozesses gespeichert werden. (Lüttgau, 2016, S. 65).

Zusammenfassung

Es gibt eine Vielzahl an Speichermedien, welche sich in ihren Spezifikationen im Hinblick auf die Zugriffszeit, aber auch auf ihre Kosten stark unterscheiden. Sie lassen sich daher in Hierarchien anordnen. Bandarchive sind ein Beispiel für Systeme, in welchen mehrere Speichermedien miteinander kombiniert werden. Hier gibt es einen Cache, welcher auf Festplatten basiert und Bänder, welche die kostengünstigere Variante mit mehr Speicherplatz darstellen, dafür aber auch eine längere Zugriffszeit besitzen. Virtuelle Bandarchive emulieren physische Bandarchive. Auf diese Weise ist es möglich sein System vollständig von Bändern auf die immer günstigeren Festplatten umzustellen und trotzdem nicht auf weitere Soft- und Hardware verzichten zu müssen, welche nur im Zusammenhang mit Bandarchiven funktionieren. HPSS ist eine von IBM

entwickelte Software für hierarchische Speichersysteme, mit welcher einige Petabytes an Daten bei einer schnellen Datenübertragung gemanagt werden können. HPSS vereint durch eine Cluster-Struktur die Rechenleistung mehrerer Rechner. Dem Nutzer wird die Gesamtzahl der Rechner nur als ein einzelnes Speichersystem angezeigt. Mit dem Simulationsprogramm FeO ist man in der Lage diese hierarchischen Speichersysteme zu simulieren. FeO wurde in Python geschrieben und verwendet das Prinzip der diskreten ereignisorientierten Simulation. Die normalerweise üblichen Ereignisse werden hier von Anfragen repräsentiert. Die Topologie des zu repräsentierenden Netzwerkes kann von einer XML-Datei geladen werden. Da es sich um ein Simulationsprogramm handelt, können die benötigten Daten beispielsweise in einer CSV-Datei gespeichert werden. Man kann mit Hilfe der Simulation auch auf Daten schließen, welche man mit einem realen System nicht erhalten kann.

3. Evaluation

In diesem Kapitel werden Szenarien erschaffen, die mit dem Simulationsprogramm FeO simuliert und ausgewertet werden. Die inneren Vorgänge in hierarchischen Speichersystemen werden auf diese Weise näher betrachtet. In Abschnitt 3.1 wird zunächst der grundlegende Versuchsaufbau beschrieben. In Abschnitt 3.2 wird die Bedeutung und das Verhalten des Cache untersucht. In Abschnitt 3.3 wird das Verhalten des hierarchischen Speichersystems bei Veränderung ausgewählter Komponenten auf die Verarbeitungszeit von eingehenden Anfragen evaluiert.

3.1. Grundlegender Versuchsaufbau

Für die durchgeführten Szenarien wurde stets derselbe grundlegende Versuchsaufbau verwendet. Es wurde ein vereinfachtes System konfiguriert. Die Netzwerktopologie wird in Abbildung 9 dargestellt.

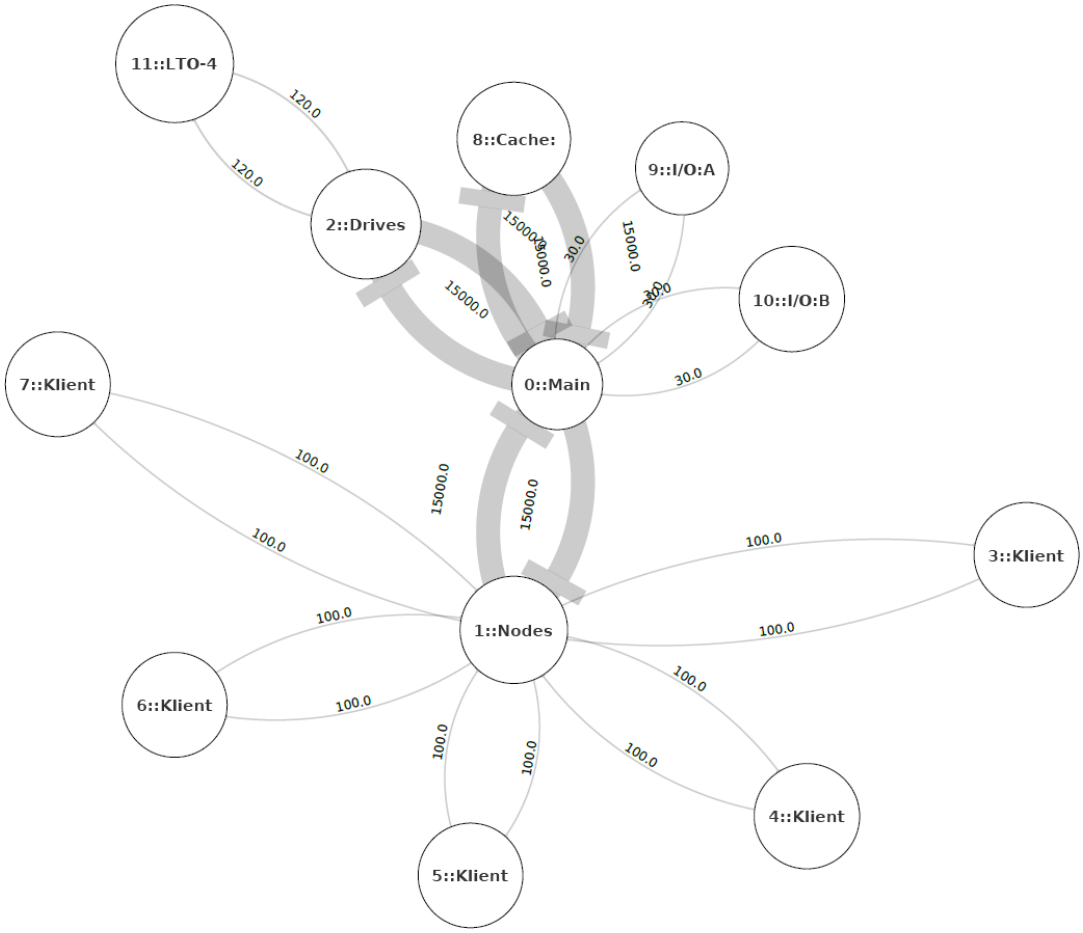


Abb. 9: Die für die Szenarien verwendete grundlegende Netzwerktopologie.

Die verwendete Topologie besteht aus einem Laufwerk der Generation *LTO-4*, einem Cache und fünf Klienten, welche über Switches miteinander verbunden sind. Die jeweilige Bandbreite in MB zwischen den Komponenten ist Abbildung 9 zu entnehmen. Zu finden sind sie auf den Verbindungslinien zwischen den jeweiligen Netzwerkknoten. Als Nutzlast wurden an das Bandarchiv von unterschiedlichen Klienten fünf lesende Anfragen zu unterschiedlichen Dateien gesendet, die zur Vereinfachung je eine Größe von 50 GB besitzen.
Für einzelne Szenarien wurden Veränderungen an diesem Versuchsaufbau vorgenommen, welche an den entsprechenden Stellen beschrieben werden.

3.2. Cacheverhalten

In diesem Versuch soll das Verhalten des Cache untersucht werden. Dafür wird zunächst der Unterschied im Verhalten bei der Bearbeitung von Anfragen mit gecachten und nicht gecachten Informationen analysiert. Im zweiten Schritt wird ermittelt, wie der Cache intern reagiert, wenn seine Kapazität ausgereizt ist und trotzdem weiter neue Daten aufgenommen werden sollen.

3.2.1. Gecachte und nicht gecachte Informationen

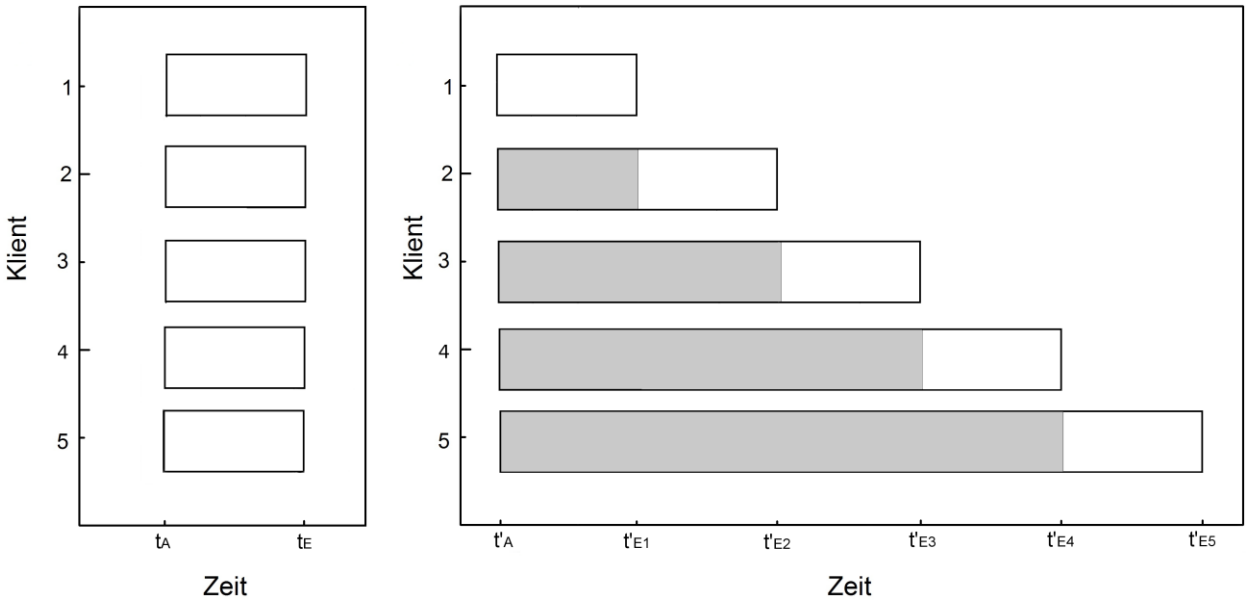
Wenn an ein Bandarchiv eine lesende Anfrage gesendet wird, gibt es zwei mögliche Zustände:

- Die gewünschte Information ist bereits im Cache vorhanden.
- Die gewünschte Information ist noch nicht im Cache vorhanden.

Der Cache ist zu Beginn der Simulation leer. Die Simulation wird mit der in Abschnitt 3.1 beschrieben Nutzlast durchgeführt. Im selben Simulationsdurchlauf wird die gleiche Nutzlast zu einem späteren Zeitpunkt erneut an das Bandarchiv gesendet. Die Anfragen treffen jeweils zu demselben Zeitpunkt im System ein. Die Anfragen sind zunächst an Dateien gerichtet, welche sich nicht im Cache befinden. Bei der zweiten Ausführung der Nutzlast befinden sich die angefragten Dateien bereits im Cache.

3.2.1.1. Erwartungen

In Abbildung 10 ist das Ausgabeverhalten von gewünschten Informationen an den Klienten für bereits gecachte und nicht gecachte Informationen dargestellt. Das Bandarchiv besitzt ein Laufwerk und es werden fünf lesende Anfragen zeitgleich an das System gesendet.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf gecachte Dateien.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien.

Abb. 10: Vergleich vom Ausgabeverhalten an Klienten von gecachten und nicht gecachten Informationen bei einem Bandarchiv mit einem Laufwerk.

In Abbildung 10a ist das Ausgabeverhalten für bereits gecachte Informationen dargestellt. Fünf Klienten senden zum Zeitpunkt t_A eine lesende Anfrage an das System. Da die gewünschten Informationen bereits im Cache vorhanden sind, können diese direkt an die Klienten weitergegeben werden, so dass die Anfragen gleichzeitig zum Zeitpunkt t_E abschließen. In Abbildung 10b ist das Ausgabeverhalten für noch nicht gecachte Informationen dargestellt. Die Informationen werden hier vor der Ausgabe zunächst von einem Band in ein Laufwerk geladen, in den Cache geschrieben und letztendlich an den Klienten ausgegeben. Hier werden zum Zeitpunkt t'_A fünf lesende Anfragen an das Bandarchiv gesendet. Da hier von einem System mit einem Laufwerk ausgegangen wird, müssen die weiteren Anfragen warten, bis die erste Anfrage abgeschlossen wurde. Die jeweilige Wartezeit wird in der Abbildung durch den grauen

Bereich dargestellt. Zum Zeitpunkt t'_{E1} wird die erste Anfrage abgeschlossen, ab hier beginnt die Bearbeitungszeit der zweiten Anfrage. Diese ist zum Zeitpunkt t'_{E2} abgeschlossen und ab hier beginnt die Bearbeitungszeit der dritten Anfrage. Dieses Verhalten lässt sich analog auf n Klienten mit n Anfragen übertragen. Die Verarbeitungszeit t_V für eine bestimme Anfrage lässt sich nach Gleichung 1 berechnen.

$$t_V = t_E - t_A \tag{1}$$

In Gleichung 1 sind: t_V Verarbeitungszeit [s]

t_E Zeit bei Abschluss der Anfrage [s]

t_A Zeit bei Eingang der Anfrage [s]

Aus dem in Abbildung 10b beschriebenen Verhalten ergibt sich das in Abbildung 11 dargestellte Histogramm.

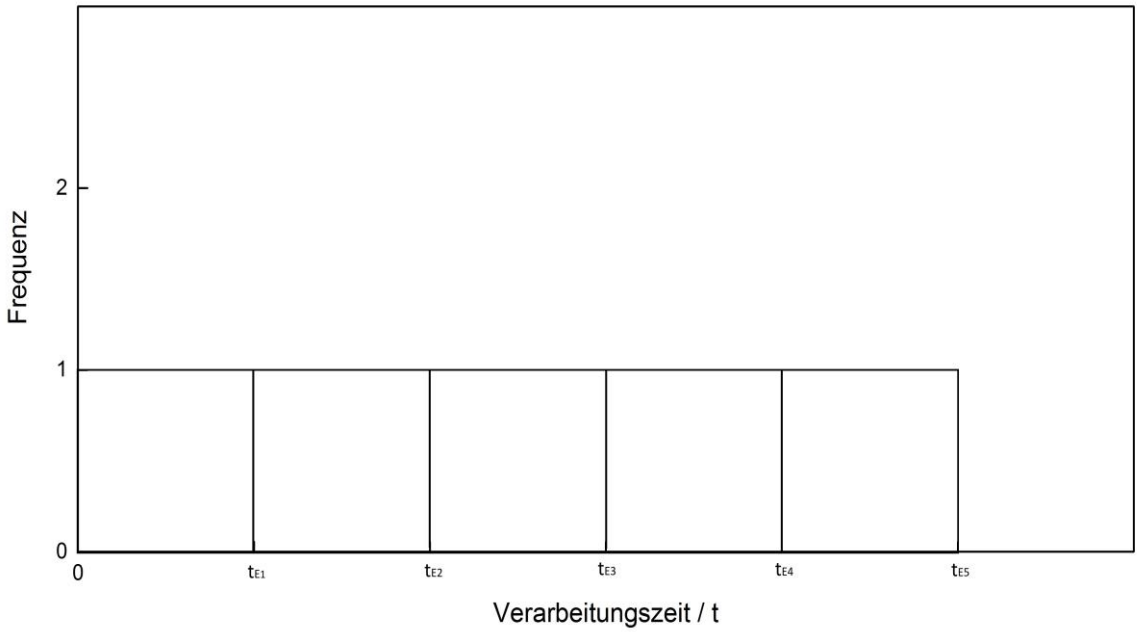


Abb. 11: Histogramm für die Bearbeitung von Anfragen eines Bandarchives.

In Abbildung 11 ist die Frequenz der bearbeiteten Anfragen gegen die Verarbeitungszeit für ein Bandarchiv mit einem Laufwerk aufgetragen. Insgesamt werden fünf Anfragen für jeweils noch nicht gecachte Informationen bearbeitet. Zum Zeitpunkt 0 beginnt die Bearbeitung der ersten Anfrage, zum Zeitpunkt t_{E1} ist diese abgeschlossen. Hier beginnt die Bearbeitung der zweiten Anfrage, welche zum Zeitpunkt t_{E2} abgeschlossen ist. Analog verhält sich die Bearbeitung der weiteren Anfragen. Aus dem Histogramm geht hervor, dass zu jedem Zeitpunkt lediglich eine Anfrage zurzeit bearbeitet wird.

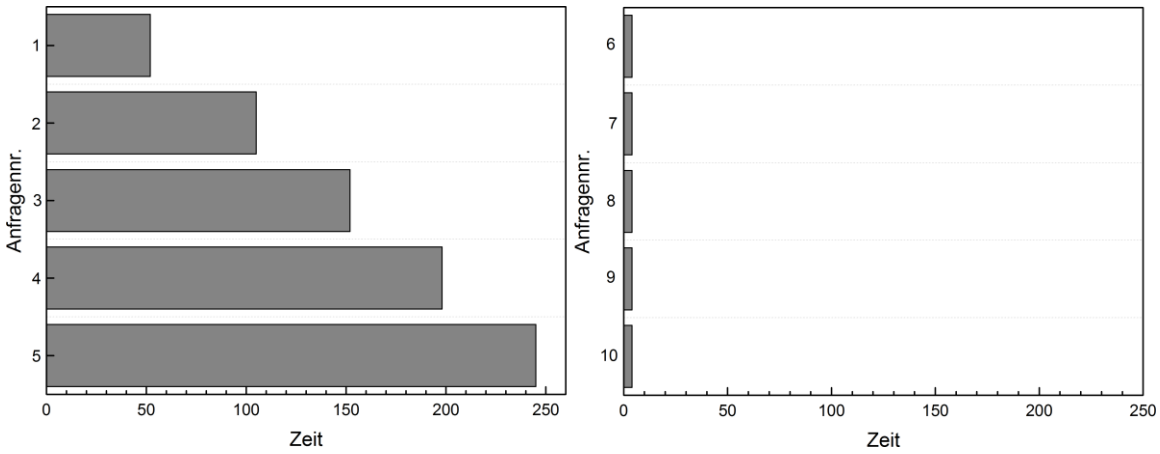
3.2.1.2. Daten und Auswertung

Es wurden zwei Simulationen durchgeführt. Beim ersten Durchlauf bezogen sich die lesenden Anfragen auf Dateien, welche sich bereits im Cache befanden. Beim zweiten Durchlauf wurde nach Dateien gefragt, die noch nicht im Cache gespeichert wurden. In Tabelle 1 sind die Daten für die Zeit des Eingangs der Anfrage, die Zeit an welcher die Anfrage abgeschlossen wurde und die nach Gleichung 1 berechnete Verarbeitungszeit aufgeführt.

Tab. 1: Daten für den Vergleich von gecachten und nicht gecachten Dateien: Zeit bei Eingang der Anfrage, Zeit bei Fertigstellung der Anfrage und die berechnete Verarbeitungszeit in Sekunden.

	$Zeit_{Eingang}$	**$Zeit_{Beendet}$**	**Verarbeitungszeit**
Nicht im Cache	00:00:03	00:01:01	52 s
	00:00:04	00:01:49	105 s
	00:00:05	00:02:37	152 s
	00:00:06	00:03:24	198 s
	00:00:07	00:04:12	245 s
Im Cache	01:00:03	01:00:07	4 s
	01:00:04	01:00:08	4 s
	01:00:05	01:00:09	4 s
	01:00:06	01:00:10	4 s
	01:00:07	01:00:11	4 s

Aus Tabelle 1 ergibt sich Abbildung 12, in welcher die Anfragen gegen ihre Verarbeitungszeit aufgetragen wurden.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf gecachte Dateien.

Abb. 12: Verarbeitungszeit für gecachte und nicht gecachte Dateien.

In Abbildung 12a sind die Verarbeitungszeiten in Sekunden für lesende Anfragen auf nicht gecachte Dateien dargestellt. Es ist zu erkennen, dass die Erwartungen aus Abbildung 10b erfüllt werden. Die Anfragen werden mit einer Differenz von einer Sekunde an das Bandarchiv gesendet. Ab hier benötigt die erste Anfrage 52 Sekunden bis zum Abschluss der Verarbeitung. Die zweite Anfrage benötigt insgesamt 105 Sekunden für die vollständige Verarbeitung. Hier ist zu berücksichtigen, dass diese für ihre Bearbeitung auf den Abschluss der Verarbeitung der ersten Anfrage warten muss. Die zweite Anfrage besitzt also eine Grundwartezeit von 52 Sekunden und somit eine effektive Verarbeitungszeit von 53 Sekunden. Für die Verarbeitungszeit der weiteren Anfragen auf nicht gecachte Dateien gilt dasselbe Prinzip. In Abbildung 12a ist zu erkennen, dass die Differenz der effektiven Verarbeitungszeiten nicht für jede Anfrage einheitlich ist, obwohl jede angefragte Datei eine Größe von 50 GB besitzt. Der Grund hierfür wird sein, dass die Roboter des Bandarchives verschiedene Zeiten benötigen, um die Bänder der jeweiligen Datei zum Laufwerk zu transportieren.

Im Anschluss wurden dieselben lesenden Anfragen erneut an das Bandarchiv gesendet. Die Verarbeitungszeiten für diese sind in Abbildung 12b dargestellt. Die angefragten Dateien waren zu diesem Zeitpunkt bereits im Cache gespeichert. Aus diesem Grund konnten diese direkt aus vom Cache an den Klienten gesendet werden und es war nicht nötig, die angefragten Dateien zuerst von einem Band einzulesen. Aus diesem Grund betrugen die Verarbeitungszeiten pro Anfrage jeweils 4 Sekunden. Der Wert der Verarbeitungszeit verändert sich nicht, da in diesem Fall jede Datei dieselbe Größe besitzt. In diesem Szenario wurde gezeigt, dass der Unterschied zwischen lesenden Anfragen auf gecachte und nicht gecachte Dateien einen starken Unterschied im Hinblick auf die Verarbeitungszeit und damit auf die Dauer, die der Klient auf seine Informationen warten muss, hat.

3.2.2. Datenmanagement eines vollen Cache

In diesem Versuch wird das Verhalten des Datenmanagements eines vollen Cache überprüft. Der Cache besitzt in diesem Szenario eine Größe von 100 GB, also genau das doppelte der Größe der Dateien. Auf diese Weise ist sichergestellt, dass lediglich zwei Dateien zeitgleich in den Cache geschrieben werden können. Es werden zehn lesende Anfragen für zehn verschiedene Dateien generiert. In jedem Teilschritt der Simulation wird nachvollzogen, welche Dateien sich aktuell im Cache befinden.

3.2.2.1. Erwartungen

Das angenommene Verhalten ist in Abbildung 13 dargestellt. In diesem Versuch wird davon ausgegangen, dass alle Dateien die gleiche Größe besitzen und die Kapazität des Cache auf zwei Dateien begrenzt ist.

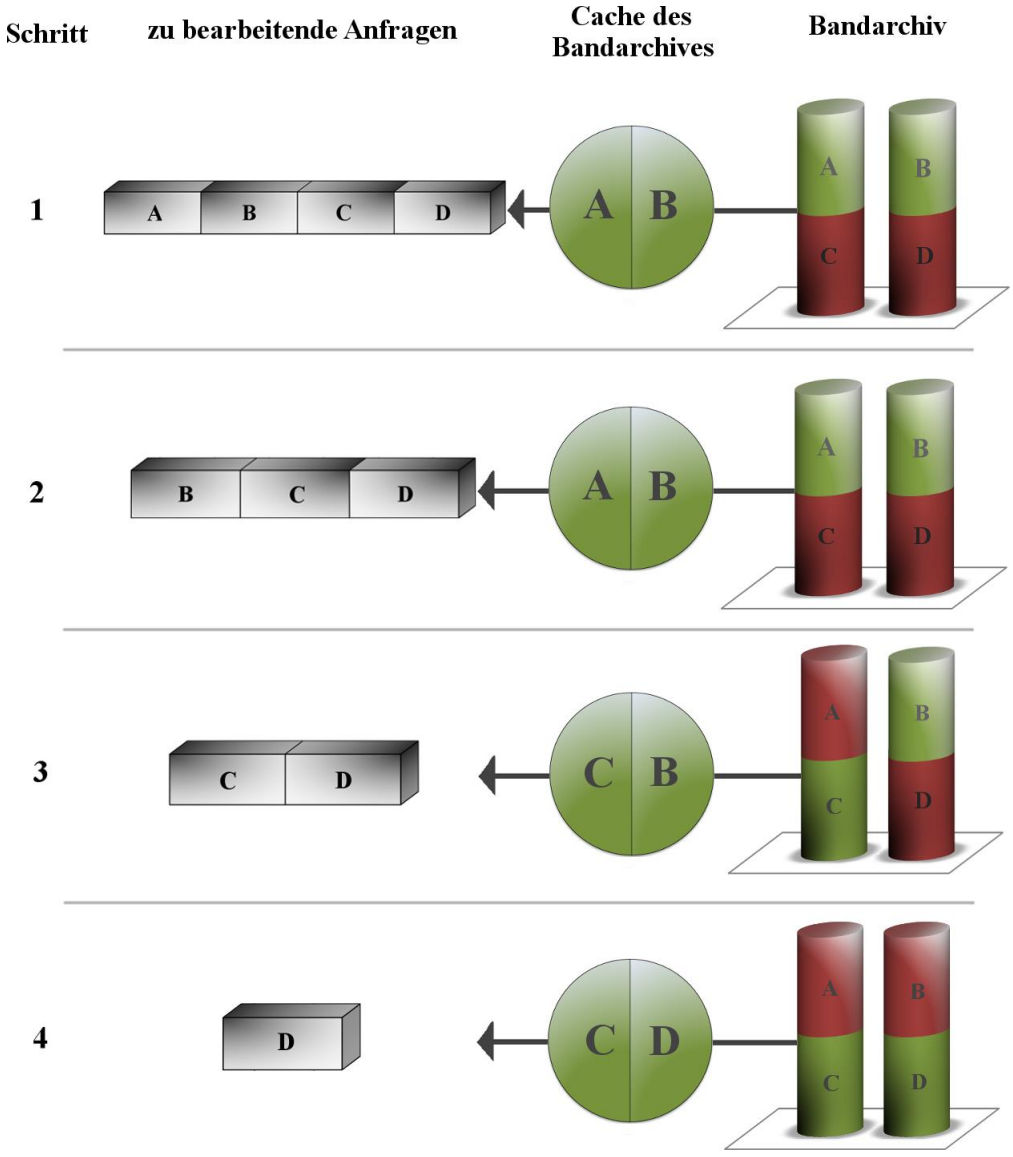


Abb. 13: Verhalten eines vollen Cache bei weiteren Anfragen.

In diesem Beispiel werden an das System vier lesende Anfragen gesendet. In Abbildung 13 werden diese durch die linke Spalte repräsentiert und sind in die Anfragen nach den Dateien *A, B, C* und *D* unterteilt. Die im Bandarchiv enthaltenen Dateien werden durch die rechte Spalte repräsentiert. In der mittleren Spalte ist der Cache des Bandarchives dargestellt. In den Cache passen lediglich zwei Dateien. Die Anfragen werden sukzessiv bearbeitet. Im ersten Schritt wurden bereits die Dateien *A* und *B* im Cache gespeichert. Zuerst wird die Anfrage nach Datei *A* bearbeitet und die entsprechende Information wird direkt aus dem Cache heraus an den Klienten gesendet. Diese Anfrage wird nun aus der Liste gestrichen. In Schritt zwei wird zunächst die Anfrage nach Datei *B* beantwortet und ebenfalls direkt aus dem Cache gelesen und an den Klienten gesendet. Diese Anfrage wird nach Beendigung auch aus der Liste gestrichen. Im dritten Schritt soll die Anfrage nach Datei *C* beantwortet werden. Datei *C* befindet sich zu diesem Zeitpunkt noch nicht im Cache und muss daher aus dem Dateisystem zuerst in den Cache geschrieben werden. Der Cache ist bereits voll, daher wird Datei *A* aus dem Cache entfernt und mit Datei *C* überschrieben. Nun kann die Anfrage beantwortet werden und ebenfalls aus der Liste gestrichen werden. Im letzten Schritt soll die Anfrage nach Datei *D* beantwortet werden. Diese befindet sich zu Beginn nicht im Cache und wird über das Bandarchiv in den Cache geschrieben. Hier wird nun Datei *B* aus dem Cache entfernt und mit Datei *D* überschrieben. Nach Beendigung wird auch hier diese Anfrage aus der Liste gestrichen.
Das Bandarchiv bleibt stets unverändert, da die Daten nicht aus diesem gelöscht werden. Weitere Anfragen werden nach dem selben Vorgang bearbeitet.

3.2.2.2. Daten und Auswertung

Es wurde eine Simulation mit dem grundlegenden Versuchsaufbau durchgeführt. In jedem Simulationsschritt wurden die im Cache befindlichen Dateien nachvollzogen. In Tabelle 2 sind die im Cache befindlichen Dateien zu dem jeweiligen Simulationsschritt aufgeführt.

Tab. 2: Daten für das Datenmanagement eines vollen Cache. Simulationsschritt und zugehörige, im Cache befindliche Dateien.

Simulationsschritt	Dateien im Cache	
1	A	-
2	B	A
3	B	C
4	D	C
5	D	E
6	F	E
7	F	G
8	H	G
9	H	I
10	J	I

In Tabelle 2 wurden die Werte für die Dateien im Cache in der Reihenfolge notiert, wie sie von dem simulierten Cache ausgegeben wurden. Es ist zu erkennen, dass sich der bereits volle Cache bei weiterer Datenaufnahme wie erwartet verhält. Die Simulation startet mit einem leeren Cache. Daher befindet sich nach dem ersten Simulationsschritt lediglich Datei *A* im Cache. Im zweiten Schritt befinden sich die Dateien *B* und *A* im Cache. Interessant ist hier, dass *A* an zweite Stelle der Liste gerückt wurde. Der Grund ist, dass der Cache zuerst seine Liste auffüllt und neue Dateien an den Anfang der Liste gesetzt werden. In einem realen System wäre diese Speicherverwaltung problematisch, da das stetige verschieben des bereits Gespeicherten unnötig Ressourcen und Zeit verbrauchen würde. Im dritten Simulationsschritt befinden sich die Dateien *B* und *C* im Cache. Hier wurde die schon länger im Cache befindliche Datei *A* mit *C* überschrieben. In der weiteren Simulation wurden sukzessiv die am längsten im Cache befindliche Datei mit der neu zu speichernden überschrieben.

3.3. Verhalten bei Veränderung verschiedener Komponenten

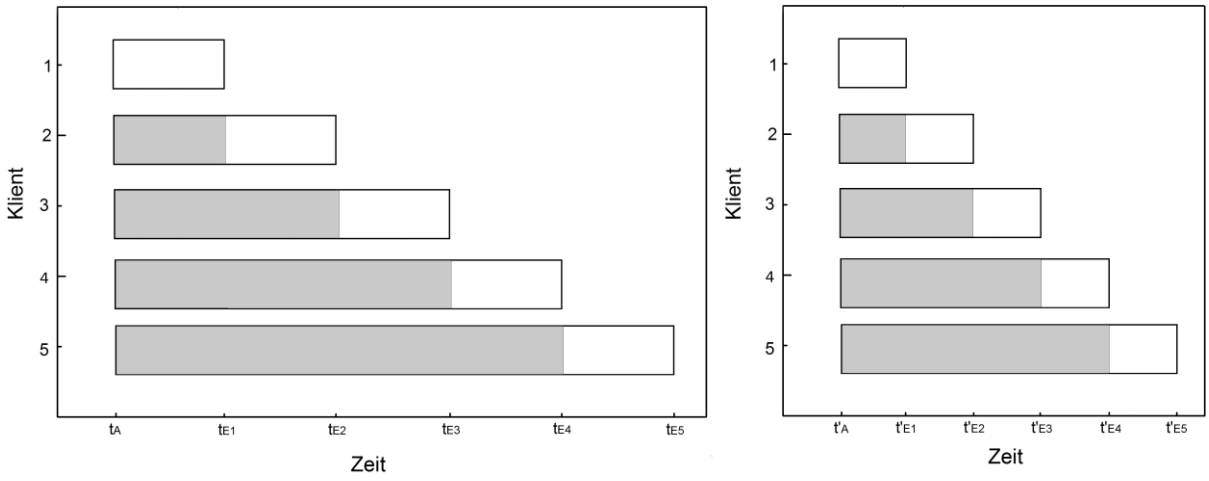
In diesem Versuch wird das Verhalten des Bandarchives bei Variation verschiedener Komponenten untersucht. Als erstes Fallbeispiel wird die Verbesserung der Laufwerke betrachtet, darauf folgt der Einbau mehrerer Laufwerke. Die Erhöhung der Bandbreite des Klienten bildet das dritte Szenario.

3.3.1. Verbesserung der Laufwerke

Im ersten Fallbeispiel wird das Verhalten des Systems bei Aufwertung der Laufwerke untersucht. Hierfür wird ein vereinfachtes System mit lediglich einem Laufwerk betrachtet. Im ersten Simulationsdurchlauf wird das System mit dem voreingestellten Laufwerk der Generation *LTO-4* verwendet. In der Simulation werden zehn lesende Anfragen generiert, welche jeweils zu demselben Zeitpunkt stattfinden sollen und auf unterschiedliche, nicht gecachte Dateien abzielen. Im zweiten Simulationsdurchlauf wird ein System mit einem Laufwerk der Generation *LTO-6* aufgesetzt. Das System wird mit denselben Anfragen belastet.

3.3.1.1. Erwartungen

Das angenommene Verhalten für lesende Anfragen auf nicht gecachte Informationen ist in Abbildung 14 dargestellt.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem normalen Laufwerk.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit schnellem Laufwerk.

Abb. 14: Vergleich des Zeitverbrauchs verschiedener Laufwerken bei nicht gecachten Informationen.

In Abbildung 14a ist das erwartete Verhalten mit einem normalen Laufwerk dargestellt. Das Prinzip ist analog zu dem in Abbildung 10b beschriebenen Verhalten zu deuten. Es werden parallel mehrere lesende Anfragen an das System gesendet, wobei nur eine zurzeit bearbeitet werden kann. Weitere Anfragen müssen warten. In Abbildung 14b ist das erwartete Verhalten mit einem vergleichsweise schnelleren Laufwerk dargestellt. Das Bearbeitungsverhalten dieses Systems ist dasselbe wie das des Systems in Abbildung 10a. Hier sind die nach Gleichung 1 zu berechnenden Verarbeitungszeiten jedoch kleiner, da die Anfragen schneller verarbeitet werden können. Der Grund ist, dass jede neue Generation der Laufwerke schneller ist, als die vorherige. Für jede lesende Anfrage auf nicht gecachte Informationen gilt somit Gleichung 2.

$$t_{V,a} > t_{V,b} \tag{2}$$

In Gleichung 2 sind:

t_{Va}	Verarbeitungszeit des Systems mit normalem Laufwerk [s]
$t_{V,b}$	Verarbeitungszeit des Systems mit schnellem Laufwerk [s]

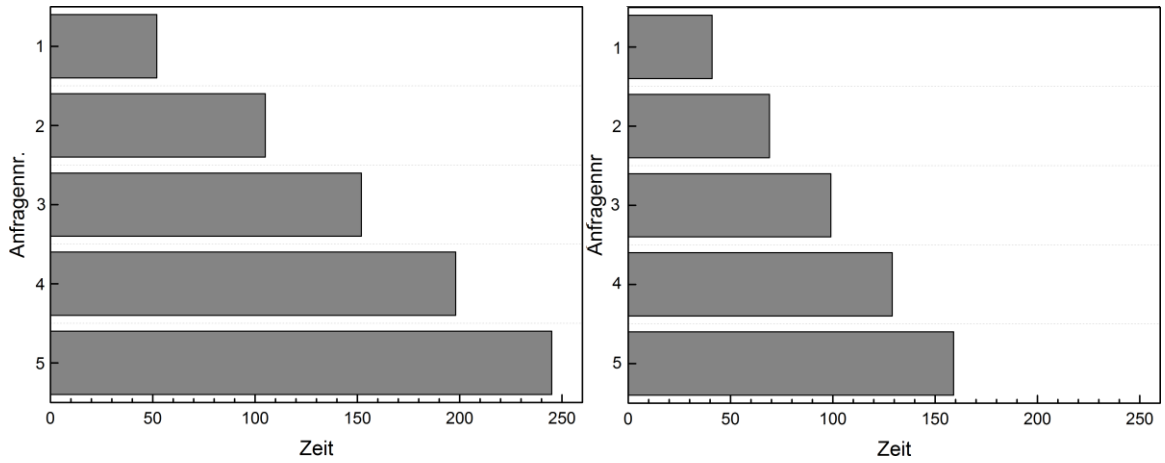
3.3.1.2. Daten und Auswertung

Es wurden zwei Simulationen durchgeführt. Beim ersten Durchlauf wurde das System mit einem LTO-4 Laufwerk verwendet. Beim zweiten Durchlauf wurde dieses mit einem LTO-6 Laufwerk ausgetauscht. In Tabelle 3 sind die Daten für die Zeit des Eingangs der Anfrage, die Zeit an welcher die Anfrage abgeschlossen wurde und die nach Gleichung 1 berechnete Verarbeitungszeit aufgeführt.

Tab. 3: Daten für den Vergleich von LTO-4 und LTO-6 Laufwerk: Zeit bei Eingang der Anfrage, Zeit bei Fertigstellung der Anfrage und die berechnete Verarbeitungszeit in Sekunden.

	$\mathbf{Zeit_{Eingang}}$	$\mathbf{Zeit_{Beendet}}$	**Verarbeitungszeit**
LTO-4 Laufwerk	00:00:03	00:01:01	52 s
	00:00:04	00:01:49	105 s
	00:00:05	00:02:37	152 s
	00:00:06	00:03:24	198 s
	00:00:07	00:04:12	245 s
LTO-6 Laufwerk	00:00:03	00:00:44	41 s
	00:00:04	00:01:13	69 s
	00:00:05	00:01:43	99 s
	00:00:06	00:02:13	129 s
	00:00:07	00:02:43	159 s

Aus Tabelle 3 ergibt sich Abbildung 15, in welcher die Anfragen gegen ihre Verarbeitungszeit aufgetragen wurden.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk der Generation LTO-4.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk der Generation LTO-6.

Abb. 15: Verarbeitungszeit für verschiedene Laufwerkgenerationen.

In Abbildung 15a sind die Verarbeitungszeiten des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk der Generation LTO-4 dargestellt. Die Verarbeitung der Anfragen verhält sich hier genau wie bei Abbildung 12a, da hier ebenfalls ein LTO-4 Laufwerk verwendet wurde. Die erste Anfrage wird bearbeitet

und da keine weiteren Laufwerke frei sind muss die Verarbeitung der nächsten Anfrage warten. In Abbildung 15b sind die Verarbeitungszeiten mit einem Laufwerk der Generation LTO-6 dargestellt. Es ist zu sehen, dass das grundsätzliche Verhalten dem von Abbildung 15a gleicht. Allerdings sind die Werte der effektiven Verarbeitungszeiten um bis zu 19 Sekunden geringer. Da weitere Anfragen stetig warten müssen, macht dieser Unterschied besonders in der Summe der gesamten Verarbeitungszeiten einen großen Unterschied, wodurch die fünfte Anfrage bei einem Laufwerk der Generation LTO-6 86 Sekunden eher abgeschlossen wurde, als die fünfte Anfrage mit einem Laufwerk der Generation LTO-4.

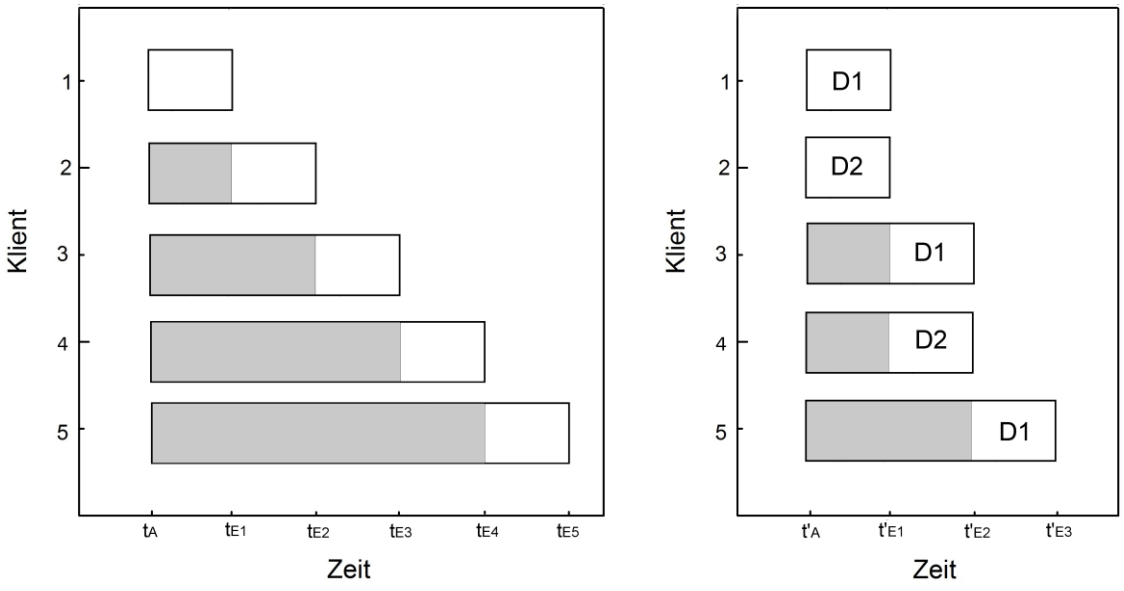
In diesem Szenario wurde der Unterschied in der Leistung verschiedener Laufwerkgenerationen deutlich. Neuere Generationen der Laufwerke arbeiten viel schneller, als ältere Generationen. Auf diese Weise können lesende Anfragen auf Dateien, welche sich nicht im Cache befinden, schneller verarbeitet werden und der Klient erhält die gewünschte Datei eher.

3.3.2. Einbau mehrerer Laufwerke

Im zweiten Fallbeispiel wird das Verhalten des Systems bei Einbau mehrerer Laufwerke untersucht. Hierfür wird der Versuchsaufbau um ein Laufwerk erweitert, so dass insgesamt mit zwei Laufwerken simuliert wird. Es wird dieselbe Nutzlast wie in Versuch 3.3.1 betrachtet. Im ersten Simulationsdurchlauf wird das System mit einem Laufwerk betrieben. In der Simulation werden fünf lesende Anfragen generiert, die jeweils zu demselben Zeitpunkt stattfinden sollen und auf unterschiedliche, nicht gecachte Dateien abzielen. Im zweiten Simulationsdurchlauf wird das Bandarchiv mit zwei Laufwerken simuliert. Das System wird mit denselben Anfragen belastet.

3.3.2.1. Erwartungen

Es wird ein Anstieg der parallel bearbeitbaren Anfragen auf nicht gecachte Informationen proportional zur Anzahl der verbauten Laufwerke erwartet. Je mehr Laufwerke im System verbaut sind, desto mehr Anfragen sollen parallel bearbeitet werden können. In Abbildung 16 ist das daraus abzuleitende Verhalten für ein System mit zwei Laufwerken im Vergleich zu einem System mit einem Laufwerk analog zu Abbildung 10b dargestellt.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit zwei Laufwerken.

Abb. 16: Vergleich der Verarbeitungszeit von einem und zwei Laufwerken in einem Bandarchiv mit Anfragen zu nicht gecachten Informationen.

Abbildung 16a ist zu entnehmen, dass bei dem System mit einem Laufwerk nur eine Anfrage zurzeit beantwortet werden kann, alle weiteren Anfragen müssen warten. In Abbildung 16b ist dargestellt, dass bei Systemen mit zwei Laufwerken zwei Anfragen zurzeit beantwortet werden können. Die Verarbeitungszeit t_V ist bei dem System mit zwei Laufwerken dementsprechend deutlich geringer als bei dem System mit einem Laufwerk. Dieses Prinzip lässt sich auf Systeme mit *n* Laufwerken übertragen. In einem System mit *n* Laufwerken können *n* Anfragen auf nicht gecachte Informationen parallel beantwortet werden, alle weiteren Anfragen müssen warten.

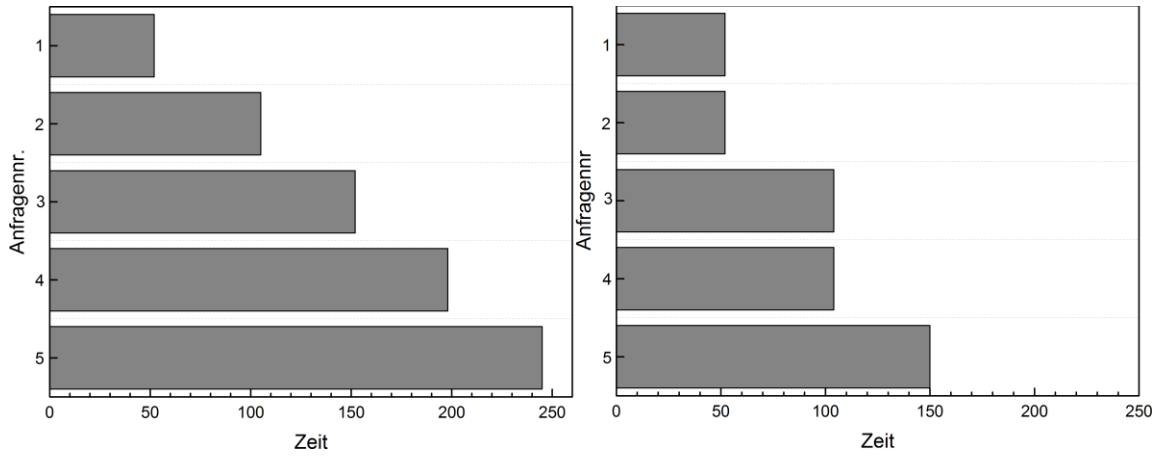
3.3.2.2. Daten und Auswertung

Es wurden zwei Simulationen durchgeführt. Beim ersten Durchlauf wurde das System mit einem Laufwerk simuliert, beim zweiten Durchlauf wurden zwei Laufwerke verwendet. In Tabelle 4 sind die Daten für die Zeit des Eingangs der Anfrage, die Zeit an welcher die Anfrage abgeschlossen wurde und die nach Gleichung 1 berechnete Verarbeitungszeit aufgeführt.

Tab. 4: Daten für den Vergleich von einem und zwei Laufwerken: Zeit bei Eingang der Anfrage, Zeit bei Fertigstellung der Anfrage und die berechnete Verarbeitungszeit in Sekunden.

	$Zeit_{Eingang}$	**$Zeit_{Beendet}$**	**Verarbeitungszeit**
Ein Laufwerk	00:00:03	00:01:01	52 s
	00:00:04	00:01:49	105 s
	00:00:05	00:02:37	152 s
	00:00:06	00:03:24	198 s
	00:00:07	00:04:12	245 s
Zwei Laufwerke	00:00:03	00:01:01	52 s
	00:00:04	00:01:02	52 s
	00:00:05	00:01:49	104 s
	00:00:06	00:01:50	104 s
	00:00:07	00:02:37	150 s

Aus Tabelle 4 ergibt sich Abbildung 17, in welcher die Anfragen gegen ihre Verarbeitungszeit aufgetragen wurden.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit zwei Laufwerken.

Abb. 17: Verarbeitungszeit bei einem und zwei Laufwerken.

In Abbildung 17a sind die Verarbeitungszeiten des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit einem Laufwerk dargestellt. Die Verarbeitung der Anfragen verhält sich hier genau wie bei Abbildung 12a. Die erste Anfrage wird bearbeitet und da keine weiteren Laufwerke frei sind muss die Verarbeitung der nächsten Anfrage warten. In Abbildung 17b sind die Verarbeitungszeiten einer Simulation mit zwei Laufwerken dargestellt. Das in Abbildung 16b erwartete Verhalten wurde bestätigt. Es ist zu erkennen, dass zwei Anfragen parallel bearbeitet werden. Die Verarbeitungszeiten der ersten beiden Anfragen betragen 52 Sekunden. Während diese beiden Anfragen verarbeitet werden, müssen alle weiteren warten. Die dritte und vierte Anfrage besitzen eine Verarbeitungszeit von 104 Sekunden, sie werden ebenfalls parallel verarbeitet. Da diese beiden Anfragen zunächst warten mussten, muss auch hier die Wartezeit von 52 Sekunden berücksichtigt werden, so dass hier ebenfalls eine effektive Verarbeitungszeit von 52 Sekunden erhalten wurde. Die letzte Anfrage muss warten, bis die Verarbeitung der dritten und vierten Anfrage abgeschlossen wurde. Die effektive Verarbeitungszeit beträgt bei der fünften Anfrage 46 Sekunden. Diese wurde etwas schneller verarbeitet, als die ersten Anfragen. Der Grund könnte sein, dass das benötigte Band von den Robotern schneller zu einem Laufwerk transportiert werden konnte.
In diesem Szenario wurde gezeigt, dass die Anzahl der im Bandarchiv verbauten Laufwerke einen Einfluss auf die Anzahl der parallel zu verarbeiteten Anfragen besitzt und durch eine höhere Anzahl an Laufwerken mehrere Dateien gleichzeitig über die Laufwerke in den Cache gespeichert und an den Klienten gesendet werden können.

3.3.3. Verbesserung der Bandbreite des Klienten

Im dritten Fallbeispiel wird das Verhalten des Systems bei Verbesserung der Bandbreite des Klienten untersucht. Hierfür wird zunächst wieder der grundlegende Versuchsaufbau verwendet. Im ersten Simulationsdurchlauf werden fünf lesende Anfragen generiert, welche jeweils zu dem selben Zeitpunkt stattfinden sollen und auf unterschiedliche, nicht gecachte Dateien abzielen. Im zweiten Simulationsdurchlauf wird dieselbe Nutzlast verwendet, aber die Bandbreite des Klienten wird stark erhöht.

3.3.3.1. Erwartungen

In Abbildung 18 ist die in diesem Szenario verwendete Netzwerktopologie dargestellt.

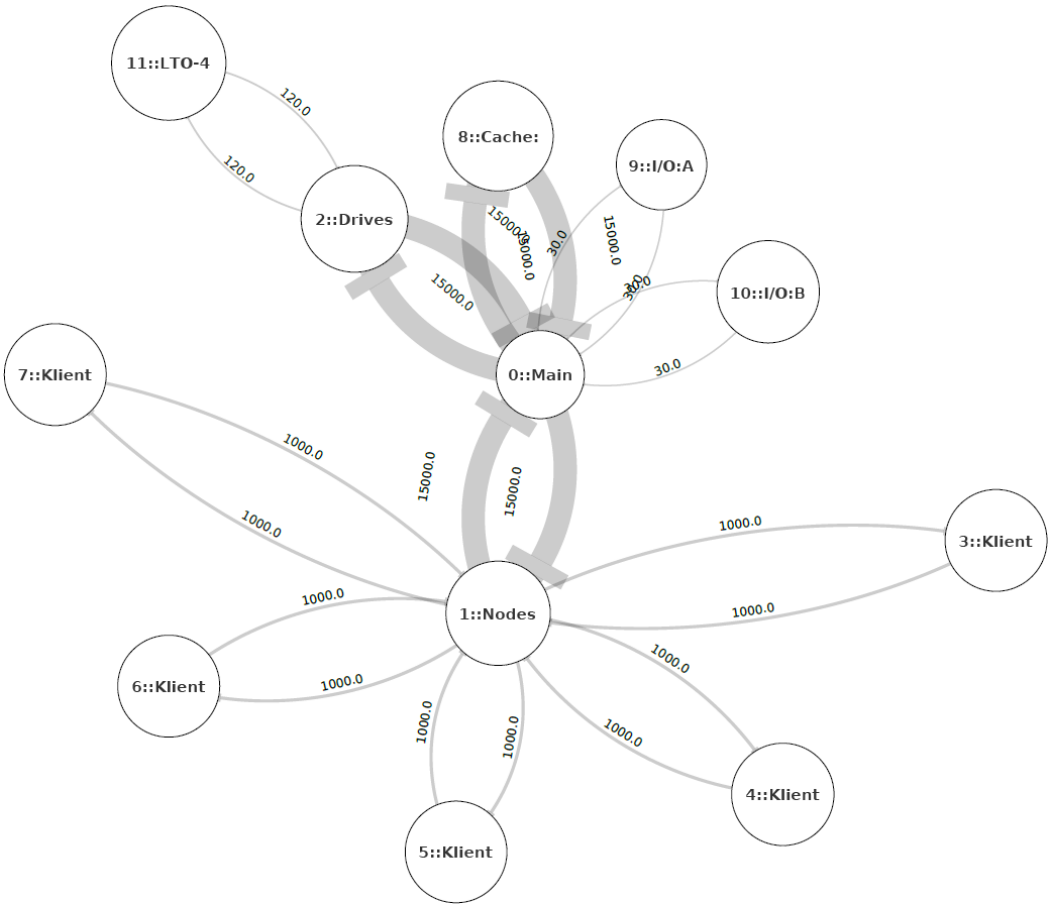


Abb. 18: Netzwerktopologie mit erhöhter Bandbreite des Klienten

Abbildung 18 entspricht dem grundlegenden Versuchsaufbau. In diesem Fall wurde die Bandbreite des Klienten um den Faktor 10 erhöht. Dies ist auch durch die, im Vergleich zu Abbildung 9, leicht breiteren Verbindungslinien gekennzeichnet. Es wird erwartet, dass das System sich in diesem Szenario analog zu Abbildung 14 verhält. Bei einer höheren Bandbreite des Klienten können die Daten zu diesem schneller transferiert werden, sodass die Verarbeitungszeit des Bandarchives sinkt.

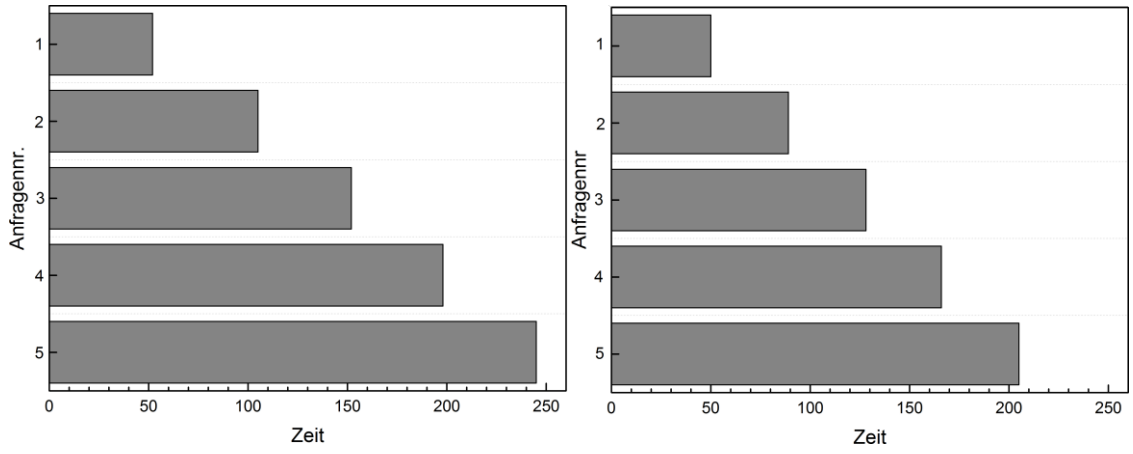
3.3.3.2. Daten und Auswertung

Es wurden zwei Simulationen durchgeführt. Beim ersten Durchlauf entspricht die Netzwerk Topologie der des grundlegenden Versuchaufbaus. Beim zweiten Durchlauf wurde die Netzwerk Topologie analog zu Abbildung 18 konfiguriert, die Bandbreite der Klienten wurde um den Faktor 10 erhöht. In Tabelle 5 sind die Daten für die Zeit des Eingangs der Anfrage, die Zeit an welcher die Anfrage abgeschlossen wurde und die nach Gleichung 1 berechnete Verarbeitungszeit aufgeführt.

Tab. 5: Daten für den Vergleich der Variation der Bandbreite des Klienten: Zeit bei Eingang der Anfrage, Zeit bei Fertigstellung der Anfrage und die berechnete Verarbeitungszeit in Sekunden.

	$\mathbf{Zeit_{Eingang}}$	$\mathbf{Zeit_{Beendet}}$	**Verarbeitungszeit**
Geringe Bandbreite	00:00:03	00:01:01	52 s
	00:00:04	00:01:49	105 s
	00:00:05	00:02:37	152 s
	00:00:06	00:03:24	198 s
	00:00:07	00:04:12	245 s
Hohe Bandbreite	00:00:03	00:00:53	50 s
	00:00:04	00:01:33	89 s
	00:00:05	00:02:13	128 s
	00:00:06	00:02:52	166 s
	00:00:07	00:03:32	205 s

Aus Tabelle 5 ergibt sich Abbildung 19, in welcher die Anfragen gegen ihre Verarbeitungszeit aufgetragen wurden.



(a) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit Klienten geringerer Bandbreite.

(b) Verarbeitungszeit des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit Klienten höherer Bandbreite

Abb. 19: Verarbeitungszeit für verschiedene Bandbreiten des Klienten.

In Abbildung 19a sind die Verarbeitungszeiten des Bandarchives von lesenden Anfragen auf nicht gecachte Dateien mit Klienten geringerer Bandbreite dargestellt. Die Verarbeitung der Anfragen verhält sich hier genau wie bei Abbildung 12a. Die erste Anfrage wird bearbeitet und da keine weiteren Laufwerke frei sind muss die Verarbeitung der nächsten Anfrage warten. In Abbildung 19b sind die Verarbeitungszeiten mit Klienten höherer Bandbreite dargestellt. Die Verarbeitung der Anfragen verhält sich analog zu dem Verhalten in Abbildung 19a. Die Verarbeitungszeiten sind jedoch geringer. Die Erwartung, dass bei einer höheren Bandbreite die Verarbeitung der Anfragen schneller funktioniert, wurde erfüllt. Die Durchschnittliche effektive Verarbeitungszeit ist um 12 Sekunden geringer, als bei einem Klienten geringerer Bandbreite.
In diesem Szenario wurde anhand der Klienten gezeigt, dass die Bandbreite der Komponenten des Bandarchives einen starken Einfluss auf die Verarbeitungszeiten der Anfragen besitzt. Durch die höheren Bandbreiten von Komponenten, in diesem Fall der Klienten, werden die Anfragen schneller bearbeitet, da die Daten schneller übermittelt werden können.

Zusammenfassung

Das Programm FeO wurde mit Hilfe ausgewählter Szenarien auf seine Korrektheit überprüft. Die verwendete Nutzlast beinhaltet lediglich lesende Anfragen auf Dateien mit einer Größe von 50 GB. Der Unterschied in der Verarbeitungszeit zwischen gecachten und nicht gecachten Informationen wird deutlich. Bei einem Bandarchiv mit einem Laufwerk müssen Anfragen auf nicht gecachte Informationen warten, wenn das Laufwerk aktuell in Benutzung ist. Sind die Informationen bereits im Cache gespeichert, so können diese direkt an den Klienten gesendet werden. Das Speichermanagement eines vollen Cache erfüllt die Erwartungen zum Großteil. Wenn der Cache trotz ausgelasteter Speicherkapazität weiter Daten aufnehmen soll, wird die am längsten im Cache gespeicherte Datei überschrieben. Lediglich das Speichermanagement bei einem nicht ausgelasteten Cache entspricht nicht den Erwartungen. Hier werden bereits gespeicherte Dateien bei Neuzugängen in der Liste nach hinten verschoben. In einem realen System würde das bedeuteten, dass der Cache bereits Gespeichertes an einen anderen, freien Speicherplatz verschieben würde.

Außerdem wurden in dem System verschiedene Komponenten ausgetauscht. Variiert wurden die Laufwerkgeneration, die Anzahl der Laufwerke und die Bandbreite des Klienten. Je neuer die Generation der verbauten Laufwerke ist, desto schneller können diese die Anfragen bearbeiten. Die parallel bearbeitbaren Anfragen auf nicht gecachte Information steigen proportional zur Anzahl der verbauten Laufwerke und Anfragen werden schneller verarbeitet, je höher die Bandbreite des Klienten ist. Das Szenario der Bandbreite des Klienten lässt sich auf die weiteren Komponenten eines Bandarchives übertragen.

In diesem Kapitel wurden, mit Ausnahme des Speicherverhaltens eines Cache mit freiem Speicherplatz, alle Erwartungen an jedes Szenario erfüllt und bestätigt. Das Simulationsprogramm FeO funktioniert wie erwartet. Die Simulation der Bandarchive ist durch eine hohe Vielfalt an Konfigurationsmöglichkeiten individuell anpassbar.

4. Schlussbetrachtung

Dieses Kapitel gibt einen abschließenden Überblick über die Arbeit. In Abschnitt 4.1 wird eine Zusammenfassung der gesamten Arbeit einschließlich der Ergebnisse vorgenommen. Zum Schluss wird in Abschnitt 4.2 ein Ausblick gegeben.

4.1. Zusammenfassung

Es existiert eine hohe Vielfalt an Speichermedien, die sich auf Grund ihrer unterschiedlichen Spezifikationen in Hierarchien einordnen lassen. In Bereichen in denen viel Speicher benötigt wird, werden häufig Bandarchive verwendet. Diese vereinen verschiedene Speichermedien und bilden damit ein hierarchisches Speichersystem.
Da Festplatten zunehmend preiswerter werden, wurden virtuelle Bandarchive entwickelt. Diese emulieren physische Bandarchive, laufen aber mittels einer Software auf einer Festplatte. Der Nutzer muss somit nicht sein ganzes System umstellen, sollte er weitere auf Bandarchiven basierende Technologien verwenden.
Um eine große Menge Daten bei einer hohen Datenübertragungsrate parallel managen zu können, wurde von IBM die Software HPSS entwickelt. Sie vereint die Rechenleistung mehrerer Rechner in einer Cluster-Struktur, wobei dem Nutzer nur ein einzelnes Speichersystem angezeigt wird.
Da magnetische Bänder nach wie vor die preiswertesten Speichermedien sind, wenn es sich um langfriste Datensicherung handelt, werden hier stets neue Technologien entwickelt. Aus diesem Grund werden Simulationsprogramme für hierarchische Speichersysteme entwickelt. In dieser Arbeit wurde das Simulationsprogramm FeO verwendet, welches speziell für Bandarchive entworfen wurde. Mit FeO kann man die benötigten Netzwerk- und Archiv Topologien über eine XML-Datei laden und direkt in der Simulation verwenden. Daten können beispielsweise in einer CSV-Datei gespeichert werden.
In dieser Arbeit wurden verschiedene Szenarien simuliert. Im ersten Versuch wurde der Cache auf sein Verhalten und seine Wichtigkeit untersucht. Im ersten Szenario des Versuches wurden lesende Anfragen auf gecachte und nicht gecachte Informationen analysiert. Die Anfragen auf bereits gecachte Informationen wurden erwartungsgemäß

sehr viel schneller verarbeitet, als Anfragen auf noch nicht gecachte Informationen. Die Anfragen auf noch nicht gecachte Informationen benötigten erwartungsgemäß mehr Zeit und mussten bei voller Laufwerkbelegung warten, bis ein Laufwerk frei gegeben wird. Im zweiten Szenario wurde das Datenaufnahmeverhalten von einem bereits vollen Cache überprüft. Der Cache überschreibt sukzessiv die am längsten gespeicherte Datei mit der neu aufzunehmenden Datei. Lediglich wenn die Kapazität des Cache noch nicht ausgereizt wurde, verschiebt er die bereits gespeicherten Dateien ans Ende der Liste, was in einem realen System zu Problemen führen kann.

Im zweiten Versuch wurde die Veränderung der Verarbeitungszeit von Anfragen an das Bandarchiv bei Variation verschiedener Komponenten untersucht. Im ersten Szenario wurden Simulationen mit Laufwerken verschiedener Generationen durchgeführt. Die Simulation wurde zunächst mit einem Laufwerk der Generation LTO-4 durchgeführt. Im Anschluss wurde das Laufwerk mit einem der Generation LTO-6 substituiert. Das LTO-6 Laufwerk benötigte bei der Verarbeitung der Anfragen deutlich weniger Zeit, als das Laufwerk der Generation LTO-4 und erfüllte damit die Erwartung, dass neuere Generationen eine schnellere Funktionsweise besitzen. Im zweiten Szenario wurde die Anzahl an Laufwerken in der Simulation erhöht. Die erste Simulation wurde mit einem Laufwerk durchgeführt, während in der zweiten Simulation zwei Laufwerke verwendet wurden. Bei der Simulation mit zwei Laufwerken wurde deutlich, dass zwei Anfragen auf nicht gecachte Dateien parallel beantwortet wurden. Dieses Verhalten lässt sich auf *n* Laufwerke übertragen. Die Erwartung an dieses Szenario, dass die parallel bearbeitbaren Anfragen mit zunehmender Anzahl der verbauten Laufwerke proportional steigen, wurde erfüllt. Im dritten Szenario wurde die Veränderung der Verarbeitungszeit von Anfragen im Zusammenhang mit der Bandbreiten der Komponenten des Bandarchives untersucht. Hierfür wurde die Bandbreite des Klienten variiert. In der ersten Simulation wurde eine niedrige Bandbreite des Klienten verwendet, im zweiten Simulationsdurchlauf wurde die Bandbreite um den Faktor 10 erhöht. Die Verarbeitung der Anfragen wurde bei der Simulation mit höherer Bandbreite signifikant schneller ausgeführt. Daraus erschließt sich, dass die Anfragen schneller bearbeitet werden können, wenn die Bandbreiten der Komponenten des Bandarchives erhöht werden, da die Daten im System schneller übermittelt werden können.

Zusammenfassend lässt sich sagen, dass das Simulationsprogramm FeO mit wenigen Ausnahmen alle im Rahmen dieser Arbeit gestellten Anforderungen erfüllt und die Simulation von Bandarchiven mit diesem Programm sehr gut funktioniert.

4.2. Ausblick

Um die Forschung an hierarchischen Speichersystemen weiter voran zu treiben, sollte die Entwicklung der Simulationsprogramme fortgeführt werden. FeO bietet sehr viele Möglichkeiten, um Bandarchive zu simulieren. Um dieses weiter zu verbessern oder einen Nutzen daraus zu ziehen, sollten weitere Experimente durchgeführt werden. Vorstellbar wären Experimente, die in den Bereich der Laufwerkplatzierung in dem Archiv gehen. Da die Platzierung der Laufwerke innerhalb des Archives eine große Rolle spielen, wenn es um Effizienz und Erreichbarkeit geht, kann mit einer derartigen Simulation potentiell ein reales System verbessert werden.
Zusätzlich könnte eine Datenbank implementiert werden, welche die Preise der verschiedenen Netzwerkkomponenten enthält. So könnten Unternehmen bei Verwendung des Programmes für ein gesetztes Budget ein System zusammenstellen und überprüfen, ob dieses den jeweiligen Anforderungen genügt.
In einem Bandarchiv gehören die Laufwerke zu den kostenintensivsten Komponenten. Ältere Generationen sind günstiger als aktuellere. Wenn man aktuelle und ältere Generationen zusammen betreibt, kann die gleiche Leistung wie bei vielen Laufwerken älterer Generationen erreicht werden und dabei sogar weniger Geld kosten. Aufbauend auf einer Datenbank mit den Preisen können in diesem Bereich gezielt Experimente ausgeführt werden, um für die jeweiligen Umstände die optimale Konfiguration zu erlangen.
Auch eine graphische Oberfläche für das Programm wäre ein Projekt, welches die Bedienung auch für Laien zugänglich und sehr viel intuitiver machen würde. Mit dieser Hilfe könnten beispielsweise die Netzwerk- und Archiv Topologien einfacher konfiguriert werden.

Literaturverzeichnis

Dee, R. H. (2008). Magnetic Tape for Data Storage: An Enduring Technology. In *Proceedings of the IEEE. Vol. 96, No. 11* (S. 1775 - 1785).

DKRZ. (2016). *Data Archive*. Online unter https://www.dkrz.de/Klimarechner-en/datenarchiv abgerufen. Letzter Zugriff: 30.09.2016.

EMC Corporation. (2016). *Virtual Tape Libary (VTL)*. Online unter http://www.emc.com/corporate/glossary/virtual-tape-library.htm abgerufen. Letzter Zugriff: 30.09.2016.

IBM. (2011). *High Performance Storage System - HPSS.*

Inman, J., Grider, G., Chen, H.B. (2014). Cost of Tape versus Disk for Archival Storage. *2014 IEEE 7th International Conference on Cloud Computing.* (S. 208-215).

Jones, Bartlett. (2014). *The Memory Hierarchy.* Online unter http://slideplayer.com/slide/7084673/ abgerufen. Letzter Zugriff: 30.09.2016.

Lüttgau, J. (2016). *Modeling and Simulation of Tape Libraries for Hierarchical Storage Management Systems.* Universität Hamburg.

Lutz, M. (2010). *Programming Python - Powerful Object-Oriented Programming.* (4. Ausg.). Sebastopol: O'Reilly.

Nelson, S. (2011). *Pro Data Backup and Recovery.* Apress.

Oracle. (2010). *StorageTek SL8500 Modular Library System - User's Guide.*

Oracle. (2015). *StorageTek SL8500 Modular Library System - Data Sheet.*

Peixoto, T. P. (September 2015). *The graph-tool python library.* Online unter https://figshare.com/articles/graph_tool/1164194/13 abgerufen. Letzter Zugriff: 30.09.2016.

Platz, T. T., Østerdal, L. P. (2012). *The curse of he first-in-first-out queue discipline.* University of Southern Denmark.

QUADStor. (2016). *Virtual Tape Library [VTL]*. Online unter: http://www.quadstor.com/virtual-tape-library.html abgerufen. Letzter Zugriff: 30.09.2016.

Quantum Corporation. (2016). *LTO-Technologie*. Online unter http://www.quantum.com/de/technologies/lto/index.aspx abgerufen. Letzter Zugriff: 30.09.2016.

SciPy. (2016). *SciPy Refrence Guide. Release 0.18.0.* Online unter http://docs.scipy.org/doc/scipy-0.18.0/scipy-ref-0.18.0.pdf abgerufen. Letzter Zugriff: 30.09.2016.

Sony. (2016). *AIT Tape Drive*. Online unter http://www.sony.net/Products/storagesolution/product/product02.html abgerufen. Letzter Zugriff: 30.09.2016.

Spectralogic. (2015). *TFinity - Flipbook.* Online unter https://www.spectralogic.com/flipbook-tfinity abgerufen. Letzter Zugriff: 30.09.2016.

Watson, R. W. (2005). *High Preformance Storage System Scalability: Architecture, Implementation and Experience. Proceedings of the 22nd IEEE.*

Wehrle, K., Günes, M., Gross, J. (2010). *Modeling and tools for network simulation.* Berlin Heidelberg: Springer Science & Business Media.

Abbildungsverzeichnis

Abb. 1: Die Hierarchie der Speichermedien als Pyramide 3
Abb. 2: Schematische Darstellung des StorageTek SL8500 Bandarchives..... 5
Abb. 3: Funktioneller Vergleich zwischen physischem und virtuellem Bandarchiv. 7
Abb. 4: Schematischer Aufbau des High Performance Storage Systems. 8
Abb. 5: Prinzip der diskreten ereignisorientierten Simulation 9
Abb. 6: Ablaufalgorithmus der diskreten ereignisorientierten Simulation..... 11
Abb. 7: UML Klassendiagramm des Programmes FeO 14
Abb. 8: Beispiel einer Netzwerk Topologie 17
Abb. 9: Die für die Szenarien verwendete grundlegende Netzwerktopologie. 20
Abb. 10: Vergleich vom Ausgabeverhalten an Klienten von gecachten und nicht gecachten Informationen bei einem Bandarchiv mit einem Laufwerk. 22
Abb. 11: Histogramm für die Bearbeitung von Anfragen eines Bandarchives. 23
Abb. 12: Verarbeitungszeit für gecachte und nicht gecachte Dateien. 24
Abb. 13: Verhalten eines vollen Cache bei weiteren Anfragen. 26
Abb. 14: Vergleich des Zeitverbrauchs verschiedener Laufwerken bei nicht gecachten Informationen. 29
Abb. 15: Verarbeitungszeit für verschiedene Laufwerkgenerationen. 31
Abb. 16: Vergleich der Verarbeitungszeit von einem und zwei Laufwerken in einem Bandarchiv mit Anfragen zu nicht gecachten Informationen. 33
Abb. 17: Verarbeitungszeit bei einem und zwei Laufwerken. 35
Abb. 18: Netzwerktopologie mit erhöhter Bandbreite des Klienten 36
Abb. 19: Verarbeitungszeit für verschiedene Bandbreiten des Klienten. 37

Tabellenverzeichnis

Tab. 1: Daten für den Vergleich von gecachten und nicht gecachten Dateien 24
Tab. 2: Daten für das Datenmanagement eines vollen Cache 28
Tab. 3: Daten für den Vergleich von normalem und schnellem Laufwerk 31
Tab. 4: Daten für den Vergleich von einem und zwei Laufwerken..... 34
Tab. 5: Daten für den Vergleich der Variation der Bandbreite des Klienten. 37

Versicherung an Eides statt

Hiermit versichere ich an Eides statt, dass ich die Arbeit eigenständig verfasst habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium.

Datum, Unterschrift