Master Thesis

Modeling and Simulation of Tape Libraries for Hierarchical Storage Management Systems

Jakob Lüttgau
Matr-No: 6146922

supervised by
Dr. Julian Kunkel

Workgroup Scientific Computing
Department of Informatics

April 9, 2016

Abstract

The wide variety of storage technologies (SRAM, NVRAM, NAND, Disk, Tape, etc.) results in deep storage hierarchies to be the only feasible choice to meet performance and cost requirements when dealing with vast amounts of data. In particular long term storage systems employed by scientific users are mainly reliant on tape storage, as they are still the most cost-efficient option even 40 years after their invention in the mid-seventies. Current archival systems are often loosely integrated into the remaining HPC storage infrastructure. However, data analysis tasks require the integration into the scratch storage systems. With the rise of exascale systems and in situ analysis also burst buffers are likely to require integration with the archive. Unfortunately, exploring new strategies and developing open software for tape archive systems is a hurdle due to the lack of affordable storage silos, the resulting lack of availability outside of large organizations and due to increased wariness requirements when dealing with ultra durable data. Eliminating some of these problems by providing virtual storage silos should enable community-driven innovation, and enable site operators to add features where they see fit while being able to verify strategies before deploying on test or production systems. The thesis asseses moderns tape systems and also puts their development over time into perspective. Subsequently, different models for the individual components in tape systems are developed. The models are then implemented in a prototype simulation using discrete event simulation. It is shown that the simulation can be used to approximate the behavior of tape systems deployed in the real world and to conduct experiments without requiring a physical tape system.

Acknowledgements

I would like to thank my advisor Dr. Julian Kunkel and Wolfgang Stahl for their time and the expierience they shared with me discussing the long-term storage architecture of DKRZ.

Contents

Chapter 1

Introduction

This chapter gives an overview over the broader supercomputing landscape followed by a more in-depth look into storage technologies and the demands by its users. Section 1.1 also covers prospected usage scenarios in the near future. In particular, it is focused on long-term storage technologies where tape storage systems are todays dominant technology. Afterwards, a short discourse in Section 1.2 rejects claims suggesting magnetic tape may be about to become obsolete in the near future. Finally, the need for comprehensive models and tools to simulate hierarchical storage systems and tape libraries is motivated in Section 1.3. Section 1.4 presents the goals of the thesis.

1.1 Motivation

High-performance computing (HPC) as we know it today mostly consists of cluster computers (Top500, 2016). For the last couple of decades CPU performance increased roughly every 18 months, this phenomena is commonly referred to as Moore's Law. Though while Moore's Law is slowing considerably, dependent on how we look at it, other areas such as network and storage technologies simply did not enjoy comparable technological progress.

In fact, especially scientific workloads, which tend to be very data intensive, have hit what is called the memory wall. While CPU speeds and disk capacity both grew their fair share at about factor of 500 (Kunkel et al., 2014), CPUs outperform memory technologies in terms of speed, thus making memory the main bottleneck. Applications that are dominated by reading and writing a lot of data from disks or transmitting a lot of data through a network are considered I/O bound. Many scientific applications are I/O bound. To still provide long-term storage technologies with adequate performance modern systems combine the best properties of different storage technologies to meet *quality of service* (QoS) requirements. Furthermore, new storage hierarchies are constantly added such as Intel's XPoint, a class of non-volatile random access memory (NVRAM), resulting in ever more complexity. Storage demands for supercomputers grow geometrically (Inman et al., 2014) with a growth rate of roughly three times the clusters main memory per month. With capacities between 0.2 to 1.5 PB for the leading 20 supercomputers, it won't be long before the first exascale storage systems are needed.

Most scientific projects operate on tight budgets and have to find a cost efficient system. Multiple factors have to be considered, and while the hardware for a supercomputer alone costs easily tens of million of Euros, operators usually look at the *total cost of ownership* (TCO). TCO includes not only acquiring the hardware but also accounts for energy, wear parts and sometimes staff required to run the system. Notably, energy has become a relevant factor, as big data

centers approach limits imposed by the energy infrastructure.

The need to find a compromise of balancing cost and performance usually results in considerably more complicated systems. Many different technologies are mixed, which sometimes leads to unforeseen side effects. And complex systems require more effort to optimize for better system exploitation. In fact, often it is hard to decide what can be considered a good level of exploitation. In general, we acknowledge the lack of experts in the field. While the systems are expensive on their own, what is actually valuable is the insight we extract from data generated on these systems. While sometimes such insight may be very compact (a parameter, a formula) in many cases it is not and we want to preserve the data for later use because calculations consumed time and money. Incentives for long-term storage are manifold, but often relate to to one of the following:

- Preservation of human knowledge
- Preservation of cultural goods (arts, literature, music, movies, etc.)
- Archival of organizational data (e.g., raw movie footage)
- Preservation of personal documents and photos
- Compliances with legal requirements

In science, preservation is especially important for scenario's involving a lot of uncertainty. A good example is the case of climate research where revisiting the data in five, ten, twenty or fifty years is necessary, to evaluate how accurate predictions have been. Data describing the state of the earth, thus earth system data, intuitively will require to store a lot information. Predictions in earth sciences become more accurate as we increase the resolution, but doubling the resolution increases the amount of data by a factor of four (2D) to eight (3D). Only recently resolutions for regional models of about 100m became feasible (DKRZ, 2015c). In addition for a prediction of future climate, it is necessary to average over many different scenarios, e.g., accounting for natural disasters such as the eruption of a volcano or more importantly providing prognosis on the enactment or failure to enact certain policies. The German Climate Computing Center (DKRZ) estimates to add 75PB per year over the course of the next five years to the long-term storage archive (DKRZ, 2015a).

Similarly, microscopic research enjoys no practical limit for storage requirements, especially as a sheer infinite number of combinations is possible for experimental setups. CERN conserves data collected during the experiments, but already a single second accounts for 10 million proton collisions registered by the detector, of which only a tenth are considered for further processing and finally only about 2000 of these observations (or 250GB/h) will be made persistent. CERN estimates to produce 20PB worth of data per year, which is archived and replicated around the world.

The broadcast and movie industries has become a large user of tape archives to store raw footage. Increasingly on demand streaming services are discovering tape as a cost-effective solution for data storage which becomes hot for a while but is only rarely requested later. Streaming content such as movies or music are requested a lot when released, but as time proceeds they are requested only every once in a while, with spurious bursts sparked by increased media coverage,

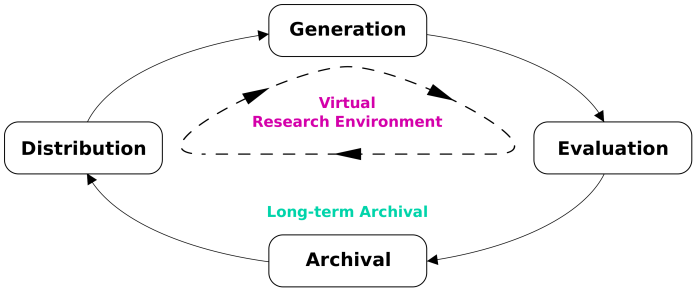


Figure 1.1: The data life cycle as described by the DKRZ (DKRZ, 2015b).

e.g., due to the death of an artist. A similar problem is observed at Facebook and other social media websites where users upload billions of photos and videos, with interest cooling down shortly after being posted. Only a few of these posts remain hot for longer because they went viral.

The *Department of Energy* (DOE) requirements maybe somewhat different as there is a lot of confidential calculations performed. The procedure is to lock a complete system for a while and clean it again after usage. Therefore, requiring the systems to move vast amounts of data before and after running a simulation to ensure classified information does not leak. Doing so required a tight integration of tape storage archives into the storage hierarchy. In a recent report (Ornl et al., 2015), the DOE further acknowledges an increasing demand of allowing in situ data analysis even for large scale applications, sometimes consulting data stored in the archive, thus requiring a tighter integration with the scratch storage system. The DKRZ describes a data life cycle commonly observed when working with earth system data (compare with Figure 1.1): A cycle starts with the *generation* of data, e.g., using a simulation. The results are then *evaluated* and valuable results are *archived* and *distributed* for further research. Iteratively this cycle allows to improve the underlying simulation models. Ideally any technical concerns of long-term storage become transparent to the users resulting in a more "distraction-free" *virtual research environment*.

Storage technologies experienced an eventful past (Amouroux, 2014). For a while, CDs attracted a lot interest as a media for long-term archival, until problems due to micro oxidation started to surface. Hard disks featured large capacities but the mechanical design make vulnerable to a range of threats, for instance dust or physical impact in the worst case, leading to head crashes which might render the hard disk useless or at least making recovery extremely difficult.

Besides new technologies (NAND, XPoint, etc.), cloud architectures are becoming very popular lately. But for large-scale systems cost comparisons find outrages prices, e.g., using a service such as Amazon Glacier. However, institutions and companies need to carefully deliberate if moving trade secrets into the cloud is really a good idea, as it opens new opportunities for industrial espionage. Distributed approaches to long-term storage give rise to problems that have been more or less negligible in centralized approaches before. One such problem is the accumulation of errors as data is replicated. Though the use of additional parity information provides protection against data corruption to a certain extent.

The limited life-time of data is a problem generally acknowledged. For some applications we seek data resilience not satisfied by any of the current systems. To have a glimpse at the future, two approaches appear especially notable though their relevance from a data center perspective remains somewhat speculative. Focusing on ultra durable data storage using fused silica glass, Hitachi and Kiyataka (2014) were able to read and write digital recordings that supposedly may last for hundreds of millions of years and under extremely rough conditions withstanding water and fire. While no information on expected transfer rates ware published the recording density is announced to match those of Blu-ray discs at 1.5 GB/inch^2.

In another approach, Goldman et al. (2013) encoded digital information using *Deoxyribonucleic acid* (DNA). This comes with a number of advantages, for one DNA is fairly resilient, and on the other hand DNA features extremely high data densities (2 bits per 0.34nm) when compared to other technologies that are currently in use. For example it is estimated that it maybe possible to fit all the knowledge of humanity into two cubic meters of space, supposedly not accounting for strategies to efficiently access this information. The biggest problem right now is to synthesize DNA which is rather expensive. Reading and reproducing DNA on the other hand is relatively easy. Interestingly copying data does not require to digitally interpret it. Simply by performing *Polymerase Chain Reaction* (PCR) or by embedding information into a living organism virtually infinite amount of copies can be produced. An interesting choice for an organism to store data might be *Deinococcus radiodurans*, which is frequently found in radioactive waste and in nuclear facilities, and is known to be extremely radiation-resistant. Of courser, as of today, no practical systems using such technology are commercially available.

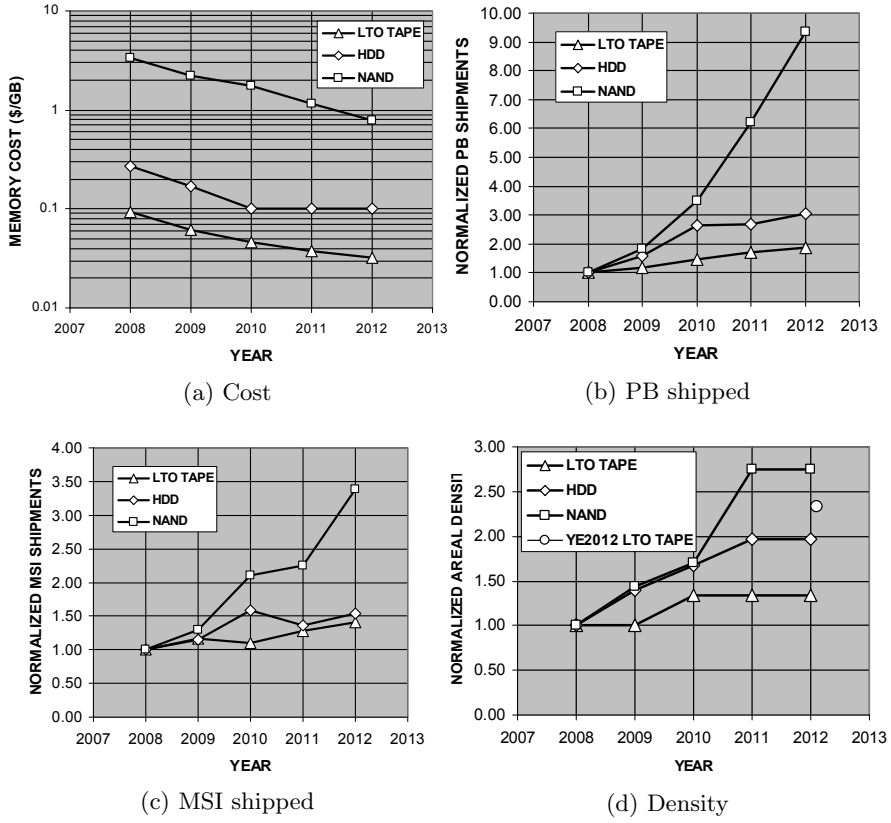
Despite all progress being made on different fronts, some are astonished to find that even today most large scale setups for long-term storage are realized by employing tape in automated robot libraries. This is also interesting as magnetic tape and variants such as floppy discs and diskettes have a long history in computer science. None the less, in the wake of new technologies and increasing demands to the storage infrastructure to be expected in the coming years it appears adequate to questions the current regime of tape archives.

1.2 The Relevance of Tape in the Future

Tape has many drawbacks for being a linear storage technology resulting in unbearable penalties when dealing with random-access data. Modern tape drives, however, are quick for sequential reads and tapes outperform other commonly used media by several orders of magnitude when it comes to costs per gigabyte. Tape is further reusable, energy efficient, and has proven to be very durable easily providing for 30 years of reliable conservation of data.

For many years, prices for hard drives and NAND decreased, but according to an analysis by researchers at IBM (Fontana et al., 2013), this trend is stagnating as higher density technologies require investments into the manufacturing process. It is notable that tape has a small user base in comparison to NAND or disk, yet it is competitive without benefiting from the economics of scale to an extend comparable to those of disk and NAND.

Fontana et al. (2013) revealed some useful insight considering the cost per



(a) Cost (b) PB shipped

(c) MSI shipped (d) Density

Figure 1.2: Comparison of Tape, Disk and NAND storage (Fontana et al., 2013).

GB, petabytes and *million of square inches* (MSI) [1] shipped or the areal density used in the technology, see Figure 1.2 for more details. The graphs also confirm the effects of some interesting events such as reduced manufacturing requirements for tape in 2010 due to higher capacity cartridges, or the impact of natural disaster 2011 in Thailand on the HDD supply. Finally, it is interesting to see that the areal density stagnated as well, comparing HDDs to next generation LTO tapes HDD might even loose the density advantage. Sony and Fujifilm separately announced in 2014 to bring cartridges with 185 TB and 154 TB respectively to market by the end of 2015. As of this writing though no such "super" tape is commercially available though the technology was demoed in lab conditions and on conferences. Nonetheless at 0.01 EUR/GB tape is quite competitive with all the major storage technologies. Also for many application fair amounts of compression can be assumed, though other technologies benefit from this as well. Assuming retail prices as of February 2016 costs for LTO-6 and LTO-7 are as follows:

- LTO-6: 0.011 USD/GB native, 0.005 USD/GB compressed, (2.5 to 6 TB)
- LTO-7: 0.028 USD/GB native, 0.012 USD/GB compressed, (6 to 15 TB)

Of course, this price per gigabyte figure is not to be mistaken with TOC, but also in this regard tape is attractive, as in comparison to spinning disks or NAND tape does not require energy making it one of the greenest solutions (Eleftheriou et al., 2010). This can also be observed in terms of carbon emissions. SATA RAID systems are the worst offenders at about 450.000 lbs of CO_2 per year. Automated tape systems are closer to 30.000 lbs of CO_2 per year. One might argue it is possible to power down other persistent media as well. After all the actual robot library system could be ignorant about which type of storage media it delivers. But any removable media comes with hefty penalties at read time, which at some point do not justify the higher cost, especially as for sequential reads modern tape drives are quite competitive at 400MB/s. One example for different media types is Massive Array of Idle discs (MAID)[2], where Facebook made headlines (O'Toole, 2016) in 2014. Blu-ray discs maybe easier to access than tape, they allow for random access, but with Blu-ray fading as a medium to distribute movies, the future of such approaches remains somewhat open.

Various types of tape systems have been commercially available since around 1950. Given the years of experience with tape, the expected lifetime for tapes is considered to be thirty years (A.M.P.A.S, 2007), provided a functioning tape drive is kept. Compare this to HDD lifespans which are about five to seven years (Backblaze, 2016). For NAND some manufacturers (Niset and Kuhn, 2005) propose data retention times of up to 100 years, more conservative estimates for viable products range anywhere from two to ten years. This limit is mostly established because flash storage technologies allow only for a limited amount of write cycles which may be a factor for storage systems in general. Archival is an appreciated exception because data usually needs to be written only once and is mostly read ever after.

Cloud storage was mentioned earlier and, indeed, this can be an interesting option for mid-term on demand requirements; but for large systems that touch

[1] A measure commonly used in the semiconductor industry.

[2] Emissions for MAID systems are at about 100.000 lbs of CO_2 (Eleftheriou et al., 2010)

and move a lot of data owning and operating the system usually will be cheaper. Current cloud storage costs are advertised (Amazon, 2016) at 0,007 USD per GB and month, 5% of average monthly storage can be accessed free of charge. Additional access is charged at 0.01 USD / GB. Further keeping data for less than 90 days also results in additional charges. To give an example, assuming the DKRZ will constantly add 6.25 PB/month for the next five years to the existing 50 PB of storage. It seems reasonable to assume read access will not imply additional cost, as even in the beginning 5% of average monthly storage allow for free transfer of 2.5 PB/month which leaves a generous margin to about 0.5 PB/m observed reads requests. Let's further allow very optimistic price drops of 0.002 cents every 12 months. The cost over 5 years still adds up to about 47 million USD. LTO-6 and the recently released LTO-7 cartridges can be obtained in retail for 30 to 150 USD respectively, not accounting for discounts for large quantities. Buying 450 PB in LTO-7 on the spot will cost about 12 million USD (6 million, LTO-6), leaving at least 35 million USD for hardware, facilities, operation, staff, and energy.

In addition, many institutions are already invested in tape storage so that transitioning to more modern storage technologies would be a gradual process. But instead of replacing tape and considering it merely a legacy, it appears much more likely that users will increasingly turn to hybrid approaches, thus taking the best properties of all worlds. As a result, operators can design their systems to efficiently meet their purpose. For example findings by LeeAhnKim suggest huge potential for cost savings for broadcasting and streaming services of up to 50% only by introducing clever usage of different storage classes by including tape. With tape to stay part of the storage hierarchy, until the cost advantage is no longer present data centers will have to make choices on how to integrate and operate tape archives. In addition, as long as no single best technology surfaces, storage will maintain different hierarchies to provide a certain quality of service at the lowest cost in terms of TOC. Much of the insight and abstractions required to model tape remains applicable even with tape replaced.

1.3 Simulating Tiered Storage Systems

Many areas in computer science have largely profited from contributions driven by the user community. Linux, once an underdog, mocked by professionals in the field and by big cooperations, is today the de-facto standard providing most of the infrastructure we enjoy everyday and call the internet. In addition, until today the majority of super computers are using Linux or a variant of Unix. Increasingly, consumer electronics are relying on open software, in fact, the two dominant operating systems for smartphones accounting for more than 90% of the market are based on Linux or BSD.

For tape libraries, the situation is radically different. Access to enterprise class systems is quite prohibitive for a number of reasons. On the one hand these systems tend to be very expensive, and only large institutions employ them. On the other, the very reason to buy such a system is to ensure data security, thus any experiments bearing the risk of inducing downtimes or data loss are generally met with healthy reluctance by operators.

With LTO, there now exists a common industry standard that enables competition between different vendors which can be reassuring for customers as it

should keep prices fair and provides for better planning security. None the less, given the scale of these systems, technical improvements of the hardware are mostly in the hand of these vendors. This is not necessarily a problem but levers for customers are few because the active vendors are mostly industry giants following a release schedule optimized for profits. With a comparable small user base providing limited financial incentives and only a handful of experts there is no foundation for rapid innovation.

A cheap way to enable innovation and train new talent has always been simulation. Before letting people work with delicate systems, e.g., pilots learn to fly in a simulator. In robotics simulators provided a cheap playground for young people, and, for some tasks, also allowed to test and evolve systems much more rapidly than would have been possible with physical systems. Similar concerns hold true for archival system where mistakes or flawed strategies may render the stored data useless or require costly restoration. For example accidentally deleting the tape libraries global file index can be performed almost instantaneous. Luckily, for newer tape systems, that are self-describing, it is possible to rebuild the registry. However, with ten thousands of tapes it will mean considerable downtime of operations. A comprehensive model then also maybe used to suggest recovery times, and elaborate on different restoration strategies.

1.4 Goals

The goals of the thesis are to allow for more informed decisions for the deployment of tape systems and hierarchical storage systems. From a methodological point of view this can be divided into multiple individual tasks:

1. Development of models to describe key aspects of tape systems
2. Simulation of tape systems using discrete event simulation
3. Collection of key metrics during simulation and generation of reports to gain new insight and to allow for more educated decisions.
4. Reporting and data analysis workflows for hierarchical storage types
5. Tooling to gain insight on the benefits of different configurations for HSM

Some questions that should be easier to answer with such models and tools to simulate should include:

- How to deploy a cost-efficient system from a data center perspective?
- What are the minimal requirements to meet a specification or QoS?
- Which features do we need for the next generation of systems?

In a long-term perspective, the hope is that the general architecture to model and simulate can, at some point, be used to operate an actual tape library. The basic idea being that, e.g., algorithms and scheduling policies – once implemented with the simulator will directly work with a production system. The goal for the implementation consequently also strives to be modular and reusable.

1.5 Thesis Outline

Chapter 1 motivated the topic of the thesis and provided an overview of current challenges to storage systems and tape archives in particular. In Chapter 2 the necessary technical background and terminology is established. It follows related work in Chapter 3 to encompass the state of the art. Armed with the information from the previous chapters, in Chapter 4 different design decisions are discussed and a model for simulation is proposed. Implementation details are covered in Chapter 5 on simulation. It follows the evaluation in Chapter 6 where the simulation is fed with real world workloads and limitations of the implementation are discussed. The thesis concludes with Chapter 7 where the results are summarized followed by an outlook on future work.

Summary

Many different storage technologies exists but limited budgets and different requirements across data centers result in hierarchical storage systems to be deployed. Especially scientific users, who make extensive use of supercomputers, rely on efficient large scale long-term storage systems. Exascale class storage systems will be required by many organisations in the near future. Tape is very affordable in comparison to other storage technologies and it appears like the cost-advantage for tape will remain for years to come. Yet innovation in the field could be quicker, and current tape systems leave a lot to be desired. The problem is that mainly vendors are improving tape systems, but on profit oriented release strategies. To make experiments and research with tape systems easier models and tools for simulation should be developed.

Chapter 2

Background

Tape libraries and, therefore, hierarchical storage architectures are complex systems, which are constantly evolving to account for updated requirements and new technologies. Section 2.1 covers the necessary backgrounds for tape technology, including a brief history of tape systems. Section 2.2 is dedicated to the most important features commonly supported by systems. Section 2.3 takes a look at automation and commercially available solutions. Section 2.4 briefly explains how LTFS works and changed the semantics of how tape is used today fundamentally. As pure tape systems are rarely seen today, hierarchical storage management systems are introduced in Section 2.5. Section 2.6 provides an overview of the German Climate Computing Centers (DKRZ) storage infrastructure.

2.1 Magnetic Tape Storage

In 1928, Fritz Pfleumer for the first time used magnetic oxides (ferric oxide to be specific) to record information. Since then magnetic based and magnetic tape based storage have come a long way from compact cassettes, to floppy discs, and video cassettes. All of these storage media share a common working principle. A thin magnetic coating is applied to a polyester base, together only a few microns thick. As polyester is fairly robust and has a high tensile strength, it does not impose practical limits to the length of such a film which makes it ideal to to be stored on a reel. Over time different enclosures where added to better protect the tape from external conditions. Figure 2.1 shows a typical cartridge as it is used in modern tape systems, mostly for its small form factor. But over the course of time many different principles have been tried; in the past dual reel cartridges were quite common. To read and write data, a special device is required, often called a tape drive.

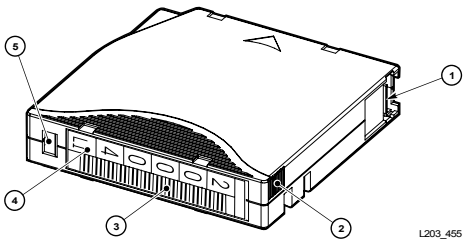


Figure 2.1: LTO tape cartridge. (1) Tape access door, (2) Finger grips, (3) VOLID Label, (4) Media ID Label, (5) Write-Protect Switch Not depicted in the illustration is the actual tape and, in particular, the lead pin that is used to pull the tape out of the cartridge. Illustration: Sun (2006)

(a) linear (b) serpentine (c) helical-scan

Figure 2.2: The most common on tape data layouts.

2.1.1 Tape Data Layouts

Multiple competing data layouts existed over time, but commercially available products mostly use one of the following layouts. Today linear technologies are favored for being less complex and because they allow multi-channel/track reading and writing.

Linear: data layouts on tape are not very common anymore, but are the simplest layout. The drawback with strictly unidirectional tracks are the need to rewind when reaching the end of the tape.

Linear-serpentine: data layouts are today the predominant way to layout data on magnetic tape. The benefit for sequential data is that there is no need to rewind the tape. Bidirectional read/write heads had to be developed first.

Helical-scan: layouts are not really relevant for storing digital data anymore, though some systems are using the technique. Helical-scan tape layouts are of advantage when dealing with high frequency data such as analog video.

2.1.2 Resilience of Tape Storage

While tapes are reliable and suitable to preserve data for long periods, there are some means that may lead to data loss, including biochemical reactions, disaster and smaller accidents that occur in day to day handling. When tape sits unused for longer periods the so called *Sticky-shed syndrome* occurs which may render tapes unusable due to moisture. With enclosed cartridges the moisture is generally less of a problem but can still occur. Luckily, in many cases, it is possible to "bake" the tape which makes it readable again for a while. In order to protect electronic components, most automated tape libraries specify operation temperatures which are in the range of 15 to 35°C. Tape is flammable and fire prevention advice is to operate in oxygen reduced environments.

Tapes are fairly forgiving to physical mishandling in comparison to hard drives or CDs. Still, especially for single rail cartridges a common problem is that parts of the cartridge holding the tapes lead-pin in place may break when accidentally dropped. Such a cartridge needs to be fixed before it can be used again in a tape drive. The main problem here is that for many formats on-site repair is not possible or feasible considering the risk of damaging the drives. Different vendors have therefore service centers with special equipment, which are available as part of a service agreement. A broken cartridge has to be sent in and the data will be restored, however service centers are often in other countries. This sometimes has implication when dealing with classified data, sending a cartridge to another country for service may not be an option.

2.1.3 History

Magnetic tape has a long history. Table 2.1 lists important events that made tape possible as it is used today.

1890s	Valdemar Poulsen invents **Magnetic Wire Recording**. Only limited use through the 1920s and 1930s, but popular from 1946 to 1954. One hour of audio recording required about 2200m of thin wire (0.10 to 0.15 mm).
1928	Fritz Pfleumer uses ferric oxide (Fe_2O_3) as a recording medium. The approach is improved by AEG and reel-to-reel tape recorder for tapes produced by BASF is released. The method was kept secret during World War II.
1947	John Bardeen, Walter Brattain and William Schockley invent the **Transistor**
1950	Reel-to-Reel recording and playback devices become affordable enabled by transistors.
1951	Data storage UNIVAC I (UNIVersal Automatic Computer I) 128 chars per inch, written on 8 tracks
1952	IBM introduces the first magnetic data storage devices often referred to as *7 Track*.
1962	Phillips invents **Compact Cassete** for audio recordings, though it was also sometimes used for data storage.
(1956)	Quadruplex for video tapes. *Focus on tape from here on, as other media such as floppies and diskettes are beyond the scope of the section.*
1959	Toshiba introduces helical scan as tape draw speed determines the maximum recordable frequency. The signal may not get imprinted which was a problem for video recording. Sony later pushes this technology forward.
1980s	Introduction of **automated robotic tape libraries** by Sun with the Brand StorageTek. Tape suddenly accessible within a minute instead of hours or days. The term *nearline storage* was created.
1990s	Linear Tape Open (LTO) Consortium is founded. LTO today is the most wide-spread tape format.

Table 2.1: A brief history of early magnetic tape storage including relevant historic breakthroughs paving the way to modern computing and storage.

2.1.4 Magnetic Tape Formats

Given the long history of tape storage technologies, it is no surprise a wide variety of formats was invented throughout time. The following collection limits itself to the most notable technologies with Linear Tape Open (LTO) being the most important standard today. Some standards were updated, denoted by the "Super" prefix, as is the case for DLT and AIT. Even though the are phased out they are still relevant for legacy reasons.

LTO: Linear Tape Open is the most important standard today and features capacities of up to 6 TB at 300 MB/s for LTO-7 generation tapes employing serpentine linear tape. Refer to Section 2.1.5 which is dedicated to a more detailed description of LTO.

Txxxxx: StorageTek tape format, with T10000D being the most recent variant, which features 8.5 TB native capacity at up to 252 MB/s. Older cartridges (until T9840D) had dual reels but the layout changed to a single reel rectangular variant as they proved useful in automatic libraries.

(S)DLT: (Super) Digital Linear Tape has been around since 1984 and features capacities of up to 800GB and transfer rates of 60 MB/s using serpentine tape layout. The last generation was released in 2007 by Quantum.

(S)AIT: (Super) Advanced Intelligent Tape has been around since 1996 and is based on DAT. The most recent variant SAIT-2 from 2006 featured transfer rates of up to 45 MB/s and capacities of up 800GB, using helical scan layout. The format was popular for being backward and forward compatible for many generations. In March 2010 Sony announced (S)AIT to be discontinued.

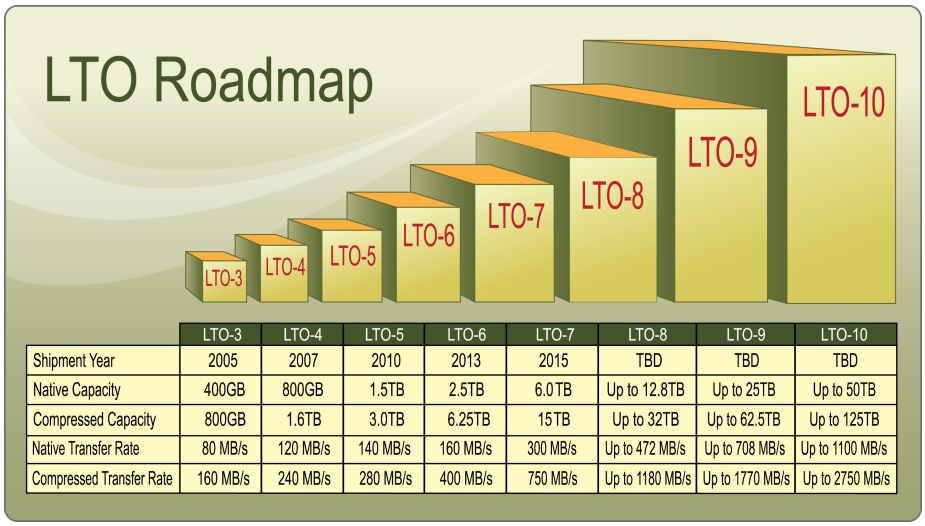
DAT/DDS: Around since 1989, last generation (DAT-320) of DAT/DDS tapes was released in 2009, but is not competitive anymore. Using helical scan and dual reel cartridges transfer rates up to 12 MB/s and capacities up 160 GB (native) were supported. The format is discontinued with the last press release documented from 2010, but the website is no longer active.

VXA: Started in 1999, VXA is a helical scan based tape format, similar to the common video cassete recorders (VCR). The last released format from 2005, VXA-3, featured speeds 12 MB/s and capacities up to 160 GB. As of November 2006 the format is not being further developed and supported.

Other: A large number of form factors is no longer relevant, including ADR (until 2003), SLR (until 2003), Travan (until 2002), Mammoth (until 1999).

2.1.5 Linear Tape Open (LTO)

Linear Tape Open (LTO) is a storage technology for magnetic tape, governed by the LTO Consortium controlled by Hewlett-Packard, IBM, and Quantum. Initially, Seagate was member of the consortium, but the tape devision was spun off and, ultimately, acquired by Quantum taking the seat. Branded as LTO



	LTO-3	LTO-4	LTO-5	LTO-6	LTO-7	LTO-8	LTO-9	LTO-10
Shipment Year	2005	2007	2010	2013	2015	TBD	TBD	TBD
Native Capacity	400GB	800GB	1.5TB	2.5TB	6.0TB	Up to 12.8TB	Up to 25TB	Up to 50TB
Compressed Capacity	800GB	1.6TB	3.0TB	6.25TB	15TB	Up to 32TB	Up to 62.5TB	Up to 125TB
Native Transfer Rate	80 MB/s	120 MB/s	140 MB/s	160 MB/s	300 MB/s	Up to 472 MB/s	Up to 708 MB/s	Up to 1100 MB/s
Compressed Transfer Rate	160 MB/s	240 MB/s	280 MB/s	400 MB/s	750 MB/s	Up to 1180 MB/s	Up to 1770 MB/s	Up to 2750 MB/s

Figure 2.3: LTO Roadmap (Spectralogic, 2016a).

Gen	Thickness (μm)	Length (m)	Tracks	Bit Density	EEPROM
1	8.9	609	384	4880	4 kb
2	8.9	609	512	7398	4 kb
3	8.0	680	704	9638	4 kb
4	6.6	820	896	13250	8 kb
5	6.4	846	1280	15142	8 kb
6	6.1	846	2176	15143	16 kb
7	5.6	960	3584	NA	16 kb

Table 2.2: Ultrium Tape Cartridge Specifications. It is noteable that capacity gains are mostly achieved increasing the number of tracks and by reducing the thickness, which in turn allows to increase the length of the tape.

Ultrium, the only realized of two planned form factors, LTO is the best-selling super tape format used by many organizations in their computer systems. LTO is a specification for tape cartridges, as depicted in Figure 2.1. As of October 2015 the newest generation Ultrium 7 (LTO-7) hardware and tapes are commercially available featuring capacities of 6 TB (15 TB compressed) and transfer rates of up to 300 MB/s. LTO follows an interesting strategy for introducing technological improvements, which is attractive for customers when planning their system. A new generation is being introduced every 2-3 years, roughly doubling capacity and bandwidth every generation. Drives and tapes are designed to ensure backwards compatibility for writes with the prior generation and read-compatibility for the prior two generations. Thus, protecting investments of customers for about a decade, see Figure 2.3 for more details. A more detailed specifications are collected for the cartridges (Table 2.2).

2.2 Features of Modern Tape/Storage Systems

2.2.1 Write once read many (WORM)

Storage devices, for which information cannot be modified after writing are often termed *Write-Once-Read-Many* (WORM). It is mentioned here, as for many applications (e.g., backup) and also for optimizations, the guarantee for data not being modified can be very beneficial. As of 2005, LTO provides WORM as an option. Prior to tape, optical media types such as the Compact Disk (CD), Digital Versatile/Video Disk (DVD), and BluRay were commonly used as a WORM media.

2.2.2 Archive and Backup

Often tape library systems are employed for backup archival or to comply with regulation. As such, the systems have to care about best practices in archival and backup and allow replication for disaster recovery when handling very valuable data. Most tape library systems offer, therefore, convenience features such as special management software, that can perform these tasks automatically.

2.2.3 Self-describing tape formats

A common limitation of older tape systems was the lack of portability for data cartridges. The file structure and metadata information were managed by the library. Simply inserting a tape would leave the user with a blob of data. Today, self describing cartridges often use LTFS (see Section 2.4). But also before LTFS *tar*, short for *tape archive*, was used to allow for a file structure on tape.

2.2.4 Data Reduction and Compression

A lot of data in its operational form is quite unoptimized and repetitive, which is why data is often compressed before it is stored. Depending on the data, savings of 50 percent or more are not unreasonable. Some systems perform this transparently (that is the user is not even realizing the data is compressed). Another approach, where copies of files or parts of files are very common, deduplication is automatically performed by the file systems. Until changes are made the same data on disk is referenced just under different filenames.

2.2.5 Encryption

Some of the users of large scale storage system require their data to be encrypted. In particular, the movie industry, government, and military organizations are pushing for this. But as data breaches and information leakage are increasing and more and more of personal and sensitive data being stored in such systems, demand for such measures is rising.

2.3 Automated Library Complexes

Most tape systems today are automated tape libraries. Using a shelving system with slots for the tape and robots to receive the tape to mount them in a tape drive for I/O. Usually each library unit has a dedicated area where tape drives can be installed. The drives are standardized to some extend so LTO compliant libraries usually have no problems with drives from other vendors. In the past, also circular tape silos were common, but all the commercial highly scalable systems are tending towards rectangular libraries.

Most systems arrange the tapes in vertical shelves thus in two dimensions, so that robots can easily pick and place the tapes. But this is not the most space efficient way to store the tapes, which is why some vendors offer special high-density libraries. Spectralogic, for example, chose to store tapes in a kind of drawers to achieve a small high capacity footprint. High tape densities also can be quite attractive for gradual transition strategies, because sticking with 2-3 tapes of older tape media can be cheaper then always using the most recent tape generation. Most enterprise tape systems are scalable, by adding additional racks/frames or library units. Each vendor is using a slightly different vocabulary, but a combination of multiple units into a larger structure is generally considered a library complex. Surprisingly, while most systems should in theory be extendible without limits, all vendors have artificial limits for size the library can handle. It is also notable that each vendor has at least one unique selling point.

IBM TS3500 Series While IBM markets a higher density variant with the TS4500, the TS3500 is older and scales much better supporting up to 2.25 exabyte of storage. IBM counts in frames that can be connected to form a library. Special high density frames can accommodate up to 3592 cartridges. Up to 16 of such frames can be combined into a library supporting up to 192 drives. Using overhead pass-through lanes, it is possible to connect up to 15 libraries to a library complex. Allowing for a maximum if 2880 tape drives. IBM uses robot hands that can fetch up to two tapes, thus allowing to reduce robot movement. Refer to Figure 2.4 for a photo of an TS3500 library complex.

Figure 2.4: IBM TS3500 Library Complex (IBM, 2011b)

Oracle StorakeTek SL8500 Using a so called "centerline" layout for the drive bays the SL8500 has the most exotic extension schema. Positioning the drives in the center is advertised to guarantee short travel times for the robots averaging at below 11 seconds per robot, regardless of the size of the library complex. Each library can be extended with up to 5 so called Storage Extension Modules (SEM) allowing for up to 10.088 slots handled by up to 8 robots per library. Using pass-through-ports, it is then possible to connect up to 32 libraries to form a library complex containing over 320.000 cartridges and up to 2048 drives. Refer to Figure 4.7 for a photo of an SL8500 library complex.

Figure 2.5: Oracle StorageTek SL8500 Library Complex (Oracle, 2015)

Spectralogic TFinity With space for 400.800 LTO cartridges, the TFinity can handle by far the most tapes in a single library complex and is maintaining the smallest footprint while being able to provide up to 7.6 EB of storage. It does so by organizing tapes in little drawers called TeraPacks. On the other hand, the number of supported drives is comparable moderate with 920 drives per complex. The TFinity consists of frames which can be extended in two directions. Using am over head pass-through mechanism multiple, such racklines can be connected. Refer to Figure 2.6 for an illustration of an TFinity library complex.

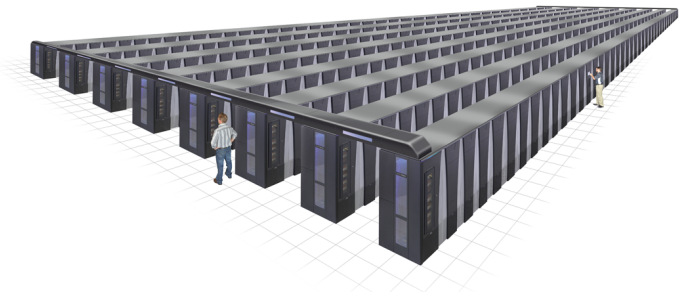


Figure 2.6: Spectralogic TFinity Library Complex (Spectralogic, 2016b)

Quantum Scalar i6000 Turning to standard 19" racks the libraries by Quantum are among the smallest but also the least scalable systems by a major vendor. The i6000 can be extended into two directions to a total of 16 modules and supports up to 96 drives (2011) and dual robots for tape-retrieval. Special high density library modules are also advertised. At only up 12,016 cartridges it is possible to manage up to 192 PB (using LTO-7). Refer to Figure 2.7 for a photo of an i6000 library complex.

Figure 2.7: Quantum Scalar i6000 Library Complex (Quantum, 2015)

HP StoreEver Tape ESL G3 HP also relies on 19" racks for it's library complexes. The performance and features are very similar to the ones given for Quantums i6000. HP's StoreEver Tape ESL G3 can be extended to form library complexes with up to 16 library frames and support 180 GB on up to 12,006 cartridges. Every frame can host up to 24 tape drives accounting for a maximumg of 192 drives per library complex. Refer to Figure 2.8 for a photo of a StoreEver TapeESL G3 library complex.

Figure 2.8: HP StoreEver Tape ESL G3 Library Complex (**?**)

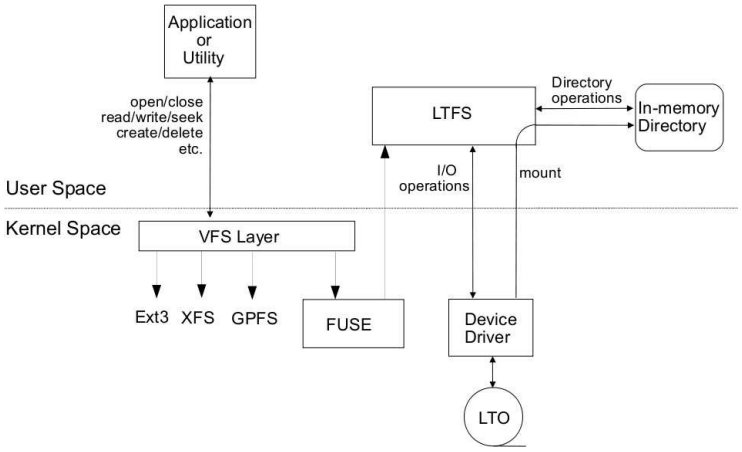


Figure 2.9: How LTFS allows tape media to appear like a regular file system that can be mounted to the VFS (Pease et al., 2010).

2.4 Linear Tape File System (LTFS)

The *Linear Tape File System* (LTFS) provides a standard to access files on tape that is similar to traditional storage media, e.g., USB thumb drives. On older tape systems no metadata was stored on the tape, and tapes itself were not self-describing. Therefore, often external databases were used to keep track of files in tape systems, which is of course bad for interoperability when moving between systems. In 2009, IBM released an early prototype (NAB 2009), followed by the first official release 2010 with support by LTO Consortium. LTFS is often used to refer to both the standard as well as to implementations. LTFS offers a number of features the most notable being the following:

- Self-describing cartridges using XML to describe file-system
- Hierarchical directory structures and metadata that is compatible to other file systems
- Custom attributes definable by users and organizations
- High interoperability supported by the LTO Consortium
- Support for small data files close to the beginning of the tape
- Support for sparse file types [1]

In Figure 2.9, it is illustrated how LTFS interacts with the operating system and the tape systems to provide similar properties to removable mass media. When mounting a LTO tape cartridge, LTFS loads the file system information and provides a POSIX-like file system to the Virtual File System (VFS) via FUSE. Applications and users that issue I/O operations can act on the data like they are used to with other filesystems by using standard I/O calls.

[1] Sparse files, while often large in filesize, have continuous regions that carry no valuable information. This can be exploited to reduce I/O by omitting to write these regions. Sparse files are common for many simulations, e.g., vegetation is unlikely to be found on the oceans.

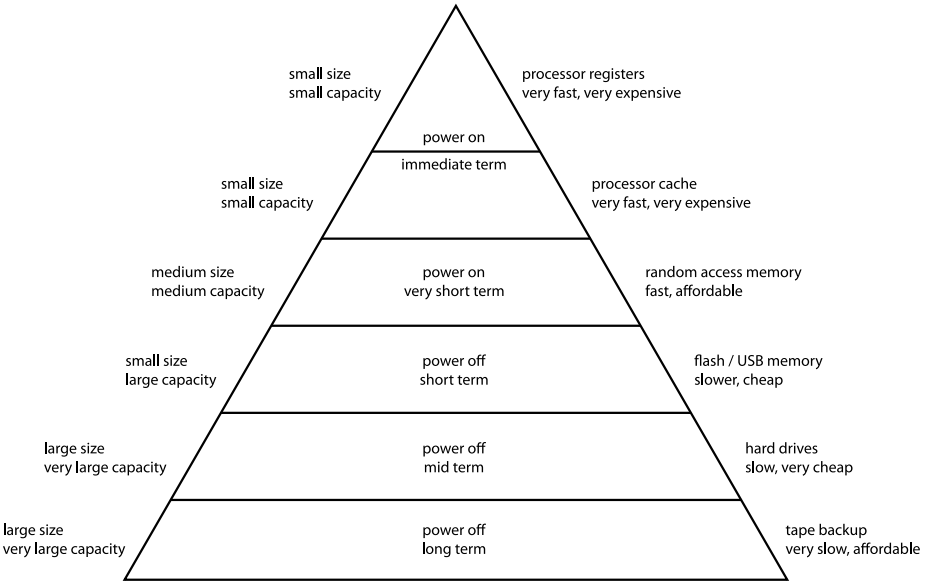


Figure 2.10: The computer memory hierarchy (Quibik, 2010).

2.5 Hierarchical Storage Systems

Different technologies that serve a similar purpose come with different qualities and storage technologies are no exception. Often it is possible to combine multiple different technologies to achieve certain desired performance properties that would be impossible or much more expensive than when using only one. And, in fact, computers and storage systems in general employ a wide range of different technologies. This derives a sort of storage hierarchy. For example, the fastest technologies that are easy enough to produce, only work when some sort of power supply is present. To this fact, we can attribute at large the existence of storage such as volatile random access memory (RAM) and persistent hard drives.

A common pattern is to used caches, which improve access times for recent or regular accessed data considerably. The basic idea for hierarchical storage management is is consequently to stage data at a higher level, in the hierarchy whenever faster access to the data is desired. E.g., data might be processed in a certain order, but seeking data for the next work package can be performed in parallel to another useful computation that does not require to use I/O.

2.5.1 Virtual Tape Libraries (VTL)

For a while replacing tape archives with spinning disk was quite popular. Multiple efforts (Lingfang Zeng, 2005; QUADStor, 2015) include the emulation of tape systems commonly referred to as *Virtual Tape Library* (VTL). The goal of emulation, of course, is not to gain insight on how to best use the system but to maintain compatibility with otherwise deprecated or incompatible technology. Often this is done when a technology already enjoyed wide spread adoption and is due to be replaced by a newer technology. As legacy in terms of hardware

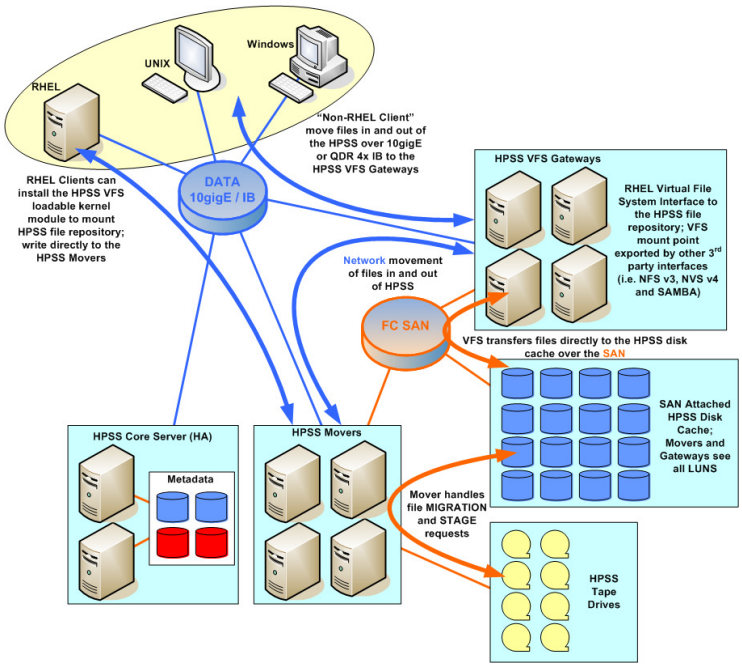


Figure 2.11: A schematic of the basic building blocks forming HPSS as described by IBM (IBM, 2011a).

and software is associated with any organization that's not newly founded, it is often more cost efficient to choose emulation for some time though it may come with other drawbacks. Emulation, therefore, usually follows pragmatic considerations and is, in the context of tape is not used to gain insight on system exploitation.

2.5.2 High Performance Storage Systems (HPSS)

A hierarchical tape storage solution that integrates transparently into a data centers storage system marketed by IBM is High Performance Storage System (HPSS) (IBM, 2011a). It is a joint effort by five national laboratories, the Department of Energy (DOE), and IBM. A schematic overview of the system is illustrated in Figure 2.11. Data is staged by so called "Movers", where data is basically cached on a disk based filesystem to be served by I/O servers and tape drives. Metadata is stored in IBM's DB2. HPSS is advertised as being capable of handling archives with many hundred million files and tenths of petabytes. Furthermore, the HPSS tape archives can be partitioned into namespaces which maybe used to provide different service levels etc. A HPSS file system can interface with Linux VFS and even provide read/write POSIX semantics. As an alternative special kernel modules can be used to interact with the "Movers" directly. HPSS is not limited to work with IBM hardware but supports the Oracle StorageTek SL8500 Libraries.

2.6 DKRZ Storage Archive

The German Climate Computing Center (DKRZ) is a national facility and provides various services related to climate research including (DKRZ, 2016):

- Providing the compute power and architecture necessary to run climate simulations (HLRE-3/Mistral).
- Providing storage infrastructure to archive simulation results for decades.
- Offering expertise to scientists to maximize system utilization and optimize scientific workflows.
- Allowing visualisation of the simulation including accurate 3D rendering for in-situ analysis and post-processing.
- Carrying out complex simulations which are often part of international research projects.
- Informing the public on high-performance computing and climate research in guided tours.

Hosting the storage archive is only one of many responsibilities. Because data management is a vital part of any research effort and the thesis is interested in tape archives, a brief analysis the storage system currently deployed including the main objectives (DKRZ, 2016) should be performed. The DKRZ documents the research data with the World Data Center Climate (WDCC). The long-term archive of climate data again serves as documentation, but also as a resource for data mining for external users. In an on-going effor the accessibility should be improved through standardized metadata and datamodels. The data shall be safe also in case of disaster.

To achieve this the DKRZ is using IMBs HPSS introduced earlier in Section 2.5.2 in combination with eight Oracle/StorageTek SL8500 (covered in more detail in Section 2.3 and Section 4.5.3). The archive is partitioned to optimize for different file-sizes and to increase data safety. For reasons of fire security and control the facility is segmented into two areas separated by fire protection doors. The current setup has six storage silos in the first area and one silo in the other. In addition one silo is off-site for disaster recovery.

- 8 automated StorageTek SL8500 Tape Libraries
  - a combined 75.000 slots for tape cartidges
  - 65 tape drives of different generations (LTO-4/5/6 and T10000B)
  - 8 robot arms per library to mount tapes
- 5 Petabyte disk cache for HPSS
- 15 Gigabyte/s sustained bidirectional bandwidth (18 GB/s peak)

Summary

Magnetic tape has a long standing history in computer science. With LTO, there is an industry standard that provides operators with some level flexibility and planning stability. Modern tape libraries come with a variety of features including WORM, data compression, and encryption, as well as convenience functions for easy archival and library management. Multiple competing archive systems are available. Interestingly, each system has unique advantages making a recommendation dependent on the requirements of an organization. LTFS radically changed how users and libraries interact with tape, making tape access comparable to USB flash drives. Sometimes tape can be replaced transparently, e.g., with disks by the use of VTL. Finally, pure tape systems are ceasing in relevance but tape is increasingly integrated into a hierarchical storage model. However, the operation of such systems usually depends on proprietary hardware and software maintained by the vendors, sometimes in cooperation with some national research institutes.

Chapter 3

Related Work

This chapter presents the state of the art and related work in the field. For a more structured presentation, innovations and publications are grouped by their respective research area. Section 3.1 covers mostly physical and hardware improvements to tape systems. Section 3.2 where research focusing on algorithmic improvements with a dedicated subsection to insights on scheduling and load balancing strategies. Section 3.3 focuses on the improvements of hierarchical storage systems. Section 3.4 compares different frameworks for simulation and modeling methodology. Performance evaluation of tape drives and libraries with related key metrics are covered in Section 3.5

3.1 Physical and Hardware Improvements

Efforts to improve tape storage system can focus on advancing the technology that is used to read and write tape. This is mostly in the domain of vendors and not much of the research conducted is published to protect a business advantage. Multiple efforts focus on improving the data placement on tape (Dashti and Shahabi, 2000) or the magnetic representation (Dee, 2008). Such efforts can be, to some extent, also considered algorithmic improvements, but tape drives and hardware generally does not provide fine grained control to the users. In addition, certain data layouts become only possible when physical means of manipulation are discovered. Fir example, magnetic fields while limited in their range, still can effect neighboring areas putting limits on how dense data can be stored on tape. Pantazi et al. (2015) covers a list of challenges faced by tape storage technologies. A good example is the necessity to develop bi-directional read/write heads that enabled serpentine tape in the first place. Pease et al. (2010) acknowledges the challenges that are introduced by shingled writing that prevent making changes after imprinted on tape without rewriting other parts of the tape as well. This is also the reason why tape is in many cases considered an append-only storage medium.

3.2 Algorithmic Improvements

Following breakthroughs to physical challenges, great potential is to be found in the ways we exploit the available technology. In fact, undesirable properties of a possible manipulation can be turned into features when used in another context. Section 3.1 already touched data placement strategies to some extend. (Dashti and Shahabi, 2000; Pease et al., 2010; Pantazi et al., 2015) all explore possible ways to layout the actual data on the tape, though with varying objectives. For example, it is possible to trade seek time for space, or reduce the number of

passes to find data on the tape. Another example, that is dependant on the way data is accessed suggest to remap data structures (Jenkins et al., 2014) before writing them. While not specifically developed for tape, strategies that work for other media types often are applicable. Zhang et al. (2006) explored different object placement strategies with in tape libraries to optimize tape switch, data seek and transfer times using a simulation. Other improvements are enabled by changing the dynamics, for example WORM media allow for radically different strategies then traditional I/O. Other approaches include using application-specific compression (Hübbe and Kunkel, 2012) or encryption.

Further approaches strive to improve the reliability of tape by introducing redundancy. Berlekamp (1975) looked at algebraic codes for more reliability of tape data by allowing for error correction. Many strategies that work with disk or other storage media of course also apply to tape. One notable example which was researched but has not yet found their way into many production system is RAIT (Hughes et al., 2009) or TapeRAID though combinations of traditional RAID and Tape have been explored (Lingfang Zeng, 2005). To allow higher levels (including users) to better utilize the technologies, MPScan* provides a model to estimate access times for serpentine tape drives (Sandsta and Midtstraum, 1999), obviously there is value here to improve the scheduling however it remains open if the algorithm is still valid for todays technologies.

3.2.1 Scheduling and Load-Balancing Algorithms

Scheduling problems are among the oldest in computer science and a number of general purpose schedulers exists. But over time specialized schedulers for different tasks where developed. E.g. many different sophisticated I/O schedulers are built into the Linux kernel. Getting into too much detail is beyond the scope of this thesis but to asses which approaches may be promising a number of relevant approaches is collection in this section. For tape and hierarchical storage management scheduling and balancing algorithms occur in at least three different contexts:

Load Balancing (Bandwidth and Workload) When work is distributed across multiple workers, a common problem is that work is not evenly split between all workers because some workers are working on harder problems than others. A *load balancer* assigns work to prevent some workers from idling while others working hard.

I/O Scheduling Processing I/O in the order it arrives often leads to an under-utilization of of the hardware. By reordering and postponing some activity the hardware can be used more efficiently.

Robot Scheduling Similar to I/O schedulers the goal is to get more work done by performing a number of tasks in a more efficient order.

Xu and Lau (1996) covers different load balancing approaches for parallel systems extensively. A helpful overview to load balancing algorithms can be obtained by grouping the different types of approaches in a hierarchy Figure 3.1. Similarly different classes of more or less universal scheduling strategies exists.

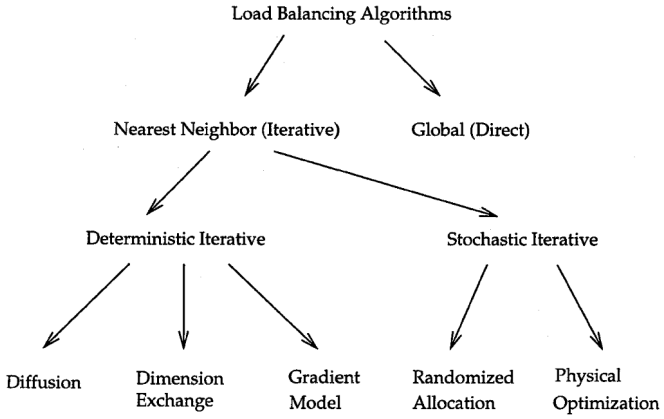


Figure 3.1: Taxonomy of load balancing algorithms by Xu and Lau (1996).

Often the name of a scheduling algorithm already suggest how basic strategy works. Some examples for common scheduling approaches are:

First In, First Out (FIFO): The first task that arrives, is also the first to be served. In many situations the order in which data is process matters, as such it is commonly used by buffers.

Last In, First Out (LIFO): The last task to arrive, is the first to be processed. It is commonly used when one operation performs another and has to wait for the result before itself can proceed.

Random Scheduling: From a list of available tasks, one is taken randomly. This strategy is useful when fairness is a requirement.

Shortest Seek First aka Shortest Seek / Service Time First (SSTF): A I/O related strategy employed in disk drives. The disks read/write head is always piloting towards the position of the closest seek request.

3.3 Improving Hierachical Storage Systems

The department of Energy (DOE) is investing considerable efforts and bases decisions on research conducted using super computers, making it one of the largest civilian operators of large scale data centers. On a regular basis, a report assessing the state in the field of scalable Input/Output is published to identify where research efforts should be concentrated. For tape, the report acknowledges its relevance for long term storage, but finds that most tape systems are operating relatively independently of the rest of the storage architecture. As in situ data analysis is anticipated to increase (Ornl et al., 2015), which will also involve access to archived data, a better integration into of tape into the storage architecture is demanded.

Pure tape systems cease in relevance and hybrid and hierarchical storage systems promise to provide cost efficient solutions with the best properties of all technologies, this section collects research made to improve HSM. Dee (2008) stresses the opportunities of automation, which enabled scalable solutions that seamlessly integrate into storage hierarchy at large. For example, Koltsidas et al. (2015) focuses especially on the integration of disk and tape. In addition, for such efforts to become feasible advancements in network technologies have been necessary. Other approaches try to integrate tape into the existing architectures, most notable are HPSS (IBM, 2011a) and LTFS which were discussed already in Chapter 2 in more detail. LTFS blends the boundaries between tape and disk based storage systems by transparently providing a POSIX like file system. LTFS was first introduced by Pease et al. (2010), but other research extends on the idea. For example, Real et al. (2015) propose an advanced I/O scheduler for dual-partitioned tapes to further improve LTFS. A lot of information in this area appear to be influenced by marketing, some sources suggest that LTFS currently does not live up to hopes that where placed into it.

3.4 Simulation and Modeling

Discrete-event simulation (DES) is a simple and effective way to model and simulate a wide variety of problems. Activity is represented as a discrete sequence of events, where a each event marks a state where the system changes. The advantage of DES in comparison to, e.g., a continuous simulation is that it is possible to "hop" from event to event without requiring any computations even if a long time has passed between events. Theory is established around the topic, in particular a formalism termed *Discrete Event System Specification* (DEVS) enjoys broad acceptance. Various frameworks or simulation kernels are available that aim to make setting up simulations easier by abstracting the most general concepts. Many of them are open source, though also commercial applications are available. Unfortunately, many are not longer actively developed or come with a small user community.

adevs (Nutaro, 2016) Written in C++ and released under the BSD License adevs implements DES based on the principles on parallel and dynamic DEVS. Adevs uses so called atomic models to govern the behavior of individual actors these then can be connected using a network model. The suite further supports continuous models as they are common especially in systems involving physical components. Care was taken to support multi-core computers.

DESMO-J (Page, 2016) Written in Java and published under the Apache 2.0 License, DESMO-J aims for an object oriented approach to discrete event simulation. Developed since 1999 at the University of Hamburg DESMO-J is referenced in no less than 200 publications and appears to be under active development as the last release is from November 2015. DESMO-J follows a philosophy of strict separation of model and experiment.

SIM.JS (Varshney, 2016) Written in JavaScript and licensed under the LGPL, SIM.JS is an interesting framework for being able to run in a browser thus making it accessible through a browser. Examples featuring a GUI for drag and

drop model building are included. Being HTML and CSS it allows for a lot of visualization opportunities thanks to the large community and the number of JavaScript frameworks for virtually any kind of problem. While the last commit is from 2012, the author seems still very responsive.

SimPy (SimPy, 2016) Written in Python and licensed under the MIT License SimPy follows a process-based approach to event simulation. Using Python generators, it is possible to model active components. SimPy also provides support for real-time simulations. Unfortunately, the simulation time is kept as a single integer value.

3.5 Performance Evaluation

It seems reasonable that most of the tools and the performance metrics used to asses and benchmark I/O systems in general are also applicable to tape systems. But the workloads that are well suited for tape system differ a lot due to the unusual performance characteristics of tape. Johnson and Miller (1998) looked at performance measurements of tertiary storage devices and came up with several metrics wich mostly appear valid also for todays systems:

- Robotic arm access time
- Mount/Unmount time
- Seek time / Spool time
- Transfer and compression rates

An interesting pitfall that once again stresses how little information value individual peek performance metrics can have is that many of the drives tested by Johnson and Miller (1998) had large discrepancies in startup time ranging between 7% and 40% of the tape seek time. As a result, for drives with high startup costs other strategies such as file placement may become negligible while they make a lot of sense on another device.

Summary

Physical breakthroughs are what enable technology to work in the first place. For magnetic tape storage, the main principles have not changed much in decades. Improvements to the properties of tape and increasing the capacity of tape cartridges is mostly in the hands of vendors. The same holds to true for the hardware such as the automated tape libraries. Often algorithms and physical boundaries are intimately related. For example, the desire to use certain access strategies such as those possible with serpentine tape required the development of bidirectional read-heads first. A lot potential to improve the performance also can be found in the data itself so approaches that remap and compress data are also relevant for tape archives. Similarly, adapting proven algorithms from long standing fields such as scheduling or load balancing are becoming more relevant, especially as automated systems are integrated into the storage hierarchy. RAIT has potential to substantially boost tape performance yet drive costs maybe

prohibitive to some extent. The transition to hybrid systems creates new opportunities for research, that is very complex to asses as the interaction between many competing and dependent feedback mechanisms are involved. Experiments with physical systems is not always feasible to explore more exotic approaches making simulation an indispensable tool for insight.

Chapter 4

Modeling Hierarchical Storage Systems

This chapter describes the requirements for a comprehensive model to simulate tape systems in hierarchical storage systems based on considerations introduced in the previous chapters. Section 4.1 recalls the requirements and objectives the model should satisfy. Section 4.2 collects the key components involved in tape library systems. Section 4.3 starts with the workloads that occur in the system and describes how they are processed. Section 4.4 covers relatively small models related to the workings of physical components such as drives and tapes. Section 4.5 covers what is necessary to abstract a tape library. This is biased to some extend by what is necessary to describe StorageTek SL8500 complexes. Section 4.6 takes a look at different ways to model how data moves through the network. Section 4.7 lists the types of software components involved to run tape systems in hierarchical storage systems.

4.1 Model Objectives

The models to be employed need to reflect what kind of insight we would like to gain from simulating a complete tape system, which was covered more in-dept in Section 1.4. Ultimately, the goal is to improve the quality of service (QoS) of tape systems from a user a user perspective and cost-efficiency from a data center perspective. Using the results of simulations for different setups, we then would like to make more educated decisions and possibly infer rules of thumb about the setups which suit a specific requirement best. To do so we have to keep track of a number of performance metrics which need to be collected during simulation. Based on these metrics, it is then possible to benchmark and compare different setups. Key metrics for I/O systems and tape systems in particular include:

- Cost for: acquisition, energy, wear parts
- Transfer rates and indication of limiting components
- Wait and access times for robots, mount, seek/spool

Identifying the relationships of these metrics in later analysis has to allow the generation of reports on the TOC/ROA and the QoS of a chosen configuration. Ultimately, it would be nice if these would be in a form that allows us to treat most questions as an optimization problem. Thus, we can fix certain

parameters and others can be varied, e.g., using the Monte-Carlo method. Obvious candidates may be varying the number of tape drives and tape generation to find the highest throughput.

4.2 Model Overview

In order to model tape systems in hierarchical storage systems, it is not sufficient to limit efforts to the tape libraries alone. Instead, it is necessary to consider the complete path data takes from the clients all the way to the tape and the other way around respectively. In Chapter 2, we looked at different components that are commonly found in modern tape systems. Here we want to find an abstraction that captures the necessary detail without introducing to much complexity. By following a request through the system and anticipating how it is handled on the way, we identify the *key components* and can describe a typical *data path* that is common to most HSM systems.

4.2.1 Data Path

While in actual tape systems more components are involved and a lot of complexity stems from utilizing all the components to realize a scalable solution, it appears sensible to start with minimal approach and iteratively add complexity later on. As far as a single request is concerned Figure 4.1 illustrates the different components that data has to pass very well:

1. Multiple *clients* which may issue requests to read of write
2. An *I/O Server* to receive and handle the requests
3. Different *cache levels*, to speed up access for recently touched files
4. An automated *tape silos* and *tape drives* to access the archive

All components are connected through a network. Modeling networks is a well researched field, nonetheless a network model needs to be implemented to work with the system specific components. It is important to identify the main bottlenecks for tape systems. A comprehensive approach appears to follow the path data takes in the system to find out which are the limiting factors. As with other I/O systems seek times times and data transfer rates are among the determining factors.

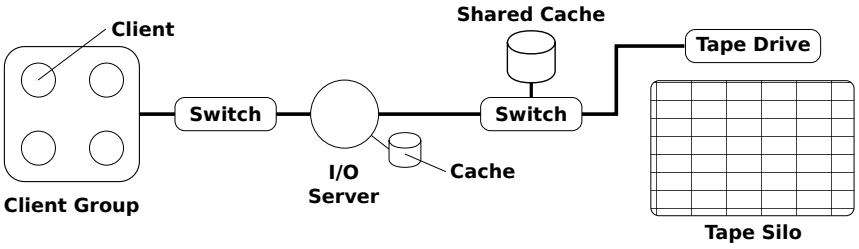


Figure 4.1: The path data takes when read or written to from/to a tape archive in a hierarchical storage system.

4.2.2 Key Components

Clients: A client issues requests to read or write files stored in the tape system. Sometimes clients work coordinated and form *client groups*. For example a parallel job that is started on multiple nodes.

Switches: To connect components in the network, switches are used. Switches and any network component are limited in their bandwidth. Switches simplify modeling the networks because a switch represents a connection between all attached components.

I/O Server: The I/O Servers handle incoming requests. They are used to delegate and move data between the clients, a global disk cache and the tape system. As huge quantities of data are moved on modern systems serving thousands of clients it is common to have multiple I/O servers. Sometimes I/O servers are exclusively moving files between the disk cache and the tape archive.

Caches: Most modern I/O systems utilize caches to speed up operations on recently used data by keeping copies on faster, but usually more expensive, storage technologies. Two types of caches are relatively unique to tape systems: A large shared cache cache usually with disks, and local caches for I/O servers.

Tape Drives and Tape: To read and write data on tape special drives are required. A variety of different technological approaches exists. At the moment, innovation with the most impact and performance improvements is directly related to the capabilities of the tape drives itself.

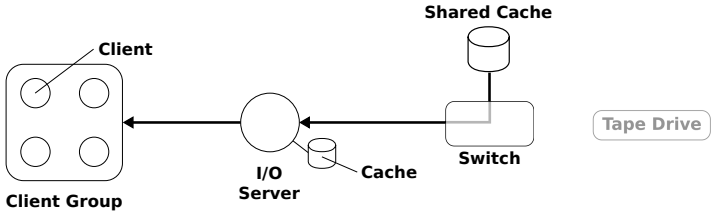
Tape libraries/silos with robots: Tape is accessed automatically by using tape libraries. A critical part to the performance of tape libraries are the robots and the way the tape cartridges and tape drives are organized in the library.

Software: The interaction between the different components is governed by various algorithms and software components. There is load balancing for tapes, drives and I/O servers. Resource and file management to allocate and access data. And scheduling of drives, robots and requests to optimize throughput.

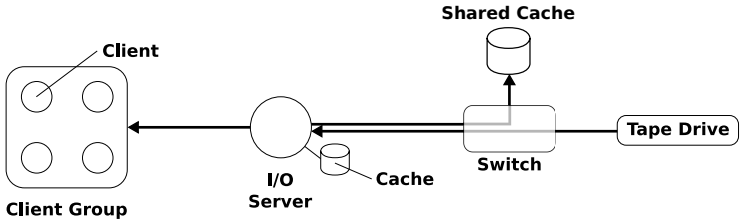
4.3 Request Types

Different request types occur on tape systems, a number of these requests are related to implementation details and they will not be modeled. Most requests are generated by clients but sometimes migrations or other service related workloads are issued which can have impact on the overall system performance. Access to libraries is provided either through a POSIX-like interface or via FTP usually to copy from or to a POSIX-like system. Typical calls are:

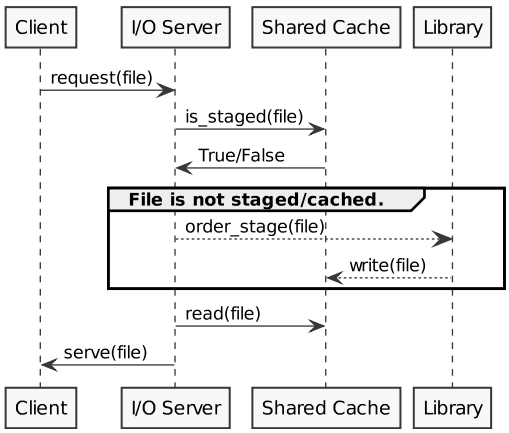
- POSIX like
  - `open`, `read`, `write`, `seek`, `unlink`, `stat` etc.
- FTP/HPSS
  - reads: `PST_Cmd`, `STOR_Cmd`
  - writes: `PRTR_Cmd`, `RETR_Cmd`



(a) cached



(b) uncached

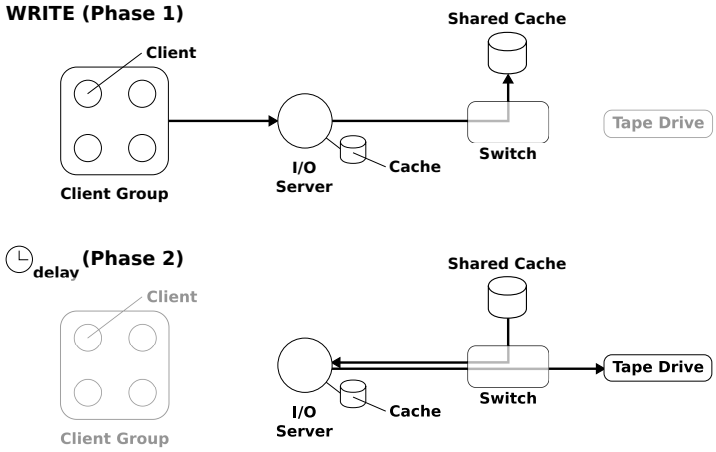


(c) UML Sequence Diagram

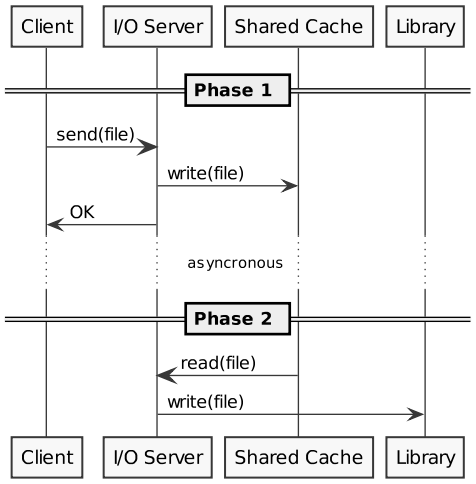
Figure 4.2: Schematics for handling a) cached and b) uncached read requests.

4.3.1 Reading

Read requests can be handled in two ways dependent on whether they need to be read from tape or if they can be served directly from cache. Figure 4.2 illustrates the ways the data flows. In the case where the requested file is not cached (b), the tape needs to be mounted in a tape drive and data is send to the I/O servers. The I/O server then reads the file and sends it to the client. A copy of the data kept in the cache. For requests to files which were staged earlier (a) the I/O servers can read directly from the shared cache which can be considerably faster. The process is also depicted as a UML sequence diagram (c) which captures the responsibility of components.



(a) Data flows in the two phases of processing a write request.



(b) UML Sequence Diagram

Figure 4.3: Schematics for handling write requests.

4.3.2 Writing

Writes are handled in two phases. Analog to the illustrations for read-requests, Figure 4.3 shows how data is flowing from clients to the cache and ultimately to the library. The UML sequence diagram describes the control flow and the responsibility of the different components. The phases are performed as follows:

1. In the first phase, the I/O Servers receive the request and direct data to a shared disk cache.

2. In phase two, independently of the original request, the data is copied to tape.

4.4 Hardware Components

Section 4.2 briefly introduced the most important components. This section will focus on the hardware components in more detail. As a lot of hardware has it's own software, the boundaries between software and hardware components are often blurred, especially when APIs to control hardware behavior are exposed.

Many hardware components are, to some extent, exclusive. For example with only one drive we can only read one tape at a time. Other resources are available in certain quantities. Usually computer hardware can be considered as not consumable, thus, after using a resource it can be used again. Though also consumables exists in tape systems, for example special service and cleaning cartridges are required to clean the tape drives reading heads, which can be used only for a limited number of times.

4.4.1 Fault-Tolerance

Another factor to consider is that hardware sometimes fails. A drive or a tape may break, or a robot might get stuck. Failure behavior and characteristics are well formalized and many vendors have to publish failure rates as part of their specifications. In particular four tape library related measures are commonly encountered:

Mean Time Between Failures (MTBF): gives the average time that passes between two downtimes of the system. To model a complex system, we are interested in the MTBF for the individual components. Usually a failure that does not halt the system operation, but may degrade performance considerably, is not accounted for in MTBF.

Mean Time to Recover (MTTR): describes the average time it takes to repair a component before it becomes operational again.

Mean Swap Between Failure (MSBF): is a measure sometimes seen for tape systems. It borrows from the principle of MTBF but replaces the measure of time with the number of tape can be exchanged before a failure occurs. Sometimes this is also called Mean Exchanges Between Failure (MEBF).

Mean Time To Failure (MTTF): As the name implies MTTF measures the average time until a component or system fails in ways it can not be repaired. It is thus usually not appropriate for computer systems that are actively maintained, though it is for individual components which need to be replaced.

4.4.2 Tape and Tape Drives

Tape and drives to access them are intimately related and they can not be modeled reasonably when ignoring one of the two. The relation thus introduces a variety of compatibility issues which are averted mostly by standardization efforts. For example, LTO warrants for backwards compatibility for up to two generations when reading from LTO tapes. While this slows innovation to some extend, it also protects investments. Another consequence is that vendors can

innovate mostly by improving their tape drives which is why the tape model remains fairly simple.

Tapes

Tapes are differentiated by the **cartridge form factor**, the tape technology **features** (e.g., data, WORM, cleaning, diagnosis) and by their **length**. Cartridge and function can be expressed as simple identifying text strings. The length depends mostly on the thickness of the tape, thus the length may be implicitly limited by the specifications of a standard. The capacity of a tape is not controlled solely by the tape, but also depends on the drive. In addition, tapes are usually labeled with barcodes, so that if we want to model a diversion from the intended use, this should be reflected in the label so that a library does not serve tapes to incompatible drives. Furthermore, a drive might detect and eject a tempered tape. Special tape formats may have additional features such as *write-protection* or an *EEPROM* to store some information about the tape. As such, a tape may hold information on it's fill level. Usually most of these information are also handled by a library management software as otherwise sensible scheduling is not possible.

Tape Drives

Tape drives are somewhat harder to abstract; a wide variety of different drives exists. A tape drive is usually compatible to a very limited amount of cartridge form factors and maybe limited to the most recent generations of this cartridge type. Dependent on the mounted tape cartridge, the drive can have different performance characteristics. Thus, the performance for individual cartridges depends mostly on the drive, and to a limited extend on the tape itself. Each tape drive needs to maintain a list of compatible cartridges and specify how it performs with these cartridges for a given mode of operation. The important factors, most of which are based on the tape, are:

- Compatibility
  - Read-Compatibility
  - Write-Compatibility
- Read and write performance/transfer rates based on the tape
- Capacity based on the tape
- Spool/wind speeds based on the tape
- Time on mount/unmount based on the tape
- Multi-Channel support or the number of Read/Write heads

The model does not account for the case of under-utilization: If the tape drive does not receive data fast enough, the drive needs to reposition the head, which dramatically degrades the tape drives performance. Using the listed information, it is then possible to compute an approximate performance for handling a request and how long the drive remains unavailable (see Section 4.4.3).

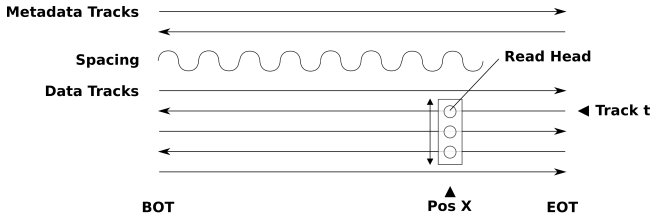


Figure 4.4: Serpentine Tape Model

4.4.3 Tape-Seek- and Drive-Busy-Time Models

Section 2.1.1 introduced three layouts for writing data on tape (linear, linear-serpentine and helical-scan). We will consider only linear-serpentine tape as it appears to be the most relevant on modern systems. Figure 4.4 takes the perspective of the tape drive and illustrates how data is actually read or written on tape. An array of read/write heads can be positioned relative to the tape in two dimensions and imprint or read a magnetic signature as the tape passes underneath. When spooling to a specific position it is possible to move the tape quite fast, when reading or writing lower speeds yield the best results. By measuring or estimating the following characteristics, it should become possible to approximate the time it takes to serve a request and how long a drive remains busy:

- $pos := (x, t)$: a tuple describing horizontal x and vertical (track) t displacements relative to the tape. pos_{BOT}, pos_{EOT} are used to reference the Begin-of-tape and End-of-tape. A single reel cartridge is mounted and unmounted with pos_{BOT}.
- T_{mount} and $T_{unmount}$: the time it takes to mount and unmount a tape
- v_{spool}, v_{head}: the speeds to reposition the tape and read-heads
- v_{read}, v_{write}: the speed in, e.g., $bytes/second$ to read and write

The time to transition from a current position pos_i to target position pos_j is calculated as follows:

$$T_{seek}(pos_j, pos_i) = max\left(\frac{|pos_{ix} - pos_{jx}|}{v_{spool}}, \frac{|pos_{it} - pos_{jt}|}{v_{head}}\right)$$

The time $T_{read/write}$ to read or write from tape is calculated as follows:

$$T_{read/write}(bytes) = \frac{bytes}{v_{read/write}}$$

The time a tape drive remains busy T_{busy} would account for possibly multiple seek and reading phases, before it ejects the tape and becomes available again.

$$T_{busy} = T_{mount} + \left(\sum_{pos_i, pos_{i+1}}^{BOT,\ldots,BOT} T_{seek}(pos_i, pos_j) + T_{read/write}(bytes_i)\right) + T_{unmount}$$

These formulas do not consider the time it takes a robot to receive a tape $T_{receive}$ yet, as this requires to look at modeling the library topology first.

4.5 Modelling Library Topologies

When we looked at commercially available tape libraries in Section 2.3, the focus was to obtain an idea of the kind of systems which are available and what quantities of data they could manage. This section tries to abstract different aspects that are relevant for the performance of a library. What sets different library systems apart is the way tapes and drives are organized and how tapes can be moved from one position to another, thus the topological characteristics of the library. Besides the spatial distribution of the different components, it is sometimes possible to partition a physical library into multiple logical libraries. Partitions can change dynamics within a library dramatically. Modern libraries employ modular designs with the basic building blocks being as follows:

Enclosures/Shelving Units: Every vendor uses different names, but to realize scalable storage libraries multiple smaller units are designed in a way so they can be combined to form what is often called a library complex. Usually adding a shelf extends the library by a number of a few thousand slots for tapes and tens of drive bays. Some systems separate drives and tapes by offering shelves exclusively for tapes or drives.

Slots: A slot typically can hold one tape cartridge and usually they are arranged in grids or columns of slots. Some libraries use drawers to condense the required space to store tapes at the cost of an additional step before they can pick up the tape. Many libraries have reserved slots that are used for internal service operations or optimization.

Drive Bays: Drive bays accommodate tape drives. Usually a number of multiple drive bays are located close together as the drives need special power supplies and are connected to the network. The placement and the number of the drives within a library also has an impact on the performance.

Railing System: To allow robots to bring tapes from one point to another usually a rail is provided. This helps to keep robots simple and also to allow for robots to move quickly. A rail often spans multiple shelving units; thus the rails usually are more important to determine the logical library structure than the shelves are. The rails also supply the robots with power. Usually it is relatively easy to remove a robot from a rail.

Robots: A number of different robots are employed in the libraries. Older tape system even where using industrial robot arms to handle tapes, but nowadays these systems are usually outperformed by more specialized approaches. Most robots can move only in a linear fashion on a rail, and the robots that pick and place tapes are often referred to as robot hands.

Partitions: Logical constructions that allow to treat a single physical library as multiple virtual libraries. Usually moving tapes between two partitions is forbidden. Sometimes tape drives are exclusively reserved for use within a partition thus allowing to define QoS in the partitions.

4.5.1 Robots

The robots with the railing system are an integral part, which enables the other components within a library to fulfill their function. Robots come in variety of different shapes, but can carry only a few tapes at a time.

Robot Hands: The robot hands are usually mounted to the main railing systems. As such they can pick up tapes from a slot and serve them to a tape drive. The robot hands are also commonly equipped with a barcode scanner and sometimes proximity sensors to identify tapes and sense empty slots. Most robot hands can carry only one tape at a time. IBM also has system that can pick or place two tapes in rapid succession. Spectralogic can potentially move up to ten tapes at a time.

Elevators: For systems that have multiple vertically aligned levels of rails, elevators are available to move tapes from one level to another. The capacity of an elevator is limited, for example the SL8500 can move up to 4 tapes between level at a time.

Pass-Through-Ports: Pass-Through-Ports[1] are a mechanism to connect library complexes and move a limited amount of tapes from complex to complex. They are attractive to have because they allow for loading and unloading tapes through a single port and automatic rebalancing of tapes within a library.

Rotary High-Capacity Modules: Some libraries feature high capacity modules. A common schema is to have large drums with tapes vertically aligned that allows to access only a single column of tapes at a time. A motor can rotate the drum and change the front-facing column.

4.5.2 Generic Library Models

Ideally, it would be possible to find an abstraction that would generalize for the different types of library extension schemes that are available. But when considering past systems and current system trends it becomes clear that a generic approach will always introduce inaccuracies. It is hard to evaluate how severe the error is, without a direct comparison by experiment. Three models are proposed, each has a number of advantages but also comes with a number of pitfalls. We will start by discussing a simple approach which only uses graph theory and then turn to more accurate approaches, which use 2D and 3D representations. As software components such as the robot scheduler and path finding will rely on the topology, the idea is that the graph model is exposed to the software, but the underlying behavior and time calculation can be overwritten by more complex models when it appears appropriate. A similar mechanism is required for stateful components, e.g., for some of the high-capacity modules the tape access time is not fixed. From a user perspective, configuring a virtual library in all detail can become a tedious task.. So besides the abstract models, an additional layer is required that provides presets for the most common library systems and a GUI for configuration.

[1] The vocabulary is somewhat flexible and depends on the vendor.

Graph-based Model

In this modelling approach a graph can be used that represents the railing network or the paths tapes and robots can traverse. A vertex in the graph consequently is used for components and edges can be used to store distance or travel times from component to component. In principle, the approach is very flexible and arbitrary accurate depending on the level of detail. For highly detailed models, the approach can be tedious to configure and will be more expensive to compute. For lower levels of detail, errors may quickly accumulate. Figure 4.5 illustrates the concept, and in this case mixes distances and times; this is just one of many ways to interpret edges and nodes in a graph based topology. Usually an implementation will receive times and distances by invoking a method to allowing more detailed models to hide behind an edge or vertex. In most systems, a robot can not pass another robot, the exact rulebook however should be handled in a separate component that prevents illegal actions.

This model is attractive as we can utilize algorithms that are familiar from graph theory. The task of serving a tape that sits in `Slots-2` to `Drive-1` becomes the problem of finding the shortest path which is easily achieved using, e.g., Dijkstra's Shortest Path in $\mathcal{O}(|E| + |V|log|V|)$ or more generally the A*-Algorithm. Only positive weights are assumed. For large topologies with static distances, it may pay off to look into the *all-pairs shortest path problem* to avoid needlessly recomputing the same paths over and over again. For two vertexes v_i and v_j and edges e_{v_i,v_j} the time T_G to get from v_i to v_j calculates in principle as follows. v_{robot} is used to denote the maximum robot velocity:

$$\text{get\_time}(e_{v_i,v_j} \text{ or } v) := \begin{cases} t & \text{if } e_{v_i,v_j} \text{ or } v \text{ have time } t \text{ set} \\ \frac{\text{get\_distance}(v_i,v_j)}{v_{robot}} & \text{if } e \text{ but no time is set} \\ 0 & \text{otherwise} \end{cases}$$

$$T_G(v_i, v_j) = \sum_{v_i,v_j}^{\text{shortest\_path}(v_0,v_1)} \text{get\_time}(v_i) + \text{get\_time}(e_{v_i,v_j})$$

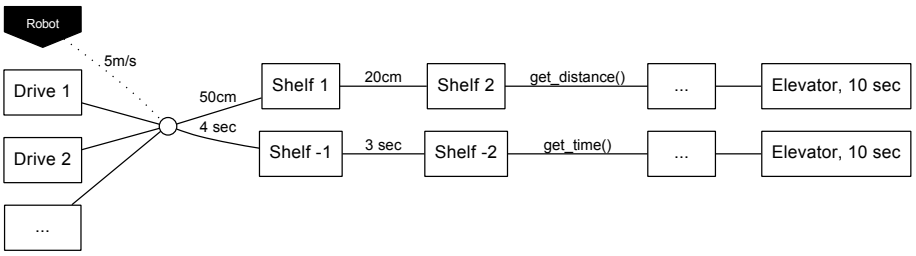


Figure 4.5: An example for a graph-based topology. The user can decide for every edge and vertex to provide either a time or a distance. Usually, similar components will consistently provide either the one or the other. The robot is currently positioned near the drive bays but may move from vertex to vertex.

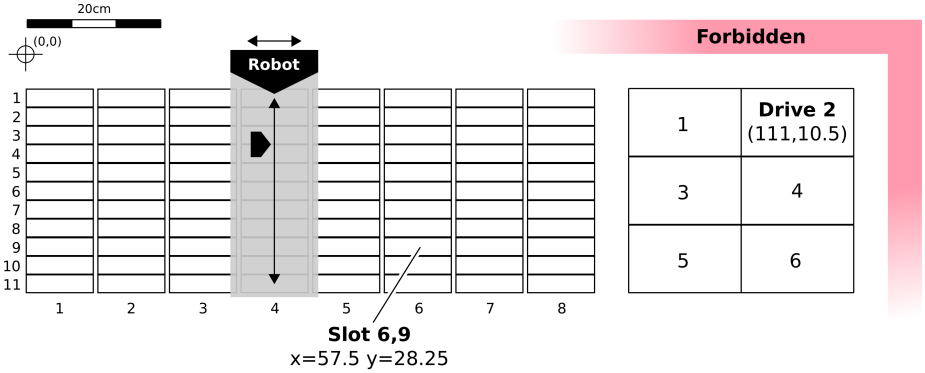


Figure 4.6: An example for a 2D topology model with a component mapping (in centimeter) coordinates and forbidden regions.

2D Topology Model

Sometimes projecting complete robot libraries into a two dimensional representation yields very good approximations. For the SL8500, this seems to be an efficient approach (see Section 4.5.3). The reasoning is that many significant movements that can be performed by the robots or the library are at most two dimensional anyways. Finding a path within a 2D model then becomes calculating the *Euclidean-distance* between a number of points and a check if the robot is crossing a forbidden area or an obstacle, in which case additional measures have to be taken. Movements usually are decomposed of multiple linear movements, thus care must be taken when calculating distances and travel times. Figure 4.6 illustrates one way to representing a library in just two dimensional space. Logical components are resolved to coordinates by providing a mapping function for, e.g., slots and drives. Mounting a tape placed in `Slot-6,9` to `Drive 2` requires visiting multiple coordinates. May $T_{2D}(path)$ be the time it takes to traverse a $path \in \{(p_1, ..., p_n) \mid p_i \in (x, y); x, y \in \mathbb{R}\}$. Assuming different robot velocities v_x and v_y for each axis, the total travel time may be defined by the sum of the time traversing between two points $T_{2D}(p_i, p_j)$ and possibly occurring work and wait times $T_{wait/work}$:

$$T_{2D}(p_j, p_i) = max\left(\frac{|p_{ix} - p_{jx}|}{v_x}, \frac{|p_{iy} - p_{jy}|}{v_y}\right)$$

$$T_{2D}(path) = \sum_{p_i, p_j}^{path} T_{2D}(p_i, p_j) + T_{wait/work}$$

An easing function $e(|p_{id} - p_{jd}|, v_{max})$ can to be used before taking the maximum should gradual robot acceleration be taken into account. The exact times also depend on other robots, which reinforce the need for a component that guards the behavior of robot as was discussed for graph-based topologies. 2D topologies can greatly reduce the burden to model, even on first sight, complex systems as such as a single SL8500. 2D models are limited when library complexes are connected or more exotic parts (e.g., a high-density rotary drum) is used in the system. In such cases, hybrid approaches that mix graph-based and 2D models are promising to achieve good approximations at reasonable effort.

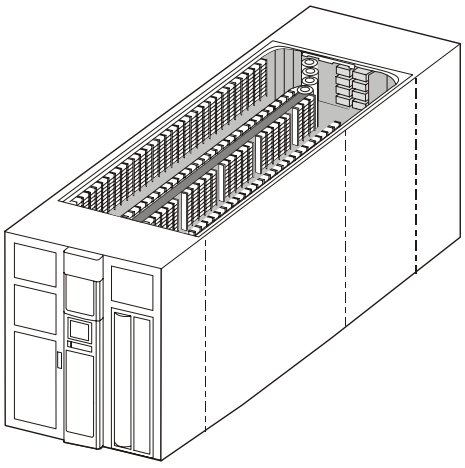


Figure 4.7: Oracle StorageTek SL8500 (Sun, 2006)

3D Topology Models

Full flexibility to cover all kinds of library systems are promised by turning to three dimensional models. Yet, for most library systems today, 3D models do not make a lot of sense when considering feasibility. When inspecting robots in detail, it is possible to find 4 or more degrees of freedom, but in many cases, the robots spend most of their time moving only a single dimension (rails/linear guides). For small actions, such as pulling the tape out of a slot it is easier to measure the time or simply estimate a time penalty on another basis. Yet, tapes are moved in three dimensional space and some rails follow curvatures, which may not always be trivial to map to a 2D representation. A 3D topology will result in a 3D model of a library with the special objects and paths added that are used to declare rails or forbidden areas or bounding boxes.

Path finding in 3D also is more expensive to compute. For example, finding the *Euclidean-shortest-path* when considering a set of obstacles in three dimensions is NP-Hard for the general case, though efficient algorithms that approximate the shortest path exist. As long as there are no systems with freely moving robots or levitating tapes it is probably not worth the effort. Though it may be attractive for visualizations and in combination with the other models.

4.5.3 StorageTek SL8500

The evaluation in Chapter 6 uses workloads provided by the DKRZ, which where collected from a system using multiple StorageTek SL8500s. For this reasons this section takes a closer look on modeling the SL8500 libraries to estimate access times and the library internal dynamics that will later influence the implementation. The SL8500 was already briefly introduced and compared to other commercially available libraries in Section 2.3.

The SL8500 is organized in container-like units called libraries which than can be combined to form library complexes. Figure 4.7 is an illustration of the SL8500's the internal structure in a slightly simplified manner. While the management software allows to combination of up to 32 library complexes, each

individual library complex can manage no more than 2.1 exabyte of storage with the current generation cartridges. A single library may hold up to 10.880 tape cartridges which are organized on 4 levels (1 - top, to 4 - bottom) each with a dedicated rail. Each each library consists of multiple modules, and the customer can decide on the number of Storage Expansion Modules to increase the number of slots. The four types of modules with relevant subcomponents in an SL8500 are:

- *Customer Interface Module* (CIM) (front)
  - Elevators (2 per CIM) (can move up to four cartridges)
- Up to five *Storage Expansion Modules* (SEM) (1728 Cartridges each)
- *Robotics Interface Module* (RIM)
  - *Pass-thru Ports* (PTP) (allows to move up to 2 cartridges at a time)
- *Drive and Electronics Module* (DEM)

Addressing Slots and Drive Bays

When discussing the generic library topologies, the desire for a user friendly way to configure (virtual) systems was discussed. Following this perspective, a user would favor the granularity that is provided by the vendor's catalogue. In addition, a schema to map from slots to coordinates may be required for path planning. A similar mapping may be also required to manage which cartridge resides in which slot, as most library management software uses some kind of visualization. Tape cartridges are usually identified and addressed by an unique ID which encoded as a barcode which is stuck on the tape. Usually, this barcode also includes tape generation, e.g., LTO-7.

For slots the situation becomes more complicated as, HPSS for example, does not honor the way the SL8500 is enumerating tape slots and drive bays and uses a different mapping. Figure 4.8 was created to illustrate the different information that could be used for addressing slots in the SL8500 across multiple library complexes. In practice, not all these information need to be included. In most cases, an enumeration schema is restricted to the configuration and needs to be updated when the library is extended. This is indeed necessary also for the SL8500, though extensions into some directions do not require a reconfiguration. Figure 4.9 shows an official slot mapping onto a 2D representation of a complete SL8500 library module taken in the user manual. As can be seen slots are addressed by `(Complex, Library, Rail, Wall, Column, Row)`. In addition, the drive bays are addressed using the same column schema as the tape slots, while maintaining a separate row schema. This schema is attractive as it keeps calculating robot actions and tape receive times organized.

StorageTek SL8500 Robot Speed The average time to serve a tape is specified to take below 11 seconds on average. Due to the SL8500's centerline design, this value can be maintained regardless of the number of libraries that form a library complex. When modeling the SL8500 it is therefore assumed that traveling half a rail will also take 11 seconds. At a depth of about 6.5 meters with 5 expansions, this results in robot velocities of 0.6m/s not accounting for any easing. Easing would improve accuracy, as robots accelerate gradually.

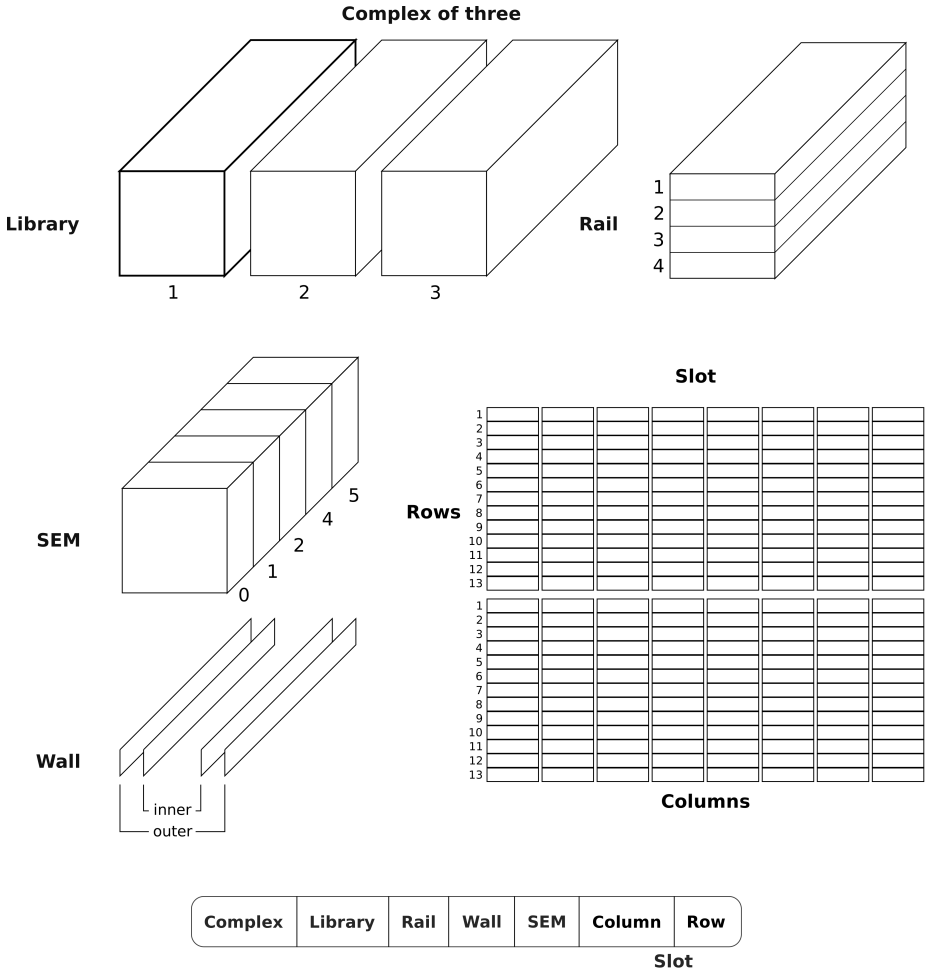


Figure 4.8: Library structure and possible divisions for a schema to address StorageTek SL8500 slots. From a costumers perspective a library and the number of expansion modules (SEMs) are considered. In contrast, from an operational perspective, SEMs do not exists: only the slot in terms of rows and columns for a given rail, wall and library are relevant.

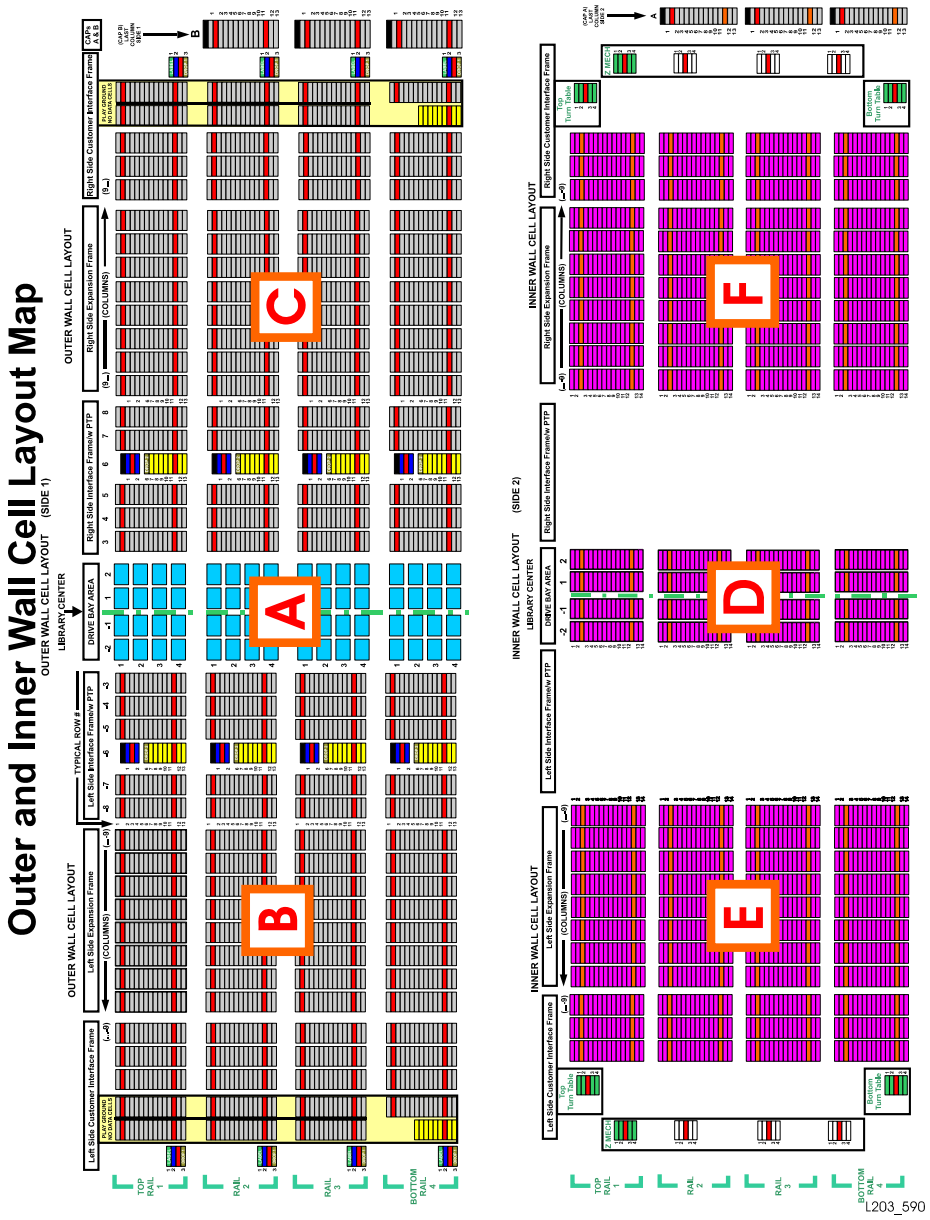


Figure 4.9: Internal representation of a StorageTek SL8500 (Sun, 2006).

4.6 Network Topology

Once a tape is mounted for reading in a tape drive, the data is transferred through the network to a I/O server . For simplicity and to allow to selectively increase the level of detail, a broad definition of what qualifies as a network is applied. In particular any bus within a computer as well as also any interconnection between computers that carries data, can be modeled as part of the network as long as it is reasonable to describe the performance in terms of bits or bytes that can be handled within a certain time frame. The components that are relevant for network topology in the context of tape systems have been already introduced in Section 4.2.2, though software components and the actual tape library will be usually excluded, which leaves: *clients*, *switches*, *I/O servers*, *caches*, and *tape drives*. The physical network topology is represented using a directed graph with vertexes for components and edges for the connections between the components. Two approaches to model and simulate network activity are discussed,

Flow-Based Model: Transmissions between hosts are realized by determining the maximum-flow from the sending to the receiving party. The approach may be lacking accuracy in some contexts and is optimistic. For the prototype implementation only the *Flow-Based Model* is implemented (see Chapter 5).

Packet-based Model: Simulating packets will be more accurate as it is closer to how data is transmitted in reality. The model is usually more expensive to compute, because every package has to be considered by the simulation.

4.6.1 Communication

Before the transmission models are discussed in more detail, we to investigate potential bottlenecks within the network that is deployed in support of a tape system. Both approaches can get very detailed, though especially for low-level communication, a flow-based model is not really appropriate anymore. For HSM in general, it could be interesting to even consider circuitry within a chip. Figure 4.10 illustrates transfer rates as they can be expected between different components within a tape system. While data has to pass RAM, it seems fair to model using only the three most limiting factors, thus ignoring the effect that may emerge from further up in the cache hierarchy. The limiting bandwidths are observed when performing tape or disk I/O.

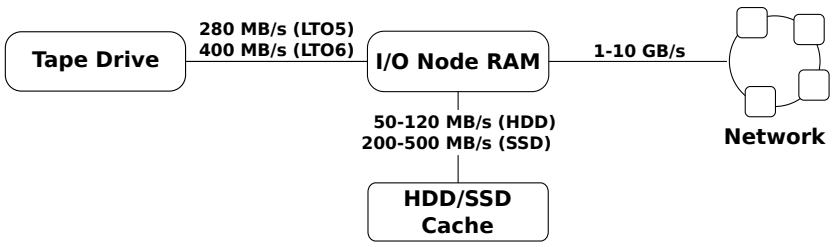
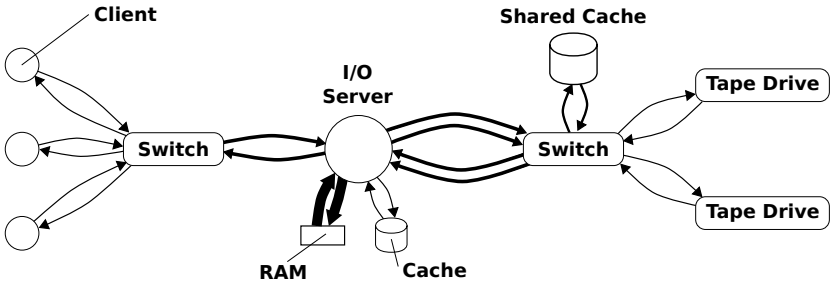


Figure 4.10: Typical transfer rates (uncompressed) between different components within a tape system.

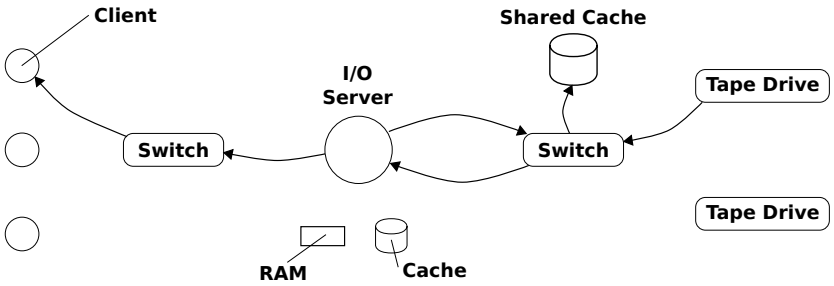
4.6.2 Flow-based Model

A straight forward approach to model network bottlenecks between multiple hosts in a network makes use of *Maximum-Flow/Minimal-Cut Algorithms*. Hosts are represented by vertexes and bandwidths by weighted edges. Algorithms for directed and undirected graphs are well researched. The configuration is simpler with undirected graphs as less edges need to be created. Directed graphs allow to better differentiate between upstream and downstream connections. In both cases, we can account for multiple network adapters by adding multiple edges between two nodes. The limiting network connections are bidirectional and full-duplex by nature, meaning communication is possible in both directions and at the same time respectively. For this reason, a directed graph representation seems favorable. This representation can be exploited to account for unequal read and write performance of storage technologies. The downside to this approach is that it cannot represent the true dynamics within computer networks. In general it can be assumed to be overly optimistic.

In a flow-based model the communication between to hosts can be realized by calculating the maximum-flow between the sending and the receiving party. If a flow exists, the networks capacity is reduced for the time of transmission. As additional capacity becomes available in the network the maximum-flow is be recalculated. The concept is also illustrated in Figure 4.11. It does not matter which algorithm is used, but the *Push-Relabel Algorithm* with time complexity of $\mathcal{O}(V^3)$ appears to produce solutions that include fewer edges.



(a) Network topology with the edge width encoding for current capacity.



(b) A combined flow as it may occur for a read request limited by a tape drives upstream. A first flow may be obtained for data flowing from tape drive to clients. A second flow originates from the I/O server to write a copy to the shared cache.

Figure 4.11: Illustration of topology and network activity for a flow-based model.

	Layer	Protocol data unit	Function / Example
Host layers	7 Application	Data	CSS, HTML, Javascript
	6 Presentation		
	5 Session		FTP, NFS, HTTP, HTTPS, RPC, SMTP
	4 Transport	Segment (TCP) / Datagram (UDP)	TCP, UDP, SSH, NetBIOS
Media layers	3 Network	Packet	IPv4, IPv6, ICMP
	2 Data link	Frame	IEEE 802.2, L2TP, MAC, PPP
	1 Physical	Bit	Ethernet, SCSI, USB, ISDN, DSL

Figure 4.12: ISO/OSI model for data communication and protocol examples.

4.6.3 Packet-based Model

As real systems transfer data in streams of packets, a more accurate way to model traffic that captures the dynamics of computer networks is based on packages. When allowing packages to be the size of a single bit, using this representation any digital communication can be modeled. In principle a packet-based network model acts according to the Open Systems Interconnection (OSI) (Day and Zimmermann, 1983) model, a seven layer model that characterizes communications independently of the underlying technologies (see Figure 4.12). It is not strictly necessary to implement all involved layers if errors that can occur during in lower level communication are acceptable to be ignored. In terms of the OSI model with respect to tape systems, it appears sufficient to focus on the network layer. How a package, or more abstract a protocol data unit (PDU), arrives at the target destination depends on the protocol. Multiple protocols and transmission types can be mixed. PDUs are stored in special buffers that are maintained for every protocol. Usually no paths need to be explicitly calculated, instead for example when handling the network layer, gateways are used to pass packets until they reach their destination or are dropped on their way. A packet-transmission model is more expensive to compute, because every PDU has to be considered by the simulation. Figure 4.13 illustrates how the network might look like for a tape system. Note how for example switches will only have buffers for the protocols that are relevant for routing, e.g., IP traffic.

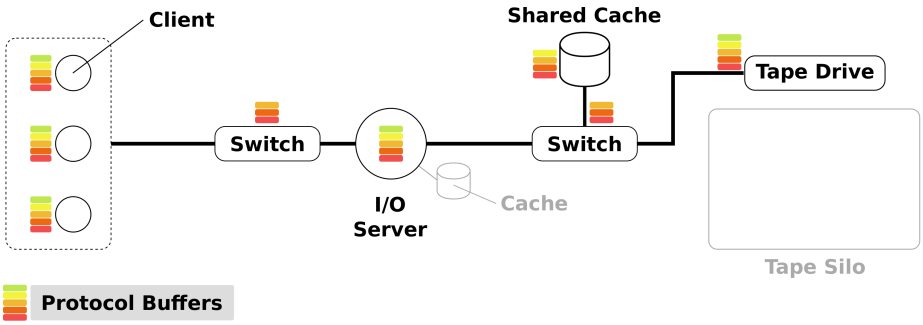


Figure 4.13: The network communication schema for a tape system, with different protocols depending on the desired simulation accuracy.

4.7 Modelling Software Components

In this section we thrive to collect abstractions of relevant software components that would also be found in real deployments. As such there are mostly algorithms for load balancing, scheduling for resource management as well as for file- and tape system management. Quality of service is sometimes artificially limited by quota systems or provide multiple service levels. In addition, some requests may enjoy priority status and should be handled privileged. Finally, the robots in the libraries need to be orchestrated, so that they provide tape drives with tapes as fast as possible. Software components that are essential for a tape system are the following:

Filesystem: Modern tape systems are exposed to users transparently and appear to be just like any other filesystem. Therefore, special purpose filesystems that support mapping files to tapes and implement special cache coherency mechanisms are deployed. The implementation uses a sparse file management component to provide the most important functionality.

Tape and Slot Management: Somehow a system has to keep track of the tapes in the library and which slots are still available to place a tape. In addition routines that rearrange tapes in a predictive way and collaborative use of tape drives (RAIT) for higher data security and faster access can be considered.

I/O Scheduling / Drive Scheduling: Among the most critical components are I/O and drive scheduling algorithms. Only naive approaches are considered in this context as this is a research field of its own.

Robot Scheduling: Before a drive can read a tape, a robot has to mount the tape. Controlling the robots and prevent them from blocking each other is taken care of by robot scheduling and path finding (also refer to Section 4.5).

4.7.1 Stack

As software architectures are often hard to overlook, it is helpful to visualize how the different software components are related to each other. Figure 4.14 arranges common software as it may be found in a tape system as a stack. Decoupling components in such a way can give away optimization opportunities. On the other, hand monolithic solutions are extremely hard to maintain.

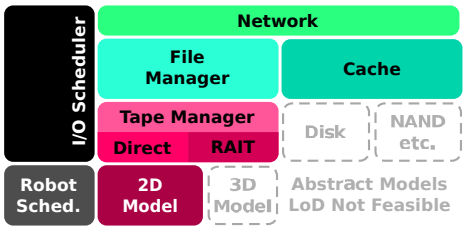


Figure 4.14: A stacked overview of software components for tape system. The dashed components may be feasible only when analyzing of higher-level HSM.

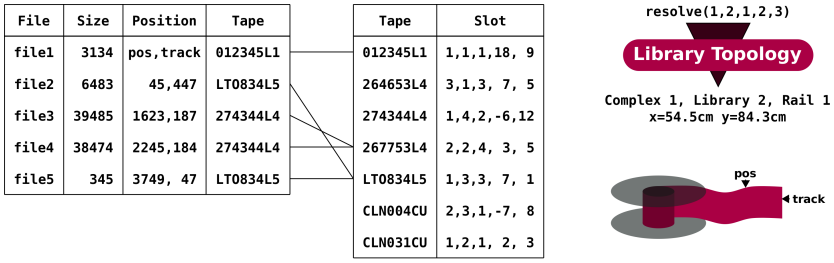


Figure 4.15: File and tape registries with metadata hidden for simplicity.

4.7.2 File, Tape and Cache Management

The software components managing the filesystem and the tapes within the library are critical to a tape system, yet they are fairly straight forward from a modeling perspective. Keeping track of a few hundred thousand tapes and map them to slots in a library is relatively easy for modern systems and most database management system are capable to do so. Challenges emerge for the large quantities of files and associated metadata. These concerns , however, are not relevant for many scenarios that the simulation is intended for. Figure 4.15 illustrates two tables including the only the very basic information required for tape management, regular metadata is omitted in the illustration but is of course part of the schema as well. Slots addresses can be resolved by the library topology as will be required by the robot scheduling. In addition, it can be useful to store the files position on the tape in the registry as it allows for more intelligent strategies when scheduling tape I/O.

Cache Coherence and Policy Algorithms: Modern tape systems heavily rely on caches to speed up operations. Software components implementing cache coherence protocols and cache policy algorithms are therefore required. They should also be easy to exchange by another implementation to make experiments easier. For tape systems, caches are usually acting on files which may have some unique dynamics but many caching algorithms are universal. It is notable that there is in theory an optimal caching algorithm sometimes called *Baledy's Algorithm* (Belady, 1966): The most efficient strategy is to discard the information that will not be needed for the longest time in the future. Unfortunately, the practical value to this insight is fairly limited as workloads are generally considerd unpredictable. But it becomes very clear that the heart of any cache algorithm is a replacement strategy. Cache performance is commonly evaluated by considering two performance characteristics:

- *Lifetime* – How long a item resides in cache.
- *Hit rate* – How often a desired item is actually found in cache.

Countless replacement strategies for caches exist, adequate for the requirements of the simulation prototype are: *Least Recently Used* (LRU), *Most Recently Used* (MRU), *Random Replacement* (RR), *Least-Frequently Used* (LFU). In addition, there are combined approaches such as the *Adaptive Replacement Cache* (ARC), which mixes LRU and LFU. IBM holds a patent on ARC.

4.7.3 Concurrency and Request Bundling

Tape libraries can serve many requests concurrently when multiple tape drives are installed. Because requests to tape systems usually take some time, it is advisable to bundle requests to the same objects. Figure 4.16 illustrates how a sequence of outstanding writes can be made more efficient by bundling multiple request by tape and by reordering the requests to match the order they appear on tape. Some limitations apply when tracks are written shingled on tape layout, as in this case it may not be possible to overwrite a existing file without rewriting other parts of the tape as well. Another problem can occur when multiple clients are concurrently reading and writing the same file. POSIX semantics for example require individual reads and writes to be atomic. For non-local filesystems especially when dealing with large reads and writes, these guarantees are hard meet as multiple copies of different versions may need to be kept until overlapping requests are completed. In general more relaxed semantics allow for more optimization opportunities. The problem is usually formalized as weak versus strong ordering. Strong ordering guarantees that changes get visible in the order that they occur. Weak ordering architectures use more relaxed semantics and may reorder operations to archieve better system utilization. If strong ordering is a requirement, theory also used in database design and optimization can be applied. A good trade-off in cost and usage is given for the class of conflict serializeable transactions. Classes with broader guarantees often become NP-Hard to decide. Operations O_i, $O_j \in \{read, write\}$ are considered to be in *conflict* if and only if there exists some data D accessed by both O_i and O_j, and at least one of these operations was a write access. Four cases need to be considered:

1. $O_i = read(D)$, $O_j = read(D)$. Maybe handled concurrently.
2. $O_i = read(D)$, $O_j = write(D)$. Can not be handled concurrently.
3. $O_i = write(D)$, $O_j = read(D)$. Can not be handled concurrently.
4. $O_i = write(D)$, $O_j = write(D)$. Can not be handled concurrently.

The consequence of a conflict is always that, operations that conflict need to be handled in the order they arrived. There exists a temporal order. For non conflicting types interchanging the order will yield the same result. For the simulation of tape archives it is expected that relaxed semantics will be sufficient and strong ordering is not considered in the simulation. For higher level HSM simulation they may become relevant.

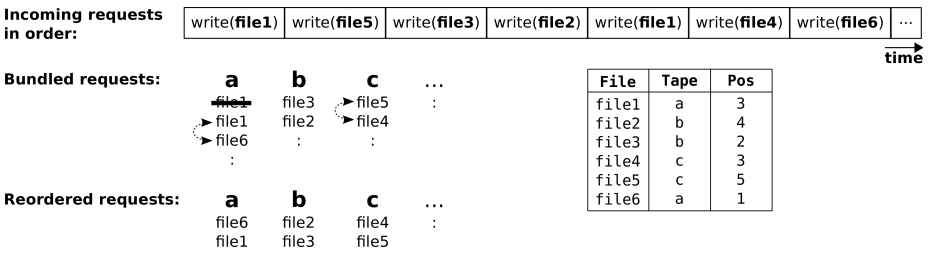


Figure 4.16: Multiple requests are bundled and reordered to optimize tape I/O.

4.8 Drive Allocation and Robot Scheduling

Drive allocation and robot scheduling are intimately related because a tape can only be read or written when it was mounted first by a robot. Thus, scheduling the two independently leaves unused optimization potential. But finding a scheduling strategy that optimizes for drive utilization and robot utilization is complex. Ultimately, the situation is worsened as scheduling strategies that work well for one type of library system can perform miserably on another. Factors contributing to the complexity of deciding for a allocation include:

- Configuration of library complexes
- Internal library structure and partitions
- Distribution of tapes and drives
- Number of robots and their carrying capacity
- Heterogeneity of the system
- Redundant Arrays of Inexpensive Tapes (RAIT)

In this section we will only briefly look at a procedure that allows to reasonably approximate the behavior within a SL8500 library. The simplest approach is to rely on the specification of average tape receive time, and apply a 11 seconds penalty to any request. As we thrive for a more accurate model that allows to evaluate tape placement strategies a more sophisticated approach is needed. Figure 4.17 shows a configuration with 8 robots and the two elevators. It illustrates further the locality of the inner and outer walls as they appear from a robot perspective. Robots that are accommodated by the same rail can not pass each other. Though in case of failures, the system is designed so that one robot can push away another robot into a maintenance area to continue operation.

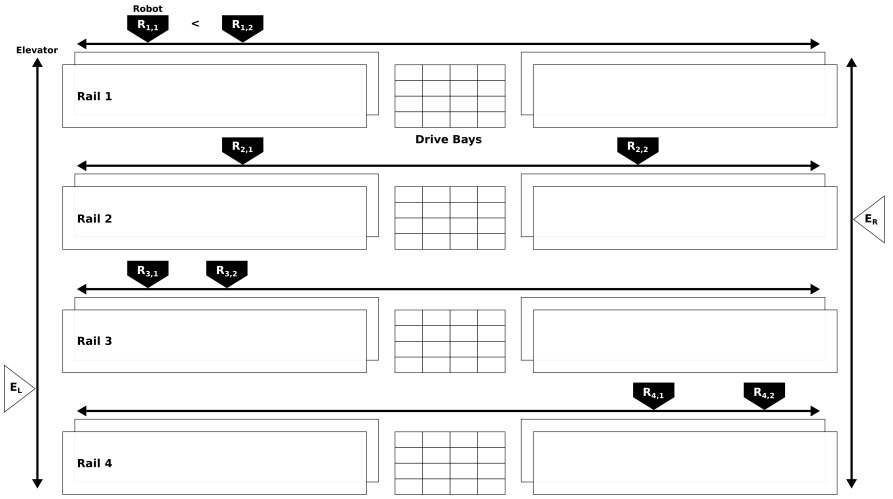


Figure 4.17: Most significant degrees of freedom for robots in a SL8500 library module and how these relate to the spatial distribution of tapes and drives.

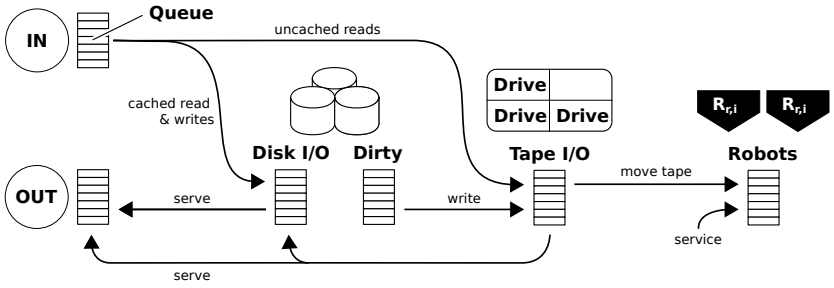


Figure 4.18: A network of queues to process requests to a tape library.

First In First Out (FIFO)

A straight forward strategy to handle incoming requests is to process them in the order they arrive. For tape systems this is somewhat unsatisfying as it passes up a chance of optimization. Strictly applying FIFO may disdain many of the benefits introduced through the use of caches. In particular caches allow us to prioritize read requests when scheduling tape I/O because by using sufficiently large caches we can delay writing on tape until the system becomes idle.

Chained Request Queues

To make the different resources in a tape system more manageable is by keeping multiple request queues. Each request queue is allowed to implement a prioritization schema as appears appropriate for the type of resource. Figure 4.18 illustrates the concept of multiple queues: Incoming requests are analysed and distributed depending on their type and if the requested files already resides on disk. Writes and cached reads can be served immediately but uncached reads will require to perform higher latency tape I/O. Items in a cache that are not made coherent yet are commonly referred to as dirty. As dirty files accumulate, writing tape I/O has to be enqueued to keep the cache operational. The tape I/O queue may prioritize reads and requests tapes to be served. An approach similar to this one is used by the simulation.

Sliding-Window Optimization

Picking the best schedule according some criteria is possible by comparing all possible schedules and pick the one that completes all tasks within, e.g., the shortest time. For systems with high degrees of freedom, such as robot libraries, this approach quickly becomes unfeasible as the system is overwhelmed by an exploding number of combinations to compare. Yet as tape systems are operating on relatively large latencies in the range of tens of seconds a heuristic or a reduced problem can potentially increase the general library throughput. Assuming a number of tasks that need to be completed, we can reduce the problem by only considering a limited number of requests e.g., a sliding window that compares different schedules for the next 10 outstanding requests. Another approach to reduce the problem is to limit the amount of time the system is allowed to look for optimized schedules, and stick with the best that was found in this time. In both case heuristics become important that, e.g., allow to discard combinations that are unlikely to yield good results.

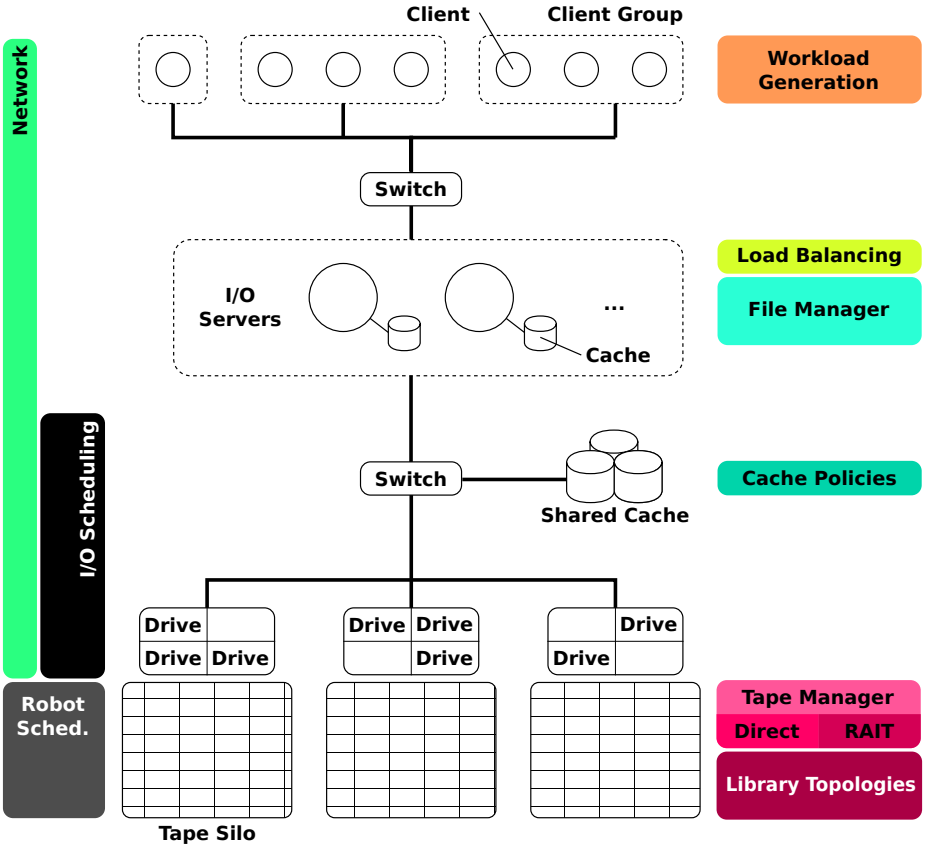


Figure 4.19: Model overview for the operation of a tape library simulation. Colorful components represent software. The dashed lines are used to hide the complexity of the network where appropriate.

4.9 Summary of the Model

A complete overview Figure 4.19 combines the most important hardware and software components that are necessary for the operation of a simulated tape library system. From top to bottom: There is clients which stress the system by sending requests through a *network*. They are represented using workload generators which are covered in more detail in Section 5.4. Requests are then handled by the I/O servers and constantly interact with various *filesystem* related components such as the *caches* but also the *tape management*. Writing to disk and tape requires special *I/O scheduling* strategies. But before it is possible to write on or read from a tape it must be mounted to a tape drive, which requires to look up where the tape is and then send a *robot* to get it. To do so we need to have a model of the *library topoloy* which allows us to plan a path and also how long this takes. As the systems is used for a while, files will accumulate in the cache which requires to enforce *cache policies* to decide which files should be replaced.

In many occasions the models take shortcuts as the scope of the thesis is limited and mainly aims to capture the fundamental characteristics required to

explore the performance and conduct experiments on automated tape library systems. This section is dedicated to have an overview of the limitations of the model. It also serves as a starting point for improvements and future work. In most cases reasoning to model in one way or another is provided in the respective section which covers the "sub-model" in detail. Noteable simplefications are as follows:

- Hardware – many smaller components (e.g., power supply, cleaning cartridges) are not explicitly modeled
- Workloads and Requests – Countless protocols with different operations interact with tape systems an many different levels, but only `read` and `write` are modeled
- Network – The *Flow-based model* assumes optimal bandwidth exploitation
- Library – Service interruptions, e.g., cleaning cycles are covered only briefly
- Software – For the beginning only very simple algorithms are used, which may diverge considerately from the state of the art
- Global View – Gathering system state information in most situations comes for free, but obtaining a global view is very hard on distributed systems

Summary

Modeling a tape library in a hierarchical storage system is a complicated process that touches many topics which are research fields by their own right. The objective of the model to serve as a foundation for I/O experiments for tape systems and hierarchical storage system is curse and blessing at the same time. On the one hand, it allows to ignore many facets that can be considered implementation details, on the other hand, it requires the invention of models for behavior which otherwise simply emerges from a physical system. Workload on the system can be expressed almost exclusively by looking at `reads` *and* `writes`*. Hardware components can fail but establishing a authoritative models require to collect reliability statistics. It turns out finding a generic abstraction to describe the dynamics of tape library complexes is surprisingly hard as the many details that may make a difference accumulate for all the library systems. Besides describing the topology of the tape library we also have to use a topological description of the network to model communication between the different hardware components. Finally all physical systems are governed by software, which should be to some extend modular to allow replacement with a experimental alternative.*

Chapter 5

Implementation

This chapter covers the technical aspects and the implementation of the tape simulation. Section 5.1 motivates the choice of the programming language. Section 5.2 presents the general architecture including a class hierarchy. Section 5.3 gives brief introduction to discrete event simulation and implementation specific cues. Section 5.4 discusses how to automatically run workloads on the virtual tape library. Section 5.5 presents features to make experiments more convienient.

5.1 Programming Language

At the beginning of any software project, one of the first decisions is the choice of programming language. Given the requirements and the nature of the problems as discussed in Chapter 4, an object oriented language seems to provide the most intuitive concepts, allowing to wrap physical and logical components into separated classes. Which will ultimately be easier to maintain. From a wealth of object oriented languages, Python was chosen for a number of reasons. Python is a relatively high level language and allows for convenient inspection of the system state as it is an interpreted language and offers reflection. Secondly it features a extensive standard library and makes many third party libraries easily available, e.g., Numpy, Scipy (**?**). It is popular for rapid prototyping situations and reduces the areas where we might get entangled in low level problems as they are mostly hidden. That is not to say there are legitimate concerns with Python, on the one hand it is comparably slow. Secondly, the idea is that at some point software components used for scheduling and actuate the tape archive maybe used to drive an actual tape library and not just a simulation. Also support for multi threading could be better. But as there exist python bindings allowing to use C/C++, this concerns can be neglected, as moving over is possible when performance becomes a problem. Generally it appears, maintainable code is more important during prototyping than efficiency.

5.2 Architecture

The tape/HSM simulator was build with modularity in mind. Besides the software development benefits such maintenance and reuseability, this also provides a more intuitive access to the simulation of a complex system. While it was originally planned to use a framework for discrete event simulation, it turned out that the available frameworks were lacking in the way times could be represented, which is why a simple simulation kernel that was compatible with Python's `datetime` to represent times was developed instead.

5.2.1 Classes for the Key Components

Analog to the requirements determined in Chapter 4 hardware and logical components are modeled and would inherit from a generalised component that would provide the basic interface that is used by the simulation kernel. For common component types abstractions have been created. Also refer to Figure 5.4 for a UML class diagram to see how the different components are related.

Simulation Kernel A relatively simple kernel that implements a discrete event simulation. As real world systems are simulated and most events in practical systems are timestamped instead of an abstract model time.

Components Per convention every component that should be considered by the simulation needs to register with a simulation and implement the `next_action()` method.

Network Topology The network is represented by a directed graph where a weight of an edge represents the capacity of the network interconnect. As requests are processed, parts of the network are allocated and freed again when the transfer is over.

Network Components When registered to a Network Topology, these components are considered when sending data from one host to another.

Library Topology The idea is to generalize the library topology so that it would be easy for generic scheduling algorithms to find and calculate the best schedule to receive a sequence of tapes. The network topology describes the paths a tape can travel.

Library Components When registered to a Library Topology, the component will be considered when calculating the path to reposition or mount a tape in the library.

Software Components The abstraction is not strictly necessary at this point but is made to separate hardware emulation and managing software as good as possible. Hopefully this allows to easier port components that have proven useful to a production system.

Convenience and Reporting To simplify the collection of data and to allow to potentially resume a simulation run a centralized mechanism to store data that is related to one simulation is available with the *PersistencyManager*. As a second mechanism it is possible for components to register CSV reports, and append rows and data points as the simulation progresses. The CSV files are then stored together with the other files related to the simulation run. Besides data collection, routines to visualize the network utilization and device states are available for easier inspection and debugging.

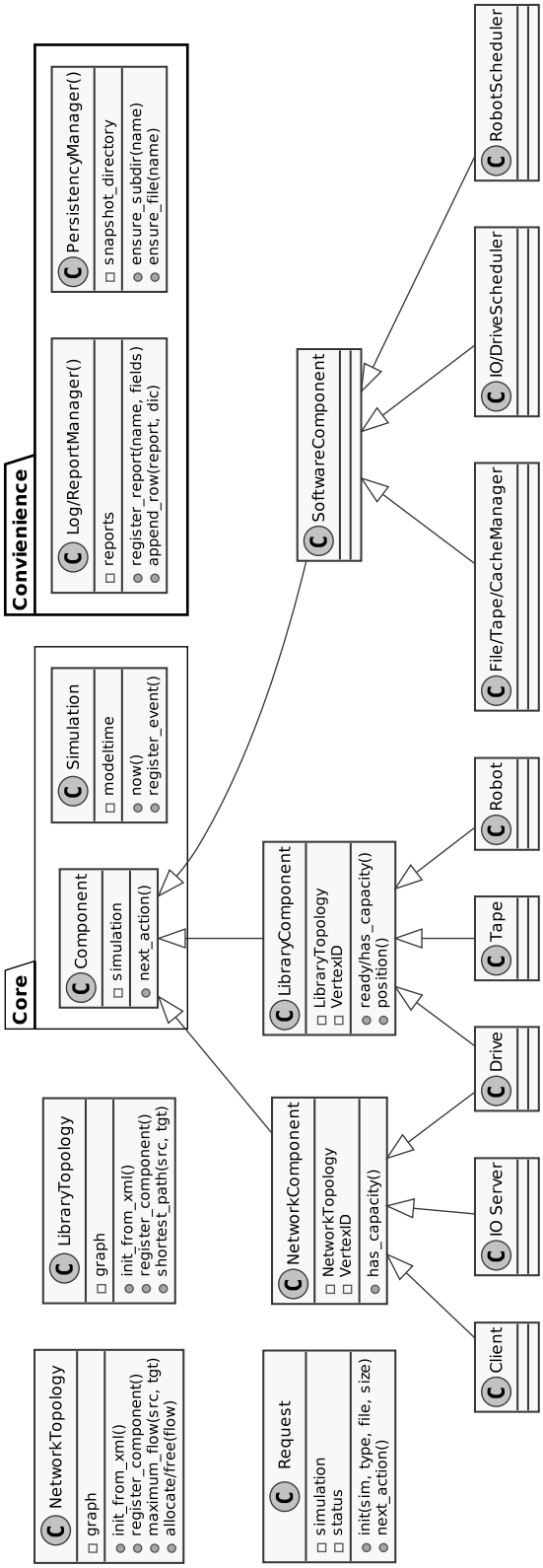


Figure 5.1: UML Class Diagram

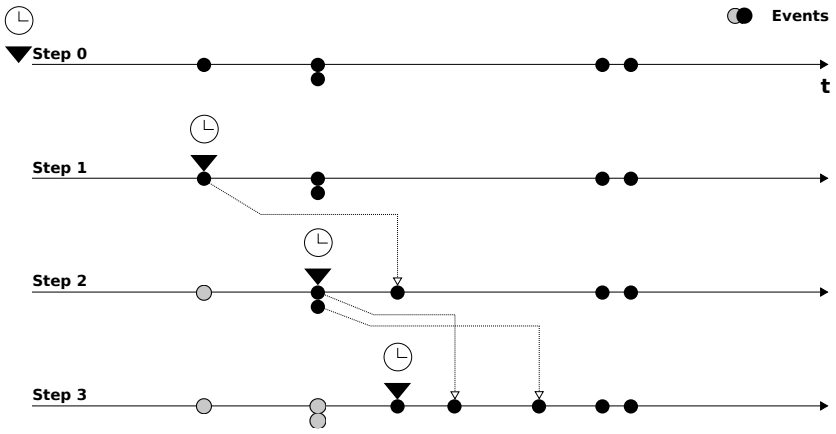


Figure 5.2: Discrete Event Simulation

5.3 Discrete Event Simulator

To model complex systems a common tool is *discrete event simulation* (DES) already introduced in Section 3.4. Figure 5.2 illustrates how DES was implemented in the simulation prototype for this thesis. Events are ordered by their time of occurrence, and always the first event becomes the model time. When a event is processed it may submit new events to the simulation which are then considered when determining the next model time. It is possible for events to occur at the exact time in which case the sorting algorithm dictates the order in which they are processed. Python by default uses a stable sorting algorithm, which makes the simulation honor the order in which requests are submitted. The simulation uses Python `datetime` and `timedelta` objects to represent time, which allow microsecond precession. For the simulation with tape systems it appears that rounding to second precision yields good approximations at a fraction of the runtime, also see Section 6.4.

The simulation terminates when no more events are in the event queue. Some other termination conditions have proven useful during conducting experiments and for testing and development:

- *on system stall* (default)
- *after number of iterations*
- *after passing a specific model time.*

5.4 Workload Generation

To stress the virtual tape library, a request object can be instantiated and submitted to the simulation. In addition to explicit submission it possible to register event providers to the simulation. As the simulation proceeds event providers are polled for future events until they indicate they have been drained. A workload provider could be used to create requests on the fly according to a script, a probability distribution or a trace file.

Xferlog Provider For the evaluation of the simulator with real world workloads, tracefiles using the *xferlog format* as provided by the FTP server daemon (proftpd) were used. The trace files are to some extend anonymized which has the potential to introduce unwanted collisions though an analysis of requests suggest that the effect is negligible. The Xferlog provider discards entries it can not parse and accepts requests types based on a white list. Four request types are accepted and mapped to `read` and `write` as follows:

- read – `PST_Cmd`, `STOR_Cmd`
- write – `PRTR_Cmd`, `RETR_Cmd`

On startup of the simulation, the trace is analysed to populate the filesystem and the tape library. The Xferlog Provider automatically registers unknown hosts as network components to a predefined node in the network topology. Connections are automatically made bidirectional and all hosts are assumed to have the same bandwidth. For very large traces the simulation may run out of memory as no hosts are removed. In practice this has never been a problem, as the memory profile usually remains relatively stable (see Section 6.4).

Assumptions for Tracefiles

To ensure simulations are meaningful when using workload traces, a number of assumptions were made, mostly because a trace alone does not provide all the necessary information.

- Assume only serviceable reads: Incoming reads with not entries in the simulated filesystem are created and then served despite they may have been dropped in reality.
- Ignore hardware failures and service. Components do not fail, though a event generator that disables components is straight forward to implement.

5.5 Configuration of a Simulation

5.5.1 Experiments

To make the setup of experiments more convenient a number of features were implemented. In particular command line arguments to quickly change the configuration were used, though a comprehensive method for components to register additional options is not provided yet. Topologies can be saved and loaded from XML files. The workload generators from Section 5.4 and the reporting tools covered in Section 5.6 are also important features to make the life of experimenters easier.

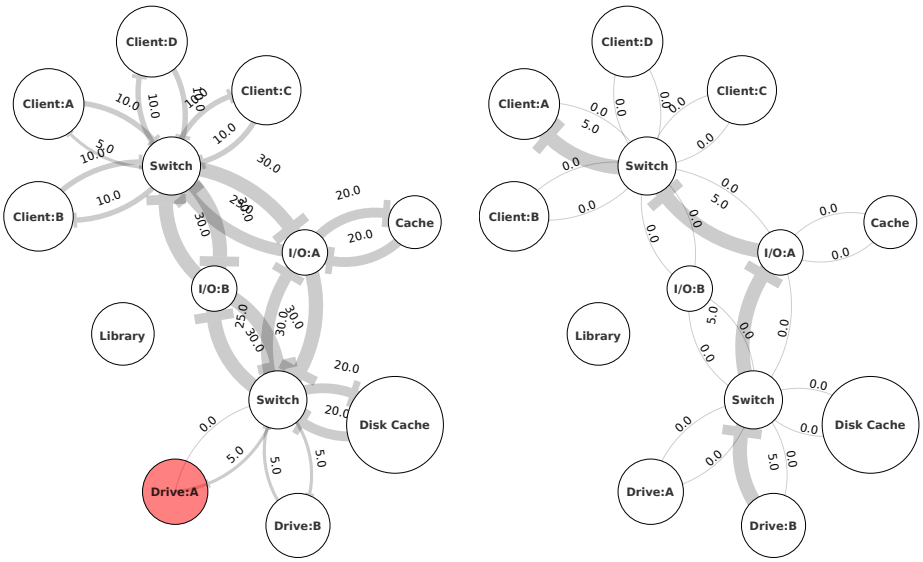
5.5.2 Graph based Network Topology from XML

Every data center uses a different configuration, to account for this and also to allow easily switching a configuration, the topologies currently implemented offer import functions to initialize a topology from an XML file. As described in Section 4.6.2, the network represented by a directed graph that connects physical components. Every component provides methods to query for the components capacity. The default for network components is a capacity of one which can not be shared. Parts of the network capacity can be allocated and freed, e.g., as a path from source to target. Figure 5.3 demonstrates the concept on a very simple configuration with already one tape drive busy (a); (b) shows the result of a available flow from a free drive to a client. This approach was chosen to serve as light weight alternative to a paket-based simulation of the network.

It was tried to avoid third party dependencies. For the graph representation it was decided to use a library. Network and library topologies are based on a graph representation, and *graph-tool* (Peixoto, 2014) bundles a collection of useful tools in graph theory. Algorithms are implemented in C++ as provided by Boost are wrapped so they can be called from within Python. As a note of caution: the library wrapping C++ features relatively "unpythonic" access to data structures. Graph-tool cannot be installed from `pip` but needs to be compiled from source which may take some time as the library makes heavy use of template code. Graph-tool can be compiled with OpenMP to exploit parallel algorithms.

5.5.3 Command-line Options

Command-line interfaces have proven to be extremely helpful to ensure reproducibility and to enable simple means for automation. Using shell scripts it is then possible to start simulation runs varying different parameters, in a similar fashion a optimization engine may start multiple runs and decide based on the results which configurations to try next. The parameter requirements for the simulation of tape drives have not been formalized but were added as they became useful during development. This way arguments to control the trace provider using `--tracefile FILE` and `--limit NUM`. Arguments to switch the topology configuration are available including `--networktopology FILE` and `--librarytopology FILE` and `--drives NUM` to specifiy the number of tape drives. To manually step through the simulation one can set `--confirmstep`. Most class constructors have additional parameters which are not exposed yet.



(a) The current state of the system. `Drive:A` is busy as indicated by it's color.

(b) A flow from a free `Drive:B` to `Client:A` for the state as depicted in (a).

Figure 5.3

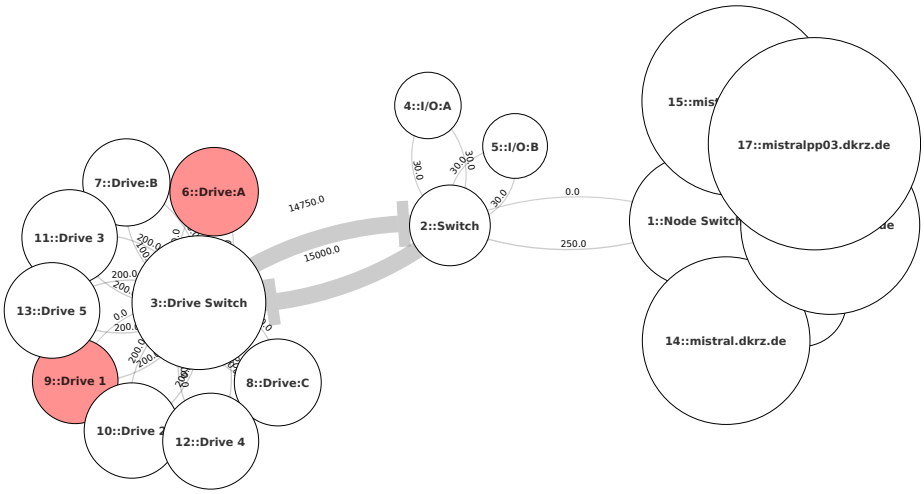


Figure 5.4: Network graph visualisation with two busy drives and the remaining bandwidths as edge weights. Note the link from `1::Node Switch` which is dimensioned so the network would be quickly saturated.

5.6 Reporting and Debugging

The simulation was build with certain reporting capabilities in mind from the beginning. For every simulation run a dedicated and timestamped directory is created which can be used by other modules to store data associated with the current run. This way it is easy to inspect and analyse the data during simulation and post-processing. The consolidated structure of simulation results makes automated report generation much easier.

- Request data
  - `requests.csv` for request summaries (e.g. throughput, duration, size, status/errors)
  - `stages.csv`
  - `wait-times.csv`
  - Detailed request histories including bandwidth changes (optional)
- Simulation process log when enabled (Default: `stdout`)
- Simulation state in limited detail (dumped at the end of the simulation)
  - Filesystem state
  - Tape system state (Tapes and Slots)
  - Global cache state
- HSM/Tape System Configuration
  - Network Topology as XML
  - Library Topology (pickle/XML)

Post-Processing

Performance reports are not generated automatically after a simulation run completes, but a number of R and post-processing scripts to generate plots from snapshot directories or for trace files are available. In particular this includes:

- Timeline plots for *requests*, *stage-counts* and *wait-times*, *trace-activity* and *memory consumption*
- Density, histogram and boxplots for performance metrics like *throughput*, *duration* and *wait-times*

Summary

The programming language has implications on the execution of any software project. For a prototype rapid development is important which is why the simulator is implemented in Python. Python, being a object oriented language encourages the use of classes, which the simulator uses to provide abstract components that are related to tape systems. The simulation itself is based on the principles of discrete event simulation, and a simple simulation core was developed for the thesis. To conduct experiments a wide range of support tools had to be developed to allow for workload generation, library configuration and visualisation and debugging.

Chapter 6

Evalution

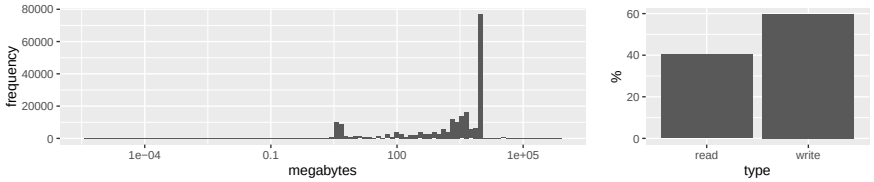
This chapter validates whether the models and the simulation prototype meet to the goals of the thesis. In Section 6.1 we take a look at a real world workload as it was observed on the storage archive of the DKRZ. Section 6.2 compares the simulated library performance to plots with the monitoring of the real system. Section 6.3 looks at how varying the number of drives effects the performance of the simulated system. In Section 6.4 we briefly look runtime and memory usage of simulation runs to see if running lots of experiments can be feasible.

6.1 A month of PFTP activity

For the evaluation of the simulator, the DKRZ generously provided workload traces of FTP traffic that cover a periods of roughly a month as well as matching snapshots of of the information captured monitoring during that time. The monitoring plots covers a period of about 3 weeks. This section will give a quick overview on the workload and discuss in how far it is suitable for the verification of the virtual tape system. Notable but not problematic appear the following:

- PFTP traffic accounts for the majority of the workload on the system, but other services to browse and access the data are also available e.g., through CERA.
- No files are deleted from the climate database. This may be reasonable as users are charged for using the archive and won't submit data that is not meant for long-term archival.
- The trace data was anonymized by stripping the filenames, which may lead to collision in the virtual systems on files which are not the same file in the real system.
- The trace includes a period of scheduled downtime. This takes away from simulations thriving to improve normal operations, but maybe more meaningful for verification as the recovery from a downtime can be simulated.
- I/O in the trace touches only about 0.5 PB of data, but for the evaluation of of caching strategies for a 5PB disk cache no replacement of cached files need to be performed.

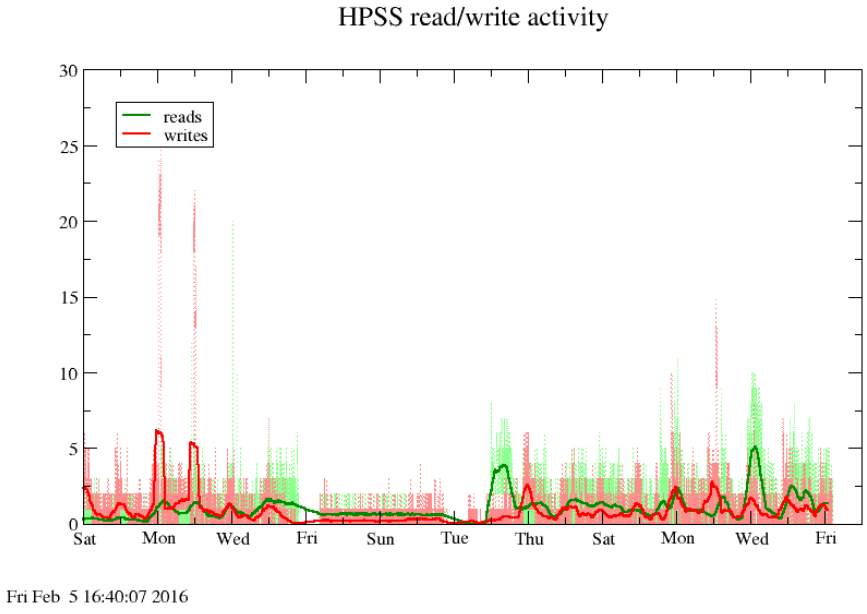
Figure 6.1 plots the frequency of different file sizes in megabytes as they occur in the trace. It is notable that most files are about 10 GB in size, which is consistent with the recommendation communicated to users of the DKRZ. Figure 6.2 counts the number of read and write requests in the trace in comparison to the DKRZ's monitoring.



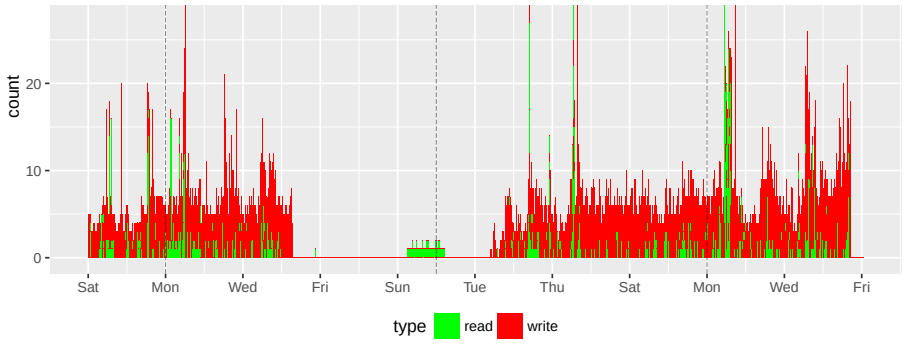
(a) Frequency of different request sizes. With a large peak for requests with a size of 10 GB.

(b) Percentage of reads/writes.

Figure 6.1: Composition of the workload in terms of request size and type.



(a) Reads and writes as they are observed by DKRZ's monitoring.



(b) Read/Writes as they occur in the trace that is fed to the simulation.

Figure 6.2: Observed request types over time for a period of 3 weeks.

6.2 Model Verification

In order to stress the virtual system, the workload trace discussed in Section 6.1 was used and similer graphs as observed in the DKRZ's monitoring system were created. A special provider to support the *xferlog format* had to be developed, as had to be a strategy to approximate the system state of the library. A number of assumptions and simplifications had to be made, which limit the accuracy of the simulation:

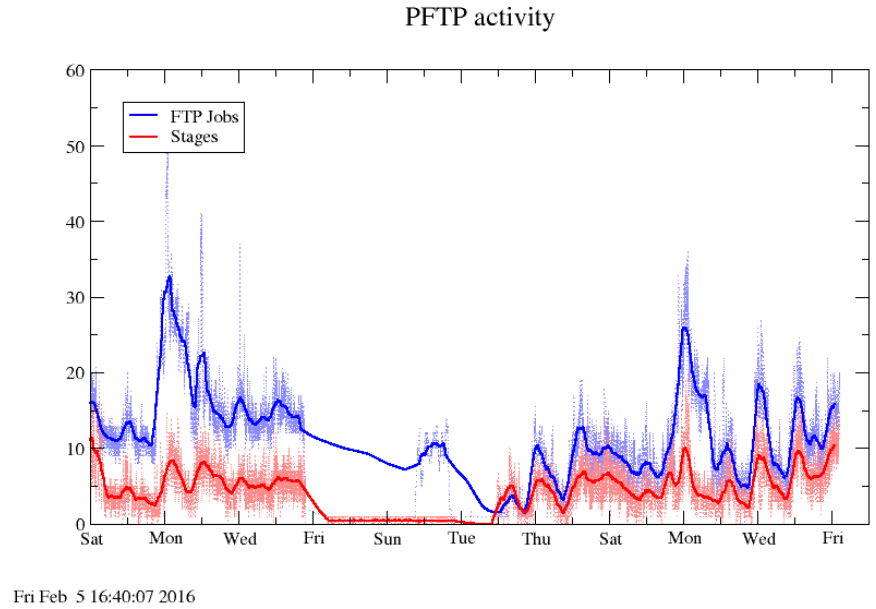
- As a distribution of files, tapes and tape types could not be obtained a random distribution is generated when starting a simulation. This is a multi-step process, and requires to randomly pick a tape and the tape generation followed by randomly choosing a free slot if a new tape was created.
- Files which are read the first time they occur in the trace, are created in the file system, as it is assumed that they exist. Thus dropping requests to non-existing files does not occur.
- Hardware-side compression was ignored completely for the lack of a sensible heuristic to map compression rates to files.
- The trace does not provide information on where the files reside in the tape system. For the simulation the virtual tape systems files, tapes and slots are populated randomly. Files which are read only are ensured to exists.
- The type and generation of 65 tape drives is considered, but the exact placement in the library is not. Considering construction of the SL8500 the effect should be negligible especially as already the tape generations do not reflect the original system state.
- Local caches within the I/O nodes are not taken into account.

6.2.1 Comparison to DKRZ Monitoring

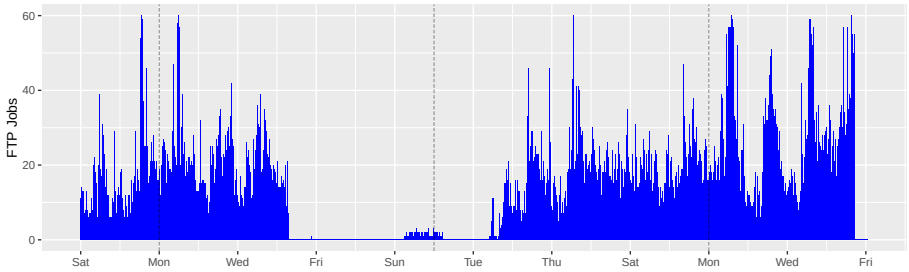
The monitoring system collects multiple metrics concerned with the library utilization and performance the system. By collecting similar metrics using the simulation and comparing the virtual performance to the monitoring data it is possible to verify how good the approximation of the simulation is. Two particular interesting monitoring metrics for the purpose are the cache stages and the wait-times for requests.

Verification of Staging Behavior

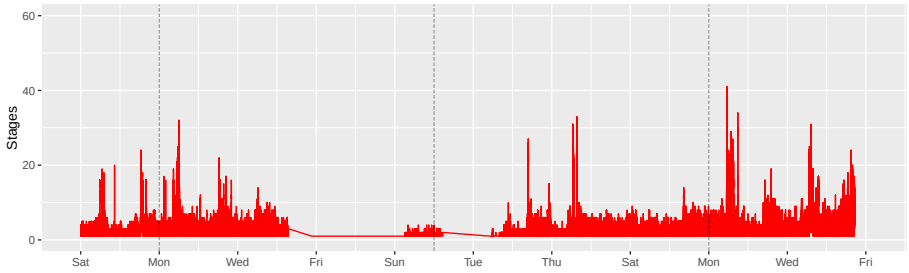
Figure 6.3 combines the number of FTP requests and the number of stages in one plot. It therefore can be used to visualize backlogs should the system not be able to keep up with the workload. While not identical, the simulation behavior clearly matches the behavior observed by DKRZ's monitoring. The simulation stages files slightly faster then the real system. This is consistent with the results of the *wait-time* analysis.



(a) FTP jobs and stage counts as reported by the monitoring.



(b) FTP Jobs as observed by the trace file

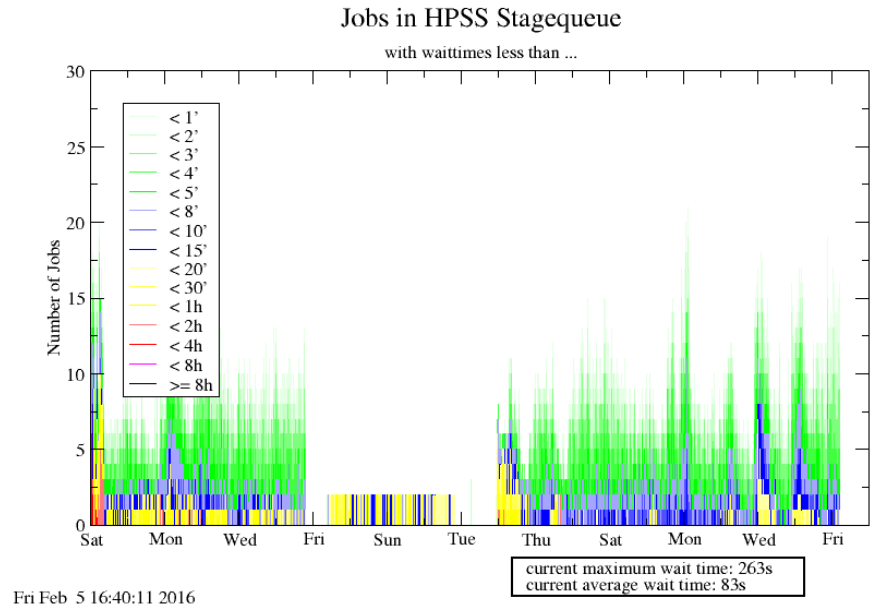


(c) Stages (red) as they are performed by the simulation.

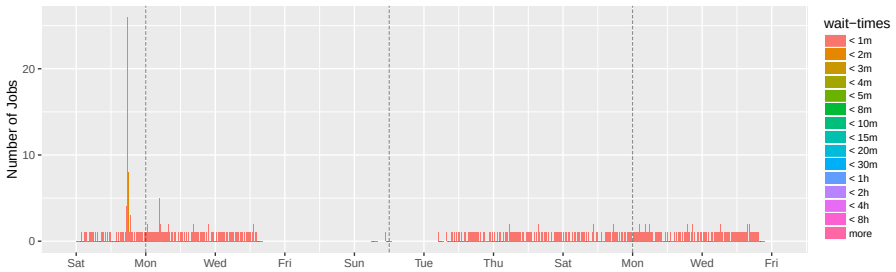
Figure 6.3: FTP activity for a period of three weeks. Validation of stage-counts by comparing the DKRZ's monitoring (a) to the results of the simulation (c).

Verification of Wait-times

Figure 6.3 groups requests by the time that they are already waiting. Shorter wait-times are visualized at a higher resolution than longer waiting requests. Most requests are waiting only a few minutes before they are served, but as workloads are bursty, sometimes wait-times of up to a few hours can occur. As observed when analysing the *stage-counts*, the simulation appears to handle requests faster than observed in reality.



(a) Wait times as observed by monitoring.



(b) Wait times as observed by the simulation.

Figure 6.4: Wait times as observed over a period of three weeks.

6.2.2 Virtual Setup

In Section 2.6 we briefly introduced the storage archive of the DKRZ. For the experiments a network topology and a library topology to resemble the general structure of the deployment at DKRZ were prepared. Notable characteristic of the setup are:

- Figure 6.5 is an illustration of the used network topology. Components colored in green are consistent across all experiments in this chapter.
- For the verification run only 60 Tape drives were used. For other experiments the number of used tape drives varies and is also specified by the caption of each plot.
- The library topology used six SL8500 tape library modules, but unlike the DKRZ's only a single library complex was assumed. The Pass-Thru-Ports and elevators were disabled in all experiments.
- The disk cache could fit up to 5 petabyte of data.

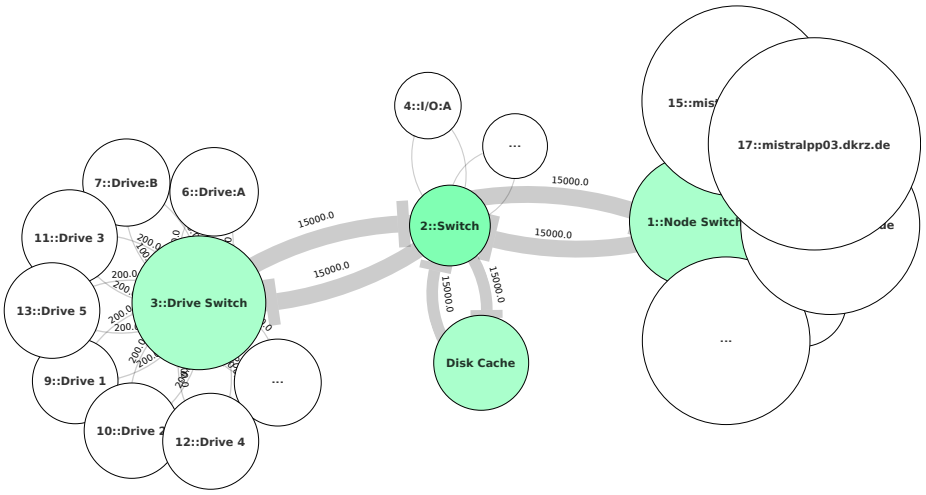


Figure 6.5: The network topology used to approximate the DKRZ'S deployment in the experiments.

6.3 Experiments

6.3.1 Experiment Candidates

Different experiments are anticipated, but are out of scope of the thesis. Still they shall be discussed as a starting point for future work. Experiments at the moment usually touch *verification* and *optimization*. Verification-experiments require more effort to construct. Optimizations tend to be resource intensive as many different configurations are tried to determine which performs best.

Drive Placement or Partitioning Optimization: Drive placement and partitions can have dramatic effects on the dynamics within tape library. By varying how partitions or drives are configured, a existing library may be improved.

RAIT Experiments RAIT provides many opportunities for tape optimzation. RAIT provides many parameters which can be varied, e.g., number of stripes, number of redundant copies in the system, RAIT layout strategies.

Budget Optimization Many systems are obtained on a budget. By including a price catalogues it becomes possible to minimize the initial cost, TOC or energy consumption.

Drive Type/Generation Optimization: Tape drives are among the most expensive components in a tape library. Older generations are usually more affordable, thus picking a mix of different tape drive generations may provide a similar performance at a reduced cost.

Tape Type/Generation Optimization: As with drives, tapes generations can be varied. To find a cheap transition strategy when a new tape generation is released, the mix of tape can be varied and compared.

Scheduling and Cache Policies: Scheduling occurs in multiple places within a tape system. By exchanging the scheduling components by experimental ones it is possible to evaluate their performance. Similarly it is possible to evaluate different caching policies.

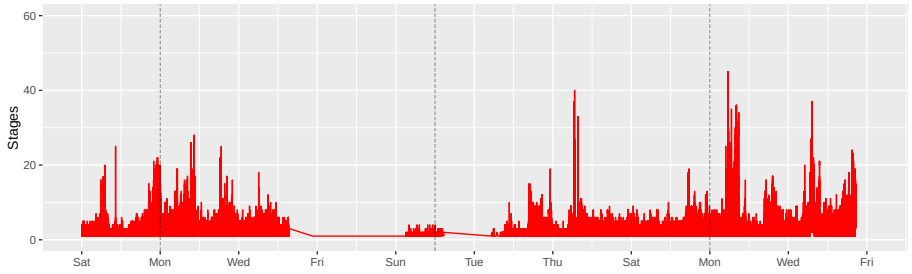
6.3.2 Experiment: Varying the Number of Tape Drives

To demonstrate that the simulation can be used to conduct experiments, the workload from Section 6.1 is used again. The same limitations as discussed in Section 6.1 and Section 6.2 also apply for the experiments in this section. However, this time the library configuration is changed in two ways and results are compared to the baseline run from Section 6.2.

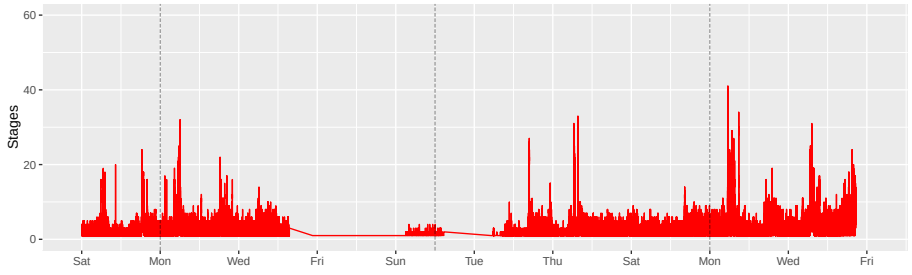
1. Reduced number of drives to understand system degradation.
2. Increased number of tape drives to validate performance improvement.

Effect on Stage Counts

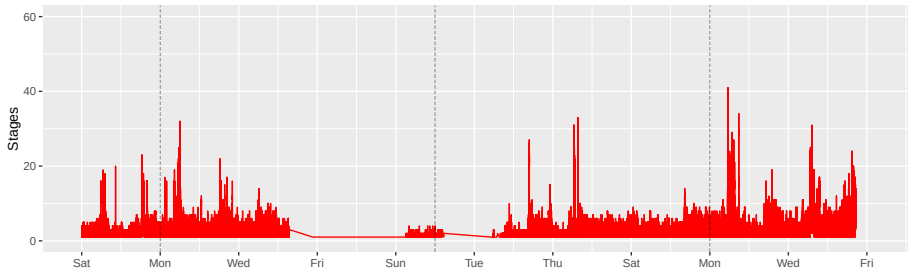
Figure 6.6 shows how the number of stages over time changes when decreasing (a) and increasing (c) the number of tape drives that are available in the system; (b) serves as a reference run with the same configuration as in Section 6.2. Consistent with expectations, reducing the number of tape drives distributes the number of stages over a longer period of time. Increasing the number of tape drives results in more immediate realisation of requested files being staged.



(a) Stages when number of drives is reduced to 30.



(b) Reference run, with original configuration with 60 drives (also see Figure 6.3)

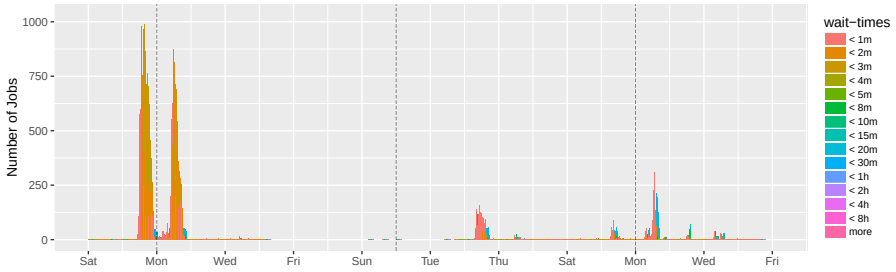


(c) Stages when number of drives increased to 75.

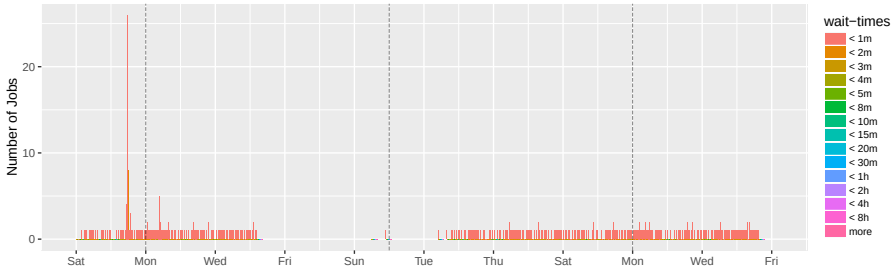
Figure 6.6: Stages for varying configurations with (a) reduced and (c) increasing number of tape drives; (c) serves as a reference run (see Section 6.2).

Effect on Wait-times

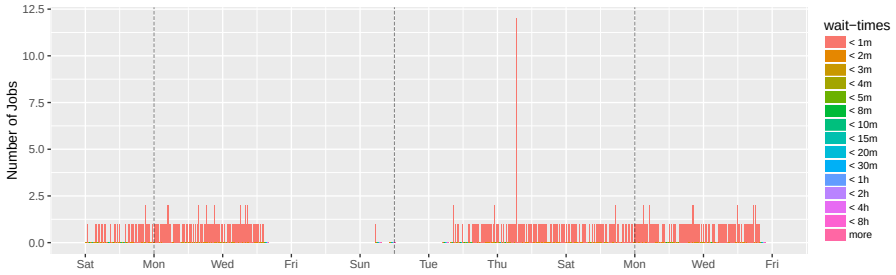
Figure 6.7 shows how the wait times change over time when decreasing (a) and increasing (c) the number of tape drives that are available in the system; (b) serves as a reference run with the same configuration as in Section 6.2. Because stage counts and wait times are to some extent related, it is no surprise that the observations match our expectation of a decrease in wait-times as we increase the number of drives.



(a) Wait-times when number of drives reduced to 30.



(b) Reference run, with original configuration with 60 drives (also see Figure 6.4)



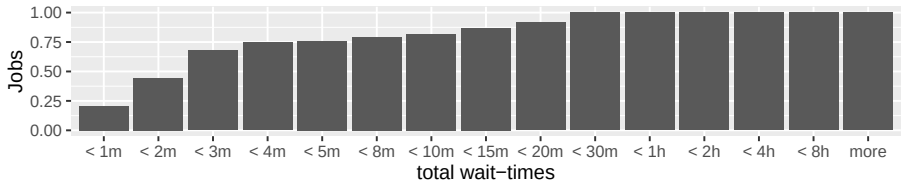
(c) Wait-times when number of drives increased to 75.

Figure 6.7: Wait-times for varying configurations with (a) reduced and (c) increasing number of tape drives; (c) serves as a reference.

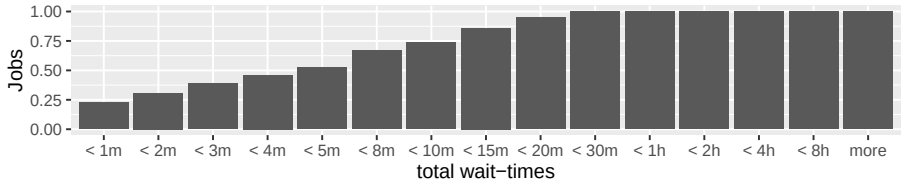
6.3.3 Optimizing for Quality of Service

By aggregating the total wait-times for users collected for different configurations, it is possible to narrow down the optimal configuration to provide a certain quality of service. Figure 6.8 plots the discrete cumulative distribution of wait-times for varying numbers of drives. It is now easy to make statements such as:

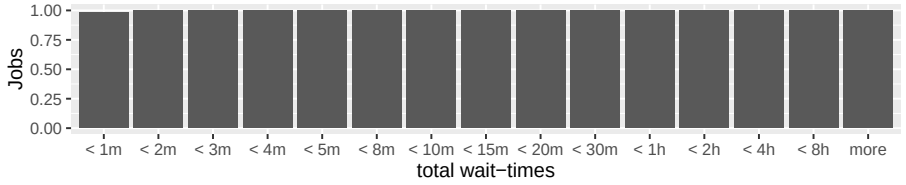
- All setups are suitable to serve 90% of requests in under 20 minutes.
- To serve at least 50% of the requests within 3 minutes, use 45 drives.



(a) Wait-times when number of drives set to 30 drives.



(b) Wait-times when number of drives set to 45 drives.



(c) Wait-times when number of drives set to 60 drives.

Figure 6.8: Wait-times for varying configurations with wait-times aggregated to pick optimal configuration for QoS.

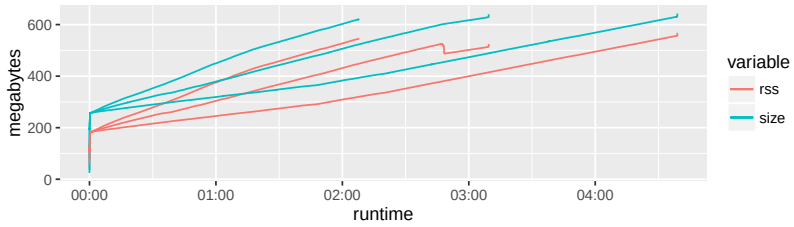


Figure 6.9: Memory profile of simulation runs. The number of drives effects the runtime. In ascending order by runtime: 30, 60 and 75 drives.

6.4 Performance of the Simulation

As using the model to automatically explore and optimize library configurations was an objection a brief analysis of the memory footprint and on the runtime should be performed. Figure 6.9 plots the *resident set size* (rss) and the *total memory consumption* of the process running the simulation over time. A typical simulation run that processes 140.000 requests takes about 3-4 hours mostly depending on the complexity of the network topology. There is some potential further reduce the runtime by providing a better schema to disabled debugging information and logging. All measurements were performed single threaded on a system featuring:

- Intel(R) Core(TM) i7-2677M CPU @ 1.80GHz
- 4GB system memory

Memory and Runtime Targets Memory consumption and runtimes should be kept as low as possible because a large number of different configurations is required for some experiments. As many parts of the prototype implementation use a simplified model, it is expected that the demands of a simulation will grow. A reasonable target seems to maintain the 3-4 hour time frame to process a month of FTP activity for systems similar to the one deployed at DKRZ.

Summary

Settling for a method to test a complex system implies significant limitations on the information value of later obtained results. The situations is worsened by the short-comings that come with any abstraction, especially when still a prototype. Yet, the simulation manages to approximate the behaivor observed in a real system. An approach to show how experiments could be performed the systems is consistent with expected experiment outcomes. The runtime performance and memory consumption are within bounds that also allow to run larger number of experiments on a notebook computer.

Chapter 7

Conclusion

This chapter summarizes the findings from the previous chapters and covers future work. Section 7.1 quickly revisits the essence of each chapter reviews what was achieved. It follows an assessment of future work in Section 7.2.

7.1 Summary

With the increasing demand for long-term storage, automated tape libraries will likely remain a integral part of the storage hierarchy for many years to come. Tape as a storage medium has many attractive properties. It es fairly resilient and provides high data densities, but by far the most important factor is that tape is very affordable in comparison to other storage technologies. Standardization efforts such as LTO make tape attractive and future proof, thus protecting investments. Despite tapes long history the technology is still advancing, though on a artificially slowed schedule.

In an effort to speed up innovation and enable also talent without access to large scale tape systems to contribute, the objectives of the thesis were to develop a simulator, the tools, and primarily the appropriate models required to reproduce the dynamics of hierarchical storage systems and tape libraries. In Chapter 4 we have seen that modeling a complete tape system is a incredibly complex task because so many different components are involved. Nonetheless, it was possible to identify a number of key components that are essential to any tape system. It was further possible to provide comprehensive models to describe the dynamics of many of these key components. In particular models for hardware and software components where proposed and isolated in such a way that turning to a more accurate model is possible when deemed necessary.

The problem with computer systems is the complexity that unfolds because of the virtually infinite number of possible combinations for hardware and software. Modeling hardware is particular cumbersome because in the real world the performance of a device emerges as a result of the laws of physics, but for a virtual model the dynamics have to be understood and abstracted. For standardized components it often is relatively easy to find a model that is adequately applicable for the whole class of of components. Unfortunately, composite components, such as the library topologies turn out to be harder to generalize in a simple way than expected. Nonetheless, by mixing mostly 2D and a graph-based topology approaches very good approximations of the library dynamics can be achieved. Another problem occurs with proprietary designs for which detailed information are hard to find. The same is true for benchmarks and a comprehensive catalogue of performance parameters.

The network is an integral part to hierarchical storage systems and can be used to model and simulate even low-level components and communications.

As a result, simpler algorithms similar to those used in real world systems could not be used directly but had to be adapted to account for the slightly different dynamics of the flow-based approach. In general it appears advisable to avoid faux models for components and subsystems when most other components depend on them. Simulation control flow and maintainability trumps shortcuts and short-term optimizations.

Hardware devices and subsystems such as the library topology and the network are controlled by software. For a proof of concept prototype it was often sufficient to turn to "naive" implementations. But because parts of the virtual software stack already begin to stabilize and because of the modular structure more sophisticated algorithms can be integrated in the future.

We could show that the footprint of the current simulation is fairly moderate at 400-600 MB main memory and 3-4 hours runtime. Components such as the library and network topologies, that differ from data center to data center, can be configured simply by providing a XML file. Adding or removing components is always possible through the topology APIs, so we can load a configuration and change a few parts by executing additional startup scripts that e.g., "installs" additional drives. A combination of commend-line arguments, configuration files and well designed APIs has proven to make experimenting much easier.

Besides starting a simulation, collecting results is important. Many tools scatter result files all over the current working directory. The simulator build for this thesis, creates a unique and timestamped directory for every run to store generated data in a consistent way. In addition components can register named CSV reports which can be used for logging of performance and debugging information. This structure proved useful, as it made aggregating results and generating reports much more user and machine friendly.

In an effort to verify the accuracy of the simulation additional tools and reporting capabilities were added. In particular it is now possible to directly use FTP *xferlog* files found in real systems as workload traces for the simulation. By feeding these workload traces to simulations prepared to use a library configuration similar to the one deployed at DKRZ it was possible to show that the models can be used to approximate the behavior of a real system.

Two key performance metrics were measured and compared to records from the DKRZ's tape library monitoring as a reference. For both metrics, the number of staged files and wait-times in the systems stage queue, the simulator could demonstrate a behavior analog to the one observed by the monitoring. While some fine-tuning is still required, the approximation is good enough to conduct first experiments. As a proof of concept, the number of tape drives installed to the virtual system was varied. Consistent with our expectations a reduction of tape drives degraded the tape systems performance and increased wait-times. Likewise an increase of tape drives reduced the wait-times and files were staged quicker.

Equipped with comprehensive models and a simulator to approximate tape archives within hierarchical storage systems, it is now possible to improve modern tape libraries without requiring a physical tape library for testing. Workflows to experiment and asses the performance directly influenced the design of the simulator, consequently the next step is to put it to use in experiments. Also gradually turning the simulator into a open source tape library management solution for production systems is now within reach.

7.2 Future Work

With a prototype to proof the feasibility of the concept, only the first step is made into production ready tools. The broader vision is to gradually turn the simulation into an actual open source tape system management tool. For this to become a reality some effort is necessary in terms of software development, in particular:

- Further improve the configuration process, for example through a GUI to setup library and network topologies.
- Mature existing architecture to allow physical tape library management.
- Port core APIs to more efficient programming languages were necessary so they would be suitable for deployment in real system.

From a research perspective the interesting part of a simulation is to apply it to practical problems to learn and generate new insight which was out of the scope the thesis. The logical next step is to carefully construct experiments with the current system and iteratively improve the tools required to conduct more experiments. In particular this might include parametrized Monte-Carlo methods to optimize for budgets or quality of service. The foundation to perform these kinds of experiments is provided with the work of this thesis. In addition, it would be useful to have a comprehensive database for benchmarks that collect the characteristics of drives, libraries and other devices. The database should also include component prices, though utilizing online price comparison APIs to fetch prices on demand may also be an attractive option when combined with a way to apply a correction factor for discounts.

Bibliography

Amazon (2016). Amazon Glacier Pricing . `https://aws.amazon.com/en/glacier/pricing/`. [Online; accessed 2016-02-08].

Amouroux, V. (2014). The end of memory? (movie).

A.M.P.A.S (2007). The Digital Dilemma - Strategic Issues in Archiving and Accessing Digital Motion Picture Materials. Technical report, Academy of Motion Picture Arts and Sciences.

Backblaze (2016). What Can 49,056 Hard Drives Tell Us? Hard Drive Reliability Stats for Q3 2015. `https://www.backblaze.com/blog/hard-drive-reliability-q3-2015/`. [Online; accessed 2016-01-20].

Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101.

Berlekamp, E. R. (1975). Algebraic codes for improving the reliability of tape storage. pages 497–500.

Dashti, A. and Shahabi, C. (2000). Data placement techniques for serpentine tapes. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pages 1–10.

Day, J. D. and Zimmermann, H. (1983). The OSI reference model. *Proceedings of the IEEE*, 71(12):1334–1340.

Dee, R. H. (2008). Magnetic tape for data storage: An enduring technology. *Proceedings of the IEEE*, 96(11):1775–1785.

DKRZ (2015a). Das weltweit größte Archiv für Klimasimulationsdaten steht in Hamburg. `https://www.dkrz.de/about/kontakt/presse/aktuell/archiv-2015/HPSS_archive`. [Online; accessed 2016-02-09].

DKRZ (2015b). Data Life Cycle Management at DKRZ. `https://www.dkrz.de/daten`. [Online; accessed 2016-02-09].

DKRZ (2015c). New supercomputer "Mistral" at DKRZ delivers particularly detailed regional climate simulations for Germany. pages 45–47.

DKRZ (2016). Services of DKRZ. `https://www.dkrz.de/about/dienste`. [Online; accessed 2016-02-20].

Eleftheriou, E., Haas, R., Jelitto, J., Lantz, M., and Pozidis, H. (2010). Trends in storage technologies. *Data Engineering*, pages 1–10.

Fontana, R. E., Decad, G. M., and Hetzler, S. R. (2013). The Impact of Areal Density and Millions of Square Inches (MSI) of Produced Memory on Petabyte Shipments of TAPE , NAND Flash , and HDD Storage Class Memories. *Proceedings of the 29th IEEE Symposium on Massive Storage Systems and Technologies*.

Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E. M., Sipos, B., and Birney, E. (2013). Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*.

Hitachi and Kiyataka, M. (2014). Successful read / write of digital data in fused silica glass with a recording density equivalent to Blu-ray Disc.

Hübbe, N. and Kunkel, J. (2012). Reducing the hpc-datastorage footprint with mafisc—multidimensional adaptive filtering improved scientific data compression. *Computer Science - Research and Development*, 28(2):231–239.

Hughes, C. J., Fisher, D., Dehart, K., Wilbanks, B., and Alt, J. (2009). HPSS RAIT Architecture 07/01/2009.

IBM (2011a). High Performance Storage System. Technical report.

IBM (2011b). IBM System Storage TS3500 Tape Library Connector and TS1140 Tape Drive support for the IBM TS3500 Tape Library. pages 1–15.

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

Jenkins, J., Zou, X., Tang, H., Kimpe, D., Ross, R., and Samatova, N. F. (2014). RADAR: Runtime asymmetric data-access driven scientific data replication. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8488 LNCS:296–313.

Johnson, T. and Miller, E. (1998). Performance measurements of tertiary storage devices. *Proceedings of the International Conference on Very Large Data Bases*, pages 50–61.

Koltsidas, I., Sarafijanovic, S., Petermann, M., Haustein, N., and Seipp, H. (2015). Seamlessly Integrating Disk and Tape in a Multi-tiered Distributed File System. pages 1328–1339.

Kunkel, J. M., Kuhn, M., and Ludwig, T. (2014). Exascale Storage Systems – An Analytical Study of Expenses 2 . Characteristics of Future Systems. pages 116–134.

Lingfang Zeng, D. F. (2005). Hybrid RAID-Tape-Library Storage System for Backup. *Second International Conference on Embedded Software and Systems (ICESS'05)*, pages 31–36.

Niset, M. and Kuhn, P. (2005). Typical Data Retention for Nonvolatile Memory. *Freescale Semiconductor Engineering Bulletin*, page EB618/D.

Nutaro, J. (2016). Adevs (A Discrete EVent System simulator). `http://web.ornl.gov/~1qn/adevs/`. [Online; accessed 2016-01-20].

Oracle (2015). StorageTek SL8500 Modular Library System User's Guide.

Ornl, S. K., Lanl, G. G., Snl, R. O., Pnnl, E. F., Lanl, G. S., Llnl, M. G., Lbnl, J. W., Exmatex, D. R., Asc, R. N., Asc, M. G., and Climate, D. W. (2015). DOE: Storage Systems and Input / Output to Support Extreme Scale Science.

O'Toole, J. (2016). Why Facebook is stockpiling Blu-ray. http://money.cnn.com/2014/08/21/technology/facebook-blu-ray/index.html. [Online; accessed 2016-01-20].

Page, B. (2016). DESMO-J. http://desmoj.sourceforge.net/home.html. [Online; accessed 2016-01-20].

Pantazi, A., Furrer, S., Rothuizen, H. E., Cherubini, G., Jelitto, J., and Lantz, M. A. (2015). Nanoscale track-following for tape storage. pages 2837–2843.

Pease, D., Amir, A., Villa Real, L., Biskeborn, B., Richmond, M., and Abek, A. (2010). The linear tape file system. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, 4.

Peixoto, T. P. (2014). The graph-tool python library. *figshare*.

QUADStor (2015). QUADStor Virtual Tape Library. http://www.quadstor.com/virtual-tape-library.html. [Online; accessed 2015-12-08].

Quantum (2015). Quantum Scalar i6000 Datasheet.

Quibik (2010). Computer Memory Hierarchy. https://commons.wikimedia.org/wiki/File:ComputerMemoryHierarchy.svg. [Online; accessed 2016-12-08].

Real, L. C. V., Richmond, M., Biskeborn, B., and Pease, D. (2015). An I / O Scheduler for Dual-Partitioned Tapes. pages 234–243.

Sandsta, O. and Midtstraum, R. (1999). Improving the access time performance of serpentine tape drives. *Data Engineering, 1999.*

SimPy (2016). SimPy (Event discrete simulation for Python.). https://simpy.readthedocs.org. [Online; accessed 2016-02-10].

Spectralogic (2016a). LTO Roadmap. https://www.spectralogic.com/features/lto-7/. [Online; accessed 2016-01-24].

Spectralogic (2016b). Spectralogic TFinity - Enterprise Performance. https://www.spectralogic.com/products/spectra-tfinity/tfinity-features-enterprise-performance/. [Online; accessed 2016-02-12].

Sun (2006). StorageTek StreamLine SL8500 - User Guide. (96154).

Top500 (2016). Top500 Supercomputer Sites. http://www.top500.org/. [Online; accessed 2015-12-08].

Varshney, M. (2016). http://simjs.com/. http://simjs.com/. [Online; accessed 2016-02-10].

Xu, C. and Lau, F. (1996). *Load Balancing in Parallel Computers: Theory and Practice*. The Springer International Series in Engineering and Computer Science. Springer US.

Zhang, X., He, D., Du, D., and Lu, Y. (2006). Object Placement in Parallel Tape Storage Systems. *Proceedings of the 2006 International Conference on Parallel Processing (ICPP'06)*, pages 0–7.

List of Figures

List of Tables

Versicherung an Eides statt

Ich versichere, dass ich die Masterarbeit im Studiengang Informatik selbstsandig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum, Unterschrift