TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät für Informatik

Dissertation

# QUANTILE FUNCTION-BASED MODELS FOR RESOURCE UTILIZATION AND POWER CONSUMPTION OF APPLICATIONS

Christoph Möbius
Born on: 26th July 1978 in Wolfen

to achieve the academic degree

Doktoringenieur (Dr.-Ing.)

Supervisor
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Referee
Prof. Dr. Miklós Telek

Submitted on: 18th December 2015
Defended on: 14th June 2016

*Den Menschen, die mir geholfen haben, der zu werden, der ich bin*

# ACKNOWLEDGEMENT

First of all I thank Prof. Dr. rer. nat. h. c. Alexander Schill for offering me the opportunity to write this dissertation at his chair and, most importantly, for his patience.

I need to thank my student assistants, Hongjin Zhu, and Chenjie Yu who supported me. I would particularly like to thank Jianjun Wen for developing and maintaing the video platform, and for coding many neat scripts that simplified the measurement procedures. I'm afraid I really bugged him sometimes. Also, Franz Eichhorn killed a lot of bugs and extended/changed/recoded the server applications as I wished.

Likewise, I thank my colleagues who let me find solutions to problems by mere listening.

I thank Neil Stephenson for his always intriguing and inspiring books. It was the reading of *The Diamond Age* that let me eventually choose Distributed Systems as my major and Wireless Sensor Networks as a specific lecture where I came across Dr. Waltenegus Dargie who eventually offered me the PhD position.

And talking about inspiration I also need to thank Sylvia Nasar for writing the (albeit unauthorized) biography of John Forbes Nash Jr., and Ron Howard who realized the movie. Without any exaggeration was it A Beautiful Mind that got me open to mathematics and let me lose all the reservation I inherited from school.

Not to forget, I would like to thank Donald Knuth and Leslie Lamport for making the writing of this thesis less arduous. Kudos!

Finally, my dear friends – it's only the cooking sessions that kept me going. Without you I would have been lost. I am *so* greatful to have you.

Dresden, August 2016

Thank you for the music, intergalactic.fm.

# CONTENTS


Acronyms 3

List of Symbols 5

Remarks 7

1. Introduction 9

2. Thesis Overview 13
   - 2.1. Testbed . . . 13
   - 2.2. Contributions and Thesis Structure . . . 14
   - 2.3. Scope, Assumptions, and Limitations . . . 15

3. Generation of Realistic Workload 17
   - 3.1. Statistical Properties of Internet Traffic . . . 17
   - 3.2. Statistical Properties of Video Server Traffic . . . 19
     - 3.2.1. Interarrival Times . . . 22
     - 3.2.2. File Size . . . 23
     - 3.2.3. Video Popularity . . . 24
     - 3.2.4. Popularity Dynamics . . . 26
     - 3.2.5. Temporal Locality . . . 28
     - 3.2.6. Prefix Duration . . . 29
     - 3.2.7. Considerations Regarding a Transcoding Service . . . 29
   - 3.3. Implementation of Workload Generation . . . 31
     - 3.3.1. Video Server . . . 31
     - 3.3.2. Transcoder . . . 35
     - 3.3.3. Benchmarks . . . 37
   - 3.4. Summary . . . 39

4. Models for Resource Utilization and for Power Consumption 41
   - 4.1. Introduction . . . 41
   - 4.2. Prior Work . . . 41
     - 4.2.1. Models for Resource Utilization . . . 41
     - 4.2.2. Models for Power Consumption . . . 44
   - 4.3. Test Cases . . . 49
     - 4.3.1. Video Server . . . 49
     - 4.3.2. Transcoder . . . 49
     - 4.3.3. Benchmarks . . . 50

4.4. Applying Regression To Samples Of Different Length . . . . . 50
4.4.1. Finding Pairs of Dependent and Explanatory Variables . . . . . 52
4.4.2. Estimate Differences between Empirical Distribution Functions . . . 54
4.4.3. Error Measures using the Shift Function . . . . . 58
4.4.4. General Remarks on the Approach . . . . . 63
4.5. Models for Resource Utilization as Function of Request Size . . . . . 63
4.5.1. Request Size . . . . . 64
4.5.2. CPU Utilization . . . . . 68
4.5.3. Storage Utilization . . . . . 71
4.5.4. Network Utilization . . . . . 74
4.6. Models for Power Consumption as Function of Resource Utilization . . . . 78
4.6.1. Video Server . . . . . 78
4.6.2. Transcoding Service . . . . . 80
4.6.3. SPEC CPU 2006 Benchmarks . . . . . 81
4.7. Summary . . . . . 82

5. Conclusion & Future Work 85
5.1. Summary . . . . . 85
5.2. Future Work . . . . . 86

Appendices 89

A. Graphs 91

Bibliography 95

# ACRONYMS

APE absolute percentage error 59

CCDF complementary cumulative distribution function 19, 32, 33

CDF cumulative distribution function 5, 22, 83

EDF empirical distribution function 5, 32, 36, 52, 53, 58, 60–62, 64–67, 69–76, 79, 80, 82, 83

GMM Gaussian mixture model 46

HAEC Highly Adaptive Energy-Efficient Computing 11, 85

HMM hidden Markov model 41, 42

ICT Information and Communication Technology 9

LAvg load average 49

LLC last level cache 64

MAPE mean absolute percentage error 58–62, 66, 68, 69, 71, 72, 76, 79–83

medAPE median absolute percentage error 60, 62, 66, 68–73, 76, 77, 81–83

MSE mean squared error 58, 60, 61, 71, 73–75, 77

NIC network interface card 42, 74, 75, 78

PACF partial autocorrelation function 63

PMC performance monitoring counter 46, 64

PME performance monitoring event 37, 44, 46, 47, 64

PSU power supply unit 46

**QoS** Quality of Service 58

**RAPL** running average power limit 46

**sMAPE** symmetric mean absolute percentage error 59, 60

**VMM** virtual machine monitor 42

# LIST OF SYMBOLS

### Greek Symbols

| | |
|---|---|
| $\lambda$ | request arrival rate |

### Mathematical Symbols

| | |
|---|---|
| $\lceil x \rceil$ | smallest integer greater than $x$; `ceil(x)` |
| $\lfloor x \rfloor$ | largest integer smaller than $x$; `floor(x)` |
| $=^?$ | "Is left-hand side equal to right-hand side?" |
| $=_d$ | follows in distribution |
| $a(s)b$ | range of integers from $a$ to $b$ with step size $s$ |
| $\hat{\Delta}(x)$ | empirical shift function |
| $\Delta^*(x)$ | upper confidence band for $\hat{\Delta}(x)$ |
| $\Delta_*(x)$ | lower confidence band for $\hat{\Delta}(x)$ |
| $F(x)$ | shorthand for $F_X(x)$ if context is clear |
| $F_X(x)$ | cumulative distribution function (CDF) of $X$ |
| $F_m(x)$ | empirical distribution function (EDF) of a sample of size $m$ for random variable $X$ |
| $F^{-1}(p)$ | shorthand for $F_X^{-1}(p)$ if context is clear |
| $F_X^{-1}(p)$ | left inverse of $F_X(x)$; also called *quantile function* |
| $F_m^{-1}(p)$ | *sample* quantile function |
| $lg\ x$ | $log_{10}\ x$ (*decadic logarithm*) |
| $ln\ x$ | $log_e\ x$ (*natural logarithm*) |
| $Pr[.]$ | probability |
| $\hat{Q}(q)$ | estimated quantile of order $q$; see also $F^{-1}(p)$ |
| $Q_C(q)$ | quantile of order $q$ of the CPU utilization |

$Q_D(q)$ quantile of order $q$ of the storage utilization

$Q_N(q)$ quantile of order $q$ of the network utilization

$Q_P(q)$ quantile of order $q$ of the power consumption

$Q_V(q)$ quantile of order $q$ of the (video) file size

$S_r$ request size

$x_q$ $q$-quantile (sample quantile of order $q$) for $X$

$X$ random variable

$X_{(i)}$ $i$-th order statistic in a sample for $X$

$\mathsf{M}(x)$ sample median

$\overline{x}$ sample mean

# REMARKS

Throughout this thesis we occasionally refer to *the* or *all* resources, or *every* resource of a server. If not explicitly stated otherwise this refers to the four main resources CPU, memory, storage, and network. Additionally, we understand the power drawn by a server as a fifth resource.

Also, we will use the terms *service* and *application* interchangeably.

Formulation, training and evaluation of the models in Chapter 4 were done entirely in Python using mainly the `pandas`[1] and `statsmodels`[2] libraries.

[1] `http://pandas.pydata.org/`
[2] `http://statsmodels.sourceforge.net/devel/`

The secret of getting ahead is getting started.

Mark Twain

# 1. INTRODUCTION

The power consumption of computing devices has been a concern in the research community since the 1960's. Since then, the networking of our world increased by orders of several magnitudes if one may take the numbers of installed servers, or transferred bytes per day as metrics. By the end of 2010, more than 32 million servers were deployed in data centers worldwide (Koomey 2011) and surely the numbers have risen since then. According to the International Data Corporation (IDC[1]), 2.2 million servers were shipped in 2013 in Europe, Middle East and Africa (EMEA) alone[2] (more recent figures are hard to obtain, however). All these machines together consume a considerable amount of energy every year: (Koomey 2011) estimates the energy consumption of data centers worldwide by the end of 2010 to be between 203.4 and 271.8 terawatt hours which accounted for between 1.12% and 1.5% of the worldwide energy consumption.

For Germany alone, a study by the Borderstep Institute states that in 2011 energy consumption of the infrastructure of Information and Communication Technology (ICT) was 9.7 terawatt hours which accounted for 1.8% of Germany's overall energy consumption in that year. This equals the production of four middle-sized coal power plants (Hintemann and Fichter 2012). By 2014, this energy consumption has increased to 10 TWh and if the trend continues it will reach 12 TWh by 2020 (Hintemann 2015). As a comparison, 10 TWh is the energy that was produced by all wind and solar power plants in Germany together in January 2015 (IWR 2015).

(Koomey 2011) also conjectures that the *increase* in power consumption by ICT is slowing down which the author attributes to the growing application of virtualization techniques. But still, ICT's power consumption is increasing, which the author blames mostly on an increase in the amount of power consumed per server. While there has been research ongoing for several years how power consumption of single machines can be reduced, idle power consumption of servers still accounts for up to 60% of the maximum power consumption (G. Chen et al. 2008) (the server that is used for the experiments presented in this thesis consumes 47% of its maximum power consumption in idle state). Obviously, the ratio of idle to maximum power consumption is still very bad and greatly reduces the energy efficiency of any computing system. Due to this bad ratio, low utilization of servers contributes massively to a waste of energy. A much-noticed article of 2007 showed that average CPU utilization for more than 5,000 servers over a period of six months was between 10% and 50% (Luiz André Barroso and Hölzle 2007). A large fraction of the servers utilized CPU even below 10%. This situation has not changed much since then: (Delimitrou and Kozyrakis 2014) state that average CPU utilization at Twitter is below 20% -

[1] `https://www.idc.com/`

[2] `http://www.idc.com/getdoc.jsp?containerId=prDE24788414`, 20th January 2015

while up to 80% of CPU capacity is reserved but usually idle. (Reiss et al. 2012) find that CPU utilization at a Google cluster is between 30% and 75%, while always more than 100% of the cluster's CPU capacity is allocated. And according to (H. Liu 2012), the CPU utilization at Amazon EC2 is between 3% and 17%.

Apparently, it would be best to switch a computer on only for the moment it needs to actually process something - which then ideally utilizes all resources to full capacity - and immediately switch it off afterwards.

Indeed, this is the idea behind server consolidation (also called service consolidation or, sometimes, workload consolidation). Here, the services that are running on these machines are put together on one server and the remaining servers - now relieved of their load and thus, running idle - are shut down. But consolidating the workload of several machines on one machine does not come for free: As a simple example imagine that there are two servers where the first runs a web server and the second runs a video transcoder, but both machines receive only few requests to the respective applications (or *services*). Therefore, it is decided to let one machine run both programs and to shutdown the other machine. Now, the web server and the transcoder compete for the resources of the remaining machine, i.e. CPU time, memory space, access to the (permanent) storage, access to the network interface. Due to the competiton both services may not get CPU time at the exact moment they need (e.g. a request to the web server or the transcoder is received). Likewise, both services may not get all memory space they need, or need to wait for the other program when trying to access the storage. Now the question arises, *how much* do these applications interfere at the relevant resource?

Having such knowledge at hand before actually consolidating workloads can support consolidation decisions in a manner that such consolidation is prevented that would lead to considerable resource interference. Since resource utilization and thus, resource interference, depends on the workload of the application it is crucial to estimate the resource utilization in dependence on the workload. And in order to obtain reliable results the availability of realistic workloads is a prerequisite. By *workload* we understand the user requests - comprising a request rate and a request size component - to an application where we need to distinguish between short-running applications that perform search tasks, sort lists, render some results, do system clean up, etc., and long-running applications like web servers, mail servers, video servers or database systems. In the literature, authors mostly employ CPU-intensive benchmarks like the SPEC CPU suites as example applications. While the resource interference at the CPU is of predominant interest, we also want to investigate resource interference at the storage and the network interface. Other authors, who also examine the storage mostly use standardized benchmarks like *Iometer* or *iozone*; alternatively they use their own benchmark applications. The aim of these benchmarks is to stress the I/O subsystem in a repeatable manner, rather than to mimic a realistic workload. In particular, if we interpret the number of bytes that are written to or read from the disk with each access as the request size, these benchmarks produce request sizes with only a small range of variation. But request sizes in Internet traffic are heavy-tailed, meaning that while most requests are of very small size there are a few requests that are really huge. And despite there are only few huge requests their size is so large that these requests have a large impact on the total amount of transferred data. Video data is a very good example for this kind of traffic. We therefore want to employ a video server in our experiments. As it points out, there is no workload generator for a video server freely available. Thus, we need at first to answer the following question:

1 What are the statistical properties of video server traffic and how can we generate traffic that resembles these properties.

Both, workload and resource utilization are usually non-deterministic and require means from probability theory to be described properly and put into relation. Typically, stochastical

models employ regression techniques in order to predict one variable (e.g. resource utilization) as a function of the other (e.g. workload). While widely adopted and applicable in many cases, regression generally requires that for all variables incorporated by the model one has an equal number of measurements (sample points) available. Unfortunately, practice sometimes fails to satisfy this assumption: Measurement data for different parameters (i.e. model variables) is obtained by different sensors that may sample at different frequencies (resulting in different numbers of sample points) and have different reliability (resulting in more or less missing sample points). E.g. a digital power analyzer that samples power consumption of a server or of one of its subcomponents can be understood as a sensor. A system monitor application that collects utilization metrics of system resources (e.g. CPU utilization or network utilization) provided by the operating system can also be understood as a sensor. Other sensors may monitor single applications and keep track of the requests to these applications and associated responses. In case of a web server, the server log can be regarded as the output of such a sensor. If the data that is produced by the sensors is provided with timestamps, i.e. the data is a time series, one may employ the timestamps to relate some sample points to each other. But first, it is not always guaranteed that clocks of sensors are synchronous. Sometimes, time stamps do not exist at all. In such cases one can still perform a correlation analysis to determine whether a linear relationship exists between the data. This, however is not sufficient to derive a functional relationship, i.e. a model for the data. We should therefore answer the question

2 How can we apply (linear) regression to samples of different lengths where time stamps are not meaningful, or unavailable, or unreliable?

An answer to that question should enable us to develop models that answer this third question:

3 How can we stochastically describe the resource utilization of an application as a function of its workload?

The answer to the third question are models that provide information how the resource utilization of an application (or service) *would* look like *if* the application had sole access to a machines' resources. We believe that this information provide means to detect possible interference at system resources independent of changes in the application's workload and also to quantify the amount of interference. An answer to this question must be, however, deferred to future work.

As just mentioned, it would be best to switch on a server only for the moment it needs to actually process something, but of course the processing itself must be energy-efficient as well to achieve a truly energy-proportional server. To develop such an energy proportional server architecture is the defined goal of the collaborative research center (CRC) 912: Highly Adaptive Energy-Efficient Computing (HAEC)[3] funded by the Deutsche Forschungsgemeinschaft[4]. Within the CRC, 14 professorships from the electronical engineering, mathematics, and computer science domains are collaborating in order to achieve this ambitious goal. To separate concerns and organize expertise, the contributors from the electronical engineering and mathematics domains are summarized in an A part, while the contributors from the computer science domain are summarized in a B part. The present thesis is a contribution to HAEC by the B06 project - the sixth project in the B part - which investigates in the energy-efficient execution of services.

[3] `http://tu-dresden.de/forschung/forschungskompetenz/sonderforschungsbereiche/sfb912/index_html/document_view?set_language=en`, 2015-01-20

[4] `http://dfg.de`

I am wondering, why are you here?

Yoda

# 2. THESIS OVERVIEW

In the following, we will shortly summarize the scope of this thesis, its limitations, and contributions. But initially, we present details about the test environment.

## 2.1. TESTBED

All experiments are carried out on a SuperMicro server with 4 Intel Xeon E5-4603 CPUs and 64 GB of memory. The storage is implemented as a RAID 0 system comprising three Seagate ST32000645NS disks. The RAID provides maximum data rates of 485 MB/s and 459 MB/s in write and read mode, respectively[1]. Network connectivity is provided by a 10 Gb/s NIC.

To utilize the server's resources we choose three scenarios: I/O-intensive, I/O- and CPU-intensive, and CPU- and memory-intensive. In the I/O-intensive scenario mainly the network and storage shall be stressed. We will implement this scenario with a video server. To utilize the CPU in addition to network and storage, we believe that a video transcoding application is a good example. For the third scenario we decide to employ benchmark applications from the SPEC CPU2006 suite.

The video server is provided by an Apache web server that is encapsulated within a Linux container[2]. For the video transcoding scenario we wrote a Django[3] application that is as well encapsulated within a Linux container. As transcoding backend we use `avconv`[4]. The workload that is sent to both applications should be realistic as possible. As it points out, this requirement necessitates research on its own and details on the workload generation are described in chapter 3. This chapter also explains the subset of benchmarks we employ in the experiments (subsection 3.3.3).

To obtain resource utilization data we use the `dstat` tool. The tool runs locally on the server and writes measurement data every second to a log file. To measure the power consumption of the server we employ a Yokogawa WT210 digital power analyzer and sample the server's overall power demand at the wall socket. Samples are provided at frequencies between 7 and 10 Hz and are written to a log file on a second server that has no connection to the experiment server. Figure 2.1 summarizes the test environment graphically.

[1] These results were obtained with the IOZONE benchmark; `http://iozone.org/`.
[2] `https://linuxcontainers.org/`
[3] `https://www.djangoproject.com/`
[4] `https://libav.org/avconv.html`

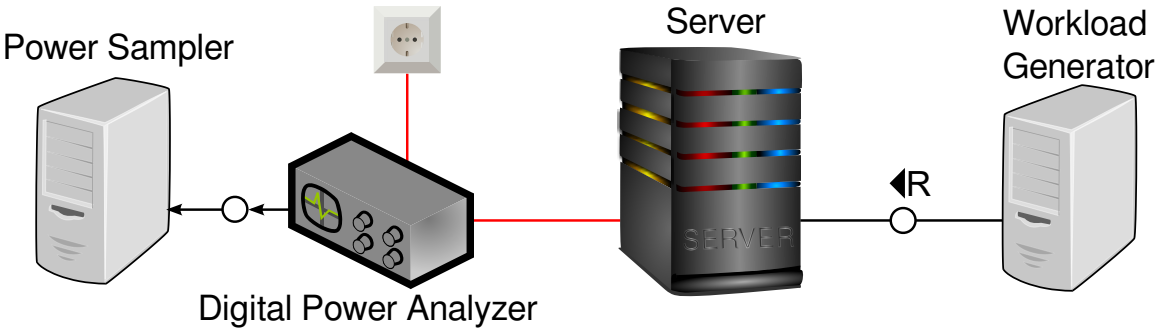



Figure 2.1.: Test environment

## 2.2. CONTRIBUTIONS AND THESIS STRUCTURE

As already mentioned, this thesis is embedded in the CRC 912 of the Deutsche Forschungsgemeinschaft. Very early in the project, we agreed together with other participating research groups on using a video server as common example and target platform. In order to obtain not purely academic results of our research it was clear to us that realistic workload for the video server needs to be generated. While other authors examine specific properties of popular video platforms like Youtube, or Dailymotion we notice that no publication considers *all* relevant statistical properties together. Furthermore, no workload generator is freely availble that reflects any of these statistical properties. In Chapter 3 we thus answer our first research question and describe

1 The design and implementation of a realistic workload generator for a video platform.

During the research on this topic we came across the interesting finding that the heavy-tailed transmission times that are apparent in Internet traffic are mainly caused by the distribution of file sizes on server side (M. Crovella and A. Bestavros 1997); the popularity distribution of files seems only to have a marginal effect on the request size distribution. It is tempting to derive a functional relationship between the distribution of file sizes on the server side and the distribution of request sizes since this would allow a server operator to predict resource demands by merely considering the (readily available) file sizes on the server(s) under her control. While the coefficient of correlation between the two distributions is easily obtainable it is, however, not sufficient since it only indicates the existence of *some* (linear) relationship. A solution to this problem would present a means to extract data points for a regression problem whenever it is not clear how data points in one sample are related to the data points in the other. In Section 4.4 we present a possible answer to our second research question and propose

2 An approach to obtain data points for a regression problem if data samples have different lengths or the data does not comprise any timing information.

This result shall enable us to answer our third research question. However, due to limitations that we summarize below, we are only able to respect one workload component. Therefore, in Section 4.5 we present our results on how to

3 Estimate the cumulative distribution function of resource utilization of an application as a function of the empirical distribution function of the application's request size distribution.

## 2.3. SCOPE, ASSUMPTIONS, AND LIMITATIONS

The main goal of this thesis is to express an application's resource utilization as function of the workload to the application. This is motivated by the expectation that such a functional relationship would allow to detect possible interference at system resources independent of changes in the application's workload and also would allow to quantify the amount of interference. The thesis' therefore focuses mainly on user-facing, i.e. interactive, applications that allow to derive meaningful statistics of the workload to the application. For batch jobs or requests that start management processes a request size is not necessarily directly deducible: Usually, such applications can be started "as is" without passing parameters that would allow to derive a *size*. Of course, one can measure the time the application needs to complete its task.

Our notion of *workload* - that it comprises a request rate and a request size component - assumes that every request to an application allows to deduce a request size. This requires to be able to monitor the requests to the application and its responses. In our test bed this is achieved by (a) accessing the Apache web server log, and (b) poviding non-server applications (e.g. the video transcoder) as web services and monitoring access to the web services.

During the evaluation of our approach to relate distribution functions we faced several problems that – at the present state of our research – restricts its general applicability. The most severe limitation arises from an unforseen correlation between the request size and the request rate component (see Section 4.4.4). Since our workload generator computes request rates independent of the request sizes this correlation must be attributed to our approach to obtain variables for the regression problem. Due to this correlation our resource utilization models are limited to simple regression problems and do not account for the time-dependent dynamics, i.e. the diurnal pattern, of request arrival rates.

To measure is to know.

Lord Kelvin

# 3. GENERATION OF REALISTIC WORKLOAD

In order to obtain realistic results we argue that it is crucial to base all experiments on realistic workloads. The following chapter will address the questions what *realistic* means in the context of video server traffic and how we generate such workloads. In detail, this chapter first presents statistical properties of Internet traffic in general in Section 3.1 and of traffic to video sites in particular in Section 3.2. Based on these findings a workload generator was implemented. Its details are described in Section 3.3.1. This workload generator was then adapted to generate video transcoding requests. Details for this adapted generator are presented in Section 3.3.2. Finally, to induce deterministic interferences, benchmarks of the SPEC CPU 2006 suite were employed. A third generator - that is described in Section 3.3.3 - issued runs of the benchmarks.

The term *workload* itself is ambiguous to some extent: It can be either interpreted as the utilization of a hardware resource during in a fixed period of time (Beloglazov and Buyya 2010; Kansal et al. 2010; Petrucci et al. 2011; Raghavendra et al. 2008; Verma et al. 2009), or it can be interpreted as the magnitude of client requests processed by an Internet server[1] (X. Fan, Weber, and Luiz Andre Barroso 2007; Moreno et al. 2013; Nathuji and Schwan 2008; Padala et al. 2007; Wood, Tarasuk-Levin, et al. 2009). Throughout this thesis the second interpretation is adopted. With this understanding, two parameters contribute to the magnitude of a workload: the request rate and the request size. The request rate denotes the frequency at which requests arrive at the server. As an alternative expression the request rate may be given in terms of interarrival times. The request size in this context is understood as the *amount of work* associated with individual requests. In this regard - since any work a server performs results in the utilization of hardware resources - the request size may denote the workload as in the first interpretation. The following section summarizes the statistical properties of request rate and request size in the context of Internet traffic.

## 3.1. STATISTICAL PROPERTIES OF INTERNET TRAFFIC

Studies that analyzed Internet traffic started to emerge in the middle of the 1990s. (Leland et al. 1994; Paxson and Floyd 1995; M. Crovella and A. Bestavros 1997; Mah 1997) and (Willinger et al. 1997) examined Ethernet, TCP, and HTTP traffic, respectively. All these

[1] As long as the context is clear, we use the terms *service* and *server* interchangeably. We use the term *physical machine* when we wish to put the emphasis on the hardware server.

studies find that Internet traffic is inherently bursty regardless of the time scale. This means that Internet traffic bears the property of self-similarity. As (Leland et al. 1994) point out: "[...] at every time scale ranging from milliseconds to minutes and hours, bursts consist of bursty subperiods separated by less bursty subperiods." Reasons for the self-similarity have been attributed to heavy-tailed distributions of interarrival times (Danzig and Jamin 1991; Paxson and Floyd 1995), the heavy-tailed distribution of transmission times (M. Crovella and A. Bestavros 1997) and, in general, to the superposition of many sources with ON/OFF behavior where the time periods of these ON/OFF phases are highly variable (Willinger et al. 1997).

As a consequence, interarrival time at the packet level can usually not be modeled by exponential distributions (Paxson and Floyd 1995). However, as the same authors also state the interarrival times for TCP *sessions* are indeed exponentially distributed. The reason that HTTP packet requests are not exponentially distributed lays in the structure of web documents: Usually, even a simple web page comprises text and images. Since the 2010's more and more web pages embed application code (mostly Javascript) and videos. A request for a web page therefore triggers several implicit requests to these embedded documents. Since these requests are made by the browser instead of by the user the interarrival times are much smaller than user-initiated interarrival times and thus, result in Pareto-distributed interarrival times as also described in (Barford and M. E. Crovella 1998) and (Z. Liu, Niclausse, and Jalpa-Villanueva 2001).

Following these observations one can conjecture that for a server that does not offer websites, but rather only video files the request interarrival times to be exponentially distributed. The conjecture is supported by observations described in (Acharya, Smith, and Parnes 1999): Although the authors did not provide a mathematical expression of interarrival times the plot shown in Figure 7 in the paper indicates a short-tailed distribution function and resembles an exponential distribution.

As mentioned above a second reason for the self-similarity of Internet traffic is the heavy-tailedness of transmission times (M. Crovella and A. Bestavros 1997). The authors argue that heavy-tailed transmission times stem back to heavy-tailed file transfers and that the root cause is the heavy-tailedness of the size distribution of files that are available at the server side (see also previous work by Mark E. Crovella et al.: (Cunha, Azer Bestavros, and M. E. Crovella 1995; Park, Kim, and M. E. Crovella 1996)). In (Barford and M. E. Crovella 1998) the authors argue that the file sizes may be best described by two distributions. They find that the body of the distribution follows a lognormal distribution while its tail is Pareto-distributed. (Z. Liu, Niclausse, and Jalpa-Villanueva 2001) present only an average value for document sizes at one web server. (Lee and M. Gupta 2007) state that the sizes of HTML objects (which follows our understanding of request size) follow a truncated lognormal distribution.

Other studies that analyzed the statistical properties of request sizes confirm the heavy-tailedness: In (Paxson and Floyd 1995) the authors find that for sizes of TELNET connections (in bytes) a general log-extreme distribution applies. (Mah 1997) find that the size of replies[2] follows a Pareto distribution.

Request interarrival times and request sizes are not the only components that determine the statistical properties of Internet traffic: In (Barford and M. E. Crovella 1998) the authors describe seven properties of web server traffic. Namely, these are statistical distributions of

1. file sizes (unique files available at the server side)
2. file popularity (relative number of requests for every individual file)

[2]The notion of reply size as used by the authors meets our understanding of request size.

3. request sizes (may differ from file sizes due to popularity of certain files)
4. embedded references (e.g. images or other files that are embedded in a web page)
5. temporal locality (likehood that a once requested file will be requested again within short time)
6. active OFF times ("browser think time", i.e. time between transfer of two files (or web objects) that belong to the same web page)
7. inactive OFF times ("user think time", i.e. time between requests to individual web pages by the same user)

In the present scenario only video files are to be requested by clients. This means that no web pages are to be requested and thus, the *active OFF* as well as the *embedded references* components are irrelevant. *Inactive OFF* times are then to be understood as request interarrival times.

As mentioned, the popularity denotes how often a certain file was requested in the past. For conventional web server traffic (HTTP requests to HTML documents and images) (Cunha, Azer Bestavros, and M. E. Crovella 1995; V. Almeida et al. 1996) and (Barford and M. E. Crovella 1998) find that popularity adheres to Zipf's law. Zipf's law is a special case of a power law and states that if elements of some multiset are ordered by the number of their occurence and this frequency is used as a ranking order then the rank is inversely proportional to its relative frequency. Since it is based on a power law it has the characteristic property of showing a straight line in a plot of the complementary cumulative distribution function (CCDF) where both axes have a logarithmic scale.

Popularity, temporal locality, and inactive OFF times are not independent from each other, but one is rather an effect of the other: In (Tang et al. 2003) the authors analyze a streaming server scenario and point out that the popularity together with the process of file introductions (i.e. uploads to the server) and the distribution of file life spans determine the temporal locality of file requests. The life span is the time after which certain file gains no more additional views (or only below some threshold). The next section will explain file popularity in the context of video or streaming servers in more detail.

Summarizing, one can record the following properties for conventional Internet traffic (i.e. static web sites):

1. File size distributions are heavy-tailed.
2. Popularity distributions typically adhere to Zipf's law.
3. Request size distributions are mainly influenced by the file size distribution. The file popularity seems to have only little effect on request size distribution.
4. Interarrival times for user-initiated requests can be modeled with exponential distributions if the to-be-modeled time period is short enough (i.e. not longer than one hour).

## 3.2. STATISTICAL PROPERTIES OF VIDEO SERVER TRAFFIC

For the case of video server traffic differences in the statistical properties of request rate, request size, file popularity, etc. are to be expected compared to conventional Internet traffic. This section summarizes the respective statistical properties and, where applicable, the reasons for these differences. In detail, the following factors that influence video server traffic are considered: request rate, file sizes, request sizes, video popularity (start

popularity), popularity dynamics (i.e. popularity gain), temporal locality, prefix duration. Additionally, some basic considerations regarding a transcoding service are presented.

One problem that will become apparent throughout this section is that no *recent* work exists where *all* components of the workloads of a video server were analyzed together. Additionally, many studies are comparatively old if one considers the speed of evolution of video platforms like YouTube or Vimeo. Therefore, it is not feasible to implement a workload generator would reflect current statistical properties of one particular major video platform entirely. As a result one is left to judge whether the reported statistical properties of workload components are (still) valid and to "pick" values that seem to be *reasonable* at the present time.

A second problem arises with respect to the evaluation of such a workload generator. First of all, it is obviously not possible to obtain access to servers of YouTube, or Vimeo, or other video platforms. Such an access would be valuable to learn the current statistics of file sizes. At the same time, the platforms forbid crawling which prevents "brute-force" approaches that just download a vast amount of videos in order to learn the videos' file sizes, and the file size is also not obtainable via API requests (YouTube, Vimeo, Dailymotion, myvideo, etc.)

For other workload components some authors intercepted traffic to YouTube within a campus network. One could argue that the same can be done for the evaluation of the workload generator. While this is in principle feasible to implement in a fashion to adhere to the German data protection law we argue that the necessary effort would not be justified by the mere result of validating the workload generator. To be reminded: The workload generator is mererely a tool to improve the reliability of this thesis' results, not its main goal.

To get an overview, all related work that was specifically considered with respect to the video server scenario will be shortly mentioned. The data collection methods and the workload components that were analyzed in the respective contributions will be summarized. Afterwards we will focus on the single workload components and the respective statistical properties as reported in the relevant work will be summarized.

(Acharya, Smith, and Parnes 1999) analyzed the accesses to a server of a Swedish University with 139 videos that were requested at least once. They investigated the statistical properties of file sizes.

In (Chesire et al. 2000) the authors analyze accesses to a streaming server from within the network of the University of Washington over the period of one week. They recorded connections of 4786 clients to 866 distinct servers with 23738 streaming objects. They were interested in the statistical properties of file popularity.

(J. M. Almeida et al. 2001) analyze accesses to educational videos on servers of two major U.S. universities (Berkeley, Wisconsin-Madison). The number of available files (requested at least once) were 1506 and 73, respectively. The paper provides statements with respect to statistical properties of interarrival times, and file sizes.

The authors of (Tang et al. 2003) employed long-term traces of two media streaming services at HP for their analysis: HP Corporate Media Solutions (29 months, 2999 files, 666074 sessions), and HPLabs Media Server (21 months, 412 files, 14489 sessions). They provide results for file popularity, popularity dynamics, and temporal locality. Additionally, the authors confirm that the request arrival process can be assumed to be stationary if the time period under consideration does not exceed one hour.

(Veloso et al. 2006) analyze logs for over one month of accesses to two streams of a Brazilian "Big Brother"-like TV show (691,889 users; 1,422,021 sessions) with respect to interarrival times, and file popularity.

For the analysis in (Gill et al. 2007) the authors intercepted accesses to YouTube from a university campus network; 323,677 unique videos were requested during the experiment.

They present statistical properties of file sizes, file popularity, and temporal locality.

The authors of (Cheng, Dale, and J. Liu 2008) utilized the API of YouTube to analyze statistical properties of file sizes, file popularity, popularity dynamics, and temporal locality. During their experiments they accessed more than 3 million unique videos.

In (Cha et al. 2009) the authors employed website crawlers to collect data from YouTube, Daum (Korea), Netflix, and Lovefilm (Europe's then largest online DVD rental store). The crawlers accessed YouTube's "Entertainment" category for one day (1,687,506 videos), YouTube's "Science and Technology" category once a day for a period of six consecutive days and once a day for two other days (252,255 videos), YouTube's 100 daily most popular videos once a day for 24 consecutive days (2091 videos on average per day), and the entire *Daum Videos* (a Korean video platform similar to YouTube) once (196,037 videos). The authors present results regarding file popularity and popularity dynamics.

Similar to (Gill et al. 2007) the authors of (Zink et al. 2009) also intercepted accesses to YouTube from a university campus network. During the experiment between 12955 and 303,331 unique videos (they collected multiple traces on different occasions) were requested. The paper presents statistical properties for file sizes, file popularity and temporal locality.

Websize crawlers were also employed by (Mitra et al. 2011) to obtain metadata from the sites of Dailymotion, Yahoo! video, Veoh, and Metacafe. For Dailymotion the "most popular" and "most recent" music videos on two separate days were accessed. On Yahoo! video all categories were crawled on three consecutive days and additionally the site-provided search engine was utilized on one day. On Veoh and Metacafe all channels were crawled on one distinct day for both platforms. The authors analyzed the file popularity and popularity dynamics.

In (Borghol et al. 2011) the authors first collected metadata for a selection of videos at YouTube and then observed the respective videos and their metadata over a period of eight months. The videos were selected over a period of one week (27th July to 2nd August 2008) based on two different approaches using the YouTube API: The first approach used an API call that returns the 100 most recent uploads. 29791 videos were selected in this way. The second approach used the API-provided keyword. The keywords were random draws from a dictionary. Based on the keyword the API call returns matching videos ordered by "relevance". 1,153,235 videos were selected using this appoach. The authors point out that sampling videos from "most popular"-lists or using keyword-based searches leads to skewed datasets due to the bias towards more popular content. Using the so collected videos the authors collected weekly view counts over a period of eight months. Borghol and collegues focus on popularity dynamics

This data was reused three years later by (Islam et al. 2013) to reevalute the model for popularity dynamics as proposed in (Borghol et al. 2011). They collected the view counts per week for the same videos that were used in (Borghol et al. 2011) over a period of two months from October 2011 to December 2011.

In (Li and Ma 2013) the authors collected log data of a video on demand system in the University of Science and Technology of China over 13 months. Overall they recorded data of 3,177,100 sessions that accessed 13,301 unique files. This data was used to analyze the statistical properties of file popularity.

Recently, (Karkulahti and Kangasharju 2015) criticize the methods to obtain video metadata as applied in previous work for producing biased samples. This comprises the search for related videos (BFS) as e.g. applied by (Cheng, J. Liu, and Dale 2013), the search for most recent uploads (MR) (see e.g. (Borghol et al. 2011; Szabo and Huberman 2010; Islam et al. 2013)), or intercepting traffic to YouTube (as applied e.g. by (Gill et al. 2007; Zink et al. 2009)). They suggest to generate random strings of length 4 and to ask the YouTube API to return videos of which the video ID's contain the random string. They find that the

popularity distribution of videos obtained by this method is more Zipf-like than for the other methods. Also, they find for any random string 6.9 videos are returned on average by the API. Additionally, they note that every video ID they get as response contains a dash (–), but the authors assure that this does not lead to any bias.

### 3.2.1. INTERARRIVAL TIMES

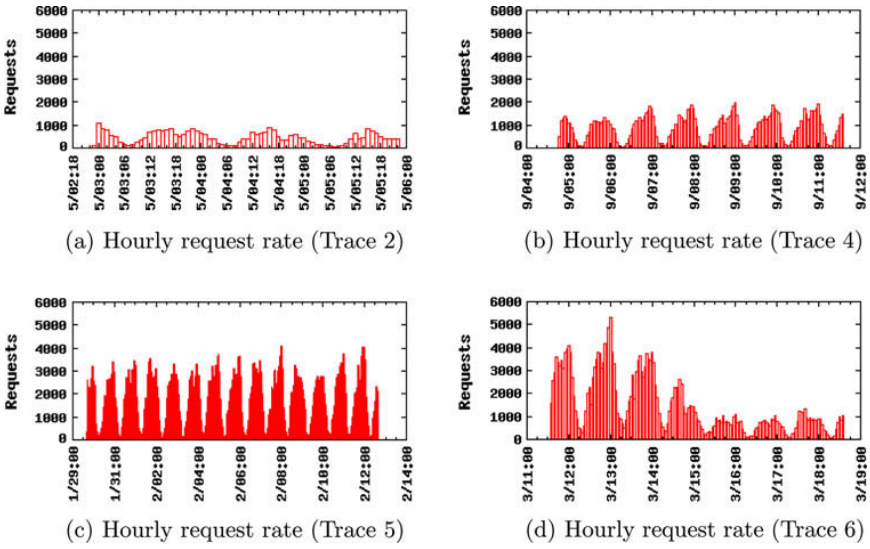
Unfortunately, there are no recent studies (i.e. not older than two years ) that provide details on the nature of interrarival times specifically for video servers of platforms like YouTube, Vimeo, Metacafé, etc. The reasons are a) the video platform operators or the operators of the associated content delivery networks do not publish data in that regard[3] and thus, b) such studies necessitate either access to the servers or the interception of a greater network which gives in turn rise to privacy concerns. While for some studies the authors indeed employed network interception (Gill et al. 2007; Zink et al. 2009) they did not investigate the nature of interarrival times or the request rate, respectively. The graphs shown in (Zink et al. 2009) allow at least a graphical analysis. The nature of the depicted cumulative distribution functions lets one conjecture that the interarrival times could be normally distributed with variances in the range of one order of magnitude (the CDFs do not resemble lognormal distributions).

However, two teams of authors examined the nature of interarrival times for small-scale video servers and streaming media servers, respectively: In (J. M. Almeida et al. 2001) the authors find that for the Berkely Internet Broadcasting System - which offers live multicasts as well as previously recorded videos of lectures - the client session interarrival times are exponentially distributed, while for the eTeach system - which offers videos of lectures and laboratory demos - the session interarrival times are Weibull-distributed or follow a combined distribution of Weibull (body) and Pareto (tail). The authors do not give an explanation for a possible cause of the heavy-tailed distributions. In (Veloso et al. 2006) the authors confirm that also for a media streaming server the client session arrivals can be modeled by a stationary Poisson process if the modeled period does not exceed one hour.

Based on the afore-mentioned findings, and the considerations in Section 3.1 it is assumed that interarrival times to the video server are exponentially distributed and that the arrival rate can be considered constant for time periods no longer than one hour.

Usually, the diurnal patterns of request arrival rates have a wave-like shape. This meets the expectations as the waves follow the day and night cycle of human users. Examples of this wave-like behavior can be found in (Paxson and Floyd 1995; J. M. Almeida et al. 2001; Tang et al. 2003; Veloso et al. 2006; Gill et al. 2007; Cha et al. 2009; Zink et al. 2009; Szabo and Huberman 2010) for web servers, and video servers. Unfortunately, no team of authors tries to characterize the nature of the wave forms, but a y-shifted sine or cosine function appears to be a sufficiently good approximation for many of the cases (Paxson and Floyd 1995; Tang et al. 2003; Veloso et al. 2006; Gill et al. 2007; Zink et al. 2009). The nonstationarity of interarrival times could thus be modeled by letting request arrival rates follow such a wave form. A similar wave-like pattern is recognizable between the single days of a week (J. M. Almeida et al. 2001; Veloso et al. 2006; Gill et al. 2007), and (Zink et al. 2009). An example of these patterns is shown in Figure 3.1. Section 3.3.1 provides details on how the diurnal pattern and the weekly pattern are actually modeled.

[3] Only very coarse-grained data is published; e.g. as of 2009-10-09 YouTube served one billion views per day (roughly 11574 requests per second) according to `http://youtube-global.blogspot.de/2009/10/y000000000utube.html`




(a) Hourly request rate (Trace 2)

(b) Hourly request rate (Trace 4)

(c) Hourly request rate (Trace 5)

(d) Hourly request rate (Trace 6)

Figure 3.1.: Examples of wave-like diurnal patterns of request arrivals; taken from (Zink et al. 2009).

### 3.2.2. FILE SIZE

Regarding the file size component the availability of recent studies for the video server scenario is similarly pitiable: The most recent publication that analyzed the statistical properties of file sizes for YouTube is dated by 2009 (Zink et al. 2009). The paper finds that the average file size of YouTube is between 6.3 and 7.5 MB (depending on the recorded trace). The minimum file size is between 484 and 95,760 Bytes while the maximum file size is between 130 and 218 MB. A median file size is not given, but by means of a graphical analysis on Fig. 8(b) this value can be estimated to 5 MB for all traces. The authors do not try to determine the underlying distribution. A similar study two years earlier (Gill et al. 2007) finds that average file size is 10.1 MB, the median file size is 8.125 MB, and that 10% of all videos are larger than 100 MB. Cheng and colleagues (Cheng, Dale, and J. Liu 2008) find the average file size to be 8.4 MB and 98.3% of all videos to be smaller than 25 MB. The median value can also only be estimated by means of graphical analysis. Employed on Fig. 5 a value of 6.8 MB can be assessed.

An earlier study finds that the median file size on a video server of a Swedish University is 110 MB (Acharya, Smith, and Parnes 1999). For videos that are part of educational systems at two different Universities (J. M. Almeida et al. 2001) first grouped the available files into 5-minute intervals. For the educational system (Berkeley University, BIBS) the presented graphs show that approx. 50% of all files are between 5 to 10 minutes, and approx. 25% of all files are between 55 and 60 minutes long. The relative frequency of other 5-minute intervals between 10 up to 135 minutes is nearly uniformly distributed. The authors state that for BIBS more than 96% of the transfered minutes of videos are for videos with encoding rates between 350 and 500 kbps. For the eTeach system at Wisconsin-Madison almost 40% of all files have a length up to 5 minutes with a sharp drop to approx. 10% for videos between 10 and 15 minutes. Approx. 15% of all files are between 45 and 50 minutes long. Between 10 and 45 minutes the relative frequency of lengths decreases from approx. 12% to approx. 1%. The encoding rates for these files are given to be 300 kbps in most of the cases.

An important insight given by (Gill et al. 2007) is the large deviation between the number of requests for certain file types and the respective contribution to overall traffic size: 86% of all requests are for images and text, 10% of all requests are for application data (Javascript, Flash) and only 3% of all requests are for video files while the responses to

video file requests account for 98.6% of the entire traffic volume. A realistic workload generator for a video platform should thus also generate requests to images, text files and application data (i.e. swf, Javascript) since processing these requests induces non-negligible CPU load. However, the workload generator that is considered in the paper is intended to issue requests to a server that merely offers video files.

Summarizing, the average file size on YouTube seems to have been decreasing over the period from 2007 to 2009 ((Gill et al. 2007): 10.1 MB; (Cheng, Dale, and J. Liu 2008): 8.4 MB; (Zink et al. 2009): 6.3 MB - 7.5 MB). This observation is particularly interesting since Gill et al. and Zink et al. employed the same method to obtain file statistics (intercepted campus network) and both teams collected data for a similar number of files (323,677 and 303,331, respectively). But since upload limits have been increased after the studies one may safely assume that the *average* file size has increased since then. For the median only (Gill et al. 2007) report a value (8.125 MB), but median values for (Cheng, Dale, and J. Liu 2008) and (Zink et al. 2009) can be inferred by graphical analysis. Here, the same decreasing trend as for average values can be discerned: ((Gill et al. 2007): 8.125 MB; (Cheng, Dale, and J. Liu 2008): approx. 6.8 MB; (Zink et al. 2009): approx. 5 MB). This development suggests that either users tend to record and upload shorter and shorter videos over the years or that better encoding algorithms contributed to the reduction of file sizes.

To gauge how request sizes may have developed the following is reflected: On the one hand the resolution of devices and thus of videos has increased since 2009 over the years which would suggest that the median value has increased as well. On the other hand it appears unlikely that users have changed their preferences to record (and upload) short videos. Since no more recent values are reported in the literature a compromise is made and the values as reported in (Cheng, Dale, and J. Liu 2008) are taken as reference points for the workload generator.

The question remains which distribution the file sizes follow. It seems reasonable that the reason that neither of (Gill et al. 2007), (Cheng, Dale, and J. Liu 2008), and (Zink et al. 2009) report heavy-tailed file size distributions is the then existing limit at YouTube for uploaded files; the limitation of video lengths was increased from 10 minutes to 15 minutes for normal users as of July 2010[4]. Also, since December 2010, it allows the upload of videos longer than 15 minutes and larger than 20 GB for users with *good conduct record*[5]. YouTube now allows uploads of up to 11 hours with a 128 GB limit on file size [6]. Likewise, Vimeo currently allows uploads of up to 5 GB for standard users and up to 20 GB per upload for *Pro* users[7]. Several other video sharing sites enforce no limit on uploads (e.g. Internet Archive; Veoh[8]). Therefore, it is considered justified to assume a heavy-tailed distribution for the file sizes. Since no particular information for the case of a video platform, or video server is available the knowlegde regarding file size distribution in usual Internet traffic is reemployed.

### 3.2.3. VIDEO POPULARITY

Chesire and colleagues (Chesire et al. 2000) analyze the workloads of a streaming-media server. They find that the popularity of RTSP streams follows a Zipf function with $\alpha = 0.47$.

[4] http://youtube-global.blogspot.de/2010/07/upload-limit-increases-to-15-minutes.html, 2014-09-18

[5] http://youtube-global.blogspot.de/2010/12/up-up-and-away-long-videos-for-more.html, 2014-09-18

[6] https://support.google.com/youtube/answer/71673?hl=en&rd=1, 2014-09-18

[7] https://vimeo.com/pro, 2014-09-18

[8] http://www.veoh.com/static/faq/Uploading_and_Publishing_Video/Is_there_a_size_limitation_for_video_.htm, 2014-09-18

In (Tang et al. 2003) the authors likewise investigate characteristics of a streaming-media server. They find that the popularity of files does not follow the classical Zipf function. They observe a circular shape in the rank/frequency plot and suggest a Zipf k-transformation such that the plot shows the characteristic straight line after transformation. The authors state that one possible explanation of the phenomenon that the lifespan of many files is very short and thus, many files gain a similar amount of requests. Since the observation period is much longer than these short lifetimes one will observe more files with few requests than a pure Zipf function models which leads to a circularly shaped rank-frequency plot. The authors hence suggest to employ a rank-frequency plot for popularity groups rather than individual files.

A streaming scenario is also the basis for the considerations in (Veloso et al. 2006). Here, the authors observed Zipf-like functions, but they point out that in the context of live streaming it may not quite be appropriate to use the term "popularity" since clients cannot revisit the specific content. They rather speak of "interest" as the frequency of a client accessing a particular live stream.

Allmost all authors who analyzed statistical properties of the workloads to YouTube servers find that the file popularity follows a Zipf-like function: Gill et al. observe a pure Zipf-like function (Gill et al. 2007) and determine the exponent $\alpha = 0.56$ from the rank-view plot. Cha et al. assess Zipf functions with exponential cutoff that they attribute to post-filtering due to recommendation systems (Cha et al. 2009); They find that the cutoff is dependent on the age of videos and that younger videos follow Zipf's law better, in particular videos with an age of one day. Zink et al. only provide graphs for the popularity analysis, but a pure Zipf function is apparent (Zink et al. 2009). Cheng et al. observe a skewed Zipf function (Zipf with cutoff) for the video popularity and find that a Weibull or a Gamma function fit the data better than a pure Zipf function (Cheng, Dale, and J. Liu 2008). They attribute the skewed behavior to users who access their own videos after upload in order to ensure everything went correct, but the videos will not be accessed any more in the future. Contrary to (Cha et al. 2009), the cutoff does not depend on the videos' age.

In (Mitra et al. 2011) Mitra and colleagues examined other video platforms than YouTube and find that file popularities for Dailymotion and Veoh follow a power law with exponential cutoff, and a pure power law for Yahoo! video and Metacafe.

Borghol et al. (Borghol et al. 2011) present a different approach to characterize file popularity: They distinguish three phases, where a video is before, at, or after its peak popularity. However, they find that the distribution types are phase-invariant and within each phase the distributions are week-invariant (parameters are constant). For each of the three resulting distributions the authors modeled body and tail of the respective distribution function separately where they defined the tail as the part of the empirical distribution that comprises the largest 10 % of the views. For the body, Borghol and collegues employed a $\beta$-distribution, and a lognormal distribution for the tail. To decide which video is in which phase the authors take samples from a time-to-peak distribution which they define as follows: Approx. 75% of all videos gain their peak popularity within the first six weeks of the observation period and the distribution function takes the form of an exponential distribution. The remaining 25% of videos peak uniformly after six weeks until the end of the observation period (32 weeks).

Li and Ma (Li and Ma 2013) also observe a cutoff in the popularity distribution for their data, but miss to explain this observation. They suggest to employ a stretched exponential distribution to model the file popularity and the CDF of the stretched exponential distribution is defined $Pr_c[X \leq x] = e^{-(\frac{x}{x_0})^c}$ where $x_0$ is a constant named *reference scale* and $c < 1$[9] denotes an addtional exponent to the common exponential distribution that represents the "fatness of the tail" (Laherrère and Sornette 1998).

[9] For $c = 1$ it is the common exponential distribution and for $c > 1$ it is a compressed exponential distribution.

Summarizing, popularity distributions follow a power law or power law with exponential cutoff in many cases which means that the rank-frequency plots follow Zipf's law. Where deviations from the power law (Zipf's law) can be observed these are attributed to recommendation systems (Cha et al. 2009), or to the presence of many files with similar (low) popularity (Tang et al. 2003; Cheng, Dale, and J. Liu 2008). Other authors find that file popularity can be modeled by either pure power law, power law with cutoff, or a lognormal distribution (Mitra et al. 2011). The video server that is used throughout our experiments does not implement a recommendation system. Thus, it would be justified to assume that file popularity adheres to a power law. However, the authors of (Mitra et al. 2011) make their original measurement traces publicly available which provides, among other data, popularity of the files they accessed. We will use the data for Dailymotion[10] to obtain file popularity variates.

### 3.2.4. POPULARITY DYNAMICS

In (Tang et al. 2003), the authors presented an approach for a load generator especially for media servers. The authors criticize SURGE (Barford and M. E. Crovella 1998) particularly for the assumption of a stationary file popularity. To account for the nonstationarity, Tang et al. employ the concept of file lifespan as the characterization how a file's relative popularity changes over time. This is the share of total requests in relation to a file's age. The authors determine two types of lifespan distributions: regular lifespan distribution and news-like lifespan distribution, respectively. The authors state that a lognormal distribution fits the regular lifespan distribution best while the news-like lifespan distribution follows a Pareto distribution. For both, lognormal and Pareto distribution, Tang et al. find that the values of the respective distributions' parameters are normally distributed across files and determined following values:

| Parameter | $\mu$ | $\sigma$ | |
|---|---|---|---|
| $ln\,\mathcal{N},\ \mu$ | 3.0935 | 0.9612 | regular |
| $ln\,\mathcal{N},\ \sigma$ | 1.1417 | 0.3067 | |
| Pareto $\alpha$ | 0.7023 | 0.2092 | news-like |

Table 3.1.: Normal distributed values for lifespan distributions (Tang et al. 2003).

As (Cheng, Dale, and J. Liu 2008) point out the evolution of popularity is very different between videos. To account for the differences, Cheng and colleagues define the following *growth trend* model

$$v(x) = v_0 \cdot \frac{(x+\mu)^p}{\mu^p} \tag{3.1}$$

that returns the number of views after $x$ weeks for a video that is already $\mu$ weeks old and gained $v_0$ views until that time point, and where $p$ denotes the *growth trend factor*. For values $p > 1$ the video is getting more popular while values $p < 1$ indicate a decreasing popularity. The authors only provide a plot for the distribution of $p$, but graphical analysis indicates that $p$ is very well modeled by a Weibull distribution $\mathcal{W}(2, 0.9)$. The age of a video can be determined by the upload trend presented in the same paper which denotes the added videos per week. The authors find that this trend follows a power law with $\alpha = 2.61$.

To examine the change of popularity Cha and colleagues (Cha et al. 2009) first investigate whether the number of requests to a video correlate with the videos' age (binned to five

[10] `http://www.cs.usask.ca/faculty/eager/TWeb10.html`, 2014-08-30

days) and then how popularity changes with the video's age. For YouTube they observe the added views of videos in the "Scientific" category over a period of six days (only videos that gained at least one additional request are factored in). They find that except for videos younger than a month the *average* number of requests is insensitive to the videos' age. Very new videos gained more views than older ones since younger videos outnumber older videos. However, on a daily basis the authors notice that more recent videos gain the most requests: For a single day they observe that almost 50% of the 20 most popular videos are not older than three weeks. But the emphasis on more recent videos abates if the time window is increased. And the age of a video also affects the probability to become very popular: The authors notice that the probability to gain more than 10 requests decreases with a video's age. For the periods of 24 hours, six days, three weeks, and 11 months the probabilities that a video was requested more than 10 times are 0.43, 0.18, 0.17, and 0.14, respectively. Thus, as Cha et al. point out, it is unlikely that a video becomes very popular if it did not already gain many views within the first two days after upload - they state a correlation coefficient of 0.9367 between the views after two days and seven days (after upload), and of 0.8525 for the views after three days and 90 days (after upload), respectively. To develop a model that captures the popularity dynamic they observed the 100 most popular videos for a period of 24 days and analyzed the evolution of the daily added views for every video that ever belonged to this group during the observation period. They define the *growth rate* for an individual video as

$$d_i(x) = \frac{\sum_{t=0}^{x} r_i(t)}{\sum_{t=0}^{T} r_i(t)} \tag{3.2}$$

where $r_i(t)$ denotes the number of requests for video $i$ on day $t$, $T$ denotes the observation period, and $x$ is the timepoint of interest. A graph is provided that plots the 10, 50, and 90 percentiles for the growth rate over a period of 14 days. The plot indicates that for 10% of the videos the growth trend is 0.8 after one day (10% of the videos gain 80% of their total views after one day), and that for 50% of the videos the growth trend is 0.3 after one day (50% of the videos gained 30% of their total views after one day). However, the authors do not provide a more detailed analysis.

Mitra and colleagues (Mitra et al. 2011) crawled Dailymotion on seven consecutive days for recently added videos. They notice that more than 40% of the 0.01% of most popular videos change from one week to the other; only around 20% of the 0.01% most popular videos are still in the 0.01% most popular videos set after 12 weeks (*popularity overlap*). For the 0.1% most popular videos this portion increases to more than 40%, more than 60% of the 1% most popular videos are still in the 1% most popular set after 12 weeks and if one considers the 10% most popular videos then approx. 75% of the videos in this popularity set are still in this set after 12 weeks. The most churn can be observed within the first 4 weeks after the measurement where the association to the 0.01%, 0.1% 1% and 10% most popular video sets seem to stabilize after that time (i.e. the ratio between the videos that are in one of these popularity sets in one week and are still in the respective set in the following weeks becomes nearly constant).

In (Borghol et al. 2011) the authors state that popularity dynamics for single videos are "futile" and should rather be considered for collections of videos. They suggest a three-phase model (before, at, and after popularity peak) to model the popularity dynamics of new uploaded videos. The transition from before-peak to at-peak and from at-peak to after-peak phases happens according to a time-to-peak distribution which is two-part: 75% of all videos reach their peak popularity within the first six weeks and values in this .75 quantile are exponentially distributed with $\lambda = 0.598$. Values in the upper .25 quantile are uniformly distributed between 7 and 32 (until the end of the observation period). Popularity dynamics are then modeled by moving videos from either the before-peak to the at-peak

phase or from the at-peak to the after-peak phase. Moving between phases is actually modeled by assigning a certain number of (additional) views to a video where the number of additional views are sampled from three distinct distributions - one for each phase - that estimate the number of additional views per week. The authors assume that not only the type, but also the parameters of the respective distributions are week-invariant so that for all weeks the number of additional views are always sampled from the same distribution for the respective phase. For all three phases the distributions of weekly added views are modeled by a $\beta$-distribution for the body and a lognormal distribution for the tail where the tail contains those 10% of videos that gain the most additional views in the respective week. The authors find that this basic model does not model the popularity churn well enough and define an extended model that randomly exchanges the added view counts of two videos,$u$, $v$ within the same phase for a certain week, $i$, where $u$'s and $v$'s added view counts for that week, $x_i^u$ and $x_i^v$, respectively also need to be in a so-called exchange window which is defined as $W_i^v = \left[\frac{x_i^v}{g}, min(x_i^v \times g, x_{max}^v)\right]$, where $g \in [1, \infty)$, and $x_{max}^v$ is the maximum added views count over the lifetime of $v$, and $g$ determines the amount of possible churn.

(Islam et al. 2013) reuse and reevaluate the model of (Borghol et al. 2011) to better understand long-time effects. If the popularity overlap is determined between consecutive weeks then for both, the 1% and 10% most popular video sets, the extended model provides better estimates where the best choice of values for $g$ depends on the videos' age: The older the videos get the better is a choice of small values for $g$. Similar results are observed in the case where the popularity overlap is determined between the second week and all following weeks of the observation period.

While the models suggested by (Borghol et al. 2011) and (Islam et al. 2013) seem appealing the author discovered the respective papers only after the workload generator had been implemented and used for a considerable amount of experiments already. Reimplementing the workload generator with an alternative scheme of popularity dynamics would have rendered new experiments incomparable with previous tests. For the current implementation of the workload generator the scheme suggested by (Cheng, Dale, and J. Liu 2008) is employed. Details are explained in Section 3.3.1.

### 3.2.5. TEMPORAL LOCALITY

Temporal locality is considered by only very few authors: In (Tang et al. 2003) the authors state that temporal locality for sessions within a single day is entirely determined by the file popularity distribution for that day.

(Gill et al. 2007) confirm that for their data 10% of videos watched on one day are watched on the following day as well.

(Cheng, Dale, and J. Liu 2008) employ the concept of file lifespan (cf. (Tang et al. 2003)) to model the temporal locality. However, they do not distinguish between regular and news-like lifespans. In their understanding the active lifespan denotes the period after a video does not gain any more views (or views only below a certain threshold). For every video with age $\mu$ in weeks and a growth trend factor $p$ less than 1 (see Equation (3.1)) the active lifespan, $l$, is determined by

$$l = \left(\sqrt[p]{1+t} - 1\right)^{-1} + 1 - \mu \tag{3.3}$$

where $t$ denotes the threshold. The authors find that a value of 0.05 for $t$ fits the data best.

Since the scheme by (Cheng, Dale, and J. Liu 2008) to compute popularity dynamics is used and since (Tang et al. 2003) assures that temporal locality is determined by daily file popularity, the temporal locality is not modeled explicitly by the workload generator.

### 3.2.6. PREFIX DURATION

A typical property that influences traffic for a streaming site is the distribution of prefix durations. The prefix denotes the duration between the start of a media session and the termination of the session. For the video site scenario this translates to an aborted video streaming, or an incomplete download of a video file, respectively.

Tang et al. (Tang et al. 2003) observed that the prefix duration depends on the video length: For short videos (approx. 12 minutes in that case) it followed an exponential distribution while for long videos (approx. 70 minutes in a particular example) it followed a heavy-tailed distribution. To model prefix durations the authors segmented all videos into 1-minute bins and determined the ratio of completed sessions to all sessions. A log-log plot of these values (ratio of completed sessions against bins) showed a Zipf-like distribution, but with a flattened head, however. The flatted head can be attributed to the fact that the maximum ratio of completed sessions is 0.74 (instead of 1). Furthermore, within each bin the ratios were found to be uniformly distributed. Thus, at first a maximum ratio of completed sessions, $r_b^{c_{max}}$, for each bin, $b$, is generated and for every file within a bin the prefix is drawn from a uniform distribution $U(0, r_b^{c_{max}})$. The distribution of incomplete sessions is computed as follows: First, to account for the different distributions for short and long videos a cutoff-point is defined (the authors chose 5 minutes in their scenario). Prefixes below the cutoff point are modeled with an exponential distribution while prefixes above the cutoff point are modeled by the concatenation of an exponential and a uniform distribution. $r_b^e$ denotes the ratio of incomplete sessions with prefix durations below the cutoff point, i.e. termination of the video streaming within the first 5 minutes. $r_b^u$ denotes the ratio of incomplete sessions with prefix duration above the cutoff point, i.e. termination of the video streaming at the earliest after 5 minutes. The authors noticed that $r_b^e$ varied between a lower and an upper bound that depends on the value of $r_b^c$. The upper and lower bound, $r_b^{e_{up}}$ and $r_b^{e_{lo}}$, could be computed as follows:

$$r_b^{e_{up}} = 1 - r_b^c \tag{3.4}$$

$$r_b^{e_{lo}} = r_b^{e_{up}}(0.6 + 0.4\frac{r_b^c}{r_b^{c_{max}}}) \tag{3.5}$$

To actually compute the exponential distribution for sessions that were terminated within the first five minutes, the parameter $\mu$ is needed. The authors found that $\mu$ is normally distributed, but failed to provide the respective mean and standard deviation.

Prefix duration is important to model if the server sends data in chunks as it is e.g. the case for YouTube. However, to simplify the task of workload generation it is assumed that the server sends a requested file as a whole, an approach that is e.g. implemented at Vimeo. Thus, prefix durations will not be implemented in the workload generator.

### 3.2.7. CONSIDERATIONS REGARDING A TRANSCODING SERVICE

Platforms like YouTube or Vimeo allow the upload of a wide range of formats, but present videos in a platform-default format. Thus, uploads that do not match the platform-preferred format are transparently transcoded. However, no study is known to the author that examines the distribution of different video formats of the uploads for any platform. Therefore, only assumptions can be made regarding the statistical properties of file sizes and request rate for a transcoding service.

First, in lack of any more detailed information, it is assumed for now that the distribution of sizes for files that must be transcoded is the same as for files that are already in the platform-preferred format.

Second, for the upload request rate, the authors of (Gill et al. 2007) mention that out of 100 million requests to YouTube servers only 65,000 are uploads. This means that the upload request rate is less than a thousandth of the download request rate and thus which at the same time represents an upper bound to the transcode request rate. A similar, yet not fully comparable, value is presented by YouTube itself[11]: more than six billion hours of video are watched every month (roughly $1.39 * 10^5$ hours per minute), and every minute 100 hours of video material are being uploaded which means that for every hour worth of video material uploaded, 139 hours worth of video material are being watched.

The above numbers suggest that request arrival rates to the transcoding service can be safely assumed to be at least two orders of magnitude lower than for usual download requests. This under the assumptions that a) *every* upload triggers transcoding requests, and b) *only* uploads triggers transcoding requests. The justification of the first assumption depends on the policy of the video platform (that operates the video server): Popular platforms (YouTube, Vimeo, Etc.) store video content in a platform-preferred format, i.e. video and audio codec; uploads will be automatically transcoded if they do not meet the preferred format, or the preferred resolution. Additionally, some platforms generate an additional video if the upload meets specific criteria: E.g. Vimeo generates an HD version of the upload if the video is greater than 700 x 700 pixels[12]. Thus, depending on whether the video is encoded with the platform-preferred video and audio codec, or meets or does not meet additional criteria, an upload *may* trigger a transcode. Also, depending on the target group of the video platform, users may invest less or more time to prepare a video before upload. E.g. Vimeo has built up a sophisticated audience as can be judged from the artistical, high-quality videos. One can assume that users that put much effort in creating their videos will also pay much attention to prepare their content before uploading to the platform to achieve best results. Contrary to that, YouTube's community seems less ambitious and one can expect that users of this platform care less whether their content would trigger a server-side transcode as long as the video is uploaded successfully. Since no more detailed information is available, the upper bound is assumed that all uploads trigger a transcode. For the second assumption, that only uploads may cause a transcode request, it must be noted that some video platforms present an option to watch a video in different resolutions and formats. E.g. YouTube offers videos in a variety of up to 20 combinations of resolution and format[13]. Naturally, not all combinations are readily available at server side, and thus, download requests for specific combinations will cause a live transcode. As a consequence the request rate to the transcode service can be even higher.

These assumptions and consequences hold if the transcoding service is considered as an integral component of the video platform. However, several companies already offer standalone transcoding services. Currently existing examples of such services are Brightcove's Zencoder[14], Akamai's video transcoding service[15], Encoding.com[16], or Amazon's Elastic Transcoder[17]. Since no information regarding utilization statistics are available for any of these services no additional assumptions concerning the request arrival rate can be made.

For reasons of simplification it is thus assumed that the request rate to the transcoding service is at least as the upload request rate, which in turn is approx. two orders of

[11] `http://www.youtube.com/yt/press/de/statistics.html`, 2014-09-17

[12] `http://vimeo.com/help/faq/uploading-to-vimeo/uploading-basics#can-i-upload-videos-in-hd`, 2014-10-07

[13] The extension *DownloadHelper* to the Firefox browser lists all the known combinations under → Preferences → Capture → YouTube.

[14] `http://zencoder.com/en/file-transcoding`, 2014-10-07

[15] `http://www.akamai.com/html/resources/video-transcoding.html`, 2014-10-07

[16] `http://www.encoding.com/features/`, 2014-10-07

[17] `http://aws.amazon.com/de/elastictranscoder/`, 2014-10-07

magnitude lower than the download request rate.

## 3.3. IMPLEMENTATION OF WORKLOAD GENERATION

While there are workload generators for classical HTTP traffic (`httperf`,(Mosberger and Jin 1998), `SURGE`, (Barford and M. E. Crovella 1998)), for TELNET, FTP, SMTP, and NNTP (tcplib, (Danzig and Jamin 1991)) there are no publicly available workload generators for video server traffic. (Tang et al. 2003) presents `MediSyn`, a workload for streaming media servers, but it is not publicly available. Although requests and responses to video sites employ HTTP as well both, `httperf` and `SURGE` lack several features that appear to be crucial to generate realistic video site traffic. The most apparent missing feature is the modeling of changing file popularity which is one central feature of `MediSyn`. Based on the previous analyses in Section 3.2 a custom workload generator is implemented and its details are presented hereinafter.

For reasons of better maintainability distinct workload generators are developed for the download of videos (Section 3.3.1), the transcoding of videos (Section 3.3.2) and the invocation of benchmarks (Section 3.3.3).

### 3.3.1. VIDEO SERVER

First, to repeat, the statistical properties of those components of video server traffic that will be explicitly modeled are file size, file (start) popularity, and popularity dynamics. The statistical properties of request size, request rate, and temporal locality follow from those of these three components.

#### 3.3.1.1. FILE SIZE

Regarding the file sizes it was mentioned in Section 3.2.2 that a) a median value of approx. 6.8 MB is chosen, b) the maximum file size should be between 20,000 MB and 25,000 MB, and c) the information about file size distributions for usual Internet traffic will be reemployed. In Section 3.1, file size distributions are found to be lognormal, Pareto lognormal/Pareto, truncated lognormal, or log-extreme. However, the parametrizations of the respective distributions is not reusable since file sizes are completely different from those of conventional Internet traffic. We therefore try to find a parametrization that meets the specification. At first, since most authors report a lognormal distribution, we employed the `rlnorm` function provided by GNU R's stats package. As the lognormal distribution has no closed-form CDF we cannot determine parameters for a specific quantile directly and therefore need to find a parametrization manually. Another point that aggravates the task of finding appropriate parameters is that the minimum size of a video file is by technical reasons larger than for conventional files. E.g. (M. E. Crovella, Taqqu, and Azer Bestavros 1998) report that web documents are typically in the range of 256 to 512 Bytes, but that there are also virtually no documents smaller than 100 Bytes. In (Zink et al. 2009) the authors report minimum file sizes between 452 Bytes and 95760 Bytes, depending on the time period under consideration. We searched YouTube for the shortest (length $\leq 1$ second) and smallest (preferrably entirely black, no audio) video. The smallest video we find is 10.7 kB [18].

To actually determine appropriate choices for $\mu$ and $\sigma$ of the `rlnorm` function we performed an iterative search using varying values of $\mu$ and $\sigma$. For every parameter setting we called `rlnorm` 20 times, multiplied the values with $1024^2$ and rounded the results in

[18] `http://youtu.be/7-qGKqveZaM`, 2014-10-14.

order to be interpretable as bytes, added an offset of $10700$, and then checked whether the generated values meet, on average, the requirements for the median and maximum values. We find that $\mu = 1.8$ and $\sigma = 2.1$ produce the best results. For better comprehensibility, Listing 3.1 shows the statistics of the generated values that are used in the later experiments, and Figure 3.2 depicts the EDF of the resulting data.

```
>>> vidsizes.describe()
count     5.000000e+03
mean      9.828375e+07
std       5.350871e+08
min       1.292300e+04
25%       1.881168e+06
50%       8.528066e+06
75%       3.870591e+07
max       2.055312e+10
```

Listing 3.1: Statistics of the lognormally distributed file sizes.

The number of random values, 5,000 in our case, is originally determined by the maximum storage capacity.

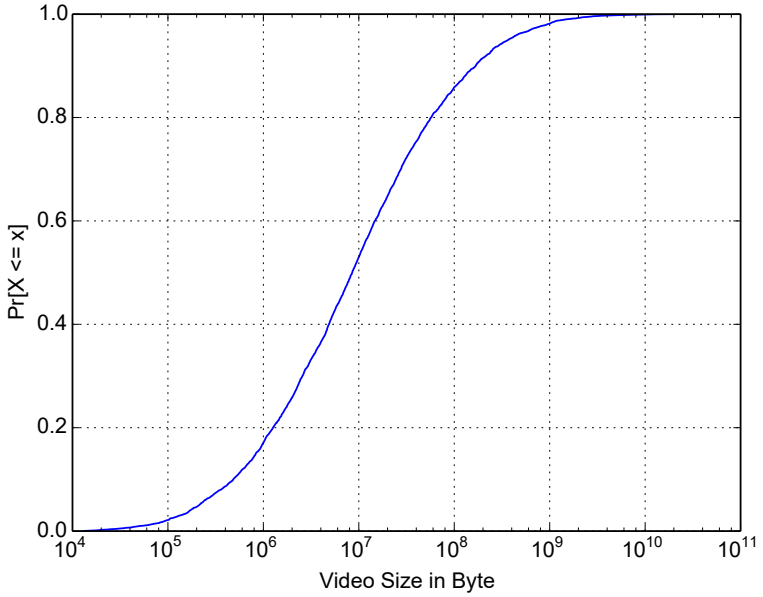



Figure 3.2.: EDF of the generated file sizes. Note that the x-axis is in log scale.

#### 3.3.1.2. FILE POPULARITY

As mentioned in Section 3.2.3 we will use the original measurement data for Dailymotion provided by the authors of (Mitra et al. 2011) to obtain file popularity variates. Listing 3.2, Figure 3.5a, and Figure 3.4 show the statistics, CCDF, and rankview plot of the original data, respectively. While the number of file size variates is determined by storage capacity – which could be increased – a more restricting bound is implied by the maximum network bandwidth: The workload generator will generate requests for every video that is available at the server. Thus, the more videos are available at server side the higher the request rate. See Section 3.3.1.4 for details regarding request generation. During initial experiments we find that network bandwidth becomes saturated if 2,000 videos reside on the server. We therefore uniformly sample the original measurement data to obtain a subset of 2,000 file

popularity variates. Listing 3.3 and Figure 3.5b show the respective statistics and CCDF, respectively.

```
count      1194185.000000
mean          1502.941747
std          12188.925169
min              0.000000
25%             71.000000
50%            210.000000
75%            679.000000
max        2895396.000000
```

Listing (3.2) Statistics of the original measurement data.

```
count       2000.00000
mean        1526.64900
std         8911.22964
min            0.00000
25%           72.00000
50%          207.50000
75%          697.50000
max       249188.00000
```

Listing (3.3) Statistics of the 2000 sample subset.

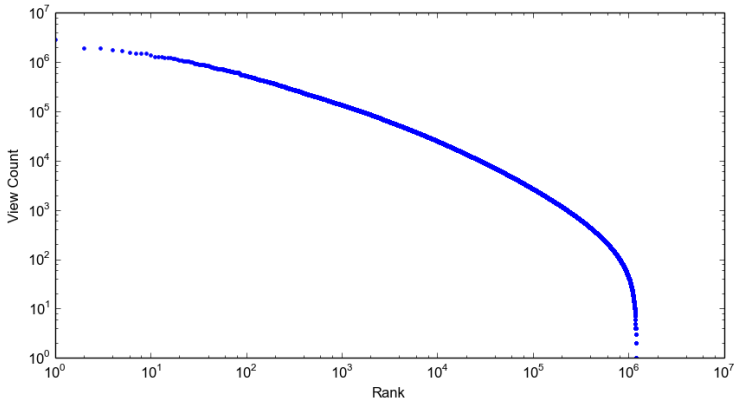
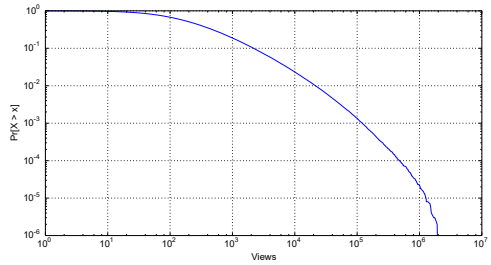
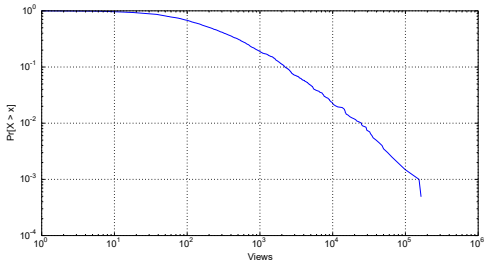



Figure 3.4.: Rank/view plot (ranked distribution) of the generated start popularities. Note that the plot is in log-log scale.




(a) File popularity CCDF for the original measurement data.




(b) File popularity CCDF for the 2,000 samples subset.

Figure 3.5.: CCDFs for the original and sampled data.

#### 3.3.1.3. POPULARITY DYNAMICS

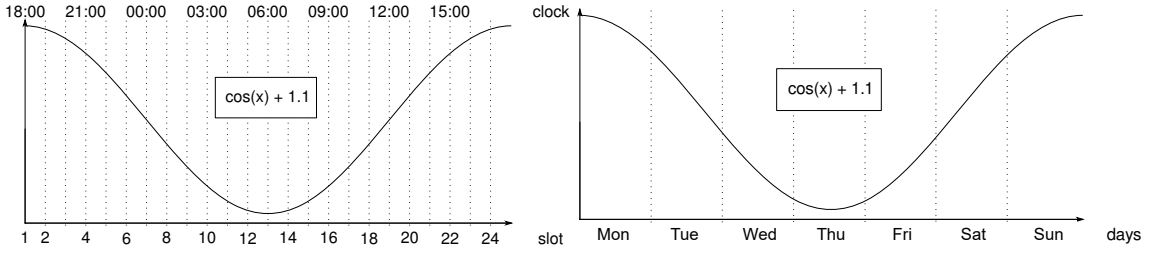
Building upon the start popularity values we then calculate additional views that every video gains using Equation (3.1): $[v(x) = v_0 \cdot \frac{(x+\mu)^p}{\mu^p}]$ where $v_0$ is the start popularity obtained by the previous step. As explained in Section 3.2.4, $p$ follows a Weibull distribution $\mathcal{W}(2, 0.9)$, and the upload trend follows a power law with $\alpha = 2.61$. To determine the age, $\mu$, we proceed in two steps: First, we calculate the age of the oldest video. For reasons of

simplicity we assume a) that the oldest video on the server was the first video that was uploaded and that later uploads obey the upload trend, b) that in the first week only one video was uploaded. Thus, starting at week $w = 1$, we increment $w$ for every subsequent week and the number of videos uploaded is given by $\lceil w^{\alpha} \rceil$. We proceed until the cumulative amount of uploaded videos is $\geq$ 2,000. The last value of $w$ is the age of the oldest video. Then, in the second step, we assign the first video the age $w$, the next $2^{\alpha}$ videos are $w - 1$ weeks old, and so on. For 2,000 videos the first video is 13 weeks old. After this step we are able calculate the number of additional views for every video for the following week. How these additional views are distributed over the days of the week and hours of every day is explained subsequently.

#### 3.3.1.4. REQUEST GENERATION

The request generation starts by reading the external file of file size variates and generating a file name for every file entry. The file name is just file size as a string preceeded by a counter to ensure that multiple occurences of the same file size result in a different file name. The same approach has been applied before at the server side such that file names requested by the workload generator match file names at the server.

In Section 3.2.1 we note that diurnal patterns of request arrivals have a wave-like shape and that the same holds for the change of request arrivals between the days of the week (weekly pattern). We find that cosine waves are good approximations for the diurnal pattern and for the weekly pattern in most of the cases. To account for the fact that even at times with the lowest request rate the request rate is not zero we add an offset of 1.1 to shift the cosine curve along the y-axis. For the cosine curve that describes the weekly pattern we additionally flatten the curve such that the maximum request rate is only approx. 1.5 times the minimum request rate (hours of day fixed) since changes in request rates between weekdays are not as pronounced as changes between day and night. In Figure 3.6 we show the respective cosines.




(a) Diurnal pattern (b) Weekly pattern

Figure 3.6.: Cosines that describe the weekly and diurnal pattern.

Note specifically in Figure 3.6a the mapping between time and slots: We decide that the request rate reaches its peak around 18:00 o'clock and its bottom in the morning hours. Similarly for the weekly pattern, we decide that the most requests arrive on Monday and Sunday and that Thursday the fewest amount of requests are generated. Indeed, the choice for Monday and Sunday as load peaks is arbitrary. It is merely important to reflect the weekly pattern in the generation of workload. On the implemenation side every weekday is represented by a number and thus, every other number (or weekday) can be selected to represent the weekly load peak. Note that any other mapping of times and days to the cosines could be implemented as well. For the generation of a realistic workload it is only important to model the nonstationarity of request rate - *when* exactly (regarding time and day) the request rate reaches its peak and bottom is only of minor concern.

We distribute the view gain for every video initially across weekdays according to the

share of the area under the cosine curve describing the weekly pattern. Thus, we calculate a day-specific ("daily") view gain for every view. In a second step we further distribute the daily view gain for every day across hours according to the share of the cosine curve that describes the diurnal pattern. Knowing the amount of requests for every video per hour we then generate exponentially distributed interarrival times (IATs) for every video. Lastly, we merge the interarrival times of all videos to obtain a *request sequence* for every respective hour. During the merging we first compute intermediate sums of the interarrival times for every video, sort all aggregated sums for all videos, and compute the differences between every contiguous pair of aggregate sums. These differences are the true interarrival times for the set of all videos per slot. The request sequence is then merely a list of IATs, each associated with one or more videos.[19] In Figure A.1 we depict the just described procedure with an example: For three videos (red, green, blue) interarrival times for Tuesday, 18:00 to 19:00 o'clock are generated.

The request sequence is finally processed in a way such that the workload generator reads the first list entry, puts its process to sleep according to the interarrival time, and sends request(s) for the (all) video(s) associated with the IAT after it wakes up. Figure A.2 summarizes the entire process of request generation in a behavior diagram.

### 3.3.2. TRANSCODER

The workload generator for the transcoding service is a derivative of the workload generator for the video server: It employs the same weekly and diurnal patterns, and issues requests in the same way. Differences exist with respect to the file size distribution, the file popularity and popularity dynamics, whereof all will be explained in the following sections. To the best of our knowledge, the literature does not yet cover statistical properties of video transcoding services. Therefore, our decisions concerning the statistical properties of these workload components are not supported by any scientific publication, but must solely rely on rational considerations. On the other hand, this offers some degree of freedom since we are in the position of a developer of a transcoding service, can therefore determine said properties and thus, also the nature of the workload generator.

#### 3.3.2.1. FILE SIZE

From a user's perspective it seems only natural to assume that the statistical properties for the file size distribution also apply to the file size distribution of the transcoding service. The rationale behind is that users who want to upload files to a video platform may use a (professional) transcoding service in the preprocessing stage. However, for the service we use in our experiments we enforce a limit on the maximum upload size for practical reasons: Users upload the file to the server via their browser. The browser keeps waiting for the response (i.e. the transcoded file) until a timeout is reached. While it is in principle possible to increase the timeout since *we* have control over the browser code, we believe that a larger value than the default 600 seconds is not realistic since *other* users may have not such control. Other solutions could be that a) users get notified on a finished transcoding via a second channel (e.g. e-mail), or b) that users are provided a unique link that can be repeatedly checked and will eventually lead to the transcoded file. Both alternatives are not well suited for automated tests and are therefore ruled out. For the transcoding scenario we limit the maximum file size to 100 MB. Similarly to the video server scenario we store the generated file size variates in a static external file. The difference between the original distribution of file sizes and the modified distribution are depicted in Figure 3.7 where we

[19] Since IATs are generated individually for every video the merging can lead to a situation that at some specific time point, i.e. aggregated sum, more than video will be requested.

compare the EDFs of both data sets. Additionally, Listing 3.4 summarizes the statistics of the modified distribution.

```
>>> vidsizes_lim = vidsizes[vidsizes <= 1e+8]
>>> vidsizes_lim.describe()
count         4292.000000
mean      15107382.525163
std       21221835.339611
min          12923.000000
25%        1449008.750000
50%        5667256.500000
75%       19505988.250000
max       99827206.000000
```

Listing 3.4: Statistics of the modified file size distribution.

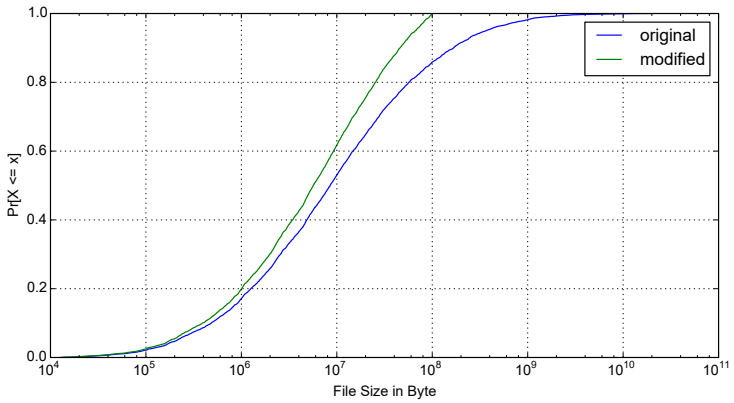



Figure 3.7.: Comparison of the EDFs for the original file size distribution and the modified distribution used for the transcoding scenario.

#### 3.3.2.2. FILE POPULARITY AND POPULARITY DYNAMICS

Start popularity and popularity dynamics are no meaningful concepts in the transcoding scenario: While one can argue that users may want a file to be transcoded to several target formats, such a request does not depend on the file, but rather on the user's preferences to which video platform the file should be uploaded and what the respective preferred formats are.

To select a file for transcoding we process the external file of file sizes in sequential order. For every request that is generated we pick the next entry in the file. If the end of the file is reached (after 4292 requests) we start over with the first entry.

#### 3.3.2.3. REQUEST RATE

Since the start popularity and popularity dynamics determine the request arrival rate in the video server scenario, we need to determine this workload component in a different way. As explained in Section 3.2.7, we assume that the request arrival rate in the transcoding scenario is approximately two orders of magnitude lower than the request arrival in the video server scenario. Preliminary experiments showed that the maximum request rate in the video server scenario is approx. 12 requests per second. We therefore parameterize the workload generator initially such that a request rate of approx. 0.1 is produced.

#### 3.3.2.4. TARGET CONTAINER

One motivation for the consideration of a transcoding application are the different platform-preferred container formats of the various video platforms. The actual target container influences greatly the performance of a transcoder and is therefore added as another parameter to the workload generator. The most widely adopted target containers are FLV (e.g. YouTube) and MP4 (e.g. YouTube, Vimeo, Dailymotion), where FLV typically contains the FLV1 video codec and the MPGA audio codec, and MP4 contains the H.264 video codec and the MPEG AAC audio codec. Lately, since the advent of HTML5, a new container format, WEBM, is becoming popular as a free alternative to FLV and MP4. WEBM typically comprises either the VP8 video and Vorbis audio codec or VP9 and Opus codecs, respectively, depending on the implementation of the encoder library. Consequently, we chose these containers with the respective video and audio codecs as targets for the transcoder.

### 3.3.3. BENCHMARKS

Additionally to the video server and transcoding service scenarios we include benchmarks from the SPEC CPU2006 suite[20] in our set of considered applications. Since the video server is mostly I/O-bound and the transcoding service is I/O and CPU-bound we expect to observe resource interference between the video server and the transcoding service due to shared access to the network and storage resources. To induce resource interference for the transcoding service due to shared access to the CPU we therefore employ benchmark applications from the SPEC CPU2006 suite. The suite comprises 12 integer and 17 floating point benchmarks.

In the video server and transcoding scenarios we assume that request arrival rates are stationary for periods of one hour. The respective workload generators are thus designed to change the request rate not more often. To obtain meaningful and comparable results when benchmarks are run in parallel the respective testruns must also run for an hour. In the transcoding scenario we restrict ourselves to two target container formats and two different request rates, resulting in four different test sets running, each taking one hour to complete. Since the entire SPEC CPU2006 suite comprises 29 benchmarks this would result in 16 days of non-stop experiments. Thus, to reduce the occupation of shared lab equipment we need to reduce the number of benchmarks used in the experiments.

It is indeed not necessary to test against the entire benchmark suite since benchmarks are similiar to each other up to some degree and thus enable subsetting (i.e. clustering). Subsets allow to use only selected representatives of the suite, and yet to obtain results for the entire test range of the suite. Especially a team of authors around Lizy Kurian John has thoroughly investigated the similiarities between benchmark programs of the SPEC CPU2000 and CPU2006 suites, respectively (Joshi et al. 2006; Phansalkar, Joshi, and Lizy K. John 2007a; Phansalkar, Joshi, and Lizy K. John 2007b). They analyze which performance monitoring events (PMEs) of the CINT2006 and of the CFP2006 (the integer and floating point sets, respectively, of the CPU2006 suite) are most descriptive in order to distinguish between the single benchmarks. For both sets they list six criteria which are summarized in Table 3.2 below.

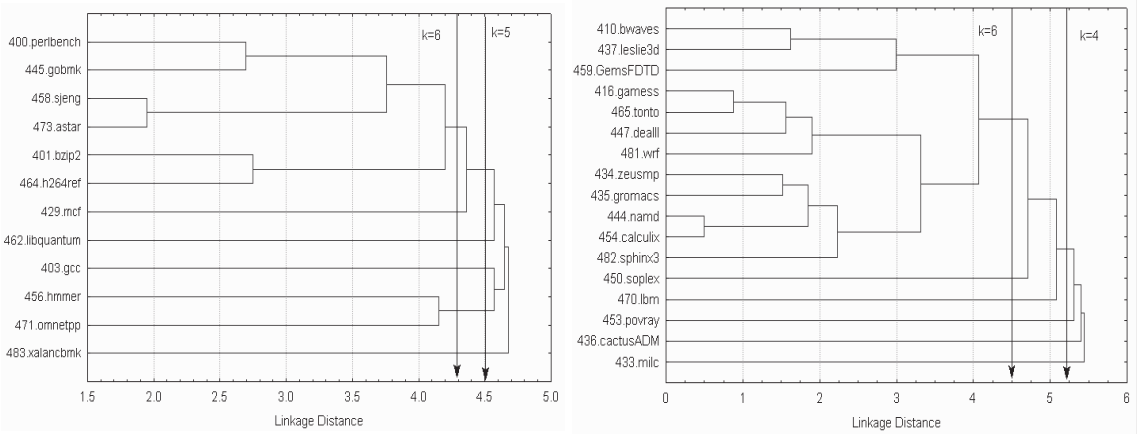
The benchmarks of the SPEC CPU2006 suite are executed on five different machines with varying microarchitectures employing also different compilers. Principal component analysis is performed to eliminate correlation due to similarities in some microarchitectural aspects. Then, hierarchical clustering is employed to obtain dendrograms for the INT and FP benchmark sets. Consulting the dendrograms in Figure 3.8 one can pick the first $n$

[20] http://www.spec.org/cpu2006/

| CINT2006 | CFP2006 |
|---|---|
| integer ops PI | floating point ops PI |
| L1-I misses PI | memory references PI |
| number of branches PI | L2-D misses PI |
| mispredicted branches PI | L2-D misses per L2 access |
| L2-D misses PI | data TLB misses PI |
| ITLB misses PI | L1-D misses PI |

Table 3.2.: Program characteristics of the CINT2006 and CFP2006 benchmarks (Phansalkar, Joshi, and Lizy K. John 2007b). The abbreviation PI denotes *per instruction*.

benchmarks that are most dissimilar to each other.




(a) SPEC CINT2006

(b) SPEC CFP2006

Figure 3.8.: Dendrograms for the INT and FP parts of the SPEC CPU2006 suites showing the linkage distance between the respective benchmarks (Phansalkar, Joshi, and Lizy K. John 2007b).

In this way one can select a few benchmarks that represent a broad range of the entire suite. We choose $n = 3$ to select the first three benchmarks in the INT and FP set, respectively. This reduces the time for a complete run of the transcoder and benchmarks in parallel to 24 hours. Table Table 3.3 summarizes the selected benchmarks that are used in our experiments.

While for the video server and the transcoding service the request size component of the workload is determined by the size of the requested file, the notion or understanding of request size must be altered for the benchmark scenario: Both previous scenarios describe situations where a (favorably infinitely) long running application like a web server is idling on the server until a request arrives. Only then the application process gets active. The associated runtime or computation time depends on the size of the very request (and pending requests, if applicable, that are processed in parallel). In the benchmark scenario the application itself is started only once a request arrives, and after finishing the computation the application terminates. While this is still comparable to the previous scenarios to some extent, users have only little control over the request size; they can choose merely between three workload sizes *test*, *train*, and *ref*, where the meaning of *size* depends on the respective benchmark. E.g. in case of `403.gcc` it is the number of files to compile, while in case of `462.libquantum` it is the number to be factorized together

| CINT2006 | CFP2006 |
|---|---|
| 403.gcc | 436.cactusADM |
| 462.libquantum | 433.milc |
| 483.xalancbmk | 453.povray |

Table 3.3.: Selected benchmarks from the INT and FP set of SPEC CPU2006 suite. For a description of the respective benchmarks see (Henning 2006).

with the module, or in `433.milc` it is the gridsize. As the name suggests, the `test` size is intended for testing purposes and generates negligible or only little load on the server depending on the benchmark. The `train` set of workload sizes is intended to be used in feedback-directed optimization where the compiler optimizes code depending on a previous run of the application. `ref` is the reference workload size and is intended to be used in normal runs of the benchmark applications. The differences between `train` and `ref` sizes thus only allow to understand both sizes as "middle" (or "small") and "large(r)" workload size. More fine grained distinctions between sizes are not available.

We decide to use the `ref` workload size and control the workload only through the request rate, following methodologies as in (Govindan et al. 2011) or (Delimitrou and Kozyrakis 2013). For the request size we choose two different scenarios with a *high* and a *low* request rate. For the *high* request rate we decide to issue a request every minute. This is motivated by our observation that the server becomes saturated at this request rate for the `436.cactusADM` and `433.milc` benchmarks: In case of `436.cactusADM` the maximum 1-minute load average is 29.26, and in case of `433.milc` the maximum 1-minute load average is 32.33. In the *low* scenario the request rate is fixed with a request every 2 minutes. While this is only the half of the *high* request rate the load average reduces to less than the half: For `436.cactusADM` and `433.milc` the maximum 1-minute load averages are 11.11 and 10.5, respectively. For the remaining benchmarks the maximum 1-minute load average is always below 6.5. which is only a fifth of the system capacity.

## 3.4. SUMMARY

This chapter presents the theoretical foundation and implemention of the workload generator for the scenarios of a video server and a transcoding service. It first gives an overview of statistical properties of traffic for classical web servers. Subsequently, it provides an extensive analysis of the specific statistical properties of video server traffic. Since related work for the transcoder scenario is non-existent, only coarse boundaries regarding the statistical properties are presented. The results of these examinations form the basis for the implemention of the workload generators for both scenarios which are explained thereupon. Additionally, it describes properties of a workload generator that issues requests to CPU-intensive applications which are represented by a subset of the SPEC CPU2006 suite.

All models are wrong.

George Box

# 4. MODELS FOR RESOURCE UTILIZATION AND FOR POWER CONSUMPTION

## 4.1. INTRODUCTION

In the this chapter we develop models for the resource utilization as function of the generated workload as described in the previous chapter. Also, we will develop models for the power consumption as function of resource utilization.

Explanations in this chapter are partially taken from (Möbius, Dargie, and Schill 2013) and (Möbius and Dargie 2014).

## 4.2. PRIOR WORK

### 4.2.1. MODELS FOR RESOURCE UTILIZATION

(Friedrich, Hollfelder, and Aberer 2000) employ a hidden Markov model (HMM) to predict the probability of system overload in a multimedia server context and to enforce an admission control scheme based on model outcome. Clients of the server are modelled to belong to an application class and each class comprises a finite number of states. States represent different "presentation modes", i.e. whether the client requests audio or video data with varying codecs and data rates. Classes represent clients with similar (or identical) request rates, state transition behavior, and state dwell time. While the model predicts possible server overutilization it represents CPU utilization only indirectly through the ratio of requests that are served within their deadline. Other resources are not considered. Furthermore, the results are obtained solely via simulation. We will also consider the storage and network, and apply realistic workload.

(Wood, Cherkasova, et al. 2008) aim to estimate the additional CPU usage of applications when they are transferred from a native to a virtualized execution environment. They develop three benchmarks to simulate CPU-intensive, network-intensive, and storage-intensive applications. All three benchmarks are run on the native and virtualized platform and resource utilization metrics are recorded for 30 second intervals. For the CPU-intensive

benchmark they record the CPU utilization in the different modes (user, system, wait)[1]. For the network-intensive benchmark the packets as well as bytes transmitted and received per second are employed. Read and write requests per second as well as read and written blocks per second are the relevant metrics for the storage-intensive benchmark. In sum, they sample 11 metrics. The model that expresses the CPU utilization of the virtual machine monitor (VMM), $dom0$, due to virtualized I/O operations is then of the form:

$$U^j_{dom0} = c_0 + c_1 * M^j_1 + c_2 * M^j_2 + \ldots + c_{11} * M^j_{11} \tag{4.1}$$

where $j$ denotes the time interval and $M^j_i$ denotes the recorded value of metric $i$ in time interval $j$. The coefficients $c_i$ are determined using iterative reweighted least squares regression (Holland and Welsch 1977) to reduce the impact of outliers, and regression is done stepwise to select only the most relevant predictors. Since the authors use solely non-standard benchmarks the approach is hardly comparable. Also, it is not clear whether metrics are read at the end of the interval or averaged over interval.

(Q. Zhu, J. Zhu, and Agrawal 2010) focus on workflows of scientific applications. They employ a HMM to predict time series of the utilization for all resources of the server. In the model, tuples of resource utilization values at time step $t$, $q_t = \langle cpu_t, mem_t, disk_t, net_t \rangle$, represent the hidden states, and observations are tuples of the form $\langle para_t, resource_t \rangle$ where $para_t$ denotes application parameters[2] and $resource_t$ a tuple of the available capacity of all resources. To train the model, the applications are executed to obtain sequences of observations and (hidden) states. The parameters of the HMM are learned using the Forward-Backward algorithm. The authors assure that the error of their model is within 4.6% for the cases of CPU utlization and memory utilization (no statements regarding utilization of the other two resources are provided). The assumptions the authors make are only valid for applications with short-termed runtimes. Additionally, they model resource utilization not as a function of the workload.

(Khan and Anerousis 2012) present an approach for cloud providers to predict CPU utilization of VMs by exploiting relations between CPU utilization of VMs despite the relations between VM instances are transparent to the cloud provider. Time series of CPU utilization are obtained with samples taken every 15 minutes and CPU utilization is discretized to 5 levels (each 20 % utilization range). These time series are then clustered to find groups of VMs with similar behavior. Correlation between clusters or autocorrelation within the same timeseries is afterwards employed to determine whether a timeseries (i.e. the CPU utilization of a VM) is *predictable*, where the prediction horizon is 1 lag (i.e. 15 minutes). In case of autocorrelation the timeseries is *self-predictable*. And if there is a self-predictable time series $C_1$ and another timeseries $C_2$, then $C_2$ is called *predictive* if the time-lagged cross-correlation, $R_{C_1,C_2}$, defined as

$$R_{C_1,C_2} = \sum_{t_1}^{n-1} \frac{(c_{1,t} - \overline{c_1})(c_{2,t+1} - \overline{c_2})}{n\sigma_1\sigma_2} \tag{4.2}$$

is above a certain threshold, where $\overline{c_1}$, $\overline{c_2}$, $\sigma_1$, $\sigma_2$ are the mean and standard deviation of $C_1$ and $C_2$, respectively and $R_{C_1,C_2}$. In the following step, self-predictable and associated predictive timeseries are combined to *prediction groups*. Now, for every prediction group, a HMM is applied to predict the workload variation – the authors sampled data over 17 days

[1] Unix-like systems distinguish between six different modes of CPU utilization: *user* mode (`usr`) where user applications are executed, *system* mode (`sys`) where the CPU executes operating system functions, *wait* mode (`wai`) where the CPU waits for I/O devices like network interface card (NIC) and storage, *idle* mode (`idl`) where the CPU does nothing, and two modes where the CPU handles software and hardware interrupts (`siq`, `hiq`). The system mode is also called *kernel* mode or Ring 0.

[2] This can be understood as some kind of request size definition.

to predict 4 days. This approach seems applicable only if VMs show repeating resource utilization patterns. Also, only 59% of all VMs were predictable and only levels (20% range) of CPU utilization are predicted which we consider too coarse grained. Finally, the authors state that errors are between 5 % and 40 % depending on the actual level.

In (Desnoyers et al. 2012) the authors suggest an approach to automatically derive models for complex web applications. In particular, they use queueing networks to build up Workload–Resource Utilization models: Every component in the web application is represented as a $M/G/1/PS$ queue, meaning a Markovian (i.e. Poisson) request arrival process, an arbitrary service time process, and $1$ processing unit with processor sharing. It is assumed that the requests to the system can be partitioned into $k$ request classes that represent requests with *similar* service times. If $s_i$ and $\lambda_i$ denote the mean service time and request arrival time for request class $i$, respectively, then the aggregate CPU utilization, $\mu$, for all request classes is formulated as

$$\mu = \lambda_i \cdot s_i + \lambda_2 \cdot s_2 + \ldots + \lambda_k \cdot s_k + \epsilon \tag{4.3}$$

where $\epsilon$ denotes an independent and random error. To decide on request classes, with every request the authors "enumerate" a set of possible features that are mainly determined by the (in)formal syntax of the request strings (e.g. (parts of) an HTTP URL, or (parts of) an SQL query). In parallel, CPU utilization and request arrival rate are recorded. To derive the model, stepwise regression is applied to find the most significant features for predicting CPU utilization. After the classes have been determined, the mean service $s_i$ time is computed by means of linear regression. While the authors assert that the approach yields very small error margins (e.g. the $0.9$ quantile of the prediction error for the CPU utilization is $\leq 2.5\%$ in case of the Apache web server and $\leq 5\%$ in case of a MySQL DB server) they admit that the prediction error depends linearly on the coefficient of variation, $c_v$, of the service time: While for $c_v = 0.5$ the error is below $2\%$ it approaches 10% with $c_v = 2$. In our datasets, $c_v$ for service time is $> 7$ which would result, as Desnoyers and colleagues state, in either longer training windows or lower accuracy, or both.

(Brinke et al. 2013) present a resource utilization model for software components that is essentially an enhanced state chart: States represent different levels of power consumption and transitions represent component invocation or internal events that lead to changes in the component's power consumption. Resource utilization is thus "an abstraction of the energy behavior of the [..] component". The transition system enables a formal analysis and verification (i.e. model checking) of properties (e.g. the component always consumes less than 1 J/s on every execution path). However, it is not clear from where and how the ground truth for the components' power consumption is derived. Also, the approach assumes that the concrete application comprises at least three software components: application, optimizer, and a low-level (hardware) component which is not the case in our testbed.

(Brunnert, Vögele, and Krcmar 2013) describe a scheme to derive performance models for single EJBs, servlets and JSPs of a Java web application. In a first step – during exection of an application – a component model is built that collects component names, names of public methods, relationships between components based on calls to public methods, and the resource demand (i.e. CPU demand in $ms$) for each method call. To reflect different component behavior, so called resource demanding service effect (RDSEFF) specifications are generated that assign probabilities to different execution paths ("probability branches") based on the collected data. Also based on the collected data, the mean CPU demand in $ms$ is assigned to every probability branch. This method requires full execution of the entire application and assumes finite execution time of components; it is in particular not appropriate for server applications with conveniently infinite run time like the ones we employ in our testbed.

(Delimitrou and Kozyrakis 2014) develop an interference-aware data center scheduler that employs collaborative filtering to determine an application's resource demand with respect to scale-up (more server resources), scale-out (more servers), heterogeneity, and interference. First, for scale-up classification, in a preliminary offline step different applications (∼20) are profiled on the most powerful server with many combinations of resource allocations ("server configurations", $sc$) and with various amounts of interference and the performance metrics (e.g. queries per second of a web server) are recorded in a *utility matrix*, $A$, with one row per application and one column per server configuration. Later, in the running system a newly submitted application is profiled against *two* server configurations for 10 seconds up to 5 minutes depending on the application and the performance values are added to $A$. To recover the missing matrix entries for the other server configurations, $A$ is first factorized using singular value decomposition (SVD): $A_{app,sc} = U \cdot \Sigma \cdot V^T$. Then, the following matrices are initialized: $Q = U$, $P^T = \Sigma V^T$ and a matrix $R$ is defined as $R = QP^T$. In the next step, the matrix $R$ is built using PQ reconstruction with stochastic gradient descent (Bottou 2010) during which process the missing entries in $A$ are estimated. For scale-out classification, classification for heterogeneity and for interference the process works similar: In case of scale-out the applications are profiled on up to hundred similar machines in the offline step and the columns of $A$ are multiples of servers. For heterogeneity, applications are profiled on different server platforms. And for interference, a few handcrafted micro benchmarks are employed during scale-up classification to induce interference at a few out of 15 so-called sources of interference (SoIs) (see (Delimitrou and Kozyrakis 2013)): The intensities of the micro benchmarks are increased until the application suffers performance degradation of 5% of the user-provided performance goal. The columns of $A$ are the SoIs in this case. To detect changes in workload the authors proactively induce interference at some SoIs for some applications and reclassify repeat interference classification for those applications where the measurements significantly deviate from previous classification results. The authors assure that 78% of load changes are detected within 10 minutes with 8% false positives. While it seems a very interesting and promising approach it is not meaningful in our situation since in our testbed only one server exists. Generally, as the authors mention, micro benchmarks – to induce interference at specific SoIs – need to be handcrafted for every CPU architecture and model. Indeed many PMEs the authors employ are – in Intel's terminology – so-called *non-architectural events*. That means that availability and consistent semantics are not guaranteed across CPU models (Intel Corporation 2012) which renders the approach hard to port to other systems.

Summarizing, a broad range of approaches to model resource utilization exists. But no team of authors attempts to model the resource utilization of applications by means of the workload they need to process. We believe that models of this kind would form a building block in a different approach to assess the resource interference between applications: In cloud data centers, both, workloads and resource utilization are continuously monitored to enable resource management. Therefore, an application's actual resource utilization could be continuously compared against the model and possible interference could be detected readily since the model already accounts for changes in workload.

### 4.2.2. MODELS FOR POWER CONSUMPTION

Power consumption can be regarded as a fifth resource additional to CPU, memory, storage, and network. Since energy efficiency has been a topic in the research community for a considerable amount of time a large body of work on modeling the power consumption of servers and server components already exists. Trivially, all modeling approaches necessitate the availability of reference values through measurements. In the following we will briefly

present different means to obtain measurement data, directly or indirectly, for the power consumption of a server's subcomponents.

#### 4.2.2.1. MEASUREMENT

With direct measurements the electric lines supplying power to the system are intercepted and either power measuring devices (Allalouf et al. 2009; Molka et al. 2010; Du Bois et al. 2011) or shunt resistors ($< 100m\Omega$) (Bircher and Lizy Kurian John 2007) are inserted. In an indirect setting, lossless inductors are attached to the power lines to measure the electromagnetic fields created by the current that is drawn by the system and the induction law is applied to assess the actual current (Bircher, Valluri, et al. 2005; Kusic et al. 2008; Abbasi, Varsamopoulos, and S. K. S. Gupta 2010; Dhiman, Mihic, and Rosing 2010; W. Lang and Patel 2010). Indirect approaches are in particular sensitive to electromagnetic interference, a phenomenon we cannot guarantee to exclude in our lab environment. Therefore, indirect measurement approaches are not further considered.

(Economou et al. 2006) develop custom hardware to measure the power consumption of the subsystems of a blade server. They use four power rails to carry out the measurements: a 12V line that is predominantly used by the processor and memory; a 5V line that supplies power to the disk drive; and a 5V auxiliary line and a 3.3V line supplying power to the remaining components. The 12V line is branched to provide the processor and the memory subsystems with power. The authors report they modified the motherboard's network structure to insert an extra shunt resistor, but do not provide information regarding the modification process. Nor do they describe where exactly the shunt resistor is added. Furthermore, details are missing pertaining to the employed measurement devices and the acquisition board.

Bircher et al. (Bircher and Lizy Kurian John 2007) measure the power consumption of an IBM x440 server using on-board current-sensing resistors[3] that are originally used by the built-in server management system for over-current protection. They identify five such resistors for the CPU, the memory, the Southbridge, the disk drives, and the chipset. Measuring the voltage drop across these resistors enables to compute the power consumption of the respective subsystems. To reduce the effect of noise and to improve the sampling resolution, the authors developed an amplifier board and amplified the voltage drops across the shunt resistors. The amplifier outputs are then quantized by a National Instruments AT MIO-16E-2 card, which is able to sample 8 channels in parallel at 10KHz. Pre-instrumented server hardware as in this case simplifies the measurement task. However, our server architecture is not equipped with such ready-to-use measurement points which renders this approach inapplicable.

(Dargie and Schill 2012) investigate the power consumption of the various subsystems of a multimedia server (Fujitsu-Siemens D2641) cluster under different load balancing policies. They first investigate how the voltage regulators of the different subsystems are supplied with DC power and observe that the main voltage regulator that generates the processor core voltage predominantly uses a separate 12V line while the voltage regulator of the memory unit predominantly uses a 5V line. Likewise, the voltage regulator of the Southbridge uses a 12V line. All the I/O controllers draw current through the 3.3V line. Unfortunately, some of the power rails supply multiple voltage regulators with power while some of the voltage regulators use multiple rails as their input which complicates the measurement task. For the components that are not accessible – for example, the NIC which is connected to the PCI Express slot – the authors provide a custom-made raiser board and intercepted the 3.3V power line of the raiser board to measure the power consumption. The authors analyze the relationship between the probability distribution

[3] Most voltage regulators provide a lossless current sensing inductor to enable overcurrent protection.

functions of the workload of the server and the power drawn through the various power rails to estimate the power consumption of the various subsystems. Due to limitations in the number of digital power analyzers we cannot port this approach to our server.

As of the Sandy Bridge microarchitecure, Intel CPUs offer a PME called running average power limit (RAPL) to obtain power consumption for e.g. the entire CPU, or the cores of the CPU. While in principle being an attractive means to read the CPU's power consumption the values reported by RAPL diverge strongly from true power consumption if HyperThreading is enabled (Hackenberg et al. 2013).

Therefore, we will take the *wall power* as reference power consumption, i.e. we measure the servers' total power consumption using a digital power analyzer between the outlet and the server's power supply unit (PSU).

#### 4.2.2.2. MODELING

(Dhiman, Mihic, and Rosing 2010) employ CPU utilization and additional PMEs along with a Gaussian mixture model (GMM) to estimate the power consumption of a virtual machine. The input values form a vector $x = (x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{pwr})$ which are assigned to different classes (or $bins$), where $ipc$ denotes the instructions per cycle, $mpc$ denotes the memory transactions per cycle, $ctpc$ denotes the cache transactions per cylce, $util$ denotes the CPU utilization, and $pwr$ denotes the active power consumption (i.e. the dynamic pwoer consumption attributable to executing the workload, in particular without static power consumption if the system is idle). Depending on the CPU utilization, bin sizes range from 20% utilization (5 bins) to 100% CPU utilization (1 bin). Within every bin the data vectors are quantized using a Gaussian Mixture Vector Quantization (GMVQ) technique (Gray 2001). The output of this quantization are multiple Gaussian components $g_i$ of the GMM that are determined by their mean $m_i = \{m_{ipc}, m_{mtc}, m_{ctpc}, m_{util}, m_{pwr}\}$, covariance $K_i(x_{ipc}, x_{mpc}, x_{ctpc}, x_{util}, x_{pwr})$, and probability $p_i$. To train the model, the $x_{pwr}$ component is removed from the $x$ vectors and the resulting vectors are compared with those in the training set. A GMVQ algorithm then finds the nearest vector $m$ in the training set to the input vector, and the value for the $m_{pwr}$ component is retrieved. The retrieved value is compared to the actual $x_{pwr}$ value from which the accuracy of the model is determined. This part of the training phase is repeated multiple times with different sizes of utilization bins. The bin size resulting in the smallest error is selected as the model parameter. The same method is applied during the running phase: Performance monitoring counter (PMC) values are obtained and the vector is compared to the model where the GMVQ algorithm finds the nearest (training) vector and returns the average power value. Depending on the utilization level, the average error of the model is between 11% (for 50% CPU utilization) and 8% (for 100% CPU utilization). The error additionally depends on the size if memory since memory power consumption is not negligible, but not represented in the model.

(Isci and Martonosi 2003) express the power consumption of a Pentium 4 CPU as the sum of the power consumption of the processor's 22 major subunits. Altogether, the authors employ 24 different PMEs as model inputs. Since the processor provides only 18 PMCs, some counters are *reconfigured* or *rotated* during the experiment. The reconfiguration of the counters is particularly essential for the floating point execution unit model. It employs eight events and only two of them can be counted at a time. Thus, four counter reconfigurations are necessary to count all interesting events. This leads to inaccuracy, as events counted at one configuration are not counted in the other. The values obtained for the single subunits are weighted and combined to estimate the power consumption of the entire CPU.

The idea of Isci and Martonosi is further refined in (Bertran et al. 2012) where a methodology to develop PMC-based models for multicore processors is presented. Similar to Isci and Martonosi the authors first determine the architectural components of the processor.

They identify more than 25 components and classify them into three categories, namely, in-order engine, out-of-order engine, and memory. The authors assert that the activity of some of the components in the in-order engine cannot be detected with distinct PMEs. As a result this engine's activity is captured as a whole with the exception of the branch prediction unit. The memory engine is further divided into three parts: L1 cache, L2 cache, and the main memory where the latter includes the front side bus. Likewise, the out-of-order engine is divided into three parts: INT unit, FP unit, and SIMD unit. This way, the authors identify 8 so called power components. The authors develop a set of 97 micro-benchmarks that shall stress each of these components in isolation under different scenarios. The aim is to detect those PMEs that reflect the activity level of these components best. Based on this approach, the power consumption of the single-core of the CPU is expressed as

$$P_{total} = \left( \sum_{i=1}^{i=comps} AR_i \times P_i \right) + P_{static} \tag{4.4}$$

where $AR_i$ denotes the activity ratio in the $i$-th component and $P_i$ is the corresponding power consumption of the component so that $AR_i \times P_i$ yields the component's dynamic power consumption. $P_{static}$ is the CPU's idle state power consumption. The activity ratio is given by the value of the designated PME, normalized by the count of the *cpu_clk_unhalted* event[4]. For the multi-core scenario the CPU power is given by:

$$P_{total} = \sum_{j=1}^{j=cores} \left( \left( \sum_{i=1}^{i=comps} AR_{ij} \times P_i \right) + P_{static} \right) \tag{4.5}$$

where $AR_{ij}$ refers to the activity ratio of the $i$-th component of the $j$-th core. For the single-core model the average estimation error is between 0.32% (453.povray) and 6.87% (470.lbm), and 1.89% for all benchmarks together. For the multi-core model the average estimation error is between 1.47% (436.cactusADM-453.povray) and 10.15% (462.libquantum-454.calculix), and 4.63% for all tested pairs of benchmarks. The authors also evaluate the model's accuracy for different frequency levels ranging from 800 MHz to 2.533 GHz. As the authors mention the estimation error increases with decreasing frequency. This suggests that the model does not entirely account for the nonlinear effects that occur with dynamic voltage and frequency scaling (DVFS) techniques. Further, while the estimation error decreases the absolute error in terms of mW increases slightly (albeit within the same order of magnitude): Since power consumption increases with frequency even a small estimation error tends to increase the absolute error. Models based on CPU frequency should therefore focus on accuracy at high power consumption where a larger estimation error might be acceptable when power consumption is small. While the authors assure to develop a power model that is application independent this desireable property makes the model hard to port to other architectures or processor families: First, intensive investigation is necessary to determine equivalent PMEs to the ones the authors employed; due to the frequent changes in processor architecture a one-to-one mapping between PMEs will exist in only the fewest cases. Second, the set of microbenchmarks needs to be adequately adapted to the (new) set of relevant PMEs. In the end one needs to develop such a model for every new processor architecture from scratch and we question whether the result (which is still not perfectly application-independent and additionally dependent on the CPU frequency) justifies the necessary effort.

Fan et al. (X. Fan, Weber, and Luiz Andre Barroso 2007) propose a nonlinear model using CPU utilization as the model input. The model training phase is similar to that of (Economou

[4] Accoring to (Intel Corporation 2012), "This event counts core clock cycles when the clock signal on a specific core is running [...]" The clock is halted e.g. in power saving modes (ACPI C-states 1 and above, if the `HLT` instruction is executed, or if the thermal monitor throttles the CPU to prevent overheating.

et al. 2006), but to account for the nonlinear relationship between the performance and the power consumption, they extend the structure of a linear model with an error correction factor. The resulting model has the form

$$P_{total=}P_{idle} + (P_{busy} - P_{idle})(2u_{cpu} - u_{cpu}^{r}) \tag{4.6}$$

where $P_{busy}$ denotes the power consumption of a 100% utilized CPU, and $r$ is the correction or calibration factor minimizing the squared error of the model. The size of the correction factor depends on the system characteristics and has to be learned during a calibration phase. According to the authors, a value of $r = 1.4$ minimizes the model error. Model evaluation was done using "a few hundred" servers, but it is not clear from the paper which benchmarks (or applications) are used for this purpose. Aside of that, the authors report an average estimation error of below 1%.

(Gandhi et al. 2010) investigate the optimality of different power management policies of a server farm that trade energy for performance in individual servers as well as in multi-server environments. They model a server as an $M/M/1$ system (Kendall 1953). Unlike the previous models, the workload of a server is modeled as a random variable having a Poisson arrival rate and an exponentially distributed size. The server can be configured to enter different power modes when it does not process a workload; but when a workload arrives, the system transits to an active state and the transition from these power modes produces a corresponding performance penalty. The operational state of the server is labeled as $S_{ON}$ and the various idle states are labeled as $S_0, \ldots, S_N$, where $S_N$ denotes the $off$ state. Corresponding levels of power consumption are denoted by $P_{ON}, P_{S_0} = P_{IDLE}, P_1, \ldots, P_{S_N} = P_{OFF}$. It is assumed that $P_{ON} > P_{IDLE} > \cdots > P_{OFF} = 0$. The transition from any lower power state $S_i$ to any higher power state $S_j$ requires a transition time $T_{S_i}$, a duration which is assumed to be independent of the target state. As an upper bound estimation, the power consumed during a transition is $P_{ON}$. Moreover, $T_{IDLE} < T_{S_1} < \cdots < T_{S_{N-1}} < T_{OFF}$. For transitions from any higher power state to any lower power state, the transition duration is assumed to be $0$. A server will transit to a lower power state $S_i$ with a probability $p_i$ as soon as it becomes idle and it will *not* wakeup immediately to $S_{ON}$ as soon as a request (or "job") arrives; instead it stays in $S_i$ with a probability $q_{ij}$ until $j$ jobs are waiting in the queue. The time between two consecutive *idle* situations is called a *renewal cycle* or simply a cycle. Such cycle comprises three parts:

1. A time spent in sleep state $S_i$ until $j$ jobs arrive in the queue. The expected duration is determined by the Poisson distributed request rate $\lambda$ and is, therefore, $j/\lambda$, and the energy consumption during this phase is $j/\lambda P_{S_i}$.
2. A time $T_{S_i}$ spent during a wake-up with an energy consumption of $T_{S_i}P_{ON}$.
3. A time to serve all $j$ requests that are in the queue when starting the wake-up plus the $X$ jobs that queued up during the wake-up phase. The mean response time of the system for one job is given by $1/(\mu - \lambda)$ and the expected duration for serving all jobs is $(j + X)/(\mu - \lambda)$, and the energy consumption is $(j + X)/(\mu - \lambda)P_{ON}$.

The expected power consumption is the expected total energy consumption for all the three parts of the cycle normalized by the expected cycle duration, formally:

$$\mathbf{E}[P] = \frac{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} \left[ \frac{j}{\lambda} P_{S_i} + T_{S_i} P_{ON} + \frac{j+\lambda T_{S_i}}{\mu-\lambda} P_{ON} \right]}{\sum_{i=0}^{N} p_i \sum_{j=1}^{\infty} q_{ij} \left[ \frac{j}{\lambda} + T_{S_i} + \frac{j+\lambda T_{S_i}}{\mu-\lambda} \right]} \tag{4.7}$$

where $\lambda T_{S_i}$ is the mean value for the Poisson-distributed value $X$. The authors do not provide an evaluation result of the model. In our scenario, we assume that users are not willing to accept any delay in processing their requests.

We conclude that an increased complexity of power consumption models does not necessarily lead to more accurate models. In particular the model presented by (X. Fan, Weber, and Luiz Andre Barroso 2007) suggests that even a very simple model that includes only CPU utilization (by no question a very abstract figure) may produce very small estimation error. The advantage of such a model that employs solely very high-level parameters is clearly its simplifed portability to other machines. We will thus focus on CPU utilization-based power models.

## 4.3. TEST CASES

In the previous chapter we presented workload generation for three types of applications or scenarios, namely the video server, the transcoding application, and benchmarks from the SPEC CPU2006 suite. In the following we describe selected workloads that will be employed throughout the experiments in order to develop models for the resource utilization as a function of workload, and models for the power consumption as a function of resource utilization.

### 4.3.1. VIDEO SERVER

As explained before, the workload comprises two components, request size and request rate. In case of the video server the distribution of request sizes is determined by the size distribution of files on the server. In the present scenario we do not allow to upload additional videos to the server. To investigate the influcene of new uploads on the request size distribution nevertheless, we conduct experiments with varying numbers of videos on the server.

In our implementation of the workload generator we assume that more videos at the server side also imply an increased request rate: The more videos are available the more videos can and will be eventually requested. Therefore, by varying the number of videos on the server we also control the request rate. Additional variations of the request rate follow by the diurnal pattern as described in Section 3.3.1.4. Preliminary experiments showed that the server can safely process a workload with request rate $\lambda = 12$. At the peak of the diurnal pattern at 6 p.m. or slot 1 (cf. Fig. 3.6a) this request rate is achieved if the server hosts 1,900 videos. Thus, 1,900 is the maximum number of videos that the server will host in order to prevent overload situations. To examine the influence of different file size distributions we conduct experiments where we reduce the amount of available videos in steps of 100 down to 1,000. Below 1,000 the maximum of the load average (LAvg) is below $1$ meaning that even at the peak load during the test case less than one physical core is utilized. Thus, we experiment with 10 different file sizes. We will refer to the described test cases following the scheme DL_<#videos>_<slot>, e.g. DL_1000_03. To refer to test cases where only the number of videos is of interest we will omit the slot indicator, e.g. DL_1200.

### 4.3.2. TRANSCODER

As explained in Section 3.3.2.3 request rates in the transcoder scenario are approximately (at least) two orders of magnitude smaller than those for the video server. Since the peak request rate in the video server scenario is 12, we will start experiments in the transcoder scenario with a request rate $\lambda = 0.1$ which roughly equals 360 requests per hour. Additionally, we choose request rates $\lambda = 0.05$ and $\lambda = 0.2$ (180 and 720 requests per hour, respectively) to examine the influcene of request rate. The request size follows a lognormal distribution with cutoff as explained in Section 3.3.2.1. Regarding the target

container we recognize that recording devices typically employ the MP4 container to store data. Thus, we initially decide to use MP4 as the source container and FLV (flash), and WEBM as target containers. FLV comprises the Flash Video (flv1) codec and the Flash ADPCM Audio (SWFa) codec. WEBM comprises the VP8 Video (VP80) codec and the Vorbis Audio (vorb) codec. However, first experiments showed that using WEBM as target container with the envisioned request rate overloads the server massively: With $\lambda = 0.1$ and $\lambda = 0.2$ the load average of the server peaked at 358.05 and 363.41, respectively, which is more than 11 times above the maximum load average of 32 for this machine. This massive overload of the server results in a denial of service shortly after experiment start. Even at a request rate $\lambda = 0.0083$ (30 requests per hour) the load average peaks at 47.3 which means a server overload by 1.47. For this reason, we decide to employ an alternative target container, MKV (matroska) which comprises the MPEG-4 Video (mp4v) codec and the Vorbis Audio (vorb) codec. We will refer to these test cases following the scheme TC_<target-container>_<requests-per-hour>, e.g. TC_FLV_360.

### 4.3.3. BENCHMARKS

In Section 3.3.3 we already made considerations regarding the request size and request rate for the benchmark scenario. To repeat, we employ the six benchmarks `gcc`, `libquantum`, `xalancbmk`, `cactusADM`, `milc`, and `povray`. The request rates are 6, 12, and 60 requests per hour in the *low*, *middle* and *high* load scenario, respectively. We refer to these test cases following the scheme BM_<benchmark>_<requests-per-hour>, e.g. BM_xalancbmk_12.

## 4.4. APPLYING REGRESSION TO SAMPLES OF DIFFERENT LENGTH

To find the relationship between variables one usually applies one form of regression technique – depending on the very problem at hand and the assumptions one is able to make about the number of independent and dependent variables, the model formula, the measurement error, etc. In its most simple variant, we model the relationship between one response variable and one explanatory variable and assume a linear relationship. In such a case we speak of *simple linear* regression and the model is of the form

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{4.8}$$

where $\beta_0$ is called the *intercept*, $\beta_1$ is the unkown model parameter, $\epsilon$ is the measurement error and $i$ is an index to the $n$ measurement - or data - points $(x_i, y_i)$ with $i = \{1, n\}$. The intercept, $\beta_0$ captures all systematic errors, i.e. a shift along the $y$-axis. Thus, $\epsilon$ captures only deviations due to measurement errors or other random deviations of $y$ and is assumed to be independent and identically normally distributed with $\mu = 0$: $\epsilon \sim N(0, \sigma^2)$.

If more than one explanatory variables form the model, we speak of *multiple* linear regression and the model has the form

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i. \tag{4.9}$$

Since it is often misunderstood we should point out that the term *linear* does not make any statements about the explanatory variables, but refers to the model parameters $\beta_j$, $j = \{1, \ldots, k\}$. E.g. the model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 \sqrt{x_{2i}} + \epsilon_i \tag{4.10}$$

employs a quadratic root of the variable $x_2$. The relationship between $y$ and the regressors is thus nonlinear. In particular, the value of $y$ may depend quadratically on $x_2$ and to obtain a linear relationship $x_2$ is transformed accordingly. But the model is still linear in

all parameters $\beta_j$, $j = 1, 2$. Any of the explanatory variables may be transformed in order to obtain a linear relationship between the (transformed) regressors and the dependent variable. The only requirement is that the transforming function is smooth (i.e. the function must be infinitely differentiable in its domain).

In Eqs. (4.8) to (4.10) an additive relationship between the variables is assumed. This expresses the expectation that – let all other explanatory variables stay unchanged – an increase by $1$ in e.g. variable $x_1$ in Eq. (4.9) leads to an increase of $y$ by $\beta_1$. On the contrary, if variable $x_2$ in Eq. (4.9) is increased by $1$ then - given that all other explanatory variables stay constant - $y$ is expected to change by $b_2$. And if both, $x_1$ and $x_2$ increase by $1$ then $y$ is expected to increase by $\beta_1 + \beta_2$. Therefore, an additive model expresses the assumption that an *absolute* change in one of the explanatory variables results in an absolute change in $y$. To express rather a *proportional* relationship between the explanatory variables and the response variable *multiplicative* models are applicable:

$$y_i = \beta_0(x_{1i}^{\beta_1}) \cdots (x_{ki}^{\beta_k})\epsilon_i. \tag{4.11}$$

By applying the logarithm to both sides of the equation the model can be transformed into an additive model and thus, methods for additive linear regression models can be applied easily:

$$\begin{aligned} lg(y_i) &= log\left(\beta_0(x_{1i}^{\beta_1}) \cdots (x_{ki}^{\beta_k})\epsilon_i\right) \\ &= lg(\beta_0) + \beta_1 lg(x_{1i}) + \cdots + \beta_k lg(x_{ki}) + lg(\epsilon_i). \end{aligned} \tag{4.12}$$

In Eqs. (4.8) to (4.12) the formula is assumed to be known and is explicitly modeled. This assumption is in particular important if the relationship between the dependent variable and the explanatory variables is nonlinear: To obtain a linear relationship in order to let linear regression methods be applicable the explanatory variables need to be transformed appropriately. Sometimes, it is not clear which transformation is most appropriate. In such cases, nonparametric regression methods can be employed. In the case of one explanatory variable the nonparametric regression model is defined as

$$y_i = f(x_i) + \epsilon_i \tag{4.13}$$

where $f$ is required to be a smooth function, and for $\epsilon$ the same assumptions hold as in the cases of parametric linear regression. While this appears similar to the transformation applied to $x_2$ in Eq. (4.10) there is now no requirement that the transformation results in a parametric form. In particular, in the linear model $f$ is known and applied in advance while in the nonparametric case the function is determined from the data. Hence, the model does not define any regression parameters. While there are many other regression models we believe that this introductory overview is sufficient to demonstrate an assumption that all regression models have in common and which will be explained in the following. Further regression approaches and more details can be found in any introductory text book on regression. For the models described above we adapted explanations from (Fahrmeir, Kneib, and S. Lang 2009).

All of the above models require pairs of dependent variable and independent variable(s), $(y_i, x_{ij})$, $i = \{1, n\}$, $j = \{1, k\}$. This assumes that both, $y_i$ and all $x_{ij}$ can be observed at the same time or, more generally, that for all $i$ every $x_{ij}$ can be associated with some specific $y_i$. This assumption leads to problems if (technical) measurement data is obtained from various sources (e.g. sensors) each with different sample frequency. In a first, illustrative example let us assume we want to develop a model that estimates the power consumption of a server depending on its CPU utilization. In our test environment (Figure 2.1) the power consumption is obtained by a digital power analyzer that measures the server's power

consumption not with an exact frequency, but produces samples at ca. 7 Hz. The CPU utilization is sampled using the `dstat` monitoring tool which reads utilization metrics with a frequency of 1 Hz. While one could in principle change the code of `dstat` to sample with, say 7 Hz it would still not account for the slight changes of the sampling frequency of the digital power analyzer. Obviously, it is not a simple task to relate one sample of CPU utilization to a certain power level value. Under the assumption that values for both variables, CPU utilization and power consumption, are associated with a timestamp one could apply resampling techniques known from time series analysis. E.g. we could accept information loss and compute the average power consumption for every second and relate the averaged power consumption to the CPU utilization. Apparently, we must ensure that the clocks of the digital power analyzer and the software tool are synchronized. In practice, this is sometimes harder to be done than said.

And it is not guaranteed, that measurement data is always accompanied by timestamps: In a second example we want to estimate the size of transferred files depending on the size of files that reside on the server. Please remember that the heavy-tailedness of internet traffic is attributed to the heavy-tailedness of file sizes, cf. Section 3.1 on page 18. For simplicity let us assume the number of files on the server is fixed for a certain time span and we observe the server's responses to file transfer requests. To derive a relationship between file and request sizes it is obviously not meaningful to relate requests to the respective size of the requested file: This would merely give the file popularity distribution. Rather, we would need to relate the *distributions* of the file sizes and request sizes. While this is often done in graphical form (i.e. superimposing distributions) no approach is known to the author that addresses the problem formally. Of course, one can determine the correlation coefficient between the sampled data of these two variables or apply goodness-of-fit tests to check whether the request sizes are taken from the distribution of file sizes. This, however still does not allow to establish a functional relationship between the two variables.

We present a technique that can address both problems, regression of different-sized samples where resampling is not available or acceptable, and expressing the empirical distribution of the dependent variable as a function of the empirical distribution function of an explanatory variable. Additionally, we suggest to use shift functions to assess the error of the fit.

### 4.4.1. FINDING PAIRS OF DEPENDENT AND EXPLANATORY VARIABLES

To find pairs of dependent and explanatory variables we will exploit properties of empirical distribution functions (EDFs).

For observations $x_1, \ldots, x_i, \ldots, x_m$, the EDF is defined as

$$F_m(x) = \frac{1}{m} \sum_{i=1}^{m} I\{x_i \leq x\} \tag{4.14}$$

where $I(x_i \leq x)$ is an indicator function with

$$I(x_i \leq x) := \begin{cases} 1 & : x_i \leq x \\ 0 & : x_i > x. \end{cases} \tag{4.15}$$

Alternatively, if $h_j$ denotes the absolute frequency of the characteristic value $v_j$, i.e. $h_j = \sum_i I\{x_i = x_j\}$, and $f_j = \frac{h_j}{m}$ denotes the relative frequency of $v_j$ then

$$F_m(x) = \sum_{j:\ v_j \leq x} f_j. \tag{4.16}$$

For every EDF $F_m$ the following properties hold: $F_m$ is non-decreasing and right-continuous. Furthermore, if $x_{min}$ denotes the smallest observed value among $m$ observations of random variable $X$, $x_1, \ldots, x_m$, then

$$F_m(x) = 0, \quad x < x_{min}. \tag{4.17}$$

If $x_{max}$ denotes the maximum value observed among $x_1, \ldots, x_m$, then

$$F_m(x) = 1, \quad x \geq x_{max}. \tag{4.18}$$

Now let us assume we have two empirical distribution functions, $F_m(x)$, and $G_n(y)$. We want to determine pairs $(x_i, y_j)$; $i = \{1, \ldots, m\}$ and $j = \{1, \ldots, n\}$. Since no $x_i$ can be directly related to any $y_j$ and vice versa we try to find pairs of *matching quantiles* where a match is defined by two quantiles having the same order: For every $x_i$ with $F_m(x_i) = p$ we search for $y_j$ such that $G_n(y_j) = p$, or

$$y_j = G_n^{-1}(F_m(x_i)) \tag{4.19}$$

where

$$F^{-1}(p) = Q(p) = inf\{x : F(x) \geq p\}. \tag{4.20}$$

is the *quantile function*. Plotting the pairs of quantiles matched by their order we then obtain the well-known quantile plot. Essentially, the approach is a regression against the QQ curve. Here, we need to address a problem that arises with discrete distributions:

The quantile function, $F^{-1}(p)$ is only defined for continuous distributions. The reason is that while for discrete distributions it is principally feasible to determine the quantile, $x$, for some order $p$ using Equation (4.20), in such case generally $F(F^{-1}(p)) \neq p$. Furthermore, for two orders, $p_1$, $p_2$, one may obtain the same quantile: $F^{-1}(p_1) = F^{-1}(p_2)$. A short example illustrates that: Let $X = \{1, 2, 3\}$, $p_1 = 0.25$, and $p_2 = 1/3$. According to Equation (4.20), $F^{-1}(p_1) = 1$, but $F_3(F^{-1}(p_1)) = 1/3 \neq p_1$. On the other hand $F^{-1}(p_2) = 1 = F^{-1}(p_1)$.

In (Hyndman and Y. Fan 1996) the authors compare how quantiles are computed in different statistical libraries. Common to all definitions is to use one, or two (adjacent) order statistics in order to determine the quantile:

$$\hat{Q}(p) = (1 - \gamma)X_{(j)} + \gamma X_{(j+1)} \tag{4.21}$$

where $X_{(j)}$ denotes the $j$-th order statistic in the sample,

$$p_k = \frac{k - \alpha}{n - \alpha - \beta + 1}\ , \tag{4.22}$$

and

$$\gamma = p_k n + l - j \tag{4.23}$$

with $j = \lfloor p_k n + l \rfloor$, $l = \alpha + p(1 - \alpha - \beta)$, and $\alpha$, $\beta$ are some constants, typically in the range $[0, 1]$

As Hyndman and Fan point out, the problem to estimate the quantiles of a specific order is related to finding plotting positions in quantile plots (plotting $X_{(k)}$ against the theoretical $p_k$) or QQ curves (plotting $X_{(k)}$ against (theoretical) $F^{-1}(p_k)$) where it is usually defined that $\hat{Q}(p_k) = X_{(k)}$ in Equation (4.21) and linear interpolation is applied in cases $p \neq p_k$. Depending on how one interpolates between the points $(p_k, X_{(k)})$ the quantile functions may possess more or less of desired properties which Hyndman and Fan list as follows:

**P1** $\hat{Q}(p_k)$ is continuous.
(common assumption)

**P2** $Freq(X_k \leq \hat{Q}(p_k)) \geq p_k n$,
where $Freq(X_k \leq x)$ denotes the number of observations less than or equal $x$.
(Sample analog of $F(Q(p_k)) \geq p_k$ where equality holds if $F$ is continuous.)

**P3** $Freq(X_k \leq \hat{Q}(p_k)) = Freq(X_k \geq \hat{Q}(1 - p_k))$.
(symmetry in the tails of the quantile function)

**P4** where $\hat{Q}^{-1}(x)$ is uniquely defined,
$\hat{Q}^{-1}(X_{(k)}) + \hat{Q}^{-1}(X_{(n-k+1)})$, for $k = 1, \ldots, n$.
(symmetry in the tails of the inverse quantile function)

**P5** where $\hat{Q}^{-1}(x)$ is uniquely defined,
$\hat{Q}^{-1}(X_{(1)}) > 0$ and $\hat{Q}^{-1}(X_{(n)}) < 1$.
(Requirement that the quantile function has probability $< 0$ for values beyond the smallest, and $> 1$ for values beyond the largest order statistics in the sample.)

**P6** $\hat{Q}(.5))$ is equal to the sample median defined by
$(X_{(s)} + X_{(s+1)})/2$ , if $n = 2s$
$X_{(s+1)}$ , otherwise .
(Specific requirement to stress the importance of the median.)

To derive estimates of $p_k$ one may either choose $p_k = \mathsf{L}(F(X_{(k)}))$ or $p_k = F(\mathsf{L}(X_{(k)}))$ where L denotes some measure of location like mean, median or mode. If, in particular, $\mathsf{L} = \mathsf{M}$, the median, both approaches are equivalent. Since order statistics are well-known to be beta-distributed (Johnson and Kotz 1970), $F(X_{(k)})$ follows the incomplete beta function ratio. According to (Hyndman and Y. Fan 1996) an approximation to $\mathsf{M}(F(X_{(k)}))$ is given by

$$\mathsf{M}(F(X_{(k)})) \approx \frac{(k - \frac{1}{3})}{(n + \frac{1}{3})}. \tag{4.24}$$

However, this approximation appears to be the same as in (Cunnane 1978) where it is listed as appoximation to $E(X_{(k)})$, the *mean* of the $k$-th order statistic. It would be justified to employ this approximation if the underlying distribution is normal since in this case mean and median are the same. In fact, Equation (4.24) is originally reported in (Tukey 1962) as approximation to find quantiles of a normal distribution. According to (Cunnane 1978), a value of $\alpha = 0.31$ was determined by (Beard 1962) as approximation to $\mathsf{M}(F(X_{(k)}))$. We therefore use $\alpha = \beta = 0.31$ in Equation (4.24) to estimate quantiles.

### 4.4.2. ESTIMATE DIFFERENCES BETWEEN EMPIRICAL DISTRIBUTION FUNCTIONS

The task to compare samples origins from the realm of goodness-of-fit tests and usually asks the question whether two samples are taken from the same distribution. To examine such hypotheses several tests – differing how the test statistic is defined – exist: The two-sample Kolmogorov-Smirnov test, and the Cramér-von Mises criterion provide means to test whether the empirical distributions of two samples differ. The k-sample Anderson-Darling test expands this to $k$ samples. Despite the differences in the respective test criteria, all tests examine the difference between the reference distribution and the other distribution by means of the deviation along the y-axis between the (empirical) distribution functions. In our case, albeit we try to express one empirical distribution by another, we are aware that results that would justify the application of named goodness-of-fit tests are hardly achievable. The main reason is that the models which describe the mapping need to be general enough to provide reasonably good fits even if input data changes. Therefore, the models will produce results of varying quality. In advance we can thus

assume that conventional goodness-of-fit tests will almost always reject the hypothesis that both samples origin from the same distribution in almost every case. This assumption is also supported by the sensitivity of the tests to sample size: It is well known that the power of goodness-of-fit tests increases with sample size and that beyond a certain sample size the test power is $1$ (Razali and Wah 2011). At a significance level of $0.05$ the critical sample size for the Kolmogorov-Smirnov test, and the Anderson-Darling test is 2000 and 300, respectively. In our test environment, where we observe workload to a server over the period of up to 24 hours we easily obtain sample sizes $> 100000$.

For these reasons we need to employ a different means to assess the deviation between empirical distributions. The following explanations on shift functions and confidence bands are based on and summarize (Doksum 1974; Doksum and Sievers 1976; Doksum 1977). A *shift function*, $\Delta(x)$, can be used as a means to measure the horizontal deviation between two distribution functions, $F_X(x)$ and $G_Y(y)$:

$$\Delta(x) = inf\{\Delta : F_X(x) \leq G_Y(x + \Delta)\} \tag{4.25}$$

where $x \in supp(F) = \{x : 0 < F(x) < 1\}$. The estimate for $\Delta(x)$, $\hat{\Delta}_N(x)$, is given by the empirical shift function

$$\hat{\Delta}_N(x) = G_m^{-1}(F_n(x)) - x. \tag{4.26}$$

Equation (4.26) is expressible in terms of the order statistics of $Y$: if $Y_{(i)}$ denotes the $i$-th order statistic in a sample of $Y$ and if $\lfloor k \rfloor$ denotes the largest integer smaller than $k$, then

$$\hat{\Delta}_N(x) = Y_{(\lfloor mF_n(x) \rfloor + 1)} - x. \tag{4.27}$$

Shift functions are originally intended to answer questions like – e.g. in case of testing new medicine – whether a new treatment has a positive effect for all members of the test group ($\Delta(x) > 0$ for all $x$) or, if not, for which part of the test group it is ($\{x : \Delta(x) > 0\}$. For these reasons $\Delta(\cdot)$ is also called the *response function*. In that domain it is preferrable to observe equal (positive or negative) effects which can be expressed by asking $\Delta(x) =^? \theta$ for some $\theta$ and all $x$ or, if such a constant effect cannot be observed, whether the effect may depend linear on $x$ (e.g. if $x$ is a numerical representation of some medical condition the effect of the medication shall depend linear on that condiction) which is expressible in terms of $\Delta(x) =^? \alpha + \beta x$ for some $\alpha$ and $\beta$ and all $x$.

To assess the reliability of $\hat{\Delta}_N(x)$ as estimate for $\Delta(x)$ Doksum and Sievers construct different confidence bands. The fundamental idea is to employ distribution-free test statistics $T(F_m, G_n)$ that have the property that the inequality $T(F_m, G_n) \leq K$ is equivalent to

$$H_* \left(F_m(x)\right) \leq G_n(x) \leq H^* \left(F_m(x)\right) \tag{4.28}$$

for all $x$ and some functions $H_*$, $H^*$, and some critical value $K \in \mathbb{R}$. If now the value of $K$ is chosen such that

$$Pr\left[T(F_m, G_n) \leq K\right] = 1 - \alpha \tag{4.29}$$

then a level $1 - \alpha$ simultaneous, distribution-free confidence band for $x = (-\infty, +\infty)$ is given by

$$\left[G_n^{-1}(H_*(F_m(x))) - x, G_n^{-I}(H^*(F_m(x))) - x\right) \tag{4.30}$$

where $G_n^{-1}$ denotes the left-inverse according to Equation (4.20) and $G_n^{-I}$ denotes the right-inverse which is defined as

$$F^{-I}(p) = sup\{x : F(x) \leq p\}. \tag{4.31}$$

The band in Equation (4.30) can be expressed in terms of order statistics as follows:

$$[\Delta_*(x), \Delta^*(x)) = \left[Y_{(\lceil nH_*(\frac{i}{m}) \rceil)} - x, Y_{(\lfloor nH^*(\frac{i}{m}) \rfloor + 1)} - x\right). \tag{4.32}$$

Depending on how $H_*$ and $H^*$ are chosen different confidence bands can be designed with varying properties. Now, define $G_{\Delta,n}$ as

$$\begin{aligned} G_{\Delta,n}(x) &= G_n(x + \Delta(x)) \\ &= |\{j : Y_j \leq x + \Delta(x)\}| \, n^{-1} \\ &= \left|\{j : Y_j \leq G^{-1}(F(x))\}\right| n^{-1} \\ &= \left|\{j : F^{-1}(G(Y_j)) \leq x\}\right| n^{-1}. \end{aligned} \tag{4.33}$$

Since $F^{-1}(G((x)) =_d F(x)$ we have $G_{\Delta,n}(x) =_d F_n(x)$. Therefore, a first choice is to employ the Kolmogorov-Smirnov (KS) test statistic for $T(F_m, G_{\Delta,n})$:

$$T_S(F_m, G_{\Delta,n}) = M^{1/2} \sup_x |F_m(x) - G_{\Delta,n}| \, . \tag{4.34}$$

where $M = {}^{mn}/_{m+n}$. In this case $H_*(x) = x - {}^{K}/_{M^{1/2}}$ and $H^*(x) = x + {}^{K}/_{M^{1/2}}$ (Doksum and Sievers 1976). If inserted in Equation (4.30) we obtain

$$\left[ \, G_n^{-1}(F_m(x) - {}^{K_\alpha}/_{M^{1/2}}) - x, G_n^{-I}(F_m(x) + {}^{K_\alpha}/_{M^{1/2}}) - x \, \right) \tag{4.35}$$

which is named the *S band* due to it was first described in (Switzer 1976). A second approach to the confidence band employs a weighted supremum norm statistic which is a weighted version of the KS test and is defined as

$$T_W(F_m, G_{\Delta,n}) = \sup_{\{x : a \leq F_m(x) \leq b\}} \frac{|F_m(x) - G_{\Delta,n}(x)|}{\hat{\sigma}_N(F_m, G_{\Delta,n})} \tag{4.36}$$

where $0 < a < b < 1$, $\hat{\sigma}_N(F_m, G_{\Delta,n})$ is an estimate of the standard deviation $\sigma_N$ of $G_{\Delta,n}(x) - F_m(x)$, and $N = m + n$. As an estimate for $\sigma_N^2$ Doksum suggests

$$\hat{\sigma}_N^2(x) = \frac{N H_N(x)(1 - H_N(x))}{mn} \tag{4.37}$$

where $H_N = \lambda F_m + (1 - \lambda) G_{\Delta,n}$ and $\lambda = {}^{m}/_{N}$. Then, Equation (4.36) can be rewritten as

$$T_W(F_m, G_{\Delta,n}) = M^{1/2} \sup_{\{x : a \leq F_m(x) \leq b\}} \frac{|F_m(x) - G_{\Delta,n}(x)|}{\Psi(H_N(x))} \tag{4.38}$$

where $\Psi(t) = (t(1-t))^{1/2}$. The test statistic

$$\frac{|F_m(x) - G_{\Delta,n}(x)|}{\hat{\sigma}_N(F_m, G_{\Delta,n})} \tag{4.39}$$

is the absolute value of an approximately standard normal distributed variable and therefore, we can use critical values from $|\mathcal{N}(0,1)|$ from to solve the inequality $T_W(F_m, G_{\Delta,n}) \leq K_{\alpha/2}$ for $G_{\Delta,n}$. Together with the structure of $\Psi(t)$ we obtain

$$\begin{aligned} M^{1/2} \frac{|F_m(x) - G_{\Delta,n}(x)|}{\Psi(H_N(x))} &\leq K_{\alpha/2} \\ |F_m(x) - G_{\Delta,n}(x)| &\leq K_{\alpha/2} \Psi(H_N(x)) M^{-1/2} \\ (F_m(x) - G_{\Delta,n}(x))^2 &\leq K_{\alpha/2}^2 \Psi(H_N(x))^2 M^{-1} \\ &\leq K_{\alpha/2}^2 (t(1-t)) M^{-1} \\ &\leq K_{\alpha/2}^2 \left[\lambda F_m(x) + (1-\lambda) G_{\Delta,n}(x)\right] \\ &\quad \cdot \left[1 - (\lambda F_m(x) + (1-\lambda) G_{\Delta,n}(x))\right] M^{-1} \end{aligned} \tag{4.40}$$

with $x \in \{x : a \leq F_m(x) \leq b\}$. To find appropriate functions $H_*$ and $H^*$, first note that Equation (4.40) can be rewritten as $d(v) \leq 0$:

$$\begin{aligned} -(u-v)^2 &\leq -c(\lambda u + (1-\lambda)v)(1-(\lambda u + (1-\lambda)v)) \\ -u^2 - 2uv - v^2 &\leq -c(\lambda u - \lambda^2 u^2 - (1-\lambda)\lambda uv + (1-\lambda)v - (1-\lambda)\lambda uv - (1-\lambda)^2 v^2) \\ d(v) &= -c\lambda u + c\lambda^2 u^2 + u^2 - v(2u - c(1-\lambda)(2\lambda u - 1)) + v^2(1 + c(1-\lambda)^2) \end{aligned} \tag{4.41}$$

where $c = {}^{K^2_{\alpha/2}}/_{M}$, $u = F_m(x)$, and $v = G_{\Delta,n}(x)$. Now, to determine when $d(v) \leq 0$ we need to find the real roots of $d(v) = 0$ which are

$$H^{\pm} = \frac{u + {}^1/_2 c(1-\lambda)(1-2\lambda u) \pm {}^1/_2 (c^2(1-\lambda)^2 + 4cu(1-u))^{1/2}}{1 + c(1-\lambda)^2} \tag{4.42}$$

where $H^-$ corresponds to $H_*$ and $H^+$ to $H^*$. Inserting $H^{\pm}$ in Equation (4.30) we obtain a simultaneous, level $(1-\alpha)$ confidence band for all $\{x : a \leq F_m(x) \leq b\}$. This band is dubbed *W band*.

The W band employs a weighted supremum norm statistic where the weight is a function $(H_N(x)(1-H_N(x))^{1/2}$ and the supremum is taken over $\{x : a \leq F_m(x) \leq b\}$ (see Equation (4.36)). If the weight is merely $H_N(x)$ and the supremum is over $\{x : F_m(x) \leq c\}$, one obtains the Renyi statistic. For the associated *R band* the $h$ functions are defined as

$$H^{\pm}(u) = \frac{1 \pm \lambda \cdot {}^r/_{M^{1/2}}}{1 \mp (1-\lambda) \cdot {}^r/_{M^{1/2}}} \tag{4.43}$$

where $r$ is the level $\alpha$ critical value for the Renyi statistic.

Specifically in (Doksum and Sievers 1976) these bands are compared with regard to their asymptotic relative efficiency. First, it is assumed that $H_*(u)$ and $H^*(u)$ converge to $u$ as $M^{-1/2} \to 0$ and that right-hand derivates of $H_*(u)$ and $H^*(u)$ with respect to $M^{-1/2}$ exist and are continuous in $F_m(x)$. Let $h_*(u)$ and $h^*(u)$ denote the derivates of $H_*(u)$ and $H^*(u)$, respectively for some confidence band and $k_*(u)$ and $k^*(u)$ the respective derivatives for some other band. The relative asymptotic efficiency is then defined as

$$e_{h,k}(x) = \left( \frac{k^*(u) - k_*(u)}{h^*(u) - h_*(u)} \right)^2. \tag{4.44}$$

In the case of the S and W bands, this efficiency is

$$e_{S,W}(x_p) = ({}^{K_{W,\alpha}}/_{K_{S,\alpha}})^2 \, p(1-p) \quad (a \leq p \leq b) \tag{4.45}$$

where $x_p = F^{-1}(p)$, and $K_{W,\alpha}$, $K_{S,\alpha}$ denote the asymptotic level $\alpha$ critical values of the W and S band, respectively. It turns out that the S band is more efficient (i.e. narrower) than the W band in the center quantiles – $[0.4, 0.6]$ percentiles if $a = 1 - b = 0.25$ and within the $[0.3, 0.7]$ percentiles if $a = 1 - b \leq 0.1$ – while it is less efficient in the tails. Complementary, due to the structure of $\Psi(t)$ in Equation (4.38) the asymptotic variance of $T_W$ is independent of $x$ and thus, the W band forms more evenly around the shift function for all $x$ than the S band. The W band is also more efficent than the R band except in the lower quantiles (up to the 0.25 percentile) of the distribution.

The performance of the models in the upper quantiles , i.e the deviation between the actual and estimated distribution, is of paramount importance to us: Firstly, in the previous chapter we pointed out that the distribution of request sizes is heavy-tailed and therefore also the distribution of transmission times. If we would like to model the utilization of system resource as a function of workload – and in particular request sizes – we must consider the relation between the upper quantiles of the request size distribution and

the resource utilization distributions. Secondly, upper quantiles of resource utilization are used to enable overbooking of servers, such that more applications (or "services") can be scheduled on one machine in order to increase its overall utilization and improve its energy efficiency: E.g. (Urgaonkar, Shenoy, and Roscoe 2009) use the $.95$-percentile of the CPU utilization of an application, (Speitkamp and Bichler 2010) use the $.95$-percentile of the workload distribution, and (Tomás et al. 2014) use the $.95$-percentile of the response time as targets to guarantee Quality of Service (QoS) in overbooked situations. In either case the regarded percentile is above $0.8$ and thus the higher efficiency of the S band in the central quantiles is of secondary interest. We thus decide to employ the W band as confidence band to the shift function. This band can be computed using Equation (4.32) together with Equation (4.41). Tables of critical values can be found in (Borokov and Sycheva 1968), (Canner 1975), and (Doksum and Sievers 1976). While the tables by Canner provide values for $\{x : a \leq F_m(x) \leq b\}$ with $a = 0$ and $b = 1 - a$ these values are limited to cases where $m = n$. Doksum and Sievers provide asymptotical values for cases where also $m = n$. Complementary, the tables by Borokov and Sycheva do not make this assumption, but are limited to quantiles in the range of the $[0.1, 0.9]$ and $[0.25, 0.75]$ percentiles. Thus, we should consider the deviation between the actual and modeled EDFs in the upper tail and compute appropriate confidence bands. But a value $a > 0$ restricts the band to central percentiles, ignoring percentiles in the very lower and very upper tails. We should thus take critical values for $a = 0$; a criterion that is met only by the values provided by Canner. As mentioned right before, these values are only valid for $m = n$, however. To meet this requirement we will take uniformly distributed samples from the $X$ and $Y$ samples, both of the same length. Since the tables in (Canner 1975) are given only for values of $n$ up to 1000 we take samples of this length from the actual data.

### 4.4.3. ERROR MEASURES USING THE SHIFT FUNCTION

Remember that we try to express one empirical distribution function by another and that shift functions are a means to assess the horizontal deviation between two EDFs. We will therefore apply the empirical shift to compute errors. A simple error measure in the mean squared error (MSE) which is defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2 \tag{4.46}$$

where $\hat{x}$ denotes the estimated value and $x$ the actual value. Although widely applied, it has the disadvantage of not being scale-free which means that the extent of the error depends on the value range of $x$. This property renders the MSE unattractive since we apply our model approach in various scenarios where estimated and actual values take different value ranges: In one scenario we would obtain very small errors while in a second scenario error values would be orders of magnitudes larger. And even for a specific scenario we would not gain much information about how large the error is compared to the actual data. For this reason we prefer percentage errors. A very old definition is that of the mean absolute percentage error (MAPE):

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{x}_i - x_i}{x_i}\right| \tag{4.47}$$

This error measure is scale-free, but it only exists if the actual data contains no zero values. (Makridakis 1993) mention that with MAPE "equal errors above the actual value result in greater $APE$ [...] than those below the actual value". They argue that in the two cases $|100 - 150|/150$ and $|150 - 100|/100$ the absolute error is 50 and the percentage error should thus be equal as well. Therefore they propose the following definition which they later dub

symmetric mean absolute percentage error (sMAPE) in (Makridakis and Hibon 2000):

$$sMAPE_M = \frac{100}{n} \sum_{i=1}^{n} 2 \left| \frac{\hat{x}_i - x_i}{\hat{x}_i + x_i} \right| \ . \tag{4.48}$$

While this error is attractive to some extent since it exists if $\hat{x}_i + x_i \neq 0$ we find the argumentation of Makridakis (and later also Hibon) not convincing since to us it makes truly a difference whether the actual value is $100$ or $150$. Also, this definition produces misleading results: Consider e.g. the two situations that the estimated value is (a) $2 \cdot x_i$ and (b) $1/2 \cdot x_i$: Since the term in the sum Equation (4.47) is essentially $\frac{|k \cdot x_i - x_i|}{x_i}$ that term in Equation (4.48) becomes

$$\begin{aligned} & 2 \left| \frac{k \cdot x_i - x_i}{k \cdot x_i + x_i} \right| \\ =& 2 \left| \frac{k-1}{k+1} \right| \end{aligned} \tag{4.49}$$

In case (a) and (b) this term is $2/3$ which is counter-intuitive since error measures for these very different cases should be different as well. Finally, we find that the term "symmetric" is not justified: Consider case (c) where $k = 1.1$ and case (d) with $k = 0.9$. As just mentioned the conventional absolute percentage error (APE) term is essentially

$$\begin{aligned} & \left[ \frac{k \cdot x_i - x_i}{x_i} \right| \\ =& |k-1| \ . \end{aligned} \tag{4.50}$$

Thus, both cases result in a MAPE of $10\%$. With $sMAPE_M$, for case (c) one obtains $10.5\%$ while for case (d) the result is $9.5\%$. An alternative definition of the sMAPE is given by (Z. Chen and Yang 2004)[5]:

$$sMAPE_C = \frac{1}{n} \sum_{i=1}^{n} 2 \frac{\hat{x}_i - x_i}{|\hat{x}_i| + |x_i|} \tag{4.51}$$

This definition is unfortunately even less attractive since its range is limited to $[0, 2]$ and it is thus not appropriate to call it a *percentage* error. In the domain of time series forecast (Z. Chen and Yang 2004) additionally suggest a modification to the sMAPE to counter situations when $|\hat{x}_i| + |x_i| = 0$ by adding a term to the denominator resulting in a *modified* sMAPE:

$$msMAPE_C = \frac{1}{n} \sum_{i=1}^{n} \frac{\hat{x}_i - x_i}{(|\hat{x}_i| + |x_i|)/2 + S_i} \tag{4.52}$$

where

$$S_i = \frac{1}{i-1} \sum_{k=1}^{i-1} |x_k - \overline{x}_{i-1}| \tag{4.53}$$

and

$$\overline{x}_{i-1} = \frac{1}{i-1} \sum_{k=1}^{i-1} x_k \tag{4.54}$$

The term $S_i$ denotes the average of the normalized previous values where the normalization term, $\overline{x}_{i-1}$, is the average of these previous values. In principle, this error measure could also be applied to our domain. Yet two problems come to our attention: First, the definition

[5] Actually, Chen and Yang attribute this defintion to Makridakis and Hibon.

is not defined for $i = 1$ which can be treated by defining $S_1 = 0$. The second problem is more serious: While Chen and Yang promise to prevent inflation of the error measure if the sample of actual values contains zeros exactly this will happen if by chance $x_1 = 0$ and $\hat{x}_2 = x_2 = 0$ since, essentially,

$$S_2 = |x_1 - \overline{x}_1| = 0 \tag{4.55}$$

Additionally, the definition does not prevent the inflation of the error in presence of very small values in the actual data sample. Summarizing,

- MAPE is not defined if the series of actual values contains $0$,
- sMAPE is misleading and also not symmetric,
- all kinds of percentage errors suffer from inflation if the actual values are close to zero.

We will therefore use the conventional MAPE where it exists, i.e. actual values contain no zeros. Additionally, since mean values are not robust against outliers we define the median absolute percentage error (medAPE):

$$medAPE = 100 \cdot \mathsf{M}\left(\left|\frac{\hat{x}_i - x_i}{x_i}\right|\right) \ . \tag{4.56}$$

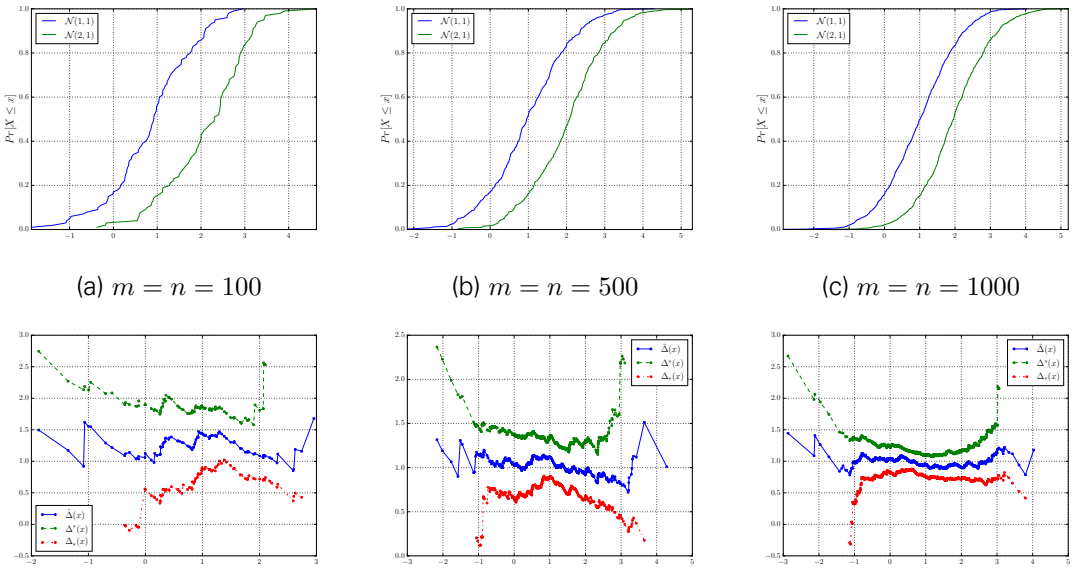
This definition also allows to obtain percentage errors if the MAPE does not exist as long as less than $50\%$ of the actual data point are zero. In cases where the medAPE does not exist as well, i.e. at least $50\%$ of the actual data are zero, we employ the MSE. Of course, statements regarding the relative accuracy of the model cannot be made in these cases. For a better understanding we will in the following demonstrate shift functions and confidence bands together with the error measures by means of three examples. In the first example, let $X$ be a sample from a $\mathcal{N}(1,1)$ distribution and $Y$ be a sample from $\mathcal{N}(2,1)$. Thus, $F_Y(y)$ follows $F_X(x)$, but is shifted by $1$ to the right. We take samples of sizes $100$, $500$, and $1,000$ to also demonstrate the influence of the sample size on the width of the confidence band. In Figure 4.1 we show the EDFs for the three samples sizes and the respective shift functions. Three things shall be of particular interest: Firstly, $sup\{\Delta_*(x_i)\} < inf\{\Delta^*(x_i)\}$. This allows to fit a straight line between the confidence bands which suggests a linear shift along the x-axis between the two distribution functions. Secondly, the empirical shift function $\hat{\Delta}(x)$ is approximately constant around $1$ which suggests that $F_n(y)$ is shifted by $1$ to the right. The empirical shift function is thus $\hat{\Delta}(x) = 1$. Thirdly, it is clear to see that the width of the confidence bands decreases with increasing sample size. To gain more insight we compute the MAPE and medAPEs, and tabulate the values in Table 4.1.

| Sample Size | 100 | 500 | 1000 |
|---|---|---|---|
| MAPE | $600.819$ | $542.022$ | $866.069$ |
| medAPE | $146.479$ | $110.928$ | $86.798$ |

Table 4.1.: Error measures for the shift between $\mathcal{N}(1,1)$ and $\mathcal{N}(2,1)$.

These values demonstrate the inflation of the percentage errors when the actual data contains small values. At the same time we see that the median versions of the percentage errors are much more robust: medAPE returns a value around $100\%$ which meets the value that is to be naïvely expected.

In the second example we take a look at how the shift function and the error measures behave depending on variance. Similar to before we generate normally distributed values

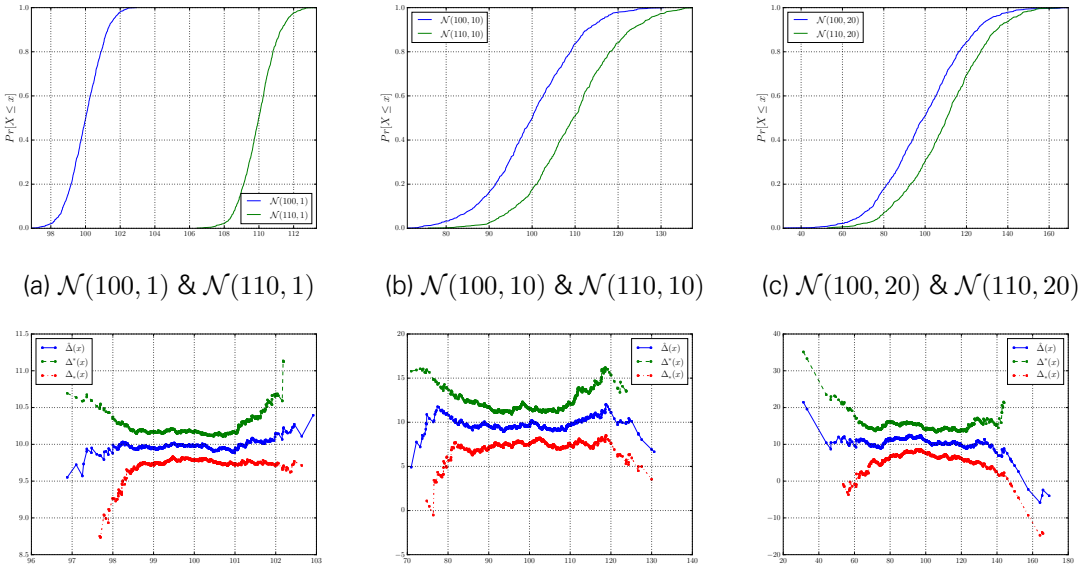


(a) $m = n = 100$ (b) $m = n = 500$ (c) $m = n = 1000$

(d) $m = n = 100$ (e) $m = n = 500$ (f) $m = n = 1000$

Figure 4.1.: EDFs for $\mathcal{N}(1,1)$ and $\mathcal{N}(2,1)$ for three different sample sizes and respective shift functions with $1 - \alpha = 0.0$ W bands for $a = 0.05$.

and, to exclude negative values, we now take mean values $\mu_1 = 100$ and $\mu_2 = 110$. The MAPE of the shift between the EDF with $\mu_2$ and the EDF with $\mu_1$ is thus expected to be $10\%$. As variances we take values $\sigma_1^2 = 1$, $\sigma_2^2 = 10$, and $\sigma_3^2 = 20$. Figure 4.2 shows the EDFs and the empirical shift function. The MSEs and percentage errors are given in Table 4.2.



(a) $\mathcal{N}(100,1)$ & $\mathcal{N}(110,1)$ (b) $\mathcal{N}(100,10)$ & $\mathcal{N}(110,10)$ (c) $\mathcal{N}(100,20)$ & $\mathcal{N}(110,20)$

(d) $\sigma_1^2$ (e) $\sigma_2^2$ (f) $\sigma_3^2$

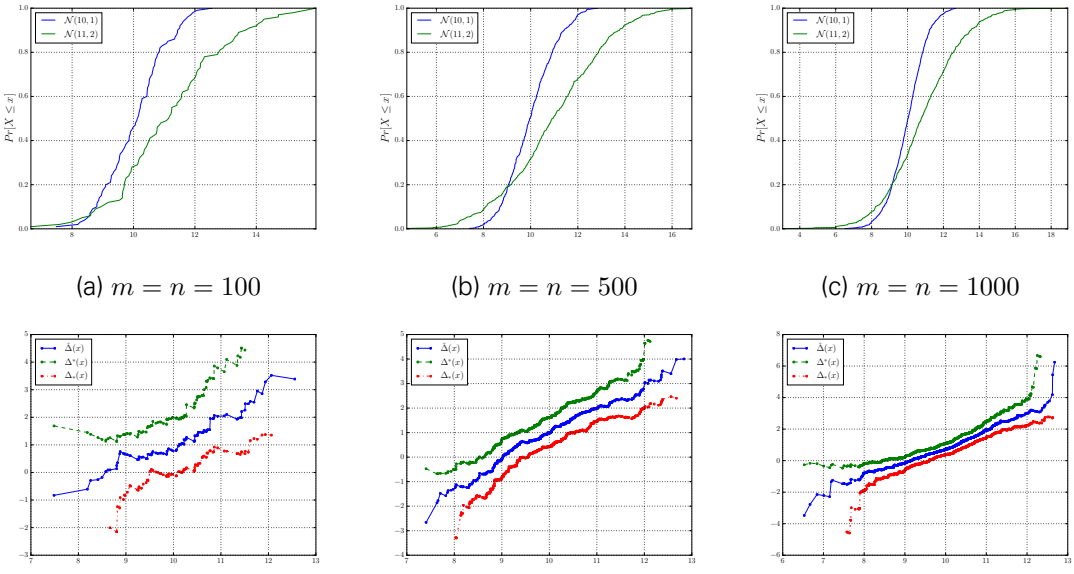
Figure 4.2.: Behavior of shift functions depending on variance where $m = n = 1000$, $1 - \alpha = 0.95$, $a = 0.0$.

Examining Figure 4.2 we cannot conclude that the variance influences the shift function: In all three cases the confidence bands allow to assume a linear shift and the shift function

| | $\mathcal{N}(100,1)$ vs. $\mathcal{N}(110,1)$ | $\mathcal{N}(100,10)$ vs. $\mathcal{N}(110,10)$ | $\mathcal{N}(100,20)$ vs. $\mathcal{N}(110,20)$ |
|---|---|---|---|
| MAPE | 10.066 | 9.741 | 10.289 |
| medAPE | 10.074 | 9.539 | 9.955 |

Table 4.2.: Behavior of the mean and median absolute percentage errors depending on the variance of the actual data.

is nearly constant around $10$. The values in Table 4.2 do also not indicate that if the shift is constant the errors increase with variance. In the third example let $X \sim \mathcal{N}(10,1)$ and $Y \sim \mathcal{N}(11,2)$. Sample sizes are again $100$, $500$, and $1,000$. Analogous to before, Figure 4.3 shows the EDFs and respective shift functions and Table 4.3 tabulates the respective errors. The linear shift functions in Figure 4.3 support the assumption of a linear location-scale



(a) $m = n = 100$ (b) $m = n = 500$ (c) $m = n = 1000$

(d) $m = n = 100$ (e) $m = n = 100$ (f) $m = n = 1000$

Figure 4.3.: EDFs for $\mathcal{N}(10,1)$ and $\mathcal{N}(11,2)$ for three different sample sizes and respective shift functions with $1 - \alpha = 0.95$ W bands for $a = 0.0$.

| Sample Size | 100 | 500 | 1000 |
|---|---|---|---|
| MAPE | 11.161 | 11.853 | 10.0816 |
| medAPE | 10.044 | 11.990 | 8.419 |

Table 4.3.: Percentage errors for the shift between $\mathcal{N}(10,1)$ and $\mathcal{N}(11,2)$.

model, i.e. the $Y$ sample is shifted along the x-axis and scaled linearly. The empirical shift function in a location-scale model can be described by $\Delta(x) = \mu_2 + \sigma_2(x - \mu_1)/\sigma_1 - x$ (Doksum and Sievers 1976) and by setting the appropriate values we obtain $\Delta(x) = x - 9$ for this example.

To summarize, we note that the MAPE does not exist if the true values contain at least one zero value. Also, the percentage error measures are sensitive to the presence of small values in the actual (or *measured*) data, and it is sensitive to the variance of the data. To the best knowledge of the author, no percentage error measure exists that is robust to these issues.

### 4.4.4. GENERAL REMARKS ON THE APPROACH

To estimate coefficients of a regression problem one usually employs the Ordinary Least Squares estimator (OLS). This estimator makes several assumptions in order to guarantee the optimality of the result. These are:

1. independent variables are not disturbed by measurement errors,
2. the variance in the errors of the dependent variable is constant (i.e. homoskedastic),
3. the errors of the dependent variable are uncorrelated,
4. the mean of the dependent variable is a linear combination of the independent variables,
5. the independent variables are no linear combinations of each other, i.e. they are pair-wise independent.

Assumptions 3 and 5 are of particular interest to us. Simplified, assumption 3 means that one should not be able to make reliable predictions about the residual for some $x_{i+1}$ by knowing the residual for the previous $x_i$. However, since the points that we are about to fit by a regression curve are essentially the points on a QQ curve the residuals of a regression will be correlated. While this does not affect the estimates of the regression coefficients, it does not guarantee that the estimate is *efficient*, meaning that it is not necessarily the *best* linear unbiased estimator. To account for this, a heteroskedastatic and autocorrelation consistent (HAC) covariance matrix must be used with the appropriate lag of autocorrelation. To determine the correct lag we fit the models in a first round and consult the partial autocorrelation function (PACF) of the residuals. Using the PACF we can find the number of lags after which a residual has no longer significant influence on successive residuals. This number is then used as parameter to correctly specify the HAC covariance matrix in a second round of fit.

Assumption 5 shall assure that no independent variable is a linear combination of one or more other independent variables. However, we note that with our approach the workload components, i.e. request size and reqeust rate – or interarrival times – are highly correlated. Typically, the Pearson correlation coefficient between request size and interarrival time is $> 0.95$. For this reason we are also limited to models that consider only one component of the workload: request size *or* request rate where request size is more meaningful in our cases. In principle, the request rate could be factored into the models if we interpret request rate as a categorical variable. This, however, leads to problems if a fitted model is used on new data: If the new data contains a request rate that was not present in the fitted model the prediction fails and therefore the dynamics of request rate cannot be reflected by the models. This puts a serious limitation on the general applicability of the approach and we are thus limited to static models, i.e. fixed request rates. For the following considerations we hence focus on the data for entire 24-hour periods contrary to single hours.

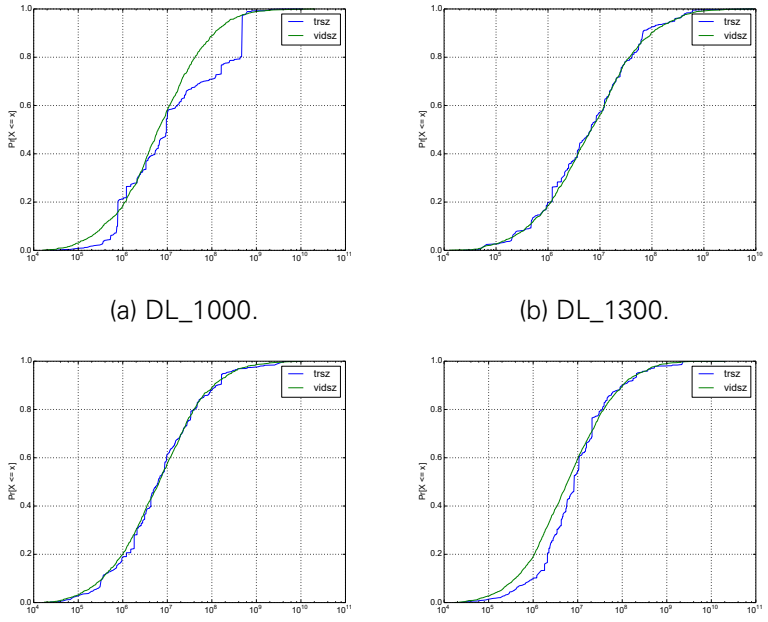
## 4.5. MODELS FOR RESOURCE UTILIZATION AS FUNCTION OF REQUEST SIZE

Now we will derive models that describe the resource utilization of the respective services as function of the request size. While memory is not a negligible resource, with our toolset we are currently only able to evaluate the utilization of memory in terms of occupied space. But the mere amount of memory taken by a specific application barely reflects its

"utilization". To make statements in that regard we would need to observe memory traffic, e.g. misses in the last level cache (LLC). While this is feasible by accessing PMCs and count LLC miss PMEs our current tool to read and record resource utilization statistics, `dstat`, does not offer such functionality. And the standard tool to read PMCs, `perf`, does not produce output that could be used to obtain empirical distribution functions for, say, Bytes read from or written to memory per time interval[6]. We thus focus on the CPU utilization, followed by the storage utilization, and network utilization. In advance to derive resource utilization models we demonstrate the abovely described approach using the example of request size and file size.

### 4.5.1. REQUEST SIZE

To be reminded, the distribution of request sizes in the video server scenario mainly depends on the distribution of server-side file sizes. We therefore try to express the request size distribution as a function of the file size distribution. In Figure 4.4 we first show the EDFs of the file sizes versus the EDFs of request sizes for the different tests.




(a) DL_1000. (b) DL_1300.

(c) DL_1600. (d) DL_1900.

Figure 4.4.: EDFs for the request size (trsz) and the file size (vidsz) for test cases DL_1000, DL_1300, DL_1600, DL_1900.

While in the graphs the EDFs for file size and request size are close to each other, smaller or larger deviation can be observed in Figure 4.4a, and 4.4d. These deviations are caused by variation in the request size EDFs since the distribution of file sizes is nearly the same in all test cases - yet with small deviations - as can be seen in Figure 4.5. We believe that these deviations are attributable to imperfect calculations of additional views (i.e. popularity dynamics). Remember that the start popularity records a video's past number of views up to the time point of starting the experiment and that it serves as parameter to compute the additional number of views a video will gain over the next week (see Section 3.3.1.3). The number of additional views is independent of the file size. But since the growth rate factor

[6] It came only very late to the author's attention that the Intel Performance Counter Monitor tool offers just the desired functionality, see `https://software.intel.com/en-us/articles/intel-performance-counter-monitor-a-better-way-to-measure-cpu-utilization`, 2015-09-02.

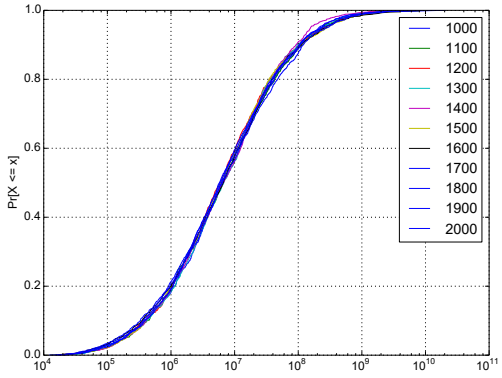
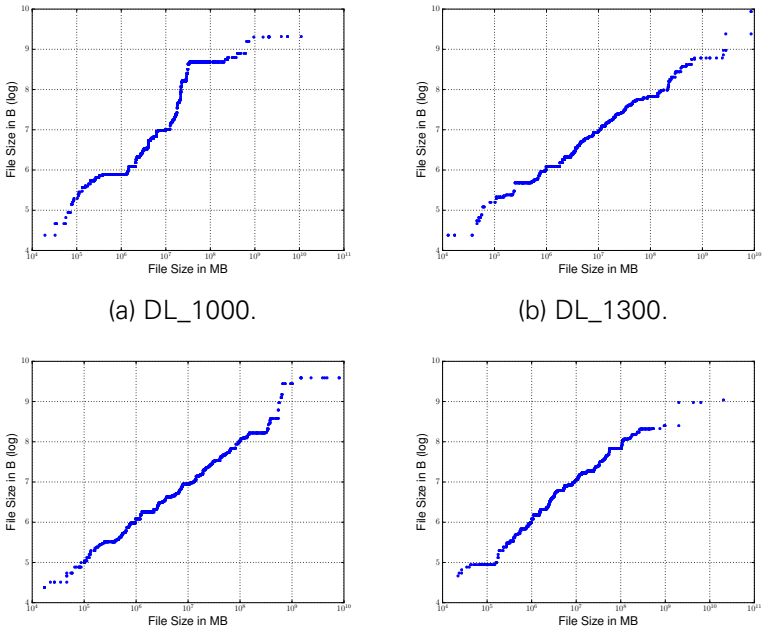



Figure 4.5.: File size distribution for test cases DL_1000 to DL_1900.

is a random variable it cannot be excluded that files of a certain size gain comparatively more views. In this case, the request size distribution will not be equal in distribution to the file size distribution.

However, in all cases the superimposition of file size and request size graphs suggests a linear relationship between the EDFs. This is particularly visible in the QQ curves shown in Figure 4.6. We employ the approach described in Section 4.4.1 to initially estimate the




(a) DL_1000. (b) DL_1300.

(c) DL_1600. (d) DL_1900.

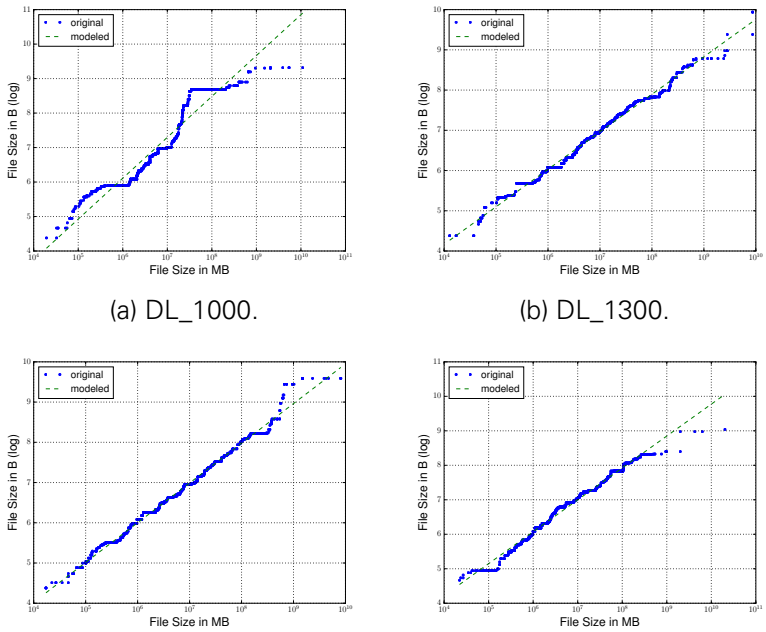
Figure 4.6.: QQ curves for the request size (trsz) and the file size (vidsz) for test cases DL_1000, DL_1300, DL_1600, DL_1900.

quantiles of the request size by the quantiles of the file size with the simple regression model

$$lg(\hat{Q}_{S_r}(q)) = \beta_V lg(Q_V(q)) + \beta_0 \tag{4.57}$$

where $\hat{Q}_{S_r}(q)$ denotes the estimated, and $Q_V(q)$ the empirical quantile functions for the request size and file size, respectively. The logarithmic transform accounts for the fact that the distributions of request sizes and file sizes are heavy-tailed and thus, show large

variance. We find that values $\beta_0 = 0.471$ and $\beta_V = 0.985$ fits the data best on average across all test cases. Please note that distinct file size distributions and distinct popularity distributions exist for each testcase, yet the regression coefficient is close to $1$ in all cases. Figure 4.7 shows the fitted QQ curves against the actual QQ curves. It is clear to see that




(a) DL_1000. (b) DL_1300.

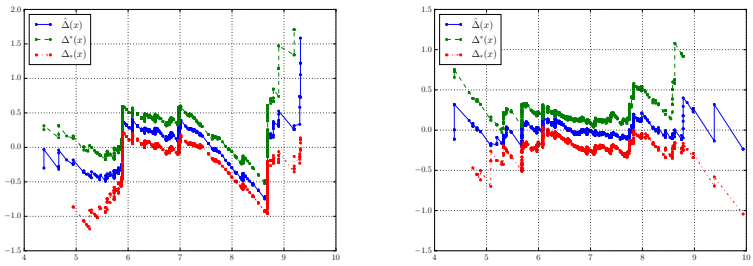
(c) DL_1600. (d) DL_1900.

Figure 4.7.: Estimated QQ curves for the logarithm of the request size (trszlog10) and the file size (vidsz) for test cases DL_1000, DL_1300, DL_1600, DL_1900.

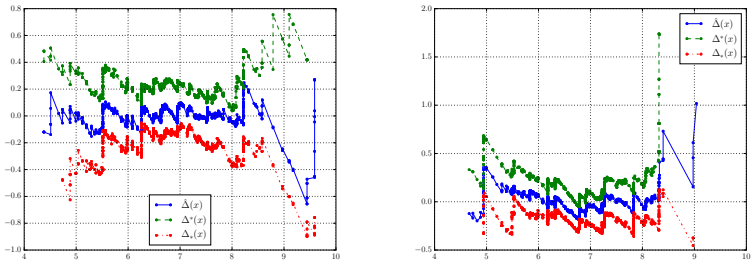
except in Figure 4.7a the linear model fits the QQ curve very well. Figure 4.7 demonstrates that the file size distribution can be indeed taken as estimate of the request size distribution and the server operator can know the request size distribution by merely calculating the distribution function of the file sizes despite the fact that the sample of request sizes is effectively a multiset whose support is the set of file sizes and the multiplicity of each element is determined by the popularity distribution. The less good fit in Figure 4.7a is expected and explained above, yet the model in Equation (4.57) appears to be a good approximation.

Next, we examine the deviations between the actual and estimated EDFs. For the above test cases Figure 4.8 shows the associated shift functions together with the $\alpha = 0.05$ level W bands. While in Figs. 4.8b to 4.8d the empirical shift function fluctuates near zero the larger deviations in the DL_1000 test case are apparent in Figure 4.8a. In all cases, however, the upper and lower confidence band do not allow the assumption of a linear shift, in particular the assumption of a perfect fit. This is due to the dependency of the efficiency of the confidence band on the sample size as explained above (see e.g. Figure 4.3). If we would have chosen a sample size of 500 the assumption of a perfect fit could not be rejected for DL_1300, DL_1600, and DL_1900 while it would be still rejected for DL_1000 as Figure 4.9 demonstrates. Yet we maintain a sample size of 1000 since we want to keep as much information as possible about the actual data.

Finally, we summarize the MAPE and medAPE for all test cases in Table 4.4. In all cases the MAPE and its median version are below $4.1\%$ and across test cases the average MAPE is $1.593$ which assures that Equation (4.57) is a good model for the data and hence the request size distribution can be sufficiently well estimated by the video size distribution in our scenario. Also, this result (a) demonstrates that the popularity distribution merely

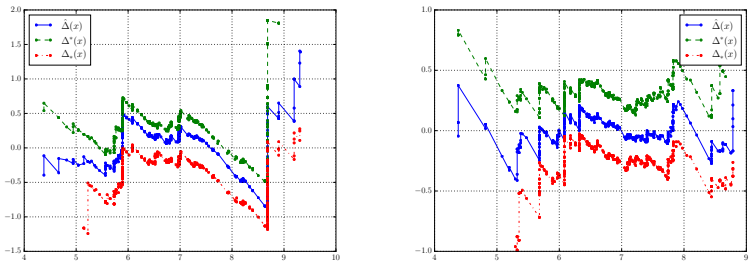


(a) DL_1000. (b) DL_1300.

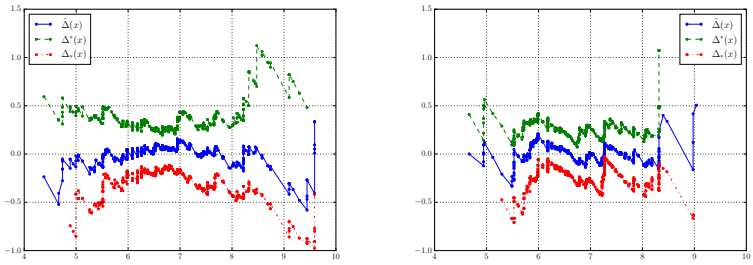


(c) DL_1600. (d) DL_1900.

Figure 4.8.: Shift functions for the deviation between EDFs for the logarithm of the request size and its estimate as function of the EDFs of the respective file size (vidsz) for selected test cases.



(a) DL_1000. (b) DL_1300.



(c) DL_1600. (d) DL_1900.

Figure 4.9.: Shift functions for the deviation between EDFs for the request size (trsz) and its estimate as function of the EDFs of the respective file size (vidsz) for selected tests with sample size 500; cf. Figure 4.8.

influences the request size distribution in the video server scenario and thus (b) suggests that a server operator can indeed predict the statistical distribution of request sizes by merely considering the distribution of file sizes.

| Video Server Test Case | MAPE | medAPE |
|---|---|---|
| DL_1000 | 4.037 | 3.835 |
| DL_1100 | 1.408 | 0.755 |
| DL_1200 | 1.743 | 1.357 |
| DL_1300 | 0.995 | 0.789 |
| DL_1400 | 1.185 | 0.937 |
| DL_1500 | 1.049 | 0.957 |
| DL_1600 | 0.799 | 0.587 |
| DL_1700 | 1.751 | 1.513 |
| DL_1800 | 1.814 | 1.589 |
| DL_1900 | 1.149 | 0.908 |

Table 4.4.: Percentage errors for $lg(\hat{Q}_{S_r}(q)) = 0.985 \cdot lg(Q_V(q)) + 0.471$

The approach seems therefore in principle applicable and in the next steps we will estimate the utilization of the CPU, storage, and network as functions of the workload.

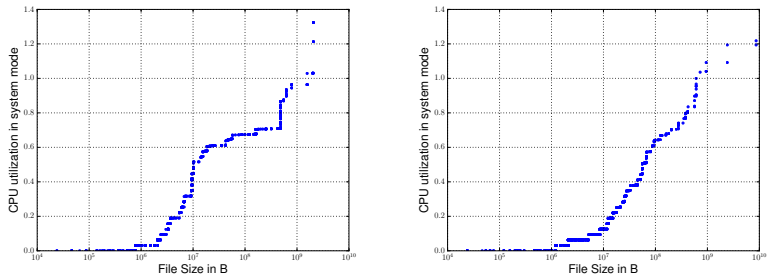
### 4.5.2. CPU UTILIZATION

Since the request size is to be understood differently depending on the application, and since resource utilization for some workload depends on the behavior of the very application, such models can only be application-specific. To find the best parameter setting per application we train the model with one test case and evaluate it against all other test cases. We repeat this procedure with the remaining test cases and finally select the parameter setting that minimizes the error across all test cases. Thus, we determine the test case that is most representative for all the others. Since the only parameter in the models is the request size (distribution) this can also be understood as the question for which request size distribution is the most representative.

#### 4.5.2.1. VIDEO SERVER

In the previous demonstration the relationship between the file size distribution and the request size distribution was clearly linear with the exception of test case DL_1000. But the linear assumption does no longer hold in the case of CPU utilization as we can see in Figure 4.10. A general transform of the data is hard to determine: Figure 4.10a and Figure 4.10c resemble a $tanh$ function for most of the data points, but show an exponential increase in the upper quantiles. A compressed $tanh$ function can also be seen in Figure 4.10b and Figure 4.10d, but these two QQ curves also resemble stretched quadratic or subexponential functions. The constant line at zero CPU utilization in the lower parts of the QQ curves corresponds to situations where the requested file was already present in the main memory. Although file popularity is independent of the file size more smaller files reside in the main memory simply due to the heavy-tailed file size distribution.

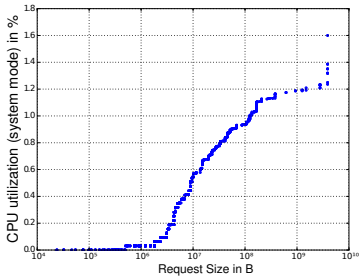
Across test cases the $tanh$ transform appears to be the more commonly applicable transform. But this transform necessitates a second transform, namely a shift of the request size data points since the $tanh$ is point symmetric around zero. In an offline step



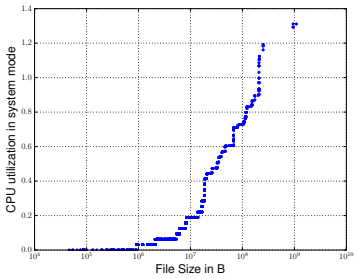

(a) DL_1000.




(b) DL_1300.




(c) DL_1600.




(d) DL_1900.

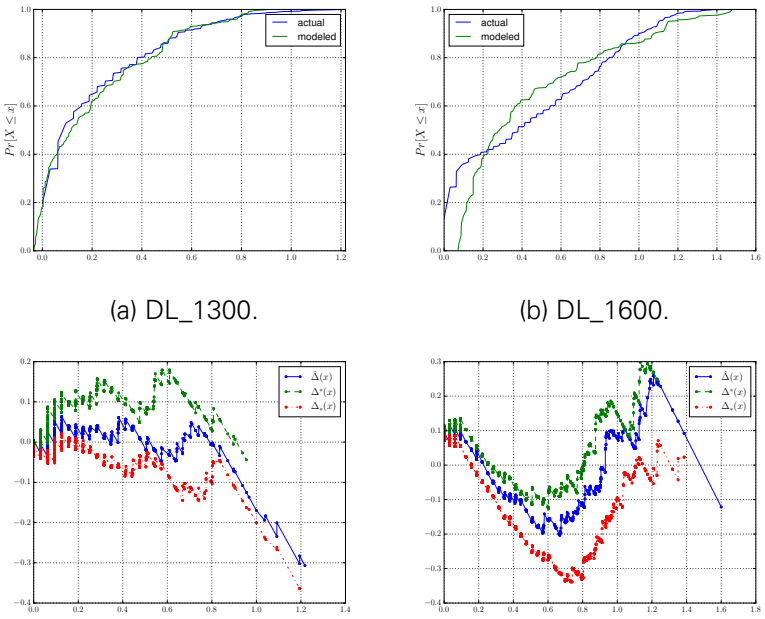
Figure 4.10.: QQ curves for the CPU utilization in system mode vs. the request size for test cases DL_1000, DL_1300, DL_1600, DL_1900.

we performed an exhausitive search over the $0.05(0.05)0.95$ quantiles and examined the relationship between the quantiles, the MAPE, and the request size. We find that on average a shift of the data by the $0.8$ quantile of the logarithm of the request size produces the best results. We thus estimate the CPU utlization quantiles, $\hat{Q}_C(q)$, as a function of the request size quantiles, $\hat{Q}_{S_r}(q)$, with the following model:

$$\hat{Q}_C(q) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.8))) + \alpha \ . \tag{4.58}$$

Across all test cases the parameter values $\alpha = 0.4069$ and $\beta = 0.3503$ minimize the errors. In Figure 4.11 we show the estimated EDFs together with the shift functions. As Figure 4.11a and Figure 4.11b demonstrate the $tanh$ transform is a good approximation of the request size quantiles. However, the exponential part of the QQ curve for DL_1300 is not captured which is reflected by the larger deviation in the right parts of the shift function plots Figure 4.11c: While in the left part the shift function tends to fluctuate around or near zero it shows a larger deviation in the right part, i.e. the upper quantiles, since the exponential part of the QQ curves is not accounted for by the data transformation. More complex functions need to be examined in order to find a suitable transformation. In particular, the relationship between base and exponent of the exponential part, and the properties of the request size distribution and system properties as well as the dependency of the breakpoint between $tanh$ and $exp$ parts and the request size distribution have to be investigated. Due to time restrictions such investigation has to be deferred to future work.

Since the actual data contains many values of zero CPU utilization the mean percentage errors do not exist. Thus, only the medAPE values are obtainable which we summarize in Table 4.5 below. These values are not too encouraging. In particular the medAPEs indicate that the estimate always deviates more than 15% from the actual value and more than 64% in testcase DL_1600. This pronounces again the inflation of the MAPE: The actual data contains almost only values $< 1.0$ and therefore even very modest deviations in the estimate have a large percentage effect.




(a) DL_1300. (b) DL_1600.

(c) DL_1300. (d) DL_1600.

Figure 4.11.: Estimated and actual EDFs for the CPU utilization in system mode for test cases DL_1300 and DL_1600 and associated shift function plots.

| Video Server Test Case | medAPE |
|---|---|
| DL_1000 | 58.035 |
| DL_1100 | 31.974 |
| DL_1200 | 33.440 |
| DL_1300 | 16.798 |
| DL_1400 | 30.719 |
| DL_1500 | 50.882 |
| DL_1600 | 64.249 |
| DL_1700 | 44.825 |
| DL_1800 | 25.757 |
| DL_1900 | 23.095 |

Table 4.5.: Percentage errors for $\hat{Q}_C(q) = 0.3503 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.8))) + 0.4069$

#### 4.5.2.2. TRANSCODING SCENARIO

In the previous model, Equation (4.58), we find that the relationship between the CPU utilization quantiles and the request size quantiles follows approximately a $tanh$ function. We observe that in the transcoder scenario this observation holds as well. But while on average the best left shift in the previous scenario is the $.8$ quantile of the logarithm of the request size we find that in the present scenario it is the $.85$ quantile independent of the target container and arrival rate. We would have assumed that since alternative target

containers utilize the CPU differently the necessary left shift would show a dependence on whether the video is transcoded to the FLV or MKV container. Apparently, this is not the case. We model the CPU utilization therefore as follows:

$$\hat{Q}_C(q) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.85))) + \alpha. \tag{4.59}$$

Still, it must be noted that the two different target containers need to be understood as two different applications and that therefore two application-specific models are trained. Since at some points the CPU utilization is zero in the actual data the MAPE does not exist and we base our decision to select the best model on the medAPE values. For the FLV test cases the best results are obtained for the parameter setting $\beta = 13.0552$, and $\alpha = 13.9436$ while for the MKV container the values $\beta = 26.3386$ and $\alpha = 29.7865$ minimize the error. The respective error values are summarized in table Table 4.6.

| Transcoder Test Case | medAPE | MSE |
|---|---|---|
| FLV_180 | 34.369 | 3.058 |
| FLV_360 | 64.966 | 52.150 |

(a) FLV test cases.

| Transcoder Test Case | medAPE | MSE |
|---|---|---|
| MKV_180 | 47.577 | 21.739 |
| MKV_360 | 65.881 | 258.628 |

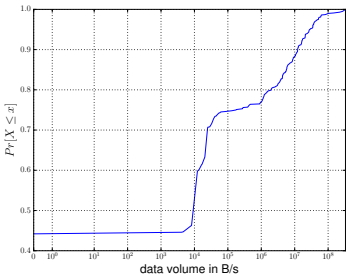
(b) MKV test cases

Table 4.6.: Percentage errors for $\hat{Q}_C(q) = 13.0552 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.85))) + 13.9436$ (a), and $\hat{Q}_C(q) = 26.3386 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.85))) + 29.7865$ (b).

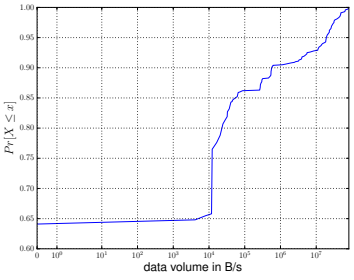
Similar to the video server scenario the error values are very large due to the presence of many small values in the actual data. Also, the error increases with the request rate since we cannot account for this workload component.

### 4.5.3. STORAGE UTILIZATION

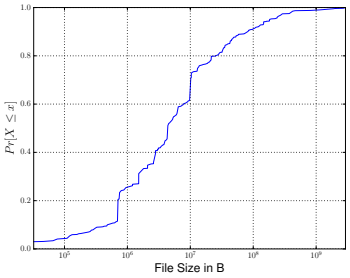
In the following we estimate the quantiles of the storage utilization as function of the quantiles of the request size. In order to apply linear regression we need to transform the request size quantiles as before. Again we find that a $tanh$ transform is a good approximation. The reason that the QQ plot follows approximately this function is revealed if we consider the EDF plots of the storage utilization and the request size together with the actual and estimated QQ curves in Figure 4.12: In case of DL_1100 zero bytes were written to, or read from, the storage for approximately 45% of the time (Figure 4.12a); in case of TC_MKV_180 the storage utilization is zero for approximately 60% of the time (Figure 4.12b). The $.45$-quantile of the request size in case of DL_1100 is approximately $10^{6.35}$ and the $.60$-quantile of the request size in case of TC_MKV_180 is approximately $10^7$. Therefore the lower parts of the QQ curves in Figure 4.12e and Figure 4.12f increase for request size while they stay constant for the storage utilization. The sudden increase in Figure 4.12a and Figure 4.12b around $10^4$ is caused by the disk system's block size and the RAID structure: The minimum amount of data that can be read from or written to a disk at one time is 4096 Bytes and the RAID system on the server comprises three disks. The jump at $10^4$ thus reflects the situation that the minimum block size is read from or written to the three disks in parallel: $lg(3 \cdot 4096) = 4.0895$. The relationship between the upper quantiles in Figure 4.12e and Figure 4.12f is due to the saturation of the storage: While the maximum request size in DL_1100 is approximately $10^{9.94}$ bytes and ca. $10^{8.16}$ bytes in case of TC_MKV_180, the maximum storage utilization is $10^{8.54}$ bytes/sec and $10^{8.05}$ bytes/sec, respectively. The larger difference in case of DL_1100 also explains why the QQ curve flattens more than in case of TC_MKV_180.



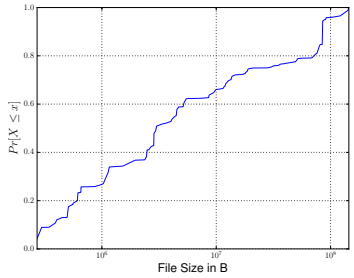

(a) storage utilization, DL_1100.



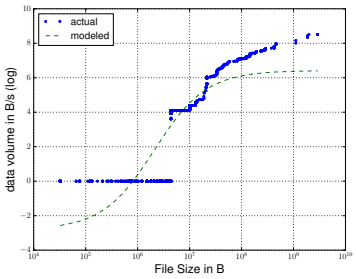

(b) storage utilization, TC_MKV_-180.



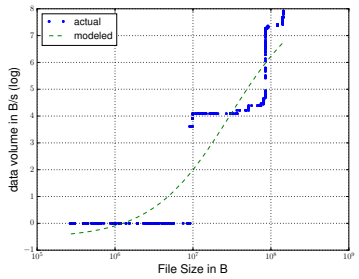

(c) request size, DL_1100.




(d) request size, TC_MKV_180.




(e) QQ curves, DL_1100.




(f) QQ curves, TC_MKV_180.

Figure 4.12.: EDFs of storage utilization and request size, and estimated and actual QQ curves for test cases DL_1100 and TC_MKV_180.

#### 4.5.3.1. VIDEO SERVER

For the video server scenario we find that the necessary left shift of the data in order to make the $tanh$ transform applicable follows the $0.5$-quantile of the logarithm of the request size. The model for the storage utilization quantiles, $Q_D$, in this scenario is therefore

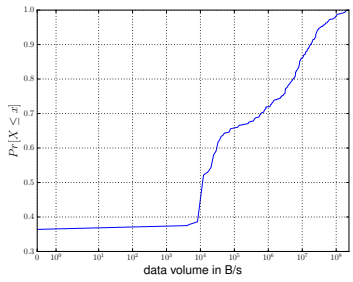
$$\hat{Q}_D(q) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.5))) + \alpha. \tag{4.60}$$

We find that with respect to the MAPE the parameter values $\alpha = 1.6086$ and $\beta = 6.8519$ minimize the error values across the test cases. The respective mean and median percentage errors are summarized in table Table 4.7. Due to several values being equal zero in the actual data the MAPE does not exist. And in cases DL_1000 and DL_1400 the storage utilization was zero for more than $50\%$ of the time. Thus, also medAPE values do not exist. The low storage utilization is due to caching of the requested files: In test case DL_1000 only in one out of the 24 slots (i.e. hours) the requested files did not entirely fit into the main memory and the RAID system was utilized. And in DL_1400 all requested files fit into the main memory over the entire 24-hour period. Hence, the files needed to be read from the disk only once and were served to the client entirely from the main memory afterwards.

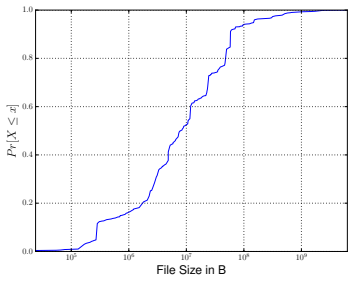
| Video Server Test Case | medAPE | MSE |
|---|---|---|
| DL_1000 | $NaN$ | 4.359 |
| DL_1100 | 17.847 | 2.170 |
| DL_1200 | 32.893 | 4.092 |
| DL_1300 | 15.468 | 1.104 |
| DL_1400 | $NaN$ | 9.065 |
| DL_1500 | 15.603 | 1.611 |
| DL_1600 | 30.636 | 3.232 |
| DL_1700 | 30.475 | 2.438 |
| DL_1800 | 19.568 | 1.950 |
| DL_1900 | 23.468 | 2.402 |

Table 4.7.: Percentage errors for $\hat{Q}_D(q) = 6.8519 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.5))) + 1.6086$

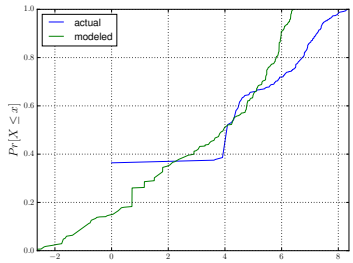
For a better understanding the relationship between the EDFs of storage utilization and request size and the QQ curve are given in Figure 4.13. To complete the picture the shift function is given as well.



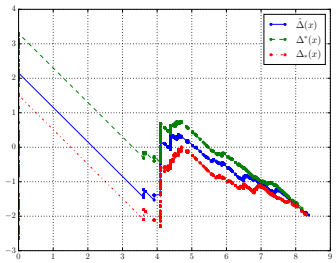

(a) storage utilization, DL_1300.




(b) request size, DL_1300.




(c) EDFs, DL_1300.



(d) shift function, DL_1300.

Figure 4.13.: EDFs of storage utilization and request size, estimated and actual QQ curves, and shift function for test case DL_1300.

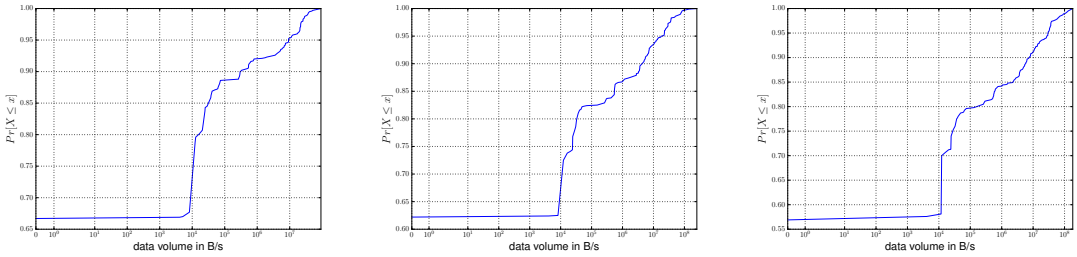
We note that about $42\%$ of the time the storage utilization is zero (Figure 4.13a) and the $.42$-quantile of the request size is $\approx 10^{6.3}$ (Figure 4.13b). Thus, the QQ curve is constant at zero up to $10^{6.3}$ (Figure 4.13c). The steep increase in Figure 4.13a around $10^4$ up to approximately the $.6$-percentile is reflected by the second constant line around $4$ in

Figure 4.13c. The deviation between the actual and the estimated quantiles is shown in Figure 4.13d where we need to pay particular attention to the vertical lines around $4$ and $0$: Such lines represent that many actual quantiles are modeled by just a single (estimated) quantile.

Judging by the error values in Table 4.7 we must conclude that our approach is also not well suited to model the storage utilization in the video server scenario. More investigation is necessary to examine whether the model errors can be improved if the left shift of the data should follow a dynamic rather than a fixed quantile of the logarithm of the request size and if so, how this dynamic quantile is related to other features of the workload (e.g. request rate or file popularity). Alternatively, one could consider segmented regression where breakpoints (or change-points) can be automatically detected using methods from change-point detection in time series analysis; see e.g. (Aue and Horváth 2013) or (Brodsky and Darkhovsky 2013).

#### 4.5.3.2. TRANSCODING SCENARIO

For the transcoding scenario we must note that storage utilization considered over time is also very low. Indeed, in all testcases the utilization was zero for more than $50\%$ of the time as can be seen in Figure 4.12b and Figure 4.14. The low storage utilization is to be




(a) TC_FLV_180. (b) TC_FLV_360. (c) TC_MKV_360.

Figure 4.14.: EDFs of storage utilization for selected test cases in the transcoding scenario.

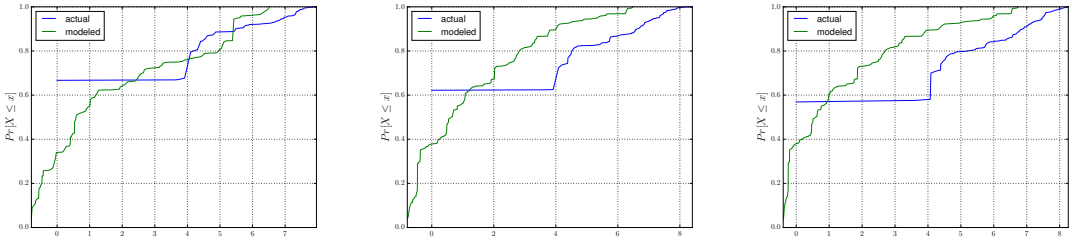
attributed to the low request arrival rates which are $0.05$ and $0.1$ requests per second (one request every 20 and every 10 seconds, respectively). Surely, a higher request rate will lead to a higher storage utilization since caching plays no role in this scenario. However, these values were chosen as upper bounds on request rates that would be expected in real systems (cf. Section 3.2.7 and Section 3.3.2.3). We find that the left shift of the data is best described by the $0.85$-quantile of the logarithm of the request size. We thus apply the following model to estimate the quantiles:

$$lg(\hat{Q}_D(q)) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.85))) + \alpha \; . \tag{4.61}$$

For the FLV container test cases the parameter values $\alpha = 21.7899$ and $\beta = 25.0529$ minimize the MSEs across the test cases and for the MKV test cases the respective values are $\alpha = 16.2004$ and $\beta = 18.8689$. In Figure 4.15 the actual and estimated EDFs are shown. Similar to the CPU model the error increases with the request rate which again demonstrates the limitation of our approach.

### 4.5.4. NETWORK UTILIZATION

Finally, we turn to the utilization of the NIC. The QQ curves show a similar relationship between resource utilization and request size as before in the case of CPU and storage. Two examples are given in Figure 4.16 that also reveal specific properties of the network




(a) TC_FLV_180. (b) TC_FLV_360. (c) TC_MKV_360.

Figure 4.15.: EDFs of storage utilization for selected test cases in the transcoding scenario.

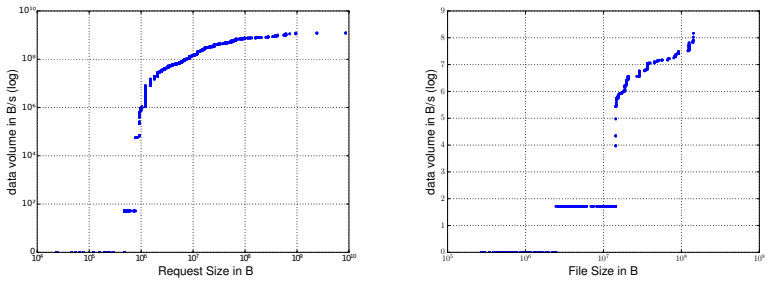
| Transcoder Test Case | MSE |
|---|---|
| FLV_180 | 0.762 |
| FLV_360 | 1.949 |

(a) FLV test cases.

| Transcoder Test Case | MSE |
|---|---|
| MKV_180 | 0.721 |
| MKV_360 | 2.668 |

(b) MKV test cases

Table 4.8.: Estimation errors for $lg(\hat{Q}_D(q)) = 25.0529 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.85))) + 21.7899$ (a), and $lg(\hat{Q}_D(q)) = 18.8689 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.85))) + 16.2004$ (b).




(a) DL_1300. (b) TC_MKV_360

Figure 4.16.: QQ curves for test cases DL_1300 and TC_MKV_360.

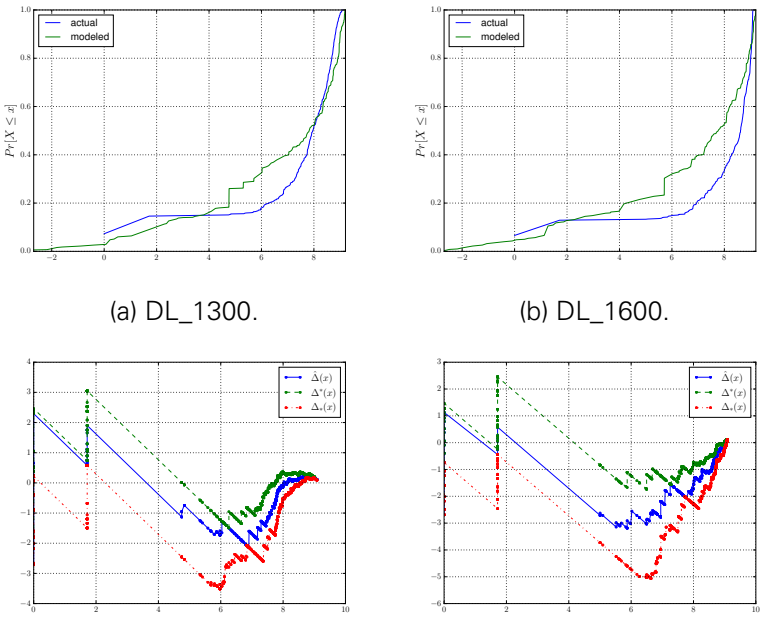
utilization and NIC. The constant lines at zero network utilization are caused by the fact that 10% of the time the network utilization is zero in the DL_1300 case and the .1-quantile of the request size is approximately $10^{5.3}$; and in the TC_MKV_360 case the network utilization is zero in approximately 37.5% of the time and the .375-quantile of the request size is ca. $10^{6.15}$. The second constant line is at a network utilization of approximately $10^{1.716} = 52$ bytes per second. This coincides with the size of empty UDP datagrams. However, none of the applications in our testbed or any connected devices are set to send empty UDP datagrams. Also, no other services or deamons on the video server or the client open a UDP port. Since the testbed is not isolated and part of our chair's network this obsevation could thus be a hint that some computer in the network is scanning UDP ports which usually means to send empty UDP datagrams. The relationship between the upper quantiles reflect the saturation of the network device and statements apply similar to those made before in Section 4.5.3. As well as similar to the previous cases we employ a $tanh$ function to approximate the course of the QQ curves.

#### 4.5.4.1. VIDEO SERVER

In the video server scenario we find that the necessary left shift of the data follows the .2-quantile of the logarithm of the request size. We thus train the model

$$lg(\hat{Q}_N(q)) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.2))) + \alpha \tag{4.62}$$

in order to estimate the quantiles of the network utilization, $Q_N$. The best results with regard to the medAPE across all test sets are obtained for parameter values $\alpha = 4.5503$ and $\beta = 5.5502$. In Figure 4.17 we present the actual and estimated EDFs together with the shift functions for test cases DL_1300 and DL_1600. Again, we notice that the constant




(a) DL_1300. (b) DL_1600.

(c) DL_1300. (d) DL_1600.

Figure 4.17.: Estimated and actual EDFs for the logarithm of the network utilization for test cases DL_1300 and DL_1600 and associated shift function plots.

lines in the QQ curves are reflected by the vertical lines in the shift function and that these lines are not well met by the $tanh$ function. Yet the $tanh$ approximation follows the actual QQ curve in general which – alternatively – is also reflected by the shift functions: Most of the data points of the shift function (dots on the solid blue line) concentrate in the range $[-1, 1]$ on the y-axis. By means of the medAPEs values in Table 4.9 we can see that for at least half of the network utilization quantiles the model performs very well. Due to the sensitivity of the MAPE to zero values in the actual data, we cannot assess the MAPE. As before, the relationship between the best values of the necessary left shift of the data and other workload or system properties should be examined thoroughly in order to improve model accuracy. Also, other functions to describe the QQ curve should be taken into consideration. E.g. the upper quantiles of the QQ curve in Figure 4.16a follow a logarithmic curve. The lower quantiles can be described by two constant functions. The crucial part is the adequate description of the break points between these functions. Without the opportunity to include the request rate in our model we currently see no way to achieve this goal. The presence of empty UDP datagrams in the observed resource utilization renders this task even harder since the percentage of these datagrams in the data sample is beyond our control. In case the perpetrating computer is sending datagrams with a fixed frequency one could then compute its percentage relative to the length of the period during

| Video Server Test Case | medAPE | MSE |
|---|---|---|
| DL_1000 | 6.518 | 1.897 |
| DL_1100 | 3.650 | 0.879 |
| DL_1200 | 3.059 | 0.308 |
| DL_1300 | 3.533 | 0.749 |
| DL_1400 | 2.744 | 0.462 |
| DL_1500 | 5.137 | 0.579 |
| DL_1600 | 8.459 | 1.052 |
| DL_1700 | 6.247 | 2.213 |
| DL_1800 | 3.907 | 0.909 |
| DL_1900 | 2.679 | 0.601 |

Table 4.9.: Percentage errors for $lg(\hat{Q}_N(q)) = 5.5502 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.2))) + 4.5503$

which the network device is observed and the average request rate. At the current state we must defer such investigation to future work.

#### 4.5.4.2. TRANSCODING SCENARIO

In the transcoding scenario we find that similar to the storage utilization in the transcoding scenario the left shift is best described by the $0.75$-quantile of the logarithm of the request size. Hence, the resulting model is defined as

$$lg(\hat{Q}_N(q)) = \beta \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{lg(S)}(0.75))) + \alpha \; . \tag{4.63}$$

In case the target container is MKV the best results are obtained for parameter values $\alpha = 2.9036$ and $\beta = 5.0971$. For the FLV target container the best results are obtained for $\alpha = 3.7450$ and $\beta = 6.8989$. The values for the estimation errors are given in Table 4.10 below.

| Transcoder Test Case | medAPE | MSE |
|---|---|---|
| FLV_180 | 154.433 | 1.282 |
| FLV_360 | 63.328 | 3.559 |

(a) FLV test cases.

| Transcoder Test Case | medAPE | MSE |
|---|---|---|
| MKV_180 | 135.340 | 1.505 |
| MKV_360 | 64.803 | 3.404 |

(b) MKV test cases

Table 4.10.: Percentage errors for $\hat{Q}_N(q) = 6.8989 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.75))) + 3.7450$ (a), and $\hat{Q}_N(q) = 5.0971 \cdot tanh(lg(\hat{Q}_{S_r}(q) - Q_{S_r}(0.75))) + 2.9036$ (b).

Similar as for the storage utilization model in the transcoder scenario we must conclude that our approach is not suitable to model the utilization of the network for this scenario with the given workload characteristics.

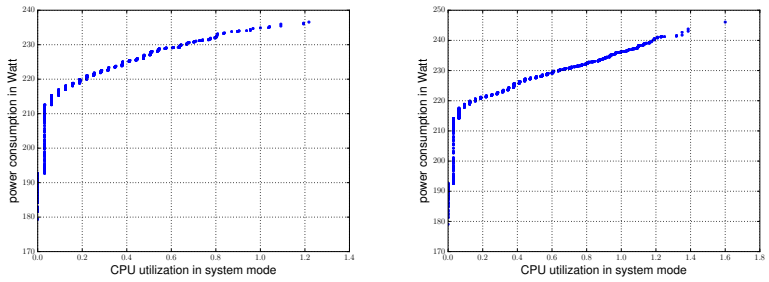
## 4.6. MODELS FOR POWER CONSUMPTION AS FUNCTION OF RESOURCE UTILIZATION

After the resource utilization models we apply our approach to estimate power consumption quantiles as function of resource utilization quantiles.

In Section 4.2.2.2 we layed out that we will focus on CPU utilization-based models. However, attention must be paid to the nature of the applications (video server, transcoder, benchmarks) and in which mode the applications use the CPU mainly: Unix-like systems distinguish between six different modes of CPU utilization: *user* mode (`usr`) where user applications are executed, *system* mode (`sys`) where the CPU executes operating system functions, *wait* mode (`wai`) where the CPU waits for I/O devices like NIC and storage, *idle* mode (`idl`) where the CPU does nothing, and two modes where the CPU handles software and hardware interrupts (`siq`, `hiq`). Since the different modes are orthogonal to each other we can sum the utilization in modes `usr`, `sys`, `wai`, `hiq`, and `siq`. We name this the *non-idle* CPU utilization. An I/O-intensive application like the video server mainly utilizes the CPU in wait and system mode, whereas the transcoder and the benchmarks utilize the CPU mainly in user mode. This knowlegde needs to be exploited for the power consumption models below.

### 4.6.1. VIDEO SERVER

As mentioned above in the video server scenario the CPU is utilized mainly in system and wait mode. We obtain the better results if the model takes CPU utilization in system mode as input and for the sake of space we therefore focus on system mode in the following. Differently to the resource utilization models the relationship in the QQ curve follows no longer a $tanh$ function as it is exemplified in Figure 4.18.




(a) DL_1300. (b) DL_1600

Figure 4.18.: QQ curves for power consumption vs. CPU utilization in system mode for test cases DL_1300 and DL_1600.

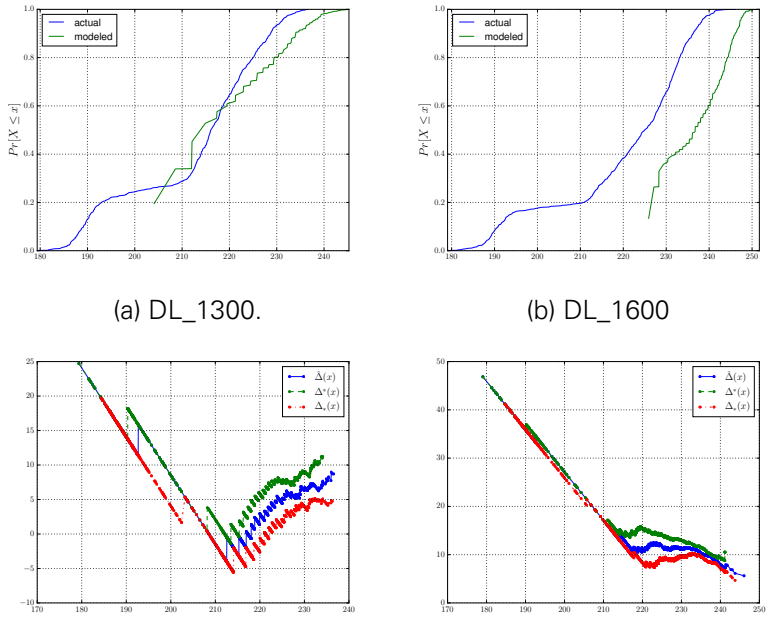
Apparently, the QQ curve follows a linear function if we consider power consumption above $210$ Watt. This value represents the maximum power consumption if the server is idle[7]. To model values also below $210$ Watt a logarithmic function seems appropriate. But the vertical lines at the left edge of the QQ curve also indicate that CPU utilization is zero for a large part of the values (cf. Figure 4.10). We therefore need to shift the CPU utilization quantiles slightly to the right to be able to take the logarithm. We find that the median of the CPU utilization is sufficient and that this definition of the right-shift is also generally applicable in the other scenarios. The model for the power consumption quantiles in this

[7] Idle power consumption of the server fluctuates between $180$ and $210$ Watt due to uncontrollable scheduling of background services.

scenario is thus

$$\hat{Q}_P(q) = \beta \cdot lg(\hat{Q}_{C_{sys}}(q) + \hat{Q}_{C_{sys}}(.5)) + \alpha. \tag{4.64}$$

We find that the best results are obtained across test cases for the parameter values $\alpha = 234.9086$ and $\beta = 36.0582$. The actual and estimated EDFs for the test cases DL_1300 and DL_1600 are shown in Figure 4.19.




(a) DL_1300. (b) DL_1600

(c) DL_1300. (d) DL_1600

Figure 4.19.: Actual and estimated EDFs for power consumption vs. CPU utilization in system mode for test cases DL_1300 and DL_1600.

We summarize the corresponding error values in Table 4.11.

| Video Server Test Case | MAPE |
|---|---|
| DL_1000 | 4.913 |
| DL_1100 | 3.203 |
| DL_1200 | 2.712 |
| DL_1300 | 2.626 |
| DL_1400 | 2.579 |
| DL_1500 | 3.375 |
| DL_1600 | 4.765 |
| DL_1700 | 2.236 |
| DL_1800 | 2.353 |
| DL_1900 | 2.648 |

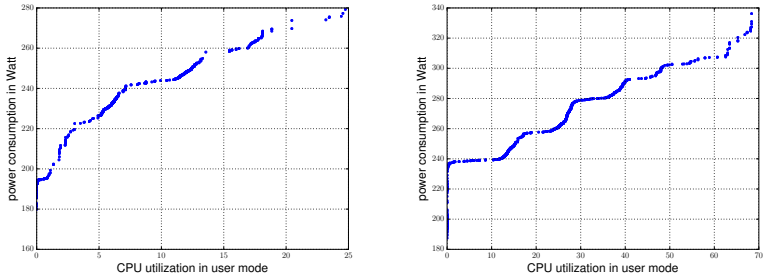
Table 4.11.: Percentage errors for $\hat{Q}_P(q) = 36.0582 \cdot lg(\hat{Q}_{C_{sys}}(q) + \hat{Q}_{C_{sys}}(.5)) + 234.9086$.

Contrary to the MAPEs values for the resource utilization models these values indicate

that our approach is in principle applicable if the relationship between quantiles is not too complex. Admittedly, the fact that power consumption is never zero, but always above $150\%$ surely contributes to the small percentage errors.

### 4.6.2. TRANSCODING SERVICE

In the transcoding scenario we take the CPU utilization in user mode as model input. Similar to the previous scenario, the relationship between the quantiles is approximately linear in the upper quantiles – in this scenario for power consumption values above $220$ Watt – but considerably many power consumption quantiles are mapped against a CPU utilization of $0$ which can be easily seen in Figure 4.20 below.



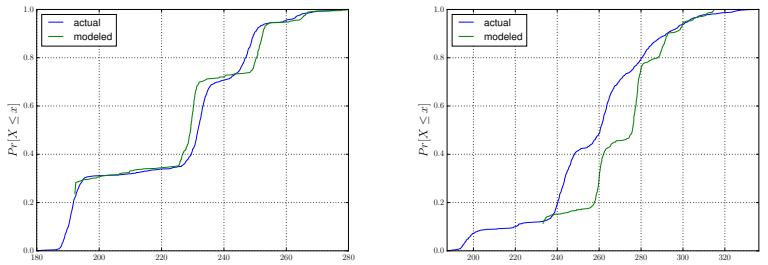

(a) TC_FLV_180. (b) TC_MKV_360

Figure 4.20.: QQ curves for power consumption vs. CPU utilization in user mode for test cases TC_FLV_180 and TC_MKV_360.

As before, we apply a logarithmic transform to the CPU utilization values and also shift these quantiles to the right following the median of the values. Consequently, we train the model in Equation (4.65) which resembles Equation (4.64):

$$\hat{Q}_P(q) = \beta \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + \alpha. \tag{4.65}$$

We find that for $\alpha = 96.9043$ and $\beta = 123.0777$ (FLV), and $\alpha = 35.6899$ and $\beta = 141.7591$ (MKV) the MAPE values are minimized across test cases. The respective actual and estimated EDFs for the two test cases TC_FLV_180 and TC_MKV_360 are shown in Figure 4.21 below. Like in the video server scenario the relationship between the CPU




(a) TC_FLV_180. (b) TC_MKV_360

Figure 4.21.: QQ curves for power consumption vs. CPU utilization in user mode for test cases TC_FLV_180 and TC_MKV_360.

utilization quantiles and power consumption quantiles is realtively smooth. For this reason we are able to derive models with a relative performance comparable to other, purely CPU utilization-based models (Economou et al. 2006; Rivoire, Ranganathan, and Kozyrakis 2008;

Dargie 2014). Yet, the percentage errors, which we tabulate in Table 4.12, again show the missing opportunity to respect for changes in request rate since our models' performance decreases if reqeust rate increases.

| Transcoder Test Case | MAPE | medAPE |
|---|---|---|
| FLV_180 | 1.164 | 1.156 |
| FLV_360 | 4.148 | 3.999 |

(a) FLV test cases.

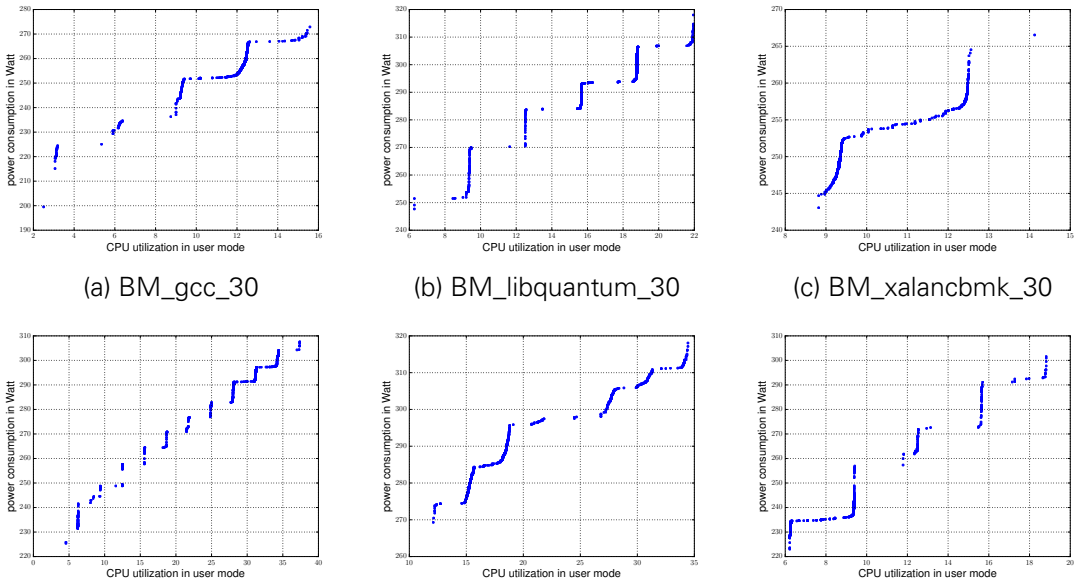
| Transcoder Test Case | MAPE | medAPE |
|---|---|---|
| MKV_180 | 1.295 | 0.698 |
| MKV_360 | 5.426 | 5.240 |

(b) MKV test cases

Table 4.12.: Percentage errors for $\hat{Q}_P(q) = 123.0777 \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + 96.9043$ (a), and $\hat{Q}_P(q) = 141.7591 \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + 35.6899$ (b).

### 4.6.3. SPEC CPU 2006 BENCHMARKS

Finally, we consider the power consumption quantiles as function of the resource utilization for different benchmarks. Naturally, since benchmarks are intended to address different computational problems, the relationship between a benchmark's resource utilization and the power consumption is more or less unique. As demonstration consider the QQ curves in Figure 4.22 which draw the power consumption quantiles vs. CPU utilization in user mode quantiles for the selected benchmarks in the *middle load* scenario.




(a) BM_gcc_30 (b) BM_libquantum_30 (c) BM_xalancbmk_30

(d) BM_cactusADM_30 (e) BM_milc_30 (f) BM_povray_30

Figure 4.22.: QQ curves for power consumption vs. CPU utilization in user mode for selected benchmark test cases.

The steps in the QQ curves represent the *operating points* of the benchmarks. Since the step-like behavior is relatively similar across benchmarks we can assume that a single *formula* is sufficent to describe the QQ curves. Comparing the QQ curves in Figure 4.22 with those in Figure 4.20 suggests to apply the same *formula* as in Equation (4.65), namely

$$\hat{Q}_P(q) = \beta \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + \alpha. \tag{4.66}$$

Yet, if we compare the figures on the x- and the y-axis pairwise for two arbitrary benchmarks, it is clearly to see that no single *model* in terms of the estimation parameters exists. We summarize the percentage errors for the floating point benchmarks in Table 4.13 and for the integer benchmakrs in Table 4.14.

| Benchmark | 436.cactusADM | | | 433.milc | | | 453.povray | | |
|---|---|---|---|---|---|---|---|---|---|
| Request Rate | 6 | 12 | 30 | 6 | 12 | 30 | 6 | 12 | 30 |
| MAPE | 9.172 | 0.636 | 12.141 | 0.870 | 3.678 | 1.725 | 1.165 | 3.516 | 3.533 |
| medAPE | 8.028 | 0.502 | 10.802 | 0.487 | 3.680 | 1.552 | 0.806 | 2.921 | 2.399 |

Table 4.13.: MAPEs and estimated model parameters for selected SPEC CFP2006 benchmarks for the model $\hat{Q}_P(q) = \beta \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + \alpha$.

| Benchmark | 403.gcc | | | 462.libquantum | | | 483.xalancbmk | | |
|---|---|---|---|---|---|---|---|---|---|
| Request Rate | 6 | 12 | 30 | 6 | 12 | 30 | 6 | 12 | 30 |
| MAPE | 6.661 | 0.734 | 5.231 | 0.909 | 8.675 | 8.269 | 3.259 | 0.624 | 1.555 |
| medAPE | 6.416 | 0.568 | 4.814 | 0.779 | 8.565 | 7.797 | 1.635 | 0.493 | 1.591 |

Table 4.14.: MAPEs and estimated model parameters for selected SPEC CINT2006 benchmarks for the model $\hat{Q}_P(q) = \beta \cdot lg(\hat{Q}_{C_{usr}}(q) + \hat{Q}_{C_{usr}}(.5)) + \alpha$.

We find that in all cases the MAPE is below $7.5\%$ and below $6\%$ in most of the cases. We conclude that in this particular scenario the modeling approach produces very promising results given that the model uses only a single, abstract input.

Compared to the previous model the MAPE values are higher in this case so that we conclude that the CPU utilization-based model is advantageous in this scenario. Judging by the MAPE values, however, we conclude that both models are competitive to other CPU utilization-based models for the overall power consumption (Economou et al. 2006; Rivoire, Ranganathan, and Kozyrakis 2008; Dargie 2014).

## 4.7. SUMMARY

In this chapter we have presented an approach to obtain models for the resource utilization as functions of workload, and models for the power consumption as function of resource utilization based on the relationship between quantiles. Several limitations of our approach have become apparent:

1. Due to the similarity between EDFs - and thus between the estimated quantile functions – multicollinearity between two explanatory variables is very likely which limits us to simple regression models.

2. For this reason we cannot account for workload dynamics like changing arrival rates and thus are limited to models for resource utilization with the request size as sole input parameter.

3. To derive suitable models it is necessary to determine and describe the function between the quantiles of the dependent and the independent variable. This step is not straight-forward and necessitates human intervention. Obviously, the accuracy of our models depends tremendously on how well the assumed function meets the actual data.

The main motivation for our approach was the assumption that time stamps in measurement data are not always reliable and sometimes not available at all. While it is always feasible to determine the correlation coefficient between arbitrary sets of data this is not sufficient to express a functional relationship. The approach described in this chapter was expected to present a solution so that functional relationships can be derived between data without timing information.

Admittedly, the QQ curves show very complex, nonlinear paths in most cases that render it a hard task to find appropriate transformation such that a linear relationship between the variables is established. Consider e.g. the QQ curves for the storage utilization in test case DL_1100 in Figure 4.12e and in test case TC_MKV_180 in Figure 4.12b. The steep increase in the middle of the curves are caused by the fact that the storage is unutilized for some time – indicated by the constant line in the left part of the curves – and that *if* it is utilized the minimum amount of data that is read from or written to the storage is $3 * 4096$ Bytes. Contrary, a request size of zero does not exist. Summarizing, lower quantiles of the storage utilization which are all zero are mapped to lower quantiles of the request size which are all greater than zero. At the same time the position of the break point on the x-axis depends on a) the application and how it accesses the storage, and b) the workload it processes. E.g. for the video server the break point will move left with increasing request sizes since the main memory's capacity will get exhausted and more data needs to be taken from storage. Similar conclusions can be drawn for the network utilization. In the QQ curves for this resource (e.g. Figure 4.16 we observe an additional horizontal line at $lg(52) \approx 1.7$ that is most probably caused by a computer beyond our control that performs UDP port scans. This line makes the task to describe the nonlinearity of the cure even more difficult.

We must conclude that except in a few cases our approach to exploit QQ relationships is not a suitable means to extract data for independent and dependent variables of a regression problem. In particular, we are not able to adequateley respect for changing request arrival rates due to multicollinearity with the request size. However, the request arrival rate would be necessary to model changes in QQ curves like breakpoints that we described above.

An additional source of the large estimation error is surely the sensitivity of percentage errors to very small and zero values in the actual data. E.g in Section 4.5.2.1 we see that while the EDFs of the actual data and the estimated CDFs are very close to each other the estimation errors – in particular the medAPE – are considerably large and the MAPE does not even exist in many cases. In Section 4.4.3 we discussed alternative scale-free errors and noticed that these do not fit our needs.

The road goes on and on.

Bilbo Baggins

# 5. CONCLUSION & FUTURE WORK

## 5.1. SUMMARY

This thesis presents an alternative approach to model an application's resource utilization as a function of the workload the application needs to process, and the power consumption caused by this resource utilization.

First, in an agreement with other research groups who participate in HAEC, we decided to focus on a video platform as common example. In order to obtain results from our experiments that are not purely academic we then analyzed the statistical properties of Internet traffic and, in particular, the properties of the workload to current video platforms, see Chapter 3. We had to note that (a) currently no workload generator for a video platform is freely available and (b) that in the body of work no publication exists that covers *all* relevant statistical properties of the workload of a video platform and (c) relevant literature is already outdated. For this reason we combined those properties that we believe are currently *reasonable* and implemented our own load generator which we employed in all subsequent experiments. Additionally to the video platform it was agreed upon to use a transcoder as a second example across the participating projects. We must note that no literature exists that analyses the workload of online transcoding platforms or the workload of the transcoders built-in to current video platforms. Therefore, we were forced to make *educated guesses* regarding the workload of transcoding applications. To produce workload we then used our workload generator and adapted the input parameters. Finally, we decided to employ benchmarks from the SPEC CPU 2006 suite as representatives of CPU intensive applications.

The core idea of our approach is to employ quantile-quantile curves as a means to extract data points for a regression problem. This idea is motivated by two observations: Firstly, measurement data for resource utilization, the power consumption and the application's workload is obtained from very different sources. Typically, the data sources – e.g. system monitoring tools, web server logs, or digital power analyzers – provide data samples with (a) different frequencies that (b) are not necessarily constant. While one can in principle employ frequency resampling techniques from the time series analysis domain in such cases, we observe secondly that for some interesting data no timing information can be provided in general (the sizes of files at the server side, in our example). It is, of course, always feasible to determine whether *any* (linear) relationship exists between two sets of data by means of a correlation analysis. But since we are interested in expressing a *functional* relationship a mere correlation analysis is not sufficient. A correlation coefficient of, say, $0.9$ between two data samples, $X$ and $Y$, indeed indicates a strong linear relationship. But it does not allow to make any statement about the specific value of some $y_j$ for some specific value

of $x_i$, in particular if $|X| = m$, $|Y| = n$ and $m \neq n$. But in a regression problem exactly this relationship needs to be available. This motivated us to find a means to determine matching $y_j$ and $x_i$ for any values of $m$ and $n$ greater than zero. We expected that quantile functions of $X$ and $Y$ can be exploited for this matter: By relating the respective quantile functions $F_m^{-1}(x)$ and $G_n^{-1}(x)$ to each other we can, for every $x_i$, estimate data points $y_i$ such that $y_i = G_n^{-1}(F_m(x_i))$. The pairs $(x_i, y_i)$ can then be used as input to a regression problem.

We assumed that this technique would enable us to derive models of an application's resource utilization as a function of the application's workload, i.e. the request size and the request rate. This assumption was not met: It points out that there exist a linear relationship between the quantiles of request size and interarrival times that we had not anticipated. We thus employed this technique to derive models for the quantiles of resource utilization of the video server and the transcoder applications as functions of the quantiles of the request size component of the application's respective workloads. But with the exception of a few cases the linearity assumption between the request size quantiles and the resource utilization quantiles was violated. Therefore we manually described the necessary transformation of the QQ curves to establish the linear relationship. Due to the complex relationships that are apparent in the QQ curves we partially applied an automated search in order to determine the best transformation parameters for every model.

In our approach we establish relationships between quantile functions which are indeed inverses of empirical distribution functions. Yet conventional goodness-of-fit (GoF) tests (Kolmogorov–Smirnov, Anderson–Darling, Carmér–von Mises) are, in our case, no appropriate means to assess the deviation between actual and the estimated distribution functions: The conventional GoF tests are hypotheses tests and are merely designed to answer the question whether two samples are taken from the same distribution. This simple yes-or-no question is not sufficient for our purposes. We thus apply empirical shift functions and employ the shift to asses the estimation errors of our models. To be able to compare model performance (a) across our own experiments and (b) to the results of other authors we review several definitions of percentage errors, and lastly employ the mean absolute percentage (MAPE) error to assess the relative performance of our models.

We find that in many cases the percentage error of our models are fairly large which we attribute to several reasons. (1), the description of the nonlinearity in the QQ curves is imperfect since data points that are caused by system properties and such that are caused by source(s) beyond our control are not explicitly accounted for. (2), some of the QQ curves show an operation point, meaning a sudden increase in resource utilization for some request size. However, the operation points are not bound to a certain request size, but rather change with the application and the workload of the application. Since the request rate component of the workload cannot be incorporated in the models, it is currently not feasible to express these operation points. (3), the resource utilization was fairly low in many of our experiments. In some experiments the resource utilization was often zero for a considerable portion of the measured data. Due to the sensitivity of the MAPE to values near zero in the actual data we observe an inflation of the error.

We conclude that our modeling approach is not applicable in general.

## 5.2. FUTURE WORK

Initially, this thesis intended to quantify the amount of interference at particular resources if applications are run in parallel. This goal was motivated by the observation that, in order to detect interference between applications, it is necessary to detect changes in the workload

for the applications (Kambadur et al. 2012; Novaković et al. 2013; Delimitrou and Kozyrakis 2014). Our expectation was to be able to detect and quantify interference directly once a model is present that expresses a relationship between the workload of an application and its resource utilization since changes in resource utilization due to changes in the workload would be readily expressed by the model. For several reasons we had to refrain from this goal.

The most important reason is that – as already mentioned multiple times – our models are only applicable under very specific conditions. Firstly, the current models do not adequateley incorporate sytem properties. E.g. in QQ curves for the request size and the storage utilization in Figure 4.12e and Figure 4.12f we observe a constant line that represents the minimum amount of data that can be read from or written to the storage. This property is not reflected in the model. An refinement of the current models must therefore at first respect these properties.

Secondly, the models also do not incorporate the operations points in the QQ curves. These can be interpreted as transitions between levels of resource utilization. Since the position of operation points on the x-axis not only depend on the respective resource, but also on the application and the workload to the application, the modeling of these operation points forms a research topic on its own.

Thirdly, it is obvious that the missing ability to incorporate the request rate component of the workload presents the most severe limitation of our approach. While our workload generator produces request rates independent of the request sizes there apparently exists a strong linear relationship between the *quantiles* of request sizes and interarrival times (i.e. the inverse of the requuest rate). Further investigation is necessary in order to determine how the request rate (or interarrival times) can be incorporated in the models.

# Appendices

# A. GRAPHS

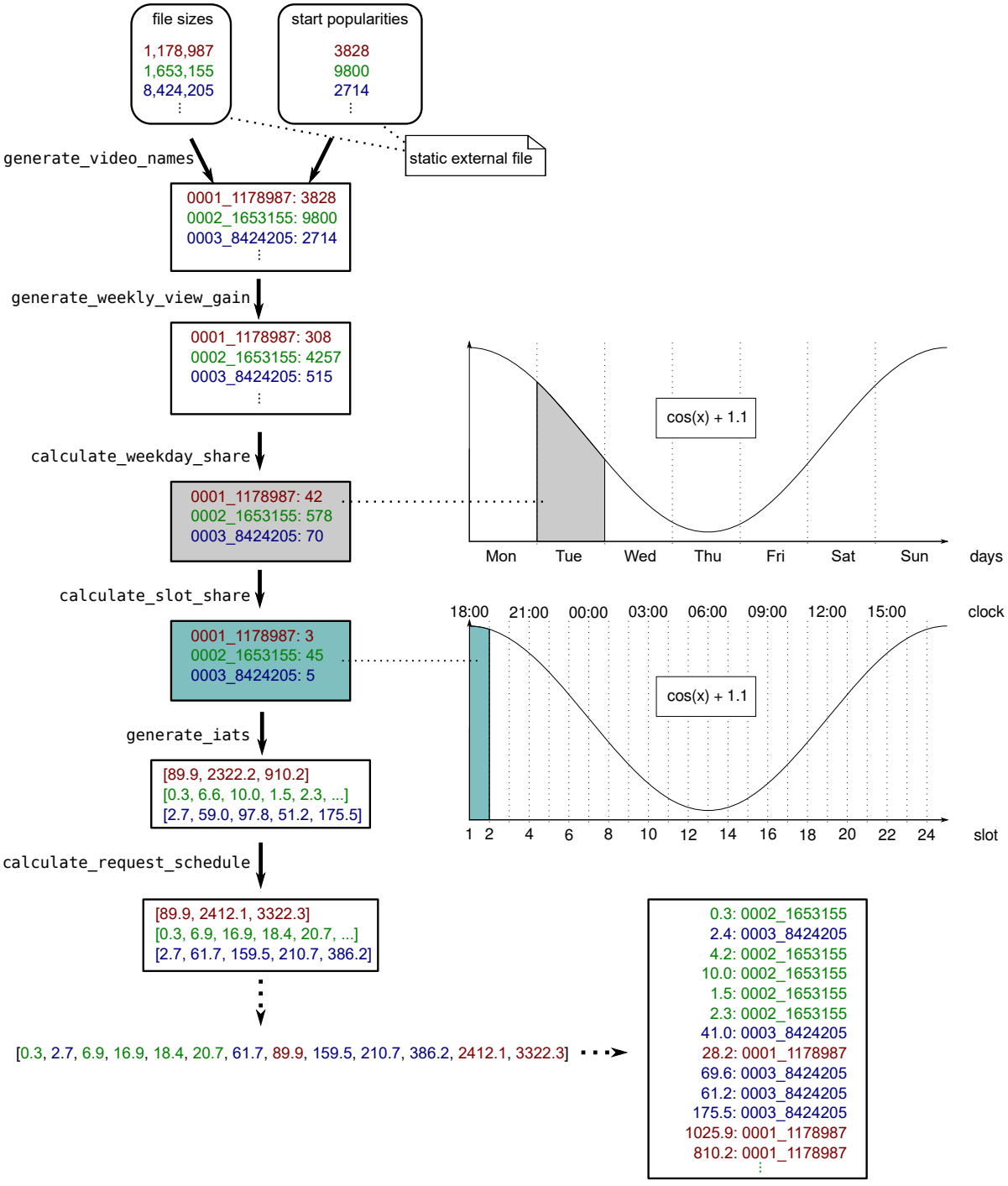



Figure A.1.: Example for the generation of a request sequence for three videos.

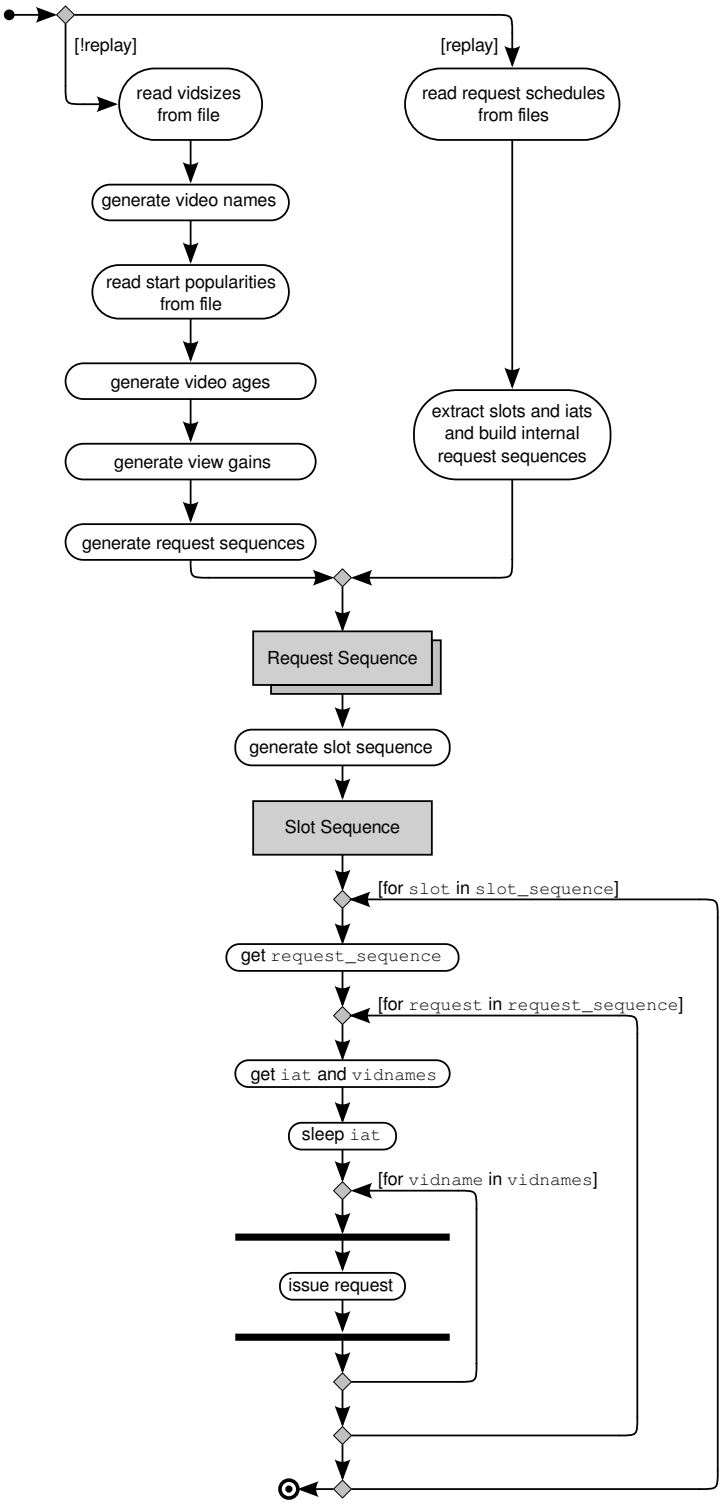



Figure A.2.: Behavior of the workload generator for the video server.

# BIBLIOGRAPHY


Abbasi, Zahra, Georgios Varsamopoulos, and Sandeep K S Gupta (2010). "Thermal aware server provisioning and workload distribution for internet data centers". In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*. New York, New York, USA: ACM Press, p. 130. ISBN: 9781605589428. DOI: 10.1145/1851476.1851493. URL: http://portal.acm.org/citation.cfm?doid=1851476.1851493 (cit. on p. 45).

Acharya, Soam, Brian C. Smith, and Peter Parnes (1999). "Characterizing user access to videos on the World Wide Web". In: *SPIE 3969, Multimedia Computing and Networking 20004*. Ed. by Klara Nahrstedt and Wu-chi Feng. San Jose, CA, pp. 130–141. DOI: 10.1117/12.373516. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=921306 (cit. on pp. 18, 20, 23).

Allalouf, Miriam et al. (2009). "Storage modeling for power estimation". In: *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference on - SYSTOR '09*. New York, New York, USA: ACM Press, p. 1. ISBN: 9781605586236. DOI: 10.1145/1534530.1534535. URL: http://portal.acm.org/citation.cfm?doid=1534530.1534535 (cit. on p. 45).

Almeida, Jussara M. et al. (2001). "Analysis of educational media server workloads". In: *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '01*. New York, New York, USA: ACM Press, pp. 21–30. ISBN: 1581133707. DOI: 10.1145/378344.378348. URL: http://portal.acm.org/citation.cfm?doid=378344.378348 (cit. on pp. 20, 22, 23).

Almeida, Virgılio et al. (1996). "Characterizing reference locality in the WWW". In: *Fourth International Conference on Parallel and Distributed Information Systems*. IEEE Comput. Soc. Press, pp. 92–103. ISBN: 0-8186-7475-X. DOI: 10.1109/PDIS.1996.568672. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=568672 (cit. on p. 19).

Aue, Alexander and Lajos Horváth (2013). "Structural breaks in time series". In: *Journal of Time Series Analysis* 34.September 2012, pp. 1–16. ISSN: 01439782. DOI: 10.1111/j.1467-9892.2012.00819.x (cit. on p. 74).

Barford, Paul and Mark E. Crovella (1998). "Generating representative web workloads for network and server performance evaluation". In: *ACM SIGMETRICS Performance Evaluation ...* 26.1, pp. 151–160. ISSN: 01635999. DOI: 10.1145/277858.277897. URL: http://portal.acm.org/citation.cfm?doid=277858.277897%20http://dl.acm.org/citation.cfm?id=277897 (cit. on pp. 18, 19, 26, 31).

Barroso, Luiz André and Urs Hölzle (2007). "The Case for Energy-Proportional Computing". In: *Computer* 40.12, pp. 33–37. ISSN: 0018-9162. DOI: 10.1109/MC.2007.443. URL:

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4404806 (cit. on p. 9).

Beard, Leo R. (1962). "Statistical Methods in Hydrology". Sacramento CA (cit. on p. 54).

Beloglazov, Anton and Rajkumar Buyya (2010). "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers". In: *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science - MGC '10*. New York, New York, USA: ACM Press, pp. 1–6. ISBN: 9781450304535. DOI: 10.1145/1890799.1890803. URL: http://portal.acm.org/citation.cfm?doid=1890799.1890803 (cit. on p. 17).

Bertran, Ramon et al. (2012). "A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs". In: *IEEE Transactions on Computers* XX.Xx, pp. 1–1. ISSN: 0018-9340. DOI: 10.1109/TC.2012.97. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6189333 (cit. on p. 46).

Bircher, W. Lloyd and Lizy Kurian John (2007). "Measurement Based Power Phase Analysis of a Commercial Workload". In: *Unique Chips and Systems*. Ed. by Eugene John and Juan Rubio. CRC Press. Chap. 8, pp. 217–237. ISBN: 1420051741 (cit. on p. 45).

Bircher, W. Lloyd, M Valluri, et al. (2005). "Runtime identification of microprocessor energy saving opportunities". In: *ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.* IEEE, pp. 275–280. ISBN: 1-59593-137-6. DOI: 10.1109/LPE.2005.195527. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1522776 (cit. on p. 45).

Borghol, Youmna et al. (2011). "Characterizing and modelling popularity of user-generated videos". In: *Performance Evaluation* 68.11, pp. 1037–1055. ISSN: 01665316. DOI: 10.1016/j.peva.2011.07.008. URL: http://linkinghub.elsevier.com/retrieve/pii/S016653161100099X (cit. on pp. 21, 25, 27, 28).

Borokov, AA and NM Sycheva (1968). "On asymptotically optimal non-parametric criteria". In: *Theory of Probability & Its Applications* XIII.2. URL: http://epubs.siam.org/doi/pdf/10.1137/1113048 (cit. on p. 58).

Bottou, Léon (2010). "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Ed. by Yves Lechevallier and Gilbert Saporta. Physica-Verlag HD, pp. 177–186. DOI: 10.1007/978-3-7908-2604-3_16 (cit. on p. 44).

Brinke, Steven Te et al. (2013). "Deriving minimal models for resource utilization". In: *Proceedings of the 2013 workshop on Green in/by software engineering - GIBSE '13*, p. 15. DOI: 10.1145/2451605.2451612. URL: http://dl.acm.org/citation.cfm?doid=2451605.2451612 (cit. on p. 43).

Brodsky, E. and Boris S. Darkhovsky (2013). *Nonparametric Methods in Change Point Problems*. Springer Science & Business Media (cit. on p. 74).

Brunnert, Andreas, Christian Vögele, and Helmut Krcmar (2013). "Automatic performance model generation for java enterprise edition (EE) applications". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8168 LNCS, pp. 74–88. ISSN: 03029743. DOI: 10.1007/978-3-642-40725-3-7 (cit. on p. 43).

Canner, Paul L. (1975). "A Simulation Study of One- and Two-Sample Kolmogorov-Smirnov Statistics with a Particular Weight Function". In: *Journal of the American Statistical Association* 70.349, p. 209. ISSN: 01621459. DOI: 10.2307/2285405. URL: http://www.jstor.org/stable/2285405?origin=crossref (cit. on p. 58).

Cha, Meeyoung et al. (2009). "Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems". In: *IEEE/ACM Transactions on Networking* 17.5, pp. 1357–1370. ISSN: 1063-6692. DOI: 10.1109/TNET.2008.2011358. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4801529 (cit. on pp. 21, 22, 25, 26).

Chen, G et al. (2008). "Energy-aware server provisioning and load dispatching for connection-intensive internet services". In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. NSDI'08 1. San Franciso, California: USENIX Association, pp. 337–350. URL: http://dl.acm.org/citation.cfm?id=1387589.1387613 (cit. on p. 9).

Chen, Zhuo and Y Yang (2004). "Assessing forecast accuracy measures". In: pp. 1–26. DOI: 10.1.1.69.1016. URL: http://www.stat.iastate.edu/preprint/articles/2004-10.pdf (cit. on p. 59).

Cheng, Xu, Cameron Dale, and Jiangchuan Liu (2008). "Statistics and Social Network of YouTube Videos". In: *2008 16th Interntional Workshop on Quality of Service*, pp. 229–238. ISSN: 1548-615X. DOI: 10.1109/IWQOS.2008.32. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4539688 (cit. on pp. 21, 23–26, 28).

Cheng, Xu, Jiangchuan Liu, and Cameron Dale (2013). "Understanding the Characteristics of Internet Short Video Sharing : A YouTube-Based Measurement Study". In: *Multimedia, IEEE Transactions on* 15, pp. 1184–1194. ISSN: 15209210. DOI: 10.1109/TMM.2013.2265531. arXiv: 0707.3670. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6522525 (cit. on p. 21).

Chesire, Maureen et al. (2000). "Measurement and Analysis of a Streaming-Media Workload". In: (cit. on pp. 20, 24).

Crovella, Mark E., Murad S. Taqqu, and Azer Bestavros (1998). "Heavy-tailed probability distributions in the World Wide Web". In: *A practical guide to heavy tails* 1, pp. 3–26. URL: http://books.google.com/books?hl=en%5C&lr=%5C&id=k4ekRZs%5C_fZkC%5C&oi=fnd%5C&pg=PA3%5C&dq=Heavy-tailed+probability+distributions+in+the+World+Wide+Web%5C&ots=jXglzgn8d%5C_%5C&sig=qFdBH6ZD14-TDdtxJXUFcLeAzpc (cit. on p. 31).

Crovella, M.E. and A. Bestavros (1997). "Self-similarity in World Wide Web traffic: evidence and possible causes". In: *IEEE/ACM Transactions on Networking* 5.6, pp. 835–846. ISSN: 10636692. DOI: 10.1109/90.650143. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=650143 (cit. on pp. 14, 17, 18).

Cunha, Carlos R., Azer Bestavros, and Mark E. Crovella (1995). *Characteristics of WWW Client-based Traces*. Tech. rep. Boston University Computer Science Department, pp. 1–18. URL: http://hdl.handle.net/2144/1571 (cit. on pp. 18, 19).

Cunnane, C. (1978). "Unbiased plotting positions — A review". In: *Journal of Hydrology* 37.3-4, pp. 205–222. ISSN: 00221694. DOI: 10.1016/0022-1694(78)90017-3. URL: http://linkinghub.elsevier.com/retrieve/pii/0022169478900173 (cit. on p. 54).

Danzig, Peter B. and Sugih Jamin (1991). *tcplib: A library of internetwork traffic characteristics*. Tech. rep. University of Southern California, pp. 1–8. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.5210 (cit. on pp. 18, 31).

Dargie, Waltenegus (2014). "A Stochastic Model for Estimating the Power Consumption of a Processor". In: *IEEE Transactions on Computers* 63.X, pp. 1–1. ISSN: 0018-9340. DOI: 10.1109/TC.2014.2315629. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6783802 (cit. on pp. 81, 82).

Dargie, Waltenegus and Alexander Schill (2012). "Analysis of the Power and Hardware Resource Consumption of Servers under Different Load Balancing Policies". In: *The 5th International Conference on Cloud Computing (IEEE Cloud 2012)*. Honolulu: IEEE Computer Society, pp. 24–29 (cit. on p. 45).

Delimitrou, Christina and Christos Kozyrakis (2013). "Paragon: QoS-aware scheduling for heterogeneous datacenters". In: *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*. New York, New York, USA: ACM Press, p. 77. ISBN: 9781450318709. DOI: 10.1145/

2451116.2451125. URL: http://dl.acm.org/citation.cfm?doid=2451116.2451125 (cit. on pp. 39, 44).

Delimitrou, Christina and Christos Kozyrakis (2014). "Quasar: Resource-Efficient and QoS-Aware Cluster Management". In: *Proceedings of the Nineteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Salt Lake City, UT, USA. ISBN: 9781450323055 (cit. on pp. 9, 44, 87).

Desnoyers, Peter et al. (2012). "Modellus: Automated modeling of complex internet data center applications". In: *ACM Transactions on the Web* 6.2, pp. 1–29. ISSN: 15591131. DOI: 10.1145/2180861.2180865 (cit. on p. 43).

Dhiman, Gaurav, Kresimir Mihic, and Tajana Rosing (2010). "A system for online power prediction in virtualized environments using Gaussian mixture models". In: *Proceedings of the 47th Design Automation Conference, DAC '10*. 3. New York, New York, USA: ACM Press, p. 807. ISBN: 9781450300025. DOI: 10.1145/1837274.1837478. URL: http://portal.acm.org/citation.cfm?doid=1837274.1837478 (cit. on pp. 45, 46).

Doksum, Kjell A. (1974). "Empirical Probability Plots and Statistical Inference for Nonlinear Models in the Two-Sample Case". In: *The Annals of Statistics* 2.2, pp. 267–277. ISSN: 0090-5364. DOI: 10.1214/aos/1176342662. URL: http://projecteuclid.org/euclid.aos/1176342662 (cit. on p. 55).

– (1977). "Some graphical methods in statistics: A review and some extensions". In: *Statistica Neerlandica* 31.2, pp. 53–68. ISSN: 0039-0402. DOI: 10.1111/j.1467-9574.1977.tb00752.x. URL: http://doi.wiley.com/10.1111/j.1467-9574.1977.tb00752.x (cit. on p. 55).

Doksum, Kjell A. and Gerald L. Sievers (1976). "Plotting with confidence: Graphical comparison of two populations". In: *Biometrika* 63, pp. 421–434. ISSN: 00063444 (cit. on pp. 55–58, 62).

Du Bois, Kristof et al. (2011). "SWEEP: evaluating computer system energy efficiency using synthetic workloads". In: *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers - HiPEAC '11*. New York, New York, USA: ACM Press, p. 159. ISBN: 9781450302418. DOI: 10.1145/1944862.1944886. URL: http://portal.acm.org/citation.cfm?doid=1944862.1944886 (cit. on p. 45).

Economou, Dimitris et al. (2006). "Full-System Power Analysis and Modeling for Server Environments". In: *Proceedings of the 2nd Workshop on Modeling, Benchmarking, and Simulation*. MoBS. Boston, MA, USA, pp. 70–77 (cit. on pp. 45, 47, 80, 82).

Fahrmeir, Ludwig, Thomas Kneib, and Stefan Lang (2009). *Statistik und ihre Anwendungen: Regression - Modelle, Methoden und Anwendungen*. 2. Auflage. Springer Berlin Heidelberg, p. 517. ISBN: 9783642018367. DOI: 10.1007/b138042. URL: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf (cit. on p. 51).

Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso (2007). "Power provisioning for a warehouse-sized computer". In: *ACM SIGARCH Computer Architecture News* 35.2, p. 13. ISSN: 01635964. DOI: 10.1145/1273440.1250665. URL: http://portal.acm.org/citation.cfm?doid=1273440.1250665 (cit. on pp. 17, 47, 49).

Friedrich, Matthias, Silvia Hollfelder, and Karl Aberer (2000). "Stochastic resource prediction and admission for interactive sessions on multimedia servers". In: *Proceedings of the eighth ACM international conference on Multimedia - MULTIMEDIA '00*, pp. 117–126. DOI: 10.1145/354384.354446. URL: http://portal.acm.org/citation.cfm?doid=354384.354446 (cit. on p. 41).

Gandhi, Anshul et al. (2010). "Optimality analysis of energy-performance trade-off for server farm management". In: *Performance Evaluation* 67.11, pp. 1155–1171. ISSN: 01665316. DOI: 10.1016/j.peva.2010.08.009. URL: http://linkinghub.elsevier.com/retrieve/pii/S0166531610001069 (cit. on p. 48).

Gill, Phillipa et al. (2007). "Youtube traffic characterization: a view from the edge". In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement - IMC '07*. IMC '07. San Diego, California, USA: ACM Press, pp. 15–28. ISBN: 9781595939081. DOI: 10.1145/1298306.1298310. URL: http://portal.acm.org/citation.cfm?doid=1298306.1298310 (cit. on pp. 20–25, 28, 30).

Govindan, Sriram et al. (2011). "Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines". In: *Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC '11*. New York, New York, USA: ACM Press, pp. 1–14. ISBN: 9781450309769. DOI: 10.1145/2038916.2038938. URL: http://dl.acm.org/citation.cfm?doid=2038916.2038938 (cit. on p. 39).

Gray, Robert M. (2001). "Gauss mixture vector quantization". In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)* 3, pp. 1769–1772. DOI: 10.1109/ICASSP.2001.941283. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=941283 (cit. on p. 46).

Hackenberg, Daniel et al. (2013). "Power measurement techniques on standard compute nodes: A quantitative comparison". In: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, pp. 194–204. ISBN: 978-1-4673-5779-1. DOI: 10.1109/ISPASS.2013.6557170. URL: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=6557170%20http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6557170 (cit. on p. 46).

Henning, JL (2006). "SPEC CPU2006 benchmark descriptions". In: *ACM SIGARCH Computer Architecture News* (cit. on p. 39).

Hintemann, Ralph (2015). *Energy consumption of data centers continues to increase in 2014*. Tech. rep. Berlin: Borderstep Institut für Innovation und Nachhaltigkeit gemeinnützige GmbH. URL: http://www.borderstep.de/wp-content/uploads/2015/01/Hintemann%5C_energy%5C_consumption%5C_data%5C_centers%5C_2014.pdf (cit. on p. 9).

Hintemann, Ralph and Klaus Fichter (2012). *Energieverbrauch und Energiekosten von Servern und Rechenzentren in Deutschland*. Tech. rep. Berlin: Borderstep Institut für Innovation und Nachhaltigkeit gemeinnützige GmbH. URL: http://www.bitkom.org/files/documents/Energieeinsparpotenziale%5C_von%5C_Rechenzentren%5C_in%5C_Deutschland(1).pdf (cit. on p. 9).

Holland, Paul W. and Roy E. Welsch (1977). "Robust regression using iteratively reweighted least-squares". In: *Communications in Statistics - Theory and Methods* 6, pp. 813–827. ISSN: 0361-0926. DOI: 10.1080/03610927708827533. URL: http://www.tandfonline.com/doi/abs/10.1080/03610927708827533%5C#.UvH6GGJdVGg (cit. on p. 42).

Hyndman, Rob J. and Yanan Fan (1996). "Sample Quantiles in Statistical Packages". In: *The American Statistician* 50.4, pp. 361–365. ISSN: 0003-1305. DOI: 10.1080/00031305.1996.10473566. URL: http://www.tandfonline.com/doi/abs/10.1080/00031305.1996.10473566 (cit. on pp. 53, 54).

Intel Corporation (2012). *Intel® 64 and IA-32 Architectures Software Developer's Manual. Combined Volumes: 1,2A,2B,2C,3A,3B and 3C*. Specification May. URL: http://developer.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf (cit. on pp. 44, 47).

Isci, Canturk and Margaret Martonosi (2003). "Runtime power monitoring in high-end processors: methodology and empirical data". In: *22nd Digital Avionics Systems Conference. Proceedings (Cat. No.03CH37449)*, pp. 93–104. DOI: 10.1109/MICRO.2003.1253186. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1253186 (cit. on p. 46).

Islam, MA et al. (2013). "Revisiting Popularity Characterization and Modeling of User-generated Videos". In: URL: http://www.cs.usask.ca/faculty/eager/mascots13b.pdf (cit. on pp. 21, 28).

IWR, Internationales Wirtschaftsforum Regenerative Energien (2015). *Rekordmonat: Strom aus Wind und Solar erstmals über 10 Milliarden Kilowattstunden*. URL: http://www.iwr.de/news.php?id=28104 (visited on 02/07/2015) (cit. on p. 9).

Johnson, Norman L. and Samuel Kotz (1970). *Continuous Univariate Distributions - 2*. Boston: Houghton Mifflin (cit. on p. 54).

Joshi, Ajay et al. (2006). "Measuring benchmark similarity using inherent program characteristics". In: *IEEE Transactions on Computers* 55.6, pp. 769–782. ISSN: 0018-9340. DOI: 10.1109/TC.2006.85. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1628963 (cit. on p. 37).

Kambadur, Melanie et al. (2012). "Measuring interference between live datacenter applications". In: *2012 International Conference for High Performance Computing, Networking, Storage and Analysis - SC'12* 3, pp. 1–12. DOI: 10.1109/SC.2012.78. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6468475 (cit. on p. 87).

Kansal, Aman et al. (2010). "Virtual machine power metering and provisioning". In: *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, p. 39. DOI: 10.1145/1807128.1807136. URL: http://portal.acm.org/citation.cfm?doid=1807128.1807136 (cit. on p. 17).

Karkulahti, Ossi and Jussi Kangasharju (2015). "Youtube Revisited: On the Importance of Correct Measurement Methodology". In: pp. 17–30. DOI: 10.1007/978-3-319-17172-2 (cit. on p. 21).

Kendall, David G. (1953). "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain". In: *The Annals of Mathematical Statistics* 24.3, pp. 338–354. ISSN: 0003-4851. DOI: 10.1214/aoms/1177728975. URL: http://projecteuclid.org/euclid.aoms/1177728975 (cit. on p. 48).

Khan, Arijit and N. Anerousis (2012). "Workload characterization and prediction in the cloud: A multiple time series approach". In: *2012 IEEE Network Operations and Management Symposium*, pp. 1287–1294. ISSN: 1542-1201. DOI: 10.1109/NOMS.2012.6212065. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6212065 (cit. on p. 42).

Koomey, Jonathan G (2011). *Growth in data center electricity use 2005 to 2010*. Tech. rep. URL: http://www.analyticspress.com/datacenters.html (cit. on p. 9).

Kusic, Dara et al. (2008). "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control". In: *2008 International Conference on Autonomic Computing*, pp. 3–12. DOI: 10.1109/ICAC.2008.31. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4550822 (cit. on p. 45).

Laherrère, J. and D. Sornette (1998). "Stretched exponential distributions in nature and economy: "fat tails" with characteristic scales". In: *The European Physical Journal B* 2.4, pp. 525–539. ISSN: 1434-6028. DOI: 10.1007/s100510050276. URL: http://link.springer.com/10.1007/s100510050276 (cit. on p. 25).

Lang, Willis and Jignesh M Patel (2010). "Energy Management for MapReduce Clusters". In: *Proc. VLDB Endowment* 3.1-2, pp. 129–139. URL: http://dl.acm.org/citation.cfm?id=1920841.1920862 (cit. on p. 45).

Lee, Jeongeun Julie and Maruti Gupta (2007). *A new traffic model for current user web browsing behavior* (cit. on p. 18).

Leland, W.E. et al. (1994). "On the self-similar nature of Ethernet traffic (extended version)". In: *IEEE/ACM Transactions on Networking* 2.1, pp. 1–15. ISSN: 10636692. DOI: 10.

1109/90.282603. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=282603 (cit. on pp. 17, 18).

Li, Jun and Shuchao Ma (2013). "Characterization and modeling of video popularity". In: *International Journal of Communication Systems*. ISSN: 10745351. DOI: 10.1002/dac.2493. URL: http://doi.wiley.com/10.1002/dac.2493 (cit. on pp. 21, 25).

Liu, Huan (2012). *Host server CPU utilization in Amazon EC2 cloud*. URL: http://huanliu.wordpress.com/2012/02/17/host-server-cpu-utilization-in-amazon-ec2-cloud/ (visited on 01/28/2014) (cit. on p. 10).

Liu, Zhen, Nicolas Niclausse, and César Jalpa-Villanueva (2001). "Traffic model and performance evaluation of Web servers". In: *Performance Evaluation* 46.2-3, pp. 77–100. ISSN: 01665316. DOI: 10.1016/S0166-5316(01)00046-3. URL: http://linkinghub.elsevier.com/retrieve/pii/S0166531601000463 (cit. on p. 18).

Mah, Bruce A. (1997). "An empirical model of HTTP network traffic". In: *Proceedings of INFOCOM '97*. Vol. 2. Kobe, Japan: IEEE Comput. Soc. Press, pp. 592–600. ISBN: 0-8186-7780-5. DOI: 10.1109/INFCOM.1997.644510. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=644510 (cit. on pp. 17, 18).

Makridakis, Spyros (1993). "Accuracv concerns measures : theoretical and practical concerns". In: *International journal of forecasting* 9, pp. 527–529. ISSN: 0169-2070. DOI: http://dx.doi.org/10.1016/0169-2070(93)90079-3 (cit. on p. 58).

Makridakis, Spyros and Miche'le Hibon (2000). "The M3-Competition : results , conclusions and implications". In: 16, pp. 451–476 (cit. on p. 59).

Mitra, Siddharth et al. (2011). "Characterizing Web-Based Video Sharing Workloads". In: *ACM Transactions on the Web* 5.2, pp. 1–27. ISSN: 15591131. DOI: 10.1145/1961659.1961662. URL: http://portal.acm.org/citation.cfm?doid=1961659.1961662 (cit. on pp. 21, 25–27, 32).

Möbius, Christoph and Waltenegus Dargie (2014). "Statistical Analysis of the Workload of a Video Hosting Server". In: *21st International Conference on Analytical & Stochastic Modelling Techniques & Applications; Budapest, Hungary*. Vol. 1. Springer Berlin / Heidelberg (cit. on p. 41).

Möbius, Christoph, Waltenegus Dargie, and Alexander Schill (2013). "Power Consumption Estimation Models for Processors, Virtual Machines, and Servers". In: *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1. ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.183. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6570721 (cit. on p. 41).

Molka, Daniel et al. (2010). "Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors". In: *International Conference on Green Computing*. IEEE, pp. 123–133. ISBN: 978-1-4244-7612-1. DOI: 10.1109/GREENCOMP.2010.5598316. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5598316 (cit. on p. 45).

Moreno, Ismael Solis et al. (2013). "An approach for characterizing workloads in google cloud to derive realistic resource utilization models". In: *Proceedings - 2013 IEEE 7th International Symposium on Service-Oriented System Engineering, SOSE 2013*, pp. 49–60. ISBN: 9780769549446. DOI: 10.1109/SOSE.2013.24 (cit. on p. 17).

Mosberger, D and T Jin (1998). "httperf—a tool for measuring web server performance". In: *ACM SIGMETRICS Performance Evaluation ...* Pp. 31–37. URL: http://dl.acm.org/citation.cfm?id=306235 (cit. on p. 31).

Nathuji, Ripal and Karsten Schwan (2008). "VPM tokens: virtual machine-aware power budgeting in datacenters". In: *Proceedings of the 17th international symposium on High performance distributed computing - HPDC '08*. New York, New York, USA: ACM Press, p. 119. ISBN: 9781595939975. DOI: 10.1145/1383422.1383438. URL: http://portal.acm.org/citation.cfm?doid=1383422.1383438 (cit. on p. 17).

Novaković, Dejan et al. (2013). "DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments". In: *USENIX ATC* (cit. on p. 87).

Padala, Pradeep et al. (2007). "Adaptive control of virtualized resources in utility computing environments". In: *ACM SIGOPS Operating Systems Review* 41.3, p. 289. ISSN: 01635980. DOI: 10.1145/1272998.1273026. URL: http://portal.acm.org/citation.cfm?doid=1272998.1273026 (cit. on p. 17).

Park, Kihong, Gitae Kim, and Mark E. Crovella (1996). "On the relationship between file sizes, transport protocols, and self-similar network traffic". In: *Proceedings of 1996 International Conference on Network Protocols (ICNP-96)*. IEEE Comput. Soc. Press, pp. 171–180. ISBN: 0-8186-7453-9. DOI: 10.1109/ICNP.1996.564935. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=564935 (cit. on p. 18).

Paxson, Vern and Sally Floyd (1995). "Wide area traffic: the failure of Poisson modeling". In: *IEEE/ACM Transactions on Networking* 3.3, pp. 226–244. ISSN: 10636692. DOI: 10.1109/90.392383. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=392383 (cit. on pp. 17, 18, 22).

Petrucci, Vinicius et al. (2011). "Optimized Management of Power and Performance for Virtualized Heterogeneous Server Clusters". In: *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 23–32. DOI: 10.1109/CCGrid.2011.15. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5948593 (cit. on p. 17).

Phansalkar, Aashish, Ajay Joshi, and Lizy K. John (2007a). "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite". In: *ACM SIGARCH Computer Architecture News* 35, p. 412. ISSN: 01635964. DOI: 10.1145/1273440.1250713 (cit. on p. 37).

– (2007b). "Subsetting the SPEC CPU2006 benchmark suite". In: *ACM SIGARCH Computer Architecture News* 35.1, p. 69. ISSN: 01635964. DOI: 10.1145/1241601.1241616 (cit. on pp. 37, 38).

Raghavendra, Ramya et al. (2008). "No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center". In: *SIGARCH Comput. Archit. News4*, pp. 48–59. DOI: http://doi.acm.org/10.1145/1353534.1346289 (cit. on p. 17).

Razali, Nornadiah Mohd and Yap Bee Wah (2011). "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests". In: *Journal of Statistical Modeling and Analytics* 2.1, pp. 21–33 (cit. on p. 55).

Reiss, Charles et al. (2012). "Heterogeneity and dynamicity of clouds at scale: Google trace analysis". In: *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC '12*. New York, New York, USA: ACM Press, pp. 1–13. ISBN: 9781450317610. DOI: 10.1145/2391229.2391236. URL: http://dl.acm.org/citation.cfm?doid=2391229.2391236 (cit. on p. 10).

Rivoire, Suzanne, Parthasarathy Ranganathan, and Christos Kozyrakis (2008). "A Comparison of High-Level Full-System Power Models". In: *Proceedings of the 2008 conference on Power aware computing and systems*. HotPower'08. San Diego, California, USA: USENIX Association (cit. on pp. 80, 82).

Speitkamp, Benjamin and Martin Bichler (2010). "A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers". In: *IEEE Transactions on Services Computing* 3.4, pp. 266–278. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.25. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5467027 (cit. on p. 58).

Switzer, Paul (1976). "Confidence Procedures for Two-Sample Problems". In: *Biometrika* 63, p. 13. ISSN: 00063444. DOI: 10.2307/2335079. URL: http://www.jstor.org/stable/2335079?origin=crossref (cit. on p. 56).

Szabo, Gabor and Bernardo a. Huberman (2010). "Predicting the popularity of online content". In: *Communications of the ACM* 53.8, p. 80. ISSN: 00010782. DOI: 10.1145/1787234.1787254. URL: http://portal.acm.org/citation.cfm?doid=1787234.1787254 (cit. on pp. 21, 22).

Tang, Wenting et al. (2003). *Long-term Streaming Media Server Workload Analysis and Modeling*. Technical Report. HP Laboratories. URL: https://www.hpl.hp.com/techreports/2003/HPL-2003-23.pdf (cit. on pp. 19, 20, 22, 25, 26, 28, 29, 31).

Tomás, Luis et al. (2014). "The straw that broke the camel's back: safe cloud overbooking with application brownout". In: *IEEE Conference on Cloud and Autonomic Computing*, pp. 151–160. ISBN: 9781479958412. DOI: 10.1109/ICCAC.2014.10. URL: http://www8.cs.umu.se/~hernandf/pubs/cac2014-preprint.pdf (cit. on p. 58).

Tukey, John W. (1962). "The Future of Data Analysis". In: *The Annals of Mathematical Statistics* 33.May 2015, pp. 1–67. ISSN: 0003-4851. DOI: 10.1214/aoms/1177704711 (cit. on p. 54).

Urgaonkar, Bhuvan, Prashant Shenoy, and Timothy Roscoe (2009). "Resource overbooking and application profiling in a shared Internet hosting platform". In: *ACM Transactions on Internet Technology* 9.1, pp. 1–45. ISSN: 15335399. DOI: 10.1145/1462159.1462160. URL: http://portal.acm.org/citation.cfm?doid=1462159.1462160 (cit. on p. 58).

Veloso, Eveline et al. (2006). "A hierarchical characterization of a live streaming media workload". In: *IEEE/ACM Transactions on Networking* 14.1, pp. 133–146. ISSN: 1063-6692. DOI: 10.1109/TNET.2005.863709. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1597229 (cit. on pp. 20, 22, 25).

Verma, Akshat et al. (2009). "Server workload analysis for power minimization using consolidation". In: *Proceedings of the 2009 conference on USENIX Annual technical conference*. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1855835 (cit. on p. 17).

Willinger, W. et al. (1997). "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level". In: *IEEE/ACM Transactions on Networking* 5.1, pp. 71–86. ISSN: 10636692. DOI: 10.1109/90.554723. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=554723 (cit. on pp. 17, 18).

Wood, Timothy, Ludmila Cherkasova, et al. (2008). "Profiling and modeling resource usage of virtualized applications". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5346 LNCS, pp. 366–387. ISSN: 03029743. DOI: 10.1007/978-3-540-89856-6_19 (cit. on p. 41).

Wood, Timothy, Gabriel Tarasuk-Levin, et al. (2009). "Memory buddies: exploiting page sharing for smart colocation in virtualized data centers". In: *ACM SIGOPS Operating Systems Review* 43.3, p. 27. ISSN: 01635980. DOI: 10.1145/1618525.1618529. URL: http://portal.acm.org/citation.cfm?doid=1618525.1618529 (cit. on p. 17).

Zhu, Qian, Jiedan Zhu, and Gagan Agrawal (2010). *Power-aware consolidation of scientific workflows in virtualized environments*. Technical Report. Ohio State University, pp. 1–12 (cit. on p. 42).

Zink, Michael et al. (2009). "Characteristics of YouTube network traffic at a campus network – Measurements, models, and implications". In: *Computer Networks* 53.4, pp. 501–514. ISSN: 13891286. DOI: 10.1016/j.comnet.2008.09.022. URL: http://linkinghub.elsevier.com/retrieve/pii/S1389128608003423 (cit. on pp. 21–25, 31).