Bauhaus-University Weimar
Faculty of Media
Media Informatics

Do Features Matter for Energy Consumption?

A Case Study on Web Servers

Masterthesis

Clement Benedict Welsch
Born October 14, 1981 in Wiesbaden

Matriculation Number 30248

Frist Referee: Prof. Dr. Norbert Siegmund
Second Referee: Prof. Dr. Benno Maria Stein

Submission: August 5, 2017

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, August 5, 2017

Abstract

As computers have become pervasive, their energy consumption can not be neglected any more. Especially cloud systems with very long uptime are a promising area for contributions of the IT industries to the global energy saving efforts. This thesis takes a feature oriented perspective on webservers as an instance of such systems. The case study aims to explore the influence of configurable features on the energy consumption of those systems.

It can be demonstrated that features do have some effect on the energy consumption and that they interact with each other. But it remains dubious if the knowledge about that effects may lead to valuable guidelines for energy efficient configurations. That is due to the documented influence of the systems load on these effects. Moreover, the assumption that energy efficiency can be achieved through performance optimization, must be rejected for webservers.

Contents

Abbreviations

AES	Advanced Encryption Standard
API	Application Programmer Interface
CPU	Central Processing Unit
DSL	Domain Specific Language
HDD	Hard Disc Drive
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MPM	Multi Processing Module
OLS	Ordinary Least Square
PDU	Power Distribution Unit
SPL	Software Product Line
SSD	Solid State Drive
TCP	Transmission Control Protocol
TDP	Thermal Design Power
TLS	Transport Layer Security
URI	Uniform Resource Identifier
WWW	World Wide Web

1. Introduction

Despite living in times where it rather seems in vogue to believe than to trust scientific evidence, the scientific community agrees on climate change to be men made. Among the scientific publications that take some position on global warming, roughly about 97% endorse consensus on human-caused global warming [1]. As all industries, IT is asked to contribute its part in lowering greenhouse emissions, too. A very promising target are the ever growing number of data centres. In Germany alone, the energy consumption of data centres increased from 2010 to 2015 within five years from 10.5 to 12 billion kilowatt-hours. The major share is caused by servers, followed by storage, network, and – almost without any change – cooling and independent power supply [2]. By comparison, US data centres consumed 70 billion kilowatt-hours in 2014. The growth rate is 4% within four years from 2010 to 2014, which is remarkable since it has been 24% from 2005 till 2010 and even 90% from 2000 till 2005. Newly calculated prognoses project a stable growth of data centres energy consumption of about 4% until 2020, although the growth rate of data centres is way higher. This effect can be mainly attributed to energy efficiency efforts undertaken by large US companies, such as Goolge, Amazon, and Apple [3]. Especially relocating data centres to regions that offer natural cooling, consolidating servers, and favouring energy efficient standard hardware in scale-out fashion were obviously fruitful strategies at the infrastructure layer.

Webservers are one of the workhorses the Internet economy heavily relies on. They are omnipresent in the data centres that form what nowadays is called the cloud. There, those systems are running 24 hours a day, 365 days throughout a year and some are up and serving for far over a decade [4]. Moreover, they became truly pervasive as they were utilized for serving configuration interfaces on devices on the Internet of things, such as home routers. Thus, webservers

are suspect to contribute a non-negligible portion of the energy consumption of information technology.

Webservers are complex, highly configurable software. As for all software systems, energy consumption can not be measured directly, but only be estimated based on other metrics. Direct measurement of energy consumption is only possible for the underlying hardware. Energy measurements can be done either for the entire computer system, just by intercepting the electrical wire or individually for distinct components, such as CPU cores or hard disk. In order to estimate the energy consumption of a software system, metrics are needed that are more attributable to that software. Fine-grained, internal measurements are possible for hardware utilization, such as CPU time or memory footprint, and can go down even to level of a distinct processes. At the moment, most practical estimations about energy consumption of software rely on hardware utilization metrics.

Beside hardware utilization metrics, there are performance metrics as another class of so-called non-functional properties. Those properties describe *how* a system works, in contrast to *what* is does functionally. Several studies have been conducted on influences and interactions between functional and non-functional properties of a variety of different software systems that become employed in data centres, from classical single node database systems to distributed map/reduce-like workflow systems [5, 6, 7, 8, 9]. Externally, the response time of a webserver can be used as a performance metric. If the energy consumption of a webserver could be estimated by its performance, energy consumption would essentially come for free, since performance optimization is common sense. But there still exists an unresolved dispute whether performance correlates with energy consumption and can therefore lead to reliable estimations. For distributed workflow applications [9] and single node databases, it was "found that the best performing configuration was also the most energy efficient"[7]. On the other hand, this does not seem to hold for

changing workloads [5]. So, it remains unclear whether optimizing a webserver's performance also means to optimize its energy efficiency.

At the software layer, two different perspectives onto the energy efficiency topic arise. First, developers might consider inefficiencies of platforms or algorithmic design approaches [10, 11, 12]. For instance, it has been best practice to poll for changing resources, to provide the best user interface experience. With the success of the smartphone, this best practice shifted dramatically towards push-services, since developers realised how much energy was wasted for polling [12]. Second, application users are interested in an optimal energy-efficient behaviour of their systems, which is actually the main stimulus for developers to undertake efforts in energy-efficiency. Although, or even because, webserver users are considered to be professionals, they are confronted with the bad habit of developers to only add functionality, while never removing outdated functionalities. That problem is called *configuration space explosion* [13].

The feature-oriented approach as a flavour of software engineering tries to deal with the situation by comprehending a feature as an increment of functionality [14]. By activating or deactivating features, a configuration space of different variants of the software is span, which is yielding a *software product line* (SPL). This abstraction allows to analyse the influences of features on non-functional properties. Furthermore, interactions between features, meaning effects that arise only when certain feature combinations are configured, can be made visible[15].
The feature-oriented perspective offers another possible approach for energy optimization. Knowledge about the energetic behaviour of features and the influence of the overwhelming number of possible configurations on the energy consumption, could lead to configuration choices that are optimized with respect to performance and energy efficiency. Which is congruent to the users aims for optimization in at least two dimensions: maximising performance while minimizing cost [9]. This is the perspective chosen for this thesis.

The aim of this thesis is twofold: First, developing a sound and elaborate workload scenario that can be used to assess a webserver's energy consumption. Second, actually applying the workload scenario and analyse the behaviour of four well-known webserver engines. The engines come from the two, at the time, major open-source webservers Apache and Nginx [16]. In a black-box approach, the engines are run with varying configurations, while exposed to changing workload conditions. During those runs energy consumption, performance, and hardware utilization metrics are tracked.
However, a sound empirical experiment requires careful preparation. Thus, a prestudy was conducted to determine suitable workloads, metrics, and configurations. It must be ensured to measure the actual load of the webserver. In particular, the prestudy identified misleading external boundaries, such as network capacity or file-handle limits. Moreover, the measurement bias was assessed to adjust the number of experiment repetitions for the actual study. Feature models were framed and edited to describe the variability of the systems, within the constraints of their features. Finally, all building blocks, including measurement tools, load testing tool and customized compiled webserver engines, were assembled in a scripting environment to be deployed in a computing cluster.

For the final study 209,664 configurations were generated, executed and ran for roughly 10 days, not taking parallelization into account. That produced about 140 GB of raw data, which needed to be refined and analysed.

It turned out that the relation between the performance of a webserver and its energy consumption is a trade-off. Despite the overarching question of whether features do influence energy consumption can be answered with yes, the observed effects, remain questionable and need further investigation. Considering feature-orientation as a user's tool for saving energy is delusive. It is the developers duty to free users from that burden.

2. Background

Subject of this thesis are webservers as an instance of configurable software systems and their energy consumption. This chapter gives a brief description of those systems, their properties and performance metrics, as well as introducing and disambiguating important terms.

2.1. Webserver

The WWW is a service in the largest man-made distributed system, the Internet. Its technological foundation constitutes of three pillars. URI for addressing, some hypermedia representation for the exchange of artefacts — historically HTML and HTTP. HTTP itself is a stateless request-response protocol that strictly separates the roles of the client as requesting and the server as responding party. Hence, webservers are programs that essentially run in an infinite loop, listening for connection attempts by clients on some interface provided by the underlying operating system. At the moment, this interface generally is a TCP-socket, but subject to change in the near future[17]. Upon that socket, webservers implement HTTP and serve static files by mapping URIs to the local file system. Beside that, they provide that service to applications via an API. As such, webservers represent a very thin layer between the operating system and a concrete application.

A webservers foremost work is parsing and interpreting requests. Part of that means mapping URIs, conditions, mime-types, and so forth. Followed by assembling the response headers and payload. Most of those tasks are delegated to the operating system layer (e.g.: IO) or some external library (e.g.: encryption or compression). That means that webservers usually delegate all the

heavy lifting to instances that are highly optimized for those jobs. For example, delegating the sending of some static payload to the operating system means that it can benefit from the capability of the kernel of directly copying from one file descriptor to another. Hence, the entire transfer is handled within the kernel and has no need for further processing in user-space. That small example shows that it is in fact impossible to observe a webserver isolated from its environment. Anything measurable that can be used to describe a webserver's behaviour is limited or influenced by external factors, such as hardware, operating system, network, or other services.

As a webserver is a component in a distributed system, some terms have to be clarified before moving on. For any distributed system, three basic states have to be distinguished:

1. A *local computation* is just what any computer usually does, even without any network connection.

2. A *sent event* is defined as the state when a message from node A to node B has entirely left the buffers of node A.

3. A *delivery event* is defined as the state when a message from node A to node B entirely reached the buffers of node B.

It is important to distinguish those states precisely, since networks always need to be assumed as unreliable. That means that, for instance, the delivery of some sent message might occur with a huge delay, probably after other succeeding messages already have been delivered, or even never if the message got lost.

2.2. Customizability

Although webservers do represent only a very thin layer, as discussed before, they offer many configuration options to adapt them to individual needs. Hence, a webserver can be seen as software product line. An SPL is the set of all valid combinations of configuration options. Each member of the SPL represents a variant of the same software product, but with a unique combination of features. Usually, a feature is defined as "an increment in program functionality"[18]. For this thesis any configuration option will be considered as a feature, regardless if it actually contributes functionality, e.g., via a module, or if it just constrains system usage to certain limits. Through this, the entire configuration space of the webserver system can be modelled by leveraging the capabilities of feature diagrams, which is a typical approach in feature-oriented software engineering[19].

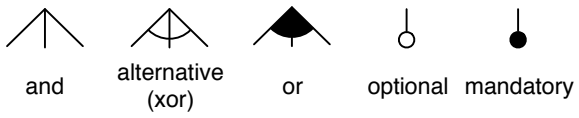


Figure 1: Feature Diagram Notations

Feature diagrams are a graphical notation for feature models. They allow a comprehensive overview of an SPL and its variants. Those graphs form a tree structure were leave nodes represent concrete features whereas inner nodes represent compounds and hence, groups of features. Groups might define the relationship between their children as

And — all children must be selected.

Alternative or logical XOR — only one children can and must be selected.

Or — one or more children can be selected.

while childnodes can be specified independently from that as *mandatory* or *optional*. Figure 1 depicts the visual notation for those relations.

A variant of the software product can be derived through different mechanisms. It can take place at [19]:

Compile-time — through static linking or compiler flags.

Load-time — through configuration files or command-line flags.

Run-time — through input data, e.g. HTTP header fields.

For this study, customization solely took place via configuration at load-time.

2.3. Non-functional requirements

In contrast to *what* a software is supposed to do, therefore its functionality and hence called functional requirements, non-functional requirements are assumed somewhat orthogonal to that. They are describing *how* a system should behave fulfilling its functionality. Important *quality attributes* are meant by the term, for instance: security, safety, fault tolerance, reliability, resilience,... and performance. Naturally, any project should be vigilant about those important features. However, they become widely under-represented and suppressed in the development process, because of the vilification as *not* functional. That is why an increasing number of enterprise developers aiming at proscribe the term. Recently, the issue reached the scientific community and Eckhardt et al.[20] mostly confirmed, that "there is no such thing as Non-Functional Requirements"[21]. For this thesis, the term non-functional requirement can be used synonymous to non-functional property, since the development process that should take care of the property behaving accordingly to the requirement should already be finalized for the systems discussed onwards.

2.4. Performance metrics

To quantify the performance of a webserver, common metrics from the area of networking are borrowed. Usual computation performance metrics, such as runtime are useless, since servers are defined to run endlessly. Measurements like CPU- or memory-utilization are interesting from an efficiency perspective, including energy consumption, but they do not help in judging the effectiveness of the server. To describe the perceived performance of a server, the following metrics are used:

Hit rate is the number of requests that reach a server in a time period without paying attention if they are accepted in any way.

Latency is the time elapsed from the request message has started being sent until the delivery of the response message has began.

Response time is the time elapsed from the request message has started being sent until the entire response message has been delivered.

Throughput is the actual performance metric. It describes units of work per unit of time. Sadly it is defined very ambiguously. (See below.)

Since *throughput* is such an ambiguous metric, a few more words must be spent on that. In physical networks, it describes the data transferred per time unit measured in bits per second (bit/s or bps). When it comes to network software, a bad habit became established measuring in Bytes per second (Byte/s), which is an annoying and regularly overseen detail. Beside that, another metric which takes the volume of the transferred data out of the equation, is called *throughput* as well. It is measured in requests per second (req/s or rps), but sometimes per minute. Furthermore, it has never been clarified if only successful or any communications count into throughput, including failed ones. For this thesis, if not stated otherwise, it will be defined as all request attempts

that were made, regardless if they succeeded or failed. Of course, whenever *throughput* is defined, it always requires the additional information of an error rate.

2.5. Performance testing

To obtain any meaningful measurements, the system must be exposed to some workload. "Workload is the stimulus applied to a system [...] to simulate a usage pattern [...]."[22] Real-world workload, as occurring in production systems, is highly unpredictable. Nevertheless, it renders any comparison between different measurements impossible. Therefore, performance testing usually is performed in controlled environments running synthetic benchmark scenarios, where benchmarking means a performance comparison against a self-defined baseline. Creating a synthetic workload requires defining the number of simulated clients, their hit rate, and how many are acting concurrently, as well as the data volumes they request. Performance testing can be distinguished further into[22]:

Load testing which aims to expose the system to a load that represents average production usage.

Stress testing which aims to expose the system to load beyond normal production. Therefore pushing the system to its limits, even until it fails.

2.6. Electrical Models

To prevent any misconceptions about the physical units that come into play when dealing with electrical circuits, some clarifications shall be given. The electric current (I) describes the flow of electric charge, which for circuits is

carried by moving electrons through a wire. It is measured in amperes (A). The voltage (U) is measured in volts (V) describing the potential difference or – more figurative – the electric pressure that is applied to the charges. When those two basic units come together, electrical power (P) results as their product $P = I \cdot U$. Power is measured in watts (W).

Now, taking a different angle, electrical power is defined as the electrical work that can be done per time interval Δt, too.

$$P = \frac{W}{\Delta t} \tag{1}$$

Electrical work is the result of the transformation of electrical energy into some other kind of energy, e.g. kinetic or thermal energy. Sadly, it is denoted by W exactly like the unit Watt, that power is measured in. Hence, it is pretty much common to favour electric energy E for the same purpose. Electric energy is the stored or transferred potential energy that can do work with a certain power.

$$E = P \cdot \Delta t \tag{2}$$

Both terms, work and energy are pretty much equivalent and both are measured in wattseconds (Ws) or the more useful scaled unit of kilowatthours (kWh). It should be mentioned that the unit Joule is exactly equivalent to wattseconds $1Ws = 1J$, although it will not be used during this thesis. The previous explanations shall motivate why the term energy consumption is the accurate one, in contrast to power consumption. It is important to differentiate those terms accurately, since a reduction of power does not imply a reduction of energy consumption. It may just cause a longer runtime.[23]

Measurement and Calibration To measure the energy consumption of a electric device, one has to measure the current and the voltage of the circuit. Since the voltage is relatively constant in the distribution networks and is given with $230V$, the power needed, is achieved by adjusting the current. Nevertheless, due to fluctuations in the networks, a reliable measurement must be based on both values. Integrating the power over time yields the energy actually consumed. Hence, energy consumption is a cumulative value.

At the time, there are basically two suitable ways to measure all of those values for a computer. First, intercepting the wire before the computer's power plug. That can be achieved by primitive *wattmeters* that are sold for home usage and known to be highly unreliable or by sophisticated power distribution units (PDUs) that become used in server racks or similar environments.

Second, modern CPUs have integrated measurement circuits that enable detailed metrics per core. The Intelligent Platform Management Interface (IPMI) is a standardized way to access those values. Older computers sometimes support measurement of the energy throughput of the mainboard. Although these features are great to inspect the actual sink of energy in the system, they do not surrogate the measurement of the power supply over all energy-consuming parts, since those values lack the leakage of the power supply and the energy consumption of internal devices, such as hard drives, which are usually directly connected to the power supply rather than obtain their energy through the mainboard.

To calibrate a PDU, for instance, a consumer must be plugged in that offers a reliable and trustworthy power consumption. There do exist load generators with adjustable power levels for professional users, but in many cases a simple light bulb of a certain power level is sufficient to ensure accuracy [24].

3. Related Work

3.1. Energy Efficiency

The energy efficiency of software systems is a cross-cutting concern for users and developers alike.

Pang et al.[12] show that programmers are not sufficiently equipped with knowledge about the reduction of software's energy consumption. Most programmers assume performance optimization as a suitable substitute for energy optimization, which is not always true, as they point out. Their survey supports the criticism on the concept of non-functional properties as mentioned before in Chapter 2, since only 18% of the participants address energy efficiency during development. One participant even claimed that clients "care first and foremost about speed of development, and secondly about reasonable quality and performance". Note that energy efficiency again is substituted by quality and performance. But even if energy consumption would be a major concern in software development, extensive training would be required first. Less than 1% was able to order the hardware components of a desktop computer correctly by their energy consumption and only 5% identified polling techniques for synchronization as a energy efficiency leak. Therefore, they conclude that energy-efficiency-aware programming should become part of undergraduate curricular.

Tsirogiannis et al.[7] made interesting observations on the energy consumption of current datacenter hardware in scale out architectures, during their analysis of the energy efficiency of database servers. As it turns out, more than 50% of the energy consumption is already set by the idle state of a modern computer and 85% of the remaining power at load is consumed by the CPUs. Only a marginal portion can be accounted for HDDs and SSDs. While SSDs provide

perfect proportional energy consumption, HDDs require approximately 80% of their energy consumption jump-fix for activation. Most interestingly "the last 30% of a node's CPU computation capacity comes essentially for free". That leads to the conclusion that hardware consolidation is a promising path to reduce energy consumption in modern datacenters. Meaning to first fully utilizing a computation core before activating another, and of course shutting down under-utilized nodes. That charmingly corresponds with currently vibrant development in virtualization techniques.

3.2. Features and Configurations

A more user-centric design philosophy is demanded by Xu et al.[13]. They claim that flexibility and simplicity must be balanced more carefully. Developers most often expose features as configurable option to users for the sake of flexibility, while not taking into their consideration the problem of *configuration space explosion*. The study focuses on system software, including the Apache webserver which according to Xu et al.[13] provides more than 550 parameters. "Setting them correctly requires domain-specific knowledge and experience.". They were able to show that "too many knobs do come with a cost: users encounter tremendous difficulties in knowing which parameters should be set among the large configuration space." Different suggestions for reducing the configuration space are given, including automatic inference or generation of values and the application of formal models or specifications. The authors distinguish between vertical and horizontal reduction of the configuration space. While vertical relates to the number of parameters, horizontal refers to their possible values. It was possible to shrink the configuration space of a system by half without affecting users by any means, simply by removing unnecessary parameters and reducing the options for the remaining ones, e.g. by transforming numeric ones to enumerative ones or boolean options.

4. Study Design

The fundament of any empirical study is the data it is based on, gathered through experimentation. In the following, the experimental setup, as well as the environment it is deployed in, will be described.

4.1. Environment

All experiments were conducted in a guarded, non-public cluster-environment. Hence, they are laboratory experiments that do not represent any real-world behaviour. On the other hand, external influences, such as random scan probes that can not be foreseen, could be ruled out to a large extend, although the cluster was not used exclusively.

Hardware The cluster provides two hardware variations as described in detail in Table 2. From now on they will referred to as *i5-* , respectively *i7*-Nodes. As it can be easily noticed, both hardware platforms do not diverge too much. The only differences can be found in the local storage technology and the CPU.

name	#nodes	CPU (Intel)	frequency	#cores	RAM	local storage
i5	14	Core i5-4590	3.30 GHz	4	16 GB	256 GB SSD
i7	4	Core i7-4790	3.60 GHz	4(8 ht)	16 GB	500 GB HDD

Table 2: Hardware configurations available as cluster nodes, *ht* means hyperthreaded

The most obvious difference is the support of hyperthreading, which has no effect for the experiments, since the cluster-management software does not support it. The thermal design power (TDP) of the CPUs are the same: 84 W. Both processors are equipped with the Intel AES-NI, which accelerates

the Advanced Encryption Standard AES, as well as INTEL Secure Key, which is a random number generator.

All nodes are connected by a reliably switched 1 GBit ethernet and have access to a shared NFS-mounted network storage.

All nodes are attached to PDUs, which provide monitoring of the utilization of the power supply for each node. They provide measurements for voltage in Volts, current in Amperes, power in kilowatts and energy consumption in kilowatt-hours. All values are sampled with a frequency of one second. Unfortunately, the accuracy of the energy-consumption goes only down to 100 watthours, which is to rough for short-time experiments that do not run for days. However, the power measurements provide accuracy down to a single watt.

From previous calibration measurements it is known that the nodes do consume power in between 20 W when idle and 80 W under full load. The accuracy of the PDUs has been evaluated by attaching a 40 W light bulb. For this study only nodes that were attached to a reliably and accurately measuring PDU are used.

System Software The cluster disposes of a homogeneous software infrastructure, which was not customizable to a large extend due to missing administrative rights. Hence, all software that was subject to customizations had to be compiled, installed, and run as an unprivileged user. That also implies the concession of only being able to bind TCP ports above 1024. The following components of the software stack were predetermined:

- The operating system UBUNTU 16.04.2 LTS
- The runtime environments for

  - Python 3.5.2
  - Java OpenJDK Runtime Environment Version 1.8.0_131
- The required libraries for Apache and Nginx
  - Transport layer security (OpenSSL 1.0.2g)
  - Regular expressions (pcre)
  - Compression (zlib)

Theoretically possible variations in the operating system or the TLS (OpenSSL vs. LibreSSL) implementations were not applicable.

The cluster is managed and accessed through the workload-management tool slurm[1]. It takes care of delegating resources and scheduling jobs. It is unknown in which way that influences the systems behaviour. It allows for limitation of resources, such as memory, number of CPU cores, etc. but does not represent a virtualization technology.

4.2. Webserver Software

Subject to this study is webserver software. Since Ubuntu Linux is the only operating system used for this study, the according to Netcraft [16] at the time most popular webserver could not be taken into account: Microsoft IIS. Thus, only the two major open-source webservers have been chosen for this study: Apache HTTPD and Nginx. The following versions were deployed:

- Apache 2.4.25 compiled with APR 1.5.2 and APR-Util 1.5.4
- Nginx 1.10.1

[1] `https://slurm.schedmd.com`

They were compiled using the compile time configurations shown in listing 1, respective 2, linked to the libraries mentioned in the previous chapter and deployed to user's home directory.

```
./configure
  --with-included-apr
  --prefix=$HOME/apache
  --enable-ssl
  --enable-mpms-shared='prefork worker event'
  --with-port=8080
  --with-sslport=8443
```

Listing 1: Configuration option chosen at compile time for Apache. Need to be passed as a single line.

```
./configure
  --prefix=$HOME/nginx
  --user=nginx
  --group=nginx
  --with-http_ssl_module
```

Listing 2: Configuration option chosen at compile time for Nginx. Need to be passed as a single line.

The Apache webserver offers different *multi-processing modules* (MPMs), which are interchangeable and implement fundamentally different approaches to server design. Three such Apache MPMs and Nginx give a variance of four webserver engines. The MPMs are compiled as shared libraries, what allows to change them by load-time configuration rather than at compile-time. Therefore, none of the used software needs to be compiled more than once.

Prefork is the classical multi-processing strategy of the Apache webserver. Requests are handled by processes. Since starting up processes takes some time a minimum amount is forked in preparation, which explains the name. The advantage of this approach is true separation of different requests. Does some external program need to be executed to answer the request, its own process-environment already exists, for instance, the not thread-save scripting language PHP can be used only with this MPM without additional efforts. The

disadvantages are obvious, too. A full process is a heavy object that consumes a lot of memory and hence can only be scaled to a certain limit. The maximum number of processes to be forked is defined by the `serverlimit`-option. At the same time, this equals to the maximum number of simultaneous requests that can be handled.

Worker favours threads over processes. Every request is handled by a thread, but the processing module does not solely rely on threads and allows multiple processes as well. Inverse to the previous module, it reduces memory usage and scales faster, since threads can be spawned faster. However, still every thread comes with a memory overhead and it must be taken care for thread-safety. The number of requests that can be handled simultaneously is the product of the maximum number of processes (`serverlimit`-option) and the maximum of threads per process (`threadlimit`-option).

Event is the latest module and represents the adoption of the approach NGINX uses. The basic idea is to handle all requests with a single eventloop and dispatch time-consuming operations, such as I/O to dedicated threads. This approach reduces memory consumption significantly, while it guaranties thread-safety at the same time. The APACHE-implementation allows the configuration of process- and thread-count, analogous to the worker-MPM by `serverlimit`- and `threadlimit`-option.

Nginx pioneered and still implements the same concept as APACHE does with the event-MPM, but the configuration is more simplistic. The maximum number of requests that can be handled simultaneously is defined by the `worker_connections`-option. For comparability and easier configuration generation, this is labelled as `serverlimit` from now on. There does not exist

any concept of configurable multi-threading like in the case for the Apache-implementation, but there is an option to influence the number of processes, which is by default equivalent to the number of CPU-cores of the system.

4.3. Feature Models

Only a small subset of the features that modern webservers offer were chosen for this study. The reason is that many features do not affect energy consumption or would require a distinct usage scenario, which makes it hard to compare with other features. The models of the four engines were kept mostly congruent, although even the different MPMs of Apache did not fully allow that (e.g.: due to the already in previous section mentioned processing- and threading-configurations). The options `serverlimit` and `threadlimit` are numerical. For the sake of simplicity, they were transformed to enumerative parameters for this experiment. Table 3 shows the values for the different engines. They represent common choices. Note that for all engines except *prefork* the theoretical count of simultaneous connections ranges from 1024 to 4096. The lowest values marked with a star are the configuration defaults of the systems. These are the only features that are mandatory and they provide alternative choices.

	serverlimit			threadlimit	
	low	mid	high	low	high
nginx	1024*	2048	4096	—	—
prefork	256*	512	1024	—	—
worker	16*	—	32	64*	128
event	16*	—	32	64*	128

Table 3: Details of the non-boolean mandatory feature options. (*default values)

Beside those two numeric features, up to eleven binary and one enumerative feature with eight possible choices where selected as features. Some of the binary features are not binary in the systems original appearance and have been discretized. Three of them are not supported by NGINX, either because they are APACHE-specific like in case of `htaccess` or because they require third party modules. All binary features are modelled truly optional. There are no alternatives or dependencies. Table 3 presents an overview of the features covered by the different systems. A graphical representation of the feature models can be found in Appendix A, Figure 16. Most features are modelled as binary choices.

feature	APACHE			NGINX	description
	prefork	worker	event		
serverlimit	yes	yes	yes	yes	number of processes
threadlimit	no	yes	yes	no	number of threads
keepalive	yes	yes	yes	yes	boolean feature
errorlog	yes	yes	yes	yes	boolean feature
accesslog	yes	yes	yes	yes	boolean feature
clientcache	yes	yes	yes	yes	boolean feature
servercache	yes	yes	yes	yes	boolean feature
compression	yes	yes	yes	yes	boolean feature
symlinks	yes	yes	yes	yes	boolean feature
sendfile	yes	yes	yes	yes	boolean feature
dns lookup	yes	yes	yes	no	boolean feature
status	yes	yes	yes	no	boolean feature
htaccess	yes	yes	yes	no	boolean feature
encryption (TLS)	yes	yes	yes	yes	8 selected suites

Table 4: Comparative overview of the feature model for all servers. Yes means feature is supported, no means not.

4.3.1. Description of Features

In the following the features are briefly described:

keepalive allows the TCP/IP stack to keep an established connection open for succeeding communications, until the defined timeout. That became introduced with HTTP 1.1. to reduce round trip times (RTT). When activated, only new HTTP requests need to be established, while TCP connections can be reused.

errorlog usually is not a boolean option and should never be turned off. It allows adjusting the common syslog levels. The feature enables switching between the levels `error` = $false$ and `warning` = $true$.

accesslog allows detailed and fine-tuned logging of requests. Usually, it is configured by format strings. For the study, it can be switched *off* or it logs with the established default called `combined` log format.

clientcache takes advantage of the clients capability of caching previously seen contents locally. So called *conditional requests* allow the server to indicate that no changes were applied to the resource since the last request and therefore the cached version can be used. The HTTP response code used for that is 304. By allowing the server to send these response codes, the clientcache can be controlled by the server's configuration.

servercache allows to keep a limited amount of data in main memory. Since it prevents repetitive hard disk access, it should decrease latency of the response.

compression reduces network traffic by compressing the data sent with the commonly used *deflate algorithm*, also known as `gzip`.

symlinks allows the webserver to follow symlinks if they are encountered during the request mapping.

sendfile takes advantage of the operating system capability for direct connection of two file descriptors in the kernel, mitigating further processing in user space.

dns lookup requires the APACHE webserver to do a reverse DNS lookup to resolve the requesting clients IP address into a domain name. This may cause additional network overhead, which, in turn means additional latency.

status requires the server to keep track of some simple indicators of its own performance.

htaccess is a feature of the APACHE webserver that allows to distribute partial configuration files all over the server folder structure. Since they can change during runtime, they must be scanned at every request.

encryption is a complex feature that will be described in the following paragraph.

Encryption Feature has its own complexity and requires a bunch of configurations options to be set. For this study, it is reduced to eight distinct TLS-Suites. Usually, the TLS feature is configured by a filter string that constrains the possible combinations of different algorithms to a well defined subset. A cipher suite consists of constraints for algorithms for up to four use cases: key exchange, authentication, block cipher, and MAC-Hashes. To ensure a certain combination of algorithms during the experiments, negotiation between client and server is prevented by fixating the cipher suite to a single

choice. The selection of those choices shown in Table 5 follows at current recommendations. The encryption feature is the only one beside the processing options that is modelled as alternative choice.

Two self signed certificates according to the X.509 standard were generated. One for encryption with an ECDSA key of type `prime256v1` and a second with an RSA key of 2048 bit size. Due to observed incompatibilities during the prestudy experiments, ECDSA variants were discarded.

key exchange	authentication	block cipher	MAC
DHE	RSA 2048	AES128	SHA256
DHE	RSA 2048	AES128-GCM	SHA256
DHE	RSA 2048	AES256	SHA256
DHE	RSA 2048	AES256-GCM	SHA384
ECDHE	RSA 2048	AES128	SHA256
ECDHE	RSA 2048	AES128-GCM	SHA256
ECDHE	RSA 2048	AES256	SHA384
ECDHE	RSA 2048	AES256-GCM	SHA384

Table 5: The eight fixed cipher suites for the TLS configuration

4.4. Configurations

Configurations are the specific instances of the feature model. They are derived through selecting values for all options. The intuition of a *configuration space* gives an impression of the challenges when configuring complex systems. Its dimensions equal to the number of features and the value range of each dimension equals the choices that are possible for that feature. Table 6 shows how many configurations are possible, even with the limited subset chosen. Over all 209,664 configurations could be generated and executed. Running

each for just 10 minutes would take almost four years. Even parallelization would not help to run experiments within the time given for a thesis.

	Apache			Nginx
	prefork	worker	event	
serverlimit	3	2	2	3
threadlimit	—	2	2	—
binary features	$2^{11} = 2048$	2048	2048	$2^8 = 256$
encryption	9	9	9	9
dimensions	13	14	14	10
configuration space size	55296	73728	73728	6912
training samples	220	245	245	183
testing samples	10	10	10	10

Table 6: Size of the configuration spaces a.k.a. number of possible configurations.

4.4.1. Sampling Strategies

Due to the need of selecting a representative subset of configurations, different sampling strategies have been applied. The basic dataset for all analysis consists of configurations derived through the following sampling strategies taken from [15]:

Feature-wise (FW) samples the subset of configurations where each feature is kept default except for one. This sampling strategy intentionally avoids any interactions among the features.

Negative feature-wise (NFW) is defined as the negation of feature-wise sampling. All binary features are active or deviate from its default, except for one.

Pair-wise (PW) chooses values that differ from default for any combinations of two features at a time. This sampling obviously allows investigations for pair-wise interactions.

It has been shown that the combination of those sampling strategies produce sufficient data to train reliable performance models [25]. Table 11 in Chapter 6 provides numbers on the size of the samplings. Additionally, a separate set of ten randomly chosen configurations was build for model evaluations. Those ten configurations, listed in Table 7 were the same for all four webserver engines and not part of any of the other samplings.

process/thread limit	binary features	cipher suite
high	01001100000	DHE-RSA-AES128-SHA256
low	00010110000	DHE-RSA-AES256-SHA256
low	01000101000	ECDHE-RSA-AES256-SHA384
low	01010100000	ECDHE-RSA-AES256-GCM-SHA384
low	10000000000	DHE-RSA-AES128-GCM-SHA256
low	10010101000	ECDHE-RSA-AES256-SHA384
low	10110000000	ECDHE-RSA-AES128-GCM-SHA256
low	11000001000	off
low	11000101000	ECDHE-RSA-AES128-SHA256
mid	10000000000	DHE-RSA-AES256-GCM-SHA384

Table 7: Randomly chosen configurations the test sets consist of. See Figure 2 for the mapping intuition.

4.4.2. Identification of Configurations

In order to assure quick resolution of configurations and results, a straightforward mapping of an entire configuration to a comprehensible string was established. As illustrated in Figure 2, it starts with the engine, followed by

the values for `serverlimit` and `threadlimit`. In cases where `threadlimit` is not defined, it is left blank. The next block is the ordered summary of all binary features and encoded with $\Sigma = \{0, 1\}$. The last block is the unencoded cipher suite string or if encryption is not in use, the string `off`. All five blocks are separated by dashes for easy tokenization.

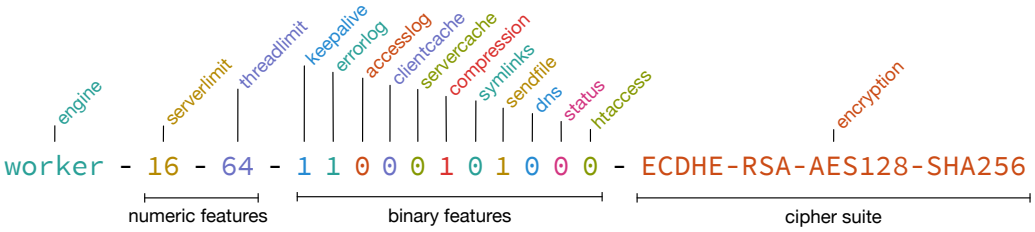


Figure 2: Mapping of configurations to identification labels

4.5. Workload Generation

To be able to measure any recognizable energy consumption above idle power, a substantial workload must be applied. Load-testing tools[2] are the appropriate solution. Those applications are designed to perform repeatable scenarios of real workloads for a server system. Therefore, they are capable of simulating a variable number of clients in parallel and a variable request rate to different resources. Fortunately, there exists a fair number of such tools. Sadly, however, they strongly vary in their feature sets and there is no single tool with a complete feature set, as the following short survey indicates.

4.5.1. Load-Testing Tools

The field can be roughly subdivided into three categories. Reading Table 8 from bottom to top, the first two tools `wget` and `curl` are not actually load-testing tools, but protocol-testing tools. They provide very fine-grained access to the protocol settings, but they do not support to do the same task rapidly over and over again. Since they do not offer task repetitions, they do not offer any statistical summaries as well, and therefore do not provide output, e.g., in the form of CSV files. Both only offer command-line interfaces.

The quite heterogeneous middle field has the exclusive command-line interface in common with the previous ones. Beside that, all tools are designed for load and stress testing. There are veterans, such as the Apache Benchmark (ab) and very modern tools, such as *vegeta* and *wrk 2*. While the new protocol version HTTP/2 is already implemented by all servers and most clients, almost all of the load-testing tools are lacking it; except for two out of that middle field. Those two are relatively new competitors that started from scratch with HTTP/2 in mind. Due to their simple command-line interfaces, most of the

[2] Also called: Performance-testing tools

middle field tools do not support plans of varying workloads, neither do they provide any further statistical outputs than cumulative summaries. Therefore, they are quite primitive compared to the last group of full-blown systems.

Tool	Language	HTTP/2	TLS	Header	Inferring	CSV
jmeter	Java	no	yes	yes	yes	yes
gattling	Scala	no	yes	yes	yes	no
locoust	Python	no	yes	yes	manual	manual
vegeta	Go	yes	yes	yes	no	no
wrk 2	C	no				no
h2load	C	yes	yes	yes	no	no
ab	C	no	yes			no
tsung	Erlang	no				no
yandx.tank	Python		SSL			no
wget	C	no	yes	yes	yes	no
cURL	C	yes	yes	yes	no	no

Table 8: Overview of load testing tools.

The top three tools share the framework character, which allows custom extensions, versatile logging of statistical data, and distributed load testing. None of the tools does support HTTP/2 testing at the moment. JMeter and Locoust do mention it on their issue tracker, but no support exists at the time. For JMeter, an external plugin and patch exists, but it does lack to many features.

There are two major differences between the established JMeter and the two newcomers Gatling and Locoust. While JMeter is based on classical multi-threading, the other two favour asynchronous event queues for load generation. Furthermore, load-testing plans for JMeter are configured through

a graphical user interface and stored in XML files. By contrast, GATLING and LOCOUST both offer a domain specific language (DSL) for definition of a load-test plan.
To simulate a webbrowser's behaviour, embedded resources such as scripts, style sheets, or images, must be loaded as well. The automated parsing and downloading of embedded resources from the loaded HTML, also called resource inferring, is only usable with JMETER and GATLING without further efforts. For LOCOUST, it can be realised through a custom extension.
Most disappointing is that only JMETER writes a full report of all requests and measurements to a comma-separated-value (CSV) file. Again, LOCOUST can be extended easily, but even though, it does not provide the same granularity.
At least, all tools throughout all three categories are capable of encrypted communication (HTTPS) as well as they allow free manipulation of any header fields.

JMeter Due to the sophisticated logging capabilities and the possibility of embedded resource inferring, the decision was made for JMETER in version 3.1.
Unfortunately, practical problems arise that might discourage the usage of JMETER in possible follow up studies. As it turned out, the logging capabilities that could be achieved through extension of LOCOUST were sufficient, too. First, load-testing plans that utilised the resource inferring feature were discarded due to too many failed requests. Instead repetitive *out of memory exceptions* occurred and required raising the heap size from 512 MB to 6 GB in the startup shell script of JMETER in `bin/jmeter`, which is the tribute to be paid for the multi-threaded approach.

```
HEAP="-Xms512m -Xmx512m"
HEAP="-Xms512m -Xmx6144m"
```

Listing 3: Adjustment for heap usage of JMETER. First line: Original configuration, second line: altered version.

It remains unclear if observed performance limitations must be accounted to the servers or to JMeter. A complete evaluation of load-testing tools was not intended to be part of this thesis, but it is worth the effort to support further research in this area.

4.5.2. Workload Plan

An extensive amount of time was spent on defining a load-testing plan that generates enough workload to be able to measure a significant power usage above idle.

First design approaches reflected the intention of a comparison of different protocol versions from HTTP 1.0, HTTP 1.1, and HTTP/2 as one of the features. Therefore, the first load-testing plan targeted specific advantages of them, such as the *multiplexing* and *server push* in the case of HTTP/2. The intuition was to embed larger resources, such as images into an HTML page, such that the system should be able to benefit from the enhancements. Due to the previously mentioned lack of HTTP/2 support from the load-testing tools, the only feature that remained is the *keepalive* feature, which represents one the major enhancements from HTTP 1.0 to HTTP 1.1.
A further finding of the prestudy was that, the concept of downloading embedded resource caused problems. Embedded resources were not handled with the same care as the main requests. Often, they failed and did not perform a sufficient number of retries, because of too many concurrent connections. Since an entire request becomes marked as failed if one subsequent request failed, that resulted in many failures. Hence, the resources targeted by the load-test plan were reduced to plain, solitary HTML files.

Another requirement for the load-testing plan were varying workloads. Large resources saturate the network link with just a few requests, while tiny files

can reach upper bounds of simultaneous connections. More on that in Section 4.7, which summarizes the lessons learned during prestudy. Table 9 shows the test plan as it was performed for the study. It consist of five different load configurations, all separated by a 10 second pause. A 40 secound warmup phase ensures that the startup processing of the server does not influence the measurements. A 60 second wait to the end of the work loads is added to allow queued requests to be worked off. All workloads are applied for 30 seconds. Overall, one experiment takes 5 minutes.
Three HTML files of different sizes were deployed as resources. The workloads' labels as printed in Table 9 refer to the filesize and whether the characteristic of the load is the one of a ramp (R) or a plateau (P). As the names suggest, a ramp should produce increasing load, whereas a plateau should aim to keep the workload constatnt at a certain level. Interestingly, that effect can hardly be recognized in the time-series visualizations in Figures 4 and 5.

label	duration	size	threads	repetitions	target	throughput	
	sec	Byte	#	#	req/sec	req/sec	MBit/s
			— 40 sec warm up —				
R1k	30	1 k	500	4,000		66,666	533
			— 10 sec pause —				
P1k	30	1 k	3,000	1,200	100,000	120,000	960
			— 10 sec pause —				
P7k	30	7 k	1,200	300	10,000	12,000	672
			— 10 sec pause —				
R7k	30	7 k	500	1,000		16,666	933
			— 10 sec pause —				
R3M	30	3 M	10	40		13	320
			— 60 sec left until end —				

Table 9: Description of the different workload intervals of the experiment.

As the local network offers a connection speed of 1 GBit/s, that is equal to 125 MByte/s theoretically maximum throughput, the different loads where designed to keep below that boundary. Nevertheless, the number of simultaneous request appears to be limited at the boundary of approximately 15,000 requests per second, although two loads do apply a much higher pressure. The reason for that behaviour remains unknown. Several factors could cause it: The server itself is the most plausible candidate, probably due to overseen misconfigurations. However, the same argument can be made for the load-testing tool, or the TCP/IP stack of the operating system. They could be configured too conservative as well. However, as Table 9 shows, the physical network link can be ruled out, since the throughput definitely does not raise above 1 GBit/s.

A final element of the workload plan worth an explanation is the variation of the *threads* and *repetitions* parameters. The threads are used to simulate clients. Therefore, they represent the number of simultaneous connections to the server. By contrast, the repetitions tell one client how often to repeat the request. The product of the two represents the number of requests performed throughout the interval of half a minute. As a matter of fact, requests performed by the same client simulating thread, should be able to leverage certain features that are chosen for this study. For example, one client has one cache. That means that a configuration with activated *clientcache* feature should perform only one request per thread and rely for all subsequent requests on its local cache. Or, as another example, configurations with the *keepalive* feature should be able to reuse the same connection.

4.6. Experiment Setup

As already mentioned, the experiments were deployed in the controlled environment of a cluster. Moreover, the cluster can be considered homogeneous, since the experiments were exclusively run on nodes of the *i5*-type.

Figure 3 exemplifies the deployment of an experiment in the cluster. Every experiment uses two nodes. One running the webserver, the other running the load-testing tool. All scripts were designed such that the use of multiple load-test clients would theoretically be possible, but the benefit gained out of the workloads distribution did not pay of.

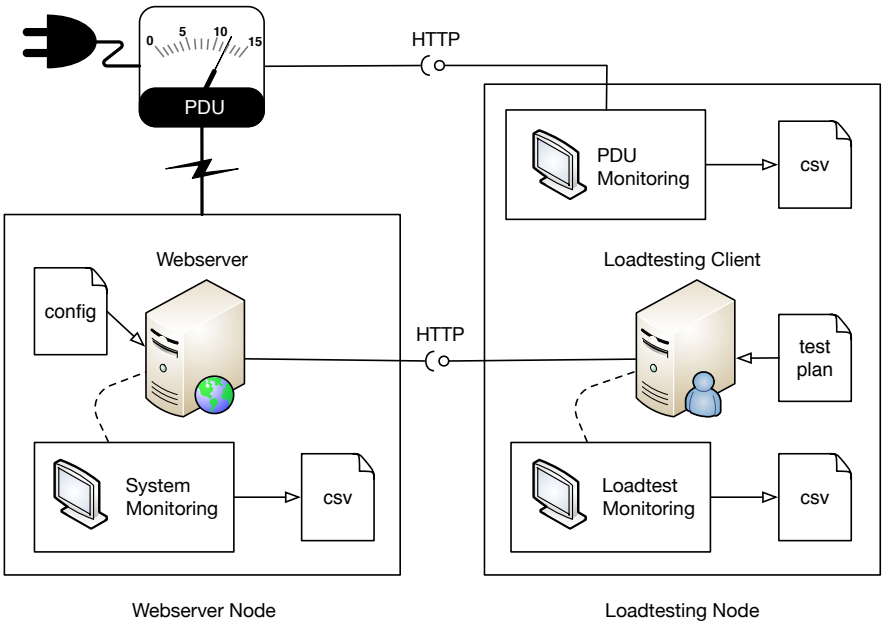


Figure 3: Deployment of the experiment setup in the cluster.

A single experiment run takes five minutes. As discussed in Section 4.5.2, the server becomes started immediately with its configuration, while the load testing tool waits for 40 seconds before performing the test plan. From the first

moment, a system monitoring runs on the same node as the server, which logs system performance metrics, such as CPU utilization, memory consumption, and network load. The same monitoring was running at the client node, but the data was only used during the prestudy.

Another monitoring script runs also at the client node from the first moment to monitor the server-nodes PDU. It logs voltage, current, power, and energy usage. The accuracy of those values has been already discussed in Section 4.1.

The load-testing tool itself logs every single request it performs. The data logged includes the label of the workload, success and failure indicators, response messages, transferred bytes in both directions, latency and response time. Since those workload logfiles can become up to 100 MB for a single five minute experiment, they were compressed with `gzip` immediately after they were written.

All scripts log their data prefixed with *UNIX epoch timestamps*. The system clocks are calibrated through the network time protocol (NTP). Therefore, they are sufficiently exact for synchronizing the logfiles posterior, since NTP guaranties milliseconds accuracy.

The two, so far unmentioned scripts for monitoring system utilization and the PDU, are self-written in the Python programming language. Both benefit extensively from the capabilities of the asynchronous IO libraries of Python (aio). Hence, they provide reliable logging, while the impact on the systems performance remains almost unrecognisable. Both scripts are configured to perform a slight oversampling to increase precision. The PDU monitoring logs every 0.3 seconds, whereas the system monitoring every 0.5 second. The data logged will be sampled down to one second posterior.

4.7. Prestudy

A quite large amount of time was spent prior the actual study in finding a suitable and realisable study design. This chapter aims to summarizes the lessons learned during prestudy.

4.7.1. Lessons Learned

Protocol Versions As already mentioned in Section 4.5.2 incorporating HTTP/2 support as a feature would be of interest, but was not possible due to lack of support by the current load-testing tools. This remains an open issue and might be worth the custom development of an appropriate load-testing tool.

Conditional Requests The effect of this HTTP feature was already mentioned in Section 4.3.1. *Conditional Requests* do allow the client to send some information about locally cached versions of the resource with its request. That means a server can respond just by acknowledging the reuse of the cached version. That behaviour is used exhaustive by real-world clients, but it had to be disabled for the study's purpose. That is because otherwise, the server would not experience any further workload, after every virtual client has made its first request and filled its cache. Therefore, it is deactivated for all configurations except for the ones that have the feature *clientcache* activated.

Open File Descriptors Webservers heavily depend upon the infrastructure that the underlying operating system offers. One important and often necessary adjustment concerns the open file descriptors. In UNIX systems, any opened file, data stream, network connection is represented as a file descriptor in the system. The number of simultaneously allowed open file descriptors is

arbitrarily limited. Historically, the values chosen for that boundary are very low. Decades ago those rigorous limits made sense for security reasons, but for modern computers, they can be raised without harm. The current limit can be inspected by the command `ulimit -n`. For the used UBUNTU systems, it was set to 1024. That is obviously not sufficient, neither for the server, nor the load-testing tool. It did limit the number of simultaneous connections to that number. Therefore, the value was raised to 8192 for all the experiments ran for this study.

Favor the many small over the few large! Requests that expect large response payloads do rapidly fill up the network connection. A webserver has only work to do until the response is assembled and it can be sent. Especially for static files, as used exclusively in this study, the actual transmission of the payload is delegated to the operating system. To keep a webserver busy, many small requests are far more effective. Of course, the upper bound for all requests is always capacity of the network link. For the same reason, the embedded resource inferring feature of JMETER was not used, as discussed in Section 4.5.2.

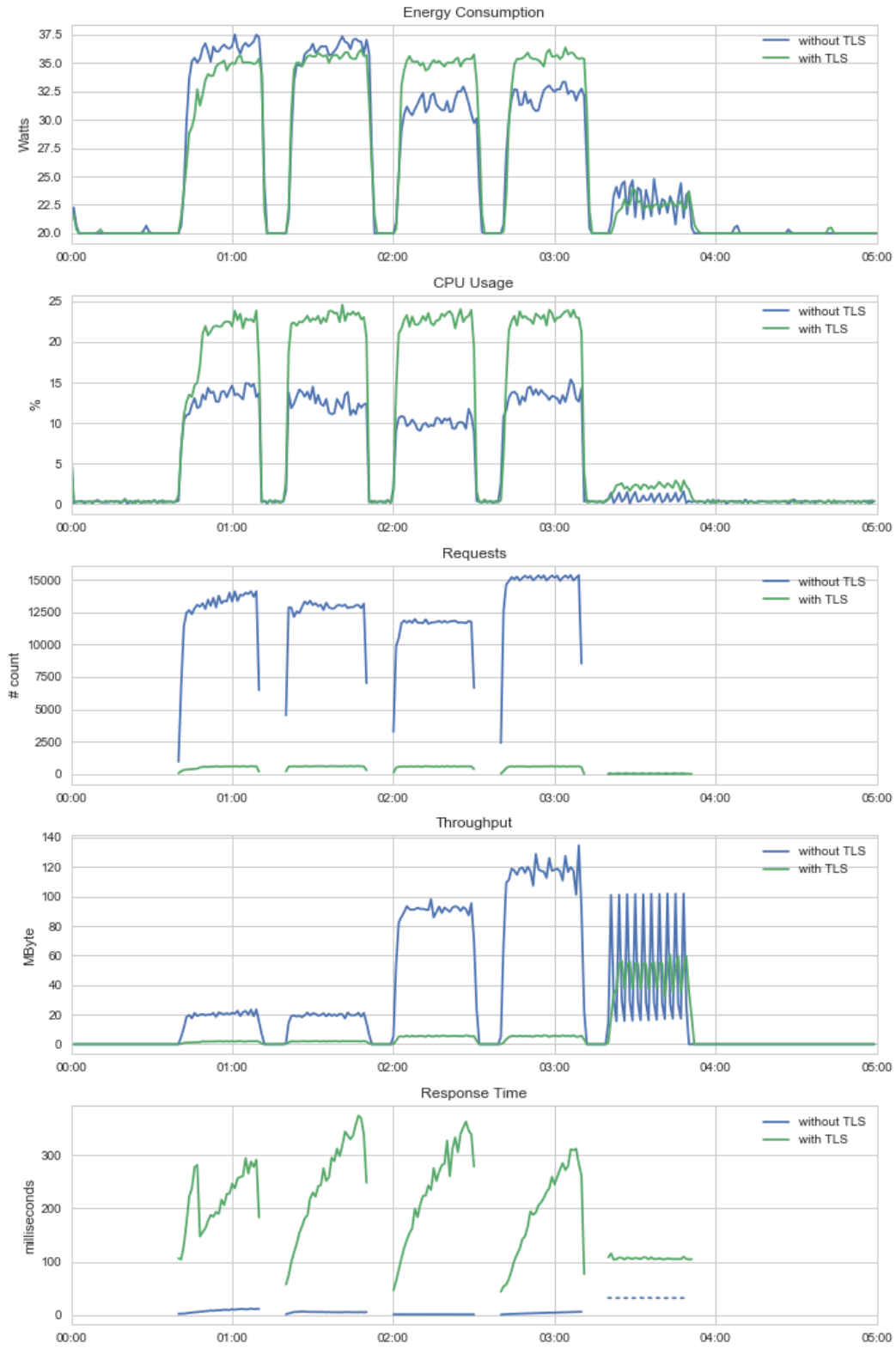


Figure 4: Observed performance degeneration of Nginx configurations with and without encryption. Exemplary configurations visualized as time series: nginx-1024–00000000000-DHE-RSA-AES128-GCM-SHA256 and nginx-1024–00000000000-off

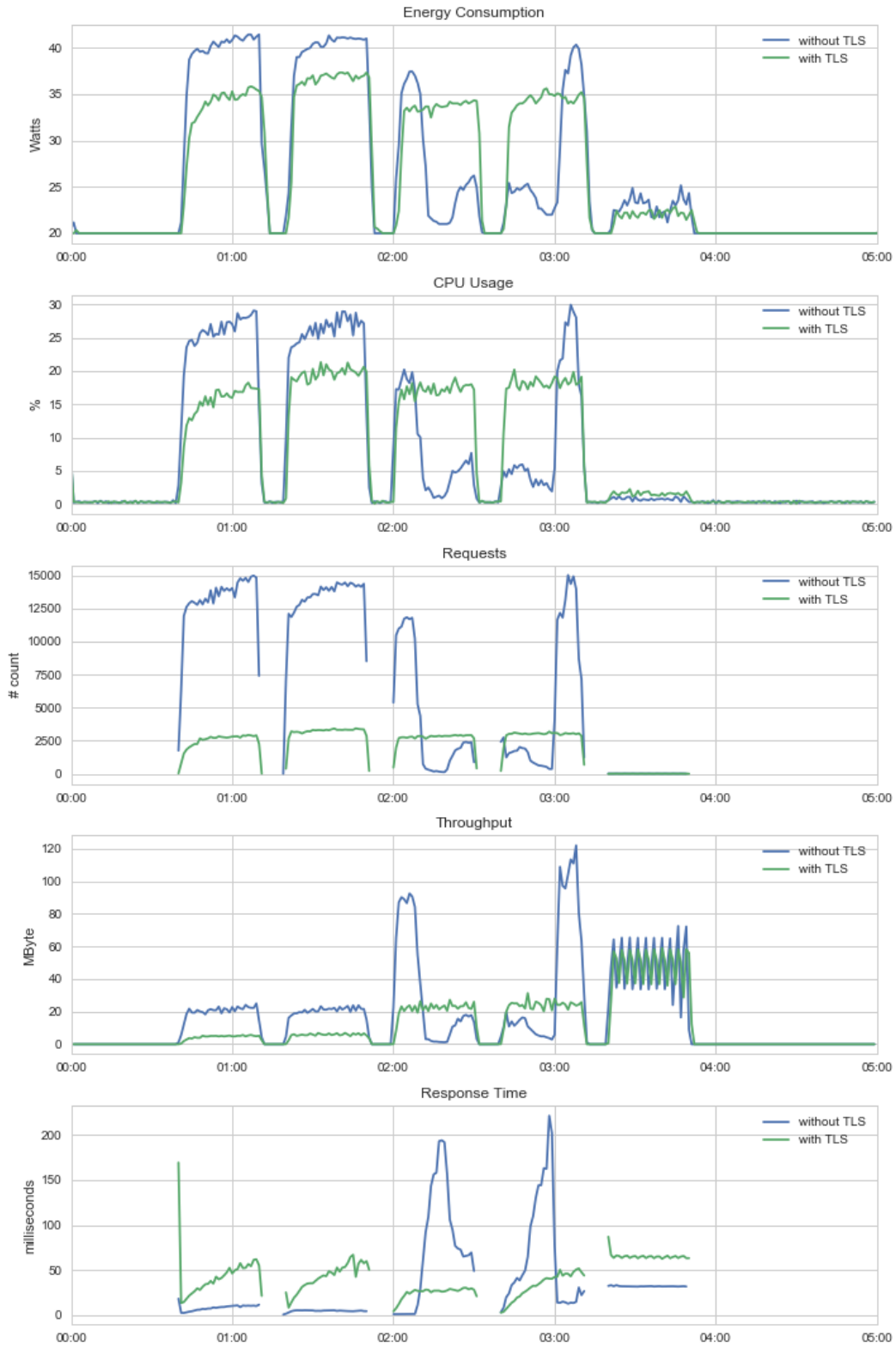


Figure 5: Observed performance degeneration of Apache worker configurations with and without encryption. Exemplary configurations visualized as time series: worker-16-64-00000000000-ECDHE-RSA-AES128-GCM-SHA256 and worker-32-64-00000000000-off

4.7.2. Open Issues

Some issues remained unresolved. Taking a look at the Figures 4 and 5, it must be acknowledged that the different servers perform significantly worse with encryption turned on. But the CPU still has spare resources to offer. It could not be resolved if that degeneration in throughput is caused by the server, the operating system, or the load-testing client.

Even more mysterious is the sudden decline of throughput in Figure 5 for the Apache worker engine, but similarly observed for the Apache event engine. It occurs during the third load interval, after it reached a peak. Suddenly the throughput declines and does not recover before the end of the fourth interval. The best guess so far, is to blame the TCP stack for that behaviour, since it could be caused by the flow control that tells a sending node to drop the packet rate, if the receiving node is overwhelmed.

5. Research Question

This thesis is situated within a research field that aims for empirical exploration of the influences of features of configurable software systems on the properties, especially the non-functional properties of those systems. Non-functional properties can be any hardware utilization metric, any performance metric or, with focus on this thesis "Do features matter for energy consumption?": energy consumption.

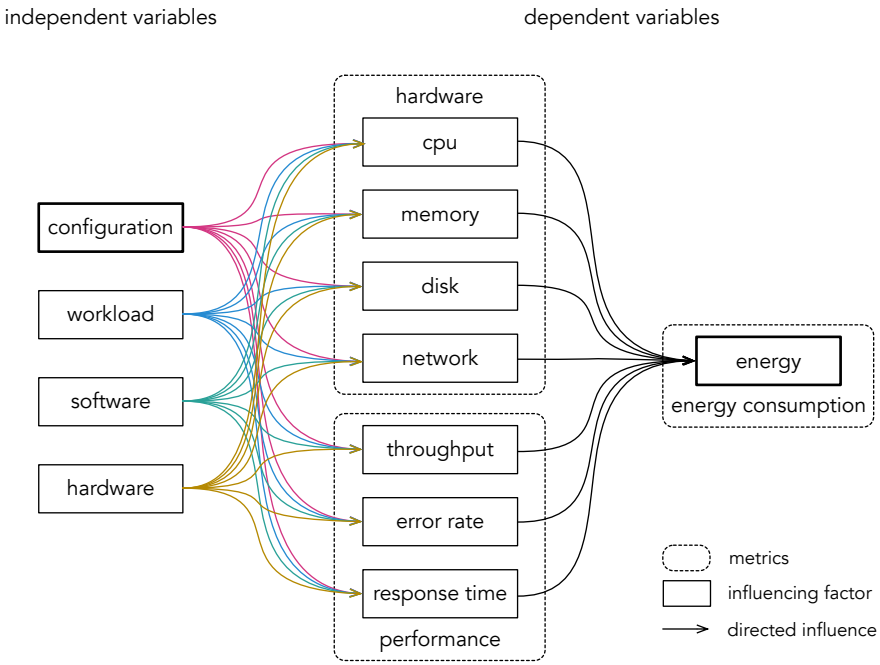


Figure 6: Independent and dependent variables as influencing factors for energy consumption.

Nevertheless, energy consumption remains difficult to measure, even though modern hardware became more sophisticated in that. Without special hardware, users are reliant on estimations that are based on other non-functional properties that are more straight forward to measure. Hardware metrics do

play an important role for that, since they inform about the utilization of the components that actually do consume the energy.

The question if performance metrics can play a similar role like hardware metrics do for estimating energy consumption was already identified as controversial in the introduction chapter. Figure 6 illustrates the constellation. The explanatory or independent variables are the hardware, software, the workloads, and, of course, the features. The hardware and performance metrics are kind of intermediate here. They are explained by the independent variables themself, but they can be explanatory for the energy consumption as well.

The main interest lies on the influence of configuration choices on energy consumption. Furthermore, there might be interactions between features to be discovered. In particular the following research questions shall be answered onwards.

RQ 1 *Do features correlate with energy consumption?*
Here, features are atomic parts, assembled to a configuration representing the independent variables that may explain changes in the dependent variable, the energy consumption. Therefore, a linear regression model will show up whether relationships exist. Hence, subsequent questions are:

RQ 1.1 *What are the effects of individual features?*
To answer this question, the coefficients of a multiple regression for the feature-activation matrix and the energy consumption values shall be inspected.

RQ 1.2 *Are there any interactions between features?*
This is acomplished analogous to the previous one, except for an extension of the feature-activation matrix that has to incorporate the activations of feature pairings.

RQ 1.3 *What is the distribution of relevant and irrelevant features with respect to energy consumption?*

To provide an intuition about this, a histogram of the coefficients of the regression model will suffice.

RQ 2 *Does performance correlate with energy consumption?*

Here, performance is defined as response time. Hence, the question boils down to: Are the mean values of response time and energy consumption drawn from the same distribution?

RQ 3 *What is the workload's influence on the energy consumption of features?*

The workload is known to affect the performance of webservers. Does it influence the way features affect the energy consumption as well? As a first step, the following question should be answered:

RQ 3.1 *Do the most energy efficient configurations maintain rank for varying workloads?*

Therefore, the energy consumptions, separately calculated for the different workload intervals, will be ranked and inspected, whether the top ten configurations maintain position across the varying workloads.

To answer this questions the hardware was kept stable, throughout the experiments. Same applies to the operating system software. The subjects of the study, the webservers were varied.

6. Evaluation

Running all 933 configurations for training and testing, each 3 times for 5 minutes, took 233.25 hours, or more human readable 9 days 17 hours and 15 minutes, and produced 138.7 GB of logged data. That does not include any test runs prior to the actual study. Table 10 provides statistics of the size for the different samplings. Before evaluating the data gathered with respect to the proposed research question, it needs some preprocessing described onwards.

sampling	prefork	worker	event	nginx
feature-wise	3.3	3.4	3.4	3.4
negative feature-wise	4.9	4.7	4.8	4.6
pair-wise	25	28	28	19
testset	1.6	1.5	1.5	1.6
sums (GB)	34.8	37.6	37.7	28.6
total (GB)				138.7

Table 10: Size of acquired data from experiments in Gigabytes

6.1. Preprocessing

The refinement of the raw data was realized in two separate processing steps. First, a single experiment with its three repetitions, each of which produced three log files, formatted as comma-separated values (csv), was aggregated to a single second-wise log file of averaged values by the following steps:

1. Temporal alignment of different metrics in different logfiles according to unix timestamp.

2. Redefining the index to time deltas beginning with zero at start.

3. Downsampling to a frequency of one second, while averaging all fields except for success and failure counts which were summed up.

4. Joining the three runs of the same experiment to one timeline to account for measurement bias.

5. Aggregating all values at the same timestamp by averaging again.

6. Cutting off the first 40 seconds of idle warm-up phase.

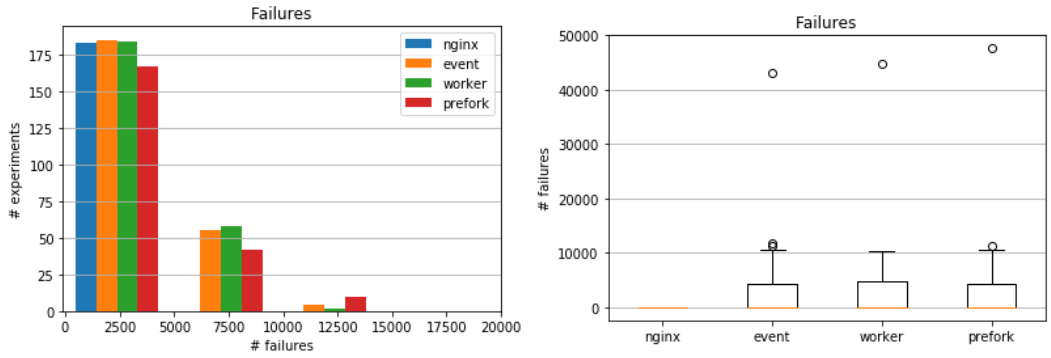
The second processing step aggregates all experiment runs with the same web-server engine in a single table, in which each experiment is represented by a single row. The first part of a row is the binary feature vector representing the configuration of the according experiment. All vectors combined represent the feature-activation matrix. The second part of a row repeats 6 times, once for the entire experiment and 5 times for every individual workload. This part provides averaged response times and electrical power, as well as summed up success and failure counts, and energy consumption.

		prefork	worker	event	nginx
training set	configurations	220	245	245	183
	– feature-wise	21	22	22	19
	– negative feature-wise	19	20	20	16
	– pair-wise	180	203	203	148
	erroneous executions	56	69	83	0
	– mean of failed requests in %	1.1915	0.9112	0.7702	—
testing set	configurations	10	10	10	10
	erroneous executions	0	0	2	0
	– mean of failed requests in %	—	—	0.0002	—

Table 11: Total number of configurations, respective errors during execution and the mean error quotient of the erroneous executions.

Only the timelines in Figures 4 and 5 are produced based on the data of the first refinement step. All other plots are based on data of the second preprocessing step.

Beside the number of individual configurations run for each engine, Table 11 provides a first glimpse of the data. It shows how many of the experiments had failed requests during execution. Furthermore, only for the erroneous experiments, the averaged percentage of failed requests is stated. Figure 7 provides a more detailed view on the distribution and variance of failures during the experiments. None of the outliers or the erroneous executions were removed from the data for further analysis. Since the case of the failures and outliers remains unclear, the were not treated differently.



(a) Distributions, outliers are cut off

(b) Boxplots, including outliers

Figure 7: Visual descriptions of the failure rates occurred during all experiments.

6.2. Results and Discussion

6.2.1. Features and their Interactions

To investigate the influences of features and their interactions on the energy consumption two linear regression analysis were performed. Initially, it was ex-

perimented with Lasso and Ridge regression in comparison to Ordinary-Least-Squares (OLS) regression. While Lasso regression leads to more interpretable and expressive models by enforcing certain coefficients of the model to be set to zero, Ridge regression improves the prediction accuracy by reducing overfitting through limiting the size of the coefficients. Lasso and Ridge regression do both work for ill-posed problems were a linear regression by OLS is not able to provide an unambiguous model. So both can prevent overfitting, but might lead to underfitting models. Since the learned models did not differ significantly, the decision was made to stay with OLS regression. Both regressions calculated are multiple regressions, as they must explain the relationship of more than one explanatory variable to energy consumption. Multivariate regression, which would be able to incorporate more than one value for energy consumption per configuration, was not performed. The data sets achieved through feature-wise, negative feature-wise, and pair-wise sampling were combined to form the training data set used for both regressions.

RQ 1.1 *What are the effects of individual features?*
In order to answer *RQ 1.1* a multiple regression model with the feature-activation matrix as the explanatory variables and the vector of energy consumptions as the explained variable was calculated. The effects obtained as coefficients of the regression model are listed in Table 12. They are sorted by the worker column to provide better intuition. The *clientcache* feature is *the* energy saving feature for all Apache processing modules, but surprisingly that is not the case for Nginx. Nginx seems to save energy through encryption, which, however, is certainly not the case. A first attempt for explanation can be made by the observation that only the cipher suites using the elliptic curve variant of the Diffie Hellman key exchange (ECDHE) appear to save energy, which could be true compared to the non elliptic curve variants. But, that hypothesis, is contradicted by the Apache family. More realisti-

cally, the energy-saving effect is caused by the decline of the handled requests, when encryption is activated. Further investigations are needed. Similarly, the *keepalive feature* allows only NGINX to save energy.

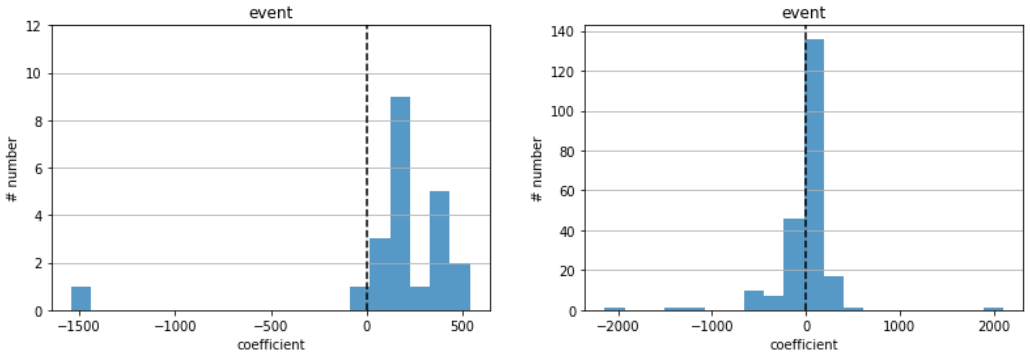
feature	nginx	event	worker	prefork
ECDHE-RSA-AES256-SHA384	-129.42	537.31	426.63	640.86
servercache	-122.13	405.99	325.37	115.40
compression	25.30	358.32	320.93	224.12
ECDHE-RSA-AES128-SHA256	-224.08	395.41	291.15	230.40
DHE-RSA-AES128-SHA256	209.38	349.15	274.55	330.94
DHE-RSA-AES256-SHA256	205.99	359.21	268.78	225.93
sendfile	93.57	300.87	199.09	131.35
keepalive	-103.15	489.97	198.43	290.79
status	—	152.38	158.50	51.06
threadlimit-high	—	148.90	147.86	—
symlinks	-43.04	176.22	144.39	49.01
errorlog	20.02	178.98	143.37	39.81
htaccess	—	163.24	139.63	40.23
serverlimit-high	4.79	137.42	106.13	-0.00
accesslog	63.81	108.42	104.27	30.69
dns	—	96.50	87.98	-14.84
ECDHE-RSA-AES128-GCM-SHA256	-317.76	211.45	81.57	31.21
ECDHE-RSA-AES256-GCM-SHA384	-305.60	217.21	74.80	41.89
DHE-RSA-AES128-GCM-SHA256	105.31	179.84	63.73	42.78
serverlimit-mid	-3.62	0.00	0.00	32.26
DHE-RSA-AES256-GCM-SHA384	105.87	28.16	-49.05	47.92
clientcache	66.52	-1540.15	-1350.48	-1119.41

Table 12: Coefficients produced by the regression analysis for different servers; sorted by worker column.

It can be resumed that the different engines of the Apache family show a similar effects for the different features, while Nginx shows very different effects.

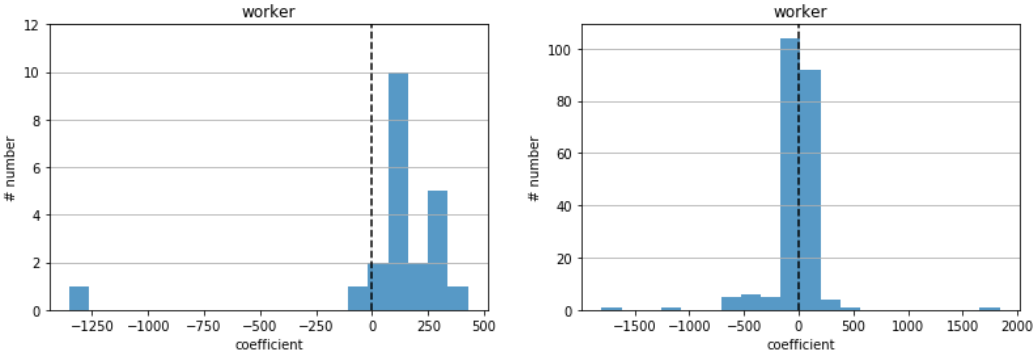
RQ 1.2 *Are there any interactions between features?*
When two or more features are active within the same configuration they might cause effects, that can not be explained by the features individually. Those unexplained effects are called interactions, since they arise out of the interaction of features. For this study, pair-wise interactions, that is, effects occurring when two features are interacting with each other, were incorporated into the linear model. Therefore, the feature-activation matrix needs to be extended by additional, *virtual* features. Those are active only when their corresponding features are simultaneously activated. This results in up to 225 coefficients of the models incorporating pair-wise interactions. An overview can be found in appendix B. Roughly, it can be stated that interactions contribute large effects to the model. Interestingly, the feature `serverlimit-mid`, that does not have any effect in the interaction-free model for the Apache event and worker, has the largest effect in the model with interactions, in order of 10^{15}, for the two engines. The same effect can be observed for the `serverlimit-high` feature in case of the Apache prefork. In contrast to the Apache family, the coefficients of Nginx do not change by that order of magnitude.

RQ 1.3 *What is the distribution of relevant and irrelevant features with respect to energy consumption?*
The histograms in the Figures 8 – 11 show the distributions of the features' effects on energy consumption for each webserver engine. The distributions affirm the observations made while answering *RQ 1.1* and *RQ 1.2*. The majority of features do have a minor or no influence at all.



(a) without interactions (b) with interactions

Figure 8: Distribution of effect sizes of Apache event. 4 outliers are removed from (b) (max: $4.4 \cdot 10^{15}$)

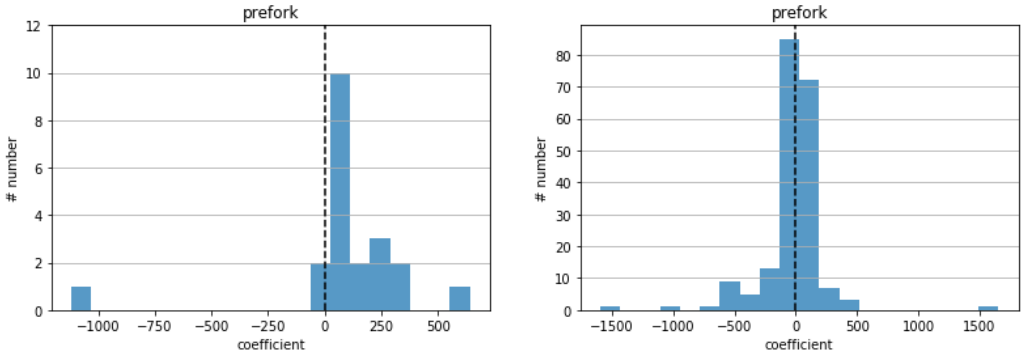


(a) without interactions (b) with interactions

Figure 9: Distribution of effect sizes of Apache worker. 5 outliers are removed from (b) (max: $3.9 \cdot 10^{15}$)

Without regard to any interactions, the Apache family (Figures 8 – 10) shares, as already mentioned, one energy saving feature, the `clientcache`, that has a large effect. The energy consuming features outnumber the energy saving ones, but their individual effects are smaller.

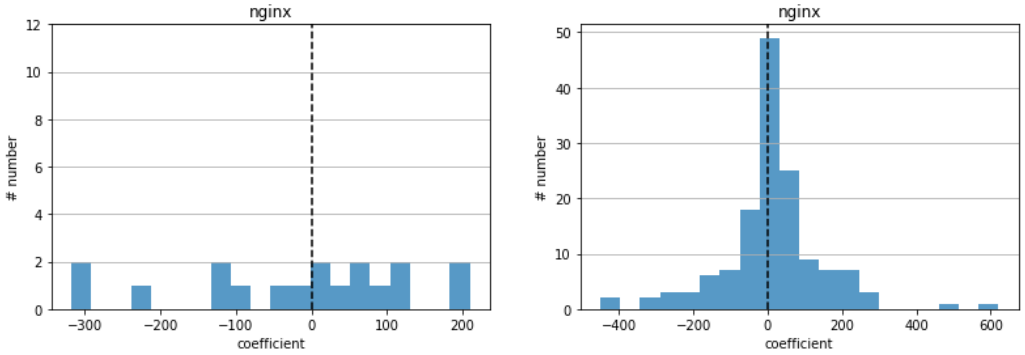
Incorporating interactions increases the range of effect sizes by magnitudes. As already discussed that does affect not only the effects of interactions, but the effects of individual features, as well. Beside that, the majority of effects are still close or equal to zero, such that they are irrelevant.



(a) without interactions (b) with interactions

Figure 10: Distribution of effect sizes of Apache prefork. 5 outliers are removed from (b) (max: $7.5 \cdot 10^{15}$)

In contrast, Nginx (Figure 11) shows an even distribution of small effects. Interestingly, incorporating interactions leads to a normal-distributed characteristic. Although the range of the effect size for Nginx does increase as well, the increase is rather moderate in comparison to the Apache family.



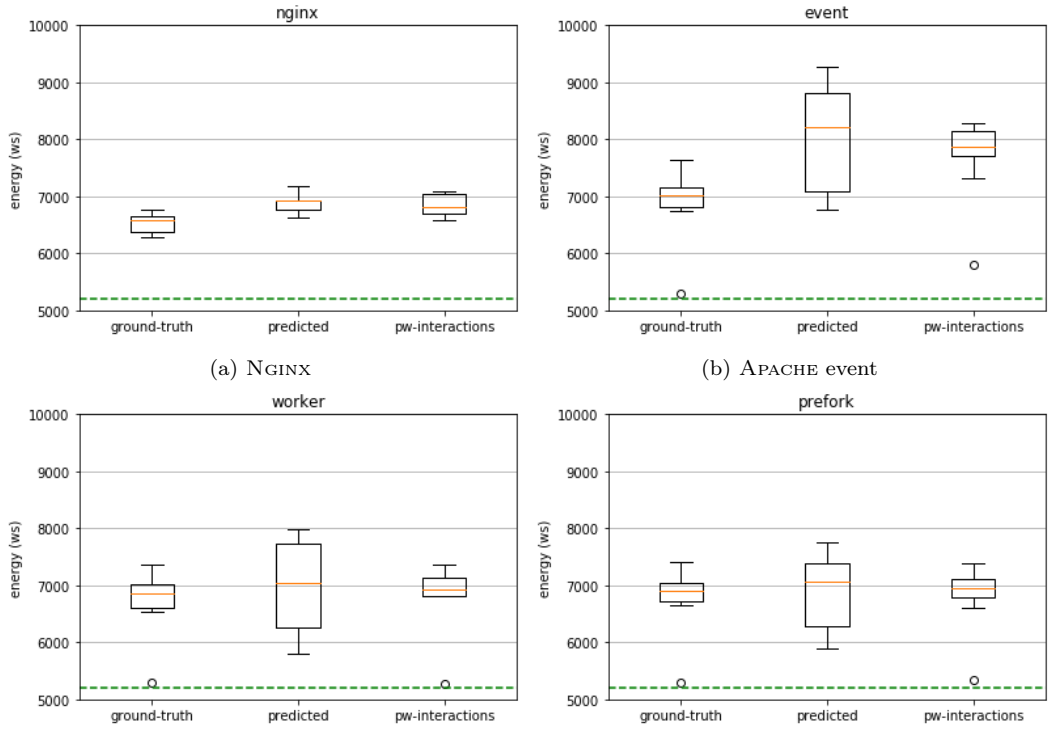
(a) without interactions (b) with interactions

Figure 11: Distribution of effect sizes of Nginx. no outliers are removed from (b)

Discussion So far, it could be shown that features have an influence on the energy consumption of webservers. The majority have a minor or no effect. But, some certainly have an impact. Furthermore, there do exist interactions

between features. Incorporating them into the linear model causes serious changes in the effect sizes by order of magnitudes.

In order to judge the quality of the two regression models, they were applied to predict the energy consumption of configurations for the test data set described in Table 7. The boxplots in Figure 12 compare the predicted values of the two models to the measured ground-truth. The dashed line marks the idle energy consumption of 5200 Ws by the systems, which is calculated by an idle power of 20 watts over an interval of 4 minutes and 20 seconds due to the cut-off of the first 40 seconds. The intercepts of models are provided in Table 13.



(a) Nginx

(b) Apache event

(c) Apache worker

(d) Apache prefork

Figure 12: Comparison of the predicted energy variances for the different server engines' models.

While Nginx roughly maintains its variance in the predictions, for the Apache family, a substantial increase of variance can be observed for the predictions

that do not include interactions. Including the interactions reduces the variance to the order of the ground-truth. This is a first indicator that the models that include the interactions are able to predict the energy consumption better. As a second indicator, the mean absolute errors provided in Table 13 can be consulted. Analogous to the variance the error decreases across all webserver engines, when interactions are taken into account.

Another observation concerns the deviation of the median in comparison to the ground-truth. It can be seen as the bias of the model. The models for Apache worker and prefork keep the median stable. While Nginx shows a slightly positive shift of the median, the Apache event shift represents a drastic bias. Those biases become reflected in the interceptions provided in Table 13.

feature	nginx	event	worker	prefork
— model without feature interactions —				
intercept (Ws)	6915.298	7545.995	6772.361	6712.407
mean absolute error (Ws)	354.20	1188.70	718.13	574.36
mean squared error	167066.13	1910317.64	551196.36	376160.21
R^2-Score for training set	0.72560	0.74568	0.74334	0.6351
R^2-Score for testing set	-5.00885	-4.31699	-0.93295	-0.26174
— model with feature interactions —				
intercept (Ws)	7001.894	7983.689	7039.996	6800.644
mean absolute error (Ws)	310.43	784.36	235.54	207.31
mean squared error	103777.92	738269.88	79131.51	70308.39
R^2-Score for training set	0.99598	0.99355	0.99827	0.99883
R^2-Score for testing set	-2.73257	-1.05483	0.72250	0.76417

Table 13: Quality metrics of the learned models for the different servers.

Finally, the standard quality metric R^2-Score can be consulted. It describes the portion of variance in the data that can be explained by the model. Its value range reaches from a complete explanation with 1.0, and can become arbitrarily bad below 0.0.

Table 13 provides the R^2-Score for all the models, not only with respect to the testing data set, but with respect to the training data set, as well. The R^2-Scores of the models for the training set confirm that the models incorporating the interactions are substantially better in fitting the variance of the observed data. The R^2-Scores of the models without any interactions applied to the test-data set are all below 0.0, but for NGINX and APACHE event they are even below -4.0. Including the interactions into the models, the R^2-Scores of NGINX and APACHE event keep values below -1.0, while APACHE worker and prefork show scores above 0.7.

To resume, it can be said, that the models incorporating feature interactions provide substantially better predictions than the models ignoring the interactions. The models learned for APACHE worker and prefork do not fit any better to the training data, but they do provide better predictions for the testing data. In contrast NGINX and APACHE event do not provide optimal predictions. In case of the APACHE event that could be accounted to a bias that tends to predict higher energy consumptions. In case of NGINX, it is difficult to pin-point what causes the low R^2-Score, since the variances of the predictions do not diverge that much; the bias is comparatively low and the absolute error is the lowest of all the models. Maybe, it must be accounted to a scaling effect, since the variance of the coefficients of the NGINX model is substantially smaller compared to the other models. Therefore, the energy consumption of NGINX is less influenced by features, than it is the case for the APACHE family.

6.2.2. Performance

If it could be shown, that performance correlates with energy consumption, life would be easy, since optimization for minimal response times is a common goal among webserver administrators already. To investigate whether that is the case, the correlation between the average response times of all training-set experiments and their averaged energy consumption is calculated. The calculation of the correlation coefficient according to Pearson requires the two variables to be normally distributed. Anderson-Darling-, as well as Shapiro-Wilk-test indicate that this is not the case. Hence, the Spearman's rank correlation remained as an appropriate tool to verify whether energy consumption and response time correlate.

feature	nginx	event	worker	prefork
Spearman's coefficent	0.04794	-0.32137	-0.31715	-0.41724
p-Value	0.51925	$2.72519 \cdot 10^{-7}$	$3.97258 \cdot 10^{-7}$	$1.11852 \cdot 10^{-10}$

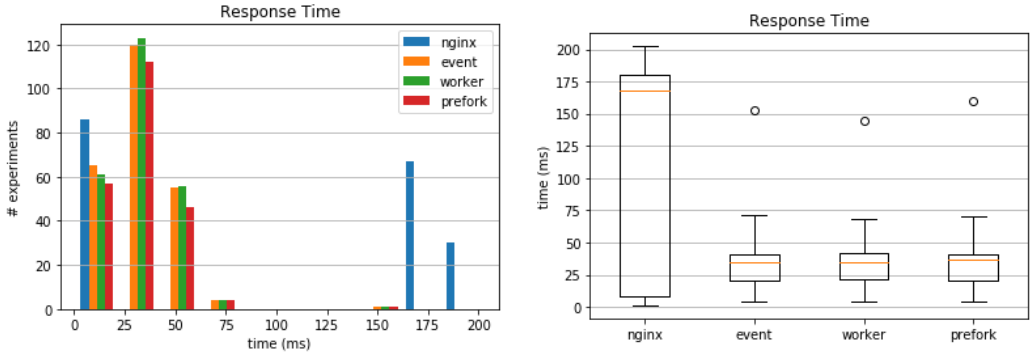
Table 14: Correlations between energy consumption and response time for the different servers.

Table 14 provides the correlation coefficients calculated with Spearman's rank correlation for the different webserver engines, accompanied by the p-values for the hypothesis test, whose null hypothesis is that the two variables are uncorrelated.

Discussion For NGINX a correlation can be rejected. The p-value indicates that the null hypothesis can not be rejected and therefore the variables are uncorrelated. Beside that, the correlation coefficient is only marginal.
For the APACHE family things look different. The p-values indicate that the null hypothesis can clearly be rejected. The correlations are weak, but they

are significant. Furthermore, all the correlation coefficients for the APACHE family are negative. That means that one of the variables increases, while the other decreases. Hence, it is not the case that a webserver's optimization for performance does lead to energy efficiency, but it appears to be a trade-off between the two. Longer response times mean less energy consumption and vice versa.

It remains questionable whether response time alone represents a sufficient metric for a webserver's performance. Other possible metrics were already mentioned in Chapter 2.4. They all do influence each other. The response time is influenced by the data rate, since it takes more time to transfer the same amount of data with a slower connection. The transfer of more data, while keeping the connection speed constant, will result in a drop of the response time. The number of responses received in an interval of time can be converted to the response time per request by the division of time through response count.



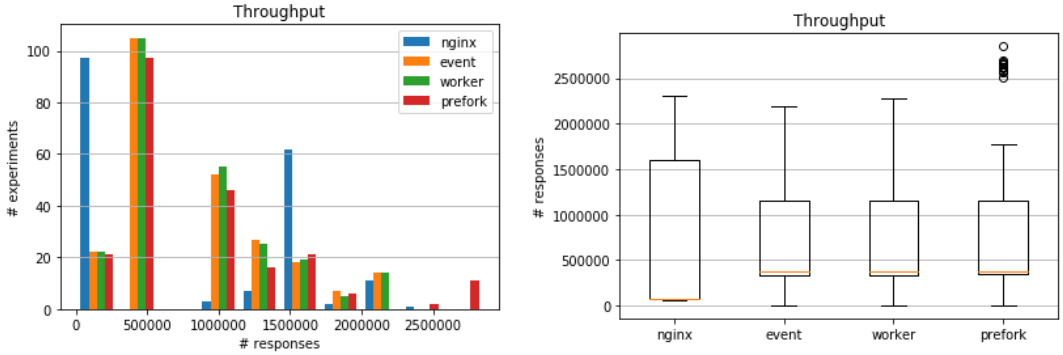
(a) Distributions

(b) Boxplots

Figure 13: Visual descriptions of the response times observed during all the experiments.

This relation between the performance metrics is visualized by Figures 13 – 15. For all servers, the median in the boxplots for both of the throughputs (Figures 14 and 15) has a tension downwards, whereas the response time (figure 13) has

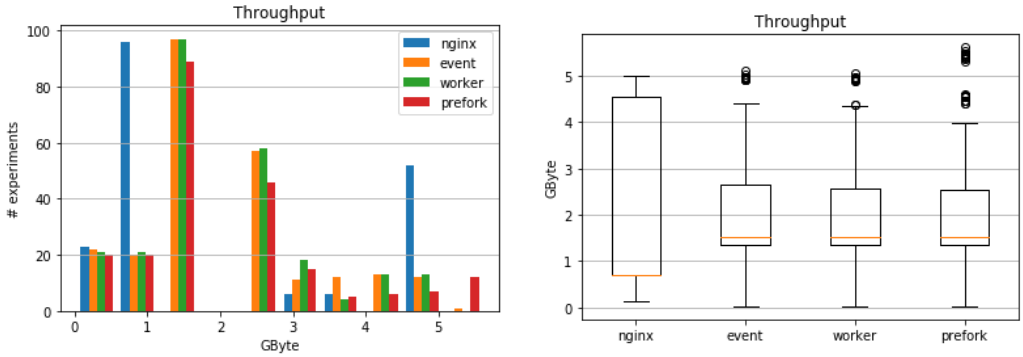
a tension upwards. That gives a hint to the existence of correlations between the different performance metrics and henceforth, they can be interchanged by each other. But, evaluating that remains a future task.



(a) Distributions

(b) Boxplots

Figure 14: Visual descriptions of the throughput of responses observed during all the experiments.



(a) Distributions

(b) Boxplots

Figure 15: Visual descriptions of the throughput of data observed during all the experiments.

All performance measures presented in Figures 13 – 15 do not reflect failures, which were shown in Figure 7. But, a throughput measurement, as a count of responses, can change its meaning substantially, when set in context to failures. For instance: According to Figure 7, there do not exist any failures for NGINX. However, bearing in mind that, while reviewing the throughput of responses in Figure 14, it is conspicuous that NGINX is the only webserver that

has a median close to zero, meaning, that a substantial number of requests was not served at all. For an unknown reason, those unserved requests were not counted as failures. All figures indicate a substantially different behaviour of NGINX in comparison to the APACHE family.

6.2.3. Workloads

Finally, to gain some intuition about the workloads' influence on the energy consumption of features (RQ 3), it was analysed whether the most energy efficient configurations maintain their rank for varying workloads (RQ 3.1). Therefore, Tables 15 – 18 provide the top-10 configurations of the webserver engines, with respect to the energy consumption under workload *P7k*, as described in Chapter 4.5.2. The other columns of the table show the ranks of the configurations under different workloads, including a cumulative column *(all)*, which represents the ranks for the entire experiment.

configuration	all	R1k	P1k	P7k	R7k	R3M
worker-16-64-00010000100-off	6	2	2	1	1	2
worker-16-64-00010010000-off	4	1	2	1	1	2
worker-16-64-00010001000-off	5	1	3	1	1	2
worker-16-64-00010000010-off	7	1	4	1	1	3
worker-16-64-00010000000-off	11	1	7	1	3	6
worker-16-64-10010000000-off	10	1	9	2	2	2
worker-16-128-00010000000-off	9	2	6	3	7	2
worker-16-64-00010000000-off	2	3	7	4	1	1
worker-16-64-00010000001-off	1	1	8	5	1	2
worker-16-64-01010000000-off	8	6	4	6	1	2

Table 15: Top-10 APACHE Worker configurations aligned to the P7K load and their respective ranks for other loads. The minimum among the maximum ranks is 226 for the R3M load.

The minimum among the maximal ranks is provided in the caption of the tables. That should provide an intuition about the actual variance of the ranks within the top-10.

configuration	all	R1k	P1k	P7k	R7k	R3M
prefork-256–00010010000-off	10	4	1	1	1	4
prefork-256–01010000000-off	3	5	5	1	1	7
prefork-256–00010000001-off	7	3	3	1	1	7
prefork-256–00010000000-off	8	5	7	1	1	5
prefork-256–10010000000-off	12	5	11	1	1	7
prefork-256–00010000010-off	2	1	2	1	1	7
prefork-256–00010001000-off	5	2	4	1	1	1
prefork-256–00010000100-off	6	5	6	1	1	6
prefork-512–00010000000-off	4	7	9	1	2	4
prefork-256–00010000000-off	1	5	8	1	3	7

Table 16: Top-10 Apache Prefork configurations aligned to the P7K load and their respective ranks for other loads. The minimum among the maximum ranks is 205 for the R3M load.

configuration	all	R1k	P1k	P7k	R7k	R3M
event-16-64-00100000100-off	22	22	18	1	22	12
event-16-128-00010000000-off	3	10	9	2	10	5
event-16-64-00010000100-off	8	7	6	3	4	1
event-16-64-00010000001-off	1	5	3	4	3	3
event-32-64-00010000000-off	9	6	7	5	6	1
event-16-64-00110000000-off	10	8	10	6	2	2
event-16-64-00010000010-off	6	4	4	7	5	3
event-16-64-01010000000-off	2	9	8	8	9	2
event-16-64-00010010000-off	7	2	5	9	7	1
event-16-64-00010001000-off	4	1	2	10	1	1

Table 17: Top-10 Apache Event configurations aligned to the P7K load and their respective ranks for other loads. The minimum among the maximum ranks is 232 for the R3M load.

Discussion The values confirm previous observations. The Apache engines worker and prefork expose a relatively stable and predictable behaviour. That still holds under varying workloads. As the Tables 15 and 16 show, the ranks throughout the top-10 configurations varies only slightly. That suggests that varying workload does not have a relevant influence on the energy efficiency of features. Although the ranks vary more in case of the Apache event (figure 17), that claim appears still to be valid.

However, the variations of the ranks for the NGINX webserver challenge this observation. As Figure 18 shows, the ranks for the *R1k* load jumps from top-10 to worse-10, while the rank for the *R3M* load varies in the upper middle. The data appears to be arbitrary and it can neither be concluded that the energy efficiency of NGINX is influenced by the workload nor by configurations. The data obtained through the experiments does not reveal any observable pattern for the NGINX webserver.

configuration	all	R1k	P1k	P7k	R7k	R3M
nginx-1024–10001000000-off	1	9	1	1	1	33
nginx-1024–10000010000-off	2	72	54	2	2	32
nginx-1024–10000000000-off	71	142	158	3	10	73
nginx-1024–10000000000-ECDHE-RSA-AES128-GCM-SHA256	4	58	56	4	6	82
nginx-1024–10000000000-off	70	141	141	5	12	50
nginx-4096–10000000000-off	63	140	83	6	11	53
nginx-2048–11101111000-DHE-RSA-AES256-GCM-SHA384	38	57	59	7	4	112
nginx-2048–10000000000-ECDHE-RSA-AES128-GCM-SHA256	6	55	55	8	5	91
nginx-2048–10000000000-off	74	174	135	9	9	29
nginx-1024–10000000000-DHE-RSA-AES128-GCM-SHA256	18	59	58	10	7	88

Table 18: Top-10 NGINX configurations aligned to the P7K load and their respective ranks for other loads. The minimum among the maximum ranks is 178 for the R3M load.

To resume, it can be stated that the influence of features on the energy consumption of the webservers of APACHE family keeps stable under varying workloads. For NGINX, this is not the case, but it remains unclear why.

7. Validity

Internal Validity Most effort was spent in maximizing internal validity. The extensive prestudy aimed to discover and eliminate limitations drawn by the operating system, the network, or the load-testing tool. There are still serious threads to validity, including the trustworthiness of the load-testing tool, possible imprecision of energy measurement, and the unknown influences and arbitrary behaviour of the underlying operating system, the cluster management software, and the network. Nevertheless, the prestudy showed that the webservers' behaviour in principle is reproducible, the possibility of failure is always present, especially in distributed systems.

A serious problem arises when comparing performance measurements of different configuration runs. Throughput is used as the common metric for performance, either in the meaning of data rate or request rate. It is unclear whether they can be used alternatively. The investigations done throughout this study suggest that they are possibly correlated. However, an deeper evaluation is needed, including varying workloads, to draw a sound conclusion. Furthermore, both of the throughput metrics lack information about the rate of failure. A combined metric that incorporates the different throughput metrics, as well as the failure rate could not be found. Therefore, all three metrics were presented separately in the evaluation (Chapter 6). Developing such a combined metric could be suspect for future research.

External Validity This study does not aim for external validity. Nevertheless, the software used are widespread standard systems. This applies to Ubuntu as the operating system, and the subjected webservers Apache and Nginx, alike. The features that were selected are well-known and part of many

real-world configurations. The HTML files utilized as workloads contained realistic contents. The fact that the webservers did only serve static files is no thread to external validity, since the serving of so called assets is a common task.

However, the load-testing plan designed and applied for this study is optimized to enforce the consumption of energy. Therefore, it does not emulate realistic traffic shaping, but mimics a moderate denial of service attack. Moreover, since the load-testing tool is the only source of requests, the study must be accepted as a lab experiment, which provides external validity only by accident.

Moreover, since all experiments were run on homogeneous hardware and operating system, the results can not be generalized for arbitrary setups by any means.

8. Conclusion

This thesis aimed at exploring the influences of features on the energy consumption in case of webservers. The resulting models were expected to support the users of those systems in discovering energy-saving potentials in their system's configuration. Henceforth, a feature-oriented perspective was taken onto four webserver engines as the study's subjects.

Do Features Matter for Energy Consumption? Yes, they do. At least in case of webservers. Roughly, that distils the conclusion of this thesis and thereby contradicts Hinchliffe's rule [26].

It could be shown, that some features have a relevant influence on the energy consumption of webservers and that incorporating interactions between them leads to substantially more precise predictions. That holds for all webservers that were subject to this thesis.
A weak, negative correlation between performance and energy consumption could be discovered only for the Apache family. Thus, it is likely, that performance must be sacrificed for energy efficiency. Moreover, the features' influences on the energy consumption of the Apache family appears not to be affected by varying workloads. The data gathered refused to reveal any patterns for Nginx. Neither, in case of influences of varying workloads, nor a significant correlation between performance and energy consumption.

The behaviour of Nginx is influenced by factors, which could not be identified during this study. Nevertheless, it also grants two insights:
First, a black-box approach is insufficient to understand or even model the behaviour of integrated systems, such as webservers, that are indistinguishable from their environment. The black-box has already been opened by adjusting operating system properties. Gathering reliable data for Nginx, requires further unfolding of the black-box.

With regard to empiric black-box studies, the reader should be reminded, that correlation does not mean causation. Tyler Vigen [27] collected great examples of *Spurious Correlations*, visualizing, what Benno Stein formulated during an interview: "Correlations are everywhere, where searched for them."[28]
Second, the largest differences of the obtained models are in-between the systems, that do not share any code. Although, the Apache family differs in their *multi processing models*, they share most of their code-base, which appears to guarantee comparable behaviour. Therefore, the most relevant choice a user has, is the choice of the system. Recommendations, which system to use, to achieve energy-efficiency for certain combinations of features, is a possible outcome of future research.

Furthermore, the second insight suggests a possible design flaw in the feature models. The three MPMs of the Apache family and Nginx were treated as four independent *software product lines* (SPLs) with disjoint feature models. That allowed for comparative discussions of the different processing approaches, but it also dissembled the actual SPL characteristic of the Apache family. Neglecting the different MPMs as a feature of a joint feature model, means rejecting a feature, that offers alternative choices.
This observation becomes relevant, when recurring to the goal of discovering energy-efficient configurations for the SPLs through application of the gained models. The assumption about the benefit gained through the models ignores the nature of most features. By definition, features provide an increment of functionality [14]. Hence, they do not offer alternative choices. Beside the discarded MPMs, the only alternative choice throughout the feature space of this study are the values for the process- and thread-counts. All other features are binary choices, that can be switched off or on. If a feature like encryption is required for a setup, it will be activated, no matter what its impact on energy consumption is. It has always been best practice among administrators to leave unneeded features deactivated. The only energy saving feature

revealed during the study is active by default for all servers. That underlines, the appeal for reverting the *configuration space explosion* [13]. Following good old engineering practice:

> "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."[3]

The reduction of the configuration space is the best way to prevent misconfiguration that may result in unnecessary energy consumption. However, this is the duty of the developers. It can not be the burden of the administrators or the users. As it is the responsibility of the developers to design software systems with energy efficiency in mind. Hence, developers must be equipped with the knowledge needed for that challenge [12].
It should not be withheld that effective and reliable energy saving opportunities have already been discovered in the major upfront energy consumption of current hardware. Meanwhile, administrators can exploit them through consolidation of server systems [7].

Consolidation of installations, reduction of configuration options, and application of energy efficient algorithms are promising approaches for saving energy. Yet, all these are aspects, that need to be addressed by design.

Henceforth, the author questions the worthwhile applicability of the feature-oriented approach to support users in optimizing their webserver's configuration with regard to energy efficiency. Even though, it might be a useful tool to help developers identifying energy-wasting features in their product lines.

Computer systems are man-made. They did not evolve from nature. Their design, implementation, and behaviour must be comprehensible by humans. Delegating energy efficiency to configurations, recommended by empirically derived correlation models, means surrendering to complexity.

[3] Antoine de Saint-Exupéry

8.1. Future Work

The time spent for a thesis is limited. Unresolved issues remain, new questions arise. Most irritating are the results gained for Nginx. It remains unclear what causes the arbitrary behaviour observed. An independent reproduction of the study is advised. That may include:

- Extending the field of subject systems.
- Treating different engines, such as the Apache MPMs, as features.
- Including parameters of the operating system into the feature space.

But, it is possible, that the reason can only be discovered by inspection of the source code of Nginx. Another suspicious component in the study is the load testing tool. It is strongly recommended to conduct an evaluation of existing load testing tools. In case they all reveal unreliability, a new one must be developed. However, there are chances, that the extension of an existing tool might be sufficient.

The observations in Chapter 6.2.2 suggest that different performance metrics of webservers might correlate with each other. Further investigation could lead to a more reliable performance metric that incorporates failures and the different flavours of throughput.

References

[1] John Cook, Naomi Oreskes, Peter T Doran, William RL Anderegg, Bart Verheggen, Ed W Maibach, J Stuart Carlton, Stephan Lewandowsky, Andrew G Skuce, Sarah A Green, et al. Consensus on consensus: a synthesis of consensus estimates on human-caused global warming. *Environmental Research Letters*, 11(4):048002, 2016.

[2] Ralph Hintemann. Energy consumption of data centers continues to increase–2015 update. *Borderstep Institut, Berlin*, 2015.

[3] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. United states data center energy usage report. *Berkeley Lab*, June 2016.

[4] Peter Bright. Epic uptime achievement unlocked. Can you beat 16 years? `https://arstechnica.com/information-technology/2013/03/epic-uptime-achievement-can-you-beat-16-years/`, March 2013.

[5] Lauro Beltrao Costa, Samer Al-Kiswany, Raquel Vigolvino Lopes, and Matei Ripeanu. Assessing data deduplication trade-offs from an energy and performance perspective. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–6. IEEE, 2011.

[6] Boliang Feng, Jiaheng Lu, Yongluan Zhou, and Nan Yang. Energy efficiency for mapreduce workloads: An in-depth study. In *Proceedings of the Twenty-Third Australasian Database Conference-Volume 124*, pages 61–70. Australian Computer Society, Inc., 2012.

[7] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A Shah. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010*

ACM SIGMOD International Conference on Management of data, pages 231–242. ACM, 2010.

[8] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Exploring power-performance tradeoffs in database systems. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 485–496. IEEE, 2010.

[9] Hao Yang. *Energy prediction for I/O intensive workflow applications*. PhD thesis, University of British Columbia, 2014.

[10] Frederico G Alvares de Oliveira Jr, Thomas Ledoux, et al. Self-optimisation of the energy footprint in service-oriented architectures. In *Proceedings of the 1st Workshop on Green Computing*, pages 4–9, 2010.

[11] Suparna Bhattacharya, Karthick Rajamani, Kanchi Gopinath, and Manish Gupta. Does lean imply green?: a study of the power performance implications of java runtime bloat. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 259–270. ACM, 2012.

[12] Candy Pang, Abram Hindle, Bram Adams, and Ahmed E. Hassan. What do programmers know about software energy consumption? *IEEE Softw.*, 33(3):83–89, May 2016.

[13] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. Hey, you have given me too many knobs!: understanding and dealing with over-designed configuration in system software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 307–319. ACM, 2015.

[14] Don Batory, Jacob Neal Sarvela, and Axel Rauschmayer. Scaling stepwise refinement. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 187–197, Washington, DC, USA, 2003. IEEE Computer Society.

[15] Norbert Siegmund, Sergiy S. Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. Predicting performance via automated feature-interaction detection. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 167–177, Piscataway, NJ, USA, 2012. IEEE Press.

[16] Netcraft. Web Server Survey. `https://news.netcraft.com/archives/category/web-server-survey/`, March 2017. Last visited 2017-04-20.

[17] Wikipedia. Quick UDP Internet Connections. `https://en.wikipedia.org/wiki/QUIC`. Last visited 2017-05-29.

[18] Don Batory. Feature models, grammars, and propositional formulas. In *International Conference on Software Product Lines*, pages 7–20. Springer, 2005.

[19] Norbert Siegmund. *Measuring and predicting non-functional properties of customizable programs*. PhD thesis, Magdeburg, Universität, Diss., 2012, November 2012.

[20] Jonas Eckhardt, Andreas Vogelsang, and Daniel Méndez Fernández. Are non-functional requirements really non-functional?: an investigation of non-functional requirements in practice. In *Proceedings of the 38th International Conference on Software Engineering*, pages 832–842. ACM, 2016.

[21] Neil Ernst. There is no such thing as a non-functional requirement. `http://www.neilernst.net/there-is-no-such-thing-as-a-non-functional-requirement/`, March 2009. Last visited 2017-05-27.

[22] J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, and Dennis Rea. Performance Testing Guidance for Web Applications. `https://`

msdn.microsoft.com/en-us/library/bb924356.aspx, September 2007. Last visited 2017-06-04.

[23] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya, et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers*, 82(2):47–111, 2011.

[24] Frank Leferink, Cees Keyer, and Anton Melentjev. Static energy meter errors caused by conducted electromagnetic interference. *IEEE Electromagnetic Compatibility Magazine*, 5(4):49–55, 2017.

[25] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. Performance-influence models for highly configurable systems. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 284–294. ACM, 2015.

[26] Stuart Merrill Shieber. Is this article consistent with hinchliffe's rule? *Annals of Improbable Research*, 21(3), 2015.

[27] Tyler Vigen. Spurious correlations. http://www.tylervigen.com/spurious-correlations, May 2015.

[28] Petra Löffler and Benno Stein. Korrelationen sind überall da, wo sie gesucht werden. *Zeitschrift für Medienwissenschaft*, 10:91–96, April 2014.

A. Featuremodels

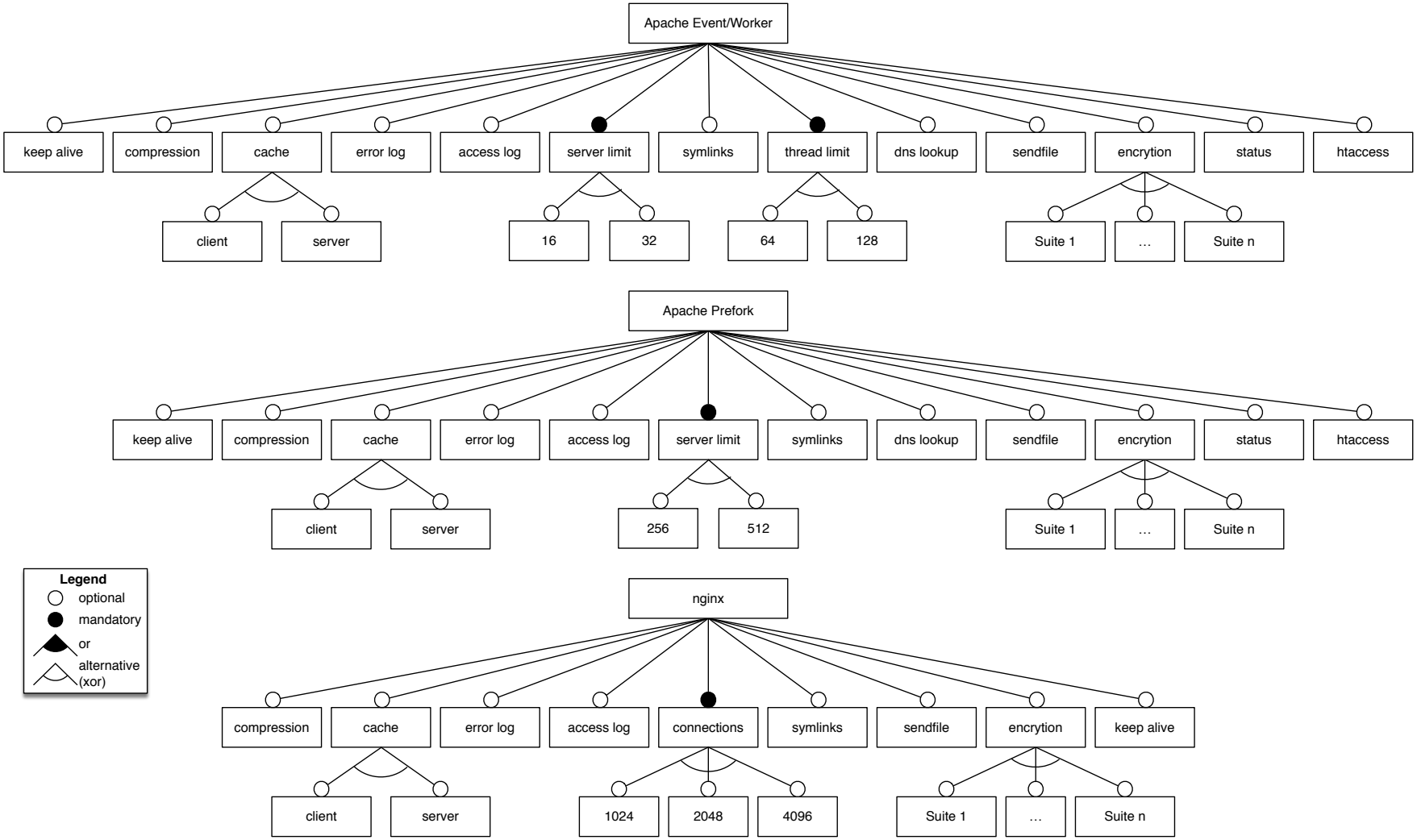


Figure 16: Assembled featuremodels of all four server. Apache Event and Worker share the same featuremodel.

B. Linear Model with Interactions

	nginx	event	worker	prefork
compression # serverlimit-mid	-2.56245	-1.459667e+03	1.157085e+05	89.6569
clientcache # compression	32.0502	2.099779e+03	1.839735e+03	1656.55
servercache	-234.79	4.625029e+02	4.889935e+02	517.252
clientcache # DHE-RSA-AES256-GCM-SHA384	-97.4911	3.597302e+02	3.699444e+02	256.782
clientcache # DHE-RSA-AES128-GCM-SHA256	-50.9162	3.137247e+02	3.274008e+02	271.233
ECDHE-RSA-AES256-SHA384	-303.134	2.495445e+02	2.649878e+02	384.83
accesslog # compression	-53.4176	2.423333e+02	2.390117e+02	222.404
compression # dns	—	2.023688e+02	1.908327e+02	178.802
accesslog # servercache	18.3005	3.480004e+02	1.873309e+02	226.081
accesslog # keepalive	123.807	2.644168e+02	1.730121e+02	158.442
ECDHE-RSA-AES128-SHA256	-334.69	1.681484e+02	1.716086e+02	278.169
dns # sendfile	—	3.007256e+02	1.697083e+02	137.582
DHE-RSA-AES128-SHA256	180.851	1.524814e+02	1.579691e+02	294.783
clientcache # DHE-RSA-AES128-SHA256	-185.643	2.029456e+02	1.563840e+02	100.664
DHE-RSA-AES256-SHA256	165.408	1.594974e+02	1.545319e+02	273.575
clientcache # DHE-RSA-AES256-SHA256	-158.789	1.976267e+02	1.534889e+02	123.822
clientcache # dns	—	2.146530e+02	1.528706e+02	179.253
accesslog # ECDHE-RSA-AES256-SHA384	-53.6587	1.896786e+02	1.522507e+02	129.966
dns # servercache	—	2.229613e+02	1.484402e+02	181.384
accesslog # DHE-RSA-AES256-GCM-SHA384	-43.4355	1.498136e+02	1.426670e+02	167.87
accesslog # ECDHE-RSA-AES128-SHA256	-27.9413	1.773838e+02	1.386842e+02	151.738
dns # symlinks	—	3.117709e+01	1.358512e+02	68.7464
dns # DHE-RSA-AES256-GCM-SHA384	—	1.330858e+02	1.347687e+02	106.823
accesslog # serverlimit-high	-4.40019	1.181244e+02	1.317654e+02	4.76007e+14
accesslog # clientcache	41.1294	2.427343e+02	1.301026e+02	170.063
accesslog # ECDHE-RSA-AES128-GCM-SHA256	39.986	1.442433e+02	1.269256e+02	158.593
dns # ECDHE-RSA-AES256-SHA384	—	8.726668e+01	1.226136e+02	90.6855
dns # DHE-RSA-AES128-SHA256	—	1.190050e+02	1.205177e+02	100.443
keepalive	-239.377	3.703700e+02	1.202826e+02	397.278

Continued on next page

	nginx	event	worker	prefork
servercache # serverlimit-high	3.82986	1.082892e+02	1.192213e+02	8.26087e-14
dns # ECDHE-RSA-AES128-SHA256	—	1.612552e+02	1.172499e+02	106.582
dns # serverlimit-high	—	2.206717e+02	1.155946e+02	5.51055e-14
accesslog # ECDHE-RSA-AES256-GCM-SHA384	6.39481	1.208304e+02	1.130962e+02	146.659
dns # threadlimit-high	—	8.861779e+01	1.130807e+02	—
accesslog # sendfile	28.1787	3.821288e+01	1.082469e+02	162.296
keepalive # ECDHE-RSA-AES256-SHA384	621.057	1.024647e+02	1.048195e+02	-35.7676
servercache # threadlimit-high	—	1.005540e+02	1.047014e+02	—
dns # ECDHE-RSA-AES128-GCM-SHA256	—	1.414675e+02	1.027441e+02	129.687
accesslog # symlinks	-13.2179	1.463973e+02	9.924990e+01	63.0144
dns # keepalive	—	2.345979e+02	9.823813e+01	108.709
dns # htaccess	—	4.563079e+01	9.668516e+01	85.5605
sendfile	102.743	1.294610e+02	9.391275e+01	148.631
compression # DHE-RSA-AES256-GCM-SHA384	47.1623	2.631419e+01	9.316779e+01	81.131
dns # status	—	7.688249e+01	9.266163e+01	57.411
dns # DHE-RSA-AES256-SHA256	—	1.120024e+02	9.079037e+01	85.441
accesslog # DHE-RSA-AES128-SHA256	-42.7749	1.494149e+02	8.998528e+01	98.4826
accesslog # DHE-RSA-AES256-SHA256	-6.42827	1.664112e+02	8.936871e+01	154.239
compression # ECDHE-RSA-AES128-SHA256	21.6165	1.390095e+02	8.732008e+01	129.637
compression # keepalive	56.6982	1.919293e+02	8.185889e+01	138.955
accesslog # errorlog	-13.6611	1.257836e+02	7.250083e+01	48.2489
compression # ECDHE-RSA-AES128-GCM-SHA256	79.9145	1.267733e+02	7.088634e+01	105.33
errorlog # servercache	9.30591	1.037109e+02	6.955143e+01	55.3848
dns # DHE-RSA-AES128-GCM-SHA256	—	6.997847e+01	6.897893e+01	66.287
compression # ECDHE-RSA-AES256-SHA384	10.9421	1.079147e+02	6.756285e+01	91.4818
accesslog # DHE-RSA-AES128-GCM-SHA256	-21.4156	9.134767e+01	6.651651e+01	133.304
servercache # status	—	1.197247e+02	6.632618e+01	121.991
compression # DHE-RSA-AES256-SHA256	-19.9704	1.233239e+02	6.413212e+01	127.118
accesslog # htaccess	—	1.164401e+02	6.308946e+01	110.642
servercache # symlinks	153.042	7.034093e+01	6.105917e+01	72.5093

Continued on next page

	nginx	event	worker	prefork
clientcache # ECDHE-RSA-AES256-GCM-SHA384	-79.9058	3.081862e+01	6.068986e+01	12.8116
dns # ECDHE-RSA-AES256-GCM-SHA384	—	1.439561e+02	5.834715e+01	113.322
accesslog # threadlimit-high	—	1.756059e+02	5.825196e+01	—
clientcache # threadlimit-high	—	7.103667e+01	5.789986e+01	—
sendfile # threadlimit-high	—	1.276630e+02	5.738469e+01	—
clientcache # ECDHE-RSA-AES128-GCM-SHA256	-83.4729	3.701553e+01	5.331485e+01	33.4656
accesslog # status	—	1.222147e+02	5.291441e+01	99.6682
htaccess # servercache	—	8.740944e+01	5.181742e+01	90.8892
keepalive # ECDHE-RSA-AES128-SHA256	501.394	-7.789030e+00	5.168542e+01	-159.138
keepalive # status	—	4.957067e+01	5.146342e+01	47.8091
sendfile # symlinks	42.4892	6.566836e+01	4.972534e+01	-7.21347
compression # serverlimit-high	26.1176	1.361110e+02	4.935170e+01	-162745
status # DHE-RSA-AES256-GCM-SHA384	—	1.812384e+01	4.674880e+01	-10.0807
compression # DHE-RSA-AES128-SHA256	3.50293	9.541154e+01	4.493955e+01	51.4015
serverlimit-high # threadlimit-high	—	3.172882e+01	4.016240e+01	—
compression # ECDHE-RSA-AES256-GCM-SHA384	64.0626	1.431627e+02	3.998561e+01	113.062
htaccess # sendfile	—	1.087133e+02	3.972166e+01	46.5241
compression # threadlimit-high	—	4.671712e+01	3.556913e+01	—
clientcache # serverlimit-high	31.6586	9.510780e+01	3.553023e+01	-7.12153e+11
threadlimit-high # DHE-RSA-AES256-GCM-SHA384	—	5.587812e+01	3.354115e+01	—
compression # status	—	1.121847e+02	3.264758e+01	66.0436
errorlog # sendfile	13.176	2.998530e+01	3.251820e+01	5.04926
htaccess # ECDHE-RSA-AES128-SHA256	—	2.350779e+01	3.239018e+01	-11.3834
serverlimit-high # DHE-RSA-AES128-SHA256	-12.6328	7.339506e+01	2.896716e+01	1.70206e-14
serverlimit-high # ECDHE-RSA-AES128-SHA256	24.0604	7.009082e+01	2.844904e+01	1.23715e-13
threadlimit-high # DHE-RSA-AES256-SHA256	—	4.672917e+01	2.617398e+01	—
compression # DHE-RSA-AES128-GCM-SHA256	99.0522	1.175250e+02	2.610266e+01	113.741
keepalive # serverlimit-high	1.98514	-6.091146e+01	2.563063e+01	6.85704e-14
compression # htaccess	—	1.386955e+02	2.363291e+01	115.398
symlinks # threadlimit-high	—	2.735636e+01	2.294071e+01	—

Continued on next page

	nginx	event	worker	prefork
keepalive # threadlimit-high	—	-5.445575e+01	2.282902e+01	—
sendfile # serverlimit-high	48.2247	6.876335e+01	2.161989e+01	-1.73954e-14
htaccess # DHE-RSA-AES256-GCM-SHA384	—	-4.688715e+01	2.090256e+01	-28.0975
symlinks # DHE-RSA-AES256-GCM-SHA384	147.556	2.038815e+01	2.088801e+01	11.7023
errorlog # ECDHE-RSA-AES128-SHA256	21.8741	2.474454e+00	2.028405e+01	21.6273
errorlog # DHE-RSA-AES256-GCM-SHA384	41.9799	1.177166e+01	1.976761e+01	15.1333
compression # symlinks	-31.0371	5.736012e+01	1.960151e+01	24.5917
clientcache # errorlog	26.7866	3.472566e+01	1.906741e+01	44.5192
threadlimit-high # ECDHE-RSA-AES256-GCM-SHA384	—	2.144232e+01	1.880394e+01	—
dns # errorlog	—	1.204076e+02	1.836702e+01	121.438
compression # errorlog	-2.1503	4.681173e+01	1.777811e+01	-11.3442
clientcache # symlinks	57.2698	5.310756e+01	1.763022e+01	55.9658
sendfile # status	—	4.372861e+00	1.534358e+01	-83.477
status # ECDHE-RSA-AES256-GCM-SHA384	—	1.268832e+01	1.465557e+01	17.0966
serverlimit-high # ECDHE-RSA-AES256-GCM-SHA384	46.175	7.731579e+01	1.456592e+01	8.20955e-14
threadlimit-high # ECDHE-RSA-AES128-SHA256	—	2.100649e+01	1.423686e+01	—
threadlimit-high # DHE-RSA-AES128-SHA256	—	-9.588020e+00	1.382023e+01	—
errorlog # DHE-RSA-AES128-SHA256	-9.52949	-3.302732e+01	1.369109e+01	-24.0807
errorlog # ECDHE-RSA-AES256-GCM-SHA384	69.3402	4.496831e+01	1.333942e+01	13.2418
htaccess # ECDHE-RSA-AES128-GCM-SHA256	—	-2.381281e+01	1.302181e+01	7.12765
htaccess # keepalive	—	-9.190407e+01	1.109971e+01	-36.3517
serverlimit-high # DHE-RSA-AES256-GCM-SHA384	-22.6884	1.061666e+02	1.033112e+01	3.66607e-14
threadlimit-high # ECDHE-RSA-AES256-SHA384	—	1.155824e+01	1.023290e+01	—
serverlimit-high # DHE-RSA-AES128-GCM-SHA256	65.9334	8.354528e+01	8.878621e+00	-8.33831e-14
status # ECDHE-RSA-AES256-SHA384	—	1.791372e+01	8.812066e+00	4.59861
serverlimit-high # ECDHE-RSA-AES128-GCM-SHA256	50.9662	6.121697e+01	7.827788e+00	1.53784e-14
symlinks # ECDHE-RSA-AES256-GCM-SHA384	174.237	-9.102600e+00	5.856426e+00	7.10986
errorlog # threadlimit-high	—	-9.349032e+01	5.626083e+00	—
clientcache # htaccess	—	2.662393e+01	5.597671e+00	59.8479
threadlimit-high # ECDHE-RSA-AES128-GCM-SHA256	—	5.320492e+01	3.595630e+00	—

Continued on next page

	nginx	event	worker	prefork
errorlog # ECDHE-RSA-AES128-GCM-SHA256	68.1329	6.598621e+01	2.151258e+00	22.6087
status # ECDHE-RSA-AES128-SHA256	—	5.313886e+01	2.026268e+00	14.3639
errorlog # DHE-RSA-AES256-SHA256	17.7571	1.784526e+01	4.270108e-01	41.8783
ECDHE-RSA-AES128-GCM-SHA256	-450.307	5.205601e+01	4.195689e-01	102.92
serverlimit-high # serverlimit-mid	-3.28573e-14	-1.545190e-13	1.489193e-13	-4.25671e-14
servercache # serverlimit-mid	16.6954	1.655157e-13	1.113557e-13	82.7227
serverlimit-mid # ECDHE-RSA-AES256-SHA384	0.451272	-1.414562e-13	1.014152e-13	-6.40699
serverlimit-mid # DHE-RSA-AES128-GCM-SHA256	39.3	2.611295e-14	7.421710e-14	41.5743
sendfile # serverlimit-mid	1.43223	5.246434e-14	4.287837e-14	35.3246
serverlimit-mid # threadlimit-high	—	3.025948e-14	4.222026e-14	—
serverlimit-mid # DHE-RSA-AES256-SHA256	26.9792	1.212707e-13	2.162007e-14	12.2796
keepalive # serverlimit-mid	28.5026	-8.131260e-14	1.676243e-14	54.0188
serverlimit-mid # DHE-RSA-AES256-GCM-SHA384	4.21315	-1.826556e-14	9.966223e-15	48.7235
errorlog # serverlimit-mid	1.53594	-4.615837e-14	1.678369e-15	-49.3512
serverlimit-mid # symlinks	-31.3699	9.271674e-14	-2.637492e-14	-44.3314
serverlimit-mid # ECDHE-RSA-AES128-GCM-SHA256	18.0058	9.889170e-14	-3.559265e-14	29.5868
serverlimit-mid # status	—	-3.868491e-14	-5.553992e-14	-33.7011
serverlimit-mid # DHE-RSA-AES128-SHA256	-7.10359	-1.947616e-14	-6.587928e-14	-5.65656
serverlimit-mid # ECDHE-RSA-AES128-SHA256	9.9869	-3.132851e-14	-7.932287e-14	19.8663
serverlimit-mid # ECDHE-RSA-AES256-GCM-SHA384	32.1015	-7.806881e-14	-1.054253e-13	23.4201
htaccess # serverlimit-mid	—	-3.858620e-13	-2.079507e-13	-94.7392
dns # serverlimit-mid	—	1.468904e-13	-4.000772e-13	103.624
status	—	-1.241249e+02	-6.701263e-01	-13.7359
htaccess # DHE-RSA-AES128-SHA256	—	-9.580554e+00	-1.268152e+00	25.7722
clientcache # status	—	8.609847e+01	-1.296584e+00	27.6817
htaccess # ECDHE-RSA-AES256-SHA384	—	-4.210222e+01	-4.990026e+00	-62.7515
status # threadlimit-high	—	-1.627480e+01	-5.141075e+00	—
errorlog # ECDHE-RSA-AES256-SHA384	-1.80148	2.864951e+01	-5.464194e+00	13.6698
symlinks # ECDHE-RSA-AES128-SHA256	130.17	2.797686e+01	-7.293490e+00	-15.2638
symlinks # ECDHE-RSA-AES128-GCM-SHA256	195.057	2.160190e+01	-7.451104e+00	1.76501

Continued on next page

	nginx	event	worker	prefork
htaccess # ECDHE-RSA-AES256-GCM-SHA384	—	1.426663e+01	-7.502446e+00	-22.7554
symlinks # ECDHE-RSA-AES256-SHA384	144.732	-2.543227e+01	-7.672063e+00	7.45273
ECDHE-RSA-AES256-GCM-SHA384	-424.775	4.801745e+01	-8.395560e+00	127.112
status # DHE-RSA-AES256-SHA256	—	-3.372876e+01	-8.993203e+00	61.0996
errorlog	-28.8474	-8.327332e+01	-1.294215e+01	-23.3923
symlinks	-139.944	-9.269461e+01	-1.326541e+01	-8.06103
status # DHE-RSA-AES128-SHA256	—	1.517665e+01	-1.521360e+01	-9.69098
htaccess # DHE-RSA-AES128-GCM-SHA256	—	-3.962885e+01	-1.724156e+01	4.50205
serverlimit-high # DHE-RSA-AES256-SHA256	1.09267	5.878677e+01	-1.822684e+01	5.61412e-14
htaccess	—	-8.989361e+01	-1.875960e+01	-22.2966
serverlimit-high # ECDHE-RSA-AES256-SHA384	6.78474	5.205184e+01	-2.125696e+01	-1.44528e-14
htaccess # threadlimit-high	—	-5.100081e+01	-2.150407e+01	—
serverlimit-high	-23.9241	-1.177028e+02	-2.152152e+01	7.48984e+15
status # DHE-RSA-AES128-GCM-SHA256	—	4.722622e+00	-2.254563e+01	15.4696
DHE-RSA-AES128-GCM-SHA256	65.077	2.838590e+01	-2.359298e+01	108.55
serverlimit-high # symlinks	-12.5785	-7.291866e+01	-2.360985e+01	-2.4083e-15
errorlog # serverlimit-high	7.89522	-4.163432e+01	-2.380593e+01	-3.35324e-14
symlinks # DHE-RSA-AES128-GCM-SHA256	129.956	4.837620e+01	-2.502379e+01	-14.2174
keepalive # DHE-RSA-AES256-SHA256	119.337	-1.487370e+02	-2.529792e+01	-252.251
errorlog # status	—	-1.581181e+01	-2.751103e+01	-100.832
errorlog # keepalive	25.6247	-2.183594e+01	-2.826934e+01	26.7955
status # ECDHE-RSA-AES128-GCM-SHA256	—	-2.427203e+01	-3.077852e+01	24.4245
keepalive # symlinks	-122.092	1.030988e+02	-3.225680e+01	-1.11886
htaccess # DHE-RSA-AES256-SHA256	—	-6.842174e+01	-3.241058e+01	-1.22723
threadlimit-high # DHE-RSA-AES128-GCM-SHA256	—	-2.154748e+01	-3.399185e+01	—
threadlimit-high	—	-1.113483e+02	-3.709167e+01	—
errorlog # htaccess	—	-5.125787e+01	-4.279145e+01	-53.0574
symlinks # DHE-RSA-AES256-SHA256	109.593	2.654188e+01	-4.415538e+01	6.91582
htaccess # serverlimit-high	—	-1.898691e+01	-4.661420e+01	8.62626e-14
serverlimit-high # status	—	-3.590183e+01	-4.773168e+01	2.03935e-14

Continued on next page

	nginx	event	worker	prefork
DHE-RSA-AES256-GCM-SHA384	100.989	-6.506298e+00	-4.935262e+01	115.087
symlinks # DHE-RSA-AES128-SHA256	99.3068	3.139370e+01	-5.705454e+01	-34.3308
compression	-28.4999	-1.863229e+02	-5.774974e+01	-86.9201
sendfile # servercache	77.8753	-1.278669e+02	-6.094423e+01	-67.1171
keepalive # DHE-RSA-AES128-SHA256	66.5804	-1.442637e+02	-6.436555e+01	-241.108
htaccess # status	—	-7.413180e+01	-6.556643e+01	-81.5474
errorlog # DHE-RSA-AES128-GCM-SHA256	60.3098	-3.081500e+01	-6.599983e+01	54.29
htaccess # symlinks	—	-3.755975e+01	-7.156345e+01	-77.076
clientcache # sendfile	-69.8335	-1.317077e+02	-7.534835e+01	-97.9324
errorlog # symlinks	-28.1406	-1.196897e+02	-7.569580e+01	-99.6179
sendfile # DHE-RSA-AES256-GCM-SHA384	-166.321	-1.814957e+02	-9.417953e+01	-167.148
compression # sendfile	34.8395	-1.556941e+02	-1.008295e+02	-64.1585
clientcache # keepalive	16.3652	-3.382973e+02	-1.039169e+02	-305
status # symlinks	—	-7.444757e+01	-1.039687e+02	-13.1578
accesslog	33.278	-2.363019e+02	-1.097639e+02	-102.075
clientcache # ECDHE-RSA-AES128-SHA256	-155.502	-8.154020e+01	-1.101814e+02	-164.288
sendfile # DHE-RSA-AES128-SHA256	-131.74	-2.446224e+02	-1.184113e+02	-157.584
dns	—	-2.263627e+02	-1.210012e+02	-123.528
sendfile # DHE-RSA-AES128-GCM-SHA256	-69.2307	-2.318935e+02	-1.295240e+02	-152.577
sendfile # DHE-RSA-AES256-SHA256	-113.833	-2.325662e+02	-1.340189e+02	-151.5
sendfile # ECDHE-RSA-AES256-SHA384	-98.0516	-1.829796e+02	-1.358094e+02	-143.801
sendfile # ECDHE-RSA-AES128-SHA256	-97.8965	-1.868206e+02	-1.435042e+02	-113.287
sendfile # ECDHE-RSA-AES256-GCM-SHA384	-63.6104	-2.100134e+02	-1.505025e+02	-169.064
sendfile # ECDHE-RSA-AES128-GCM-SHA256	-61.8108	-1.662847e+02	-1.517443e+02	-158.369
clientcache # ECDHE-RSA-AES256-SHA384	-158.419	-1.459778e+02	-2.160189e+02	-228.376
keepalive # sendfile	-13.6055	-2.239205e+02	-2.454103e+02	-234.491
keepalive # DHE-RSA-AES256-GCM-SHA384	-223.52	-5.584029e+02	-2.987718e+02	-509.189
keepalive # ECDHE-RSA-AES256-GCM-SHA384	246.36	-4.686236e+02	-3.065695e+02	-537.716
keepalive # DHE-RSA-AES128-GCM-SHA256	-238.82	-5.942171e+02	-3.339850e+02	-519.857
keepalive # ECDHE-RSA-AES128-GCM-SHA256	227.09	-5.734060e+02	-3.579388e+02	-478.911

Continued on next page

	nginx	event	worker	prefork
servercache # DHE-RSA-AES256-GCM-SHA384	198.292	-3.911225e+02	-4.268028e+02	-411.27
servercache # ECDHE-RSA-AES128-GCM-SHA256	251.949	-3.755116e+02	-4.272586e+02	-477.626
servercache # ECDHE-RSA-AES256-GCM-SHA384	247.981	-3.621592e+02	-4.526059e+02	-410.662
servercache # DHE-RSA-AES128-GCM-SHA256	210.709	-4.719053e+02	-4.684665e+02	-446.109
compression # servercache	4.39833	-5.567940e+02	-4.839253e+02	-453.458
servercache # DHE-RSA-AES256-SHA256	214.948	-5.328524e+02	-5.676146e+02	-528.048
servercache # ECDHE-RSA-AES128-SHA256	227.246	-4.323488e+02	-5.680030e+02	-516.921
servercache # DHE-RSA-AES128-SHA256	192.353	-5.998628e+02	-5.928876e+02	-583.711
servercache # ECDHE-RSA-AES256-SHA384	203.2	-4.961530e+02	-5.994492e+02	-553.147
keepalive # servercache	-255.104	-6.093050e+02	-6.657037e+02	-625.119
accesslog # dns	—	-1.272809e+03	-1.098180e+03	-1054.4
clientcache	55.9786	-2.144093e+03	-1.803392e+03	-1596.12
clientcache # serverlimit-mid	25.4084	-3.419546e+10	-3.877677e+12	69.7144
clientcache # servercache	3.29383e-14	-5.627306e+10	-5.924486e+12	-3.24025e+11
accesslog # serverlimit-mid	-22.701	8.239777e+13	-1.112204e+15	46.6786
serverlimit-mid	-14.5507	-4.383150e+15	-3.941224e+15	-42.8298

Table 19: The full models for all server, including all pairwise interactions. Sorted by the worker column.

List of Tables

List of Figures