Workload-aware System Monitoring Using Performance Predictions Applied to a Large-scale E-Mail System

Christoph Rathfelder
FZI Research Center for Information Technology
Karlsruhe, Germany
rathfelder@fzi.de

Stefan Becker
1&1 Internet AG
Mail & Media Development
Karlsruhe, Germany
stefan.becker@1und1.de

Klaus Krogmann
FZI Research Center for Information Technology
Karlsruhe, Germany
krogmann@fzi.de

Ralf Reussner
Karlsruhe Institute of Technology
Karlsruhe, Germany
reussner@kit.edu

***Abstract*—Offering services in the internet requires a dependable operation of the underlying software systems with guaranteed quality of service. The workload of such systems typically significantly varies throughout a day and thus leads to changing resource utilisations. Existing system monitoring tools often use fixed threshold values to determine if a system is in an unexpected state. Especially in low load situations, deviations from the system's expected behaviour are detected too late if fixed value thresholds (leveled for peak loads) are used. In this paper, we present our approach of a workload-aware performance monitoring process based on performance prediction techniques. This approach allows early detections of performance problems before they become critical. We applied our approach to the e-mail system operated by Germany's largest e-mail provider, the 1&1 Internet AG. This case study demonstrates the applicability of our approach and shows its accuracy in the predicted resource utilisation with an error of mostly less than 10%.**

***Keywords*-software; software performance; software architecture; predictive models; industrial case study**

I. Introduction

The dependable operation of software systems with guaranteed quality of service is one of the most important aspects when hosting large and distributed systems. Due to the complexity of such systems and the distribution across several servers, the detection of potential performance problems like overloaded resources or delayed requests, is a complex but substantially important activity. In order to detect such potential problems and malfunctions within the system and the infrastructure as early as possible, management applications enable a centralised and automated collection and aggregation of performance indicators like CPU and network utilisation, the current length of request queues, or the processing time of requests.

In most systems, the workload induced by users varies over time. Especially in the case of systems offering services to end users over the internet, significant variation regarding the system's usage depending on the time of day can be observed [1]. These changes lead to variations within the resource utilisation of servers and the system's response times. These load-dependent variations complicate the detection of anomalies such as "higher CPU utilisation than expected" that might be indicators for potential performance problems.

Existing monitoring and software management solutions support the definition of rules and conditions that are evaluated at runtime in order to detect potential performance problems and to identify malfunctions of the system. Most of these solutions use fixed thresholds as upper or lower bounds to differentiate between normal system operation and a potentially problematical system state [2]. However, using fixed values allows for detecting problems only when the system is under high load (i.e. the conditions are already critical or close to critical). In low load situations, deviations from the expected value and potential malfunctions can hardly be detected. Some more advanced monitoring systems already support time-dependent thresholds to handle load variations. These systems limited to fixed recurring patterns in the usage of the system and can not handle unexpected peak loads or other variations in the user behaviour, which might be caused by uncontrollable factors.

In this paper, we present our approach of a workload-aware performance monitoring process. We apply model-based performance prediction techniques to derive the expected behaviour and resource utilisation induced by the current workload. Deviations between predicted and measured values serve as early indicators of performance abnormalities for all workload situations. Compared to the use of fixed threshold values, the detection of unexpected behaviour can be improved, especially in the case of low load situations. We describe the application of our method to the e-mail system operated by the 1&1 Internet AG, which is one of Europe's largest e-mail providers. Additionally, this case study demonstrates and validates the applicability of our modelling and prediction approach to large and distributed systems.

The contributions of this paper are: i) A workload-aware performance monitoring process, ii) the application of our process to a large industrial case study, iii) the validation of our prediction approach based on this case study, and finally, iv) a list of experiences we gained when applying

Further information on our case study can be found on http://sdqweb.ipd.kit.edu/wiki/WICSA_ECSA_2012

our approach to this large distributed system. The case study demonstrates that our approach enables the detection of deviations from the expected behaviour that would not be detected using fixed thresholds. Furthermore, we evaluate the accuracy of the applied prediction technique, showing a prediction error of less than 10% in most cases.

The remainder of this paper is organised as follows. Sect. II introduces the foundations of our work including an overview of the e-mail system operated by the 1&1 Internet AG and an introduction of our performance prediction approach. Sect. III describes the workload-aware system monitoring process. In Sect. IV, we present the application of our process in the context of the 1&1 e-mail system. Sect. V presents the evaluation of the prediction accuracy. In Sect. VI, we summarise the experiences we gained when applying our method to this large distributed system. Next, we present an overview of related work and finally conclude with a brief summary and a discussion of future work.

II. Foundations

The following gives an overview of the e-mail system operated by the 1&1 Internet AG, which forms the basis for our case study. Furthermore, we introduce software performance engineering in general and the prediction approach Palladio we applied in our case study in particular.

A. 1&1 E-Mail System

1&1 Internet AG is a brand of the *United Internet AG*, which also includes the brands *Web.de*, *GMX*, *United Internet Media*, *Fasthosts*, *InterNetX*, *AdLINK MEDIA*, *Affilinet*, and *Sedo*. With over 13 locations, the *United Internet AG* is the world's biggest web hoster and leading domain registrar. The company is number two in DSL access and the number one in e-mail services in Germany. The 1&1 e-mail system is the basis of several national and international e-mail platforms like *GMX*, *web.de*, *mail.com* and *india.com*. The system comprises more than 2,000 servers and provides services for more than 40 million users in Germany alone.

On the systems back-end, the core functionalities for e-mail sending, receiving, requesting and persistence are realised and provided through interfaces like restful HTTP services, POP3, or IMAP. Further products such as web clients or mobile mailers are built on top of these interfaces. The different internal software components follow the service-oriented design paradigm. The dimensioning of the system allows for an individual deployment of each component on dedicated servers that exist in redundant instances. Figure 1 illustrates the components that realise the core-functionality and the number of deployed instances.

In the STORE, folder structures of mailboxes, e-mails and their attachments are saved. The SERIE and DBFM databases provide fast access to information about the internal instances to which single mailboxes are dedicated. Mail Delivery Agents (MDA) and Mail Transfer Agents

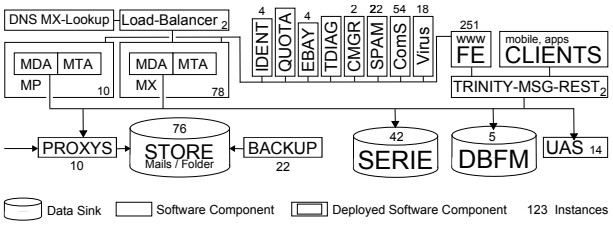

Figure 1. Back-end software components of the e-mail system

(MTA) are located on the Mail Exchanger (MX) and Mail Proxy (MP) servers. They are external interfaces of the system and responsible for sending and receiving e-mails. For requesting e-mails, external clients connect to the Proxy servers using IMAP and POP3 requests, which translate and forward the requests to the STORE servers. Internally managed clients such as web-based user interfaces and mobile mailers use the restful HTTP services. Additionally, several other components for contact-management, virus- and spam-protection, handling of quotas, trusted e-mails and other tasks are available.

B. Performance Modelling and Prediction

Over the last fifteen years, a number of approaches have been proposed for integrating performance prediction techniques into the software engineering process. Efforts were initiated with Smith's seminal work on Software Performance Engineering (SPE) [3]. Since then, a number of architecture-level performance meta-models have been developed by the performance engineering community. The most prominent examples are the UML SPT profile [4] and its successor, the UML MARTE profile [5]. Both of them are extensions of the UML as the de-facto standard modelling language for software architectures. All approaches have in common that they aim on the evaluation of the system's performance at design time. They support the comparison of different design alternatives and deployment options.

In model-based quality prediction processes at the architecture level as described in [6], the starting point of this process is a model that describes the software system itself using an established modelling language such as the UML. General software models do not include any specific information regarding the performance characteristics of a software. This information is added in the next step. If the system is not implemented yet, the resource demands are estimated. If an implementation is already available, for example in the case of a refactoring, measurements of the system can be used to gather the relevant information to annotate the model. The annotation can be done using one of the UML profiles mentioned before or using a meta-model designed specifically for this purpose, such as KLAPER [7] or the Palladio Component Model (PCM) [8], which we selected for our case study. The annotated software model is used as an input for a transformation into a dedicated performance model like layered queueing networks

or queueing Petri nets. This performance model is then evaluated by analysis or simulation techniques. In a final step, the prediction results are returned as a feedback related to the original software model.

C. Palladio Component Model (PCM)

The PCM [8] is a domain-specific modelling language for component-based software architectures. It supports an automated transformation of architecture-level performance models to predictive performance models including layered queueing networks [9], queueing Petri nets [10] and simulation models [6]. PCM supports the evaluation of different performance metrics, including response time, maximum throughput, and resource utilisation. The PCM approach provides an Eclipse-based modelling and prediction tool (http://www.palladio-simulator.com). Further details and a technical description can be found in [11].

The performance of a component-based software system is influenced by four factors [11]: The implementation of system components, the performance of external components used, the deployment platform (e.g., hardware, middleware, networks), and the system's usage profile. In the PCM, each of these factors is modelled in a specific sub-model and thus can be adapted independently. The composition of these models forms a PCM instance.

The *Repository Model* specifies system components and their behaviour. Components provide and require interfaces. The PCM provides a description language, called *ResourceDemandingServiceEffectSpecification* (RD-SEFF) to specify the behaviour of the components and their resource demands. The *System Model* describes the structure of the system by interconnecting components via their provided and required interfaces. In the *Allocation Model*, system's components are allocated to physical resources described in the *Resource Environment*. The latter model specifies the hardware environment the system is executed on (e.g. servers, processor speed, network links). The *Usage Model* describes the workload induced by the system's end-users. For example, the workload may specify how many users access the system, the inter-arrival time of requests, and their input parameters. Usage profiles within the model represent individual user behaviours.

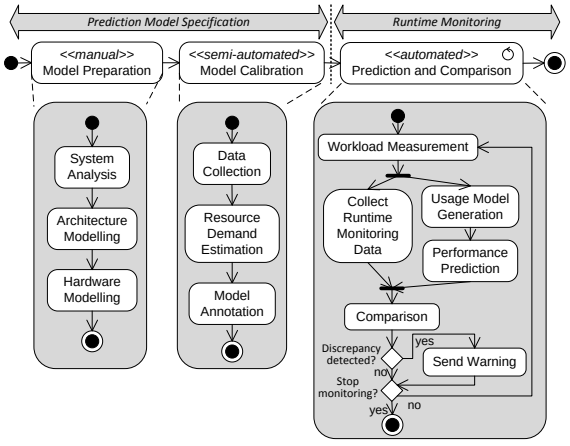

Figure 2. Workload-aware continuous performance monitoring process

Usually, parameter values have an influence on the software's behaviour and thus the resource demands. However, often it is not possible to explicitly model these dependencies. The PCM offers *random variables* to abstract such parameter dependencies. Random variables can be specified by various probability distribution functions.

III. Workload-aware Continuous Performance Monitoring Process

The integration of performance predictions into a continuous system monitoring process promises several benefits compared to the use of fixed thresholds. Performance predictions consider the influence of the workload on the system's behaviour and resource utilisation. Thus, the predicted performance metrics vary corresponding to the metrics collected on the running system. If the runtime measurements correlate with the predictions, the system is in the expected state. Using workload-aware adaptations of the thresholds allows for an improved and faster detection of deviations between measured and expected values. In contrast to the use of historical data to derive the thresholds and their variation over time, the presented prediction-based approach can also be applied in situation when the workload caused by the users differs from the normal and expected one or in situations that are not covered by the historical data. Additionally, using architecture-level performance models, allows us adapting the prediction models to changes of the systems's architecture or deployment without re-collecting a large set of historical data to derive the varying monitoring thresholds. The detailed prediction results support the system operator in detecting the root cause. Furthermore, the prediction model itself can be used by the operator to evaluate the performance impact and compare different possible counteraction like for example adding new hardware resources.

In Figure 2, we illustrate our workload-aware continuous performance monitoring process, which consists of the main activities *Model Preparation*, *Model Calibration*, and *Prediction and Comparison*. The result of the first two activities is an initial performance model. This model has to be specified only once and is used within the fully automated continuous runtime monitoring represented by the *Prediction and Comparison* activity. Each of these main activities consists of several sub activities, which we describe in the following.

A. Model Preparation

The aim of this first activity is the creation of a model describing the system. Therefore, the sub process starts with the *System Analysis* sub activity. This includes the identification of components, their interactions and their behaviour. Additionally, it is necessary to identify the available

hardware resources and the deployment of the system on those resources.

Based on this information, the architecture of the system is described in the *Architecture Modelling* step using an architecture description language like the PCM. This architecture model consists of the specification of components including a description of their behaviours as well as the provided and required interfaces and their connections to build up the complete system. In the PCM these aspects are covered within the Repository and System Model.

As a final step of the model preparation activity, the available hardware resources, namely the different servers including the available CPUs and storage devices and their network connections are described. Additionally, the *Hardware Modelling* includes the specification of the deployment of system components on available servers.

B. Model Calibration

In order to enable the prediction of resource utilisations and response times, the behavioural specifications of the architectural model need to be annotated and calibrated with resource demands (e.g., amount of data written to disk and required CPU time). In our case study, we automated the data collection and analysis. However, depending on the scenario, it might be necessary to perform these activities manually. The first sub activity is the *Data Collection*. Systematic experiments as proposed in [12] can be conducted to ease the derivation of dependencies between input parameters like the size of an e-mail attachment and the resulting resource demand. However, such experiments require to set up an equivalent system. For large systems such as the e-mail system of 1&1, distributed over hundreds of servers, this is quite infeasible. In such cases, the calibration needs to be done using monitoring and log data of the live system.

The collected data often only consists of aggregated values like the mean resource utilisation, the number of invocations per second, or the overall CPU time of a specific component. Based on the collected data, the *Resource Demand Estimation* activity is responsible for deriving the individual resource demands for each operation provided by a service based on the collected data. Furthermore, the estimation also includes the analysis of dependencies between input parameters like the size of an attachment or the type of request and the induced resource demand.

As part of the *Model Annotation* activity, the estimated resource demands including their dependencies on other parameters are integrated into the prediction model. The Stochastic Expression Language (StoEx), which is part of the PCM, allows for specifying different types of resource demands, including simple constant values, different probabilistic distribution functions, and mathematical functions reflecting the dependencies between input parameter values and the resulting resource demands.

C. Prediction and Comparison

After the specification of the performance model in the previous activities, the performance monitoring can be started. As mentioned before, the utilisation of resources often varies depending on the workload of the system. For this reason, the prediction of the expected resource utilisation requires measuring the current workload, which is performed in the measurement workload activity. In most production systems, the external interfaces are already extended with interfaces to monitor the system's usage. Hence, these interfaces can be used to collect the required data.

Based on this data, the prediction model is completed with the specification of the system's usage within the *Usage Model Generation* activity. This specification can contain several independent usage behaviour to differentiate between the normal usage of the system by customer and for example additionally data analyse or report generation processes invoked by internal users of the system. To reduce the runtime load induced by monitoring and data collection, live monitoring data often do not include as much information as the data gathered in the previous data collection step. The resulting Usage Model is then used as an input to the *Performance Prediction* activity, which in our case is a PCM-based performance simulation, to derive the expected utilisation of servers and response times of operations.

After collecting the performance metrics of interest (e.g., CPU utilisation, service response times) on the running system and predicting the expected values for the current workload, they are analysed and compared. In the case of a deviation between predicted and measured values within the *Comparison* activity, a warning is sent to the system's operator. The operator is then in charge to analyse the potential error and plan counteractions if required.

IV. Case Study

In this section we present the application of our approach in the industrial context of the 1&1 e-mail system. For the sake of brevity, we focus in this paper on the STORE subsystem, which still consists of about 100 server nodes. The whole case study [13] covers the complete incoming e-mail processing including the MX, Virus, and Spam servers as well as the SERIES and DBFM databases. After presenting further details of the STORE subsystem, we demonstrate the use of the PCM for modelling the system and the application of performance predictions.

A. STORE Subsystem

The STORE consists of several software components that are deployed on three different types of servers, namely *Proxy*, *Store*, and *Backup* servers. The Proxy servers protect the Store servers from a direct access by hosts that are not part of the e-mail system. They provide access to the e-mails using IMAP and POP3 protocols by dedicated components. Additionally, the SGATE component is responsible

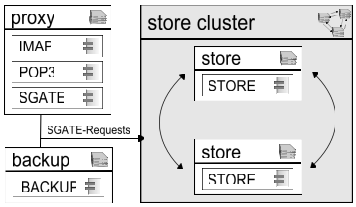

Figure 3. Composition of the Store

for handling internal requests from other components of the e-mail system. The communication between Proxy and Store servers is exclusively performed by the SGATE component. Thus, the IMAP and POP3 components act as adapters that transform IMAP and POP3 requests into requests supported by SGATE and vice versa. The Store servers are responsible for storing all data related to customers mailboxes including the folder structures, e-mail texts and attachments.

To guarantee high availability and to prevent data loss, each Store server is running as a cluster of two servers. A client's mailbox is associated with exactly one cluster. The responsibility for mailboxes is balanced over both hosts while the mailbox data is stored and continuously synchronised on both servers. In the case of software or hardware failures on one host, the other host automatically takes over the responsibility within seconds. The Backup servers are an additional instance to assert data persistency.

In order to handle the large amount of requests, multiple instances of all components are deployed on several servers to distribute the load. The SGATE, IMAP, and POP3 are running on 10 servers, the Store clusters consist of overall 76 servers, and the BACKUP component is deployed on 22 servers.

B. Performance Modelling Study

After this overview of the STORE system, we will now describe the application of the PCM as performance model in the context of the workload-aware continuous performance monitoring process.

1) Model Preparation: The selection of an adequate abstraction layer that promises a good trade-off between prediction accuracy and modelling effort requires an analysis of the system regarding the available calibration data and the potential performance-influencing factors. The monitoring tool running on the Store and Backup servers only measures the resource utilisation of the whole server and does not allow to measure the resource consumption of individual processes or components. For this reason, we introduce logical software components STORE and BACKUP that summarise all resource utilisations on one of these servers. Due to the missing calibration data, a more fine-grained model would not provide any improvements for the prediction results. On the Proxy servers, component-dependent resource monitoring is possible. Thus, we model the IMAP, POP3, and SGATE components as individual components as illustrated in Figure 3.

Table I
PERFORMANT RELEVANT REQUESTS

miweb::SGATE	%	p
SortMails	**49**	
GetMailText	**16**	
GetMails	**14**	
AppendMail	**9**	
ChangeMails	**5**	
MoveMails	4	i
RemoveMails	1	i
ReplaceMail	1	i
CreateFolder	0	
Subscribe	0	
ChangeFolder	0	
FindMails	0	
RemoveFolder	0	
Unsubscribe	0	
GetFolderInfos	0	

miweb::POP	%	p
COMPLETE	**31**	
TOP0	**21**	
ABORTED	**16**	
RETR	**12**	
DELETE	**8**	
TOPN	**8**	

miweb::IMAP	%	p
FETCH	30	
STATUS	30	
SEARCH	15	
STORE	10	
SELECT	4	i
LIST	2	i
LOGIN	2	i
CAPABILITY	2	i
ID	1	i
LOGOUT	1	i
CLOSE	1	i
APPEND	**0**	**e**
.	0	
.	0	
.	0	

msweb::STORE	%	p
AppendMail	**44**	
ChangeMails	**27**	
RemoveMails	**10**	
ExpungeMails	**9**	
MoveMails	**8**	
CopyMails	**1**	
Subscribe	0	
CreateFolder	0	
ChangeFolder	0	
SelectFolder	0	
Create	0	
Unsubscribe	0	
RemoveFolder	0	

mbweb::BACKUP	%	p
remove expired	21	
u-summary	17	
sm-summary	17	
f-summary	17	
AppendMail	11	
force deletion	8	
UpdateMails	3	
retrieve non ex.	2	
RemoveMails	2	
CopyMails	1	
.	0	
.	0	
.	0	

Legend:
%: Portion of Workload
p: *Interview based characterisation*
i: *Performance irrelevant*
e: *Expensive operation*

bold: *Considered within the model*

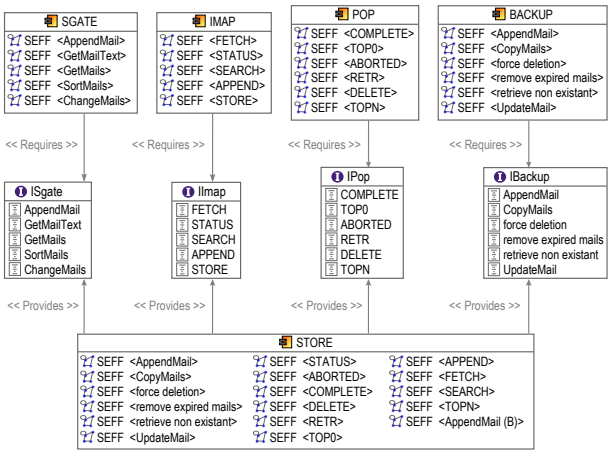

Figure 4. The Repository as part of the structural model

The API of the SGATE component supports a large set of different request types. Modelling the huge amount of different possible requests would result in an infeasible modelling effort. To identify the methods with a potentially high impact on the overall system's performance, we analysed existing log files and performed interviews with the responsible architects and developers. The selection of the performance-relevant request types is based on two criteria. A request is categorised as potentially performance relevant if its occurrence within the overall workload is higher than 1% in average or the request is expected to be very resource intensive. Interviews with the developers were used to identify these resource intensive requests and additionally to select requests that are known as performance-irrelevant (see Table I). In our performance model, we included all potentially performance relevant requests that are not classified as irrelevant as well as all requests classified as expensive operations.

Based on this system analysis, we started system modelling by describing the different components in the PCM

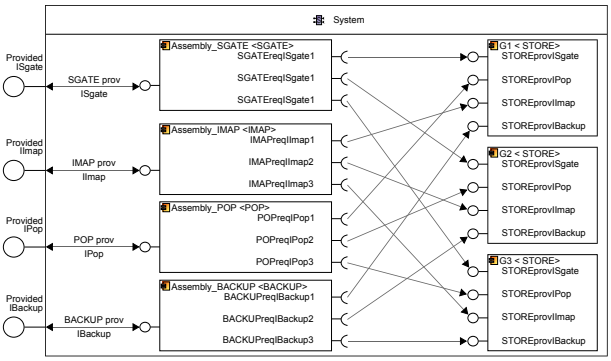

Figure 5. System Model of the store

Repository Model, which is shown in Figure 4. In addition to the interfaces, components, and roles (assignment of provided and required interfaces), the Repository includes a RD-SEFF for each provided interface.

As part of the PCM System Model (see Figure 5), the different components defined within the Repository are instantiated and the components SGATE, POP, IMAP, BACKUP and STORE are connected to each other through the interfaces ISgate, IPop, IImap and IBackup. Additionally, the model includes the definition of four system external interfaces that are used to connect the Usage Model, which contains the user behaviour generated within the performance monitoring activity, with the System Model. The model includes three instances of the STORE component, as during the interviews, we identified that the servers hosting the Store can be clustered in three server types with identical hardware. The load balancing between the different server groups is integrated into the behavioural specifications and not modelled as an explicit load balancing component.

The PCM provides a dedicated model, called the *Resource Environment*, to describe the hardware resources of the system. Caused by the huge number of involved servers – in this subsystem over 100 – we had to find a way to abstract from modelling each individual server to reduce the size of the model and the required modelling effort. As already mentioned above, the servers can be classified based on their hardware and configuration into clusters of servers with nearly identical resources and configuration. For each group, we modelled only one node within the *Resource Environment*. Since all the Proxy servers are equal in their hard- and software configuration, they are represented by one logical server in the model. All logical servers are sized with the cumulated resource capacities of the real servers they represent.

2) Model Calibration: For each server, we analysed log files and monitoring data to derive those hardware resources (e.g., CPU, HDD, LAN) which are significantly used by the implementation of the system. In the case of the Proxy servers, these resources are CPUs and the network connection. As expected, the HDD usage is very low on these servers, since they only forward requests without any local

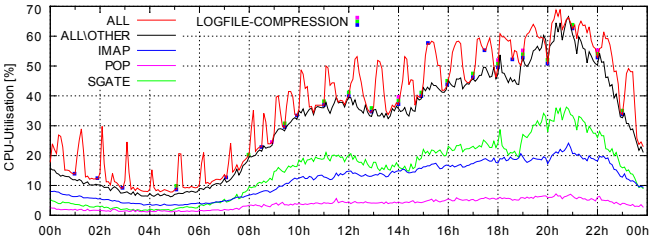

Figure 6. CPU Utilisation on a Proxy server over 24 hours

Table III
CPU DEMANDS CAUSED BY THE POP COMPONENT

Request	CPU Demand
ABORTED	16134069 Cycles / Request
COMPLETE	4520880 Cycles / Request
TOPN	3267070 Cycles / Request
DELETE	684569 Cycles / Request
RETR	353879 Cycles / Request
TOP0	4319 Cycles / Request
RETR	48 Cycles / Byte

persistence. In contrary, on the Store servers, the HDDs, CPUs, and network connections are heavily demanded resources and therefore need to be modelled.

For the sake of brevity, we now focus our description on the calibration of the Proxy servers' components and omit the Store and Backup servers. The procedure is the same for the other components. The system monitoring tools running on the Proxy servers provided us with detailed measurements of the resource utilisation. Each component, namely the IMAP, SGATE, and POP3, are individually measured by the tool, which provides logs of the induced resource consumption every 30 seconds. Additionally, the overall CPU utilisation is logged. Thus we were able to derive the resource demands that are induced by the operating system or other monitoring activities. Figure 6 shows the CPU utilisation on a Proxy server over 24 hours, induced by the software components SGATE, IMAP and POP, as well as the sum of these values and the overall utilisation.

Detailed analyses of the overhead peaks showed that they appear directly after the compression of request log files is initiated. To estimate the resource demand for each request type classified as performance relevant, we conducted further measurements to derive the distribution frequency of the different request types. We collected 864 sample sets covering 5 minutes each. In addition, data about the size of the requested data was collected. Table II shows the derived correlation coefficients of the request types and the resource utilisations caused by them.

In order to calculate the absolute resource demands of the different request types, we had to consider the available hardware resources on the servers. Each server is equipped with two dual core CPU running with 2GHz. The network connection has a capacity of 2GBit/s. Based on this knowledge, we calculated our modelling functions describing the resource demand for each resource type. Table III exemplary shows the utilisation of a Proxy server's CPU resource, caused by the processed requests of the POP component. We differentiate between resource demands with fixed value

Table II
CORRELATION COEFFICIENTS BETWEEN REQUESTS AND RESOURCE UTILISATIONS

	SGATE														POP								IMAP				
	# AppendMail	Σ AppendMail size	# GetMailText	Σ GetMailText length	# GetMails (+)	Σ GetMails (+) ids	# GetMails (-)	Σ GetMails (-) ids	# SortMails (+)	Σ SortMails (+) ids	# SortMails (-)	# ChangeMails (+)	Σ ChangeMails (+) ids	# ChangeMails (-)	# COMPLETE	# TOPO	# ABORTED	# RETR	Σ RETR size	# DELETE	# TOPN	Σ TOPN lines	# FETCH	# STATUS	# SEARCH	# APPEND	# STORE
CPU	0.79	0.83	0.86	0.7	0.86	0.74	0.84	0.86	0.86	0.86	0.85	0.58	0.34	0.86	0.9	0.82	0.91	0.86	0.84	0.83	0.03	0	0.87	0.9	0.87	0.79	0.89
CPU SGATE	0.82	0.87	0.97	0.78	0.97	0.86	0.96	0.97	0.97	0.97	0.97	0.71	0.45	0.96													
CPU POP															0.95	0.85	0.94	0.91	0.91	0.88	0.15	0.06					
CPU IMAP																							0.96	0.96	0.91	0.84	0.93
CPU OTHER	0.27	0.27	0.24	0.2	0.23	0.18	0.22	0.24	0.23	0.24	0.23	0.12	0.05	0.24	0.35	0.33	0.36	0.32	0.31	0.3	-0	-0	0.4	0.38	0.32	0.31	0.34
NET-IN	0.81	0.87	0.97	0.8	0.97	0.85	0.95	0.96	0.96	0.97	0.96	0.66	0.4	0.96	0.9	0.8	0.9	0.86	0.84	0.83	0.08	0	0.81	0.88	0.89	0.8	0.89
NET-OUT	0.84	0.9	0.96	0.79	0.96	0.85	0.94	0.95	0.95	0.96	0.95	0.65	0.39	0.96	0.91	0.8	0.91	0.87	0.86	0.84	0.05	0	0.81	0.88	0.9	0.8	0.9
DISK-READ	0.05	0.05	0.09	0.08	0.09	0.06	0.09	0.09	0.09	0.09	0.09	0.06	0.04	0.09	0.08	0.07	0.08	0.08	0.08	0.08	-0	0	0.08	0.08	0.08	0.06	0.08
DISK-WRITE	0.48	0.49	0.49	0.4	0.48	0.41	0.47	0.49	0.48	0.48	0.47	0.31	0.17	0.49	0.57	0.52	0.56	0.52	0.5	0.5	0.01	0.02	0.57	0.56	0.54	0.49	0.55

Amount of processed requests within one examined interval

∑ Sum of a numerical parameter over the requests of the examined interval

\+ Boolean parameter, indicating expensive requests

\- Boolean parameter, indicating cheap requests

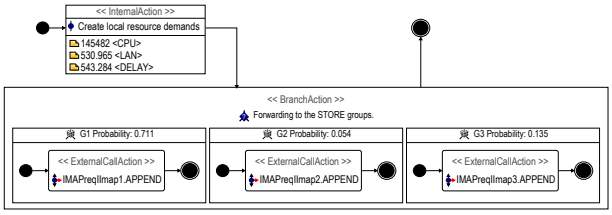

Figure 7. RD-SEFF of the APPENDMAIL request on a Proxy server

for each request and request demands that depend on the size of the contained data.

In a last step, the behavioural descriptions of the components are annotated with the derived resource demands. As already mentioned above, we integrated the load balancing functionality within the proxy component. Thus, we also needed to calibrate the distribution of requests to the different groups of servers, i.e. the three Store server groups mentioned before. We modelled the load balancing behaviour with a *ProbabilisticBranchAction* (control flow branch in the RD-SEFF) based on data available from individual request logs contained in the Store cluster's redirect logs. Figure 7 shows the annotated RD-SEFF for the APPENDMAIL request on a Proxy server. It includes the resource demands for different resources of the server and the BranchAction forwarding the requests to the different server groups.

3) Prediction and Comparison: In order to consider the current workload of the running system in the performance prediction, we first have to specify the user interactions with the system. For each external interface specified in the System Model, a dedicated usage profile is defined within the Usage Model. Each usage profile is a template for the measured workload, i.e. request frequencies can be automatically integrated.

To achieve full automation, we implemented tools that automatically collect, parse, and analyse the log files of the servers. Based on this data, the parameters for the usage profile are calculated. These values are then automatically integrated in the XML representation of the usage profiles by replacing the previously inserted placeholder values. The usage profiles are then used to execute the performance predictions. The relevant prediction results, namely the average

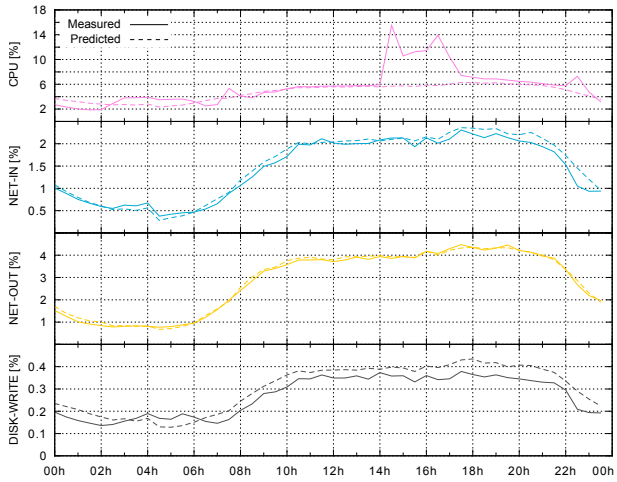

Figure 8. Measured and predicted resource utilisations of a Store server

resource utilisations, are compared with the measured values to identify mismatches.

Since we are working with the live system, we were not allowed to induce failures resulting in an unexpected behaviour of the system. To demonstrate the detection capabilities of our process, we applied our approach during a planned software update on the live system as the system administrators expected that the update induces additional load on the system resulting in deviations from the normal and expected behaviour of the system. Figure 8 illustrates the monitored and predicted resource utilisations over the day of the scheduled update. In the time slot between 2 and 5 o'clock PM, the measured CPU utilisation was significantly higher than the predicted values. This deviations from the expected respectively predicted values started exactly at the same point, when the rollout of the update was executed. With this scenario, we could demonstrate that deviations from the expected resource utilisation under the given workload can be detected using our approach.

V. EVALUATION OF PREDICTION ACCURACY

With the case study presented in the previous section, we demonstrated the applicability of PCM-based performance predictions in an industrial context. However, the accuracy of the prediction results has not been discussed. Since the

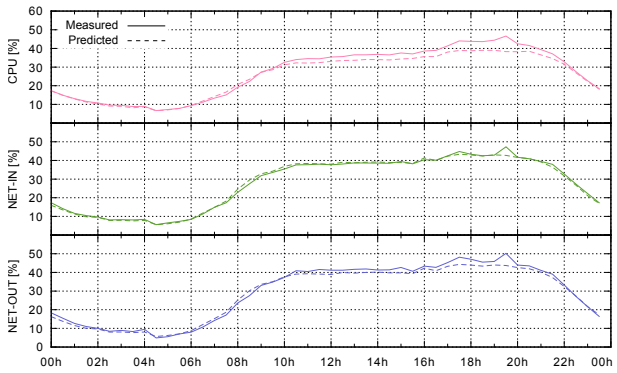

Figure 9. Measured and predicted resource utilisation of a Proxy server

accuracy is an important aspect when integrating predictions in a performance monitoring process, this section presents a detailed evaluation of the prediction results. Validating performance predictions is often done by conducting benchmarks or controlled experiments and compare measured with predicted values. Such experiments cannot be performed on the live system as they might influence the reliable operation of the system. To perform integration tests, a dedicated test system is available. However this system is downsized and running on virtualised systems with some business logic realised as mockups. Thus, this test system cannot be used to perform representative load tests and performance measurements. Due to the costs, complexity, and size of the system, we refrained from setting up an equivalent system with identical hardware resources and a load driver inducing the workload of millions of users.

Instead, we base our evaluation on measurements on the live system using the real workload. We exploit the fact that the workload of the system significantly varies during a day of operation, which allows us for collecting measurements in different load situations. The workload varies between low load situations at night and high load situations in the evening, when after work, the private mailboxes are checked for new e-mails.

Within our evaluation, we analysed one reference-day and measured the average resource utilisation every 30 minutes. Additionally, we derived the 48 usage profiles of the same intervals using the existing monitoring system. This way, we gained 48 tuples of predicted and measured values representing different load situations of the system. In contrast to the calibration measurements, these measurements cover the whole workload range and thus are a valid set of measurements to perform an evaluation of the accuracy.

Figure 9, presents the measured (solid) and predicted (dashed) resource utilisations on the Proxy servers for one day. The predicted curves have the same characteristics compared to the measured ones, with small differences between predicted and measured values. They show that the expected resource utilisation based on the performance predictions fits the measured values on the live system with

Table IV
ERROR CHARACTERISTICS

	CPU		NET-IN		NET-OUT		DISK-WRITE	
	relative [%]	absolute [PP]	relative [%]	absolute [PP]	relative [%]	absolute [PP]	relative [%]	absolute [PP]
Entire model								
Average	12.94	1.32	5.99	0.25	5.92	0.5	9.64	0.027
Upper decile	36.7	2.98	13.46	0.56	12.37	1.55	20.1	0.052
Proxy server								
Average	5.9	1.91	2.93	0.67	5.92	1.48		
Upper decile	10.1	4.24	6.4	1.43	11.09	2.5		
Store server								
Average	15.29	1.13	7.01	0.12	6.01	0.18	9.64	0.027
Upper decile	42.1	2.47	16.19	0.25	13.31	0.40	20.1	0.052

only small deviations. In Table IV, we list the averaged relative error in %, the averaged absolute error in percent points (PP), and the upper decile for the errors between measured and predicted values of the entire model and individually for the Proxy and Store servers for all observed resources. Although the percental error exceeds 10% in few cases, the averaged absolute error is always less than 2 PP and upper deciles never exceed 5 PP. This accuracy is acceptable for the integration in a performance monitoring process.

VI. EXPERIENCES AND LESSONS LEARNED

During our case study, we faced several challenges when modelling the system caused by its size and the fact, that we were not allowed to perform any experiments on the live system. But even if we had been allowed to do so, the generation of a representative workload (i.e. millions of users) would have required a lot of resources and thus makes this strategy infeasible. Since these challenges are valid for all large and distributed systems, the experiences we gained are not limited to e-mail systems and thus considered helpful for modelling large and distributed systems in general.

Often, large-scale systems provide business-critical functionality and therefore runtime monitoring of those systems is very detailed. Especially systems that offer services to end users over the internet have significant variations within their workload. Combined with the detailed monitoring data, they allow for a good estimation of the required resource demands and a validation of the model accuracy. In large systems consisting of several subsystems and components, the knowledge about the architecture is often distributed among several people or even departments. This fact significantly increases the required effort to collect information required for modelling the system's architecture. Additionally, the available architecture documentation often turns out to be outdated and the information has to be gathered in interviews. In the case of redundant components and nearly identical servers, the approach to abstract these servers as one abstract resource with multicore scheduling for each resource can significantly reduce the modelling effort without negative influences on the prediction results.

In addition to our original aim to improve the performance monitoring, applying a model-based prediction approach like PCM has further benefits. Since the PCM is an architecture modelling language, it can serve as architecture documentation which counteracts missing or outdated documentation.

Although this prohibits the application of modelling abstraction like the one we presented in our case study, the creation of a useable architecture documentation can balance the additionally required effort. While analysing the system and collecting monitoring data, we performed several plausibility checks of the predicted values. Even early stage prediction models could reveal some misconfigurations of the system. Based on the statistics collected by our analysis tools for log files, we could for example detect a server with inadvertently disabled hyper threading. When modelling a component's behaviour combined with deriving the resource demands, we furthermore identified some potential performance improvements that the developers accepted to be considered in the next version of the component.

In general, the use of the prediction models is not limited to the performance monitoring aspect. They can for instance be used to evaluate i) different deployment variations, ii) architectural changes caused by integrating new components or iii) to predict the system's behaviour in exceptional high workload situations. Furthermore, the performance models can be used to optimise resource sizing and deployment of the components and thereby improve the system's efficiency.

VII. Related Work

Software Performance Engineering (SPE) [3] as introduced by Smith forms the base for several architecture-level performance meta-models supporting model-based performance predictions (surveyed in [14]). Often they are extensions of UML as the de facto standard modelling language for software architectures like the UML SPT profile [4] and its successor the UML MARTE profile [5]. Architecture-level performance models are built during system development and are used at design and deployment time to evaluate system designs and/or predict the system performance for capacity planning purposes. A recent survey of methods for component-based performance-engineering has been published by Koziolek in [15].

Most of these approaches aim on the evaluation of different design options and support manual or automated optimising of the system's architecture and deployment at design time and do not consider an integration of performance predictions into the runtime management of systems. Menasce et al. [16] present an approach supporting the automated assurance of Quality of Service (QoS) levels using performance models. Based on queueing networks different configuration alternatives are evaluated regarding their QoS attribute. However, the approach cannot be applied to detect and identify performance problems and behaviour anomalies at runtime.

The PCM and its prediction capabilities are a central component of the presented case study. Its applicability and accuracy has been demonstrated in several research and industrial case studies (e.g., [17], [18], [19]). However, not any of these case studies modelled a system of similar size and complexity and only focus on the design-time evaluation of systems. In addition to those manually conducted case studies, PCM-based performance predictions have been integrated in an automated process for capacity planning [20] and into the architecture optimisation approach PerOpteryx [21].

In the literature, only few approaches exists that integrate performance prediction techniques into the system monitoring process. R-Capriccio [22] developed at the HP Labs is a tool supporting capacity planning and performance anomaly detection for multi-tier enterprise systems. Queuing models abstracting each server by only one queue are used to predict the expected CPU utilisation. Deviation between predicted and measured CPU utilisation can be detected. However, response times and utilisation of other resources like network or hard disk are not considered. Cherskova et al. [2] present an alternative approach supporting the detection of performance anomalies. Applying regression techniques, the approach enables the detection of unexpected behaviour, which is demonstrated in two case studies. However, the regression techniques requires the collection of a significant amount of data. In the case of a detected anomaly, the approach does not support the operator in analysing and detecting the root cause. Kadirvel et al. [23] present an approach using prediction techniques to enable an automated management of IT systems. Although they use models to predict the potential performance changes induced by reconfigurations, the detection of malfunctions and abnormalities is based on fixed threshold values. However, the approach demonstrates the use of performance models in order to solve performance problems, which could be a potential extension of our approach.

VIII. Conclusion and Outlook

In this paper, we presented an approach to integrate performance prediction techniques into a performance monitoring process. We applied our approach to one of Europe's largest e-mail systems operated by the 1&1 Internet AG running on more than 2,000 servers with more than 40 million users. Using the Palladio Component Model (PCM), we specified architecture-level performance models of the STORE subsystem. The implemented data collection and analysis tool enables a fully automated generation of the workload specification required by the performance predictions that corresponds to the current workload on the system. Based on the prediction results, even small deviations between measured and predicted resource utilisation (i.e. expected for the current workload) can be detected. As our experiences show, such analyses are suitable to identify performance problems. The evaluation of the prediction results shows a prediction error of mostly less than 10%

In addition to the workload-aware performance monitoring, the created models can be used to evaluate the performance of the system and possible system extensions and

compare different design or deployment variations. Based on the prediction results, the sizing of the hardware can be optimised and thereby the efficiency of the whole system improved. Furthermore, during our case study, the performance models helped to locate some configuration problems on the servers and to identify potential performance improvements within the component's implementations.

The case study presented in this paper forms the basis for several areas of future work. Our implementation of the data collection and analysis tool is limited to the 1&1 monitoring system and the specific log file formats. In a next step, we plan to extend our tooling to support further formats and monitoring interfaces. Furthermore, we plan to improve the integration into the PCM tool suite. The presented approach focuses on the detection of potential performance problems only. However, the performance models can also support the automated identification and evaluation of counter actions. Developing such automated and system-aware controlling algorithms is one of the future research directions resulting from the experiences we gained in this case study.

In the following references except [13], the author S. Becker is Steffen Becker and not Stefan Becker, who co-authored this paper.

REFERENCES

[1] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, July 2009.

[2] L. Cherkasova, K. Ozonat, N. Mi, J. Symons, and E. Smirni, "Automated anomaly detection and performance modeling of enterprise applications," *ACM Trans. Comput. Syst.*, vol. 27, pp. 6:1–6:32, Nov 2009.

[3] C. U. Smith, *Performance Engineering of Software Systems*. Addison-Wesley, 1990.

[4] Object Management Group (OMG), "UML Profile for Schedulability, Performance and Time," 2005.

[5] Object Management Group (OMG), "UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)," 2006.

[6] S. Becker, *Coupled Model Transformations for QoS Enabled Component-Based Software Design*, ser. Karlsruhe Series on Software Quality. Universitätsverlag Karlsruhe, 2008, vol. 1.

[7] V. Grassi, R. Mirandola, and A. Sabetta, "From Design to Analysis Models: a Kernel Language for Performance and Reliability Analysis of Component-based Systems," in *WOSP'05*. ACM Press, 2005, pp. 25–36.

[8] J. Happe, H. Koziolek, and R. Reussner, "Facilitating performance predictions using software components," *Software, IEEE*, vol. 28, no. 3, pp. 27 –33, may-june 2011.

[9] H. Koziolek and R. Reussner, "A Model Transformation from the Palladio Component Model to Layered Queueing Networks," in *Proc. of SIPEW '08*, 2008, pp. 58–78.

[10] P. Meier, S. Kounev, and H. Koziolek, "Automated Transformation of Palladio Component Models to Queueing Petri Nets," in *Proc. of MASCOTS'11*, Singapore, 2011.

[11] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, pp. 3–22, 2009.

[12] D. Westermann and J. Happe, "Performance Cockpit: Systematic Measurements and Analyses," in *Proceedings of ICPE'11*. New York, NY, USA: ACM, 2011.

[13] S. Becker, "Performance Modellierung des 1&1 Mail-Systems," Master's thesis, Karlsruhe Institute of Technology (KIT), Germany, 2011.

[14] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-Based Performance Prediction in Software Development: A Survey," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 295–310, 2004.

[15] H. Koziolek, "Performance evaluation of component-based software systems: A survey," *Performance Evaluation*, vol. 67-8, no. 8, pp. 634–658, 2009.

[16] D. A. Menascé, D. Barbará, and R. Dodge, "Preserving QoS of e-commerce sites through self-tuning: a performance model approach," in *Proc. EC'01*. ACM, 2001, pp. 224–234.

[17] N. Huber, S. Becker, C. Rathfelder, J. Schweflinghaus, and R. Reussner, "Performance Modeling in Industry: A Case Study on Storage Virtualization," in *Proc. of ICSE'10*. New York, NY, USA: ACM, 2010, pp. 1–10.

[18] A. Martens, S. Becker, H. Koziolek, and R. Reussner, "An Empirical Investigation of the Applicability of a Component-Based Performance Prediction Method," in *Proceedings of EPEW'08*, vol. 5261, 2008, pp. 17–31.

[19] C. Rathfelder, D. Evans, and S. Kounev, "Predictive Modelling of Peer-to-Peer Event-driven Communication in Component-based Systems," in *Proceedings of EPEW'10*, vol. 6342, 2010, pp. 219–235.

[20] C. Rathfelder, S. Kounev, and D. Evans, "Capacity Planning for Event-based Systems using Automated Performance Predictions," in *Proc. of ASE 2011*, 2011, pp. 352–361.

[21] A. Martens, H. Koziolek, S. Becker, and R. H. Reussner, "Automatically improve software models for performance, reliability and cost using genetic algorithms," in *Proceedings of the first joint WOSP/SIPEW 2010*. ACM, New York, NY, USA, 2010, pp. 105–116.

[22] Q. Zhang, L. Cherkasova, G. Mathews, W. Greene, and E. Smirni, "R-capriccio: a capacity planning and anomaly detection tool for enterprise services with live workloads," in *Proc. of Middleware'07*. Springer, 2007, pp. 244–265.

[23] S. Kadirvel and J. Fortes, "Towards it systems capable of managing their health," in *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*. Springer, 2011, vol. 6662, pp. 77–102.