Simulation of In-Memory Database Workload: Markov Chains versus Relative Invocation Frequency and Equal Probability - A Trade-off between Accuracy and Time

Maximilian Barnert
maximilian.barnert@tum.de
Technical University of Munich
Garching, Germany

Helmut Krcmar
helmut.krcmar@tum.de
Technical University of Munich
Garching, Germany

ABSTRACT

In the last years, performance modeling approaches have been proposed to tackle new concepts for modern In-Memory Database Systems (IMDB). While these approaches model specific performance-relevant aspects, workload representation during performance modeling is considered only marginally. Furthermore, the manual integration of workload into modeling approaches comes along with high effort and requires deep domain-specific knowledge.

This paper presents our experience in representing workload within performance models for IMDB. In particular, we use a Markov chain-based approach to extract and reflect probabilistic user behavior during performance modeling. An automatic model generation process is integrated to simplify and reduce the effort for transferring workload characteristics from traces to performance models.

In an experimental series running analytical and transactional workloads on an IMDB, we compare this approach with two other methods which rely on less granular data to reflect database workload within performance models, namely reproducing the relative invocation frequency of queries and using the same query execution probability. The results reveal a trade-off between accuracy and speed when simulating database workload. Markov chains are the most accurate independent from workload characteristics, but the relative invocation frequency approach is appropriate for scenarios where simulation speed is important.

CCS CONCEPTS

• **General and reference** → **Experimentation**; **Performance**; • **Information systems** → **Database performance evaluation**.

KEYWORDS

Performance Modeling; Workload Representation; Markov Chains; Database Workload; In-Memory Database Systems; SAP HANA

ACM Reference Format:
Maximilian Barnert and Helmut Krcmar. 2021. Simulation of In-Memory Database Workload: Markov Chains versus Relative Invocation Frequency and Equal Probability - A Trade-off between Accuracy and Time. In *Proceedings of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3427921.3450237

ICPE '21, April 19–23, 2021, Virtual Event, France

ACM ISBN 978-1-4503-8194-9/21/04…$15.00
https://doi.org/10.1145/3427921.3450237

1 INTRODUCTION

Modeling performance on In-Memory Database Systems (IMDB) has been a research topic for the last years. Models exist to evaluate performance when scenarios can not be tested on real database systems. While existing approaches mainly focus on reproducing specific aspects (e.g., thread-level operations or memory consumption), authors consider workload extraction and representation only marginally in their modeling approaches. However, the validation of modeling approaches requires numerous experiments changing workload and parameterize. To support this, a key challenge is to reproduce real user behavior sufficiently for varying workload. This is especially true for modern IMDB where performance highly depends on workload composition and intensity impacting the competition for central processing unit (CPU) resources [16]. At the same time, concepts for IMDB enable to run workload with different characteristics on the same database system [9, 15]. Challenges are intensified by cloud services such as database-as-a-service (DAAS) where database administrators have not insight into the overlying application logic as driver of the workload [21]. However, the representation of workload containing hundred or thousand database requests requires deep domain knowledge and high effort due to many manual activities during performance modeling.

In order to address these challenges, we use an automatic model creation process in this paper to predict performance for database workload on IMDB. It simplifies and reduces the effort for integrating database workload specifications into performance models. Figure 1 shows an overview of our approach. Modeling and simulating performance on IMDB builds on our previous work [3]. Proprietary database traces are translated to OPEN.xtrace [2, 14] to receive run time data in a tool-independent format. The data in OPEN.xtrace are used to generate architecture-level performance models based on the Palladio Component Model (PCM). In this paper, we extend the resulting resource-oriented models by automatically extracting and transforming workload specifications during model generation. In doing so, we use a Markov chain-based approach to extract and reflect probabilistic user behavior. It builds on existing approaches [22–25] for specifying and extracting workload for session-based application systems. We adapt them in this work to be applicable for database systems.

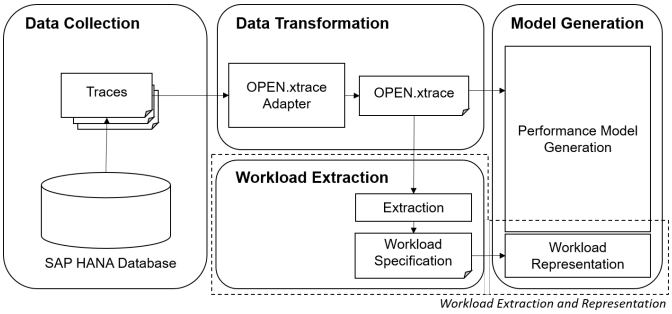


Figure 1: Overview of the automatic performance model generation approach

In an experimental study using analytical and transactional workload on SAP HANA [19], we compare this approach with two other methods which represent database workload at a lower level of detail and therefore require less granular data from traces. The first method reflects database workload by using the relative invocation frequency of statements. The second method executes the queries with the same probability. During evaluation, measurements and simulation results are compared with respect to the accuracy for predicting performance, namely throughput. Besides, we include an analysis of the accuracy in reflecting the invocation frequencies of statements and consider up scaling scenarios, as workload composition and intensity impact the overall performance on IMDB [16]. Furthermore, we look at the time needed to solve the generated models.

The remainder of this paper is organized as follows. Section 2 describes related work. Our modeling and simulation approach is outlined in Section 3. In Section 4, we evaluate the accuracy of the different workload representation techniques. Finally, we conclude our work and describe future work in Section 5.

2 RELATED WORK

The related work in this paper can be separated into two groups.

The first group focuses on modeling workload using Markov chains to represent real user behavior. For application systems, several approaches specify user behavior in Markov chains while adding probability and think times to represent user interactions. First, Menascé et al. [10] present an approach extracting Customer Behavior Model Graphs (CBMGs) from HTTP-logs. Ruffo et al. [17] automatically create behavior models from log files captured for a web application. These approaches do not reflect dependencies between requests. To close this gap, extended finite state machines (EFSMs) are used by Shams et al. [20] to describe a valid sequence of user requests within a session. As part of the WESSBAS approach, authors [23, 25] combine the concepts of CBMGs and EFSMs in a single workload model to extract probabilistic user behavior from session logs of an application system. The concept of Guards and Actions (GaA) is used to ensure valid sequences of user requests. The authors integrate Markov chains into performance modeling in order to reproduce probabilistic user behavior during simulation [25]. In particular, extracted workload specifications including the behavior models specified as Markov chains are translated to PCM in order to receive performance models representing workload for application systems. In doing so, workload specifications are extracted from session logs to reproduce real user workload. While the authors concentrate on application systems, we extend their original approach extracting workload specifications and creating performance model for database systems in this paper.

For database systems, Sapia [18] uses Markov chains to reflect query sequences in order to enhance predictive caching techniques on analytical database systems. The work focuses on analytical tasks based on multidimensional operations, why a high-level abstraction of queries is used considering dimensions, query measures and result granularity. The author uses two different Markov models to forecasts future query processing. The first Markov model (structural prediction model) reflects changes between dimensions for subsequent queries of an analysis process. The second Markov model (value based prediction model) represents navigation paths within a dimension. In contrast to our modeling approach, this work strongly focuses on analytical tasks performed on a multidimensional cube schema, which limits it applicability for other scenarios. Further, Tran et al. [21] present a general approach using Markov chains to model workload on database systems. The approach extracts workload models from the workload capture service Oracle Database Replay. The model reflects the main characteristics of a workload, which allows database administrators and application developers to analyze real database workload and identify frequent sequences. Both works do not consider integrating workload behavior in performance modeling for database systems.

The second group concentrates on performance model generation for IMDB. In this group, Kraft et al. [8] use queuing network models to predict performance (i.e., response time, throughput and CPU utilization) on IMDB. The model reflects thread-level parallelism caused by concurrent jobs processed during query execution. The authors apply measured traces to receive think times, population sizes and fork sizes for queries. The performance model is used to evaluate different admission control strategies. In doing so, the authors assign a subset of the 22 TPC benchmark H (TPC-H) queries to three classes reflecting different utilization of parallel database operators (i.e., low, medium and heavy). The execution probability for the classes is defined manually in a probability vector. This approach is reused by Molka et al. [13] to model tenant interference on IMDB. Another work is presented by Molka and Casale [11] comparing simulation techniques with response surface models to predict performance for multi-tenant scenarios on IMDB. During evaluation, the authors again use the 22 analytical queries of the TPC-H benchmark. Workload is reflected in the performance model using a class-switching approach to reproduce the recurrent submitting of a permuted sequence of all 22 query classes of the TPC-H workload. The approach focuses on the reflection of this repeating submission of the 22 queries. It excludes basic concepts for transforming and integrating complex user behavior of arbitrary workload. Molka and Casale [12] extend this performance modeling approach to predict main memory consumption on IMDB. For evaluation, the TPC-H workload is applied again. For modeling the workload, a throughput formula is predefined for each query. It is based on the number of concurrent users that execute the query, response times and think times. Another work

in this group is presented by Wust et al. [26]. They use a queuing network model to reflect the impact of intra-query parallelism, i.e., concurrent operations to boost query processing, on another query's run time. However, the applied performance model only reflects the execution of two parallel queries.

3 MODELING APPROACH

The performance modeling approach in this work builds on our previous work [3] for modeling and simulating the performance on IMDB. We reuse concepts for modeling IMDB from this previous work [3], but integrate approaches to extract and transform workload characteristics from workload traces. As the modeling process is based on PCM for generating architecture-level performance models, Section 3.1 outlines first the concepts of PCM. To make this work self-contained, a brief overview of the model generation process presented in our previous work [3] is given in Section 3.2. We outline changes more in detail. Finally, Section 3.3 describes the three approaches for representing database workload in performance models used in this work.

3.1 The Palladio Component Model

PCM is a modeling language which supports to predict performance (e.g., response time and throughput) in the domain of software systems [4]. The representation of a software system is divided into five models. Interfaces and components are represented in the repository model. Relationships between components are reflected by providing and requiring interfaces. For a provided relationship, the component implements signatures of interfaces (i.e, operations). Signatures are modeled as service effect specifications (SEFF) in the repository model. It allows to specify resource demands and external calls of other components' operations. The system model assembles the components. The resource environment model specifies containers representing associated resources (i.e., hardware or network). Assembled components are linked to resource containers in the allocation model. The usage model allows to reflect user behavior.

3.2 Modeling In-Memory Database Systems

In our previous work [3], we present a performance modeling approach for IMDB consisting of three steps (fig. 1): (1) data collection, (2) data transformation and (3) model generation. We reuse concepts of this approach for generating the system-specific parts of the database system. First, the run time data needed to create the performance models is collected during the execution of the queries. Adapters are used to transform these proprietary traces to OPEN.xtrace [2, 14] to receive run time data in a common representation. While we use query execution plans to reflect the internal query processing in our previous work, we only use query response times in this paper to evaluate the workload representation approaches. In the model generation step, the run time data represented in OPEN.xtrace is used to create the system-specific performance models based on the PCM meta model.

The database system is represented in the repository model using a single basic component and the corresponding interface. An operation is added to this database component for each distinct statement extracted from the trace. As stated above, only a statement's response time is reflected in this work. Therefore, each SEFF of a component's operation begins with an *StartAction*. It is followed by an *InternalAction* element containing a single delay resource demand reflecting a statement's response time. The delay's value is calculated using the mean response time for all executions of this statement captured in the trace. Each SEFF ends with an *StopAction*.

The other PCM models are generated automatically. The system model is created using the above described repository model. The resource environment specifies the delay demand used in the SEFF. The resource demands in the resource environment are mapped within the allocation model to the system model.

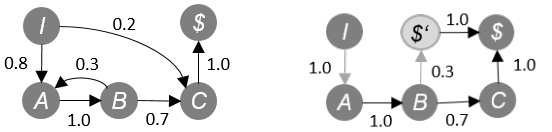
3.3 Modeling Database Workload

This section presents the three approaches which are used in this paper to extract and represent database workload during performance modeling. Each method differs in the level of detail in reflecting the real user behavior. Section 3.3.1 describes the Markov chain-based approach to reproduce probabilistic user behavior. In Section 3.3.2, the relative invocation frequency-based approach is presented. The third technique is described in Section 3.3.3 executing statements with the same probability.

3.3.1 Markov Chain-based Approach. The first approach is based on Markov chains to reproduce probabilistic user behavior within database workload. It builds on previous works [22–25] specifying and extracting workload for session-based application systems (WESSBAS). This workload modeling formalism defines several components to reflect real workload for application systems [25]. The number of concurrent users is captured by the *workload intensity*. The sequence of service invocations is specified as a two-layered hierarchical finite state machine (FSM) in the *application model*. It consists of a session and protocol layer [23]. *Behavior models* are specified as Markov chains to provide a probabilistic representation of invoked services and think times between services. The domain specific language WESSBAS-DSL models these workload specifications [25]. It is used in the approach of Vögele et al. [25] to transform workload specifications of WESBASS to PCM performance models. It is extended in this paper by transforming it to the area of database systems. We exclude the concept called GaA, which allows to reflect results-driven dependencies between requests, because the necessary information is not available in database traces.

To support database systems, we extend the application model of WESSBAS. The element *SQLRequest* is added to enable modeling of database requests. In addition, the class *SQLProtocolLayerEFSMGenerator* is integrated to generate Protocol Layer FSMs for such database requests. As WESSBAS extracts workload specifications from session logs captured on application systems, we need to translate the OPEN.xtrace data to session logs to integrate our approach. Further, WESSBAS uses clustering methods to create separate behavior models for each transaction identified in the session logs. For database systems, we exclude clustering as database workload can be represented in a single behavior model.

In the next step, behavior models of WESSBAS-DSL are transferred to performance models. Due to restrictions of the PCM's simulation engine Simulation for Component-Based Systems (SimuCom)



(a) Original Markov chain **(b) Modified Markov chain**

Figure 2: Modifications to behavior models

to solve performance models representing Markov chains with activity cycles [1], the behavior models need to be adapted for database workload. An exemplary Markov chain is shown in Figure 2a where an activity cycle exists. The cycle is created following the backward transition from Markov state B to A. To enable the simulation of such performance models as described in our previous paper [1], the original behavior model needs to be modified (figure 2b). First, the Markov state with the highest probability for being entered by the initial state *$* is determined as unique starting node of the behavior model, i.e., state A in our example. Next, an artificial Markov state *$'* is inserted into the behavior model. It is linked to the final state *$* of the model with a probability of 100 %. Afterwards, backward transitions to our newly defined unique entering state (i.e., state A) are replaced by an edge to *$'*. When we now enter the original start node (i.e., state B) of the backward transition, we immediately leave the Markov chain following the path from state *$'* to *$*. Afterwards, it is instantly re-entered at the original start node (e.g., state A) of our backward transition.

Next, the workload specifications (including the behavior models) are attached to the system-specific performance model (section 3.2). Again we reuse concepts of Vögele et al. [25], but adapt elements to support database workload. The authors utilize the PCM repository model to reflect the Markov chains. This approach violates the separation of PCM models [4]. However, it is valid as the PCM usage model does not support representing incoming and outgoing edges [25]. For each behavior model, a basic component with the corresponding interface is added to the repository model. We add just one basic component to the repository model as we use a single behavior model to reflect database workload. An operation as SEFF is added to the component for each Markov state within the behavior model. A probabilistic branch in each SEFF represents transitions between Markov states. A branch incorporates three elements. The transition's think time is modeled as *InternalAction* with a delay resource using the mean value. It is followed by an *ExternalActionCall* linked to the SEFF of the basic component (i.e., the database system) in the system-specific performance model. This integrates the workload representation into the system-specific PCM models (Section 3.2). A second *ExternalActionCall* reflects the transition to another Markov state. In the behavior model two artificial states tag the initial *I* and final state *$* of the Markov chain. The transition to the final state *$* is reflected by another branch in the SEFF. It contains a *StartAction* and *StopAction*. The initial state *I* is represented as a separate operation *INITIAL* in the basic component of the behavior model. It's SEFF consists of an *ExternalActionCall* linked to the start node of the Markov chain (i.e., Markov state A in our example). A closed workload defined in the PCM usage model calls the operator *INITIAL*. This initiates the workload execution.

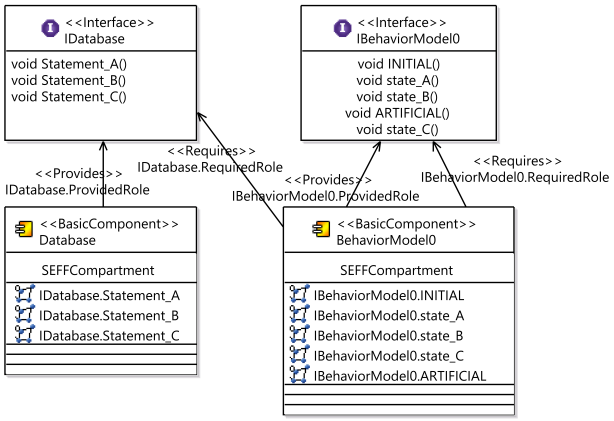


Figure 3: Repository model containing components to represent the database system and behavior model

As stated above, repository models representing backward transitions in Markov chains can not be solved by PCM's simulation engine. Therefore, we modified the behavior model during workload extraction as described above. During performance model creation, we also need to adapt the repository model as described in our previous work [1]. An operation named *ARTIFICIAL* is added to the behavior model's basic component. It represents the artificial state *$'* in the behavior model. Additionally, *ExternalActionCalls* reflecting the transition to another Markov state are surrounded by *ForkedBehaviors*. This allows writing concurrent measurements when running through an activity cycle during simulation. An exemplary repository model is illustrated in figure 3. It results from performing the described approach for the behavior model presented in figure 2b. Additionally, figure 4 shows the SEFF *state_B* representing Markov state B.

3.3.2 Relative Frequency-based Approach. Our second approach uses the relative invocation frequency of statements to reflect the user behavior on IMDB. The invocation frequency is calculated based on the query execution counts stored in OPEN.xtrace. The workload is represented in the PCM usage model. It reflects a closed workload without any think time. The workload's population, i.e., the number of concurrent database users, is derived from the session numbers extracted from OPEN.xtrace. The usage model contains a single probabilistic branch with one transition for each distinct statement. This enables selecting statements on a probabilistic basis, e.g., *StatementA* is processed in 20 % of all cases, while *StatementB* is called for 80 % of the cases. A statement's invocation frequency defines the probability for entering the corresponding transition during simulation. The execution of a statement is reflected by an *EntryLevelSystemCall* in the corresponding branch transition. A link between the *EntryLevelSystemCall* element and the corresponding query's signature in the repository model connects the workload representation with the system-specific parts of the performance model (Section 3.2).

3.3.3 Equal Probability-based Approach. The third approach simply picks statements with the same probability from a predefined

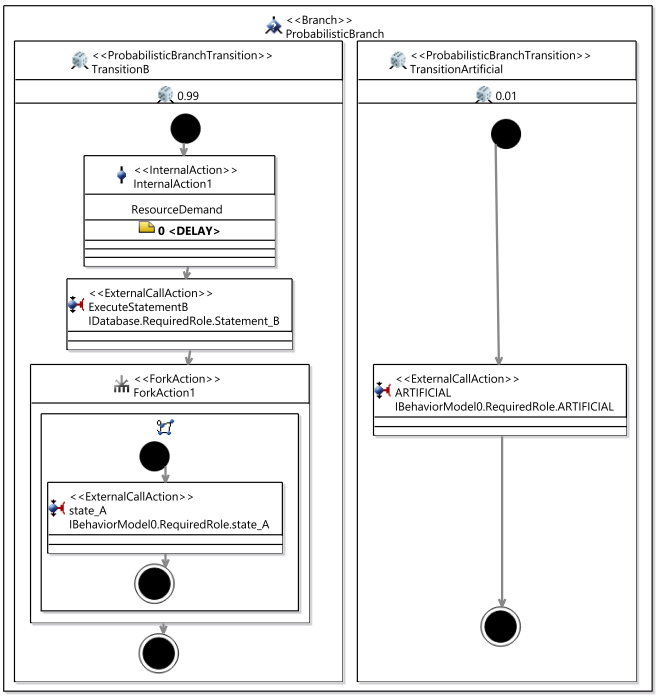


Figure 4: SEFF model representing Markov state B

query set. As with the *relative frequency-based* approach, the PCM usage model reflects user behavior based on a closed-loop workload. The population is derived from the OPEN.xtrace data based on the number of different sessions. A single probabilistic branch is added containing a transition for each distinct statement in the trace. In contrast to the *relative frequency-based* technique, the probability for entering a specific transition is the same. This approach reflects an equal distribution of all statements. The execution of a statement is represented by *EntryLevelSystemCalls* in the branch linked to the statement's SEFF in the system-specific PCM repository model.

4 EVALUATION

To evaluate the accuracy of the workload representation techniques, we apply our performance model generation approach to the CH-benCHmark [5]. In Section 4.1, the research questions and methodology is described. Section 4.2 presents the system under test (SUT). Section 4.3 details the utilized database workload. Finally, we compare measured to simulated results in Section 4.4.

4.1 Research Questions and Methodology

We address the following research questions during evaluation:

RQ1: How accurately do Markov chains predict workload performance on IMDB compared to the relative frequency- and the equal probability-based approach? Our first question aims at comparing the accuracy for predicting the performance on IMDB while varying the level of detail when reproducing real user behavior. The accuracy is evaluated using a performance-based metric, namely the throughput.

RQ2: Does the prediction accuracy depend on workload characteristics? Workload on database systems is traditionally categorized as being analytical or transactional. Both types differ in characteristics. For evaluating the accuracy using different workload types, we analyze the simulated throughput for analytical and transactional workload.

RQ3: How representative is workload simulated by the Markov chain-based approach compared with the relative frequency- and equal probability-based method? The composition of a workload impacts the performance on IMDB [16]. In worst case scenarios, blocking situations prevent queries from being executed in time. Therefore, the simulated query mixture is analyzed using a session-based metric, namely the invocation frequency of database requests.

RQ4: What is the impact of workload intensity on the prediction accuracy? The parallel processing of database requests influence the overall performance on IMDB [16]. Consequently, we include the evaluation of the prediction accuracy for up scaling scenarios.

RQ5: What impact does the level of detail used in representing real user behavior has on the simulation duration? The fifth question aims at exposing time costs, which result from using a higher level of detail when reproducing real user behavior within database workload. For this purpose, we look at the time needed to solve the performance models for the different workload representation techniques.

During evaluation, we compare measurements on our test environment with simulation results for using the different workload representation approaches. Our evaluation consists of an experiment series running the CH-benCHmark against SAP HANA. The benchmark driver stores information about sessions, queries and response times for each run. This allows modeling database workload as it provides query response times incorporating the processing time on the database and time for fetching the results. The CH-benCHmark is parameterized using 150 warehouses, a warm-up time of 5 minutes and a test phase of 10 minutes. We run the benchmark separately for analytical and transactional workload increasing the number of users from 1 (single session) to 8 respectively 16. We repeat the benchmark execution 5 times for each scenario. To compare throughput for analytical and transactional workload, we use the number of successfully completed queries per hour (Qph) (equation 1).

$$Qph = \frac{\text{Successfully Completed Queries}}{\text{Test Phase Duration (s)}} * 3600 \quad (1)$$

Our performance model generation approach is implemented by a software prototype. It creates performance models based on the collected data. Separate models are created for each workload representation technique. The generated performance models are automatically solved by the prototype applying PCM's simulation engine SimuCom. The simulation environment is based on hardware containing an Intel Core i5-7300U CPU with 2.71 GHz and 8 GB random access memory (RAM). Model generation and simulation are repeated 10 times for each measurement and workload representation method.

4.2 System Under Test

The SUT and the benchmark driver are deployed on separate virtual machines. The corresponding hardware environment consists of an IBM Power E870 server with four sockets and 40 physical CPU cores at 4.19 GHz, eight-thread SMT mode (SMT-8) as well as 4 TB RAM in total. For virtualization, the SUT and the benchmark driver are deployed on logical partitions (LPAR). The SUT is equipped with 2 cores and 250 GB RAM. It runs a SAP HANA 2.0 with Support Package Stack (SPS) 2 and a tenant database. The benchmark is executed on a virtual machine with 1 core and 8 GB RAM.

4.3 Workload Description

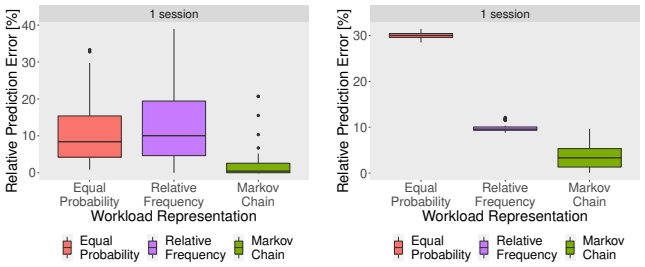
We use the CH-benCHmark [5] to generate analytical and transactional workload on SAP HANA. The benchmark consolidates the two industry-standard TPC benchmark C (TPC-C) [7] and TPC-H [6] and provides an unified data structure based on the TPC-C entities and relationships. To generate analytical workload, the benchmark executes adapted versions of the 22 read-only TPC-H queries. Following the TPC specifications, each user runs a random permutation of all 22 analytical queries against the database system. The five TPC-C transactions New-Order, Payment, Order-Status, Delivery and Stock-Level are used by the CH-benCHmark without modifications [5]. The two transactions New-Order and Payment are executed with a probability of 44 %. A relative frequency of 4 % is set for the other transactions. Each transaction executes a subset of queries to perform specific tasks. In total, the transactional workload consists of 41 queries performing read- or write-tasks.

For our experiments, we use the CH-benCHmark implementation provided by Psaroudakis et al. [16]. This C++ program uses an ODBC interface to access the DBMS. It creates a data set, initializes the database for a given number of warehouses and executes both workloads. By using the CH-benCHmark the experiments become reproducible. The original software implementation executes the 22 TPC-H queries in a sequential manner. To fit with the TPC-H specification, we change this behavior using a random permutation of all 22 statements for each user. Because we are interested in the number of successfully executed queries and the query response times, we modified the original implementation by writing this information to the file system. The workload traces contain query start and end time as well as the corresponding session.

4.4 Results

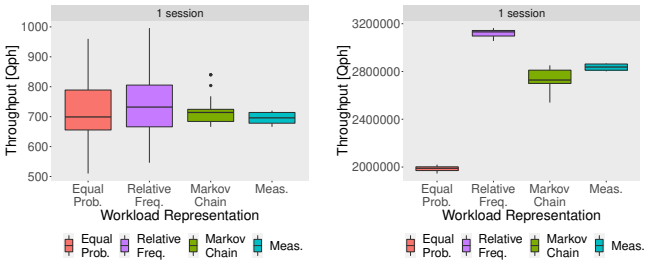
This section presents the results of our evaluation. In Section 4.4.1, we investigate research question RQ1 and RQ2. Sections 4.4.2 deals with RQ3. Section 4.4.3 answers RQ4 and Section 4.4.4 inquires RQ5.

4.4.1 Performance Prediction Accuracy. First, we analyze the impact of workload representation on the performance predicting accuracy (RQ1). Furthermore, results for analytical and transactional workload are compared (RQ2). The resulting relative prediction errors are presented in figure 5. When simulating analytical workload, we obtain a median value of 8.41 % (mean: 11.23 %) for the *equal probability-based* (EP) method, 12.57 % (mean: 10.04 %) for the *relative frequency-based* (RF) method and 0.43 % (mean: 2.59 %)



(a) Analytical workload (b) Transactional workload

Figure 5: Relative prediction errors for throughput comparing measurements and simulation results (single session scenario)

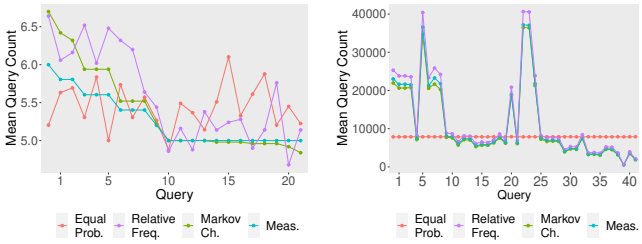


(a) Analytical workload (b) Transactional workload

Figure 6: Throughput in measurements and simulation results (single session scenario)

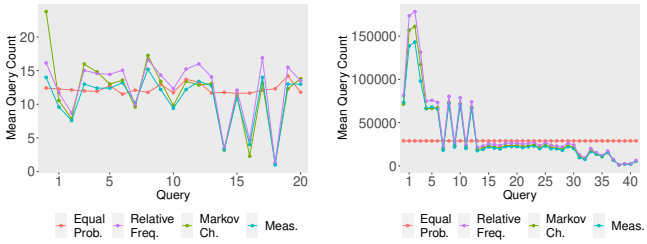
for the *Markov chain-based* (MC) method. The simulation of transactional scenarios results in prediction errors (median) of 30.02 % (mean: 29.99 %) for EP, 9.58 % (mean: 9.97 %) for RF and 3.33 % (mean: 3.80 %) for MC. As shown in figure 6, all representation techniques overestimate analytical throughput. We observe the same behavior for RF in transactional scenarios. In contrast, MC tends to estimate throughput very similar to measurements. RP underestimates throughput to a high degree. In conclusion, the simulation results show that Markov chains reproduce user behavior most precisely for both workload types in single session scenarios. It returns relative prediction errors below 4 %. Besides, all methods return acceptable prediction errors below 13 % for simulating analytical workload. Very large error rates can be observed for EP applied in transactional scenarios.

4.4.2 Workload Representativeness. In this section, we investigate research question RQ3 analyzing the simulated invocation frequency of statements. Looking only at the findings for throughput in section. 4.4.1, one may conclude that all approaches reproduce analytical workload adequately. However, we notice evident differences in simulating the frequencies of statement executions (fig. 7a). Simulation results for MC are quite similar to measurements and run almost in parallel. We observe a higher fluctuation for RF and EP. A parallel trend is not obvious. For transactional workload containing a higher number of requests, the fluctuation almost disappears. As presented in figure 7b, results for transaction workload



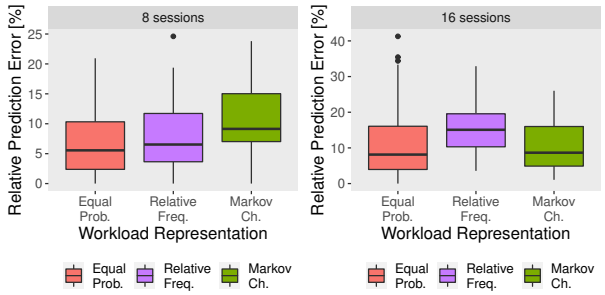
(a) Analytical workload (b) Transactional workload

Figure 7: Measured and simulated query invocation frequencies (single session scenario)

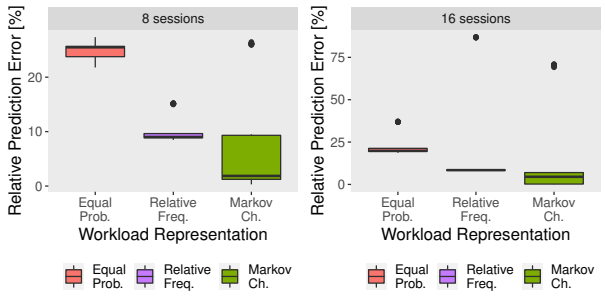


(a) Analytical workload (b) Transactional workload

Figure 9: Measured and simulated query invocation frequencies (16 parallel sessions)



(a) Analytical workload



(b) Transactional workload

Figure 8: Relative prediction errors simulating throughput in up scaling scenarios (8 and 16 sessions)

appear almost congruent when using the methods MC and RF. As we expected, using equal probability for executing the statements results in the same invocation frequency for all queries. Therefore, we conclude that Markov chains reflect the workload composition most accurately for both workload types in single session scenarios. In general, RF and MC show better results for transactional workload.

4.4.3 Changing Workload Intensity. Next, we analyze the impact of session concurrency on the prediction accuracy (RQ4). In particular, we compare measurement to simulation results for running 8 and 16 users in parallel. The resulting prediction errors are presented in figure 8. Simulating analytical throughput with 8 users (fig. 8a) returns a median value of 15.08 % (mean: 15.6 %) for EP, 8.13 % (mean: 11.46 %) for RF and 8.66 % (mean: 10.29 %) for MC. Increasing the number of parallel sessions to 16 results in median error rates of 8.13 % (mean: 11.46 %) for EP, 15.1 % (mean: 15.6 %) for RF and 10.29 % (16.0 %) for MC. For transactional scenarios with 8 users, we observe error rates of 25.49 % (mean: 24.84 %) for EP, 9.02 % (mean: 10.25 %) for RF and 1.89 % (mean: 7.89 %) for MC. Running 16 parallel users provides prediction errors of 19.8 % (mean: 23.26 %) for EP, 8.42 % (mean: 24.05 %) for RF and 4.49 % (mean: 16.38 %) for MC. Besides, the results show median error rates between 8 % and 16 % for EP and RF when changing the workload intensity in analytical scenarios. It confirms our observations for single session scenarios in section 4.4.1. MC seems to loose accuracy when increasing the number of parallel users and we receive higher error rates over 10 % for 16 sessions. For transactional workload, MC still outperforms the other approaches (errors below 5 %). Nevertheless, RF also provides acceptable accuracy for 8 and 16 sessions with error rates around 9 %. In contrast, EP returns less precise results starting at about 20 %.

When looking at the workload composition, we notice a better accuracy for RF in up scaling scenarios. The fluctuation of the data disappears and the invocation frequency is reflected quite acceptable for both workload types (fig. 9). In contrast, EP moves toward an equal distribution for all statements. There are no changes in the prediction behavior for MC. Invocation frequencies in the simulation results are still very similar to measurements.

The results allow the conclusion that MC simulates single and up scaling scenarios very accurately independent from workload characteristics. It reflects the workload composition most precisely. RF also predicts transactional and analytical throughput acceptable returning low prediction errors. The simulated workload composition becomes more accurate for high load when using RF.

4.4.4 Simulation Duration. Finally, we analyze the impact of detail level on simulation times (RQ5). Figure 10 shows the times needed to solve the generated performance models depending on workload type and representation technique. For simulating analytical workload, we observe comparable mean times of 9.36 seconds for RF, 9.62 seconds for EP and 11.42 seconds for MC in single session scenarios. We note similar ranges for up scaling scenarios. Solving models which represent transactional workload come with higher run times. This is due to different workload characteristics. Analytical workload consists of long running requests in contrast to mostly short running queries in transactional workload. This results in higher number of query executions as shown in figures 7 and 9, which increases the effort for model simulation. Further

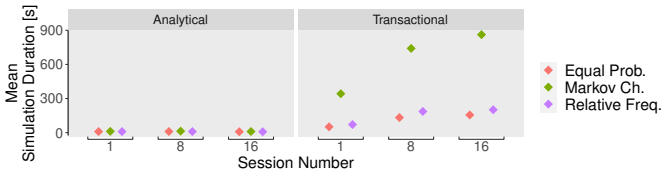


Figure 10: Simulation duration depending on workload type and representation approach

effort is created by a higher level of detail in reproducing real user behavior. MC reflects probabilistic behavior within transactional workload, which brings complexity into the corresponding models resulting in longer run times. In contrast, models based on RF and EP are solved faster for transactional workload. A growing number of sessions reinforces this. At the same time, our results (sections 4.4.1, 4.4.2 and 4.4.3) reveal lower accuracy for EP and RF compared to MC. Simply put, accuracy and simulation of higher load come at the expense of speed. However, unlike EP, the results for simulating throughput and workload composition are still in an acceptable range when using RF. Therefore, we consider RF appropriate for scenarios where simulation speed is important.

5 CONCLUSION AND FUTURE WORK

In this paper, we compare three approaches to represent database workload during performance modeling for IMDB. In addition, we propose an automatic performance model generation process for extracting workload characteristics and creating performance models for these representation techniques. It simplifies and reduces the effort creating component-based performance models for changing workloads. The evaluation of the workload representation techniques allows to understand what technique performs best for specific tasks. The results show that Markov chains are a very accurate technique to simulate analytical and transaction workload on IMDB (prediction errors below 5 % for single session scenarios). This makes Markov chains the best choice for areas where high prediction accuracy is mandatory. However, solving the resulting performance models can take a long time when high loads are to be simulated. The approach based on the relative invocation frequency of statements is an adequate alternative. It returns acceptable prediction errors between 8 % and 16 % for both workload types on IMDB. The generated models are solved faster which makes it the better choice when speed is important, e.g., almost real time performance prediction for IMDB.

Future work includes extensions to our performance model creation approach reflecting resource consumption, e.g., CPU. In addition, we plan to evaluate the representation techniques for other workloads.

REFERENCES

[1] Maximilian Barnert and Helmut Krcmar. 2020. Supporting Backward Transitions within Markov Chains when Modeling Complex User Behavior in the Palladio Component Model. In *Symposium on Software Performance*.

[2] Maximilian Barnert, Adrian Streitz, Harald Kienegger, and Helmut Krcmar. 2017. Converting Traces of In-Memory Database Systems to OPEN.XTRACE on the Example of SAP HANA. In *Symposium on Software Performance*.

[3] Maximilian Barnert, Adrian Streitz, Johannes Rank, Harald Kienegger, and Helmut Krcmar. 2019. Using OPEN. xtrace and Architecture-Level Models to Predict Workload Performance on In-Memory Database Systems. In *Symposium on Software Performance*.

[4] Steffen Becker, Heiko Koziolek, and Ralf Reussner. 2009. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software* 82, 1 (2009), 3–22.

[5] Richard Cole, Florian Funke, Leo Giakoumakis, Wey Guy, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Meikel Poess, Kai-Uwe Sattler, Michael Seibold, Eric Simon, and Florian Waas. 2011. The mixed workload CH-benCHmark. In *Proceedings of the Fourth International Workshop on Testing Database Systems*. ACM, 1–6.

[6] Transaction Processing Performance Council. 2010. *TPC Benchmark H (Standard Specification)*. Report.

[7] Transaction Processing Performance Council. 2017. *TPC Benchmark C (Standard Specification)*. Report.

[8] Stephan Kraft, Giuliano Casale, Alin Jula, Peter Kilpatrick, and Des Greer. 2012. Wiq: work-intensive query scheduling for in-memory database systems. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 33–40.

[9] Norman May, Alexander Böhm, and Wolfgang Lehner. 2017. SAP HANA–The Evolution of an In-Memory DBMS from Pure OLAP Processing Towards Mixed Workloads. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)* (2017).

[10] Daniel A Menascé, Virgilio AF Almeida, Rodrigo Fonseca, and Marco A Mendes. 1999. A methodology for workload characterization of e-commerce sites. In *Proceedings of the 1st ACM conference on Electronic commerce*. 119–128.

[11] Karsten Molka and Giuliano Casale. 2015. Experiments or simulation? A characterization of evaluation methods for in-memory databases. In *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 201–209.

[12] K. Molka and G. Casale. 2016. Efficient Memory Occupancy Models for In-memory Databases. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 430–432.

[13] Karsten Molka, Giuliano Casale, Thomas Molka, and Laura Moore. 2014. Memory-aware sizing for in-memory databases. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 1–9.

[14] Dušan Okanović, André van Hoorn, Christoph Heger, Alexander Wert, and Stefan Siegl. 2016. *Towards Performance Tooling Interoperability: An Open Format for Representing Execution Traces*. Springer International Publishing, Cham, 94–108.

[15] Hasso Plattner. 2009. A common database approach for OLTP and OLAP using an in-memory column database. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 1–2.

[16] Iraklis Psaroudakis, Florian Wolf, Norman May, Thomas Neumann, Alexander Böhm, Anastasia Ailamaki, and Kai-Uwe Sattler. 2014. Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 97–112.

[17] Giancarlo Ruffo, Rossano Schifanella, Matteo Sereno, and Roberto Politi. 2004. Walty: a user behavior tailored tool for evaluating web application performance. In *Third IEEE International Symposium on Network Computing and Applications, 2004.(NCA 2004). Proceedings*. IEEE, 77–86.

[18] Carsten Sapia. 2000. PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems. In *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery*. Springer-Verlag, 224–233.

[19] SAP SE. 2018. *SAP HANA Administration Guide*. Report.

[20] Mahnaz Shams, Diwakar Krishnamurthy, and Behrouz Far. 2006. A model-based approach for testing the performance of web applications. In *Proceedings of the 3rd international workshop on Software quality assurance*. 54–61.

[21] Quoc Trung Tran, Konstantinos Morfonios, and Neoklis Polyzotis. 2015. Oracle Workload Intelligence. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 1669–1681. https://doi.org/10.1145/2723372.2742791

[22] André Van Hoorn, Matthias Rohr, and Wilhelm Hasselbring. 2008. Generating probabilistic and intensity-varying workload for web-based software systems. In *SPEC International Performance Evaluation Workshop*. Springer, 124–143.

[23] André van Hoorn, Christian Vögele, Eike Schulz, Wilhelm Hasselbring, and Helmut Krcmar. 2014. Automatic extraction of probabilistic workload specifications for load testing session-based application systems. In *Porceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2014)*.

[24] Christian Vögele, Robert Heinrich, Robert Heilein, André van Hoorn, and Helmut Krcmar. 2015. Modeling Complex User Behavior with the Palladio Component Model. In *Softwaretechnik-Trends 35(3)*.

[25] Christian Vögele, André van Hoorn, Eike Schulz, Wilhelm Hasselbring, and Helmut Krcmar. 2018. WESSBAS: extraction of probabilistic workload specifications for load testing and performance prediction—a model-driven approach for session-based application systems. *Software and Systems Modeling* 17, 2 (2018), 443–477.

[26] Johannes Wust, Martin Grund, Kai Hoewelmeyer, David Schwalb, and Hasso Plattner. 2014. *Concurrent Execution of Mixed Enterprise Workloads on In-Memory Databases*. Springer International Publishing, Cham, 126–140.