Using OPEN.xtrace and Architecture-Level Models to Predict Workload Performance on In-Memory Database Systems

Maximilian Barnert, Adrian Streitz, Johannes Rank, Harald Kienegger, Helmut Krcmar
{maximilian.barnert, adrian.streitz, johannes.rank, harald.kienegger, krcmar}@in.tum.de
Chair for Information Systems
Technical University of Munich, 85748 Garching, Germany

Abstract

In-Memory Database Systems (IMDB) come into operation on highly dynamic on-premise and cloud environments. Existing approaches use classical modeling notations such as queuing network models (QN) to reflect performance on IMDB. Changes to workload or hardware come along with a recreation of entire models. At the same time, new paradigms for IMDB increase parallelism within database workload, which intensifies the effort to create and parameterize models. To simplify and reduce the effort for researchers and practitioners to model workload performance on IMDB, we propose the use of architecture level performance models and present a model creation process, which transforms database traces of SAP HANA to the Palladio Component Model (PCM). We evaluate our approach based on experiments using analytical workload. We receive prediction errors for response time and throughput below 4 %.

1 Introduction

New paradigms on In-Memory Database Systems (IMDB) shift workload characteristics on traditional Enterprise Resource Planning (ERP) systems. Adjustments to traditional workload come along with dynamic placement on hardware within on-premise or cloud environments. Available approaches for modeling performance on IMDB [2, 4, 7, 8, 9] require detailed domain knowledge and high effort to extract run time data as well as create and parameterize performance models. The use of classical modeling notations such as queuing network models (QN) or being of empirical nature [1] implies a rebuilding of the entire model when changing one aspect (e.g., workload or hardware), since all relevant aspects are included in a single model. Especially modeling workload performance is challenging for IMDB due to a large number of simultaneous query executions, complex query execution plans and the utilization of many database operators. We consider architecture level performance models to be advantageous for reflecting workload performance on modern IMDB due to a separate specification of workload, software architecture and hardware. When single aspects (e.g., workload) change, only affected submodels need to be adapted. To simplify and reduce the effort for generating models, we present a performance model creation approach on the example of SAP HANA. The Palladio Component Model (PCM) is used to generate architecture level performance models. Query execution plans captured by proprietary database traces are extracted to parameterize the models. To apply a common representation of query execution plans in our approach, we translate the traces to the OPEN.xtrace format [10].

The remainder of the paper is structured as follows: Section 2 presents our performance model generation approach. In Section 3, we evaluate our approach. Section 4 gives an overview over related work. The conclusion and future work are drawn in Section 5.

2 Model Generation Approach

Our performance model generation approach consists of three steps, which are shown in Figure 1: (1) data collection, (2) data transformation and (3) model generation. We implement a prototype of our approach as plugin of the Palladio bench, which uses traces as input, transforms them and creates the models.

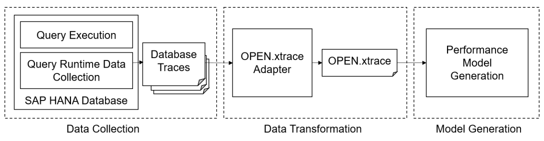

Figure 1: Performance Model Generation Process

Data Collection. In the first step, the data to create the performance models is collected during the execution of the queries. We focus on query parallelism in this work as it is a crucial factor for workload performance on IMDB [2, 5, 6]. Parallelism is created by simultaneously executing operators within a query execution plan. We utilize *Plan Traces* of SAP HANA in this work to receive information about control flow and processed physical plan operators with millisecond precision for read queries. The traces are stored by SAP HANA on the file system using a proprietary Extensible Markup Language (XML) structure.

Data Transformation. The second step consists of translating the database traces to OPEN.xtrace

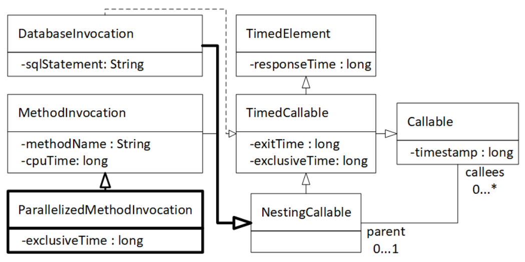

Figure 2: OPEN.xtrace Meta Model Modification (adapted from [10, 11])

[10] to receive a common representation of the captured query execution plans. We utilize the adapter provided in one of our previous work [11] to transform the *Plan Traces* to OPEN.xtrace. The adapter uses a modified OPEN.xtrace meta model [11], which enables to represent a query execution plan for a database call (i.e., *DatabaseInvocation*). We further extended the meta model in this work to support concurrent plan operator runs within a query execution plan by adding the element *ParallelizedMethodInvocation* (Figure 2). In contrast to the *MethodInvocation*, it calculates the *exclusiveTime* based on the children's maximum *responseTime*. We adjusted the adapter from our previous work to support this extension.

Model Generation. In the final step, the data in the OPEN.xtrace format is extracted to generate and parameterize a PCM performance model instance.

In order to model the database system, a single component is created in the PCM repository model. An operation is added for each statement extracted from the OPEN.xtrace data. The service effect specification (SEFF) of the operation defines the activities and resource demands within the statement's query execution plan. The assigned plan operators are mapped to *InternalActions* in the SEFF. The operator's run time behavior is reflected by the resource demands CPU and delay. As fork behavior supports the modeling of query parallelism on IMDB [2, 4, 7], we use *ForkActions* with synchronized *ForkBehaviors* to represent concurrent plan operator executions in PCM. Figure 3 shows an example where the two operators *PlanOperatorA* and *PlanOperatorB* are processed in parallel. To limit the number of simultaneously active queries and operators on the database as a whole, two passive resources can be adjusted to the thread pool settings on the referenced system.

To model the multi core environment of an IMDB, we use a network of processor sharing queues [2, 4, 7]. Our implementation in PCM consists of a CPU and delay resource with the scheduling policies *ProcessorSharing*. The number of replicas is adjusted to the total logical cores on the referenced system.

Database workload can be modeled in the PCM usage model. A query execution is mapped to an *EntryLevelSystemCall* element, which is linked to the statement's SEFF. Currently, this model is generated

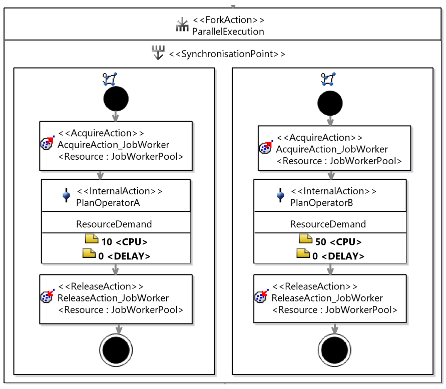

Figure 3: SEFF of Concurrent Plan Operator Runs

automatically for the workload used in this paper. For other database workloads, it can be adjusted before being used for prediction.

3 Evaluation

We evaluate our approach by generating performance models for the TPC benchmark H (TPC-H). Our test environment consists of a single tenant database on a SAP HANA 2.0 with support package stack (SPS) 2. The database runs virtualized using a logical partition (LPAR) on an IBM Power E870 server with 2 central processing units (CPU), eight-thread SMT mode (i.e., 16 logical cores) and 250 GB RAM.

We measured throughput, response time per query set and CPU utilization for the TPC-H workload using the openly available CH-benCHmark implementation [5]. It creates a data set, initializes the database in a column store and executes the set of 22 TPC-H queries sequentially. We parameterize the benchmark using 100 warehouses, a single analytical client, a warm-up time of 120 seconds and a test phase of 900 seconds. We modified the implementation to capture query response times, too. The CPU utilization is traced using the *top* command with a refresh interval of 1 second. To vary parallelism during query runs, we adapted the database configuration by limiting the number of concurrently active operators (P) within a statement to a maximum of 1, 4 and 16.

Afterwards, we created performance models for the three scenarios using our software prototype. The *Plan Traces* are captured for isolated TPC-H query runs separated from our measurements. The simulation engine SimuCom of PCM is utilized to evaluate the accuracy of the created models. The simulation results are compared to the measurements. Figure 4 presents the measured throughput (MT), simulated throughput (ST), measured mean response time (MMRT), simulated mean response time (SMRT), measured mean CPU utilization (MMCPU) and simulated mean CPU utilization (SMCPU). As shown in Table 1, the simulation predicts the throughput with relative prediction error rates between -2.14 % and 1.30 %. For simulating the response time, we receive

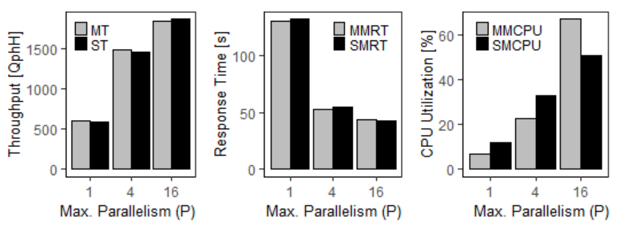

Figure 4: Measured and Simulated Results

error rates between -1.29 % and 3.36 % compared to average errors of 6 % [7], 8 % [4] and 25 % [2] presented for existing monolithic models. Simulating CPU utilization returns (relative) high error rates up to 72.18 %, which results from a difference of 7.01 to 12.07 % for a maximum intra-query parallelism of 1.

Table 1: Relative Prediction Errors

P	Throughput	Response Time	CPU Util.
1	-0.67 %	1.44 %	72.18 %
4	-2.14 %	3.36 %	45.01 %
16	1.30 %	-1.29 %	-24.34 %

4 Related Work

Traditional research on workload performance focuses on disk-based databases, while Merkle and Stier [3] apply PCM to reflect database locks. For IMDB, Schaffner et al. [1] use an empirical model excluding intra-query parallelism to predict query response time for SAP TREX server clusters. Other approaches are based on monolithic performance model notations. Wust et al. [6] use a QN to analyze the impact of a single query's sub tasks on another query's run time. Other authors [2, 4, 7] incorporate thread parallelism into a QN to simulate the performance for long running queries on SAP HANA. It is extended to map tenant interference and main memory [8, 9]. In contrast to our work, the approaches utilize coarse grained thread samples on the Operating System (OS) and the model creation implies profound domain specific knowledge. The parameterized models are valid for a preset queue of queries and threads.

5 Conclusion and Future Work

This paper presents an architecture level model creation approach based on OPEN.xtrace and PCM to predict workload performance on IMDB. Our current implementation is applicable for SAP HANA. Simulating TPC-H workload returns prediction errors below 4 % for throughput and response time. In order to reduce error rates, the next step in our work will be to optimize modeling of CPU utilization by adding information about the breakdown of plan operators into subtasks. In addition, we will parameterize our models using aggregated performance data including main memory. Furthermore, we intend to make our prototype executable outside of the Palladio bench to enable performance prediction on run time.

References

[1] J. Schaffner et al. "Predicting in-memory database performance for automating cluster management tasks". In: *27th International Conference on Data Engineering.* IEEE, 2011.

[2] S. Kraft et al. "Wiq: work-intensive query scheduling for in-memory database systems". In: *5th International Conference on Cloud Computing.* IEEE, 2012, pp. 33–40.

[3] P. Merkle and C. Stier. "Modelling database lock-contention in architecture-level performance simulation". In: *Proceedings of the 5th ACM/SPEC international conference on Performance engineering.* ACM, 2014, pp. 285–288.

[4] K. Molka et al. "Memory-aware sizing for in-memory databases". In: *Network Operations and Management Symposium.* IEEE, 2014.

[5] I. Psaroudakis et al. "Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling". In: *Technology Conference on Performance Evaluation and Benchmarking.* Springer, 2014, pp. 97–112.

[6] J. Wust et al. "Concurrent Execution of Mixed Enterprise Workloads on In-Memory Databases". In: *Database Systems for Advanced Applications.* Cham: Springer International Publishing, 2014, pp. 126–140.

[7] K. Molka and G. Casale. "Experiments or simulation? A characterization of evaluation methods for in-memory databases". In: *11th International Conference on Network and Service Management.* IEEE, 2015, pp. 201–209.

[8] K. Molka and G. Casale. "Contention-Aware Workload Placement for In-Memory Databases in Cloud Environments". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 2.1 (2016).

[9] K. Molka and G. Casale. "Efficient Memory Occupancy Models for In-memory Databases". In: *24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.* IEEE, 2016.

[10] D. Okanović et al. "Towards Performance Tooling Interoperability: An Open Format for Representing Execution Traces". In: *Proceedings of the 13th European Workshop on Performance Engineering.* LNCS. Springer, 2016.

[11] M. Barnert et al. "Converting Traces of In-Memory Database Systems to OPEN.XTRACE on the Example of SAP HANA". In: *Softwaretechnik-Trends* 37.3 (2017).