Converting Traces of In-Memory Database Systems to OPEN.XTRACE on the Example of SAP HANA

Maximilian Barnert, Adrian Streitz, Harald Kienegger, Helmut Krcmar
Chair for Information Systems
Technical University of Munich
Boltzmannstr. 3
85748 Garching, Germany
{maximilian.barnert, adrian.streitz, harald.kienegger, krcmar}@in.tum.de

Abstract

The shift of data-intensive application logic to in-memory Database Management Systems increases their importance for the overall performance of the software system. The performance of a processed query on a Database Management System is influenced by the utilized query execution plan, while traces capture the runtime behavior of the processed execution plan. However, the use of proprietary trace formats limits the usability within Application Performance Management tools and Software Performance Engineering approaches. OPEN.XTRACE is an open format to exchange execution traces, but its current data model does not support the integration of internal Database Management System operations. In this paper, we propose a modification to OPEN.XTRACE that enables a common representation of a query execution trace. In addition, we convert traces of the state-of-the-art in-memory Database Management System SAP HANA into this format.

Keywords: OPEN.XTRACE, database trace, query execution plan, SAP HANA

1 Introduction

Modern in-memory Database Management Systems (DBMS) boost the shift of data operations to the database layer [7, 11]. This shift implies that the DBMS adopts responsibilities from the application server, which increases a DBMS's impact on the overall software system performance. An important source for analyzing the DBMS performance are query execution plans, which provide insights into the control flow of a query execution and the applied plan operators. Fig. 1 shows an example of an execution plan representing the following query:

```
SELECT p.ProductName, AVG(s.Price)
 FROM Product p, Sales s
 WHERE p.ProductID = s.ProductID
 ORDER BY p.ProductName
```

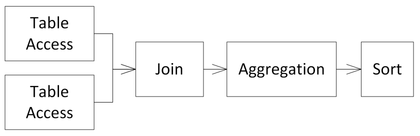

Figure 1: Example of a Query Execution Plan (tree representation)

The utilized plan operators and the execution sequence within such a plan influence a query's performance. DBMS like the state-of-the-art in-memory database SAP HANA capture information about the runtime behavior and performance of an execution plan in traces. However, the utilization of proprietary trace formats restricts their applicability to DBMS specific tools. This avoids the use of capabilities provided by other Application Performance Management (APM) tools. Additionally, it limits the reuse of Software Performance Engineering (SPE) approaches, e.g., for performance prediction.

The objective of OPEN.XTRACE [5] is an increased interoperability between APM tools and SPE approaches. This common format allows the representation of an execution trace that reflects a software system's runtime behavior. Adapters enable the compatibility to APM tools. However, the data model excludes internal activities that take place on a DBMS. In this paper, we modify OPEN.XTRACE to receive an open and common format for query execution traces that reflects an execution plan's runtime behavior and performance on a DBMS. The goal is an enhanced reuse of APM tools and more general SPE approaches in the context of DBMS.

The paper is structured as follows: Section 2 describes the necessary modifications to OPEN.XTRACE to represent a query execution trace. Section 3 outlines the transformation of a query execution trace to the modified OPEN.XTRACE model using a proprietary trace of the in-memory DBMS SAP HANA. Section 4 gives an overview over related work. Finally, the conclusion and future work are drawn in Section 5.

2 Query Execution Trace Representation in OPEN.XTRACE

OPEN.XTRACE consists of a meta-model, a default-implementation and several adapters that enable the data exchange with existing APM tools [5], e.g., Dynatrace AM [1] or Kieker [13].

The meta-model describes a set of components that represent the elements within an execution trace. The callable behavior (e.g., a method) is mapped to the entity *Callable*, which is extended by *TimedCallable*. The *NestingCallable* inherits *TimedCallable* to enable the assignment of a child *Callable*. This allows the reflection of a method-to-method call. Okanović et al. [5] do not assume the monitoring of internal DBMS operations. Therefore, the *DatabaseInvocation* representing a database call extends the *TimedCallable* and not the *NestingCallable*. This prevents the assignment of an operation to a database call. The representation of a query execution trace to the OPEN.XTRACE format necessitates the modification of the *DatabaseInvocation's* inheritance hierarchy (Fig. 2). The extension of the *NestingCallable* instead of the *TimedCallable* allows the assignment of a *Callable* (i.e., a plan operator) to the *DatabaseInvocation*.

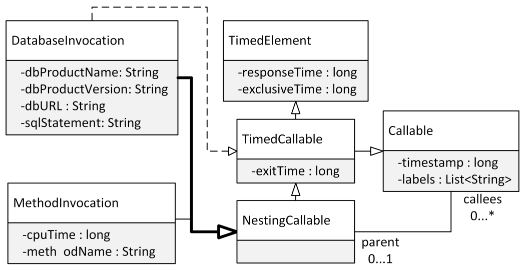

Figure 2: OPEN.XTRACE Meta-Model Modification (adapted from [5])

The modification of the meta-model makes the adaption of the OPEN.XTRACE default implementation necessary. As the *DatabaseInvocation* extends the *NestingCallable* in the modified meta-model, the *DatabaseInvocationImpl* has to extend the *AbstractNestingCallableImpl* in place of the *AbstractTimedCallableImpl* in the default implementation.

There are various adapters that translate proprietary trace formats into the default OPEN.XTRACE data model. They need to be adjusted to support the modified OPEN.XTRACE meta-model. However, there is no adapter that transforms a proprietary database trace. This necessitates the implementation of an adapter performing this kind of transformation by mapping a database call to *DatabaseInvocation* and a plan operator to *MethodInvocation*.

3 Example SAP HANA

The in-memory DBMS SAP HANA supports the analysis of a query's performance by providing different kinds of traces, for instance the plan trace [6]. It captures information about the runtime behavior (e.g., control flow, utilized plan operators, response time and CPU utilization) while executing a SELECT statement. The DBMS stores the trace on the file system, whereas it uses a separate file for each query execution. The use of a proprietary format limits the application to SAP specific tools (e.g., the SAP HANA Studio), but the trace's Extensible Markup Language (XML) structure allows the extraction of information about the query execution. Fig. 3 highlights the node hierarchy within this structure concentrating on the most important elements. The trace begins with the node *Plan*. It has besides others the children *StmtMemStats*, *RootRelation* and *Relation*. The *RootRelation* and *StmtMemStats* contain overall information about the query's processing including the start time, the response time and the memory usage. The node *Relation* represents an executed plan operation. A plan operator can call other operators, which are depicted by a child *InnerPlans* having at least one node *Plan*. This element contains nodes of the type *Relation* reflecting the called plan operators.

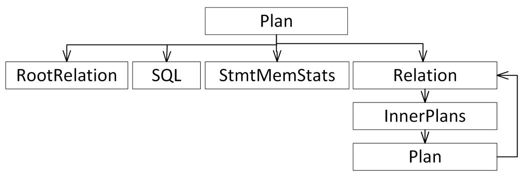

Figure 3: XML Structure of the SAP HANA Plan Trace (extract)

The implemented adapter transforms this proprietary XML structure into the modified OPEN.XTRACE data model, which is described in Section 2. It maps the database call to a *DatabaseInnvocation* object and assigns the start time and response time as attributes. In addition, each plan operator listed in the trace is translated to an object *MethodInvocation*. The information about the CPU time, the start time and the response time are assigned to it. Each plan operator is either a child of the entity *DatabaseInvocation* or *MethodInvocation* depending on the path in the XML structure. Fig. 4 shows a simplified example for a resulting object model where a SELECT statement is executed. It utilizes the plan operator *BWPopSearch* to fetch the data from a column table. Afterwards, the *BWPopAggregateParallel* aggregates several columns.

4 Related Work

The related work is divided into common execution trace formats and SPE approaches based on query execution traces. In addition, it lists SPE approaches in the context of in-memory DBMS that utilize traces.

There exist a few approaches that cover common execution trace formats. Most are limited to high-

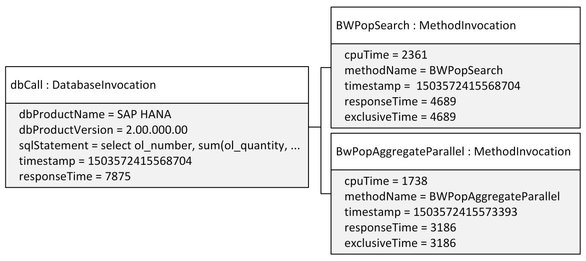
dbCall : DatabaseInvocation
dbProductName = SAP HANA
dbProductVersion = 2.00.000.00
sqlStatement = select ol_number, sum(ol_quantity, ...
timestamp = 1503572415568704
responseTime = 7875
BWPopSearch : MethodInvocation
cpuTime = 2361
methodName = BWPopSearch
timestamp = 1503572415568704
responseTime = 4689
exclusiveTime = 4689
BwPopAggregateParallel : MethodInvocation
cpuTime = 1738
methodName = BWPopAggregateParallel
timestamp = 1503572415573393
responseTime = 3186
exclusiveTime = 3186

Figure 4: Example Object Model for a SAP HANA Plan Trace

level events or specific tools [5]. This includes the Open Trace format [3], the Common Base Event format [2] or the Application Response Measurement [3]. OPEN.XTRACE tries to close this gap and provides an open and common execution trace format focusing on the runtime behavior and control flow of a software system, but it excludes the monitoring of DBMS internals [5].

Several SPE approaches concentrate on query execution traces. Simitsis et al. [12] transfer the query execution trace of MonetDB to Stethoscope to analyze the performance of an executed query plan. Gawade and Kersten [4] use the Vertica Management Console to collect performance data on the DBMS and operation system to get insights into a plan operator's performance. Besides, Kraft et al. [8], Molka et al. [10] and Molka and Casale [9] utilize system traces to simulate the performance on the in-memory DBMS SAP HANA.

5 Conclusion and Future Work

In this paper we describe an approach to represent a query execution trace in the common OPEN.XTRACE format. Additionally, we utilize the plan trace of the in-memory DBMS SAP HANA to demonstrate how to translate a proprietary trace to this data model. As shown, the modification of OPEN.XTRACE enables the representation of a database trace in an open and common format.

In the future, we plan to incorporate the memory usage that is also captured by DBMS traces during a query execution. Our long term objective is the application of OPEN.XTRACE for various SPE activities in the context of in-memory DBMS including performance modeling and simulation.

References

[1] *Application Monitoring — Dynatrace.* https://www.dynatrace.de/loesungen/application-monitoring/, accessed: 30.08.2017.

[2] B. Jacob et al. "A Practical Guide to the IBM Autonomic Computing Toolkit". In: *IBM Redbooks* 4 (2004), p. 10.

[3] A. Knüpfer et al. "Introducing the Open Trace Format (OTF)". In: *Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006. Proceedings, Part II.* Ed. by V. N. Alexandrov et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 526–533.

[4] M. Gawade and M. Kersten. "Stethoscope: A Platform for Interactive Visual Analysis of Query Execution Plans". In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 1926–1929.

[5] D. Okanović et al. "Towards Performance Tooling Interoperability: An Open Format for Representing Execution Traces". In: *Computer Performance Engineering: 13th European Workshop, EPEW 2016, Chios, Greece, October 5-7, 2016, Proceedings.* Ed. by D. Fiems, M. Paolieri, and A. N. Platis. Cham: Springer International Publishing, 2016, pp. 94–108.

[6] SAP. *SAP HANA Troubleshooting and Performance Analysis Guide.* Report. 2016.

[7] S. Balko and A. Barros. "In-Memory Business Process Management". In: *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, pp. 74–83.

[8] S. Kraft et al. "WIQ: Work-Intensive Query Scheduling for In-Memory Database Systems". In: *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on.* IEEE, pp. 33–40.

[9] K. Molka and G. Casale. "Experiments or Simulation? A Characterization of Evaluation Methods for In-Memory Databases". In: *Network and Service Management (CNSM), 2015 11th International Conference on.* IEEE, pp. 201–209.

[10] K. Molka et al. "Memory-Aware Sizing for In-Memory Databases". In: *2014 IEEE Network Operations and Management Symposium (NOMS).* IEEE, pp. 1–9.

[11] H. Plattner. "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, pp. 1–2.

[12] A. Simitsis et al. "VQA: Vertica Query Analyzer". In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data.* ACM, pp. 701–704.

[13] A. Van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering.* ACM, pp. 247–248.