Matthias Böhm, Dirk Habich, Wolfgang Lehner, Uwe Wloka

DIPBench Toolsuite: A Framework for Benchmarking Integration Systems

Erstveröffentlichung in / First published in:

IEEE 24th International Conference on Data Engineering. Cancun, 07.-12.04.2008. IEEE, S. 1596-1600. ISBN 978-1-4244-1836-7

DOI: https://doi.org/10.1109/ICDE.2008.4497630

DIPBench Toolsuite: A Framework for Benchmarking Integration Systems

Matthias Böhm #1, Dirk Habich *2, Wolfgang Lehner *3, Uwe Wloka #4

#*Database Group, Dresden University of Applied Sciences*
01069 Dresden, Germany
1mboehm@informatik.htw-dresden.de
4wloka@informatik.htw-dresden.de

**Database Technology Group, Dresden University of Technology*
01187 Dresden, Germany
2dirk.habich@inf.tu-dresden.de
3wolfgang.lehner@inf.tu-dresden.de

***Abstract*— So far the optimization of integration processes between heterogeneous data sources is still an open challenge. A first step towards sufficient techniques was the specification of a universal benchmark for integration systems. This DIPBench allows to compare solutions under controlled conditions and would help generate interest in this research area. However, we see the requirement for providing a sophisticated toolsuite in order to minimize the effort for benchmark execution. This demo illustrates the use of the DIPBench toolsuite. We show the macro-architecture as well as the micro-architecture of each tool. Furthermore, we also present the first reference benchmark implementation using a federated DBMS. Thereby, we discuss the impact of the defined benchmark scale factors. Finally, we want to give guidance on how to benchmark other integration systems and how to extend the toolsuite with new distribution functions or other functionalities.**

I. Introduction

The importance of heterogeneous systems integration is continuously increasing because new application types and technologies are emerging. This results in a multiplicity of different integration systems. The authors of the *Lowell Report* [1] already pointed out the need for further work on the optimization of information integrators. In this report, they encouraged the generation of a testbed and a collection of integration tasks. The presented testbed and benchmark THALIA [2], [3] realizes this but addresses the functional comparison, using the number of correctly answered benchmark queries, rather than the comparison of the integration performance. There are benchmarks available which partly contribute to heterogeneous systems integration. First, the newly standardized TPC-DS benchmark [4], [5], [6] includes a server-centric Extraction Transformation Loading (ETL) process execution. In order to separate DIPBench from this, it should be noticed that only flat files are imported into the data warehouse. Thus, it addresses the DBMS performance rather than the performance of a real integration system. Second, there are very specific ETL benchmarks available which mainly address the raw data throughput and are thus not sufficient for a universal benchmarking of integration systems. An example of such a specific benchmark is the so-called "RODIN High Performance Extract/Transform/Load Benchmark" [7]. That means, there is currently no performance benchmark for integration systems available. However, in order to evaluate the integration performance of such solutions, a standardized universal performance benchmark is sorely required.

Therefore, we developed a scalable benchmark, called DIPBench (*D*ata *I*ntensive *I*ntegration *P*rocess *Bench*mark) [8], [9], for evaluating the performance of integration systems. Figure 1 depicts our domain-specific benchmark scenario. In addition to the global scenario, we specified a mix of 15 conceptual process types representing typical integration tasks. The practical relevance of this specific benchmark design was verified within several industry integration projects. The benchmark specification further comprises the three scale factors *datasize* d^x, *time* t^z and *distribution* f^y. Aside from these scale factors, we defined the benchmark schedule and performance metrics.

In contrast to existing benchmarks for XML data management [10], [11], [12], the execution of an integration benchmark is much more complex. This is caused by the lack of a platform-independent and declarative description for integration processes. In order to reach the necessary benchmark simplicity, according to the benchmark design principles (domain-specific, portable, scalable) mentioned in [13], we provide a sophisticated toolsuite to minimize the effort for benchmark execution. The *Workload Driver* of the TPoX benchmark [12] follows similar aims in another XML Benchmark context.

The availability of such an integration benchmark allows comparing existing integration systems with regard to their performance. In addition to that, it will generate more research interest in the area of integration technologies. Therefore, our motivation of this demonstration proposal includes the presentation of reference benchmark implementations as well as explanations on how to extend the toolsuite with new integration system connections; we also discuss further research aspects. This will guide research as well as industrial groups on how to use our toolsuite for benchmarking novel implementations within the field of integration.

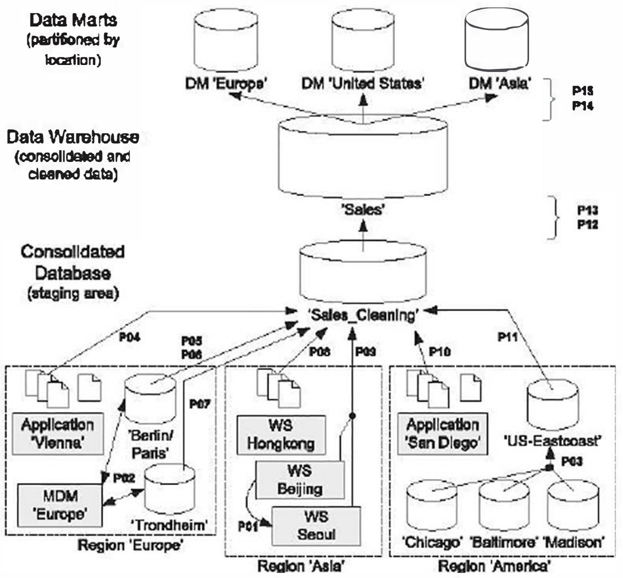

Fig. 1. DIPBench ETL Scenario

Our DIPBench specification of an integration benchmark and the presentation of the correlated toolsuite are first steps towards more extensive research on optimization techniques in the context of integration processes. So, the significance of the contribution is based on the preparation of preconditions for integration system comparison and further research on information integration optimization techniques. The contribution mainly consists of two parts: first, the conceptual discussion of the novel DIPBench benchmark specification and second, the presentation and explanation of the related toolsuite.

The remainder of the paper is structured as follows: In the following section, we give a short overview of our benchmark including the general benchmark scenario, the process type groups, as well as the specified scale factors. In Section 3, we discuss the DIPBench toolsuite architecture in detail, including its macro- and micro-architecture. Demonstration details are described in Section 4. Finally, the paper closes with a conclusion and future work.

II. DIPBench Overview

The technical context of the DIPBench comprises different types of integration tasks, which are typical for physical integration processes within an ETL environment. This benchmark addresses data manipulating integration systems rather than read-only information systems. In contrast to functional integration benchmarks, we focus on a real-life scenario rather than comprising all possible types of syntactic and semantic heterogeneities. The benchmark scenario, illustrated in Figure 1, is divided into four layers, and different process types are specified for each layer.

- *Layer A - Source System Management:* The first layer represents all regionally separated source systems, including applications, Web services and different RDBMS. Further, three integration tasks are specified.
- *Layer B - Data Consolidation:* The second logical layer consists of a consolidated database (CDB). It represents the staging area of the whole ETL benchmark scenario. Basically, seven different integration tasks between the source systems and the CDB are defined.
- *Layer C - Data Warehouse Update:* Layer three represents the actual data warehouse (DWH) system. Only clean and consolidated data is loaded into this system, based on a defined time schedule. We have defined two integration processes between the CDB and the DWH.
- *Layer D - Data Mart Update:* In order to realize physical optimizations, workload reduction as well as a location-based partitioning, the fourth layer comprises three independent data marts. So, there are also 2 integration processes between the DWH and the data marts.

The internal processing of the whole benchmark will be influenced by the three scale factors *datasize* (d^x), *time* (t^z) and *distribution* (f^y). The continuous scale factor *datasize* (d^x) allows for scaling the amount of data to be integrated. Thus, the dataset size of the external systems, and in some cases the number of process instances, depends on it. The continuous scale factor *time* (t^z) allows the scaling of time intervals between process-initiating events by influencing the benchmark scheduling. Moreover, the scale factor *time* has an impact on the degree of concurrency. The discrete scale factor *distribution* (f^y) is used to provide different data characteristics from uniformly distributed data values to specially skewed data values.

III. DIPBench Toolsuite Architecture

In this section, we describe the architecture of our developed DIPBench toolsuite. First, we present the overall macro-architecture and its position within the whole benchmark execution environment. Second, we describe the micro-architecture of the demonstrated tools in short. Figure 2 illustrates the mentioned macro-architecture. Fundamentally, the tools `Client`, `Initializer` and `Monitor` are provided. These tools, which were implemented in Java and are thus platform-independent, could be accessed using an API or via a developed GUI. The benchmark execution runtime comprises one computer system for all external source and target systems and one computer system for the actual integration system.

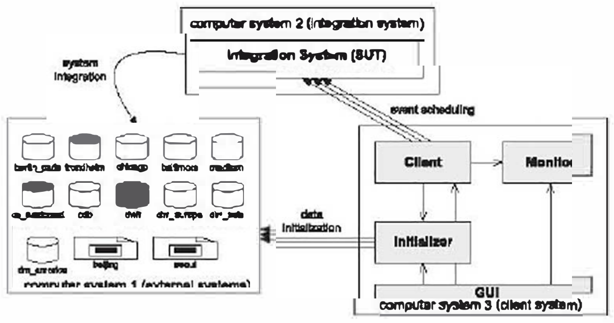

Fig. 2. Toolsuite Macro-Architecture

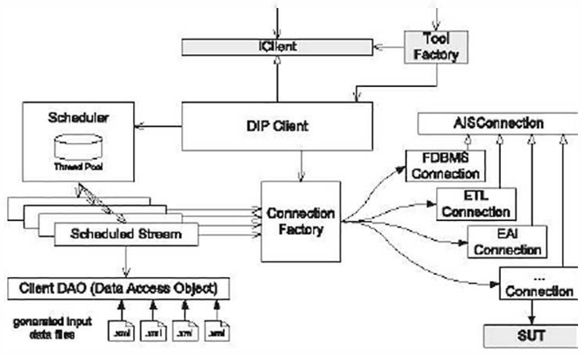

Fig. 3. `Client` Micro-Architecture

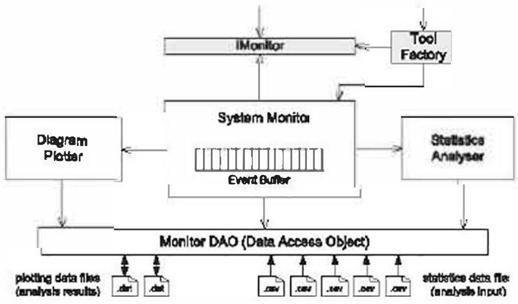
IMonitor
Tool Factory
System Monitor
Diagram Plotter
Event Buffer
Statistics Analyser
Monitor DAO (Data Access Object)
plotting data files (analysis results)
.dat
.csv
statistics data files (analysis input)

Fig. 5. `Monitor` Micro-Architecture

In the following, we want to discuss the architecture of the single tools and point out solved challenges and problems. Figure 3 shows the micro-architecture of the `Client` tool, which is used for scheduling integration processes. Thereby, it handles the connections to the integration system (system under test). From an abstract point of view, this tool is comparable to the TPoX [12] Workload Driver. The tool could be accessed using the well defined interface `IClient`. The `Scheduler` initializes and manages the execution of the four `Scheduled Streams`, which are in fact multi-threaded event streams. If an event occurs, the `ConnectionFactory` is used to invoke the integration system in a generic manner. Thus, for benchmarking one's own system prototype, only one such connection—inherited from `AISConnection`—has to be provided and registered at the `ConnectionFactory`.

In order to reach a sufficient number of statistics, the benchmark execution phase comprises 100 benchmark periods. After each period, the external systems have to be reinitialized. The `Client` uses the `Initializer` tool, whose architecture is shown in Figure 4. This tool allows schema creation and universal data initialization for relational data sources as well as XML-based systems. In analogy to the `Client`, this tool realizes the `IInitializer` interface. Basically, the schemas are specified with `TableMetadata` objects. These metadata objects also refer to foreign-key specifications, value domains, special IDBuckets and several types of constraint specifications. With these definitions, e.g., ranges of numbers could be applied for the different external systems but also single-column, multi-column, multi-table constraints can be specified. Furthermore, this tool allows multi-threaded data generation. The metadata objects are passed to the specific `Data Generator` and the generated tuples are directly put out into special tuplestreams. The aligned data connection could start to insert tuples while the data generation is not yet finished. By limiting the tuplestream to a maximum number of buffered tuples, the generation is realized with awareness of main memory restrictions. The very generic separation of the different data distribution functions and data connections makes it easy to extend the given tool. However, the current version even supports the specification of data properties (average, variance, skewness and null-values). Note that the cardinalities can be specified with the scale factor *datasize* as well as with the `ScaleTable` in order to separate linear and sub-linear scaling tables.

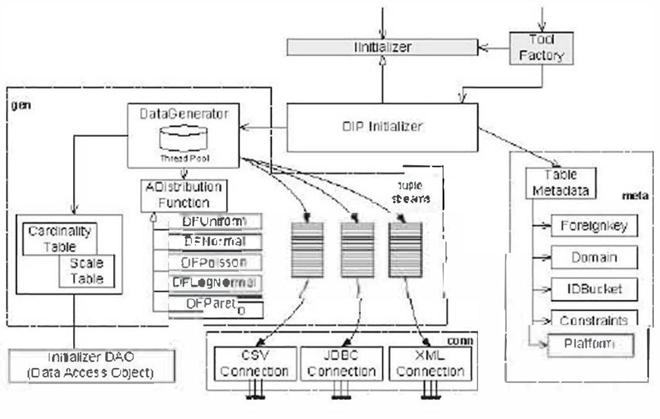

Fig. 4. `Initializer` Micro-Architecture

The `Monitor` tool is used by the `Client` and allows collecting, analyzing and plotting performance statistics. Figure 5 shows its micro-architecture, which is quite similar to the architecture of the already described tools. Thus, it implements the interface `IMonitor`. The core `System Monitor` comprises an `Event Buffer` where performance events are collected. These events are represented by a data structure, which includes the event ID (EID), the process type ID (PTID), the start time (T_0) and the end time (T_1). The whole `Event Buffer` is incrementally flushed to disk during benchmark execution. After the benchmark execution has been finished, the whole statistics are loaded and analyzed by the `Statistics Analyzer`. The result of this analysis is represented by the specific performance metrics [8] for each process type ID. In this area, a major challenge was the cost normalization. Finally, these results are flushed to disk in order to generate a performance plot with the `Diagram Plotter` using the jgnuplot library [14]. One might judge the event management as performance bottleneck. However, due to the fact that there are no side effects between the measured performance metric and the management of events, this can be disregarded.

IV. Demonstration Details

After the presentation of the DIPBench toolsuite and its architecture, we use this section to outline features we would like to present at ICDE. In general, the demonstration features can be classified into seven aspects.

General Toolsuite Presentation: First, we want to demonstrate the usage of the toolsuite by presenting the graphical user interface and explaining the provided API. Furthermore, configuration parameters and the benchmark setup are discussed in general. Second, this abstract presentation is used to illustrate the general DIPBench approach and to discuss benchmark-related aspects.

Data Generator Explanation: Considering the aspect of data generation, we will demonstrate the impact of our different data distribution functions. Especially the *poisson* distribution function—which allows different types of skewed data—will be explained in detail. Furthermore, we will demonstrate the different correlation types *one-column*, *between columns of one table* and *between columns of multiple tables*.

Reference Implementation Illustration: Further in-depth aspects are reference implementations for federated DBMS, an ETL tool, a subscription system as well as a WSBPEL process engine. First, the reference implementations are demonstrated. Second, implementation aspects and optimization possibilities are also discussed.

Scale Factor Impact Demonstration: After we have shown the GUI facilities for specifying the scale factor values, we demonstrate their impact using the mentioned reference implementations. So, for example, we show the impact of the scale factors datasize d^x, time t^z and distribution f^y.

Data Generator Implementation Guidance: As already mentioned, the `Initializer` tool could be extended with new data generators. This could be useful if new data distribution functions are needed or if special correlations should be implemented. So, we will use the demo to give guidance on how to extend the toolsuite with such data generators.

Benchmark Implementation Guidance: The `Client` tool could be extended with new integration system connections. This is necessary in order to realize a benchmark implementation. Thus, this is the extension point where research groups have to customize the toolsuite in order to benchmark their own system. With the intent of minimizing the effort of such customization tasks, we will give guidance on how to integrate their own connections.

Benchmark Discussion: Since we believe in the importance of such a benchmark, we would like to discuss open issues and challenges related to this benchmark. Further, we see some potential for more challenging integration tasks, including all types of syntactic and semantic heterogeneities. So, in the end, there is room to discuss future versions of the DIPBench specification and further research aspects like the model-driven generation and optimization of integration processes.

To summarize, the demo at ICDE comprises an in-depth explanation of all necessary aspects of our benchmark, including its reference implementations. Furthermore, visitors of our demonstration desk will get a more in-depth understanding of the benchmark and its further application areas.

V. Conclusions

We specified the DIPBench benchmark, because of the absence of an independent performance benchmark for data-intensive integration processes. In order to reach the highest possible simplicity within this complex context of integration processes, there was the need for the provision of a sophisticated toolsuite to minimize the effort necessary for new benchmark implementations. In this paper, we presented the DIPBench toolsuite, which could be used as a framework for benchmark integration systems. Thereby, the core toolsuite comprises three tools: `Client`, `Initializer` and `Monitor`. In addition to this, the number of adjusting screws within the toolsuite causes the significance of the demo. Thus, it is highly recommended to demonstrate and discuss the impact of special configurations but also to explain how one's own benchmark implementations could be realized with minimal effort. Finally, we want to use the demonstration program as a forum for discussing the benchmark specification.

Acknowledgment

We want to thank the students Katrin Braunschweig, Romain Treboux and Dirk Alsfaßer for their efforts on extending and enhancing the `Initializer` tool.

References

[1] S. Abiteboul, R. Agrawal, P. A. Bernstein, M. J. Carey, S. Ceri, W. B. Croft, D. J. DeWitt, M. J. Franklin, H. Garcia-Molina, D. Gawlick, J. Gray, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, M. L. Kersten, M. J. Pazzani, M. Lesk, D. Maier, J. F. Naughton, H.-J. Schek, T. K. Sellis, A. Silberschatz, M. Stonebraker, R. T. Snodgrass, J. D. Ullman, G. Weikum, J. Widom, and S. B. Zdonik, "The lowell database research self assessment," *CoRR*, vol. cs.DB/0310006, 2003.

[2] J. Hammer, M. Stonebraker, and O. Topsakal, "Thalia : Test harness for the assessment of legacy information integration approaches," University of Florida," Technical Report, 2005.

[3] ——, "Thalia: Test harness for the assessment of legacy information integration approaches." in *ICDE*, 2005, pp. 485–486.

[4] R. Othayoth and M. Poess, "The making of tpc-ds." in *VLDB*, 2006, pp. 1049–1058.

[5] M. Pöss, B. Smith, L. Kollár, and P.-Å. Larson, "Tpc-ds, taking decision support benchmarking to the next level." in *SIGMOD Conference*, 2002, pp. 582–587.

[6] *TPC-DS - ad-hoc, decision support benchmark*, Transaction Processing Performance Council, 2007.

[7] *High Performance Extract/Transform/Load Benchmark*, RODIN Data Asset Management, 2002.

[8] M. Böhm, D. Habich, W. Lehner, and U. Wloka, "Dipbench: An independent benchmark for data intensive integration processes," Dresden University of Applied Sciences," Technical Report, 2007.

[9] *DIPBench*, Dresden University of Technology, Database Technology Group, http://wwwdb.inf.tu-dresden.de/research/gcip/, 2007.

[10] T. Böhme and E. Rahm, "Xmach-1: A benchmark for xml data management." in *BTW*, 2001, pp. 264–273.

[11] ——, "Multi-user evaluation of xml data management systems with xmach-1." in *EEXTT*, 2002, pp. 148–158.

[12] M. Nicola, I. Kogan, and B. Schiefer, "An xml transaction processing benchmark." in *SIGMOD Conference*, 2007, pp. 937–948.

[13] J. Gray and A. Reuter, *Transaction Processing : Concepts and Techniques (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, October 1992. [Online]. Available: http://www.amazon.de/exec/obidos/ASIN/1558601902

[14] *java library for interfacing with the gnuplot plotting package*, jgnuplot project, http://jgnuplot.sourceforge.net/, 2007.