Matthias Böhm, Dirk Habich, Uwe Wloka, Wolfgang Lehner

DIPBench: An independent benchmark for Data-Intensive Integration Processes

Erstveröffentlichung in / First published in:

IEEE 24th International Conference on Data Engineering Workshop. Cancun, 07.-12.04.2008. IEEE, S. 214-221. ISBN 978-1-4244-2161-9

DOI: https://doi.org/10.1109/ICDEW.2008.4498321

DIPBench: An Independent Benchmark for Data-Intensive Integration Processes

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— **The integration of heterogeneous data sources is one of the main challenges within the area of data engineering. Due to the absence of an independent and universal benchmark for data-intensive integration processes, we propose a scalable benchmark, called DIPBench (*D*ata *I*ntensive *I*ntegration *P*rocess *Bench*mark), for evaluating the performance of integration systems. This benchmark could be used for subscription systems, like replication servers, distributed and federated DBMS or message-oriented middleware platforms like Enterprise Application Integration (EAI) servers and Extraction Transformation Loading (ETL) tools. In order to reach the mentioned universal view for integration processes, the benchmark is designed in a conceptual, process-driven way. The benchmark comprises 15 integration process types. We specify the source and target data schemas and provide a toolsuite for the initialization of the external systems, the execution of the benchmark and the monitoring of the integration system's performance. The core benchmark execution may be influenced by three scale factors. Finally, we discuss a metric unit used for evaluating the measured integration system's performance, and we illustrate our reference benchmark implementation for federated DBMS.**

I. Introduction

The integration of heterogeneous data sources is still one of the main challenges in the area of data engineering. This fact is caused by the tendency towards distributed system infrastructures with the simultaneous requirement of the availability of integrated data. Therefore, a lot of work has been conducted regarding integration concepts and systems. The integration concepts can be classified as follows [1]: information integration (data integration and function integration), application integration, process integration and even GUI integration.

Although there are several papers presenting individual performance experiments in order to confirm the impact of the respective integration systems, a standardized benchmark is not available. This lack may be explained by the diversity of integration systems. However, a standardized benchmark for integration systems is sorely required, both by industrial vendors and by academic research groups in order to compare and evaluate products and prototypes. The necessity of such a benchmark was even explicitly mentioned in [2], [3], [4].

Based on this motivation, we present our developed Data-Intensive Integration Process Benchmark (DIPBench) in this paper. The focus of the benchmark is on the physical data integration within the context of ETL processes. These processes, comprising the integration tasks, are initiated either by business transactions in the source systems or based on a time schedule. Obviously, we focus on data-manipulating integration systems rather than on read-only information systems. With the classification of integration concepts in mind, the DIPBench addresses the information integration as well as the application integration.

With this specification, we want to contribute to the definition of a standardized integration benchmark. Fundamentally, and in awareness of the number of different available integration systems, there are several challenges ahead. First, the specific functionalities of the different systems require a benchmark that is restricted to well-chosen source and target systems. Second, the comparability of benchmark results over different integration systems is also a major challenge. Finally, a platform-independent description of the benchmark is required in order to be able to map the benchmark to the specific systems. This platform-independent view on integration processes is achieved using our process-based Message Transformation Model (MTM) [5] conceptual description model.

Aside from the above-presented challenges, we try to accomplish the approved benchmark design principle of [6]. The following list shows these principles and requirements, and it illustrates how they are tackled by the DIPBench specification:

- *Domain-specific:* The benchmark must be relevant within the domain of heterogeneous systems integration. Thus, it has to comprise typical integration processes within an application scenario and all types of heterogeneities.
- *Portable:* Portability has to be reached by providing a platform-independent benchmark description. This principle addresses the integration system as well as the used source and target systems.
- *Scalable:* The benchmark should be scalable in order to be executed on small as well as on large computer systems. Therefore, a set of scale factors must be defined.

- *Simple:* Finally, the simplicity of the benchmark should be kept. Although this is quite a hard requirement dealing with complex integration systems, a toolsuite is provided to ensure a simple benchmark execution and analysis.

Based on the importance of such integration benchmark, we would like to invite industrial as well as research participants to contribute to this discussion in order to define the best suitable benchmark for integration systems. Therefore, the presentation of the main aspects of our defined benchmark is structured as follows: In Section 2, we distinguish our DIPBench approach from existing benchmarks, none of which addresses the actual integration system's performance. Afterwards, we introduce the domain-specific ETL scenario in Section 3. The actual benchmark description in Section 4 includes the process type definitions. Furthermore, in Section 5, we discuss the benchmark execution schedule as well as the impact of the three defined scale factors. Section 6 illustrates the performance metrics used for benchmark analysis and explains its computation. Based on the benchmark description and the performance metrics, we present our first reference benchmark implementation in Section 7, including experiments with different scale configurations. For that, we have examined a federated DBMS to prove the benchmark realizability and to show the first performance measurements. Finally, in Section 8, we conclude our discussion and give a short outlook on future work.

II. Related Work

The *Lowell Report* [2] already points out the need for further work on the optimization of information integrators. In this report, the authors encourage the generation of a testbed and a collection of integration tasks. In their opinion, this would allow the comparison of solutions and it would certainly help generate interest in this research area. The testbed and benchmark THALIA [7], [8] was a direct response to the *Lowell Report*. It provides a testbed and a benchmark for information integration. Using THALIA, however, integration systems can be evaluated based on the number of correctly answered benchmark queries and the amount of integration effort. Thus, it rather addresses the integration functionalities than the processing performance of integration systems. As mentioned within our introduction, there is little work related to benchmark integration systems. However, selected research groups illustrate performance experiments [4], [9] using self-defined queries in order to prove their own results.

Trying to keep it simple, we define the DIPBench similar to well-known benchmarks. The XMach benchmark [10], [11] is positioned as a multi-user benchmark for evaluating the query performance of XML Data Management Systems, while the TPC-H benchmark is used to determine the ad-hoc decision support performance of a DBMS. Even our toolsuite [12] was designed in analogy to the *Workload Driver* of the TPoX benchmark [13]. There are benchmarks available which partly contribute to heterogeneous systems integration. First, the newly standardized TPC-DS benchmark [14], [15], [16] includes a server-centric 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 rather addresses the DBMS performance than the performance of a real integration system. Second, there are very specific ETL benchmarks available which mainly address the raw data throughput and which 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" [17]. Further, also an ETL Benchmark with quality metrics was already discussed in [18]. Additionally, the Data Warehouse Engineering Benchmark (DWEB) [19], [20] and the MOM Benchmark SPECjms2007 [21] should be mentioned. Unfortunately, the latter addresses JMS implementations only.

III. Benchmark Scenario

In this section, we give an overview of our chosen benchmark application scenario comprising a full Extraction, Transformation, Load (ETL) process but also OLTP integration processes between the source systems. This complex technical context contains different types of integration tasks which are characteristic for physical integration processes. We have verified the practical relevance during several industry projects in the field of information integration and application integration. So, in contrast to functional integration benchmarks, we rather focus on a real-life scenario instead of covering all possible types of syntactic and semantic heterogeneities.

The business context consists of a group of companies (GP). The sub-companies are located in Europe, Asia and America. This regional separation is also used for the differentiation of special cost centers. Thus, movement data—in the form of sales data—but also master data has to be integrated from the numerous source systems into a global

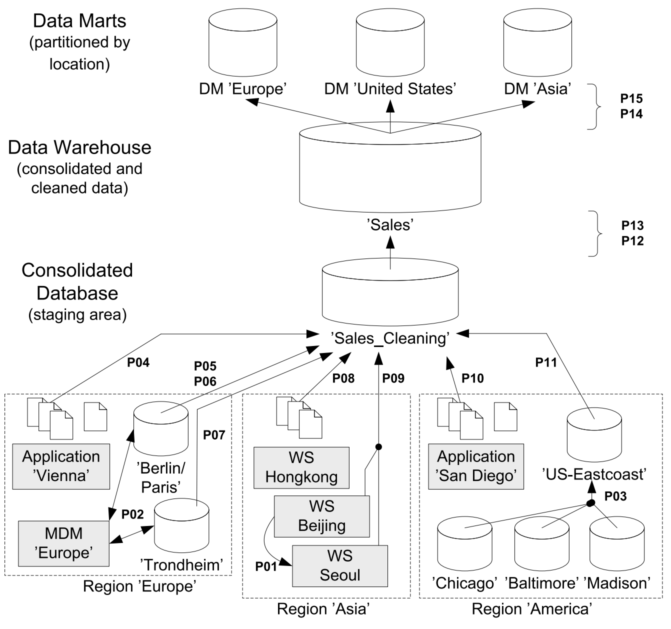

Fig. 1. DIPBench ETL Scenario

Final edited form was published in "IEEE 24th International Conference on Data Engineering Workshop. Cancun, 2008". S. 214-221. ISBN 978-1-4244-2161-9
https://doi.org/10.1109/ICDEW.2008.4498321

data warehouse and finally into cost-center-specific data marts. The whole integration scenario is illustrated in Figure 1. All necessary integration processes within this scenario should be realized using one integration system. The DIPBench, which is described in detail in the following sections, addresses the performance evaluation of the used integration system.

A. Data Flow Description

In order to support a wide range of integration systems, the external system types are limited to RDBMS, Web services and XML-based flat files. The whole system infrastructure is basically divided into four logical layers.

The first logical layer represents all source systems, including the mentioned applications, Web services and different kinds of RDBMS. As Figure 1 shows, these source systems are grouped by their locations. Thus, the three regional groups *Europe*, *Asia* and *America* are defined. Within the regional group *Europe*, there are four physical source systems. There, a specific MDM (Master Data Management) application is used. The proprietary application *Vienna* sends XML messages—driven by business transactions—to the consolidated database, where these messages have to be enriched with master data. There are two databases—for the three locations *Berlin*, *Paris* and *Trondheim*—which have to be loaded into the consolidated database using a time-based schedule. If master data changes occur during OLTP transactions, these master data have to be replicated to the mentioned databases. The regional group *Asia* comprises three Web services. Each of them manages its master data locally. Thus, there is a local master data consolidation between the Web service *Beijing* and *Seoul*. Concerning the data consolidation process, there are two integration types to be distinguished. While the Web service *Hongkong* sends its data to the consolidated database—driven by business transactions—the data from *Beijing* and *Seoul* are explicitly queried and merged in order to load them into the consolidated database. In contrast to the previously mentioned regional groups, the region *America* is part of a two-phase consolidation process. First, the data from the data sources *Chicago*, *Baltimore* and *Madison* are loaded into a local consolidated database, called *US_Eastcoast*; second, they are loaded into the global consolidated database. Additionally, there is a proprietary application *San Diego*, that sends XML messages directly to the global consolidated database. It is assumed that this application is very error-prone, which requires a detailed validation process when receiving such messages.

The second layer consists of a consolidated database. It is used for the physical source data integration of all different source system types. Due to the different source data schemas, several schema mappings and the data cleaning of master and movement data have to be processed. Thus, the consolidated database represents the staging area of the whole ETL benchmark scenario. During this staging process, the data quality increases and the accuracy decreases.

Layer three represents the actual data warehouse system. In order to minimize the system load of this layer, only clean and consolidated data is loaded into this system, based on a defined time schedule. Thus, the accuracy is even lower than the accuracy of data residing within the consolidated database. In contrast to this, the data quality is much higher because an integrated view on all data is provided.

In order to realize physical optimizations, workload reduction and a location-based partitioning, the fourth layer comprises three independent data marts. After the data is loaded into these region-specific data marts, the materialized views have to be refreshed.

B. Schema Descriptions

The source and target systems exhibit heterogeneous data schemas. First, there are syntactic heterogeneities, which refer to structural differences between the data schemas. Examples for that type are normalized and denormalized tables but also relational data in contrast to hierarchically structured XML files. Second, there are also semantic heterogeneities (e.g., there are different meanings of priority flags and order states).

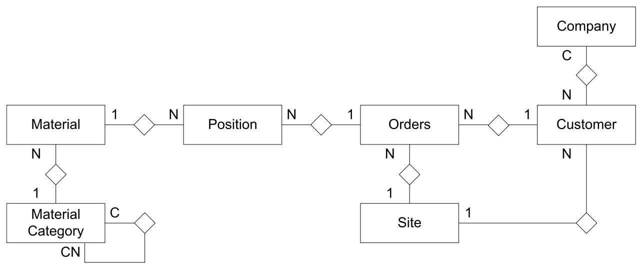

Fig. 2. Region `Europe` Data Schema

According to the regional differentiation, the region *Europe* mostly uses a self-defined, normalized data schema, which is illustrated in Figure 2. However, the applications *Vienna* and *MDM_Europe* use specific deep-structured XML schemas. The region *Asia* follows a generic approach, where all schemas are expressed with default result set XSDs. This implicates that these three Web services are simply data sources hidden by Web services. Finally, there is the region *America*,

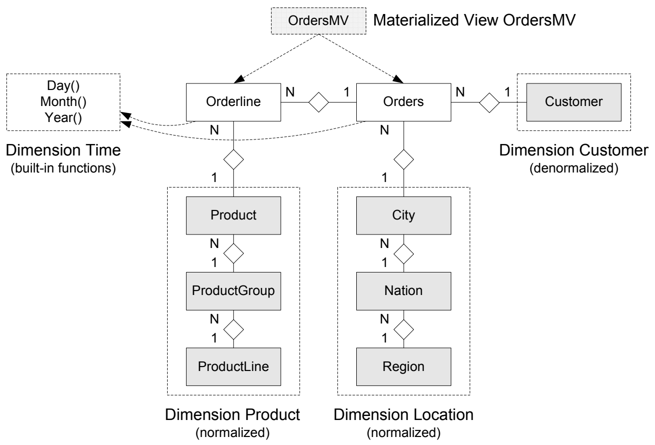

Fig. 3. Data Warehouse Data Schema

Group	ID	Name
A	P01	Master data exchange Asia
A	P02	Master data subscription Europe
A	P03	Local data consolidation America
B	P04	Receive messages from Vienna
B	P05	Extract data from Berlin
B	P06	Extract data from Paris
B	P07	Extract data from Trondheim
B	P08	Receive messages from Hongkong
B	P09	Extract wrapped data from Beijing and Seoul
B	P10	Receive error-prone messages from San Diego
B	P11	Extract data from CDB America
C	P12	Bulk-loading data warehouse master data
C	P13	Bulk-loading data warehouse movement data
D	P14	Refreshing data mart data
D	P15	Refreshing data mart materialized views

TABLE I
BENCHMARK PROCESS TYPES OF GROUPS A, B, C AND D

whose schema follows exactly the normalized TPC-H schema. However, the application *San Diego* uses a different deep-structured XML schema. Note that the detailed relational models and XML schemas can be found in the full benchmark specification and are thus not explicitly illustrated in this paper due to the lack of space.

In contrast to the heterogeneous source system schemas, the schema definitions of the consolidated database, the data warehouse, and the region-specific data marts are quite homogeneous. However, in detail, there are structural differences caused by the specific usage of these systems. Figure 3 illustrates a data warehouse's snowflake schema as the central point of this scenario. The single data mart schemas are derived from this. The data mart *Europe* comprises denormalized product and location dimensions, while the data mart *Asia* only has the product dimension denormalized and *United_States* has a denormalized location dimension. In contrast to these read-optimized schemas, the schema of the consolidated database is equal to the data warehouse schema, except for the materialized view OrdersMV.

IV. PROCESS TYPE DEFINITIONS

In this section, we define several integration process types for each of our four logical layers. The definitions include the two main event types: (E1) incoming messages and (E2) time-based scheduling events. Moreover, we explicitly point out that the modeled processes are suboptimal. This leaves enough space for optimizations as described in [22].

A. Group A: Source System Management

The process type P01 addresses the master data exchange between the Web service *Beijing* and the Web service *Seoul*. Thereby, an XML message, conforming to the defined *XSD_Beijing*, is received, translated to *XSD_Seoul* using a given STX [23], [24] translation, and finally sent to *Beijing*. In P02, an XML message is received from the MDM application and translated to the data schema `Europe`. Subsequently, a `SWITCH` operator evaluates the given Customer identifier

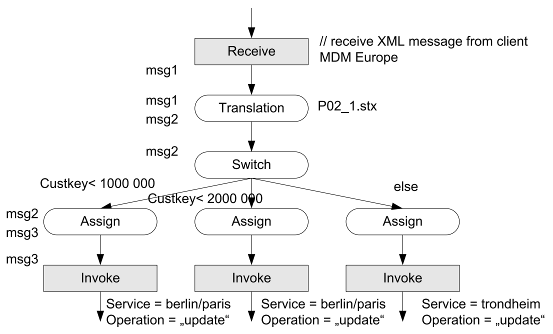

Fig. 4. Example Process Type P02

(Custkey). Depending on the evaluation result, the message is sent to either *Berlin*, *Paris* or *Trondheim*. This process type is illustrated by Figure 4.

The group of source system management is completed by P03. This process type is not initiated by incoming messages but based on a time schedule. It first extracts the data sets from *Chicago*, *Baltimore*, and *Madison*. After that, a UNION DISTINCT is processed for the tables Orders, Customer and Part. Finally, the resulting data set is loaded into the local consolidated database *US_Eastcoast*. The whole process type is shown in Figure 5.

B. Group B: Data Consolidation

The process type P04 deals with receiving *Vienna* XML messages and their enrichment with extracted master data. After that, the messages are translated in a standardized way and sent to the consolidated database. The process types P05, P06 and P07 are technically very similar. So, a dataset is extracted from the data sources *Berlin/Paris* or *Trondheim*. In case of *Berlin/Paris*, a selection is processed for filtering the right location. Furthermore, a projection is executed in order to rename the attributes. Finally, the query is prepared for inserting the dataset. Note that they are executed on a time-based schedule.

The data flow between the regional group *Asia* and the consolidated database comprises the process types P08 and

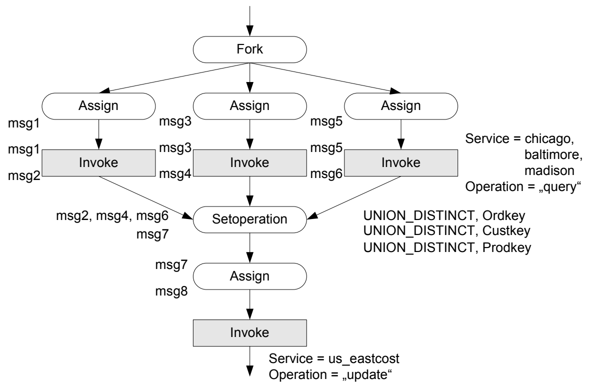

Fig. 5. Example Process Type P03

P09. There, P08 is initiated by incoming messages received from the Web service *Hongkong*. After a schema translation has been performed, the messages are loaded into the consolidated database. In contrast to that, P09 is initiated in a time-based manner. Thereby, large XML result sets are extracted from the Web services *Beijing* and *Seoul*. After that, these different result sets are translated to the CDB schema using two different STX style sheets. Before they can be loaded into the CDB, a UNION DISTINCT concerning the Orderkey, Custkey and Productkey has to be processed.

Finally, group B also includes the process types P10 and P11 concerning the American data flow. The process type P10 addresses the reception of error-prone messages from the application *San Diego*. There, the messages are first validated. In case of an error, the data is inserted into special destinations for failed data. Otherwise, the message is translated to the CDB schema and finally inserted into the consolidated database. P11 is more data-centric. Thus, it addresses the extraction of all data consolidated within the local consolidated database *US_Eastcoast* and its loading into the global consolidated database *Sales_Cleaning*. Between the extraction and load phases, several projections have to be processed, realizing a simple schema mapping.

C. Group C: Data Warehouse Update

Complementary to groups A and B, the groups C and D address data-intensive process types exclusively. Group C deals with the data warehouse delta update. So, P12 invokes a stored procedure `sp_runMasterDataCleansing` in order to eliminate master data duplicates and error-prone master data within the consolidated database. After that, it extracts the clean master data from the CDB, validates it, and if the validation succeeds, loads this data set into the data warehouse. Finally, the master data within the consolidated database is flagged as integrated but not physically removed.

The process type P13 is very similar to P12. It also invokes a stored procedure `sp_runMovementDataCleansing` in order to eliminate the movement data errors. Furthermore, it also extracts, validates and loads the movement data to the data warehouse. At this point, the differences in data set sizes should be noticed. Finally, two invocations are processed. First, the materialized view OrdersMV has to be refreshed by a stored procedure call. Second, the loaded movement data has to be removed from the consolidated database for simple delta determination in the following integration processes.

D. Group D: Data Mart Update

In addition to that, group D comprises a high degree of parallelism. The process type P14 consists of a main process and four subprocesses. First, subprocess P14_S1 is invoked in order to load all master and movement data from the data warehouse and return it. Second, three concurrent threads are processed in parallel. Such a thread consists of a selection operator and the invocation of a subprocess. The called subprocess realizes the schema mapping from the DWH schema to the special DM schema and finally loads the data into the chosen data mart. At the end of the whole ETL process scenario, the materialized views of all data marts have to be refreshed. Since there are no dependencies between the physical data marts, these could be processed in parallel.

Finally, be aware of the full platform-independent benchmark specification, which is available in [25], including detailed descriptions, the toolsuite and the reference benchmark implementation for federated DBMS.

V. Benchmark Execution Schedule and Performance Metric

The DIPBench is executed using a defined environment setup. Therefore, three independent computer systems are needed. First, there is computer system 1 (ES), where all external systems, for instance DBMS and application servers, are installed. Second, the computer system 2 (IS) represents the integration system installation (system under test). Finally, there is computer system 3 (CS), where all single tools of the DIPBench toolsuite reside.

This toolsuite is provided in order to minimize the time and effort, needed for benchmarking a special integration system.

- *Initializer:* First, it creates the different database schemas and XML files. Second, several distribution functions are available to generate synthetic source system test data sets.
- *Client:* The client application mainly includes an execution schedule. By sending messages and time-based scheduling events to the integration system, it ensures the correct scheduling. Furthermore, the client provides the autonomic benchmark execution.
- *Monitor:* The collected statistics and performance metrics are handled and stored by the Monitor. In addition to that, it also provides plotting functions for the generation of performance diagrams from the measured integration system performance.

As illustrated in Figure 6, the benchmark realization is divided into three main phases: initialization (pre), execution (work) and verification (post). Here, only the phase `work` is relevant for performance measurements, while the other phases are used for initializing the source and target systems, verifying the functional correctness of the integrated data and analyzing the measured performance events.

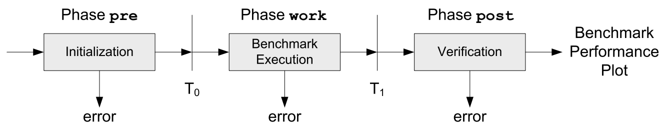

Fig. 6. Benchmark Phases

Due to its relevance, we now restrict our discussion to the phase `work`. This benchmark execution is composed of 100 benchmark periods, where each period comprises the uninitialization of all external systems, the test data initialization of the source systems and, of course, the four streams, as illustrated in Figure 7. These streams are correlated to the introduced

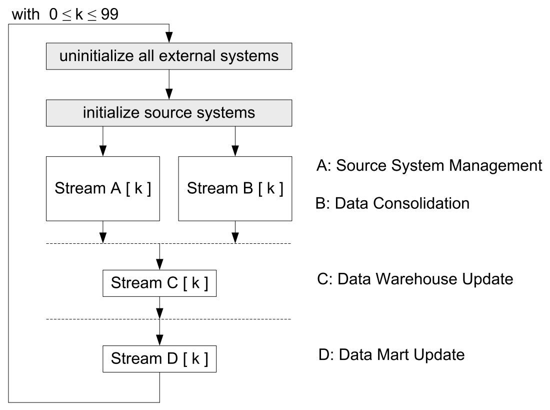

Fig. 7. Benchmark Execution Period

Group	ID	Series
A	P01	$T_0\left(Stream_k^A\right)+2\,(m-1)$ with $1 \leq m \leq \frac{(100-k)*d}{10}+1\,[tu]$
A	P02	$T_0\left(Stream_k^A\right)+2\,(m)$ with $1 \leq m \leq \frac{(100-k)*d}{10}+1\,[tu]$
A	P03	$T_1\,(P01) \wedge T_1\,(P02)$
B	P04	$T_0\left(Stream_k^B\right)+2\,(m-1)$ with $1 \leq m \leq 1100*d+1\,[tu]$
B	P05	$T_1\,(P04)$
B	P06	$T_1\,(P05)$
B	P07	$T_1\,(P06)$
B	P08	$T_0\left(Stream_k^B\right)+2000+3\,(m-1)$ with $1 \leq m \leq 900*d+1\,[tu]$
B	P09	$T_1\,(P08)$
B	P10	$T_0\left(Stream_k^B\right)+3000+2.5\,(m-1)$ with $1 \leq m \leq 1050*d+1\,[tu]$
B	P11	$T_0\left(Stream_k^B\right)$
C	P12	$T_0\left(Stream_k^C\right)$
C	P13	$T_0\left(Stream_k^C\right)+10$
D	P14	$T_0\left(Stream_k^D\right)$
D	P15	$T_1\,(P14)$

TABLE II
BENCHMARK SCHEDULING SERIES OF STREAMS A, B, C AND D

process type groups. So, such a stream should be understood as a serialized sequence of process-initiating events. These events consist of the process type ID, an execution timestamp and, in case of event type E1, an input message. Thereby, streams A and B are concurrent streams, while stream C and stream D are serialized.

The internal processing of the four benchmark streams may be influenced by specific scale factors. Basically, the three scale factors: *datasize (d^x)*, *time (t^z)* and *distribution (f^y)*, are distinguished within a three-dimensional scale space.

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 as well as, in case of event type E1, the number of process instances depend on it. In this second case, the scale factor *datasize (d^x)* influences the scheduling time series for the specific benchmark stream, as shown in Figure 8 on the left side. Thereby, the number of executed processes m depends on the specific benchmark period k and on the scale factor *datasize (d^x)*. The decreasing number of executed P01 process instances was designed with the intent to achieve a realistic scaling of master data management.

Additionally, the continuous scale factor *time (t^z)* also influences the scheduling time series. This is realized by using abstract time units (tu) for specifying the benchmark schedule series. A scale factor t^z implies $1tu = \frac{1}{z}milliseconds$. The defined scheduling series A, B, C and D define deadlines in tu for integer ranges m. The impact of this scale factor is shown in Figure 8 on the right side. An increasing t^z reduces the time interval between two successive schedule events for

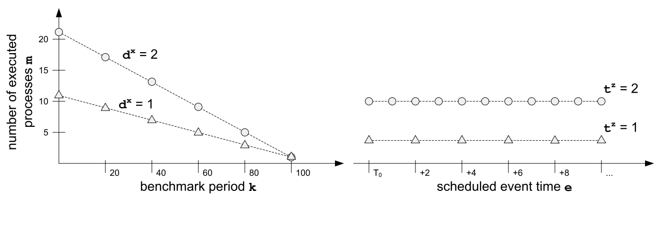

Fig. 8. P01 - Impact of Scale Factors *datasize* and *time*

this process type. A shorter interval further reduces the time for self-management and thus reduces the performance of the system. Due to the concurrent streams A and B, a shorter interval also influences the degree of parallelism.

The discrete scale factor *distribution (f^y)* is used to provide different data characteristics from uniformly distributed data values to specially skewed data values.

Note that P01 and P02 are executed in a concurrent fashion, while P03 is only executed once. Furthermore, the number of executed P01 and P02 instances depends on the number of already executed benchmark periods. In order to understand the characteristics of stream B, note the regional separation. So, there is a time shift between the European, Asian and American process type executions. However, the execution times overlap. With this schedule, real-world businesses could be modeled, where most of the business transactions take place during the core working hours. In contrast to the concurrent streams A and B, the streams C and D are serialized in order to ensure the correct results.

With the numerous different integration systems in mind, we see one of the biggest challenges in the universal comparison of benchmark performance. Therefore, we use the cost model defined in [22], where the costs of integration processes consist of the following three cost categories.

- *Communication costs $C_c(p)$:* Time; waiting for external systems (network delay and external processing costs)
- *Internal management costs $C_m(p)$:* Time; not correlated to a concrete process instance execution (plan creation and internal reorganization)
- *Processing costs $C_p(p)$:* Time; used for integration processing (all control flow oriented and data flow oriented processing steps)

Although the best suitable performance evaluation would be

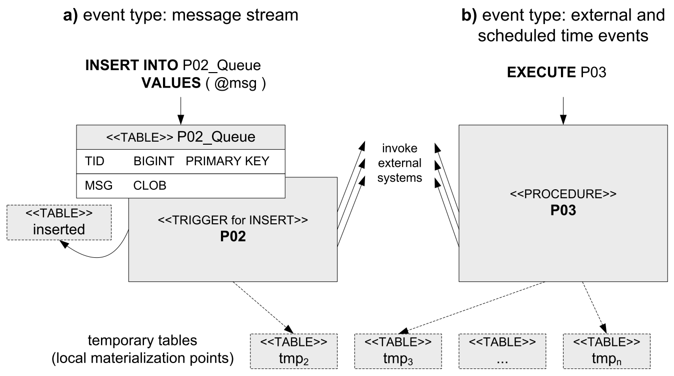
a) event type: message stream
b) event type: external and scheduled time events
INSERT INTO P02_Queue VALUES (@msg)
EXECUTE P03
<<TABLE>> P02_Queue
TID BIGINT PRIMARY KEY
MSG CLOB
<<TABLE>> inserted
<<TRIGGER for INSERT>> P02
invoke external systems
<<PROCEDURE>> P03
temporary tables (local materialization points)
<<TABLE>> tmp2
<<TABLE>> tmp3
<<TABLE>> ...
<<TABLE>> tmpn

Fig. 9. Event Type Implementation

based only on the internal management cost and the processing costs, we define that all cost categories ($C_c(p)$,$C_m(p)$ and $C_p(p)$) are included in the performance evaluation. This is acceptable because a user would rather be interested in the overall integration time than in the core processing time of the integration system. We define the benchmark performance metrics $NAVG^+(p_x)$. This metric unit represents the extended normalized average costs of a specific process type and is defined as follows:

$$NAVG^+(p_x) = NAVG(NC(p_x)) + \sigma_{p_x}(NC(p_x)).$$

Thus, this extended metric unit is computed as the sum of the average of the normalized costs of a specific process instance and of the positive standard deviation. The standard deviation is included in this metric unit in order to reward integration systems with predictable system performance.

The main problem when computing this metric unit is the determination of the normalized costs of a specific process instance. This problem is caused by (1) the parallelism of concurrent integration processes and (2) the parallelism of operator instances and subprocesses. Thus, the effective processing time could not be used to determine the costs of one single process. In order to make the costs comparable and independent of concurrent process executions, the cost normalization must be realized.

VI. Reference Implementation Experiments

In order to prove the realizability of the DIPBench specification, we experimented with a reference implementation. When we did so, we evaluated a commercial federated DBMS (called System A). In this section, we now want to point out special realization aspects and illustrate the performance plots generated by the DIPBench Toolsuite.

As already mentioned, the experimental setup for the benchmark execution comprises three independent computer systems. On the first system (ES [Athlon64 1800+, 2GB RAM]), all external systems reside. This includes one DBMS installation with eleven database instances (each with disk size 1000 MB, log size 100 MB) and one application server installation for the management of the used Web services. On the second system (IS [Athlon64 1800+, 2GB RAM]), the integration system, and thus, the system under test is located. Finally, the DIPBench Toolsuite is located on the third system (CS [Dual Genuine Intel T2400, 1.5GB RAM]). For both experiments, we used a timescale of `1.0f` and `uniform-distributed` datasets. Furthermore, the three computer systems were connected using a wireless network.

The reference implementation of the federated DBMS was realized by using System A's proprietary integration services, XML functionalities and Web service access methods. Actually, the used system was homogeneous with regard to the used external database systems. Now, we will point out the major implementation aspects only. Basically, two different event types of integration processes have to be distinguished. Figure 9 illustrates the core realization concepts. In case of the event type `message stream` (a), where processes are

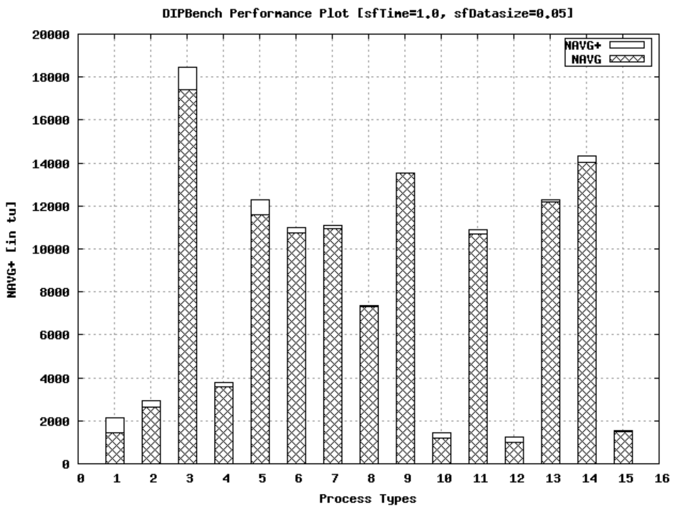

Fig. 10. Reference Implementation Performance Results ($d^x = 0.05$)

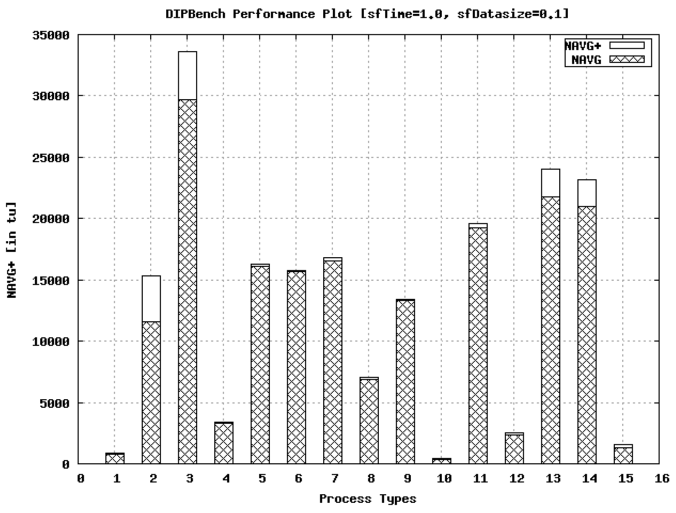

Fig. 11. Reference Implementation Performance Results ($d^x = 0.1$)

initiated in a data-driven way, a set of components is used. First, a simple table is used for queuing incoming messages. Second, the actual integration process is implemented as an insert-trigger, evaluating the logical table `inserted`. Notice that all integration processes starting with a `RECEIVE` operator are realized in such a way. In case of the event type `time events` (b), the integration process could be realized as a stored procedure because no data input—except for global configuration parameters—is needed.

Figure 10 shows the DIPBench performance plot for the discussed federated DBMS reference implementation using a datasize of $d^x = 0.05$. There are some results which have to be explained. First, the large NAVGP difference between the serialized, data-intensive processes and the highly concurrent processes should be noticed. Furthermore, the data-intensive processes have a higher standard deviation. This is caused by a smaller number of executed process instances but also by internal optimization techniques. The data-intensive processes are realized with relational operators and thus could be well-optimized. In contrast to this, the concurrent processes are realized using proprietary XML functionalities, which are apparently not included in the optimizer. Figure 11 illustrates the second experiment using a datasize $d^x = 0.1$. In particular, the influence on the process types initiated by event type E1 should be noticed. In contrast to these, the process types following event type E2, were only executed more often and thus show a decreased standard deviation rather than higher normalized costs.

VII. Summary and Future Work

Basically, the need for an independent integration benchmark was the motivation for the development of the discussed DIPBench. The initial situation comprised several challenges and problems, which have caused the lack of such a benchmark. The different integration system types, the different event models and functional properties, but also the need for an adequate performance evaluation of highly concurrent scenarios belong to these challenges. Starting from this point, we first defined a benchmark scenario, including several systems and schemas. Second, a realistic and well-balanced process mix was specified in a platform-independent manner. In this context, a set of 15 process types as well as an execution schedule was described in detail. Third, a performance metric unit for evaluating the benchmark performance was also introduced. Fourth, this metric unit was used to determine the performance of the reference implementation, based on a federated DBMS. Finally, we want to point out that the whole benchmark description is available in [25]. In order to minimize the time and effort for executing the benchmark, a sophisticated toolsuite [12] is provided. To summarize this paper, it remains to be said that we defined the first integration process performance benchmark, called DIPBench. We hope that it will be used by research groups and system vendors in order to provide comparability concerning the system performance. Since we believe in the importance of such a benchmark, we would like to discuss open issues, problems and challenges related to this benchmark.

Our future work, correlated to this benchmark, comprises further reference benchmark implementations. For example, we currently realize experiments with EAI servers and ETL tools. In addition to that, we want to enhance the benchmark by integrating quality and semantic issues as well as further types of functional challenges.

References

[1] S. Dessloch, A. Maier, N. Mattos, and D. Wolfson, “Information integration - goals and challenges,” *Datenbank-Spektrum*, vol. 3, no. 6, pp. 7–13, 2003.
[2] 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.
[3] A. Y. Halevy, N. Ashish, D. Bitton, M. J. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka, “Enterprise information integration: successes, challenges and controversies.” in *SIGMOD*, 2005, pp. 778–787.
[4] Z. G. Ives, A. Y. Halevy, and D. S. Weld, “Adapting to source properties in processing data integration queries.” in *SIGMOD*, 2004, pp. 395–406.
[5] M. Böhm, U. Wloka, D. Habich, J. Bittner, and W. Lehner, “A messsage transformation model for data centric integration processes,” University of Applied Sciences Dresden,” Technical Report, 2007.
[6] J. Gray, Ed., *The Benchmark Handbook for Database and Transaction Systems (2nd Edition).* Morgan Kaufmann, 1993.
[7] J. Hammer, M. Stonebraker, and O. Topsakal, “Thalia : Test harness for the assessment of legacy information integration approaches,” University of Florida,” Technical Report, 2005.
[8] ——, “Thalia: Test harness for the assessment of legacy information integration approaches.” in *ICDE*, 2005, pp. 485–486.
[9] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S. Weld, “An adaptive query execution system for data integration.” in *SIGMOD*, 1999, pp. 299–310.
[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. Böhm, D. Habich, W. Lehner, and U. Wloka, “Dipbench toolsuite: A framework for benchmarking integration systems (demo),” in *ICDE*, 2008.
[13] M. Nicola, I. Kogan, and B. Schiefer, “An xml transaction processing benchmark.” in *SIGMOD Conference*, 2007, pp. 937–948.
[14] R. Othayoth and M. Poess, “The making of tpc-ds.” in *VLDB*, 2006, pp. 1049–1058.
[15] M. Pöss, B. Smith, L. Kollár, and P.-Å. Larson, “Tpc-ds, taking decision support benchmarking to the next level.” in *SIGMOD*, 2002, pp. 582–587.
[16] *TPC-DS - ad-hoc, decision support benchmark*, Transaction Processing Performance Council, 2007.
[17] *High Performance Extract/Transform/Load Benchmark*, RODIN Data Asset Management, 2002.
[18] P. Vassiliadis, A. Karagiannis, V. Tziovara, and A. Simitsis, “Towards a benchmark for etl workflows,” in *QDB*, 2007, pp. 49–60.
[19] J. Darmont, O. Boussaid, and F. Bentayeb, “Dweb: A data warehouse engineering benchmark,” in *DaWaK*, 2005, pp. 85–94.
[20] J. Darmont, F. Bentayeb, and O. Boussaid, “Benchmarking data warehouses,” vol. 2, no. 1, 2007, pp. 79–104.
[21] *SPECjms2007 Benchmark*, http://www.spec.org/jms2007/, 2007.
[22] M. Böhm, D. Habich, U. Wloka, J. Bittner, and W. Lehner, “Towards self-optimization of message transformation processes,” in *ADBIS*, 2007.
[23] O. Becker, “Extended sax filter processing with stx.” in *Extreme Markup Languages*, 2003.
[24] ——, “Streaming transformations for xml-stx.” in *XMIDX*, 2003, pp. 83–88.
[25] *DIPBench*, Dresden University of Technology, Database Technology Group, http://wwwdb.inf.tu-dresden.de/research/gcip, 2007.