ISSN 0873-9293

DBench – OLTP

A Dependability Benchmark for OLTP Application Environments

Marco Vieira
Henrique Madeira

Relatórios Internos
DEI
Nº 006-2002

Coimbra - 2002

Título: Relatórios Internos do DEI **ISSN: 0873-9293**
Capa: Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Edição: Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

DBench – OLTP

A Dependability Benchmark for OLTP Application Environments

Benchmark Specification

Draft 1.2.0

Updated on January 24, 2004

DBench Project

www.dbench.org

Acknowledgments

Funding for this work was provided, in part, by the Portuguese Government/European Union through R&D Unit 326/94 (CISUC) and by the DBench project, IST-2000-25425 DBENCH, funded by the European Union.

DBench Consortium

University of Coimbra (**Portugal**)
LAAS-CNRS (**France**)
Polytechnical University of Valencia (**Spain**)
Chalmers University of Technology (**Sweden**)
Critical Software (**Portugal**)
Friedrich Alexander University, Erlangen-Nürnberg (**Germany**)

Document History

Date	Version	Description
December 17, 2002	Draft 1.0.0	First benchmark specification released in the Web
April 21, 2003	Draft 1.1.0	Clarification of the difference between Draft versions (i.e., versions under validation) and Revisions (i.e., versions ready to use). The addendum to TPC-C style was removed. Configuration of the SUT in Phase 1 and Phase 2 was clarified. The measure "system performance decreasing ration" was removed.
January 24, 2004	Draft 1.2.0	

TABLE OF CONTENTS

Clause 0: PREAMBLE

0.1. Introduction

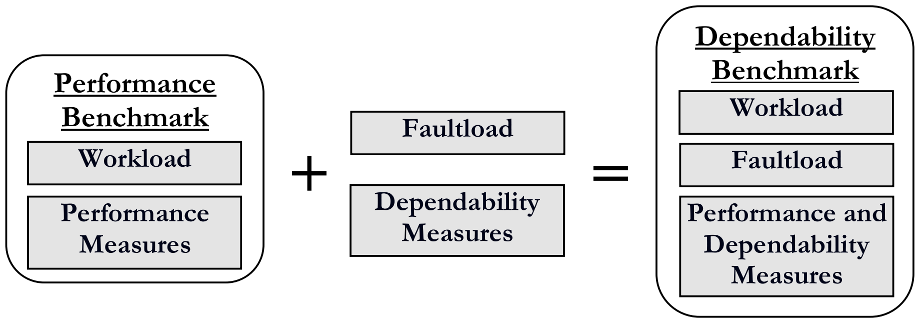
The **DBench-OLTP** is a dependability benchmark for On-Line Transaction Processing (OLTP) systems. These systems constitute the kernel of the information systems used today to support the daily operations of most of the business. Some examples include banking, e-commerce, insurance companies, all sort of traveling businesses, telecommunications, wholesale retail, complex manufacturing processes, patient registration and management in large hospitals, libraries, etc.

A **dependability benchmark** is a specification of a standard procedure to assess dependability related measures of a computer system or computer component.

Transactional systems industry holds a reputed infrastructure for performance evaluation and the set of benchmarks managed by the **Transaction Processing Performance Council** (TPC) are recognized as one of the most successful benchmark initiatives of the overall computer industry. Concerning the OLTP application environments, the **TPC Benchmark^{TM} C** (TPC-C) is one of the most important and well-established performance benchmarks. The DBench-OLTP dependability benchmark extends the TPC-C proposal to evaluate both dependability and performance in OLTP systems.

The main components of the DBench-OLTP dependability benchmark are (see also figure below):

- **Workload**: represents the work that the system must perform during the benchmark run. The DBench-OLTP benchmark uses the workload of the TPC-C benchmark (revision 5.0 available at www.tpc.org/tpcc).
- **Faultload**: represents a set of faults and stressful conditions that emulate real faults experienced by OLTP systems in the field. The DBench-OLTP faultload can be based on software faults, operator faults or high-level hardware failures. A general faultload that combines the three classes is also possible (see Clause 4).
- **Measures**: characterize the performance and dependability of the system under benchmark (SUB) in the presence of the faultload when executing the workload (see Clause 3).



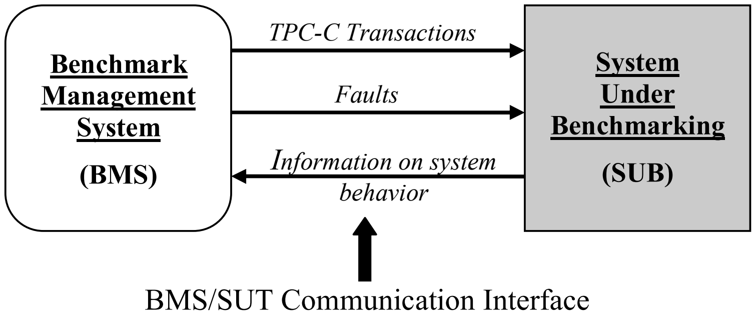
The DBench-OLTP dependability benchmark presented in this document uses the workload and the general setup of the TPC-C benchmark (revision 5.0 available at www.tpc.org/tpcc), and contains explicit references to TPC-C clauses. In order to implement and run the DBench-OLTP dependability benchmark, the benchmark performer must use both the present specification and the TPC-C specification (revision 5.0).

Clause 1: BENCHMARK SETUP

1.1. Test Configuration

The following figure presents the test configuration required to run the DBench-OLTP dependability benchmark. As in TPC-C standard benchmark (Clause 6.2), the main elements are:

- System Under Benchmarking (SUB)
- Benchmark Management System (BMS)
- BMS/SUT Communications Interface.


Benchmark Management System (BMS)
TPC-C Transactions
Faults
Information on system behavior
System Under Benchmarking (SUB)
BMS/SUT Communication Interface

Comment: Note that, in order to comply with the terminology and benchmarking concepts defined in the ambit of the DBENCH project, the terminology used in this specification differs from the one used in the TPC-C benchmark. This way, the System Under Benchmarking (SUB) corresponds to the TPC-C System Under Test (SUT), and the Benchmark Management System (BMS) corresponds to the TPC-C Driver System.

1.2. System Under Benchmarking (SUB) Definition

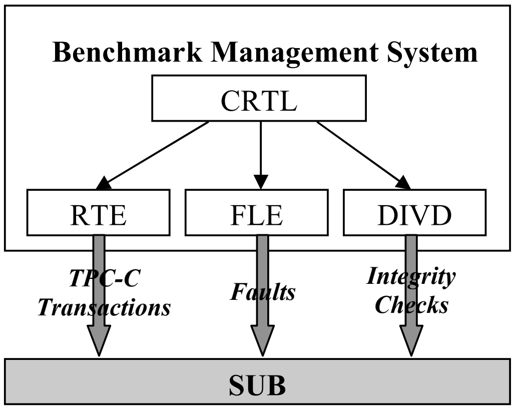
The SUB definition used in DBench-OLTP is similar to the SUT definition presented in Clause 6.3 of TPC-C.

1.3. Benchmark Management System Definition

1.3.1. The external BMS is composed by several modules, which must perform the following functions:

1. Control the benchmarking process (CTRL module).
2. Provide Remote Terminal Emulator functionalities (RTE module).
3. Provide Faultload Emulator functionalities (FLE module).
4. Detect data integrity violations (DIVD module).

1.3.2. The CRTL is responsible for controlling all the benchmarking process as presented in Clause 2.



1.3.3. The RTE is presented in Clause 6.4 of TPC-C.

1.3.4. The FLE is responsible for injecting the faultload (see Clause 4 of DBench-OLTP). The fault injection corresponds to the artificial insertion of software faults, operator faults and high-level hardware failures into a system or component, and is used to evaluate specific fault tolerance mechanisms and to assess the impact of faults in systems.

1.3.5. The DIVD is responsible for performing the integrity checks in order to detect any data integrity violations caused by the fault injection (see Clause 2.4).

1.4. BMS/SUB Communications Interface Definition

1.4.1. The RTE/SUB Communications Interface definition is presented in Clause 6.5 of the TPC-C standard specification.

1.4.2. The FLE must inject the faults using only the standard interfaces (API, SQL, etc.) provided by the SUB. No extensions to the SUT are permitted in order to facilitate/allow the injection of faults except the ones performed by the tools provided with this specification.

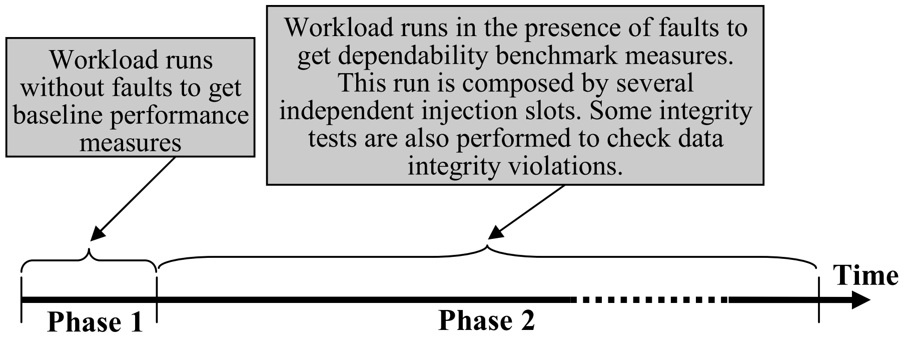
1.4.3. The DIVD must perform the integrity checks using only the standard interfaces (API, SQL, etc.) provided by the SUB.

Clause 2: BENCHMARKING PROCEDURE

2.1. Benchmarking Procedure

2.1.1. The benchmark procedure is controlled by the CRTL module.

2.1.2. A benchmark run includes two main phases, as shown in the following figure.


Workload runs without faults to get baseline performance measures
Workload runs in the presence of faults to get dependability benchmark measures. This run is composed by several independent injection slots. Some integrity tests are also performed to check data integrity violations.
Time
Phase 1
Phase 2

2.1.3. **Phase 1** - First run of the TPC-C workload without any (artificial) faults. This run is used to collect the baseline performance measures.

2.1.4. **Phase 2** – In this phase the TPC-C workload is run in the presence of the faultload to measure the impact of faults on the system dependability.

2.1.5. The configuration of the SUB must be exactly the same in both phases. The use of a configuration optimized for pure performance in phase 1 and a different configuration optimized for recovery in phase 2 is not allowed.

2.1.6. During the benchmark run, phases 1 and 2 must be completely automatic and performed without any user intervention.

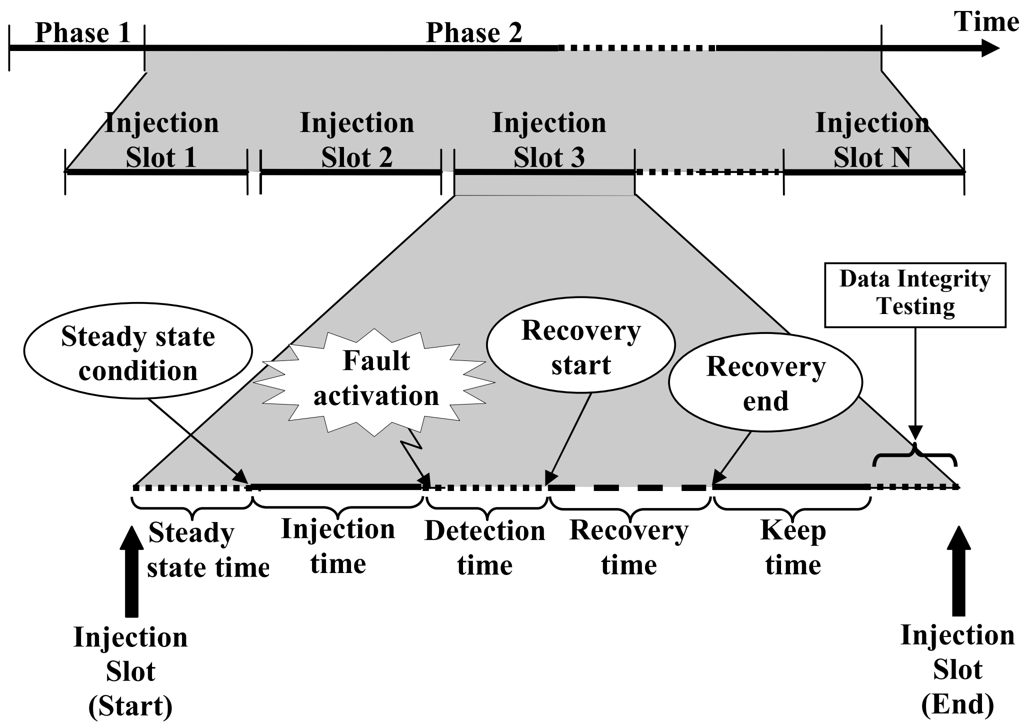
2.1.7. All the implementation details concerning the CRTL must be disclosed in the Full Disclosure Report (see Clause 5).

2.2. Phase 1 Requirements

2.2.1. Phase 1 corresponds to a TPC-C measurement interval (as defined in Clause 5.1.1 of TPC-C), and must follow the requirements specified in Clause 5.5 of the TPC-C standard specification.

2.3. Phase 2 Requirements

2.3.1. The Phase 2 interval is composed by several independent injection slots, as show in the figure below. An **injection slot** can be defined as a measurement interval during which the TPC-C workload is run and one or more faults from the faultload are injected in order to evaluate the system behavior.



2.3.2. The SUB state must be explicitly restored in the beginning of each injection slot and the effects of the faults cannot accumulate across different slots.

2.3.3. The test in each injection slot must be conducted in a steady state condition as defined in Clause 5.5.1 of TPC-C. The system achieves a steady state condition after a given time executing transactions (steady state time).

2.3.4. Some of the partial times that compose an injection slot (injection time, detection time, and keep time) are specific for each fault. Those times are defined in Clause 4.2.

2.3.5. The FLE must inject the fault a certain amount of time (injection time) after the steady state condition has been achieved. Note that, for operator faults and high-level hardware failures only one fault is injected. On the other hand, ten software faults are injected in each injection slot.

2.3.6. Due to the fact that for some types of faults the time needed to detect the effects of a fault is highly human dependent, a typical detection time must be considered for each fault (defined in clause 4.2.2). After that detection time the FLE must start an error detection procedure to evaluate the effects of the fault (i.e., detect if an error occurred).

2.3.7. If an error is detected by the error detection procedure, the FLE must evaluate and start the required recovery procedure. The recovery time represents the time that the system needs to execute the recovery procedure. If no error is detected or no recovery procedure is needed, then the recovery time is not considered (equal to zero).

Comment: Note that, for software faults the recovery procedure must be always executed independently of the visible effects of the faults injected. The reason is that some software faults can have effects that cannot be directly detected by the error detection procedure (e.g., performance degradation, dead-locks, etc.).

2.3.8. When the recovery procedure completes, the workload must continue running during a keep time (defined in clause 4.2.2) in order to evaluate the impact of the fault on the system. Note that, even if no recovery procedure is needed, the workload should also continue running during this keep time.

2.3.9. After the workload end, a set of application consistency tests must be performed to check possible data integrity violations caused by the fault injected. The integrity testing requirements are specified in Clause 2.4.

2.3.10. The duration of each injection slot depends on the fault to be injected and the correspondent partial times (steady state time, injection time, detection time, recovery time, and keep time). However, the workload must run for at least 15 minutes after the steady state condition has been achieved (15 minutes + steady state time).

2.3.11. The duration of Phase 2 is dependent on the number of faults (and subsequent injection slots) that composes the faultload (see Clause 4.3).

2.4. Integrity Testing Requirements

2.4.1. The integrity tests are performed by the DIVD module, which provides the ability to detect any data integrity violations caused by the fault injection during Phase 2.

2.4.2. The integrity checks must be performed on the application data (i.e., the data in the database tables after running the workload during a given injection slot) and must use all the business rules defined in the TPC-C specification.

2.4.3. Integrity tests must be performed in addition to the normal integrity test included in the workload and in the database engine that may also detect integrity errors during the injection slot.

2.4.4. All the implementation details concerning the DIVD must be disclosed in the Full Disclosure Report (see Clause 5).

Clause 3: MEASURES

3.1. Measures Definition

3.1.1. The DBench-OLTP dependability benchmark is composed by three sets of measures: baseline performance measures, performance measures in the presence of the faultload, and dependability measures.

3.1.2. The **baseline performance measures** reported by this dependability benchmark are specified in Clause 5.5 (transactions-per-minute-C (**tpmC**)) and Clause 7 (price-per-tpmC **($/tpmC)**) of the TPC-C standard specification.

Comment: In the context of this dependability benchmark, the baseline performance measures represent the baseline performance instead of the optimized performance (as is the case of TPC-C). The benchmark performer should decide on the best configuration of the SUB in order to achieve a good compromise between performance and dependability. Note that the same configuration of the SUB must be used in both phases (see Clause 2.1.5).

3.1.3. The **performance measures in the presence of the faultload** are:

1. Number of transactions executed per minute in the presence of the faults specified in the faultload (see Clause 4) during Phase 2 (**Tf**). It measures the impact of faults in the performance and favors systems with higher capability of tolerating faults, fast recovery time, etc.
2. Price per transaction in the presence of faults specified in the faultload during Phase 2 (**$/Tf**). It measures the (relative) benefit of including fault handling mechanisms in the target systems in terms of the price.

3.1.4. The **dependability measures** reported by this dependability benchmark are:

1. Number of data errors detected by the consistency tests and metadata tests (**Ne)**. It measures the impact of faults on the data integrity.
2. Availability from the SUB point-of-view in the presence of the faultload during Phase 2 (**AvtS)**. It measures of system availability during Phase 2 from the system under benchmarking point-of-view. The system is available when it is able to respond to at least one terminal within the minimum response time for each

transaction (defined in Clause 5.2.5.3 of TPC-C). The system is unavailable when it is not able to respond to any terminal.

3. Availability from the RTE point-of-view in the presence of the faultload during Phase 2 (**AvtR).** It measures of system availability during Phase 2 from the end-users (RTE terminals) point-of-view. The system is available for one terminal if it responds to a submitted transaction within the minimum response time for that type of transaction (defined in Clause 5.2.5.3 of TPC-C). The system is unavailable for that terminal if there is no response within that time or if an error is returned. In this case, the unavailability period must be counted from the moment when a given client submits a transaction that fails until the moment when it submits a transaction that succeeds.

3.2. Measures Computation

3.2.1. The baseline performance measures (**tpmC** and **$/tpmC)** are related only with Phase 1 interval, and must be calculated as stated in Clauses 5.4 and 7 of TPC-C.

3.2.2. The number of transactions executed per minute in the presence of faults (**Tf**) is given by the following expression:

$$\mathbf{Tf = \sum Te(i) / \sum T(i)}$$

Where:

Te(i) - is the number of transactions executed in the injection slot *i*.

T(i) – is the duration time of the injection slot *i*. This time is counted from the moment when the system achieves the steady state condition (end of steady state time) until the end of the keep time.

3.2.3. The price per transaction in the presence of faults (**$/Tf**) is given by the price of the system divided by the number of transactions executed per minute in the presence of faults (**Tf**). The pricing rules specified in Clause 7 of the TPC-C standard specification must also be applied in the computation of this measure.

3.2.4. The system performance decreasing ratio due to faults (**Tf/tpmC**) is given by the number of transactions executed per minute in the presence of the faultload (**Tf**) divided by the baseline performance (**tpmC**).

3.2.5. The number of data errors detected by the consistency tests and metadata tests (**Ne**) is a direct measure obtained at the end of each injection slot by the DIVD module (see Clause 2.4).

3.2.6. The availability from the SUB point-of-view (**AvtS**) is given by the following expression:

$$\mathbf{AvtS = (\sum (T(i) - UnavS(i)) / \sum T(i)}$$

Where:

T(i) – is the duration time of the injection slot *i*. This time is counted from the moment when the system achieves the steady state condition (end of steady state time) until the end of the keep time.

UnavS(i) - is the amount of time the system was unable to respond to any terminal during the injection slot *i* (see Clause 3.1.4.2).

3.2.7. The availability from the RTE point-of-view (**AvtR**) is given by the following expression:

$$\mathbf{AvtS = (\sum (T(i)*Nt - \sum UnavR(i,j))) / \sum T(i)*Nt}$$

Where:

T(i) – is the duration time of the injection slot *i*. This time is counted from the moment when the system achieves the steady state condition (end of steady state time) until the end of the keep time.

Nt – is the total number of terminal submitting transactions to the SUB in each injection slot.

UnavR(i,j) - is the amount of time the system was unable to respond within the minimum response time to any kind of transaction submitted by the terminal *j* during the injection slot *i* (see Clause 3.1.4.3).

3.2.8. During the Phase 2, the benchmark management system must collect the adequate amount of information about the system behavior, in order to allow the computation of the dependability measures (**Ne**, **AvtR, and AvtS**).

3.2.9. All the information collected to the computation of the dependability measures has to be disclosed in the Full Disclosure Report (see Clause 5).

5.2.5. All the information concerning the fault injection, such as techniques and interfaces, must be disclosed in the Full Disclosure Report.

5.2.6. All the information concerning the faultload definition used has to be disclosed in the Full Disclosure Report.

5.2.7. All the DBench-OLTP implementation details must be disclosed. All the information regarding the following modules implementation must be included in the Full Disclosure Report: CRTL, FLE, and DIVD. The FLE implementation is disclosed in the TPC-C Full Disclosure Report.