



DEPARTAMENTO DE
ENGENHARIA
INFORMÁTICA

DBench – OLTP

A Dependability Benchmark for OLTP Application Environments

Marco Vieira
Henrique Madeira

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

© 2002 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 1.1.0

Updated on April 21, 2003

DBench Project

www.dbench.org

© 2002 Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

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	<ul style="list-style-type: none"> • 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.

TABLE OF CONTENTS

Clause 0: PREAMBLE	3
0.1. Introduction.....	3
Clause 1: BENCHMARK SETUP	4
1.1. Test Configuration.....	4
1.2. System Under Test (SUT) Definition.....	4
1.3. Driver System Definition.....	4
1.4. Driver System/SUT Communications Interface Definition.....	5
Clause 2: BENCHMARKING PROCEDURE	6
2.1. Benchmarking Procedure.....	6
2.2. Phase 1 Requirements.....	6
2.3. Phase 2 Requirements.....	6
2.4. Integrity Testing Requirements.....	8
Clause 3: MEASURES	9
3.1. Measures Definition.....	9
3.2. Measures Computation.....	10
Clause 4: FAULTLOAD	12
4.1. Fault Injection.....	12
4.2. Fault Types Definition.....	13
4.3. Faultload Definition.....	13
Clause 5: FULL DISCLOSURE	15
5.1. Disclosure Report Goal.....	15
5.2. Disclosure Report Requirements.....	15

Clause 0: PREAMBLE

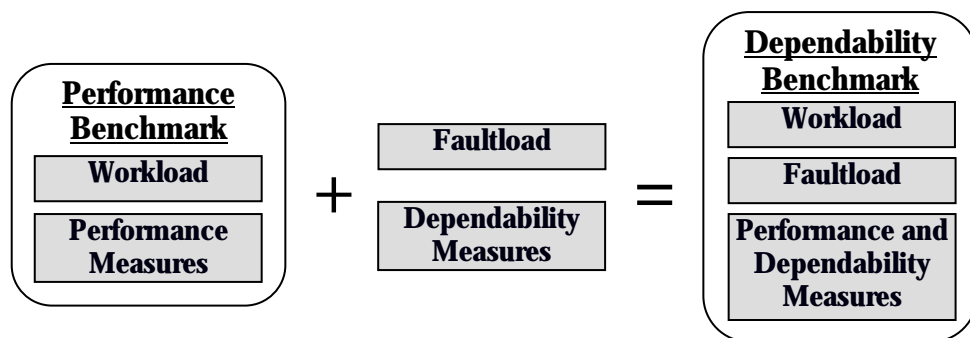
0.1. Introduction

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™ 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 TPC-C benchmark includes two major components: a workload and a set of performance measures. The DBench-OLTP dependability benchmark adds two new elements: 1) measures related to dependability; and 2) a faultload (see figure below). The faultload represents a set of faults and stressful conditions that emulate real faults experienced by OLTP systems in the field. The measures characterize the dependability features of the system under benchmark in the presence of the faultload.



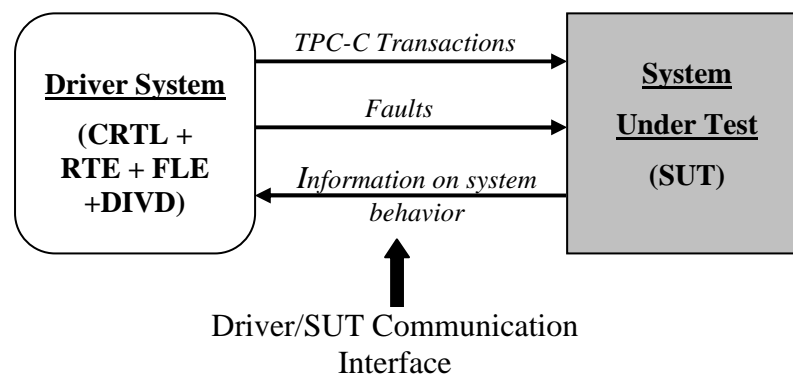
The DBench-OLTP dependability benchmark presented in this document uses the workload and the general setup of the TPC-C benchmark (revision 5.0 available at www.tpc.org/tpcc), and contains explicit references to TPC-C clauses. In order to implement and run the DBench-OLTP

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: the System Under Test (SUT), the Driver System, and the Driver/SUT Communications Interface.



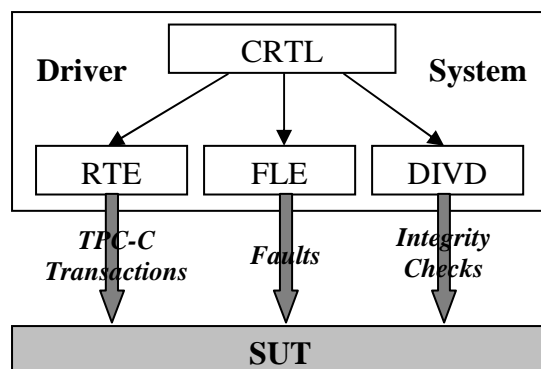
1.2. System Under Test (SUT) Definition

The SUT definition used in DBench-OLTP is the same as the one presented in Clause 6.3 of TPC-C.

1.3. Driver System Definition

1.3.1. The external Driver System 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 faults 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. Driver System/SUT Communications Interface Definition

- 1.4.1. The RTE/SUT 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 SUT. No extensions to the SUT are permitted in order to facilitate/allow the injection of faults.¹
- 1.4.3. The DIVD must perform the integrity checks using only the standard interfaces (API, SQL, etc.) provided by the SUT.

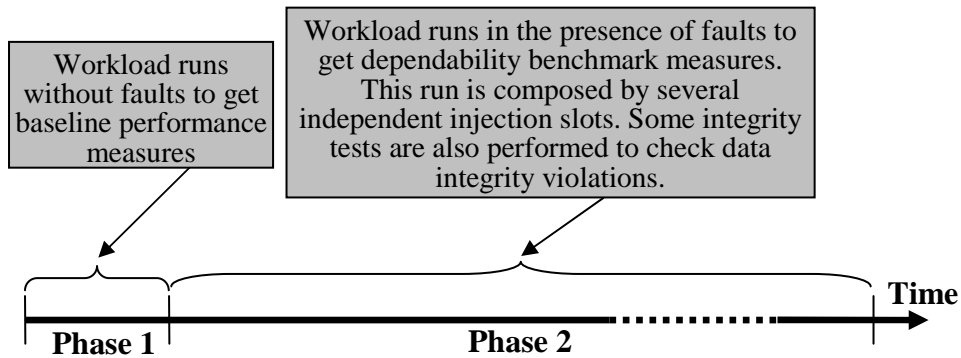
¹ Future versions of the DBench-OLTP may include specific extensions to the SUT in order to inject additional classes of faults. The current version does not allow any change in the SUT.

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.



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 SUT 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.

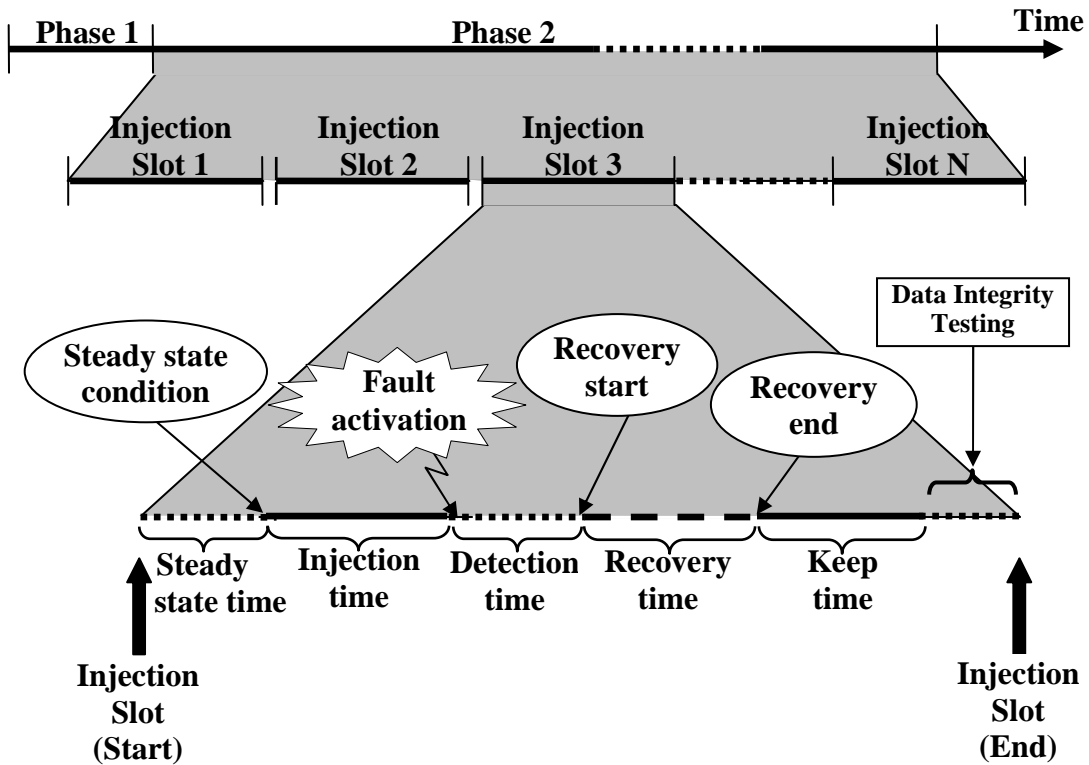
2.1.7. All the implementation details concerning the CRTL must be disclosed in the Full Disclosure Report (see Clause 5).

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 fault from the faultload is injected in order to



evaluate the system behavior.

2.3.2. The SUT 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.

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).
- 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 SUT in order to achieve a good compromise between performance and dependability. Note that the same configuration of the SUT 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 SUT 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 test 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:

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

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 SUT point-of-view (**AvtS**) is given by the following expression:

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

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:

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

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 SUT 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).

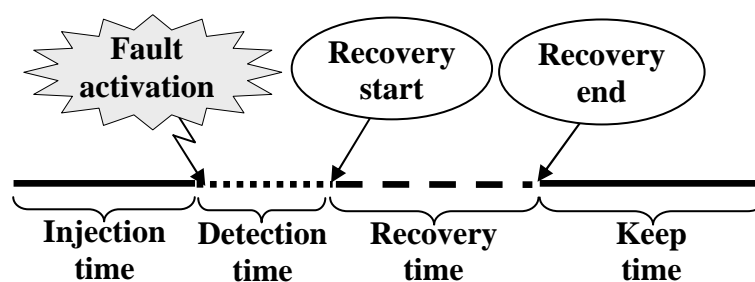
3.2.8. During the Phase 2 the driver system must collect the adequate amount of information about the system behavior, in order to allow the computation of the dependability measures (**Ne, AvtR, and AvtS**).

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).

Clause 4: FAULTLOAD

4.1. Fault Injection

- 4.1.1. Faults are injected by the FLE. The fault injection corresponds to the artificial insertion of faults into the system under test.
- 4.1.2. The following figure presents the set of steps the FLE must execute in each injection slot to inject a fault. The concept of injection slot is presented in Clause 2.3.2.



- 4.1.3. The FLE must inject the fault in a given moment known as the injection time.
- 4.1.4. After the injection of the fault, the FLE must wait for the detection of the fault (detection time). Due to the fact that for some types of faults (e.g., operator faults) the time needed to detect the effects of a fault is highly human dependent, a typical detection time is considered for each fault. After the detection time the FLE must start an error detection procedure to evaluate the effects of the fault on the SUT.
- 4.1.5. 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 the system needs to execute the recovery procedure. If no error is detected then the recovery time is not considered.
- 4.1.6. When the recovery procedure completes, the workload must continue running during a keep time in order to evaluate the system behavior. Note that, even if no recovery procedure is needed, the workload should also continue running during this keep time.
- 4.1.7. The several partial times that compose an injection slot are specific for each fault. The detection time and the keep time are defined in Clause 4.2.2. The injection time is defined in Clause 4.3.
- 4.1.8. All the information concerning the fault injection and the FLE implementation has to be disclosed in the Full Disclosure Report (see Clause 5).

4.2. Fault Types Definition

4.2.1. The types of faults to be considered in the faultload are:

1. Abrupt operating system shutdown
2. Abrupt transactional engine shutdown
3. Kill set of user sessions
4. Delete table
5. Delete user schema (or all objects belonging to a given user)
6. Delete file from disk
7. Delete set of files from disk
8. Delete all files from one disk

4.2.2. The detection time and keep time to consider for each fault are presented in the table below.

Fault Type	Detection Time	Keep Time
Abrupt operating system shutdown	0 sec.	5 min.
Abrupt transactional engine shutdown	30 sec.	5 min.
Kill a set of user sessions	0 sec.	5 min.
Delete a table	2 min.	5 min.
Delete a user schema	1 min.	5 min.
Delete a file from disk	4 min.	5 min.
Delete a set of files from disk	2 min.	5 min.
Delete all files from a disk	1 min.	5 min.

4.3. Faultload Definition

4.3.1. The faultload is composed of several faults from the types presented in Clause 4.2.1, injected in different instants (i.e., injection times). The following clauses describe how to select the faults to be injected and the correspondent injection time.

4.3.2. Ten faults from the type “*Abrupt operating system shutdown*” must be injected considering the following injection times: 3, 5, 7, 9, 10, 11, 12, 13, 14, and 15 minutes.

4.3.3. Ten faults from the type “*Abrupt transactional engine shutdown*” must be injected considering the following injection times: 3, 5, 7, 9, 10, 11, 12, 13, 14, and 15 minutes.

4.3.4. Five faults from the type “*Kill set of user sessions*” must be injected considering the following injection times: 3, 7, 10, 13, and 15 minutes. The set of sessions to be killed in each fault injection must be randomly selected during the benchmark run and consists of 50% of all the active sessions from the users holding the TPC-C tables.

4.3.5. Three faults from the type “*Delete table*” must be injected for each one of the following TPC-C tables: *order*, *new-order*, *order-line*, and *ware* (a total of 12 faults). The injection times to be considered are the following: 3, 10, and 15 minutes.

4.3.6. Three faults from the type “*Delete user schema*” must be injected using the following injection times: 3, 10, and 15 minutes. The user to be considered is the one that holds the TPC-C tables. If the objects are distributed among several users then the user holding the greater number of TPC-C tables must be selected. If the removal of a user is not allowed (due to a system limitation) then all objects bellowing to the user must be deleted separately.

4.3.7. The set of faults from the type “*Delete file from disk*” to be injected must be defined performing the following steps:

For each TPC-C table:

- 1) Select randomly 10% of the disk files containing data from the TPC-C table being considered (in a minimum of 1).
- 2) Inject 3 faults for each disk file selected before, using the following injection times: 3, 10, and 15 minutes.

4.3.8. Three faults from the type “*Delete a set of files from disk*” must be injected for each set of files containing each TPC-C table (a total of 27 faults). The injection times to be considered are the following: 3, 10, and 15 minutes.

4.3.9. The set of faults from the type “*Delete all files from one disk*” to be injected must be defined performing the following steps:

- 1) Select randomly 10% of the disks containing data from any TPC-C table (in a minimum of 1).
- 2) Inject 3 faults for each disk selected before, using the following injection times: 3, 10, and 15 minutes.

4.3.10. All the information concerning the faultload definition used has to be disclosed in the Full Disclosure Report (see Clause 5).

Clause 5: FULL DISCLOSURE

5.1. Disclosure Report Goal

- 5.1.1. A Full Disclosure Report is required in order for results to be considered compliant with the DBench-OLTP dependability benchmark specification.
- 5.1.2. The goal of the DBench-OLTP Full Disclosure Report is to allow reproducing the experiments in other sites using the same products.

5.2. Disclosure Report Requirements

- 5.2.1. The DBench-OLTP Full Disclosure Report must be provided together with by the TPC-C Full Disclosure Report (specified in Clause 8 of the TPC-C standard specification).
- 5.2.2. In order to make easy for the readers to compare and contrast material in different disclosure reports, the order and titles of sections in the Full Disclosure report must correspond, as much as possible, with the order and titles of sections from the DBench-OLTP standard specification.
- 5.2.3. The following measures must be summarized in the beginning of the DBench-OLTP Full Disclosure Report:
 - 1. **tpmC** – number of TPC-C transactions executed per minute (TPC-C measure)
 - 2. **\$/tpmC** – Price per TPC-C transaction executed (TPC-C measure).
 - 3. **Tf** – Number of transactions executed per minute in the presence of the faults specified in the faultload during Phase 2.
 - 4. **\$/Tf** – Price per transaction in the presence of faults specified in the faultload during Phase 2.
 - 4. **Ne** – Number of data errors detected by the consistency tests and metadata tests.
 - 5. **AvtS** – Availability from the SUT point-of-view in the presence of the faults specified in the faultload during Phase 2.
 - 6. **AvtR** – Availability from the RTE point-of-view in the presence of the faults specified in the faultload during Phase 2.
- 5.2.4. All the information collected to the computation of the benchmark measures must be disclosed in the Full Disclosure Report.

- 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.