Preliminary Dependability Benchmark Framework

Presenter:
Henrique Madeira
University of Coimbra
Outline

- Motivation and key issues on dependability benchmarking
- Dependability benchmarking overview
- Discussion of the different dimensions of the problem
- Dependability benchmarking scenarios
- Some examples of preliminary dependability benchmarks
- Discussion
**Dependability benchmarks: what for?**

Dependability benchmarking must provide a practical, uniform, comparable, repeatable, and cost effective way of evaluating dependability attributes.
Dependability benchmarks: what for?

• **Help to improve computer systems** (in a direct way)
  – Identify malfunctioning or less robust parts needing improvements;
  – Tune a particular component to enhance its dependability;
  – Improve/tuning the system architecture to ensure a given dependability level;
  – Tune a system for optimal performance and dependability.

• **Assess** the dependability of a component or a computer system.

• **Compare**, grade or rank the dependability of alternative or competitive solutions.
**Dependability benchmarks: what for?**

- **Help to improve computer systems** (in a direct way)
  - Identify malfunctioning or less robust parts needing improvements;
  - Tune a particular component to enhance its dependability;
  - Improve/tuning the system architecture to ensure a given dependability level;
  - Tune a system for optimal performance and dependability.

- **Assess** the dependability of a component or a computer system.

- **Compare**, grade or rank the dependability of alternative or competitive solutions.

**Goals of DBench**

- Define framework for dependability benchmarking
- Show how to specify dependability benchmarks
- Benchmark prototypes
  - Operating Systems
  - Embedded systems
  - Transactional systems
Key issues on dependability benchmarking

- **Representativeness**
  - A benchmark must be representative.
  - Representativeness affects nearly all aspects of benchmarking.
  - As an abstraction, a benchmark is always an imperfect representation of reality.

- **Usefulness**
  - Find useful representations that capture the essential elements and:
    - Allow practical ways to characterise the computer/components dependability;
    - Help the vendors/integrators to improve their products;
    - Help the users in their purchase decisions.

- **Agreement/acceptance**
  - A benchmark always represent an agreement accepted by the computer industry and/or by the user community.
Key issues on dependability benchmarking

- Representativeness
  - WP2 mainly

- Usefulness
  - WP3 mainly

- Agreement/acceptance
Dependability benchmarking overview

Experimentation

- Workload
- Faultload

Target System

Modeling

- Comprehensive Measures

Analysis and Categorization

- Experimental Measures & Features
A multidimensional problem

- Categorization dimensions
A multidimensional problem

- Categorization dimensions

**Benchmark usage**
- Benchmark performer
- Benchmark user
- Results use
- Results scope

**Life cycle phase**
- Operational phase
- Product development

**Application area**
- Divide spectrum
- **DBench**: Transactional & embedded

**Target system**
- Functional view
- Abstraction layers view
A multidimensional problem

- Categorization dimensions
- Measure dimension

Measure type
- Dependability
- Performance related

Measure extent
- Specific
- Comprehensive

Methodology
- Experimental
- Modeling
- Experimentation & modeling
A multidimensional problem

- Categorization dimensions
- Measure dimension
A multidimensional problem

- Categorization dimension
  - Examples of measures
    - Comprehensive dependability measures
      - Reliability
      - Maintainability
      - Availability
      - Safety
    - Specific dependability measures
      - Related to specific dependability features
      - Measure the efficiency feature is accomplished in value and time domains. Examples:
        - Error detection and latency;
        - Fault diagnosis efficiency and time to diagnosis;
    - Performance related measures
      - Response time
      - Throughput
      - Performance in presence of faults
      - Performance decreasing ratio due to faults
- Measure dimension
A multidimensional problem

- Categorization dimensions
- Measure dimension
- Experiment dimensions

Operating environment
Workload
Faultload
Procedures and rules
A multidimensional problem

- Categorization dimensions
- Measure dimension
- Experiment dimensions
- Property dimensions
A multidimensional problem

- Categorization dimensions
- Measure dimension
- Experiment dimension
- Property dimensions

Properties represent the main research problems of DBench
A multidimensional problem: summary

- Categorization dimensions
- Measure dimension
- Experiment dimensions
- Property dimensions
**Scenario 1: Dependability benchmarking based on experimentation only**

- Extension of the performance benchmark setting with two new components:
  - Features and experimental dependability measures
  - Faultload

- Measures are taken directly from the target system/component.
Scenario 2: Benchmarking dependability using modeling as a support for experimentation

Modelling helps in selecting the features and measures of interest to be evaluated experimentally as well as the workload and faultload.
Scenario 3: Benchmarking comprehensive dependability measures using experimentation as a support for modeling

- Experiments supports model validation and refinement
- Values for some model parameters are provided by experimental results
Scenario 4: General dependability benchmarking framework based on modeling and experimentation
Two preliminary examples

Example 1 - Dependability benchmark for transactional applications

Example 2 - Dependability benchmark for operating systems

• Goals
  - Illustrate instantiations of the framework to concrete cases
  - Early assessment
Example 1 - Dependability benchmark for transactional applications

- Built from the extension of TPC-C, especially the workload
- The target system represents a client-server system fully configured to run the workload.
- TPC-C workload also includes an external driver system to emulate the terminals and users during the benchmark run.
- Measures (except specific ones) are collected from the point of view of the end-users.
- Faultload: operation faults (as a start)
1) Workload runs without faults to get baseline performance measures

2) Workload runs in the presence of faults:
   - Runs for a fixed period $T_2$
   - $N$ faults are injected
   - After each fault:
     - The target system may continue executing transactions if nothing wrong is detected;
     - The target system may start a recovery procedure or restarted completely if recovery is not enough
   - Some integrity tests included in the workload and in the database engine are also performed concurrently.

3) Additional consistency tests run over workload data and metadata to check data integrity violations

Pan-Dependability Workshop, LAAS-CNRS, Toulouse, December 10-13, 2001
Example 1 - Some benchmark measures

- Tf - Number of transactions executed in the presence of faults
  → Measure of the impact of faults in the performance. It favors systems with higher capability of tolerating faults, fast recovery time, etc;

- €/Tf - Price per transaction in the presence of faults
  → Measure of the benefit of including fault handling mechanisms in the target system.

- Ta - Number of transactions aborted because of the faults
  → Measure of the impact of faults in the form of lost transactions or transactions that must be resubmitted;

- Ne - Number of errors detected by the workload consistency tests
  → Measure the impact of faults on the data integrity;

- Avt - Availability during the test period
  → Measure of system availability during the test period. It considers classes of availability: all terminals affected; 50% of the terminals affected; 10% of the terminals affected.
Example 2 - Dependability benchmark for operating systems

- Focus on the characterization of a key component: the OS
- Considers mainly the integrator perspective
- The target (OS) is described by its functional specification (“black-box” view typical of COTS components) and is assumed that internal detailed documentation is not available.
Example 2 - Dependability benchmark environment and key components

Input domain:

- **Faultload** - different classes of faults mainly injected through APIs
- **Workload** - start by synthetic workloads focused on the main functional components of a typical OS

Output domain:

- **Value Failures** - erroneous data observed at the API or at the workload level
- **Timing Failures** - either early or late timing failures
- **Error Detection/Signalling** - error checks typically provided by an OS
Conclusion

- We do understand the dependability benchmarking problem space now
- A framework defining the key components of the dependability benchmarking process has been proposed
- A set of different scenarios has been identified
- Focus on examples of dependability benchmarks
- Research directions are now very clear:
  - Investigate basic and technological issues related to the accomplishment of benchmark properties (WP2: representativeness, reproducibility, etc);
  - Expand and validate preliminary examples as a first step to define the benchmark prototypes planned in the project (WP3).
- Consolidation of the framework