Chapter 14
Resilience Benchmarking

Marco Vieira, Henrique Madeira, Kai Sachs and Samuel Kounev

Abstract Computer benchmarks are standard tools that allow evaluating and comparing different systems or components according to specific characteristics (performance, dependability, security, etc). Resilience encompasses all attributes of the quality of 'working well in a changing world that includes faults, failures, errors and attacks'. This way, resilience benchmarking merges concepts from performance, dependability, and security. This chapter presents an overview on the state-of-the-art on benchmarking performance, dependability and security. The goal is to identify the existing approaches, techniques and problems relevant to the resilience-benchmarking problem.

14.1 Introduction

Benchmarks are standard tools that allow evaluating and comparing different systems or components according to specific characteristics such as performance, dependability, and security. While historical benchmarks were only a few hundreds lines

M. Vieira (✉) · H. Madeira
DEI/CISUC, University of Coimbra,
Coimbra 3030-290, Portugal
e-mail: mvieira@dei.uc.pt

H. Madeira
e-mail: henrique@dei.uc.pt

K. Sachs
SAP AG, 69190 Walldorf, Germany
e-mail: kai.sachs@sap.com

S. Kounev
Karlsruhe Institute of Technology,
76131 Karlsruhe, Germany
e-mail: kounev@kit.edu

K. Wolter et al. (eds.), *Resilience Assessment and Evaluation of Computing Systems*,
DOI: 10.1007/978-3-642-29032-9_14,

long, modern benchmarks are composed of hundreds of thousands or millions of lines of code. Compared to traditional software, the benchmark development process has different goals and challenges. Unfortunately, even if an enormous number of benchmarks exist, only a few contributions focusing on the benchmark concepts and development process were published.

The best-known publication on benchmarking is Gray's *The Benchmark Handbook* [391]. Besides a detailed description of several benchmarks, the author discusses the need for domain specific benchmarks and defines four important criteria, which a domain-specific benchmark has to fulfill:

- *Relevance:* the benchmark result has to measure the performance of the typical operation within the problem domain.
- *Portability:* it should be easy to implement on many different systems and architectures.
- *Scalability:* it should be scalable to cover small and large systems.
- *Simplicity:* the benchmark should be understandable to avoid lack of credibility.

Another work dealing with the criteria that a benchmark should fulfill is [457].The questions, what a 'good' benchmark should look like and which aspects should be kept in mind from the beginning of the development process, are discussed in detail and five key criteria are presented:

- *Relevance:* the benchmark has to reflect something important.
- *Repeatable:* the benchmark result can be reproduced by rerunning the benchmark under similar conditions with the same result.
- *Fair & Portable:* All systems compared can participate equally (e.g., portability, 'fair' design).
- *Verifiable:* There has to be confidence that documented results are real. This can, e.g., be assured by reviewing results by external auditors.
- *Economical:* The cost of running the benchmark should be affordable.

The work on performance benchmarking has started long ago. Ranging from simple benchmarks that target a very specific hardware system or component to very complex benchmarks focusing on complex systems (e.g., database management systems, operating systems), performance benchmarks have contributed to improve successive generations of systems. Research on dependability benchmarking has been boosted in the beginning of the millennium, having already led to the proposal of several dependability benchmarks. Several works have been carried out by different groups and following different approaches (e.g., experimental, modeling, fault injection). Due to the increasing relevance of security aspects, security benchmarking is becoming an important research field.

Resilience encompasses all attributes of the quality of 'working well in a changing world that includes faults, failures, errors and attacks' [127]. This way, resilience benchmarking merges concepts from performance, dependability, and security benchmarking. In practice, resilience benchmarking faces challenges related to the integration of these three concepts and to the adaptive characteristics of the

systems under benchmarking. This chapter overviews the state-of-the-art on benchmarking performance, dependability and security, identifying the current approaches, techniques and problems relevant to the resilience benchmarking problem.

The outline of this chapter is as follows. The next section introduces the concept of performance benchmarking. Section 14.3 focuses on dependability benchmarking and presents existing research work. Section 14.4 introduces the security benchmarking problem. Section 14.5 discusses the current needs and challenges on resilience benchmarking. An overview of further research trends is provided in Sect. 14.6. Finally, Sect. 14.7 concludes the chapter and puts forward a potential research path to accomplish existing resilience benchmarking challenges.

14.2 Performance Benchmarking

Performance benchmarks are standard procedures and tools aiming at evaluating and comparing different systems or components in a specific domain (e.g., databases, operating systems, hardware, etc.) according to specific performance measures. Standardization organizations such as the SPEC (Standard Performance Evaluation Corporation) and the TPC (Transaction Processing Performance Council) use internal guidelines covering the development process of such benchmarks. A short summary of the keypoints of the SPEC Benchmark Development Process is provided in [573]. However, these guidelines mostly cover formal requirements, e.g., design of run rules and result submission guidelines, not the benchmark development process itself.

In general, a performance benchmark must fulfill the following fundamental requirements to be useful and reliable [545, 793, 794]:

- It must be based on a workload *representative* of real-world applications.
- It must *exercise all critical services* provided by platforms.
- It must *not be tuned/optimized for a specific product*.
- It must generate *reproducible* results.
- It must not have any inherent *scalability* limitations.

The major goal of a performance benchmark is to provide a standard workload and metrics for measuring and evaluating the performance and scalability of a certain platform. In addition, the benchmark should provide a flexible framework for performance analysis. To achieve this goal, the workload must be designed to meet a number of *workload requirements* that can be grouped according to the following five categories [793]:

1. *Representativeness*
2. *Comprehensiveness*
3. *Focus*
4. *Configurability*
5. *Scalability*

Representativeness No matter how well a benchmark is designed, it would be of little value if the workload it is based on does not reflect the way platform services are exercised in real-life systems. Therefore, the most important requirement for a benchmark is that it is based on a representative workload scenario including a representative set of interactions. The scenario should represent a typical transaction mix. The goal is to allow users to relate the observed behavior to their own applications and environments.
Comprehensiveness Another important requirement is that the workload is comprehensive in that it exercises all platform features typically used in the major classes of applications. The features and services stressed should be weighted according to their usage in real-life systems. There is no need to cover features of the platforms that are used very rarely in practice.
Focus The workload should be focused on measuring the performance and scalability of the platform under test. It should minimize the impact of other components and services that are typically used in the chosen application scenario.
Configurability In addition to providing standard workloads and metrics, a benchmark aims to provide a flexible performance analysis framework which allows users to configure and customize the workload according to their requirements. Many users will be interested in using a benchmark to tune and optimize their platforms or to analyze the performance of certain specific features. Others could use the benchmark for research purposes in academic environments where, for example, one might be interested in evaluating the performance and scalability of novel methods and techniques for building high-performance servers. All these usage scenarios require that the benchmark framework allows the user to precisely configure the workload and transaction mix to be generated. This configurability is a challenge because it requires that interactions are designed and implemented in such a way that one could run them in different combinations depending on the desired workload mix. The ability to switch interactions off implies that interactions should be decoupled from one another. On the other hand, it should be ensured that the benchmark, when run in its standard mode, behaves as if the interactions were interrelated according to their dependencies in the real-life application scenario.
Scalability Scalability should be supported in a manner that preserves the relation to the real-life business scenario modeled. In addition, the user should be offered the possibility to scale the workload in an arbitrary manner by defining an own set of scaling points.

14.2.1 SPEC Benchmarks

The Standard Performance Evaluation Corporation (SPEC) is one of the leading standardization bodies for benchmarks. While the most known benchmarks published by SPEC are the SPEC CPU series for the performance evaluation of CPUs, SPEC published benchmarks in many other areas, such as High Performance Computing, Java or Graphical Applications. Inside the SPEC, four groups exist [846]:

- *Open Systems Group (OSG)* focuses on benchmarks for desktop systems, high-end workstations and servers running open systems environments.
  Example benchmarks: SPEC CPU2006 (CPU performance), SPECjms2007 (message-oriented middleware benchmark, SPECpower_ssj2008 (power and performance benchmark), SPECvirt_sc2010 (virtualization benchmark) and SPECjEnterprise2010 (JavaEE benchmark).
- *High-Performance Group (HPG)* published a suite of benchmarks that represent high-performance computing applications for standardized, cross-platform performance evaluation.
  Example benchmarks: OMPM2001 / OMPL2001 (benchmarks for OpenMP applications and shared-memory systems) and MPIM2001 / MPIL2001 (benchmarks focusing on Message-Passing Interface (MPI) across a wide range of cluster and SMP hardware).
- *Graphics and Workstation Performance Group (GWPG)* develops graphics and workstation performance benchmarks.
  Example benchmarks: SPECapc benchmark series (addresses graphics and workstation performance evaluation based on actual software applications) and SPECviewperf 11.
- *Research Group (SPEC RG)* promotes research on benchmarking methodologies and tools facilitating the development of benchmark suites and performance analysis frameworks for established and newly emerging technology domains.

14.2.2 TPC Benchmarks

The benchmarks of the Transaction Processing Performance Council (TPC) became the de-facto standard in the database area [884]. Currently the TPC has three active benchmarks, two in the area of transaction processing (TPC-E/TPC-C) and one for benchmarking decision support. Their currently active benchmarks are based on a static workload mix. Additionally, TPC published the TPC-Energy Specification, which contains the rules and methodology for measuring and reporting an energy metric in TPC Benchmarks. It is important to note that, unlike SPEC, TPC does not provide implementations of its benchmarks. A TPC benchmark is essentially a specification that defines an application and a set of requirements on the workload that has to be run. The user is expected to implement the benchmark application and workload on the platform to be tested.

Further, TPC has released two benchmarks that can be used for benchmarking enterprise software systems. The first one is the TPC Benchmark W (TPC-W) [886], which has been available since 2000. The second one is the TPC Benchmark App (TPC-App) [885], which was released in December, 2004. However, both of these benchmarks are obsolete and there is no active benchmark for enterprise software systems.

14.2.3 EEMBC Benchmarks

The Embedded Microprocessor Benchmark Consortium (EEMBC) is developing performance benchmarks for the hardware and software used in embedded systems [325]. EEMBC microprocessor benchmark suites are targeting telecommunications, networking, digital media, Java, automotive/industrial, consumer, and office equipment products. Further, an additional suite that allows users to observe the energy consumed by the processor when performing these algorithms and applications exists. EEMBC also has a series of multicore-specific benchmarks that span multiple application areas.

14.2.4 Other Performance Benchmarks

Besides industry-standard benchmarks, numerous proprietary performance benchmarks for all kinds of systems have been developed and used in the industry and research. Due to the lack of space and the high number (e.g., we are aware of more than 15 benchmarks and performance tests suits for message-oriented middleware [793]) we will not discuss them here in detail.

14.3 Dependability Benchmarking

The notion of dependability and its terminology have been established by the *International Federation for Information Processing (IFIP) Working Group 10.4*. IFIP WG 10.4 defines dependability as *'the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers'*. Dependability is an integrative concept that includes the following attributes [576]: availability (readiness for correct service), reliability (continuity of correct service), safety (absence of catastrophic consequences on the user(s) and the environment), confidentiality (absence of unauthorized disclosure of information), integrity (absence of improper system state alterations), and maintainability (ability to undergo repairs and modifications).

A dependability benchmark can be defined as a specification of a standard procedure to assess dependability-related measures of a computer system or computer component. The main components of a dependability benchmark are: measures (characterize the performance and dependability of the system), workload (work that the system must perform during the benchmark run), faultload (set of faults that emulate real faults experienced in the field), and benchmark procedure and rules (description of the procedures and rules that must be followed to run the benchmark).

Two classes of measures can be considered when assessing dependability attributes:

- *Conditional measures:* measures that characterize the system in a relative fashion (i.e., measures that are directly related to the conditions disclosed in the benchmark report) and are mainly meant to compare alternative systems (e.g., response time, throughput, up-time, recovery time).
- *Unconditional measures on dependability attributes:* measures that characterize the system in a global fashion taking into account the occurrence of the various events impacting its behavior (i.e., reliability, availability, maintainability, safety, etc.) [576].

The conditional measures are directly obtained as results of the benchmark experiments. The unconditional measures on dependability attributes have to be calculated using modeling techniques with the help of external data, such as fault rates, MTBF, etc. This external data could be provided from field data or based on past experience considering similar systems. However, models of complex systems may be very difficult to define and the external data difficult to obtain.

Dependability benchmarks typically focus on direct measures (conditional measures), following the traditional benchmarking philosophy based on a pure experimental approach. These measures are related to the conditions disclosed in the benchmark report and can be used for comparison or for system/component improvement and tuning. This is similar to what happens with performance benchmark results, as the performance measures do not represent an absolute measure of system performance and cannot be used for capacity planning or to predict the actual performance of the system in field.

The faultload represents a set of faults that emulates real faults experienced by systems in the field. Among the main components needed to define a benchmark, the faultload is clearly the most complex one due to the nature of faults. A faultload can be based on three major classes of faults:

- *Operator faults:* operator faults are human mistakes. The great complexity of administration tasks in some systems and the need of tuning and administration in a daily basis, clearly explains why human faults (i.e., wrong human actions) should be considered in a dependability benchmark.
- *Software faults:* software faults (i.e., program defects or bugs) are recognized as an important source of system outages, and given the huge complexity of today's software the weight of software faults tends to increase.
- *Hardware faults:* includes traditional hardware faults, such as bit-flips and stuck-at, and high-level hardware failures, such as hard disk failures or failures of the interconnection network. Hardware faults are especially relevant in systems prone to electrical interferences.

Concerning the definition of the workload, the job is considerably simplified by the existence of workloads from performance benchmarks. Obviously, these already established workloads are the natural choice for a dependability benchmark. However, when adopting an existing workload some changes may be required in order to target specific system features. An important aspect to keep in mind when choosing

a workload is that the goal is not only to evaluate the performance but also assess specific dependability features.

The procedures and rules define the correct steps to run a benchmark and obtain the measures. These rules are, of course, dependent on the specific benchmark but the following points give some guidelines on specific aspects needed in most of the cases:

- Procedures for 'translating' the workload and faultload defined in the benchmark specification into the actual workload and faultload that will apply to the system.
- Uniform conditions to build the setup and run the dependability benchmark.
- Rules related to the collection of the experimental results.
- Rules for the production of the final measures from the direct experimental results.
- Scaling rules to adapt the same benchmark to systems of very different sizes.
- System configuration disclosures required for interpreting the benchmark results.
- Rules to avoid optimistic or biased results.

The awareness of the importance of dependability benchmarks has increased in the recent years and dependability benchmarking is currently the subject of strong research. The following subsections present the recent advances on dependability benchmarking, both at universities and computer industry sites.

14.3.1 Special Interest Group on Dependability Benchmarking (SIGDeB)

The *Special Interest Group on Dependability Benchmarking* (SIGDeB) was created by the International Federation for Information Processing (IFIP) Working Group 10.4 in 1999 to promote the research, practice, and adoption of benchmarks for computer-related systems dependability. The work carried out in the context of the SIGDeB is particularly relevant and merges contributions from both industry and academia.

A preliminary proposal issued by the SIGDeB was in the form of a set of standardized classes for characterizing the dependability of computer systems [934]. The goal of the proposed classification was to allow the comparison among computer systems concerning four different dimensions: availability, data integrity, disaster recovery, and security. The authors have specifically developed the details of the proposal for transaction processing applications. This work proposes that the evaluation of a system should be done by answering a set of standardized questions or performing tests that validate the evaluation criteria.

A very relevant effort in the context of SIGDeB is a book on dependability benchmarking of computer systems [514]. This book presents several relevant benchmarking initiatives carried out by different organizations, ranging from academia to large industrial companies.

14.3.2 DBench Project

The DBench project was funded by the European Commission, under the Information Society Technologies Programme (IST), Fifth Framework Programme (FP5). The main goal of DBench project was to devise benchmarks to evaluate and compare the dependability of COTS and COTS-based systems, in embedded, real time, and transactional systems. Several works on dependability benchmarking have been carried out in the DBench project. The following subsections summarize those works.

General purpose operating systems The works presented in [502, 503, 511, 512] address the problem of dependability benchmarking for general purpose operating systems (OS), focusing mainly on the robustness of the OS (in particular of the OS kernel) with respect to faulty applications.

The measures provided are: 1) OS robustness in the presence of faulty system calls, 2) OS reaction time for faulty system calls and 3) OS restart time after the activation of faulty system calls. Three workloads are considered: 1) a realistic application that implements the experiments control system of the TPC-C performance benchmark [887]. 2) the PostMark [521] file system performance benchmark for operating systems and 3) the Java Virtual Machine (JVM) middleware. The faultload is based on the corruption of systems call parameters.

Another research work on the practical characterization of operating systems behaviour in the presence of software faults in OS components is presented in [312]. The methodology used is based on the emulation of software faults in device drivers and the observation of the behaviour of the overall system regarding a comprehensive set of failure modes analyzed according to different dimensions related to different user perspectives.

Real time kernels in onboard space systems The work presented in [666] is a preliminary proposal of a dependability benchmark for real time kernels for onboard space systems. This benchmark, called DBench-RTK, focuses mainly on the assessment of the predictability of response time of service calls in a Real-Time Kernel (RTK).

The DBench-RTK dependability benchmark provides a single measure that represents the predictability of response time of the service calls of RTKs used in space domain systems. The workload consists in an Onboard Scheduler (OBS) process based on a functional model derived from the Packet Utilization Standard [864]. The faultload consists of a set of faults that are injected into kernel functions calls at the parameter level by corrupting parameter values.

Engine control applications in automotive systems The work presented in [790] represents a preliminary proposal of a dependability benchmark for engine control applications for automotive systems. This benchmark focuses on the robustness of the control applications running inside the Electronic Control Units (ECU) with respect to transient hardware faults.

This dependability benchmark provides a set of measures that allows the comparison of the safety of different engine control systems. The workload is based on the standards used in Europe for the emission certification of light duty vehicles [320].

The faultload consists of transient hardware faults that affect the cells of the memory holding the software used in the engine control.

On-line transaction processing systems The DBench-OLTP dependability benchmark [917, 918] is a dependability benchmark for on-line transaction processing systems. The DBench-OLTP measures are divided in three groups: baseline performance measures, performance measures in the presence of the faultload, and dependability measures. The DBench-OLTP benchmark can be used considering three different faultloads each one based on a different class of faults, namely: operator faults, software faults and high-level hardware failures.

In [161] it is presented a preliminary proposal of another dependability benchmark for on-line transaction processing systems. The measures provided by this dependability benchmark are the system availability and the total cost of failures. These measures are based on both measurements obtained from experimentation (e.g., percentages of the various failure modes) and external data (e.g., the failure rates and the repair rates). The external data used to calculate the measures must be provided by the benchmark user. The workload was adopted from the TPC-C performance benchmark [887] and the faultload includes exclusively hardware faults, such as faults in the storage hardware and in the network.

Web-servers The work presented in [316] proposes a dependability benchmark for web-servers (the WEB-DB dependability benchmark). This dependability benchmark uses the basic experimental setup, the workload, and the performance measures specified in the SPECWeb99 performance benchmark [845].

The measures reported by WEB-DB are grouped into three categories: baseline performance measures, performance measures in the presence of the faultload, and dependability measures. The WEB-DB benchmark uses two different faultloads: one based on software faults that emulate realistic software defects (see [314]) and another based on operational faults that emulate the effects of hardware and operator faults.

14.3.3 Berkeley University

The work developed at Berkeley University has highly contributed to the progress of research on dependability benchmarking in the last few years, principally on what concerns benchmarking the dependability of human-assisted recovery processes.

A general methodology for benchmarking the availability of computer systems is introduced in [155]. The workload and performance measures are adopted from existing performance benchmarks and the measure of availability of the system under test is defined in terms of the service provided by the system. The faultload (called fault workload by the authors) may be composed of a single-fault (single-fault workload) or of several faults (multi-fault workload).

The work presented in [156] addresses human error as an important aspect in system dependability, and proposes that human behaviour must be considered in dependability benchmarks and system designs.

A technique to develop dependability benchmarks that capture the impact of human operators on the tested system is proposed in [154]. The workload and measures are adopted from existing performance benchmarks and the dependability of the system can be characterized by examining how the performance measures deviate from their normal values as the system is perturbed by injected faults. In addition to faults injected using traditional fault injection, perturbations are generated by actions of human operators that actually participate in the benchmarking procedure.

In [151] are presented the first steps towards the development of a dependability benchmark for human assisted recovery processes and tools. This work proposes a methodology to evaluate human-assisted failure recovery tools and processes in server systems. This methodology can be used to both quantify the dependability of single recovery systems and compare different recovery approaches, and combines dependability benchmarking with human user studies.

14.3.4 Carnegie Mellon University

Vajra [674] is a research project whose goal is benchmarking the survivability in distributed systems, focusing on the objective and quantitative comparison of the runtime implementations of different Byzantine fault-tolerant distributed systems. The benchmark uses as the point of injection APIs that are common across various Byzantine fault-tolerant systems. A variety of accidental and malicious faults are injected at various rates across the system.

Although not resulting in a formal benchmark proposal, the research on robustness testing developed at the Carnegie Mellon University [540] has effectively set the basis for robustness benchmarks of operating systems. This will be further discussed in Chap. 16, which includes a survey on robustness testing techniques.

14.3.5 Sun Microsystems

Research at Sun Microsystems has defined a high-level framework [959] specifically dedicated to availability benchmarking of computer systems. The proposed framework follows a hierarchical approach that decomposes availability into three key components: fault/maintenance rate, robustness, and recovery. The goal was to develop a suite of benchmarks, each one measuring an aspect of the availability of the system. Within the framework proposed by [959], two specific benchmarks have already been developed.

In [960] is proposed a benchmark for measuring a system's robustness (degree of protection that exists in a system against outage events) in handling maintenance events, such as the replacement of a failed hardware component or the installation of a software patch.

In [629] is proposed a benchmark for measuring system recovery in a non-clustered standalone system. This benchmark measures three specific system events; clean system shutdown (provides a baseline metric), clean system bootstrap (corresponds to rebooting a system following a clean shutdown), and a system reboot after a fatal fault event (provides a metric that represents the time between the injection of a fault and the moment when the system returns to a useful state).

Another effort at Sun Microsystems are the Analytical RAS Benchmarks [324], which consists of three analytical benchmarks that examine the Reliability, Availability, and Serviceability (RAS) characteristics of computer systems:

- The Fault Robustness Benchmark (FRB-A) allows assessing and comparing the techniques used to enhance resiliency, including redundancy and automatic fault correction.
- The Maintenance Robustness Benchmark (MRB-A) assesses how maintenance activities affect the ability of the system to provide a continuous service.
- The Service Complexity Benchmark (SCB-A) examines the complexity of mechanical components replacement.

14.3.6 Intel Corporation

Work at Intel Corporation has focused on benchmarking semiconductor technology. The work presented in [236] shows the impact of semiconductor technology scaling on neutron induced SER (soft error rate) and presents an experimental methodology and results of accelerated measurements carried out on Intel Itanium microprocessors. The proposed approach can be used as a dependability benchmarking tool and does not require proprietary information about the microprocessor under benchmarking.

Another study [236] presents a set of benchmarks that rely on environmental test tools to benchmark undetected computational errors, also known as silent data corruption (SDC). In this work, a temperature and voltage operating test (known as the four corners test) is performed on several prototype systems.

14.3.7 IBM Autonomic Computing Initiative

At IBM, the Autonomic Computing initiative developed benchmarks to quantify a system's level of autonomic capability, which is defined as the capacity of the system to react autonomously to problems and changes in the environment. The goal was to produce a suite of benchmarks covering the four categories of autonomic capabilities: self-configuration, self-healing, self-optimization, and self-protection.

The first steps towards a benchmark for autonomic computing are described in [589]. The benchmark addresses the four attributes of autonomic computing and is able to test systems at different levels of autonomic maturity.

The work presented in [152] identifies the challenges and pitfalls that must be taken into account in the development of benchmarks for autonomic computing capabilities. This paper proposes the use of the workload and driver system from performance benchmarks and the introduction of changes into benchmarking environment in order to characterize a given autonomic capability of the system. The paper proposes that autonomic benchmarks must quantify the level of the response, the quality of the response, the impact of the response on the users, and the cost of any extra resources needed to support the autonomic response.

14.4 Security Benchmarking

Theoretically, a security benchmark provides a metric (or small set of metrics) able to characterize the degree to which security goals are met in a given piece of code [483], allowing developers and administrators to make informed decisions. However, one of the biggest difficulties in designing such metric is related to the fact that security assessment is, usually, much more dependent on what is unknown about the applications (e.g. unknown bugs, hidden vulnerabilities) than by what is known (e.g., known features, existing security mechanisms).

Security metrics are hard to define and compute [883] because they involve making isolated estimations about the ability of an unknown individual (e.g., a hacker) to discover and maliciously exploit an unknown system characteristic (e.g., a vulnerability). A feasible alternative is to assume that such metrics can be obtained using information about the system itself, without taking into account external factors. In fact, a security benchmark based on such metrics would allow characterizing the *degree to which security goals are met in a given web application or component*. In practice, due to the difficulties of quantifying security, most works on security benchmarking are based on analysis and qualification of configurations/systems.

Several security evaluation methods have been proposed in the past [232, 233, 288, 807]. The Orange Book [288] and the Common Criteria for Information Technology Security Evaluation [233] define a set of generic rules that allow developers to specify the security attributes of their products and evaluators to verify if products actually meet their claims. Another example is the red team strategy [807], which consists of a group of experts trying to hack its own computer systems to evaluate security. However, none of these security evaluation approaches is oriented towards security benchmarking, as comparing security has been largely absent from these security evaluation methods.

The work presented in [630] addresses the problem of determining, in a thorough and consistent way, the reliability and accuracy of anomaly detectors. This work addresses some key aspects that must be taken into consideration when benchmarking the performance of anomaly detection in the cyber-domain.

The set of security configuration benchmarks created by the *Center for Internet Security* (CIS) is a very interesting initiative. CIS is a non-profit organization formed by several well-known academic, commercial, and governmental entities that has created a series of security configuration documents for several commercial and open source systems. These documents focus on the practical aspects of the configuration of these systems and state the concrete values each configuration option should have in order to enhance overall security of real installations. Although CIS refers to these documents as benchmarks they mainly reflect best practices and are not explicitly designed for systems assessment or comparison.

A practical way to characterize the security mechanisms in database systems is proposed in [920]. In this approach database management systems (DBMS) are classified according to a set of security classes ranging from Class 0 to Class 5 (from the worst to the best). Systems are classified in a given class according to the security requirements satisfied. In [50] the authors analyze the security best practices behind the many configuration options available in several well-known DBMS. These security best practices are then generalized and used to define a set of configuration tests that can be used to compare different database installations. An improved set of best practices is then used in [52] to benchmark the security of database servers configurations.

A benchmark that allows database administrators to assess and compare database configurations is presented in [51]. The benchmark provides a trust-based security metric, named minimum untrustworthiness, that expresses the minimum level of distrust the DBA should have in a given configuration regarding its ability to prevent attacks. The use of trust-based metrics as an alternative to security measurement is discussed in [682].

14.5 Resilience Benchmarking

A resilience benchmark should provide generic ways for characterizing a system behavior in the presence of perturbations. If a system is effective and efficient in accommodating or adjusting to perturbations, avoiding failures as much as possible, it is reasonable to consider it as being resilient [33]. This capability can be benchmarked by submitting the system to various types of perturbations and by observing the failures (and their frequency), as well as time and resources dedicated to avoid/recover from them. Still, the perturbations that the system has to face may lead to performance and dependability attributes degradation without leading necessarily to catastrophic system failures. Thus, we need to assess variations of the properties of interest (e.g., performance, availability, integrity) when the system is under varying context conditions, in order to characterize its behavior from a resilience perspective.

Evaluating resilience must consider the system and environment dynamics that are beyond those typically addressed in the evaluation of performance and dependability. While maintaining similar workloads, dependability benchmarks enhanced performance benchmarks by introducing a faultload and dependability metrics, which

include performance metrics under faulty conditions. A resilience benchmark must comprise a more wide-ranging set of perturbations, which will certainly include (but will be not limited to) faults. For instance, variations on the workload or in system parameters should be part of those perturbations. New metrics for characterizing resilience are also needed, although some will naturally be based on measures of performance and dependability while facing changes.

In practice, resilience benchmarking includes performance, dependability, and security aspects, and aims at providing generic, repeatable and widely accepted methods for characterizing and quantifying the system (or component) behavior in the presence of faults, and comparing alternative solutions [514]. Although many works have been conducted in the area of performance and dependability benchmarking, it is clear that many key issues must be addressed towards the definition of concrete resilience benchmarks, which, theoretically, should include the following main components:

- *Benchmarking metrics:* the benchmark metrics should allow characterizing and quantifying the system behavior when facing perturbations (i.e., faults, attacks, and operational environment variations). At first sight, resilience benchmarking metrics must characterize performance, dependability and security.
- *Workload:* during the benchmark execution, the system under test must be submitted to a representative set of tasks, which should be as close to real conditions as possible. An important aspect is that a workload cannot be static and must exercise the resilience capabilities of the system, as the real conditions would.
- *Perturbations-load:* a system may be subjected to distinct types of perturbations during its operation, and a benchmark must try to emulate those as realistically as possible. These perturbations may be of three different types: faults, attacks, and perturbations related to system's maintenance.

In the context of the AMBER Coordination Action, funded by the European Union under the Seventh Framework Programme, a set of research needs related to resilience benchmarking have been identified, namely (see details at [127]):

1. Agreed, cost effective, easy to use, fast and representative enough dependability benchmarks for well defined domains.
2. Benchmark frameworks (components and tools) able to be reused to create benchmarks in different benchmarking domains.
3. Inclusion of adequate design methodologies to facilitate benchmark implementation and configuration in future components, systems, and infrastructures.
4. Uniform (standardized) benchmarking process that can be applied by independent organizations to offer certification of the dependability of COTS products (like in the case of standards compliance testing).

These needs raise a set of research challenges that have to be addressed in order to be able to define a (resilience) benchmark, namely (see [127] for details):

1. *Defining benchmark domains* (components, systems, application domains) in order to divide the problem space in adequate/tractable segments.

2. *Defining key benchmark elements* such as measures, workload, faultload, models, to ensure the necessary properties (e.g., representativeness, portability, scalability, repeatability) that allow agreement on benchmark proposals.
3. *Coping with highly complex, adaptable and evolving benchmark targets* (components, systems and services).
4. *Coping with human factors* in the definition and execution of benchmarks.
5. *Assuring proper validation of dependability benchmarks* in order to achieve the necessary agreement to establish benchmarks. This implies the validation of the different benchmark properties.
6. *Assuring reusability of benchmark frameworks* (components and tools) to create benchmarks in different benchmarking domains.
7. *Defining and agreeing on a domain-specific dependability benchmarking process* that can be accepted by the parties concerned (supplier, customer and certifier) and can be adapted to different products in the domain (e.g., in a product line).

14.6 Further Trends in Benchmarking Research

Besides resilience benchmarking we see some further research trends in the area of benchmarking, which we discuss in this section.

14.6.1 Benchmark Engineering

While developing benchmarks, we faced a lack of methodology that describes how to develop good and meaningful benchmarks. Since benchmark development has turned into a complex team effort, there is a need for a development methodology taking the specifics of benchmarks into account. Compared to traditional software, the development process has different goals and challenges. New concepts and processes are needed which address the whole development and life-cycle management of benchmarks. We refer to them including benchmark methodology and measurement techniques with the term *Benchmark Engineering* [793]. First work is already in progress. As example, SPEC is working on development guidelines.

14.6.2 Benchmarking of Large Scale Systems

Large scale, highly distributed systems are increasingly used in mainstream applications. However, for these systems traditional benchmarking approaches fail: how can we benchmark a system with 500,000 nodes? What does a typical workload look like and how does it scale? What should be the distribution of the faultload? etc.

Since it is not feasible to run benchmarks in a realistic environment with thousands of nodes, new methods are needed which allow us to benchmark large scale systems in a realistic way on limited resources. As a consequence, we see a need for research in the area of *simulated benchmarks*.

Similar questions are currently under discussion in several research areas. The authors in [583] discuss requirements for peer-to-peer (P2P) benchmarking and present two exemplary approaches to benchmark such systems. They point out the challenges of developing P2P benchmarks compared to conventional benchmarks.

A very active community can be found in the area of cloud benchmarking. A discussion why traditional benchmarks are not sufficient for evaluating cloud services can be found in [114]. The authors present some initial ideas how a cloud benchmark should be designed including a list of requirements for such a benchmark. In [238] the Yahoo! Cloud Serving Benchmark (YCSB) framework was introduced including a core set of benchmarks. YCSB targets cloud data serving services, allows to create new workloads and is extendible. Another example is the Cloudstone benchmark, which consists of a social-events web application (with PHP and Ruby implementations) and a set of automation tools for load generation and performance measurement [839]. When running the benchmark, the load is generated against the web application, which in turn generates load on the underlying database.

There are still many open questions in the area of P2P and cloud benchmarking. This is the reason, why the SPEC Research Group decided to launch two subcommittees working on these topics.

14.6.3 Power Consumption

In the past, benchmarking focused mainly on computation performance. Since industry and governments are increasingly concerned about the energy use of servers, there is a need to reflect the power consumption in the result of a benchmark. The first standard benchmark providing a metrics, which represents computation performance as well as energy consumption was the SPECpower_ssj2008 benchmark. Nowadays, more and more benchmarks include energy consumption in their result, such as SPEC or TPC benchmarks. Consequently, the SPEC is working on the Server Efficiency Rating Tool (SERT), a tool set to measure and evaluate the energy efficiency of computer servers [889].

A metric for power consumption has to reflect both, traditional performance metrics in relation to the power consumption and not only peak performance is of interest. However, energy consumption *scenarios* are only one example, where traditional benchmark metrics fail or are hard to apply. A major challenge of future benchmark development is the definition of meaningful metrics, which take other aspect than performance and dependability into account (see also Sect. 14.5).

14.7 Conclusion

This chapter presented the state-of-the-art on benchmarking. The work on performance benchmarking has started long ago and has contributed to improve successive generations of systems. Dependability benchmarking efforts both at universities and computer industry sites are quite recent. Security is a newcomer to the benchmarking world and little work has been performed so far.

Although performance benchmarking is a very well established field, further work on dependability benchmarking seems to be necessary in several application areas (e.g., real-time systems, grid computing, parallel systems, etc). Additionally, no dependability benchmark has achieved the status of a real benchmark endorsed by a standardization body. This may be due to several reasons (that need to be studied) but clearly shows that additional work is still needed.

In the area of security benchmarking, a lot of work is clearly needed, as this is a new and quite challenging field for which little work has been developed so far. A key issue is the definition of useful and meaningful security metrics. In fact, the problem of security quantification is a longstanding one. A useful security metric must portray the degree to which security goals are met in a given system, allowing a system administrator to make informed decisions. One of the biggest difficulties in designing such a metric is related to the fact that security is, usually, much more dependent on what is unknown about the system than on what is known about it. In fact, security metrics are hard to define and compute as they involve making isolated estimations about the ability of an unknown individual (e.g., a hacker) to discover and maliciously exploit an unknown system characteristic (e.g., a vulnerability).

To tackle the challenges related to the future implementation of resilience benchmarks, the following research steps are foreseen:

1. Study the metrics that better characterize resilience.
2. Study the definition of dynamic workloads via field studies and analysis of existing workloads.
3. Study the characterization of perturbation loads. This can be based on field studies and on the analysis of already existing faultloads.
4. Define the steps needed for the execution of a resilience benchmark. These steps define the benchmark procedure and should be as generic as possible to allow the portability of the benchmarking approach.
5. Conduct benchmarking campaigns to demonstrate the benchmark and validate its properties.
6. Generalize the resilience benchmarking approach to make possible its application in different domains.
7. Disseminate the benchmarking approach. A key aspect is to identify the best way to foster the adoption by industry and to facilitate the support by a standardization body like TPC and SPEC.

Acknowledgments The work of Marco Vieira and Henrique Madeira was partially funded by the European Commission under project *AMBER - Assessing, Measuring and Benchmarking Resilience*, IST - 216295, funded by the European Union, 2009. The work of Samuel Kounev was partially funded by the German Research Foundation (DFG) under grant No. KO 3445/6-1.