Deriving Dependability Measures of Measurements Recorded in a Matrix

Oliver Tschäche
Institut für Informatik 3
Friedrich Alexander Universität Erlangen-Nürnberg
Martensstr. 3
D-91058 Erlangen
ot@cs.fau.de

Abstract:

Dependability benchmarking is meant to measure system characteristics like availability, reliablity, data integrity etc. Todays systems are working at high levels of these characteristics. Evaluation of these characteristics demands to inject faults forcing fault tolerant mechanisms to exercise their tasks. Observing the response of the system leads to measurements assessing the quality of these mechanisms.

Our paper's focus is not on how to create a special dependability benchmark but on how to deduce significant dependability characteristics out of fault injection based measurements. We disclose which information we need for a general dependability benchmark, from whom they should be supplied and, finally, how to derive assessments of dependability metrics from this information.

Our method is universally applicable to all fault injection based dependability benchmarking methods. Using one method for the presentation of dependability has several advantages: E.g. benchmarks become comparable to each other, benchmarkers faster learn how to interpret similarly looking results.

1 Introduction

Performance benchmarks assess systems or applications according to different metrics like instructions per second (MIPS), transactions per minute (tpm), delay of response, resource usage etc. This is done by applying an application dependent workload to the system. Dependability benchmarks extend performance benchmarks to assess dependability metrics, additionally.

A common way to evaluate a system's dependability assessment is to use fault injection techniques while running the system. Therefore, a workload is run several times: The Golden Run is a faultless run serving as reference for the other runs called experiments. For each experiment at least one fault is activated and the impact of that fault is classified according to failure modes. Finally, the dependability assessment is derived from the distribution of failure modes which is evaluated by injecting each fault several times.

Todays dependability benchmarks, see [VM02, BT03, NLZ+03], differ in recording the

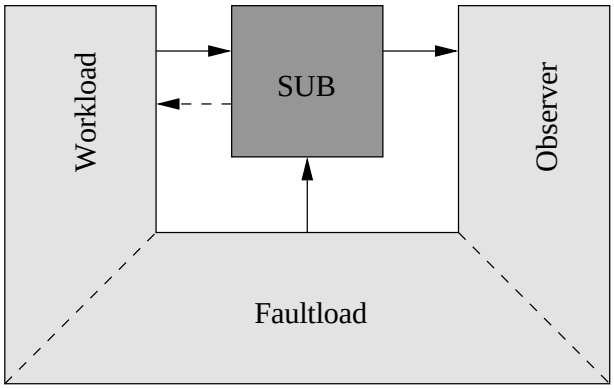

Abbildung 1: Dependability Benchmark Environment

measurements and calculating the dependability assessments. Our approach starts an abstraction layer higher consolidating these methods resulting in a universal method which could be used for all fault injection based dependability assessment approaches.

A common setup used in published dependability benchmarks, see [VM02, BT03, NLZ+03], is shown in figure 1: The system under benchmark (SUB) is embedded into a testbench which is referred to as the dependability benchmarking environment (DBE). The DBE's task, see figure 1, is to stimulate the SUB with a workload and faultload while observing the behaviour of the SUB and detecting failure modes classifying the system's response, e.g. system crash, graceful degradation etc.:

Workload: An easy way to obtain a workload is to use an already known performance benchmark. In addition a performance benchmark supplies methods to measure the application's characteristic e.g. number of transactions finished, delays, resource usage etc.

Faultload: To apply the faultload fault injection methods are used. According to the dependability benchmarked application the faults are e.g. stuck at and bitflip faults in case of embedded systems or in case of higher level applications like web or database servers harddisk, power fail and network faults.

Observer: The DBE's observer classifies the system's response to the workload and faultload. Measurements assessing the system's performance are already handled by performance benchmarks. These measurements are completed to dependability measurements by methods detecting the system's state after an activated fault. E.g. a system crashes (and needs to be repaired) or gracefully degrades after a fault injection.

The method presented in this paper specifies how to derive dependability assessments using fault injection techniques. [BT03, VM02, NLZ+03] propose dependability benchmarks using fault injection techniques, too. Although they assess systems according to the

	F_0	F_1	F_j	F_M
f_0	$p(f_0 \rightarrow F_0)$	$p(f_0 \rightarrow F_1)$	$p(f_0 \rightarrow F_j)$	$p(f_0 \rightarrow F_M)$
f_1	$p(f_1 \rightarrow F_0)$	$p(f_1 \rightarrow F_1)$	$p(f_1 \rightarrow F_j)$	$p(f_1 \rightarrow F_M)$
f_i	$p(f_i \rightarrow F_0)$	$p(f_i \rightarrow F_1)$	$p(f_i \rightarrow F_j)$	$p(f_i \rightarrow F_M)$
f_N	$p(f_N \rightarrow F_0)$	$p(f_N \rightarrow F_1)$	$p(f_N \rightarrow F_j)$	$p(f_N \rightarrow F_M)$

Tabelle 1: Matrix describing system's behaviour

distribution of failure modes and according to a set of faults, they differ in the matter how to present measurements and, finally, the dependability measures. Using a universal presentation scheme would simplify the evaluation of the results by a benchmarker. Thus, our approach is able to cover already published dependability benchmarks, too.

Focusing on reproducability of a dependability benchmark all details of how the assessment is derived have to be disclosed. Our approach discloses all parameters and the formulas using them making a dependability benchmark reproducable.

Because dependability benchmarking allows very different sight of views our approach introduces rates and costs weighting the measured distributions so that a specific dependability metric is evaluated. Other dependability benchmarks like [VM02] don't account that important issue.

2 The matrix

As indicated in section 1 a dependability benchmark using fault injection techniques tracks the system behaviour in two dimensions: The faults and their impact - the failure modes. The impact of a fault depends on the time when it is injected (is the faulty component active) and the design of the system (is a faulty system state tolerated). Therefore, each fault must be injected several times, so that the distribution to failure modes can be recorded. The more often a fault is injected the more accurate and confident are the measured distribution to the failure modes of this fault.

For our approach we propose to record and to publish the system behaviour in a matrix, see 1. The rows of the matrix list the faults f_i and the columns keep the failure modes F_j. The number of faults is N, the number of failure modes is M. The fault injection experiments provide the matrix with the frequency $p(f_i \rightarrow F_j)$ a fault f_i impacts failure mode F_i.

The phase of using fault injection techniques to fill the matrix only records the system behaviour according to the selected experiments and is not a dependability assessment of the system. For the dependability assessment of the system we need additional information which is described in section 3.

As indicated above the confidence of the matrix elements depends on the number of experiments. More precisely, the measured frequencies' confidence to a fault f_i depends on the number of experiments n_i executed with that fault. Thus, the confidence between lines

of the matrix differs if different number of experiments are made. The confidence of the matrix elements can be visualized by an interval around the measured frequency. The tighter the interval the more confident the measured value. Using Student t-distribution, see [St08], it is possible to estimate the size of the interval according to an applied significance $1-\alpha$. $1-\alpha$ is the probability that the mean value is included in that interval. Assuming a worst case standard deviation results in equation 1:

$$size_i(\alpha, n_i) = \frac{t_{\alpha/2, n_i - 1}}{\sqrt{n_i}} \tag{1}$$

Another issue on filling the matrix is, which fault should be injected how many times. The overall number n of experiments is limited by the time in which the results have to be obtained. The goal is to derive high confident dependability assessments. Section 3 shows how the dependability assessments are derived from the matrix elements and their confidence intervals, resulting in an assessment with a confidence interval. The best way is to find the global minimum of the interval size. We did not find the global minimum yet. We suggest to start with a fault distribution in which the frequency of each injected fault is proportional to its rate, see section 3.

All dependability assessments, see section 3, are based on that matrix 1. Therefore, it must be generated accurately: In order to generate reproducable dependability assessments, the matrix itself must be generated in a reproducable way. Therefore, we advice to use a fully automated experiment controller, see [BDH+02].

3 Deriving Dependability Measures

Dependability benchmarking is more than evaluating the system behaviour by using fault injection experiments. In our point of view a dependability benchmark should be split into two phases: Evaluating the matrix elements, see 2, and deriving dependability assessments from them. Thus, the first phase can be supplied by the manufacturer of the system by running the fault injection experiments. In the second phase the system's user derives the dependability assessment according to his environment. This section explains the parameters we are using to describe the user's environment.

3.1 Fault rates

Dependability benchmarking must take into account a faults' rate. E.g. the difference of two systems could be that a system is using a more reliable harddisk than the other which, in general, must be honored with a better assessment.

On the other hand a dependability benchmark must be able to relate faults to each other. E.g. if a system is benchmarked according to operator and hardware faults a dependability benchmark must take into account the quality of each service related to each other. Again,

using fault rates enables a kind of weighting.

$$W_j = \sum_{i=0}^{N} r_i \cdot p(f_i \rightarrow F_j) \tag{2}$$

The unit of the fault rates is number of fault appearances per unit of time. Multiplying each matrix element with the fault rate r_i and summing up over all columns leads to the rate W_j of each failure mode, see equation 2. In a real system this rate of each failure mode should be detected when the system is observed a very long time.

3.2 Costs

Dependability benchmarks come along with various metrics like availability, reliability, costs etc. In our approach the differences of these metrics are mapped to different weightings R_j^{metric} of the rates a failure mode appears, see equation 2. Each metric has its own weighting.

The unit of the weighting R_j^{metric} may vary according to the focussed metric. E.g. to derive an assessment for the dependability metric unavailability, $R_j^{unavailability}$ is of dimension time describing how long the system is unavailable if it enters failure mode F_j. Summing up over all failure modes, see equation 3, results in the dependability assessment $cost^{unavailability}$ which has the unit $\frac{downtim}{time}$. The availability of the system is $1 - cost^{unavailability}$.

$$cost^{metric} = \sum_{j=0}^{M} R_j^{metric} \cdot W_j \tag{3}$$

The following example illustrates the usefulness of our approach if a company has to assess several systems in order to select the best one according to the company's cost metric. This demands that the company has to provide the costs $R_j^{company}$ which are raised if the failure mode F_j is entered. Only the company itself knows about the fault rates r_i: It decides which hardware to use (hardware faults) and only the company knows at which quality level its IT service operates (operator faults). In the best case the matrix is provided by the manufacturer of the system, so that the decision maker only has to derive the assessment using equations 2 and 3. In contrast to deriving a metric of availability this company metric has the unit $\frac{Dollars}{time}$.

Equation 3 calculates the mean costs missing confidence information. We propose to derive the confidence of that assessment by calculating the minimal and maximal $edge$ values applying the confidence interval to each observed frequency $p(f_i \rightarrow F_j)$:

$$p_{edge}(f_i \rightarrow F_j) = edge(p(f_i \rightarrow F_j), size_i) \tag{4}$$

The functions $edge()$ subtract/add the $size_i/2$ (see equation 1) from/to the frequency $p(f_i \rightarrow F_j)$ according to the selected edge. If the result is less than 0 or greater than 1 they return 0 and 1, respectively. Using equation 4, the minimal and maximal costs according to that confidence can be calculated by replacing $p(f_i \rightarrow F_j)$ of equation 2 by $p_{min}(f_i \rightarrow F_j)$ and $p_{max}(f_i \rightarrow F_j)$. Using equation 3 leads to equation 5:

$$cost_{edge}^{metric} = \sum_{j=0}^{M} R_j^{metric} \cdot \sum_{i=0}^{N} r_i \cdot p_{edge}(f_i \rightarrow F_j) \tag{5}$$

In case of comparison of two benchmarked systems we demand that the intervals of the costs must not overlap. If they overlap we propose to state that the decision of which system is better is hidden in statistic noise.

3.3 Characteristics

In this section we assess our method. We explain the dependencies raised by that method and disclose advantages and disadvantages.

An improvement with respect to other known methods is that our method provides a confidence estimation. The higher the confidence of a cost calculation the more tighten is the interval around the mean value. The size of the interval depends on the number of experiments n_i exercised on each fault f_i. Increasing n_i tightens the interval and, therefore, increases confidence.

Unfortunately the size of the confidence interval is proportional to the squareroot of the number of experiments, so that increasing the number of experiments by 4 times only doubles the confidence of the results.

Currently we are investigating how confidence changes with different n_i distributions. Maybe the overall confidence for a cost calculation can be increased by distributing the total number of experiments to the different faults in a special way.

An advantage of our method is that experiments done by different groups can easily be merged. E.g. if one group investigates the behaviour of the system according to operator faults and the other group investigates hardware faults, the matrix can be extended by adding new lines for each new fault.

The seperation of recording a system's behaviour in a matrix and providing fault rates and costs seems to move work to the benchmarker. Other benchmarks set these values, sometimes implicitely, to fixed values. We think that dependability benchmarking is only useful, if the benchmarker's environment influences the system's assessment. On the other hand trying to apply our method to other methods might help to disclose inherent assumptions.

	SUCCESS	FAILS
Bitflip	48 %	52 %

Tabelle 2: Behaviour of System A

	SUCCESS	FAIL
Bitflip	74 %	26 %

Tabelle 3: Behaviour of System B

4 Example

This section explains the dependencies between the parameters needed to use our approach and shows the usefulness of our approach.

4.1 Fault rate

This section shows that confidence decreases with increasing system complexity: The SUBs are two systems, A and B. The only difference between these systems is that system A has one CPU and system B has an additional and unused second CPU.

We define the following dependability benchmark: The workload consists of the installation of a linux distribution from CDROM to harddisk. The fault load are bitflip faults in a randomly selected CPU register. The observer classifies the system behaviour into two failure modes: successfully installed or not.

We assume a hypothetically measured, very simple matrix of 100 experiments including only one line for that bitflip fault, see 2 and 3.

Investigating the matrix shows that system B seems to perform better according to the selected fault model. But the half value for the FAIL failure mode of system B is invoked by the fact, that the 100 experiments are distributed to both processors and the second processor is not used. Thus, only 50 experiments are done on the critical first processor.

Our approach avoids this missinterpretation already in calculating the failure mode rates $W_{Bitflip}$. Because system B has two processors, the fault rate is double compared to system A. Thus, setting $r_{Bitflip}$ to 1 fault per year leads to the failure mode rates shown in table 4.

Columns 3 and 4 of table 4 show the failure mode rates according to the confidence intervals calculated for a significance of 90% ($t_{0.005,100-1} = 2.626$). As it can be seen, the interval increases by adding additional hardware. Thus, our method confirms the thesis ”more complex systems are harder to assess accurately”.

Table 4 shows that the confidence is poor. To increase confidence many more experiments

	mean(FAILS)	min(FAILS)	max(FAILS)
$W_{Bitflip}^{SystemA}$	0.52 / year	0.39 / year	0.65 / year
$W_{Bitflip}^{SystemB}$	0.52 / year	0.26 / year	0.78 / year

Tabelle 4: Rates of failure mode

DB	*No effect*	*report*	*shutdown*	*corrupt*
Oracle	5.6 %	15.5 %	78.9 %	NO
Postgre	3.2 %	96.1 %	NO	0.7 %

Tabelle 5: System behaviour according to harddisk failures

have to be exercised which comes along with increasing time to evaluate the results. Thus, if time is a limiting factor of dependability assessment, fault injection based evaluation is limited by low confident results.

4.2 Costs

The matrix of the following example was evaluated with a SWIFI-tool, see [SB02]. This tool is able to simulate of-the-shelf Intel hardware running a Linux operating system. Possible faults are high level harddisk faults, power fail, network package losses, bit-flips in memory and cpu registers etc. It uses an automatically experiment controller, see [BDH+02], to create reproducable values for system behaviour recorded in the matrix of our approach.

This example illustrates how to derive assessments for different dependability metrics from the matrix. We benchmarked two systems identical in hardware design but running different database systems: Oracle and Postgre. We injected transient harddisk faults like they appear if the power supply connector is connected sloppily. We did 300 experiments for each database recording table 5.

Table 5 contains the distribution of failure modes for harddisk faults only. Because there is no other fault, we merged the matrices for both systems to one matrix. *no effect* means that no error was observed by the DBE. *report* describes the case that the database was complaining about a disk fault but keeps working. *shutdown* describes the case, that the database complains about a disk fault and shuts itself down. *corrupt* means that the database does not complain about an error but keeps corrupt data. The measured probabilities are recorded by using a SWIFI-Tool, see [FA03]. NO means not observed and is equal to 0%.

Table 6 shows two different cost distributions. Each element of table 6 contains the R_j^{metric}. The line for availability describes a model in which it is expensive if the service is not available while the loss is relatively small, if it supplies wrong data. The second line de-

Failure Mode	*effect*	*report*	*shutdown*	*corrupt*
$R_j^{availability}$	0	0	100	10
$R_j^{consistency}$	0	0	10	100

Tabelle 6: Failure modes' costs for different focusses

Database	availability	consistency
Cost(Oracle)	78,90	7,89
Cost(Postgre)	7,00	70,00

Tabelle 7: Failure modes' costs for different focusses

scribes a model in which data integrity is an important metric while unavailability comes along with relatively small loss. Using the same hardware for both system (equal fault rates) leads to costs/time shown in table 7:

This example shows that the benchmarker himself has to supply the costs, because he is the only one who knows them (loss per transaction). In other words, dependability measures are highly dependent on the dependability metric a benchmarker wants.

5 Conclusion

We presented an abstract view on the process of calculating dependability measures using fault injection based techniques. Seperating fault rates and failure modes from the process of recording system behaviour and, thus, deviding the dependability benchmarking process into two phases has the advantage of exercising the experiments for an application only once. Having the system behaviour recorded, it is easy to calculate dependability measures for different quality of hardware (hardware faults) and IT service (operator faults) by setting their fault rates. Weighting the failure modes by their costs opens the possibility to adapt dependability measures according to the benchmarker's environment in which they are used.

Our approach is the first which is able to estimate the confidence of the dependability assessments. Confidence can be increased by increasing the number of experiments. Unfortunately confidence increases only proportional to the squareroot of the number of experiments. Taking confidence into account for system comparison introduces the matter, that the decision which is the better system might be hidden in statistic noise and, therefore, can't be made.

6 Outlook

While fault rates are supplied for some hardware components by the manufacturer, the corresponding rates for operators can't be referred as easy. Because humans learn with experience fault rates of operator faults are dynamic. Nevertheless, we need fault rates for all kind of faults to weight them against each other. So, we need methods to assess system services to calculate accurate dependability measures.

Including confidence assessments for dependability measures is another import issue. A first worst case approximation would be to calculate the costs according to the edges of a confidence interval around the measured probabilities. But the resulting intervall seems to be very wide, so that statistics noise seems to be very high. Thus, more sophisticated methods must be developed, keeping confidence intervals tight.

Acknowledgement

The research presented in this paper is supported by the European Community (DBench project, IST-2000-25425). We want to thank all the people who contributed to our benchmarking environment FAUmachine and the head of Institute of Computer Architecture Prof. M. Dal Cin.

Literatur

[BDH+02] Buchacker, K., Dal Cin, M., Höxer, H., Sieh, V., und Tschäche, O.: Reproducible dependability benchmarking experiments based on unambiguous benchmark setup descriptions. Internal Report 1/2002. Institut für Informatik 3, Universität Erlangen-Nürnberg. 2002.

[BT03] Buchacker, K. und Tschche, O. Tpc benchmark-c version 5.2 dependability benchmark extensions. URL http://www.faumachine.org/papers/tpcc-depend.pdf. 2003.

[FA03] FAUmachine Team. FAUmachine. URL: http://www.FAUmachine.org/. 2003.

[NLZ+03] Nagaraja, K., Li, X., Zhang, B., Bianchini, R., Martin, R. P., und Nguyen, T. D.: Using Fault Injection and Modeling to Evaluate the Performability of Cluster-Based Services. In: *Proceedings of the Usenix Symposium on Internet Technologies and Systems*. March 2003.

[SB02] Sieh, V. und Buchacker, K.: UMLinux — a versatile SWIFI tool. In: Bondavalli, A. und Thevenod-Fosse, P. (Hrsg.), *Fourth European Dependable Computing Conference, Toulouse, France, October 23-25, 2002*. S. 159–171. Springer Verlag, Berlin. 2002.

[St08] Student: The probable error of a mean. In: *Biometrika 6*. S. 1–25. 1908.

[VM02] Vieira, M. und Madeira, H. DBench-OLTP: A dependability benchmark for OLTP application environments. URL http://eden.dei.uc.pt/ henrique/DBench-OLTP.D1.1.0.pdf. 2002.