OS Dependability Benchmarking
(La sûreté de fonctionnement comme critère de choix d’un OS)

Karama Kanoun
Karama.Kanoun@laas.fr

Research Group on Dependable Computing and Fault Tolerance
Tolérance aux fautes et sûreté de fonctionnement (TSF)

Conférence du 22 octobre 2012, Toulouse
European Project

Definition of a conceptual framework & experimental environment(s) for benchmarking the dependability of COTS and COTS-based systems

Outcomes of DBench

- Concepts, specifications and guidelines for dependability benchmarking
- Dependability benchmark prototypes

• Partners
  LAAS-CNRS, France (Coordinator), Critical Software, Portugal, University of Coimbra, Portugal, Friedrich Alexander University, Germany Polytechnic University of Valencia, Spain

• Advisory Board
  Astrium (F), CMU (USA), INDRA (E), Oracle (P), Saab Ericsson Space (S), Thales (F)
Benchmarking Objectives

- Explore system dependability capabilities
- Assess & evaluate dependability measures
- Identify weakness
- Quantify
- Compare

Benchmarking the dependability of a system consists in evaluating dependability or performance-related measures experimentally or based on experimentation and modeling → characterize objectively the system behavior in the presence of faults.

⇒ non-ambiguous comparison of alternative solutions
Benchmarks Developed

- General-purpose operating systems
  - Robustness and timing measures, faulty application, faulty drivers
- Real-Time kernels in onboard space systems
  - Predictability of the kernel response time, faulty application
- Engine control applications in automotive systems
  - Impact of application failures on system safety, transient hardware faults
- On-line transaction processing (OLTP) environments
  - TPC-C-based, operator, software & hardware faults
  - Web-servers, SPEC-based, operator, software & hardware faults

http://www.laas.fr/DBench/
Software Systems Dependability Evaluation

Information on software behavior

- Field data
- Data from development
- Controlled experiments

Ad hoc

Standard

Dependability benchmarking
Evaluation of dependability measures / features
in a non-ambiguous way → comparison

Properties

Reproducibility, portability, representativeness, acceptable cost
OS Benchmarking: User Point of View

Computer System

- Limited knowledge: functional description
- Limited accessibility and observability
- Limited intrusiveness and interference

⇒ Black-box approach ⇒ robustness benchmark

Operating System

Linux

Mac

Windows

Which OS for my computer system?
Benchmarking wrt class of faults?

Wrt application erroneous behavior
Measures

- POS: OS Robustness [%SEr %SXP %SPc %SHg %SNS]
- Texec: OS reaction time in the presence of faults
- Tres: OS Restart time after fault insertion
Execution Profile

• Workload
  – TPC-C Client, Java Virtual Machine, PostMark

• Faultload
  – Corruption of parameters of all system calls

Selective substitution

Out-of-range Data (ORD)  Incorrect Data (ID)  Incorrect Address (IA)
## Dependability Benchmarks with PostMark WL

<table>
<thead>
<tr>
<th>Operating System</th>
<th># system calls</th>
<th># experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows NT 4</td>
<td>25</td>
<td>418</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>$25 + 1 + 1$</td>
<td>433</td>
</tr>
<tr>
<td>Windows XP</td>
<td>$25 + 1$</td>
<td>424</td>
</tr>
<tr>
<td>Windows NT 4 Server</td>
<td>25</td>
<td>418</td>
</tr>
<tr>
<td>Windows 2000 Server</td>
<td>$25 + 1 + 1$</td>
<td>433</td>
</tr>
<tr>
<td>Windows 2003 Server</td>
<td>$25 + 1 + 1$</td>
<td>433</td>
</tr>
<tr>
<td>Linux 2.2.26</td>
<td>$15 + 1$</td>
<td>206</td>
</tr>
<tr>
<td>Linux 2.4.5</td>
<td>$15 + 1$</td>
<td>206</td>
</tr>
<tr>
<td>Linux 2.4.26</td>
<td>$15 + 1$</td>
<td>206</td>
</tr>
<tr>
<td>Linux 2.6.6</td>
<td>$15 + 2$</td>
<td>228</td>
</tr>
</tbody>
</table>
Experimental set-up

Host Machine

- Activity (Workload)
  - Interception & Substitution of system calls & Observation OS reaction

- API
- Target Operating System
- Hardware

System under benchmarking

Control Machine
Measurements

Experiments with Workload completion

- System Call to intercept
- OS Reaction time
- Workload Completion Time
- Restart time
- Workload End
- tWStart (n)
- tResume (n)
- tResponse (n)
- tExpEnd (n)
- tExpStart (n+1)
Experiments without Workload completion

Timeout >> Workload completion duration

OS Reaction time

Restart time

System Call to intercept

Workload End
Robustness (WL = PostMark)

**Windows**

- **Windows NT4**
  - Hang/Panic: 0.0%
  - Exception: 17.5%
  - No Signaling: 55.5%
  - Error Code: 27.0%

- **Windows 2000**
  - Hang/Panic: 0.0%
  - Exception: 20.3%
  - No Signaling: 55.2%
  - Error Code: 24.5%

- **Windows XP**
  - Hang/Panic: 0.0%
  - Exception: 20.8%
  - No Signaling: 56.1%
  - Error Code: 23.1%

- **NT4 Server**
  - Hang/Panic: 0.0%
  - Exception: 17.5%
  - No Signaling: 55.5%
  - Error Code: 27.0%

- **2000 Server**
  - Hang/Panic: 0.0%
  - Exception: 20.3%
  - No Signaling: 55.2%
  - Error Code: 24.5%

- **2003 Server**
  - Hang/Panic: 0.0%
  - Exception: 21.9%
  - No Signaling: 54.5%
  - Error Code: 23.6%

**Linux**

- **Linux 2.2.26**
  - Hang/Panic: 0.0%
  - Exception: 7.8%
  - No Signaling: 7.8%
  - Error Code: 67.5%

- **Linux 2.4**
  - Hang/Panic: 0.0%
  - Exception: 7.8%
  - No Signaling: 26.2%
  - Error Code: 66.0%

- **Linux 2.6.6**
  - Hang/Panic: 0.0%
  - Exception: 9.7%
  - No Signaling: 31.6%
  - Error Code: 58.8%
OS Reaction Time (WL = PostMark)

In the presence of faults
Without parameter corruption
Restart Time (WL = PostMark)

Windows

Linux

- **In the presence of faults**
- **Without parameter corruption**
Restart Time (WL = PostMark)

Windows XP

Linux 2.2.26

Workload Abort/hang

check disk
Validation of properties

➢ Reproducibility
  ✓ By construction
  ✓ Set of faults
    • System Calls to be corrupted
    • Substitution values

➢ Repeatability
  ✓ Each benchmark has been executed three times
    • Same robustness
    • Variation of the reaction time (< 4% for TPC-C client)
    • Variation of the restart time (< 3% for TPC-C client)
Validation - Sensitivity Analyses wrt Faultload

<table>
<thead>
<tr>
<th>Parameter corruption type</th>
<th># experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows NT4</td>
</tr>
<tr>
<td>Incorrect Data</td>
<td></td>
</tr>
<tr>
<td>Incorrect Address</td>
<td></td>
</tr>
<tr>
<td>Out-of-range Data</td>
<td></td>
</tr>
<tr>
<td>Faultload 0</td>
<td></td>
</tr>
<tr>
<td>Faultload 1</td>
<td></td>
</tr>
<tr>
<td>Faultload 2</td>
<td></td>
</tr>
</tbody>
</table>

- Equivalence of versions of the same family
- Same comparison results between the two families

Additional analysis: incorrect data = out-of-range data in the context
Validation - Cost/effort

- Benchmark implementation duration
  - For each OS family
    - Implementation of the workload: 1 - 3 days
    - Controller, parameter corruption and observation: 2 weeks
    - Definition and implementation of fault set: 1 week

- Experiment duration
<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-C client</td>
<td>2 days</td>
<td>1 day</td>
</tr>
<tr>
<td>Postmark</td>
<td>2 days</td>
<td>1 day</td>
</tr>
<tr>
<td>JVM</td>
<td>4 days</td>
<td>2 days</td>
</tr>
</tbody>
</table>
Other Benchmarks

- Other workloads
  - API
  - Other OSs (FreeBSD, OpenBSD)
  - Device drivers
- Hardware