Dependability Benchmarking of Operating Systems

Karama KANOUN
LAAS-CNRS

Workshop on Dependability Benchmarking
In conjunction with the 16th IEEE International Symposium on Software Reliability Engineering
Chicago, IL, USA — November 8, 2005
Context

**DBench (Dependability Benchmarking)**
European IST Program — Project IST-2000-25425
January 2001- March 2004

**Consortium**
Critical Software (P), University of Coimbra (P), Friedrich-Alexander University, (D), LAAS-CNRS (F), Polytechnic University of Valencia (E)

**Main results**
Benchmarking framework
A set of benchmarking principles common to all DBench benchmarks
A set of benchmarks
DBench Benchmarking Framework

- **Modeling**
  - Dependability model of the system

- **Experimentation**
  - Prototype of the system
  - Experimental result processing

- **Faultload**

- **Workload**

- **Model processing**
  - Modeling measures
  - Experimental measures
  - Benchmark measures
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

⇒ Properties: reproducibility, repeatability, portability,
   representativeness, acceptable cost/effort

Operating System

- Linux
- Windows
- Mac
Benchmarking wrt Class of Faults?

Wrt application erroneous behavior
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
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

---

Out-of-range Data (ORD)  
Incorrect Data (ID)  
Incorrect Address (IA)  

Selective substitution
## Dependability Benchmarks with PostMark WL

<table>
<thead>
<tr>
<th></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>
Experiments with Workload completion
Experiments without Workload completion

Measurements

System Call to intercept

OS Reaction time

Workload End

Timeout >> Workload completion duration

Restart time

Experiment End

$tWStart(n)$

$tResume(n)$

$tResponse(n)$

$tExpEnd(n)$

$tExpStart(n+1)$
Robustness (WL = PostMark)

Windows

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

- **NT4 Server**
  - 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%

- **2000 Server**
  - 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%

- **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: 24.8%
  - 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)

Windows

Linux

In the presence of faults

Without parameter corruption
Restart Time \((WL = \text{PostMark})\)

Windows

<table>
<thead>
<tr>
<th>OS</th>
<th>Without Parameter Corruption</th>
<th>In the presence of faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT 4</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>2000</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>XP</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>NT4 Server</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>2000 Server</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>2003 Server</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

Linux

<table>
<thead>
<tr>
<th>OS</th>
<th>Without Parameter Corruption</th>
<th>In the presence of faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.26</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>2.4.5</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>2.4.26</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>2.6.6</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

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>Incorrect Data</td>
</tr>
<tr>
<td>Faultload 0</td>
<td>x</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>
Extensions

Other workloads

API

Other OSs

Device drivers

Hardware
Extensions

• Application level:
  – Other workloads: FreeBSD, OpenBSD
  – Other faultloads stressful conditions/disturbances
  – Error propagation between applications
• Malicious faults
• Device drivers
  – Interface between OS and drivers
  – Impact of not well-debugged drivers
    • On the application
    • On other drivers
• Hardware
  – Simulation of exceptions due to the hardware fault occurrence
  – Transient faults
• Vender point of view?
References


