Debugging in the (Very) Large: Ten Years of Implementation and Experience

Galen C. Hunt
Principal Researcher
Microsoft Corporation
Paper originally appeared in 22nd ACM Symposium on Operating Systems Principles

Co-authors (and system inventors):

• Kirk Glerum
• Kinshuman Kinshumann
• Steve Greenberg,
• Gabriel Aul
• Vince Orgovan
• Greg Nichols
• David Grant
• Gretchen Loihle
A Revelation

Software has bugs

Even **shipping** software

Even **Microsoft’s** shipping software

Oh, and so does hardware
(but we’ll come back to that point later...)
Bug: a **flaw** in program logic

```c
#define MYVAR *(int*)random()
...
MYVAR = 5;
```

Error: a **failure in execution** caused by a bug
- run it 5,000 times, you’ll get ~ 5000 errors

One bug may cause **many** errors
The Challenge

Microsoft ships software to 1 billion users,

- how do we find out when things go wrong?

We want to

- fix bugs regardless of source
  - application or OS
  - software, hardware, or malware
- prioritize bugs that affect the most users
- generalize the solution to be used by any programmer
## Reported Bugs

<table>
<thead>
<tr>
<th>Error</th>
<th>Reporting Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Crashes</td>
<td>Crash dump found in page file on boot</td>
</tr>
<tr>
<td>Application Crashes</td>
<td>Unhandled process exception</td>
</tr>
<tr>
<td>Application Hangs</td>
<td>Failure to process user input for 5 seconds</td>
</tr>
<tr>
<td>Service Hangs</td>
<td>Service thread times out</td>
</tr>
<tr>
<td>Installation Failures</td>
<td>OS or application installation fails</td>
</tr>
<tr>
<td>App. Compat. Issues</td>
<td>Program calls deprecated API</td>
</tr>
<tr>
<td>Custom Errors</td>
<td>Program calls WER APIs to report error</td>
</tr>
<tr>
<td>UI Delays</td>
<td>Timer assert takes longer than expected</td>
</tr>
<tr>
<td>Invariant Violations</td>
<td>Ship assert in code fails</td>
</tr>
</tbody>
</table>
Do you want to send more information about the problem?

Additional details about what went wrong can help Microsoft create a solution.

- Show Details
- Send information
- Cancel

**Test.exe** has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

**Please tell Microsoft about this problem.**
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).
<table>
<thead>
<tr>
<th>Tier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Billions</td>
<td>Error reports collected per year (App, OS, HW)</td>
</tr>
<tr>
<td>1 billion</td>
<td>Clients</td>
</tr>
<tr>
<td>100 million</td>
<td>Reports /day processing capacity</td>
</tr>
<tr>
<td>17 million</td>
<td>Programs with error reports in WER</td>
</tr>
<tr>
<td>Many 1000s</td>
<td>Bugs fixed</td>
</tr>
<tr>
<td>Over 700</td>
<td>Companies using WER</td>
</tr>
<tr>
<td>200</td>
<td>TB of Storage</td>
</tr>
<tr>
<td>60</td>
<td>Servers</td>
</tr>
<tr>
<td>10</td>
<td>Years of use</td>
</tr>
<tr>
<td>2</td>
<td>Servers to record every error received</td>
</tr>
<tr>
<td>1</td>
<td># of programmers needed to access WER data</td>
</tr>
</tbody>
</table>
Introduction
How do we process billions of error reports?
Experiences fixing bugs from
  • software
  • hardware
  • malware
Conclusion
Debugging in the Small...
In the Large without WER...

Support technician tries to diagnose error

User calls technical support

Technicians report "top ten" issues to programmers
Can’t hire enough technicians
Data is inaccurate
Hard to get additional data
No “global” baseline
Useless for heisenbugs

Need to remove humans
Goal: Fix the Data Collection Problem

Allow one service to record

- **every error** (application, OS, and hardware)
- on every **Windows system**
- **Worldwide**

Corollary:

- **that which we can measure, we can fix...**
An Outlook Plug-in Example

**plugin.dll:**

```c
#define MYVAR *(int*)random()
...
void foo(int i, int j)
{
    if (i & 1)
        memcpy(&MYVAR, j, 4);
    else
        ...
}
```
Debugging in the Large with WER...
Engine for WER bucketing heuristics

Extension to the *Debugging Tools for Windows*

- input is a minidump, output is bucket ID
- runs on WER servers (and programmers desktops)
- [http://www.microsoft.com/whdc/devtools](http://www.microsoft.com/whdc/devtools)

500 heuristics

- grows ~ 1 heuristic/week
To Recap and Elaborate...

What I told you:

- client automatically collects a minidump
- sends minidump to servers
- `!analyze buckets` the error with similar reports
- increments the bucket count
- programmers `prioritize` buckets with highest count

Actually...

- only upload first few hits on a bucket, others just inc.
- programmers request additional data as needed
2-Phase Bucketing Strategy

**Labeling** (on client): bucketed by failure point

```
outlook.exe,plugin.dll,v1.0.2305,0x23f5
{program name},{binary},{version},{pc offset}
```

**Classifying** (on servers):
re-bucketed toward root cause by `!analyze`

- consolidate version and replace offset with symbol
  ```
  outlook.exe,plugin.dll,memcpy
  ```
- find caller of `memcpy` (because it isn’t buggy)
  ```
  outlook.exe,plugin.dll,foo
  ```
- etc.

SOSP paper contains much more detail on bucketing...
One bug can hit multiple buckets

- up to 40% of error reports
  ```c
  memcpy(&MYVAR, j, 4);
  ```
  - one bucket when &MYVAR is illegal address
  - many others when &MYVAR is in a data section
- extra server load
- duplicate buckets must be hand triaged

Multiple bugs can hit one bucket

- up to **4%** of error reports
- harder to isolate each bug

Solution: scale is our friend
Introduction

How do we process billions of error reports?
Experiences fixing bugs from
  • software
  • hardware
  • malware

Conclusion
Top 20 Buckets for MS Word 2010

3-week internal deployment to 9,000 users

Just 20 buckets account for **50% of all errors**
Fixing a small # of bugs will help **many users**
Fixing bugs in software

First use found $\geq$ 5-year old heisenbugs in Windows
Windows Vista team fixed 5,000 bugs in beta
Anti-Virus vendor fixed top 20 buckets and dropped from 7.6% to 3.6% of all kernel crashes
Office 2010 team fixed 22% of reports in 3 weeks
And you can fix yours...
WER helped fix hardware error
Manufacturer *could* have caught this earlier w/ WER
Other Hardware Bugs

SMBIOS
  • memory overrun in resume-from-sleep

Motherboard USB controller
  • only implemented 31 of 32 DMA address bits

Lots of information about failures due to
  • overclocking
  • hard disk controller resets
  • substandard memory
Early detection w/o user action (renos, blaster, slammer, etc.)
WER **scales** to handle global events
Other Things in the SOSP Paper

**Bucketing details** (Sec. 3)

**Statistics-based debugging** (Sec. 4)

**Progressive data collection** (Secs. 2.2 & 5.4)

Service implementation (Sec. 5)

WER experiences (Sec. 6)

**OS Changes** (Sec. 7)

Related work (Sec. 8)
Windows Error Reporting (WER)

- the **first** post-mortem reporting system with automatic diagnosis
- the **largest** client-server system in the world (by installs)
- helped 700 companies fix **1000s** of bugs and **billions** of errors
- **fundamentally changed SW development at MS**

WER works because bucketing *mostly* works

[http://winqual.microsoft.com](http://winqual.microsoft.com)