Using Model Checking to Find Serious File System Errors

Phil Gibbons

15-712 F15

Lecture 7

Today's Reminders

- **My office hours**
  - 4:30-5:30 pm GHC 7221
- **Only one paper to read/summarize for Friday**
  - Read Chandy/Lamport paper
- **Should receive feedback on one summary today**

Using Model Checking to Find Serious File System Errors

Best Paper

Junfeng Yang, Paul Twohey, Dawson Engler, Madanlal Musuvathi [OSDI'04]

- **Junfeng Yang (Columbia, Parrot [SOSP'13] w/CMU)**

- **Paul Twohey (Head of Tech @ ClassPass)**

- **Dawson Engler (Stanford, Mark Weiser Award 2006)**
  - SigOps innovation award, also won by: Stefan Savage, Tom Anderson, Mike Burrows

- **Madanlal Musuvathi (Microsoft Research)**

File System Stress Tests

"File system errors are some of the most destructive errors possible."

- **Prior file system stress test frameworks (Linux Test Project, Stress debian package) focus mostly on non-crash based errors**
  - Cost of crash-reboot-reconstruct cycle limits stress testing

Use Model Checking

- **Formal verification technique that systematically enumerates the possible states of a system by exploring the nondeterministic events in the system**
  - CMU Prof. Edmund Clark won 2007 Turing Award

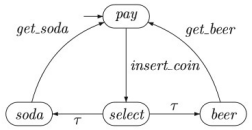


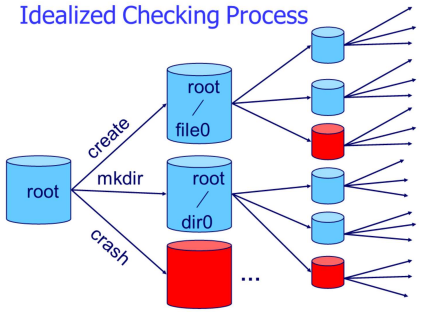
Figure 2.1: A transition system of a simple beverage vending machine.

- **Example Check:**
  Vending machine delivers a drink only after inserting a coin

Use Model Checking

- **Implementation-level model checkers: check actual code (not abstract specification of code)**
  - E.g., CMC model checker of Engler, Musuvathi, etc [OSDI'02] can run entire OS inside of it
- **Challenges**
  - Defining the reference model to check against
  - Keeping the number of states manageable (State reduction)
  - Minimizing exploration time (Prioritizing the search)
- **Paper presents File System Checker (FiSC)**
  - Found serious bugs in JFS, ReiserFS, ext3

Ideally: Explore & Check All Possible States

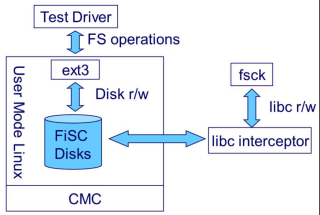


Reality: Must Simplify & Focus Search
(can't explore ALL states – Bug finding NOT verification)

Some slide images from Yang's OSDI talk

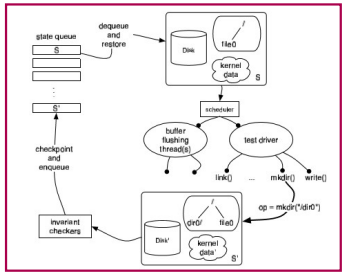
Checking Overview

- **CMC, an explicit state model checker running Linux kernel**
- **File system test driver**
  - Creates, removes, renames files /directories/hard links
  - Writes to and truncates files
  - Mounts & unmounts file system



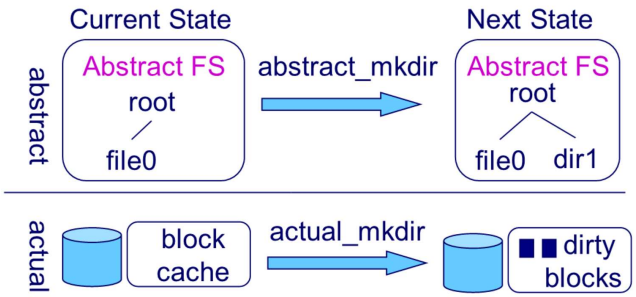
- **Permutation checker** (not shown)
  - Verifies that file system can recover no matter what order buffer cache contents are written to disk
- **fsck (file system consistency check) recovery checker**
  - Run on host system; capture disk accesses while repairing and feed into crash recovery checker

State Exploration & Checking Overview



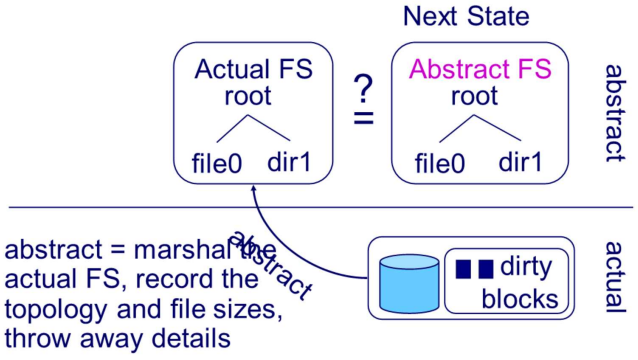
- **Pick a state S & iteratively generate each successor state by applying each possible operation to a restored copy of S**
- **Check the generated state S' for validity**
- **If S' is valid & not explored, insert S' into state queue**

Checking FS Operations are Correct



- Abstract FS: model of a file system. Currently tracks topology and file sizes. Can be extended
- Reference model, run in parallel with the actual FS

Checking FS Operations are Correct



- Generic, implemented by FiSC

Basic Setup for Checking New FS

- **Input to FiSC: minimum disk & memory sizes FS requires**
  - E.g., 2 MB disk and 16 pages of memory for ext3
- **Input to FiSC: Command to make and recover the FS**
  - E.g., mkfs & fsck
- **Modify FS to expose dirty blocks (if doesn't already) & independently manage 2 disks in a reentrant manner**
- **Modify FS to call into model checker to indicate StableFS changes**
  - StableFS = state the FS *should* recover to after crash

Time To Do: 1-2 weeks

Checking More Thoroughly

- **Downscale**
  - Small disks: make minimum size allowed
  - Small FS topologies: typically 2-4 nodes
  - Small virtual memory: few pages
- **Canonicalization**
  - Only write two different values to data blocks
  - Renaming files/directories to sequential numbering
  - Zeroing freed memory & unused disk blocks
  - Removing time fields, generation numbers, etc
- **Expose choice points**
  - FS calls choose(n), for n different alternatives

Choice Points

- Choice point = can abstractly do multiple actions, practically does one
- Want to explore all actions

```
struct block* read_block (int i) {
  struct block *b;
  if ((b = cache_lookup(i)))
    if (fisc_choose(2) == 0)
      return b;
  return disk_read (i);
}
```

return twice,
1st time return 0,
2nd time return 1

if there are N possible actions, call fisc_choose(N) return 0, 1, …, N-1

Checkers

- **Generic Checks**
  - Deadlock, NULL pointer, Paired functions, Memory leak
  - No silent failures: if resource alloc fails, then sys call should fail
- **Consistency Checks**
  - System calls map to actions
  - Changed buffers marked dirty
  - Buffer consistency
  - Double fsck (replay journal vs. entire disk)
  - Recoverable disk write ordering (permutation checker)

Basic Crash Recovery Check

"A classic recovery mistake is to incorrectly handle a crash that happens during recovery."

- Obtain a crashed disk image D
- Run fsck, recording all writes
- Simulate a crash during recovery
  - Apply prefix to D
  - Re-run fsck
  - Compare to Stable FS
- Repeat until all the prefixes are tried
- Effective☺, Speed☹ (redundant crashes)

Progress Before Run Out of Memory?

	ext3	ReiserFS	JFS
States			
Total	10800	630	4500
Expanded States	2419	142	905
State Transitions	35978	11009	14387
Time			
with Memoization	650	893	3774
without Memoization	7307	29419	4343

FS Errors Found by FiSC

Error Type	VFS	ext2	ext3	JFS	Reiser	total
Data loss	N/A	N/A	1	8	1	10
False clean	N/A	N/A	1	1		2
Security		2	2	1		3 + 2
Crashes	1			10	1	12
Other	1		1	1		3
Total	2	2	5	21	2	32

32 in total, 21 fixed, 9 of the remaining 11 confirmed

Recovery Write Ordering Bug

- Under Normal operation:
  - Changes must first be flushed to log before they can reach the actual FS
- All FS seem to get this right
- During Recovery:
  - Changes must first be flushed to the actual FS before the log can be cleared
- Found this type of bug in all FS, total 5

Towards Optimization-Safe Systems: Analyzing the Impact of Undefined Behavior

Xi Wang, Nickolai Zeldovich, Frans Kaashoek, Armando Solar-Lezama [SOSP'13 best paper]

(slides & video of talk is on SOSP'13 webpage)

Optimization-Unstable Code

Example: compiler discards sanity check

```
char *buf      = ...;
char *buf_end  = ...;
unsigned int off = /* read from untrusted input */;
if (buf + off >= buf_end)
  return;          /* validate off: buf+off too large*/
if (buf + off < buf)
  return;          /* validate off: overflow, buf+off wrapped around */
/* access buf[0..off-1] */
```

Examples of Undefined Behavior in C

Meaningless checks from real code: pointer p; signed integer x

Pointer overflow:	`if (p + 100 < p)`
Signed integer overflow:	`if (x + 100 < x)`
Oversized shift:	`if (!(1 << x))`
Null pointer dereference:	`*p; if (p)`
Absolute value overflow:	`if (abs(x) < 0)`

Problem: Unstable Code

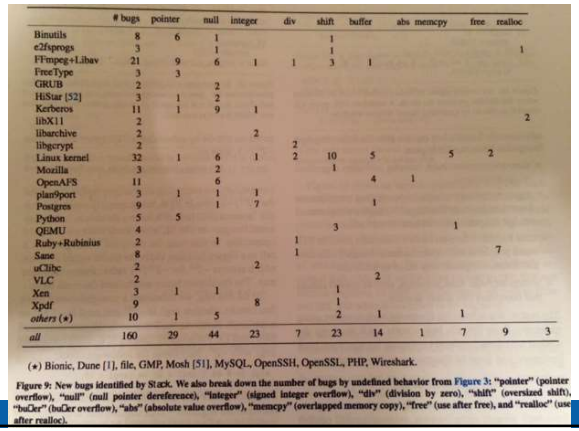
Unstable code: compilers discard code due to undefined behavior

- Security checks discarded
- Weakness amplified
- Unpredictable system behavior

Compilers Often Discard Unstable Code

	`if(p+100<p)`	`if(x+100<x)`	`if(!(1<<x))`	`*p; if(!p)`	`if(abs(x)<0)`
gcc-4.8.1	O2	O2		O2	O2
clang-3.3	O1	O1	O1		
aCC-6.25					O3
armcc-5.02		O2			
icc-14.0.0		O1		O2	
msvc-14.0.0				O1	
open64-14.0.0	O1	O2			O2
pathcc-1.0.0	O1	O2			O2
suncc-5.12				O3	
ti-7.4.2	O0	O0			
windriver-5.9.2		O0			
xlc-12.1	O3				

STACK (Static Checker for Unstable Code) Found 160 New Bugs



	# bugs	pointer	null	integer	div	shift	buffer	abs	memcpy	free	realloc
Binutils	8	6	1			1					
e2fsprogs	3		1			1					1
FFmpeg+Libav	21	9	6	1	1	3	1				
FreeType	3	3									
GRUB	2		2								
HiStar [52]	3	1	2								
Kerberos	11	1	9	1							
libX11	2										2
libarchive	2			2							
libgcrypt	2				2						
Linux kernel	32	1	6	1	2	10	5		5	2	
Mozilla	3		2			1					
OpenAFS	11		6				4	1			
plan9port	3	1	1	1							
Postgres	9		1	7			1				
Python	5	5									
QEMU	4					3			1		
Ruby+Rubinius	2		1		1						
Sane	8				1					7	
uClibc	2			2							
VLC	2						2				
Xen	3	1	1			1					
Xpdf	9			8		1					
others (*)	10	1	5			2	1		1		
all	160	29	44	23	7	23	14	1	7	9	3

(*) Bionic, Dune [1], file, GMP, Mosh [51], MySQL, OpenSSH, OpenSSL, PHP, Wireshark.

Figure 9: New bugs identified by STACK. We also break down the number of bugs by undefined behavior from Figure 3: "pointer" (pointer overflow), "null" (null pointer dereference), "integer" (signed integer overflow), "div" (division by zero), "shift" (oversized shift), "buCer" (buCer overflow), "abs" (absolute value overflow), "memcpy" (overlapped memory copy), "free" (use after free), and "realloc" (use after realloc).

Friday's Paper

Distributed Snapshots:
Determining Global States of Distributed Systems

Mani Chandy & Leslie Lamport [TOCS'85]

Quote for the Day

"There are only two hard problems in distributed systems:

2. Exactly-once delivery

1. Guaranteed order of messages

2. Exactly-once delivery"