A Study of Linux File System Evolution

Lanyue Lu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Shan Lu

Mateusz Banaszek

15 grudnia 2016

"Motto"

Documentation: fix formatting to make 's' happy

```
"That letter [the last s] is sad because all the others
have those things [=] below them and it does not."

This patch fixes the tragedy so all the letters can
be happy again.

Signed-off-by: Maisa Roponen <maisa.roponen@gmail.com>
[The author being 4 years old needed some assistance]
Signed-off-by: Tero Roponen <tero.roponen@gmail.com>
Signed-off-by: Jonathan Corbet <corbet@lwn.net>
```

Diffstat

-rw-r--r-- Documentation/filesystems/proc.txt 2

1 files changed, 1 insertions, 1 deletions

```
diff --git a/Documentation/filesystems/proc.txt b/Documentation/filesystems/proc.txt
index eb8a10e..aae9dd1 100644
--- a/Documentation/filesystems/proc.txt
+++ b/Documentation/filesystems/proc.txt
@@ -1272,7 +1272,7 @@ softirq.


 1.9 Ext4 file system parameters
-------------------------------
+-------------------------------

 Information about mounted ext4 file systems can be found in
 /proc/fs/ext4.  Each mounted filesystem will have a directory in
```

Plan

1. Założenia badania
2. Analiza łat (ogólnie)
3. Łaty *Bug*
4. Łaty *Performance*
5. Łaty *Reliability*
6. Najczęstsze

Założenia badania

Założenia badania

- systemy plików w Linuksie $\rightarrow$ otwarte oprogramowanie, krytyczny komponent, rozwijany, tworzone przez różnych autorów

Założenia badania

- systemy plików w Linuksie $\rightarrow$ otwarte oprogramowanie, krytyczny komponent, rozwijany, tworzone przez różnych autorów
  1. ReiserFS
  2. Ext3
  3. XFS
  4. JFS
  5. Ext4
  6. Btrfs

Założenia badania

- systemy plików w Linuksie → otwarte oprogramowanie, krytyczny komponent, rozwijany, tworzone przez różnych autorów
  1. ReiserFS
  2. Ext3
  3. XFS
  4. JFS
  5. Ext4
  6. Btrfs
- - Jakie łaty są najczęstsze?
    Czy łaty różnych typów są różnych rozmiarów?
  - Jakie rodzaje błędów są w systemach plików?
    Czy któreś komponenty mają więcej błędów niż pozostałe?
    Jakie konsekwencje mają te błędy?
  - Jakich technik używają systemy plików by zwiększyć wydajność?
    Jakie są pomysły na zwiększenie niezawodności systemów plików?

Założenia badania

- systemy plików w Linuksie → otwarte oprogramowanie, krytyczny komponent, rozwijany, tworzone przez różnych autorów
  1. ReiserFS
  2. Ext3
  3. XFS
  4. JFS
  5. Ext4
  6. Btrfs
- - Jakie łaty są najczęstsze?
    Czy łaty różnych typów są różnych rozmiarów?
  - Jakie rodzaje błędów są w systemach plików?
    Czy któreś komponenty mają więcej błędów niż pozostałe?
    Jakie konsekwencje mają te błędy?
  - Jakich technik używają systemy plików by zwiększyć wydajność?
    Jakie są pomysły na zwiększenie niezawodności systemów plików?
- Linux 2.6.0 (Dec '03) – 2.6.39 (May '11) → **5079 łat**

Założenia badania

- systemy plików w Linuksie → otwarte oprogramowanie, krytyczny komponent, rozwijany, tworzone przez różnych autorów
  1. ReiserFS
  2. Ext3
  3. XFS
  4. JFS
  5. Ext4
  6. Btrfs
- - Jakie łaty są najczęstsze?
    Czy łaty różnych typów są różnych rozmiarów?
  - Jakie rodzaje błędów są w systemach plików?
    Czy któreś komponenty mają więcej błędów niż pozostałe?
    Jakie konsekwencje mają te błędy?
  - Jakich technik używają systemy plików by zwiększyć wydajność?
    Jakie są pomysły na zwiększenie niezawodności systemów plików?
- Linux 2.6.0 (Dec '03) – 2.6.39 (May '11) → **5079 łat**
- `http://research.cs.wisc.edu/wind/Traces/fs-patch/`

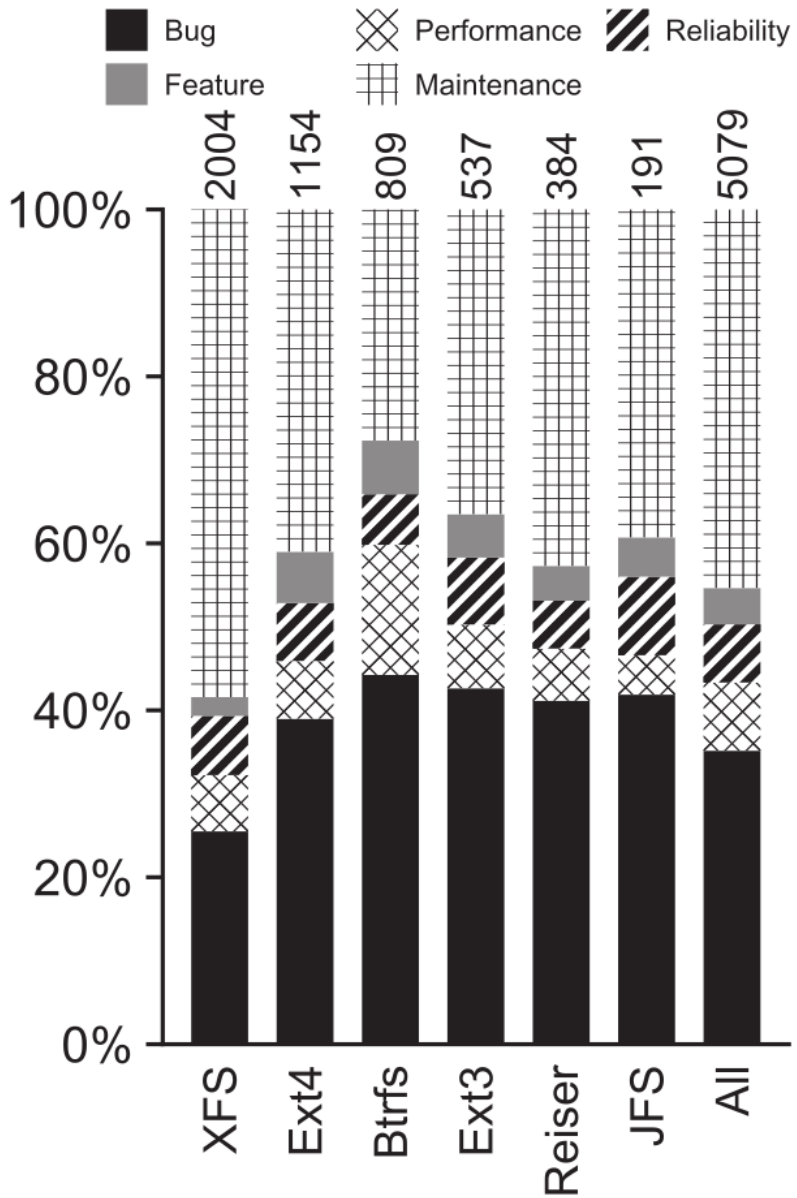
Analiza łat

> Oh, and the patch is obviously entirely untested. I wouldn't want to ruin my reputation by *testing* the patches I send out. What would be the fun in that?
>
> Linus Torvalds

Rodzaje łat

Bug (Błąd)	Naprawiają istniejące błędy
Performance (Wydajność)	Poprawiają wydajność
Reliability (Niezawodność)	Zwiększają odporność na niepożądane sytuacje
Feature (Funkcje)	Implementują nowe funkcje
Maintenance (Pielęgnacja)	Pielęgnują kod i dokumentację

Rodzaje łat


Bug
Feature
Performance
Maintenance
Reliability
2004
1154
809
537
384
191
5079
100%
80%
60%
40%
20%
0%
XFS
Ext4
Btrfs
Ext3
Reiser
JFS
All

Tendencje

- Liczba łat rośnie

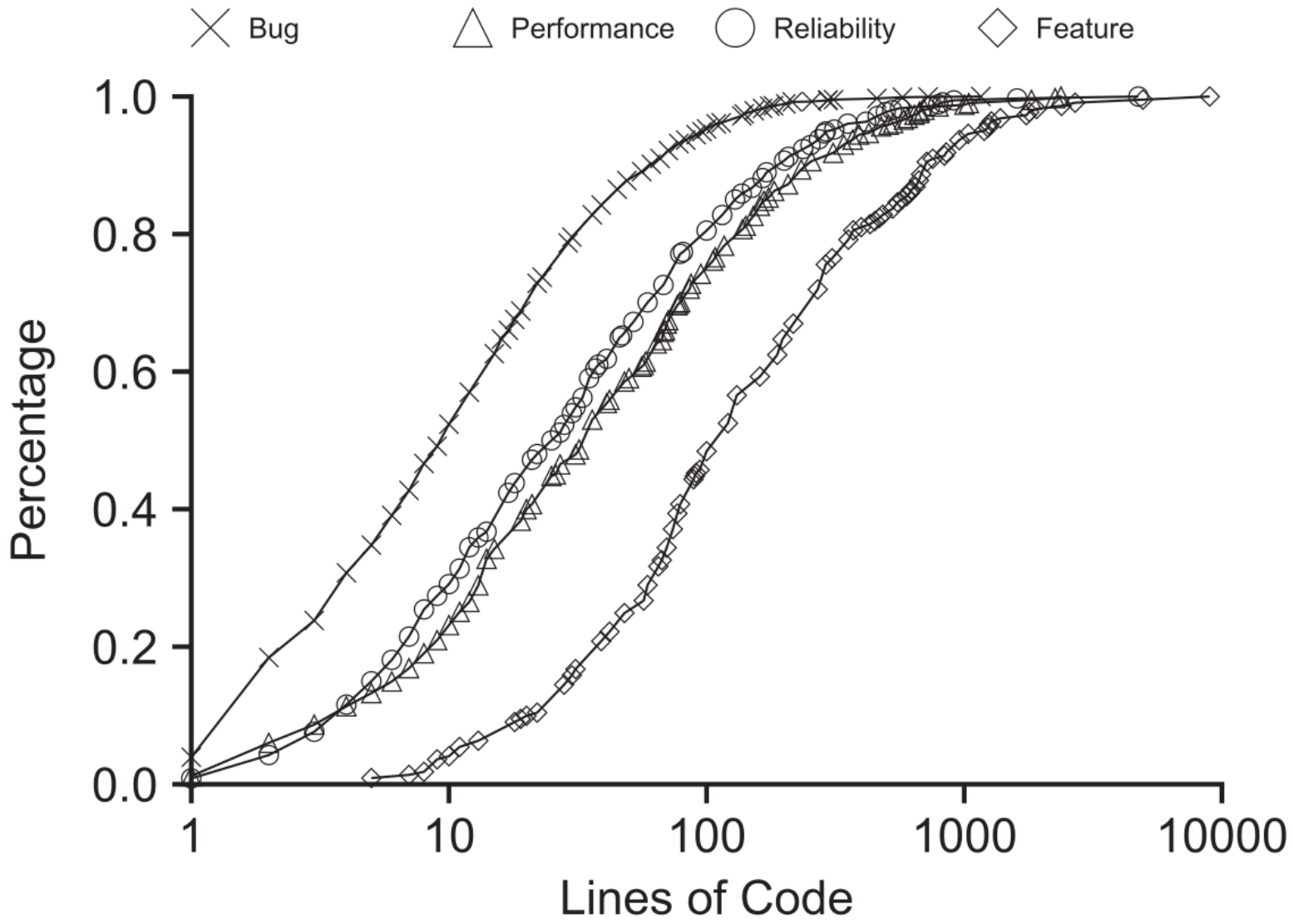
Tendencje

- Liczba łat rośnie
- Proporcje pomiędzy liczbą łat poszczególnych rodzajów są stałe

Tendencje

- Liczba łat rośnie
- Proporcje pomiędzy liczbą łat poszczególnych rodzajów są stałe
- Liczba łat naprawiających błędy nie zmniejsza się

Rozmiar łat

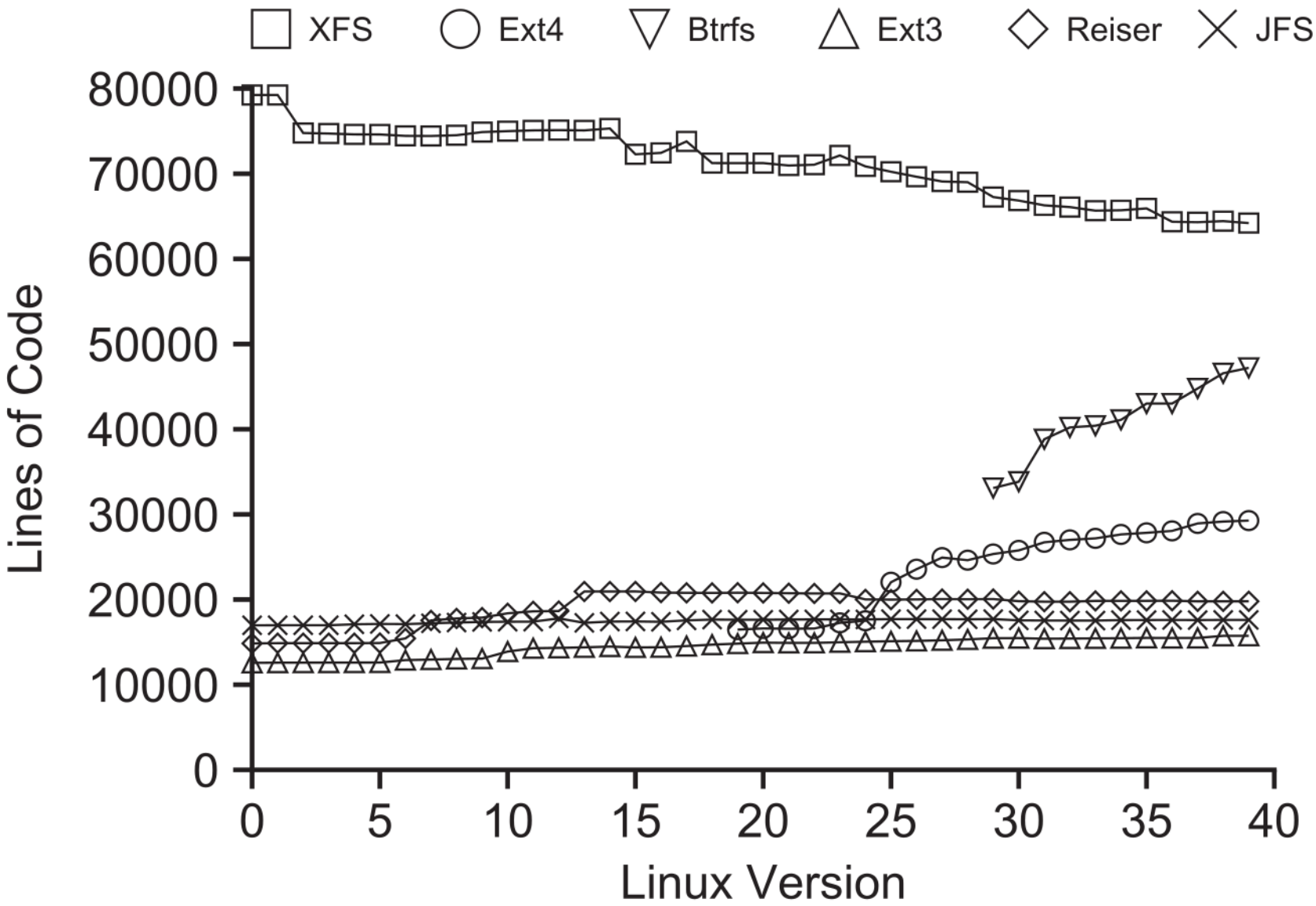

Bug
Performance
Reliability
Feature
1.0
0.8
0.6
0.4
0.2
0.0
Percentage
1
10
100
1000
10000
Lines of Code

Łaty *Bug*

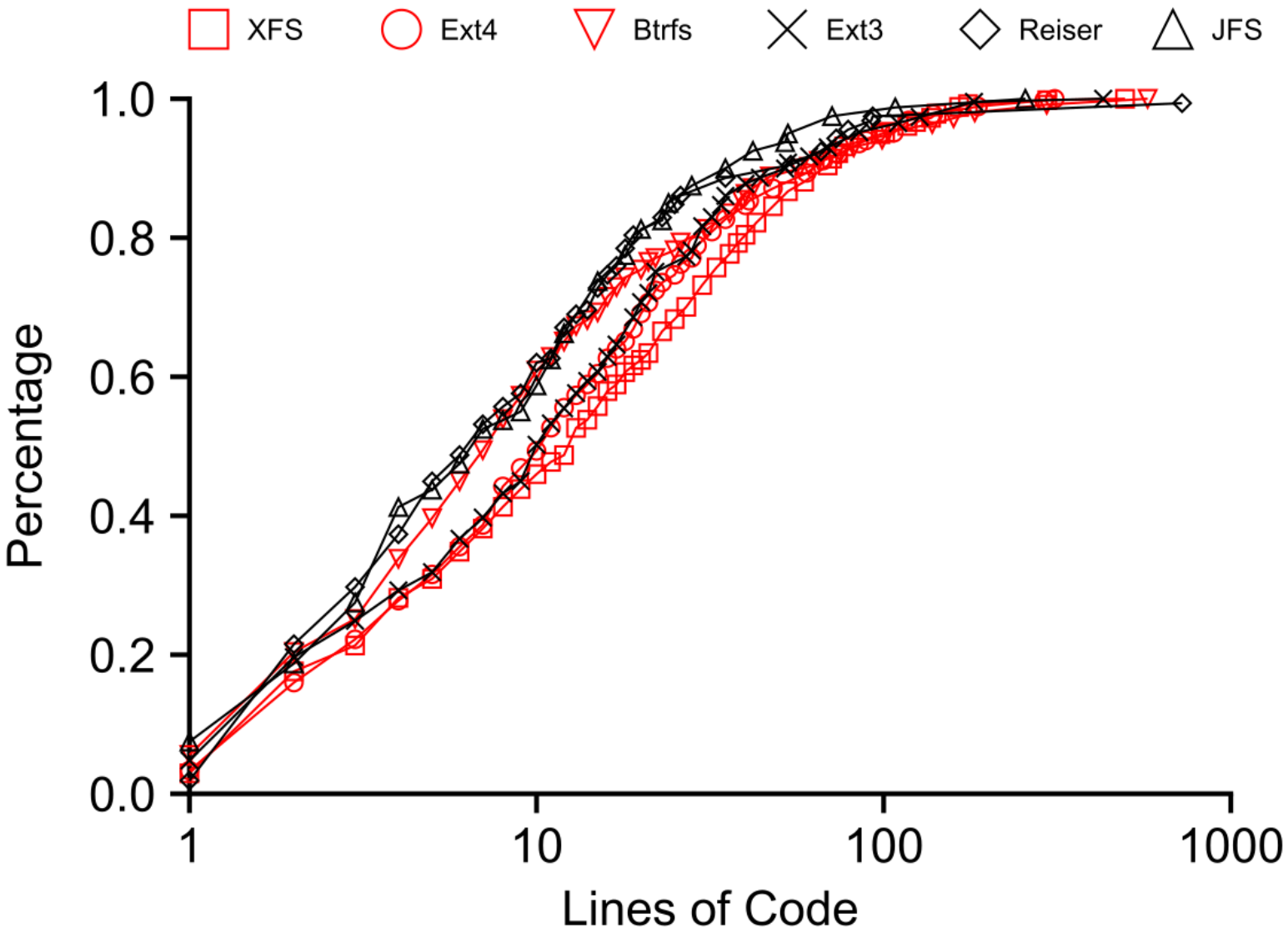
Never make any mistaeks.

Anonymous,
in a mail discussion about to a kernel bug report.

Rozwój kodu


XFS
Ext4
Btrfs
Ext3
Reiser
JFS
Lines of Code
80000
70000
60000
50000
40000
30000
20000
10000
0
0
5
10
15
20
25
30
35
40
Linux Version

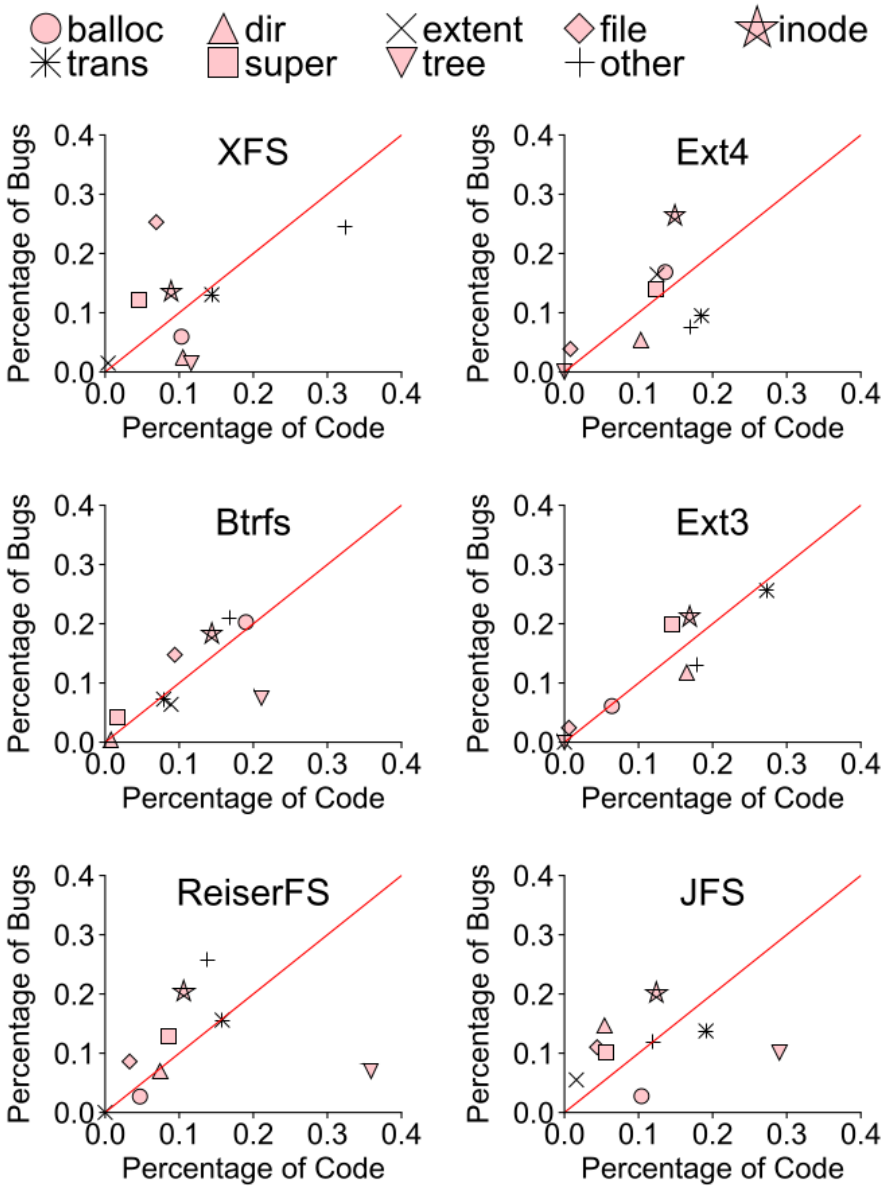
Rozmiar łat Bug



Komponenty

balloc	Alokacja i dealokacja bloków danych
dir	Zarządzanie katalogami
extent	Mapowanie sąsiednich bloków fizycznych
file	Operacje wejścia i wyjścia na plikach
inode	Zarządzanie i-węzłami
trans	Dziennik, wsparcie transakcji
super	Zarządzanie superblokami
tree	Generyczne procedury na strukturach drzewiastych
other	Inne komponenty (np. ioctl, resize)

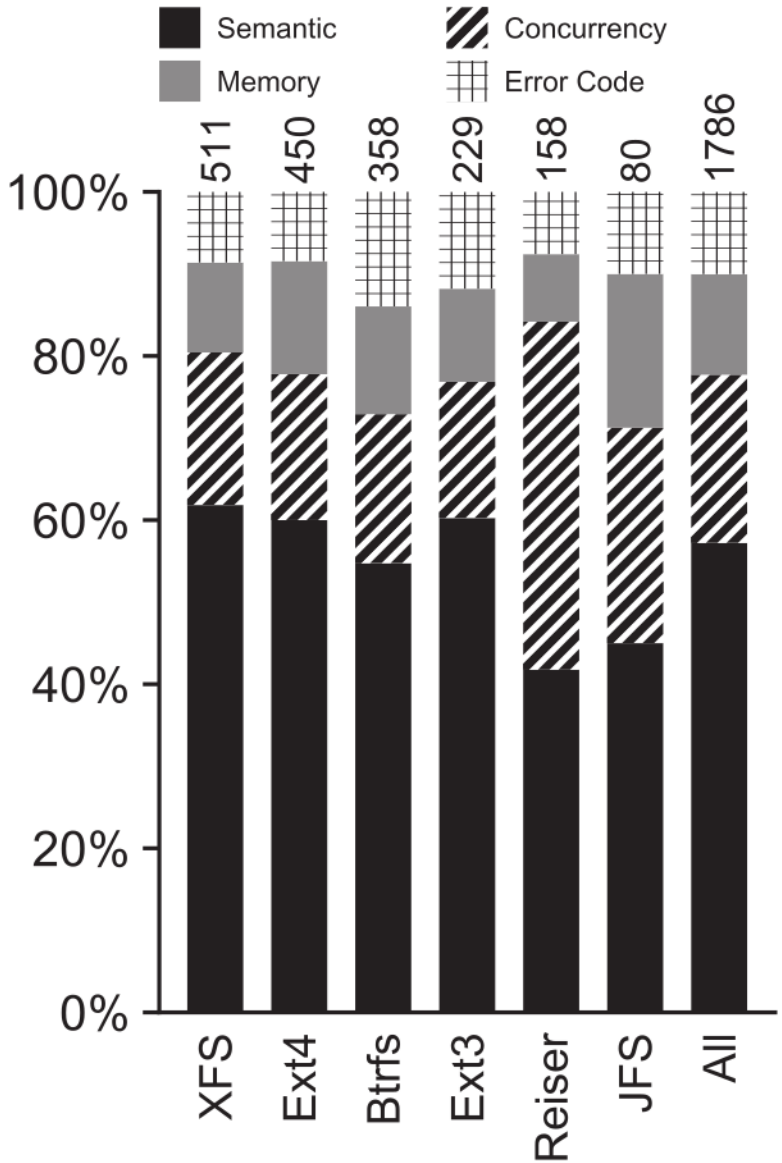
Korelacja kod — błędy



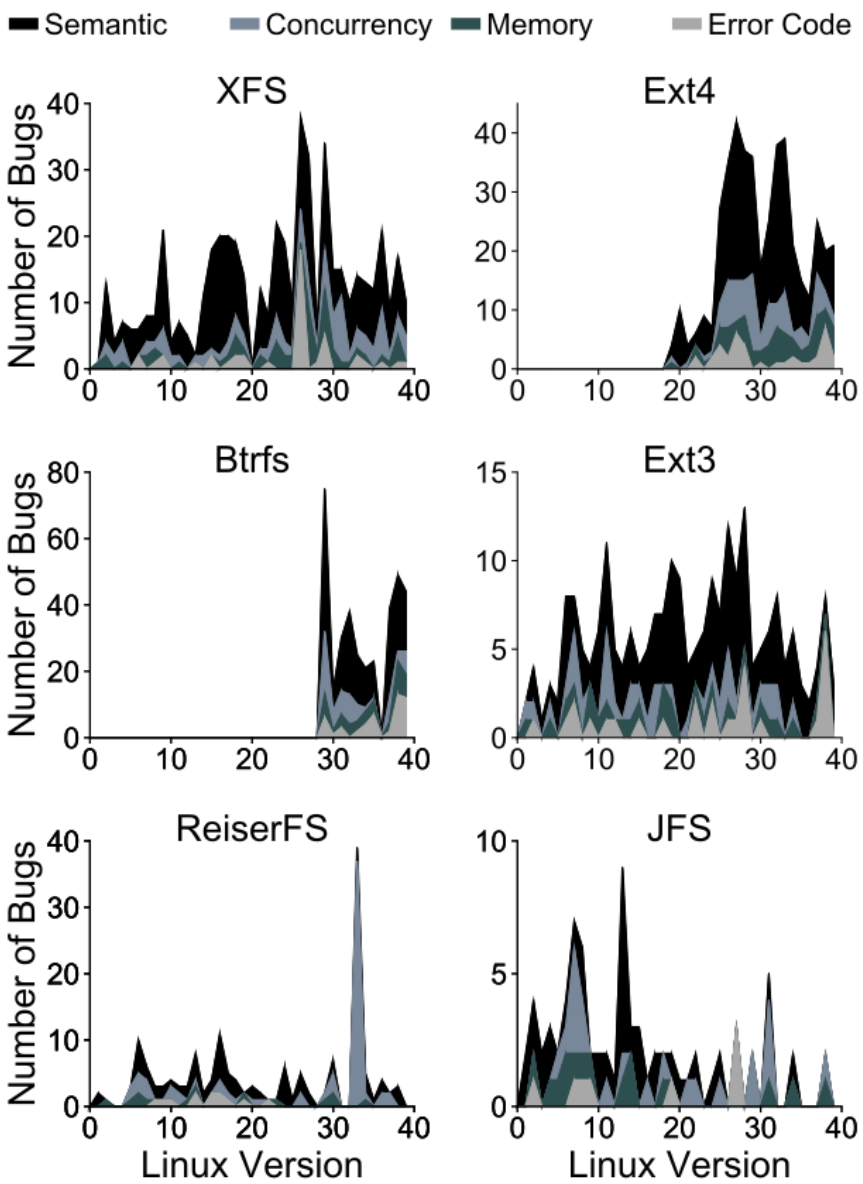
Rodzaje błędów

Semantic (semantyczne)	Błędny algorytm, niewłaściwy typ, zła kolejność żądań I/O
Concurrency (współbieżności)	Naruszenie atomowości, deadlock, podwójne zdjęcie blokady
Memory (pamięci)	Niezwalnianie pamięci, czytanie niezainicjalizowanych zmiennych, przepełnienie bufora
Error Code (kodu błędu)	Niesprawdzanie kodu błędu, zwracanie niewłaściwego kodu

Rodzaje błędów per fs



Tendencje


Semantic
Concurrency
Memory
Error Code
XFS
Ext4
Btrfs
Ext3
ReiserFS
JFS
Number of Bugs
Linux Version

Przykłady Semantic Bug

ext3 / namei.c, 2.6.26

```
1     ext3_rename(...){
2 +     new_dir->i_ctime = CURRENT_TIME_SEC;
3 +     new_dir->i_mtime = CURRENT_TIME_SEC;
4 +     ext3_mark_inode_dirty(handle, new_dir);
```

Przykłady Semantic Bug

ext3 / namei.c, 2.6.26

```
1     ext3_rename(...){
2 +     new_dir->i_ctime = CURRENT_TIME_SEC;
3 +     new_dir->i_mtime = CURRENT_TIME_SEC;
4 +     ext3_mark_inode_dirty(handle, new_dir);
```

ext4 / super.c, 2.6.37

```
1     ext4_load_journal(...){
2 -     if (journal_devnum && ...)
3 +     if (!read_only && journal_devnum ...)
4         es->s_journal_dev = devnum;
```

Przykłady Concurrency Bug

ext4/extents.c, 2.6.30

```
1     ext4_ext_put_in_cache(...){
2 +     spin_lock(i_block_reservation_lock);
3       cex = &EXT4_I(inode)->i_cached_extent;
4       cex->ec_block = block;
5       cex->ec_len = len;
6       cex->ec_start = start;
7 +     spin_unlock(i_block_reservation_lock);
```

Przykłady Concurrency Bug

ext4/extents.c, 2.6.30

```
1     ext4_ext_put_in_cache(...){
2 +     spin_lock(i_block_reservation_lock);
3       cex = &EXT4_I(inode)->i_cached_extent;
4       cex->ec_block = block;
5       cex->ec_len = len;
6       cex->ec_start = start;
7 +     spin_unlock(i_block_reservation_lock);
```

ext3/resize.c, 2.6.17

```
1     lock_super(sb);
2     if (input->group != sbi->s_groups_count){
3       ... ...
4 +     unlock_super(sb);
5       err = -EBUSY;
6       goto exit_journal;
```

Przykłady Memory Bug

btrfs/volumes.c, 2.6.34

```
1     run_scheduled_bios(...){
2 -     submit_bio(cur->bi_rw, cur);
3       if (bio_rw_flagged(cur, BIO_RW_SYNCIO))
4         num_sync_run++;
5 +     submit_bio(cur->bi_rw, cur);
```

Przykłady Memory Bug

btrfs/volumes.c, 2.6.34

```
1     run_scheduled_bios(...){
2 -     submit_bio(cur->bi_rw, cur);
3       if (bio_rw_flagged(cur, BIO_RW_SYNCIO))
4         num_sync_run++;
5 +     submit_bio(cur->bi_rw, cur);
```

ext4/xattr.c, 2.6.33

```
1     ext4_expand_extra_isize_ea(...){
2       while (...){
3         if (error)
4           goto cleanup;
5         kfree(b_entry_name);
6 +       b_entry_name = NULL;
7       }
8     cleanup:
9       kfree(b_entry_name);
```

Przykłady Error Code Bug

reiserfs/xattr_acl.c, 2.6.16

```
1     reiserfs_get_acl(...){
2       acl = posix_acl_from_disk(...);
3 -     *p_acl = posix_acl_dup(acl);
4 +     if (!IS_ERR(acl))
5 +       *p_acl = posix_acl_dup(acl);
```

Przykłady Error Code Bug

reiserfs/xattr_acl.c, 2.6.16

```
1     reiserfs_get_acl(...){
2       acl = posix_acl_from_disk(...);
3 -     *p_acl = posix_acl_dup(acl);
4 +     if (!IS_ERR(acl))
5 +       *p_acl = posix_acl_dup(acl);
```

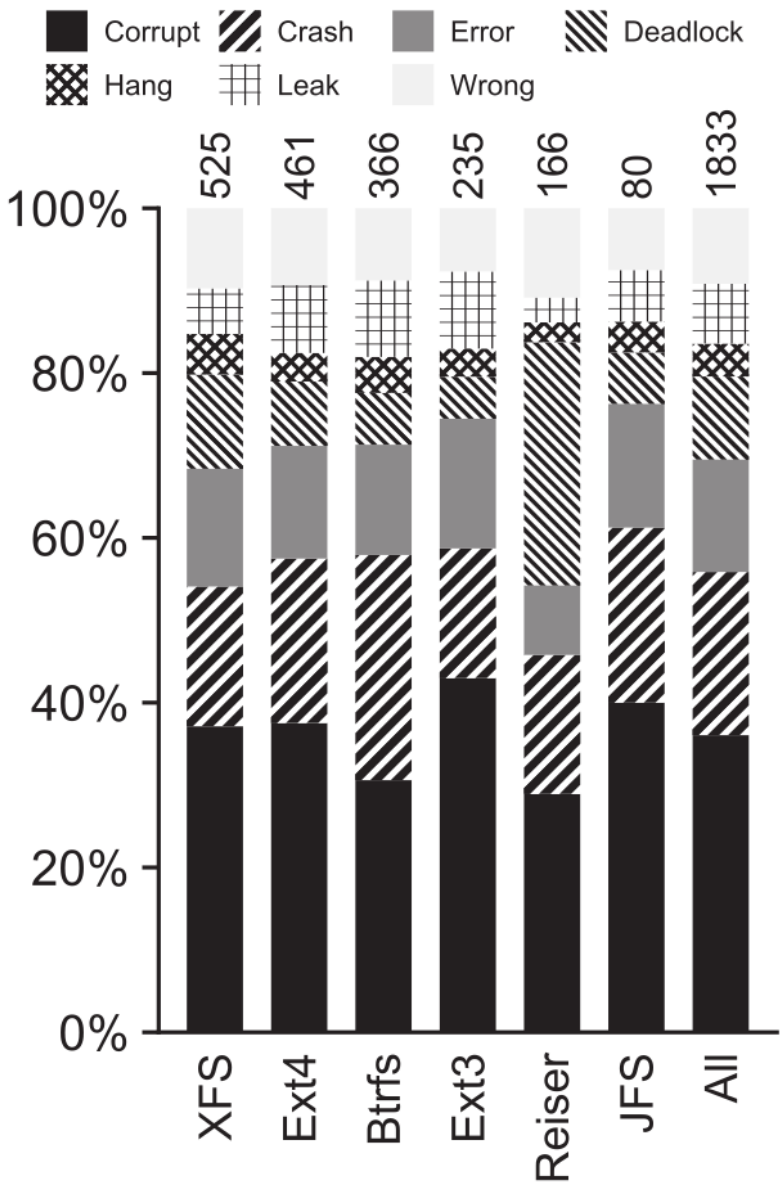
jfs/jfs_imap.c, 2.6.27

```
1     diAlloc(...){
2       jfs_error(...);
3 -     return EIO;
4 +     return -EIO;
```

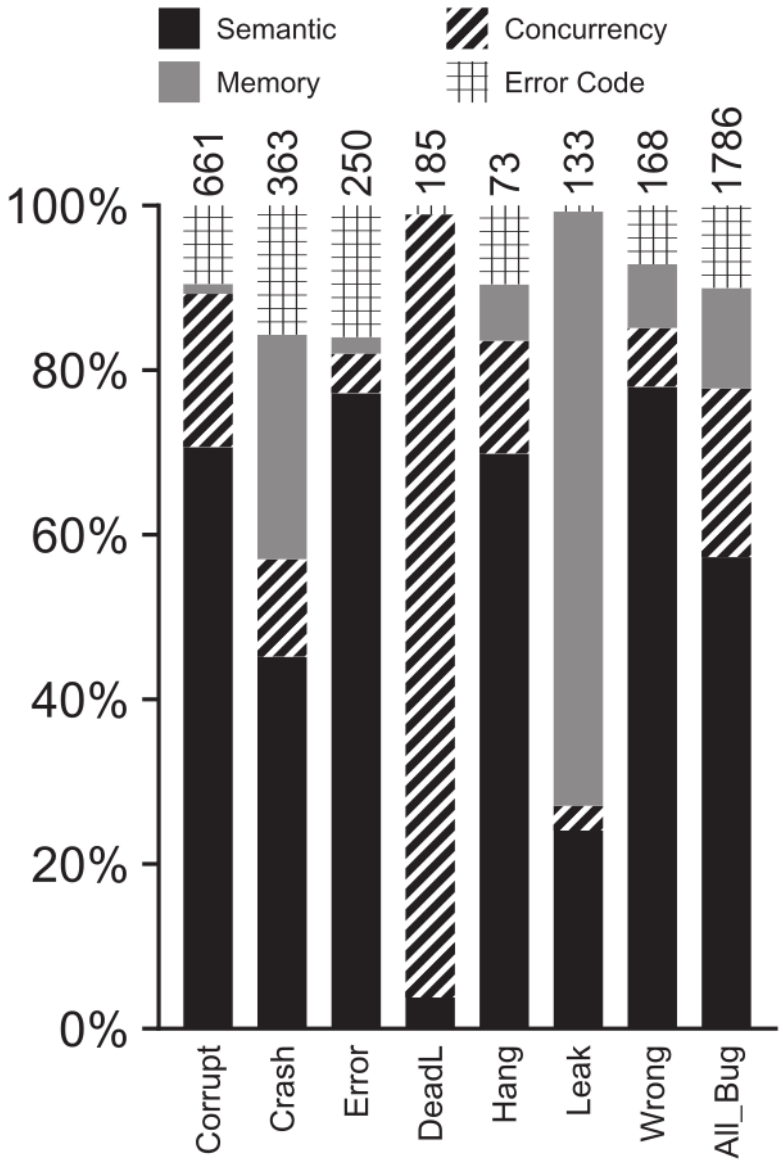
Konsekwencje błędów

Corruption (uszkodzenie danych)	Uszkodzenia danych w pliku/pamięci, błędne metadane
Crash (awaria SO)	Niespełniona asercja, kernel panic
Error (błąd)	Niepowodzenie operacji, nieoczekiwany kod błędu
Deadlock (zakleszczenie)	Cykl oczekujących na zasoby
Hang (zawieszenie się SO)	Nieskończone pętle, livelock
Leak (wyciek zasobów)	Niezwalnianie zasobów
Wrong (niepoprawne zachowanie)	Niepożądany wynik, luka w bezpieczeństwie

Konsekwencje błędów per fs



Konsekwencje błędów per rodzaj



Błędy obsługi błędów per fs

XFS	Ext4	Btrfs	Ext3	ReiserFS	JFS
200 (39.1%)	149 (33.1%)	144 (40.2%)	88 (38.4%)	63 (39.9%)	28 (35%)

Błędy obsługi błędów per rodzaj

Semantic	Concurrency	Memory	Error Code
283 (27.7%)	93 (25.4%)	117 (53.4%)	179 (100%)

Przykłady Błędów obsługi błędów

ext4/resize.c, 2.6.25

```
1     ext4_group_extend(...) {
2       ext4_warning(sb, "multiple resizers run on filesystem!");
3       unlock_super(sb);
4 +     ext4_journal_stop(handle);
5       err = -EBUSY;
6       goto exit_put;
```

Przykłady Błędów obsługi błędów

ext4 / resize.c, 2.6.25

```
1     ext4_group_extend(...) {
2       ext4_warning(sb, "multiple resizers run on filesystem!");
3       unlock_super(sb);
4 +     ext4_journal_stop(handle);
5       err = -EBUSY;
6       goto exit_put;
```

ext4 / mballoc.c, 2.6.27

```
1     mb_free_blocks(...) {
2 +     ext4_unlock_group(sb, e4b->bd_group);
3       ext4_error(sb, ... "double-free of inode");
4 +     ext4_lock_group(sb, e4b->bd_group);
```

Łaty *Performance*

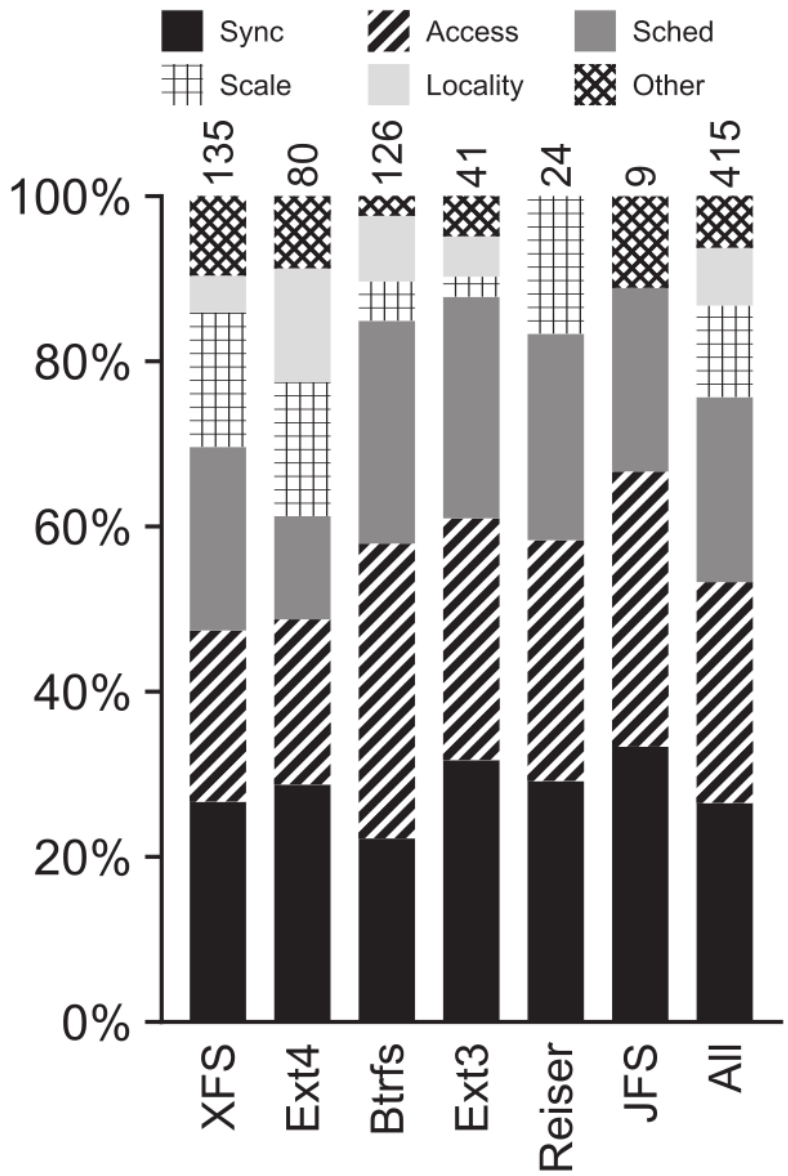
We all know Linux is great... it does infinite loops in 5 seconds.

Linus Torvalds

Rodzaje

Synchronization (synchronizacja)	Usuwanie niepotrzebnych blokad, użycie blokad czytelnicy-pisarze
Access Optimization (optymalizacja dostępu)	Cachowanie metadanych, unikanie niepotrzebnych I/O
Schedule (planowanie)	Grupowanie zapisów, czytanie "na przyszłość"
Scalability (skalowalność)	Użycie drzew/słowników, zmniejszenie wielkości i-węzłów
Locality (rozmieszczenie danych)	Zmniejszanie fragmentacji
Other (inne)	Redukcja wielkości potrzebnego stosu

Rodzaje per fs



Przykłady łat Performance

btrfs/free-space-cache.c, 2.6.39

```
1     btrfs_find_space_cluster(...){
2 +     if (bg->free_space < min_bytes){
3 +       spin_unlock(&bg->tree_lock);
4 +       return -ENOSPC;
5 +     }
6         /* start to search for blocks */
```

Przykłady łat Performance

btrfs/free-space-cache.c, 2.6.39

```
1     btrfs_find_space_cluster(...){
2 +     if (bg->free_space < min_bytes){
3 +       spin_unlock(&bg->tree_lock);
4 +       return -ENOSPC;
5 +     }
6         /* start to search for blocks */
```

ext3/xattr.c, 2.6.21

```
1     ext3_xattr_release_block(...){
2       error = ext3_journal_dirty_metadata(handle, bh);
3 -     handle->h_sync = 1;
4 +     if (IS_SYNC(inode))
5 +       handle->h_sync = 1;
```

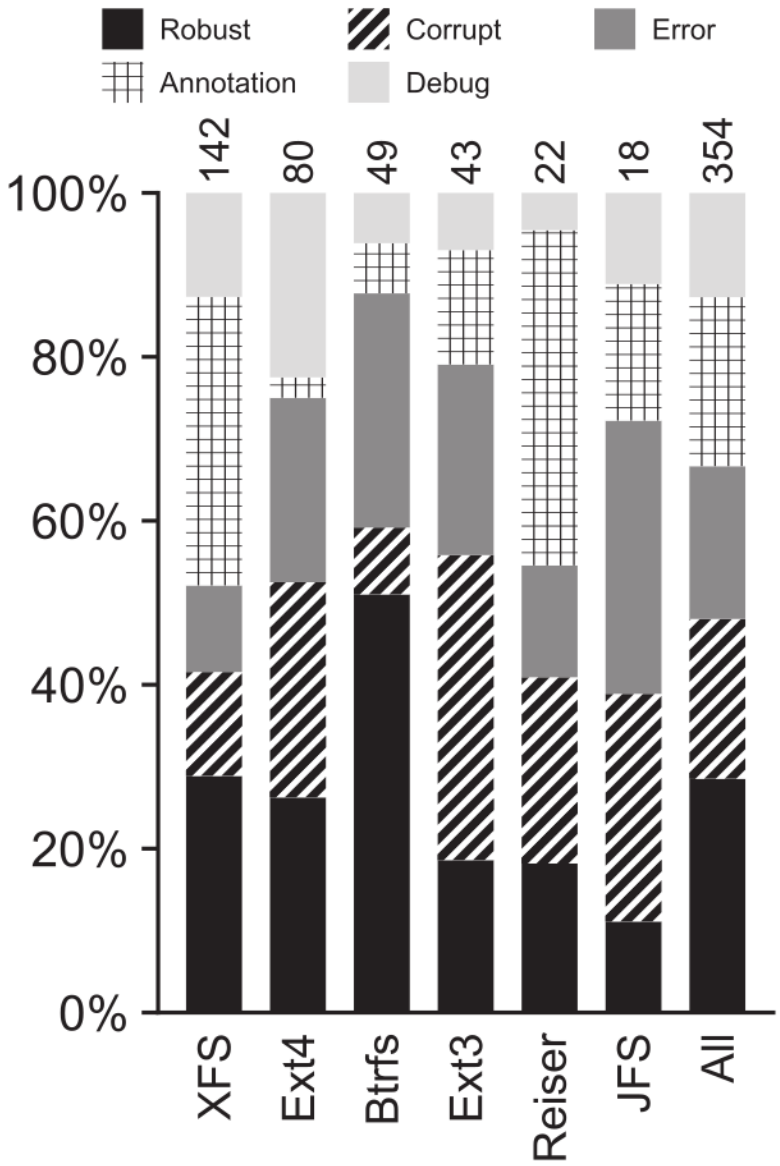
Łaty *Reliability*

> In a few minutes a computer can make a mistake so great that it would have taken many men many months to equal it.
>
> Anonymous

Rodzaje

Robust (odporność)	Sprawdzanie zakresów i uprawnień, asercje
Corruption Defense (obrona przed uszkodzeniami)	Poprawienie obsługi możliwych uszkodzeń
Error Enhancement (ulepszanie błędnych sytuacji)	Odpowiednia obsługa błędnych sytuacji, lepsze kody błędów
Annotation (komentarze)	Opisanie blokad, grubo/cienko-końcówkowości,
Debug	Dodanie wewnętrznego debug'u

Rodzaje per fs



Przykłady łat Reliability

jfs/ioctl.c, 2.6.24

```
1     jfs_ioctl(...) {
2 +     /* Is it quota file? Do not allow user to mess with it */
3 +     if (IS_NOQUOTA(inode))
4 +       return -EPERM;
5       fs_get_inode_flags(jfs_inode);
```

Przykłady łat Reliability

jfs/ioctl.c, 2.6.24

```
1     jfs_ioctl(...) {
2 +     /* Is it quota file? Do not allow user to mess with it */
3 +     if (IS_NOQUOTA(inode))
4 +       return -EPERM;
5       fs_get_inode_flags(jfs_inode);
```

btrfs/file.c, 2.6.38

```
1     prepare_pages(...) {
2       pages[i] = grab_cache_page(...);
3       if (!pages[i]) {
4 -       BUG_ON(1);
5 +       for (c = i - 1; c >= 0; c--) {
6 +         unlock_page(pages[c]);
7 +         page_cache_release(pages[c]);
8 +       }
9 +       return -ENOMEM;
10      }
```

Najczęstsze

Najczęstsze

Patch Type	Typical Cases	XFS	Ext4	Btrfs	Ext3	Reiser	JFS
Semantic	forget sync	17	11	6	11	5	1
	forget config	43	43	23	16	8	1
	early enospc	5	9	14	7		
	wrong log credit	6	8	1	1	1	
Concurrency	lock inode update	6	5	2	4	4	2
	lock sleep	8	8	1	1	8	
	wrong kmalloc flag	20	3	3	2		1
	miss unlock	10	7	4	2	2	4
Memory	leak on failure	14	21	16	11	1	3
	leak on exit	1		1	4		1
Error Code	miss I/O error	10	11	8	15	4	1
	miss mem error	4	2	13	1	1	
	bad error access		3	8			2
Performance	remove lock	17	14	14	8	5	1
	avoid redun write	6	4	5	4		2
	check before work	8	5	15	2	1	
	save struct mem	3	9		1	3	
Reliability	metadata validation	12	9	1	7	2	1
	graceful handle	8	6	5	5	1	4

Pytania?