Understanding File System Performance Analytics in Response to Simulated Aging and Metadata Corruption

Chase S. Gilbert

New Mexico State University

July 9, 2018

Presentation Summary

- Introduction
- Related Work
- Design and Implementation
- Evaluation
- Future Work
- Conclusion

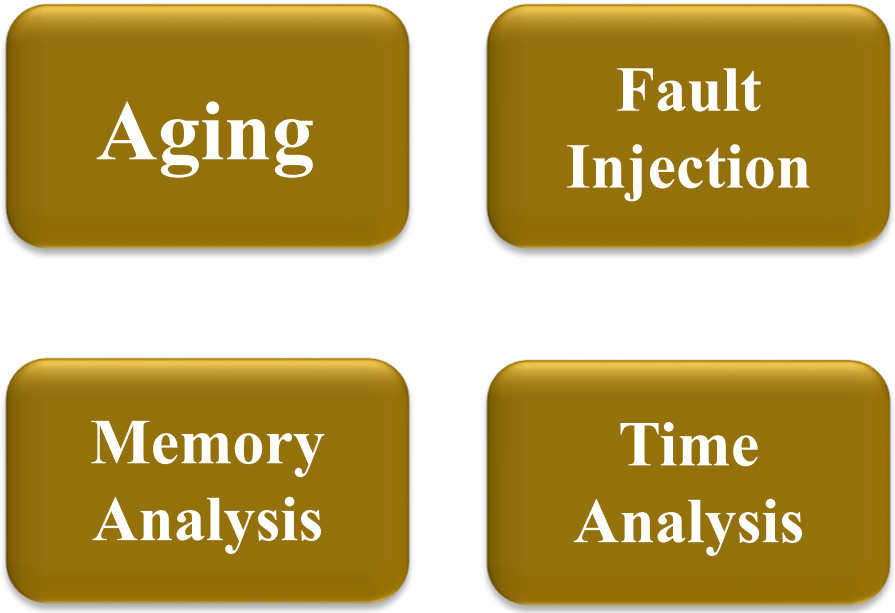
Introduction

- File system checker runtime is a problem. Why?
  - Halts workflow
  - Increases crash possibility
- What can we do about this?

- Project consists of looking at two popular file systems (EXT4 and XFS) and their respective checkers, to gain insight about what metadata corruption types have the largest contribution to checker time and memory usage.

Introduction

- Analysis consists of four parts:
  - **Aging**
    - Two tools made.
    - Compared against a pre-existing method.
  - **Fault Injection**
    - Different metadata layouts per system.
    - One tool created with different approaches for each system.
  - **Memory Analysis**
    - Memory peak monitored with Massif.
  - **Time Analysis**
    - Runtime monitored with the Linux *time* command.

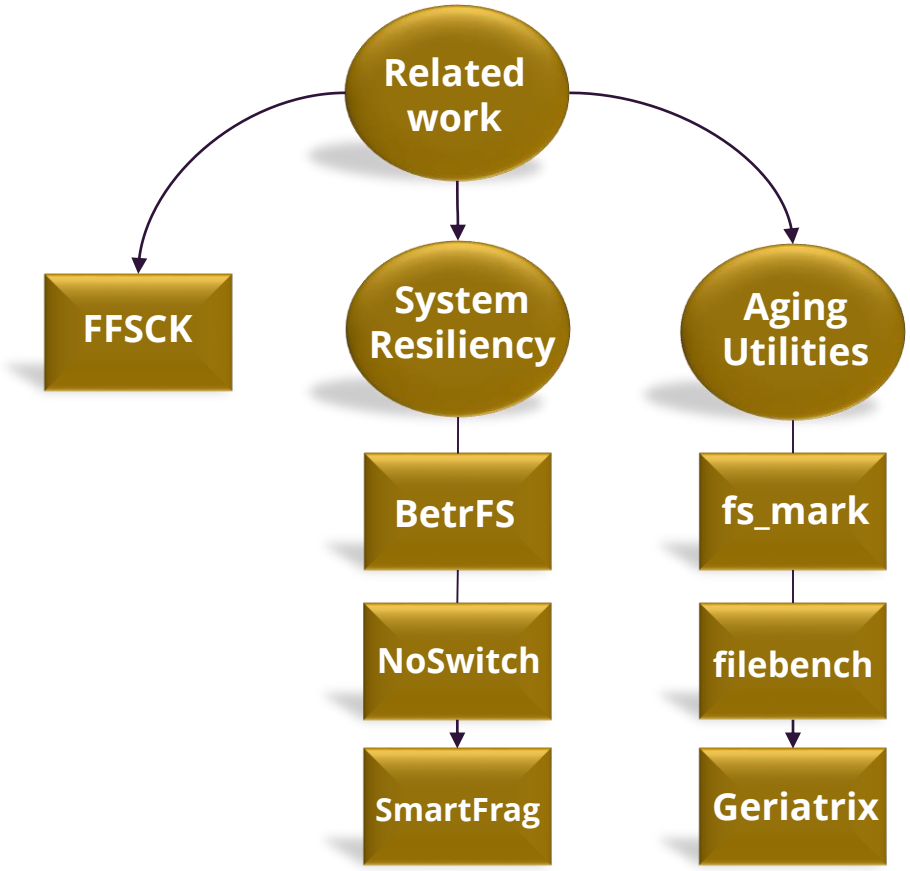


Related Work

- **The Fast File System Checker**

- **System Resiliency**
  - BetrFS
  - NoSwitch
  - SmartFrag

- **Aging Utilities**
  - fs_mark
  - filebench
  - Geriatrix



Design and Implementation

- **Goals:**
  I. **Observable and competitive file system aging.**
  II. **Effective and high-level corruption.**
  III. **Data pool completeness.**
  IV. **Clear results of the effects of corruption.**

Design and Implementation

- **Aging Tools:**
  - roundHouse
  - holePuncher

- Compared against the “git pulls” method from BetrFS.

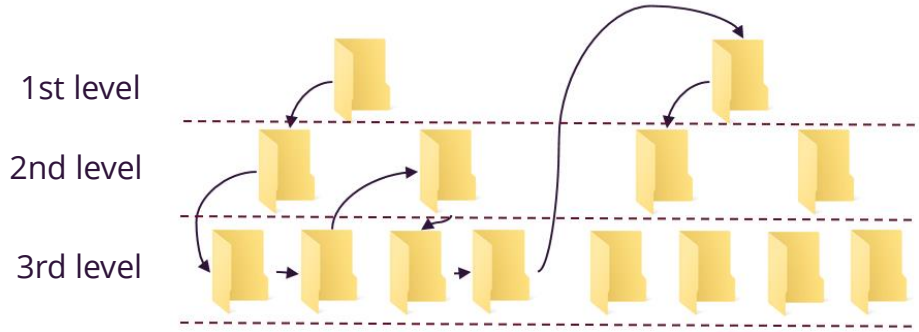
roundHouse.py

holePuncher.py

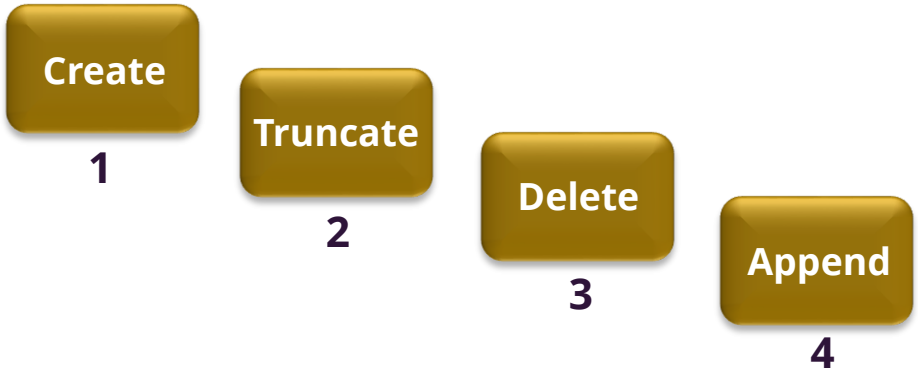
Design and Implementation

- **Aging Tools:**
  - roundHouse

- roundHouse first makes three main folder levels, to create a dense hierarchy.
- It then calls a function called *roundRobin* once per folder, until the file system is full.
- After each iteration over the folder set, a counter is either incremented or reset, to offset the file operation type per folder.



roundRobin operations



Design and Implementation

- **Aging Tools:**
  - roundHouse

- roundRobin has four cases, one for each file operation: Create, Truncate, Delete, and Append.
- After each operation is complete, a counter is incremented or reset to point to the next case in the sequence.

roundRobin operations

Create

Truncate

Delete

Append

Design and Implementation

- **Aging Tools:**
  - holePuncher

- Uses hole punching as an aging method.
- What is hole punching?

6f73 696d 766f 7369

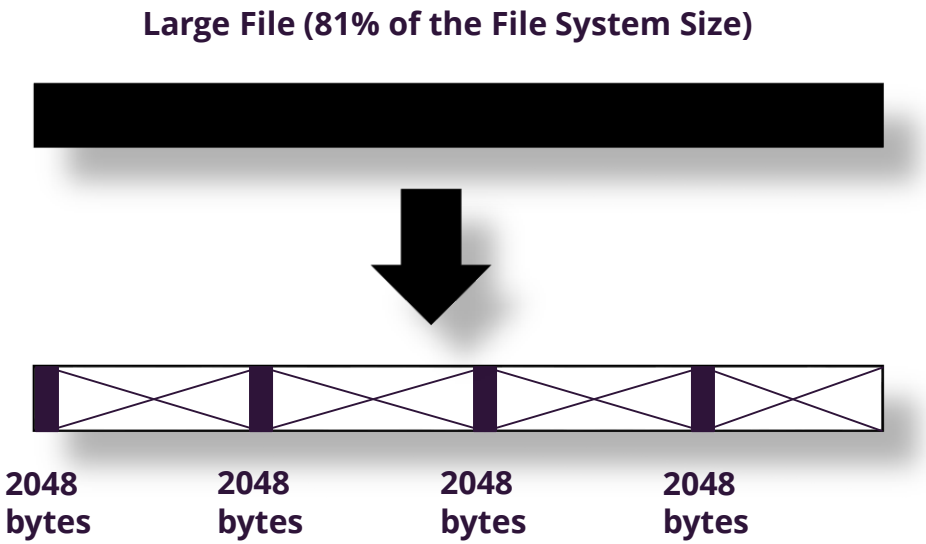
```
HolePunch --offset 1 --length 4
```

6f00 0000 006f 7369

Design and Implementation

- **Aging Tools:**
  - holePuncher

- holePuncher first creates a very large file, then punches four large holes in it.
- 2048 bytes of user data are left before each hole, so that the data is spread out over the file system, but the file only takes up 8192 bytes.
- This process repeated one million times, or until the file system is full.



Design and Implementation

- **Aging Tools:**
  - Git pulls

- Performs a user-specified number of git pulls from a given Github repository to a local repository located in the desired file system.
- Aging occurs because each pull slightly alters the system contents, leading to fragmentation.

Design and Implementation

- **Corruption Tool:**
  - corruptFS

- EXT4 and XFS have different metadata layouts. How do we handle this?

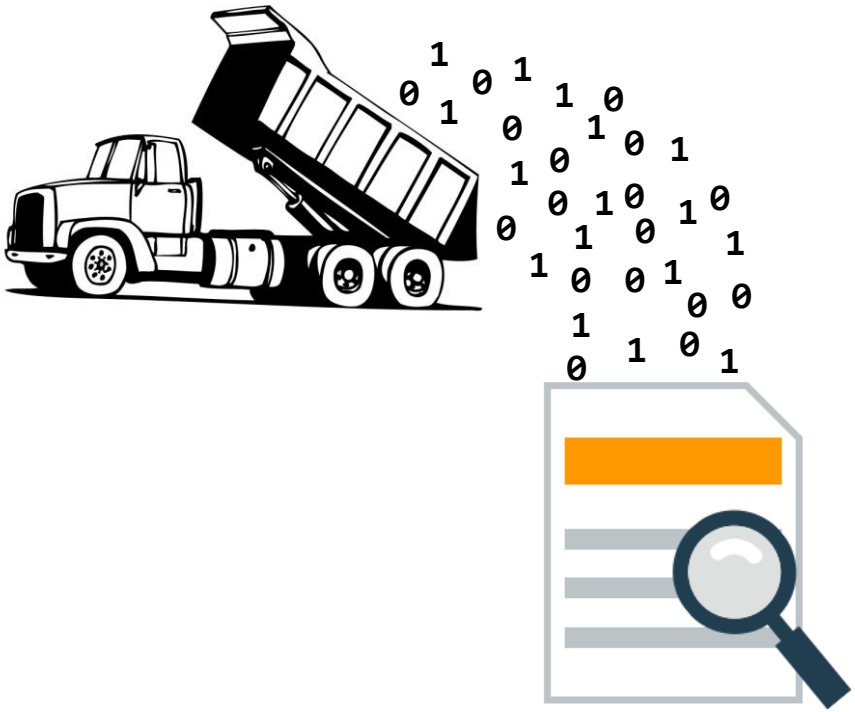
- First, the tool decides via user input which function to call:
  - ext4Attack() for EXT4
  - xfsAttack() for XFS

corruptFS.c

Design and Implementation

- Corruption Tool:
  - **corruptFS**
    - **ext4Attack**

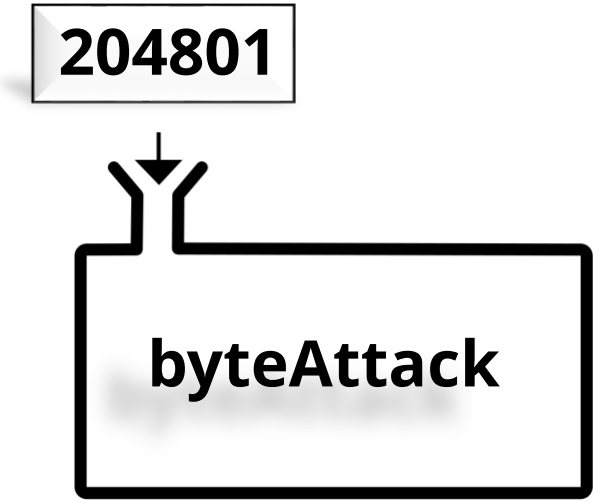
- First, vital system information from the ext4 system is dumped to a file.
- The total number of block groups is calculated from the file contents, and then displayed.
- The desired block group is then chosen by the user.



Design and Implementation

- Corruption Tool:
  - **corruptFS**
    - **ext4Attack**

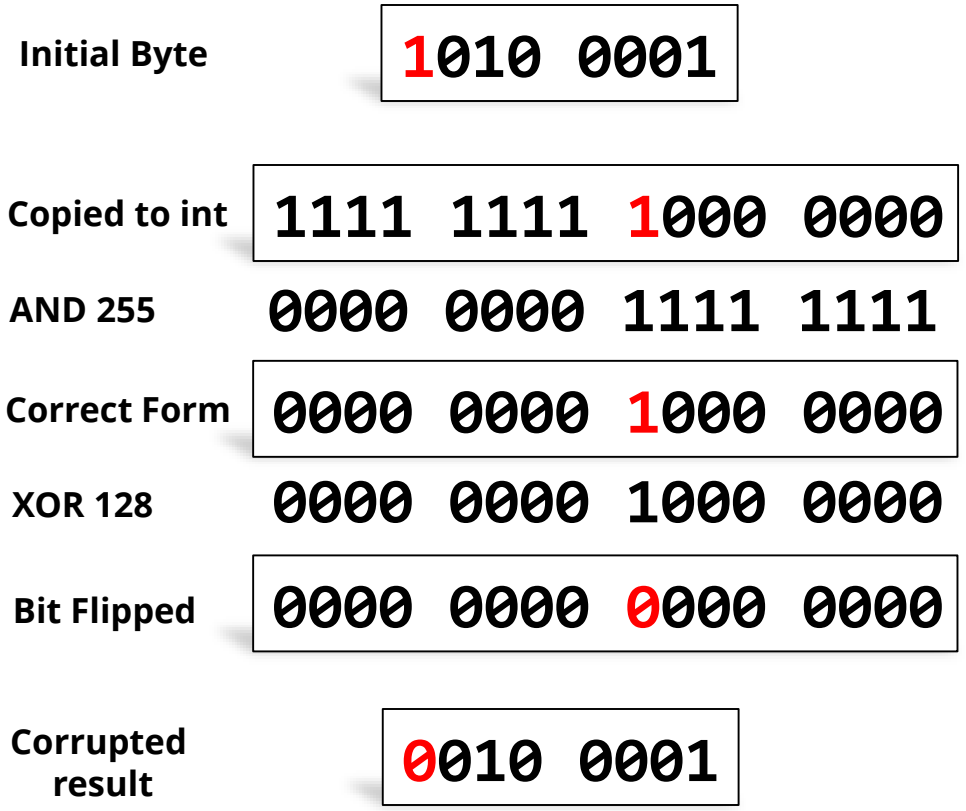
- Next, the tool iterates through the dump file to find the entry for the selected block group.
- The associated metadata components are displayed, and their locations are stored.
- One metadata item is then chosen by the user for corruption. The location is retrieved via a switch statement, and passed to a new function, called *byteAttack.*



Design and Implementation

- Corruption Tool:
  - **corruptFS**
    - **byteAttack**

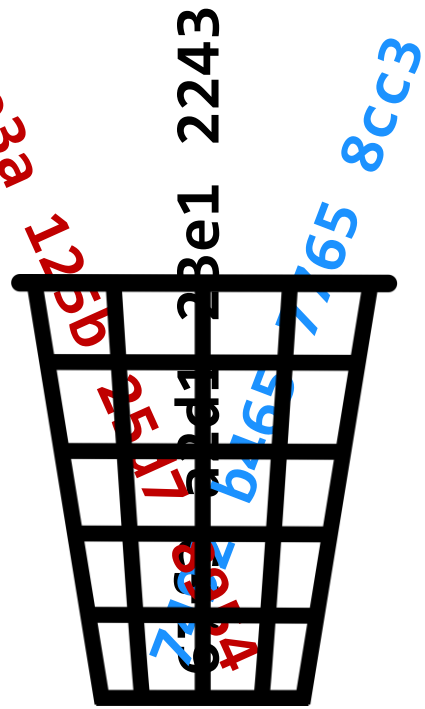
- First, the function seeks to the byte location provided by ext4Attack. It then copies the sought byte into a char buffer.
- Then, the first bit of the buffer is copied into an int, which is then bitmasked to zero out the leading bits.
- Finally, the int is XORed with 128 to invert the selected bit, copied back into the buffer, and the corrupted buffer is written back to the file system.



Design and Implementation

- Corruption Tool:
  - **corruptFS**
    - **xfsAttack**

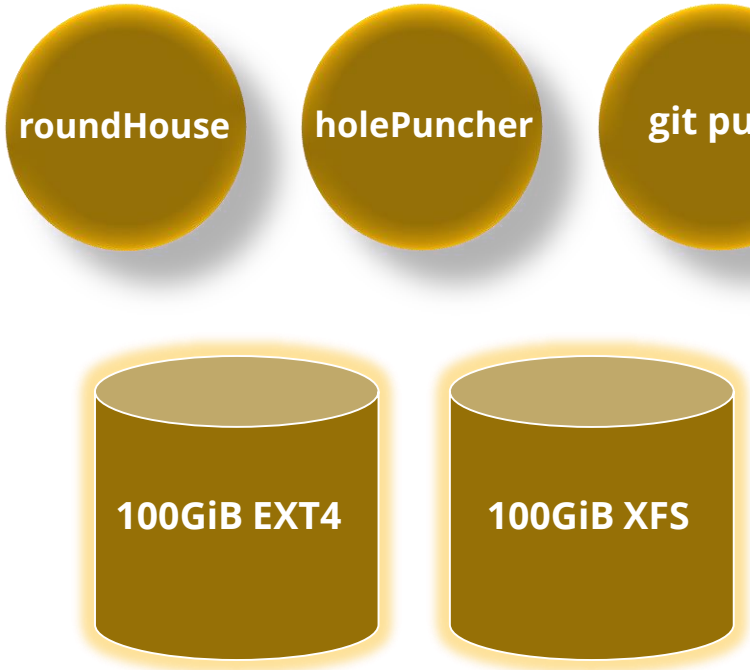
- No clear way to get metadata byte locations.
- corruptFS makes use of an external tool: xfs_db.
- Takes user input to select the metadata component to corrupt, and then calls the *blocktrash* option of xfs_db to "trash" random blocks of the desired type.



Evaluation

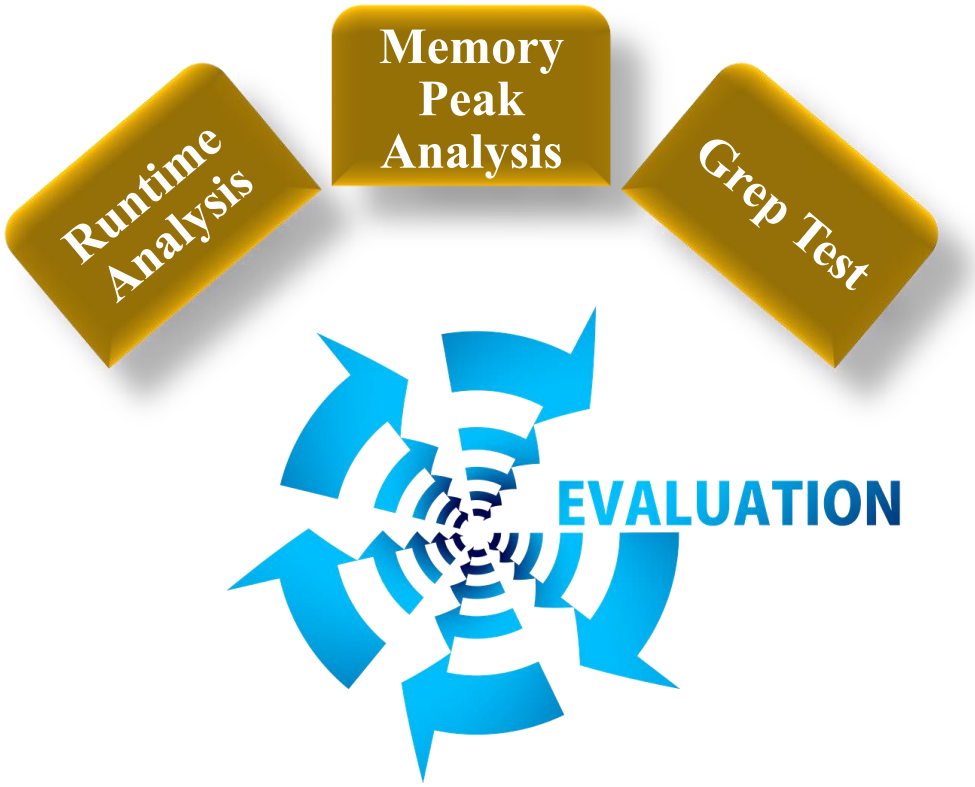
- All tests were done using a 3.3GHz Intel i5 CPU, and a 2TB Dell HDD with Linux 4.40-97, e2fsprogs 1.44.1 and xfs_progs 4.15.1.

- Initial aging analysis:
- Three aging methods compared:
  - roundHouse
  - holePuncher
  - git pulls
- Tested on 100G EXT4 and XFS systems with default configurations.
- Each tool ran for 12 hours.



Evaluation

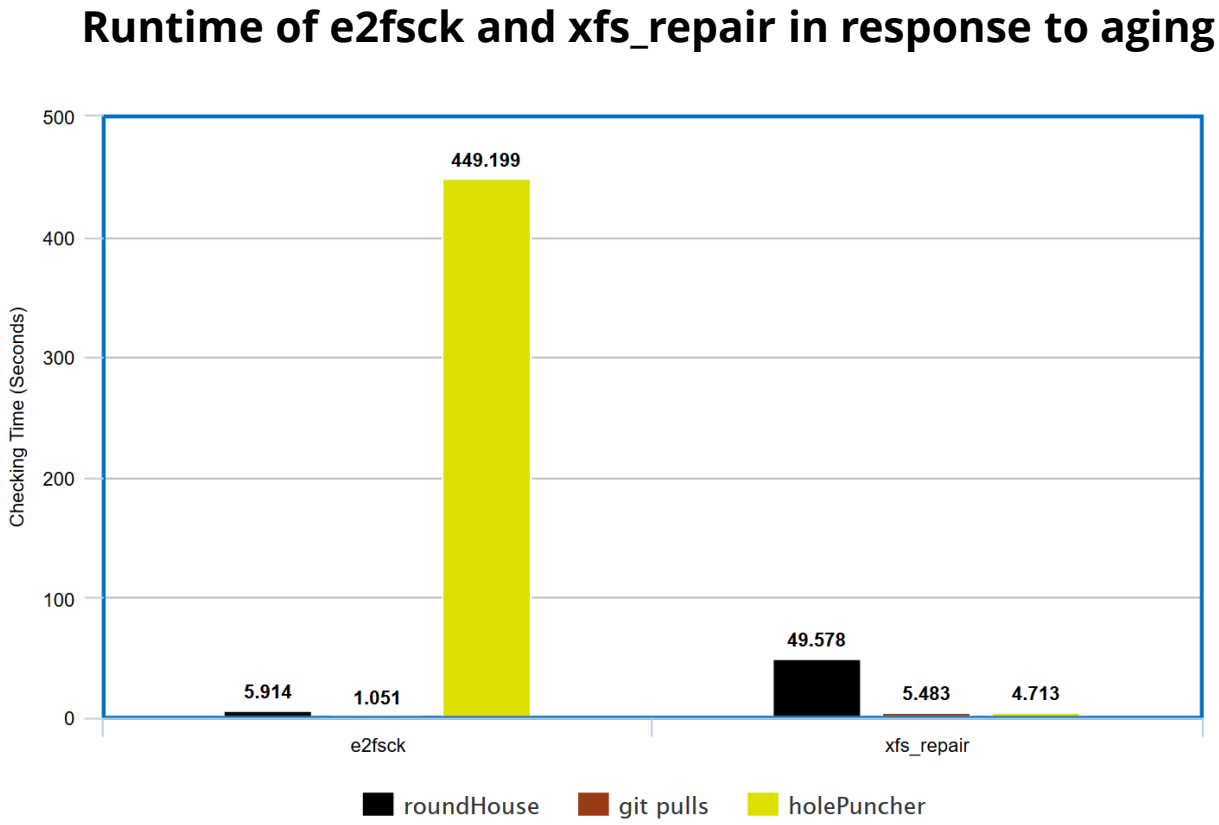
- **Three evaluation methods:**
  - **Runtime Analysis:** Linux “time” command paired with the associated checker call.
  - **Memory Peak Analysis:** Valgrind “Massif” tool to isolate memory usage peak.
  - **Grep Test:** Linux “time” command paired with a recursive grep call.



Evaluation

➢ Runtime Analysis

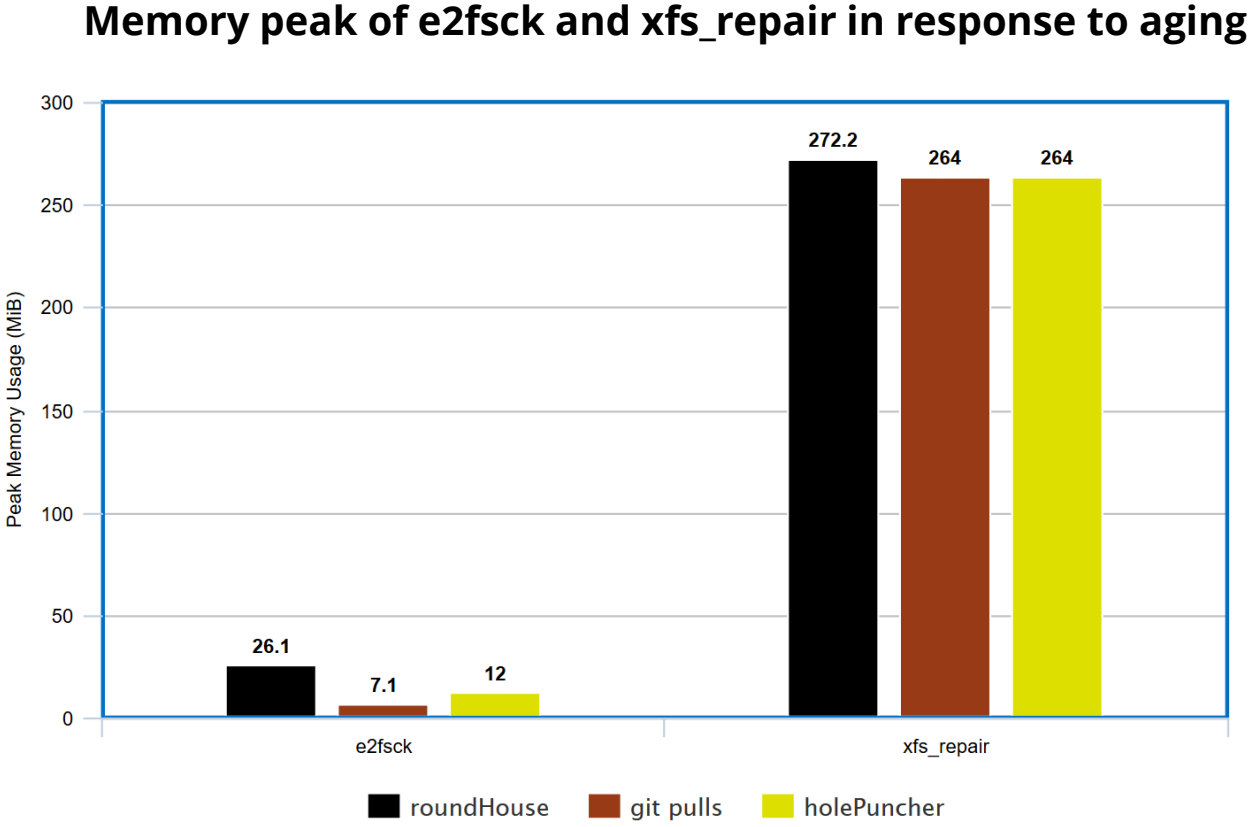
- **E2FSCK**
  - Most affected by holePuncher.
- **XFS_REPAIR**
  - Most affected by roundHouse.



Evaluation

- **Memory peak Analysis**

- **E2FSCK**
  - Most affected by roundHouse.
- **XFS_REPAIR**
  - Most affected by roundHouse.

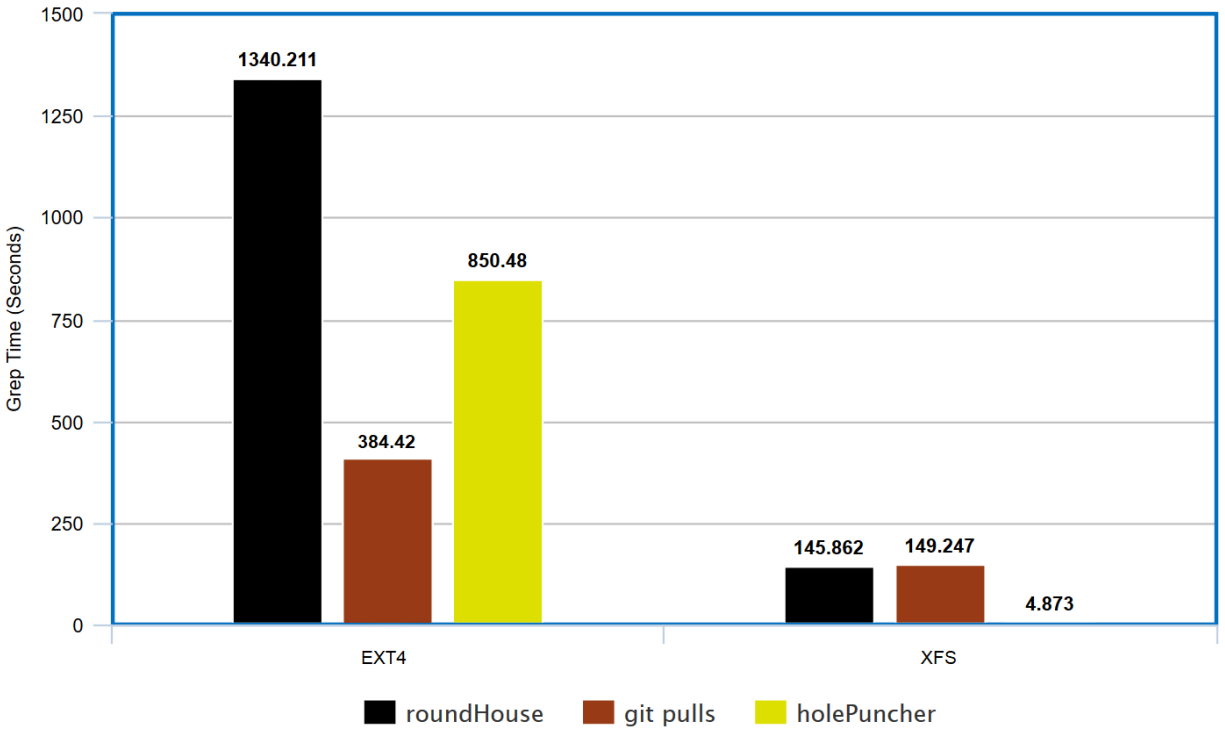


Evaluation

➢ Grep Test

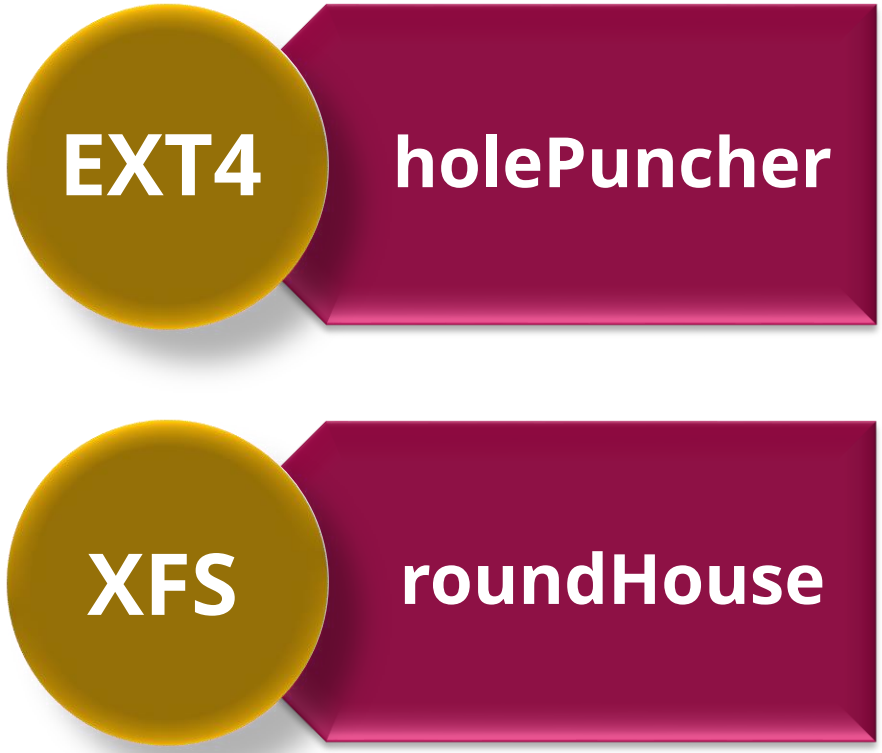
- **E2FSCK**
  - Most affected by roundHouse.
- **XFS_REPAIR**
  - Similarly affected by roundHouse and git pulls.

Grep test time of e2fsck and xfs_repair in response to aging



Evaluation

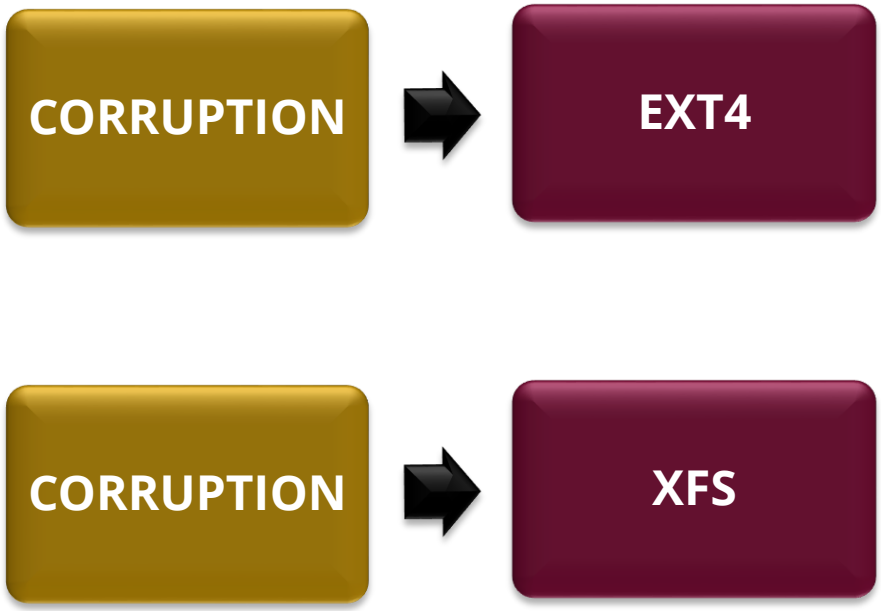
- Based on the results, *holePuncher* is chosen as the tool to age EXT4 systems, and *roundHouse* is chosen as the tool to age XFS systems



Evaluation

- **Checker response to corruption.**

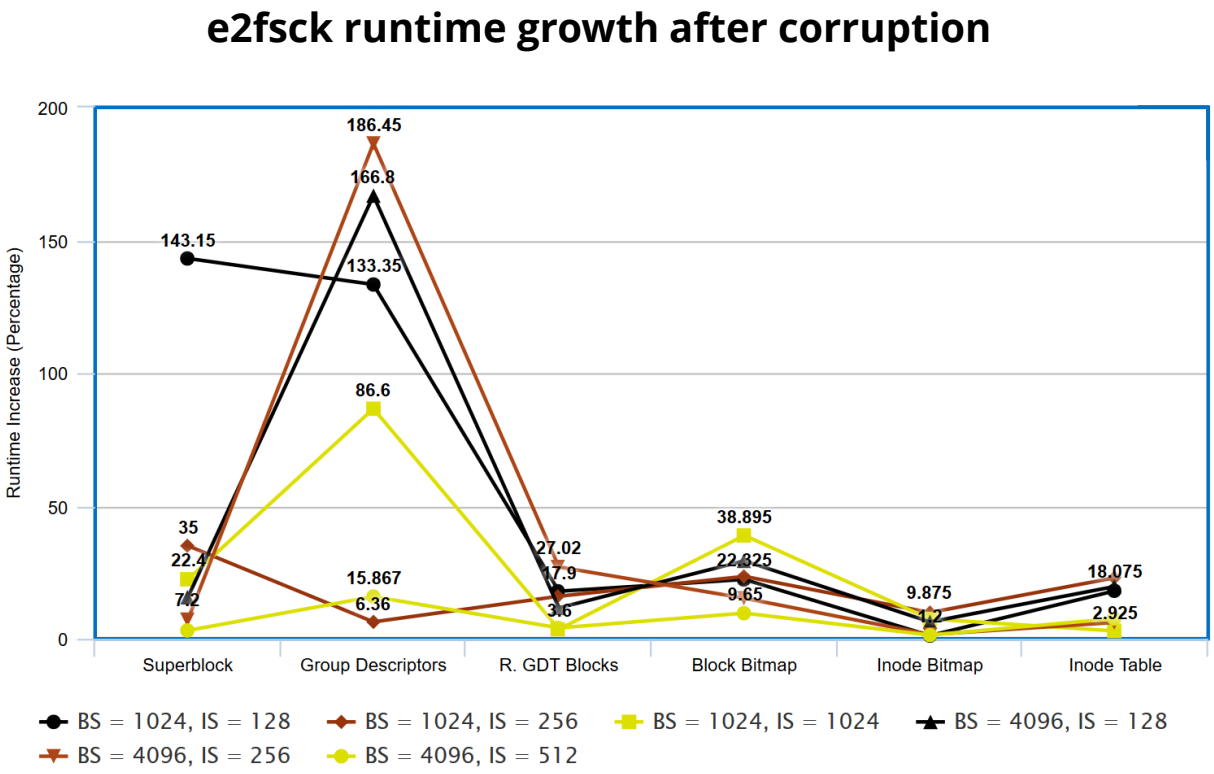
- Tests were conducted on file system sizes of 100GiB, 300GiB, 500GiB, and 800GiB, block sizes of 1024 and 4096, and inode sizes of 128, 256, 512 and 1024.
- Each configuration was aged for four hours with the appropriate tool.
- The aged systems were corrupted, and the checker time and memory usage were compared before and after corruption.



Evaluation

- **Runtime Comparison (EXT4)**

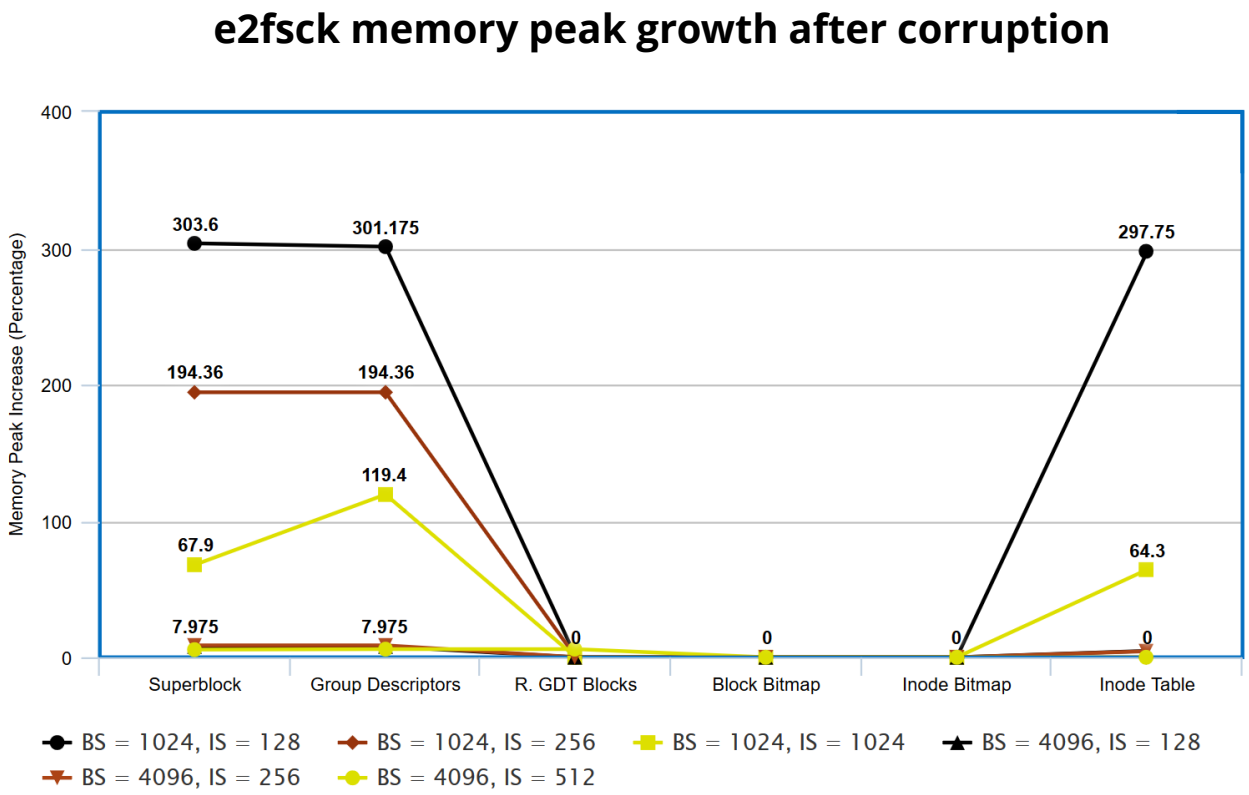
- Greatest slowdown caused by corruption of the group descriptors.



Evaluation

- **Memory Peak Comparison (EXT4)**

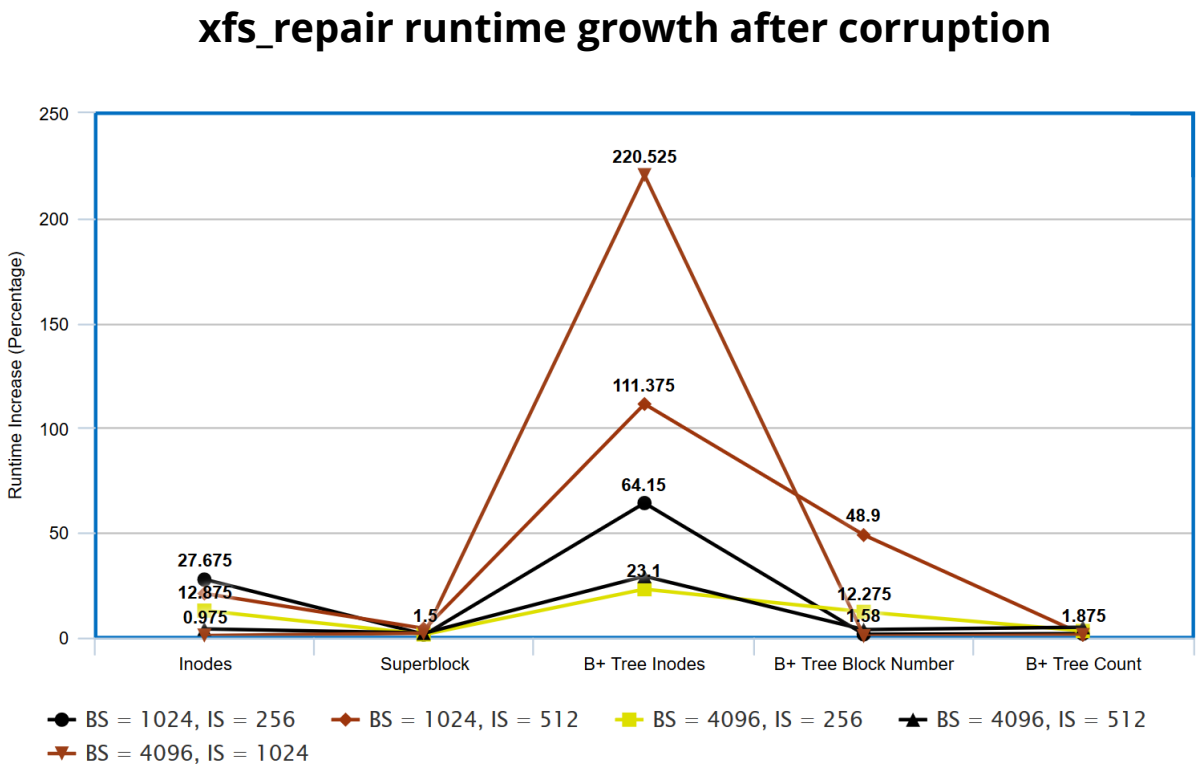
- Greatest memory increase caused by corruption of the group descriptors and the superblock.



Evaluation

- **Runtime Comparison (XFS)**

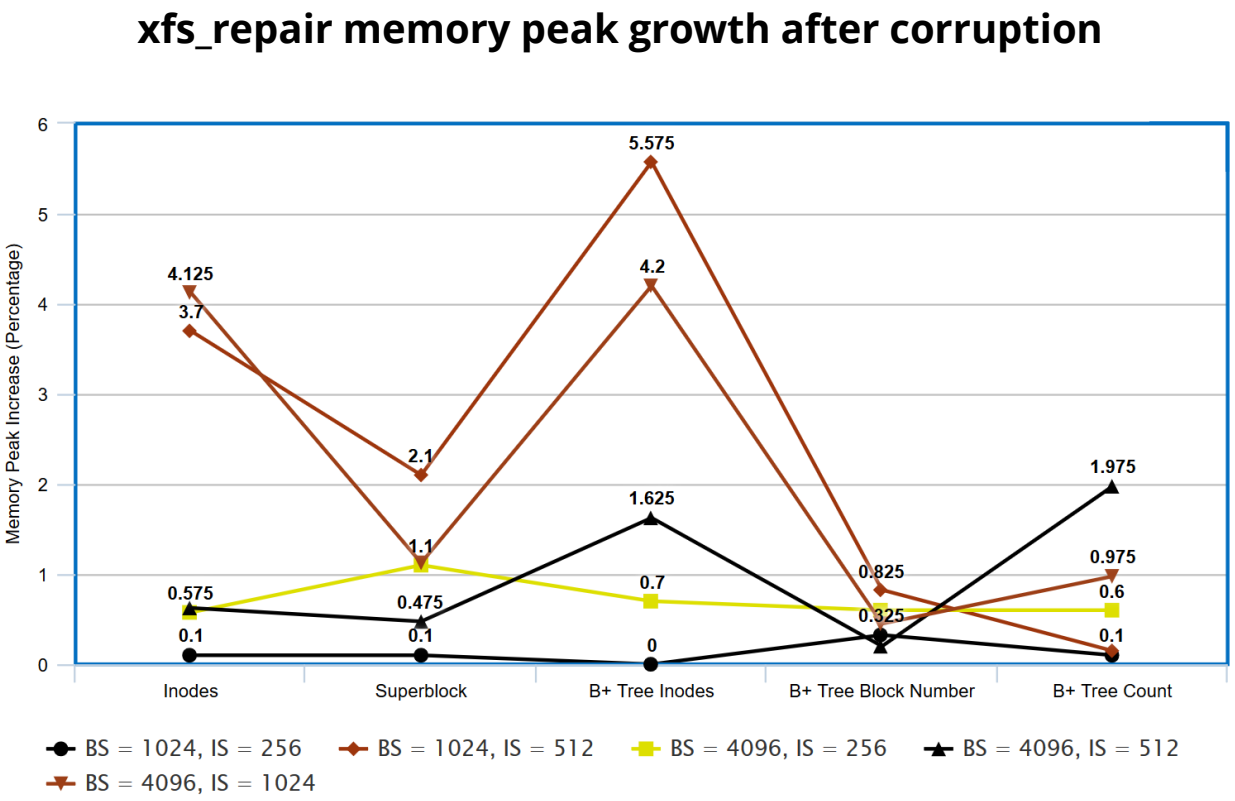
- Greatest slowdown caused by corruption of the B+ tree inodes.



Evaluation

- **Memory Peak Comparison (XFS)**

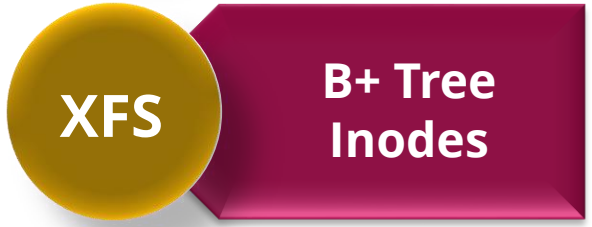
- Greatest memory increase caused by corruption of the B+ tree inodes.



Evaluation

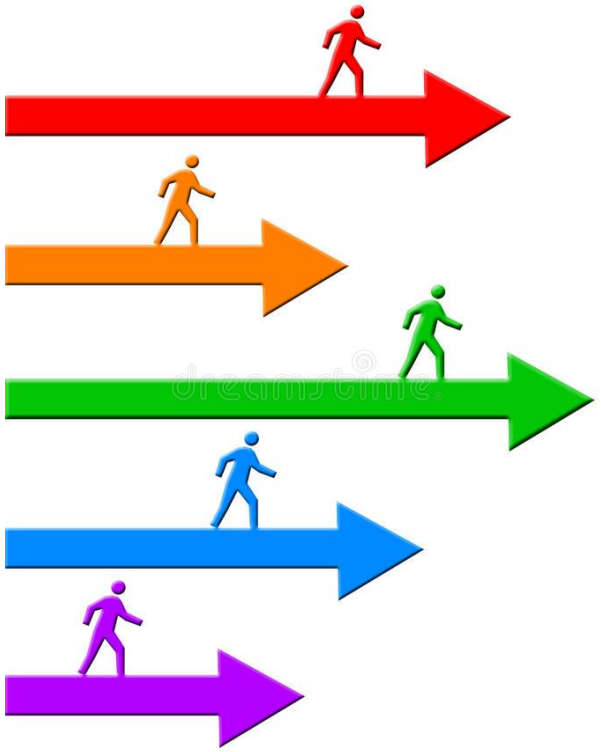
- Based on the results, we can see that for XFS and EXT4, our proposed aging tools behave competitively in regard to grep tests, and are in fact preferable for scenarios that require checker slowdown.

- Metadata corruption with the largest impact:
  - EXT4 – Group Descriptors
  - XFS – B+ Tree Inodes



Future Work

- Apply the analysis to other file systems!
- Compare with different degrees of aging.
- Improve the checker source code in response to the weaknesses found.



Conclusion

- Group descriptor corruption had the greatest effect on the performance of e2fsck, and B+ tree inode corruption had the greatest effect on the performance of xfs_repair.
- Three accessible, user-friendly tools were created, that are not only useful for this project, but could be applied in other contexts.

References

[1] A. Ma, C. Dragga, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. ffsck: The Fast File System Checker. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '13)*, San Jose, California, February 2013.

[2] A. Conway, A. Bakshi, Y. Jiao, W. Jannen, Y. Zhan, J. Yuan, M. A. Bender, R. Johnson, B. C. Kuszmaul, D. E. Porter, et al. File Systems Fated for Senescence? Nonsense, Says Science! In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '17)*, Santa Clara, California, February – March 2017.

[3] O. R. Gatla, M. Hameed, M. Zheng, V. Dubeyko, A. Manzanares, F. Blagojevic, C. Guyot, and R. Mateescu. Towards Robust File System Checkers. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '18)*, Oakland, California, February 2018.

References

[4] High Performance Computing Center (HPCC) Power Outage Event. Email Announcement by HPCC, Monday, January 11, 2016 at 8:50:17AM CST. https://www.cs.nmsu.edu/mzheng/docs/failures/2016-hpcc-outage.pdf, 2016.

[5] V. Tarasov, E. Zadok, and S. Shepler. Filebench: A Flexible Framework for File System Benchmarking. USENIX ;login: 41, 1 (2016).

[6] fs mark: https://github.com/josefbacik/fs mark

[7] S. Kadekodi, V. Nagarajan, and G. A. Gibson. Aging Gracefully with Geriatrix: A File System Aging Suite (2016).

[8] M. Seltzer, and K. Smith. File system aging: Increasing the relevance of le system benchmarks. In Proceedings of the 1997 ACM SIGMETRICS, ACM, New York, 1997.

Questions?