



## Using the HFS+ Journal For Deleted File Recovery

*By*

**Aaron Burghardt, Adam Feldman**

*Presented At*

The Digital Forensic Research Conference

**DFRWS 2008 USA** Baltimore, MD (Aug 11<sup>th</sup> - 13<sup>th</sup>)


DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

**<http://dfrws.org>**

# Using the HFS+ Journal For Deleted File Recovery

Aaron Burghardt  
Adam Feldman

# Introduction



*R.I.P.*

*Mac Norton*

*Utilities*

*April, 2004*

- Client-sponsored assignment
- Tasked to replace existing deleted file recovery tools
- Increase automation and improve accuracy

# Catalog File Records

Key

Cat. File Rec.

CNID

Create Date

Mod Date

Access Date

Owner ID

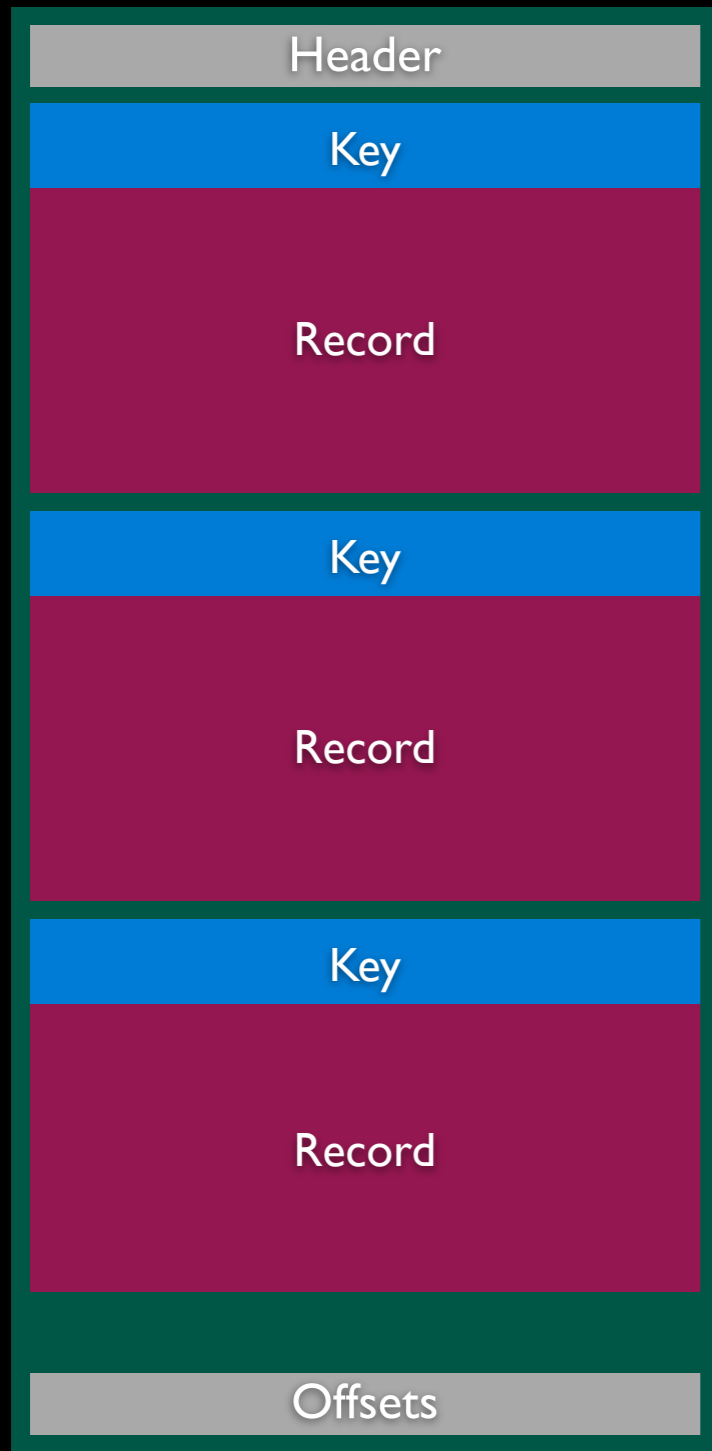
Group ID

Unix permissions

Extents

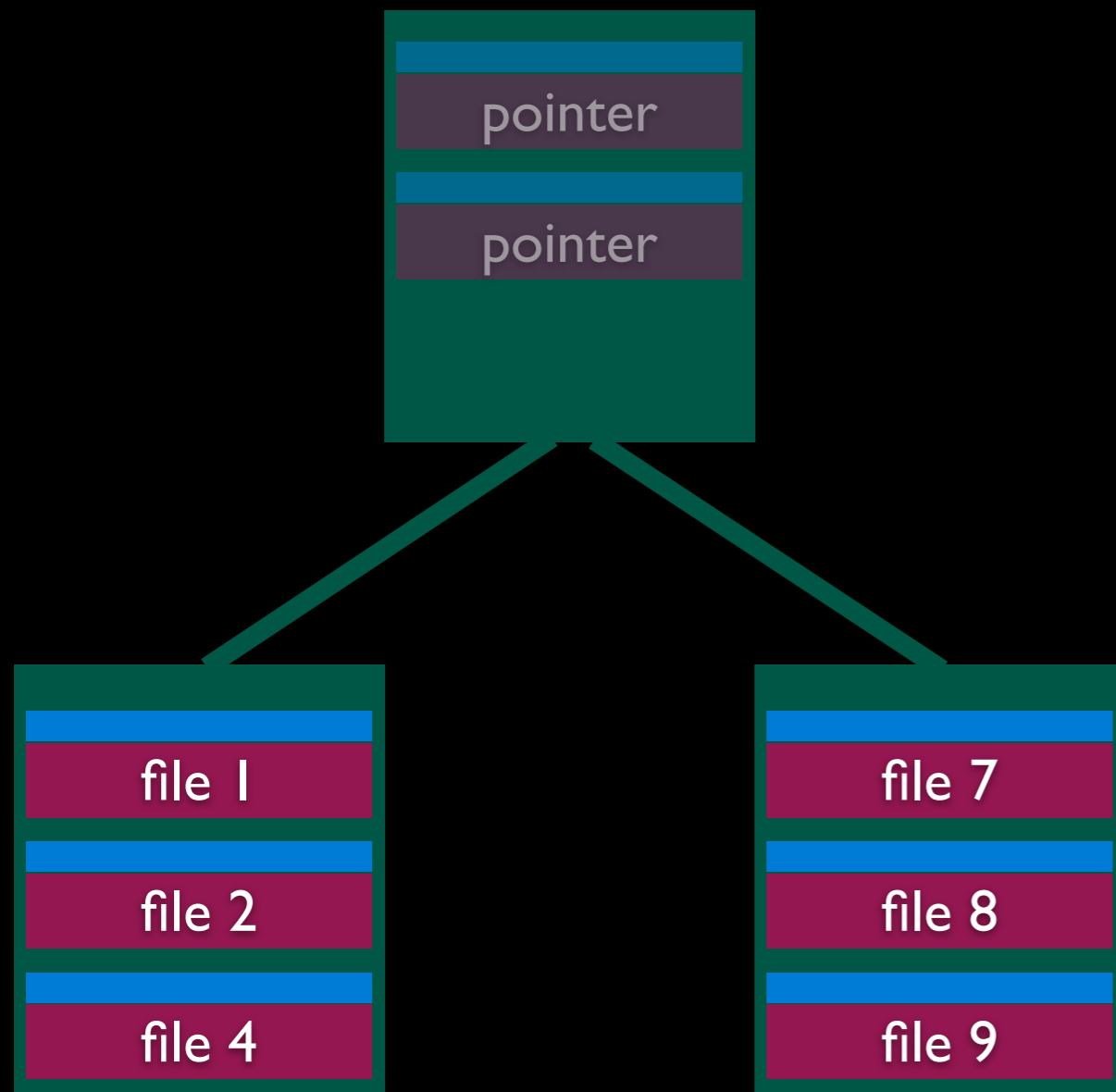
- Catalog Node ID (unique, like an inode)
- Create, mod, access times
- Owner and group IDs
- Unix permissions
- Extent Records (i.e., fragment descriptors)
- Stored adjacent to Key in B-tree node
- Key = Parent CNID + file name

# B-tree Node



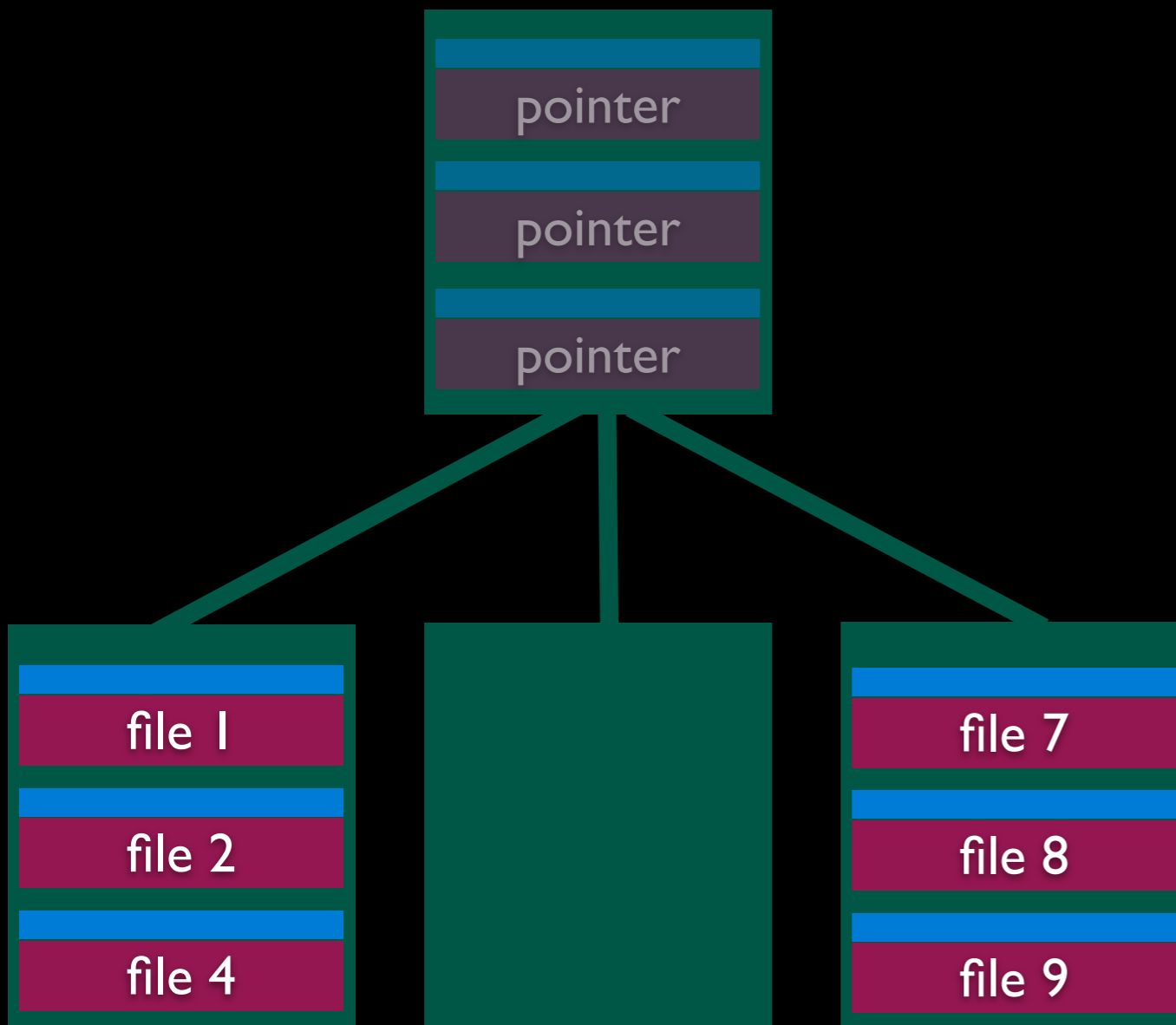
- Typically 8 KB
- Records and keys vary in size
- Records/Keys packed top-to-bottom

# B-tree Storage



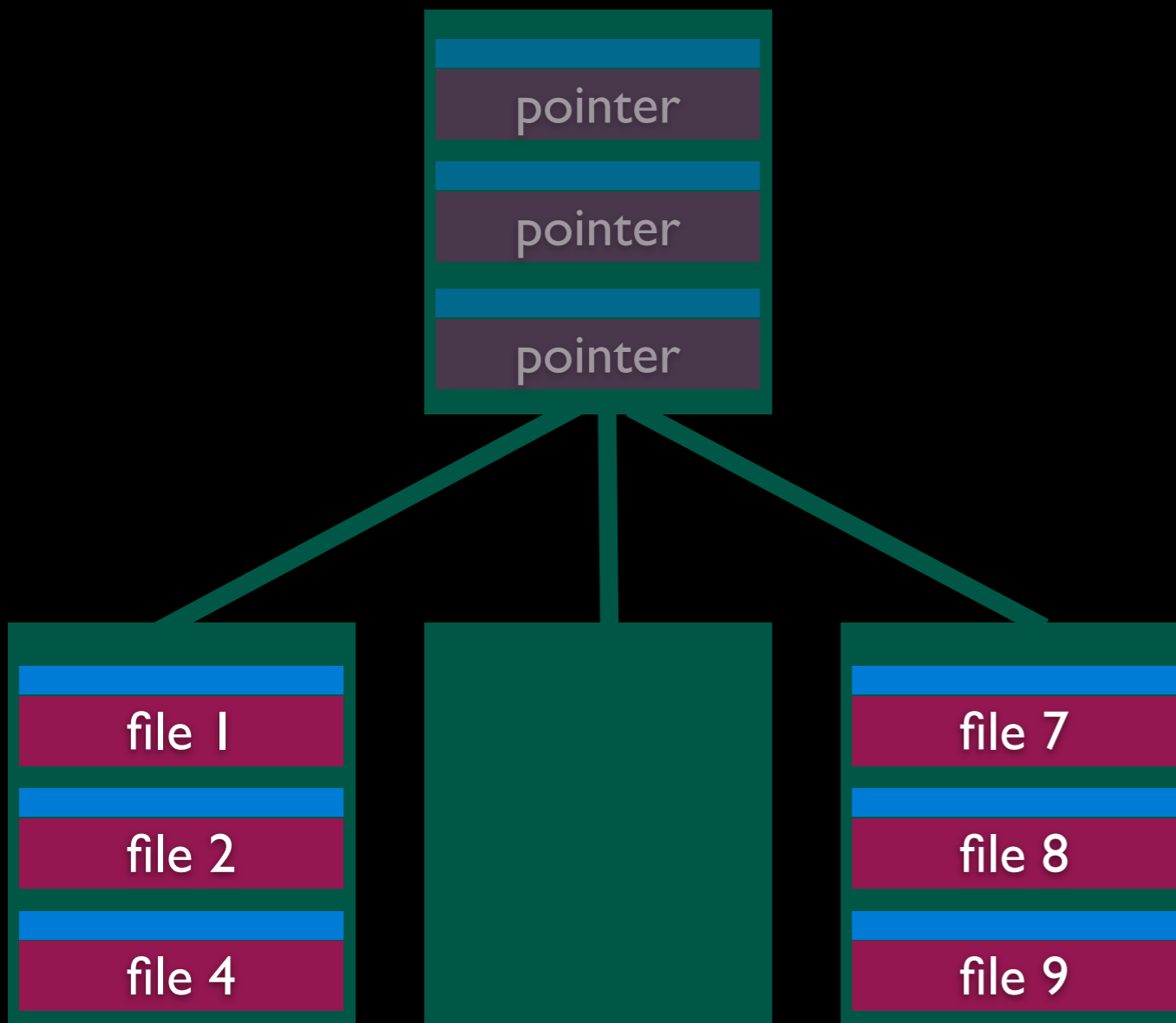
- Nodes are organized in a tree
- Records always maintained in sorted order
- Creation and deletion of files causes records to rearrange

# B-tree Storage



- Nodes are organized in a tree
- Records always maintained in sorted order
- Creation and deletion of files causes records to rearrange

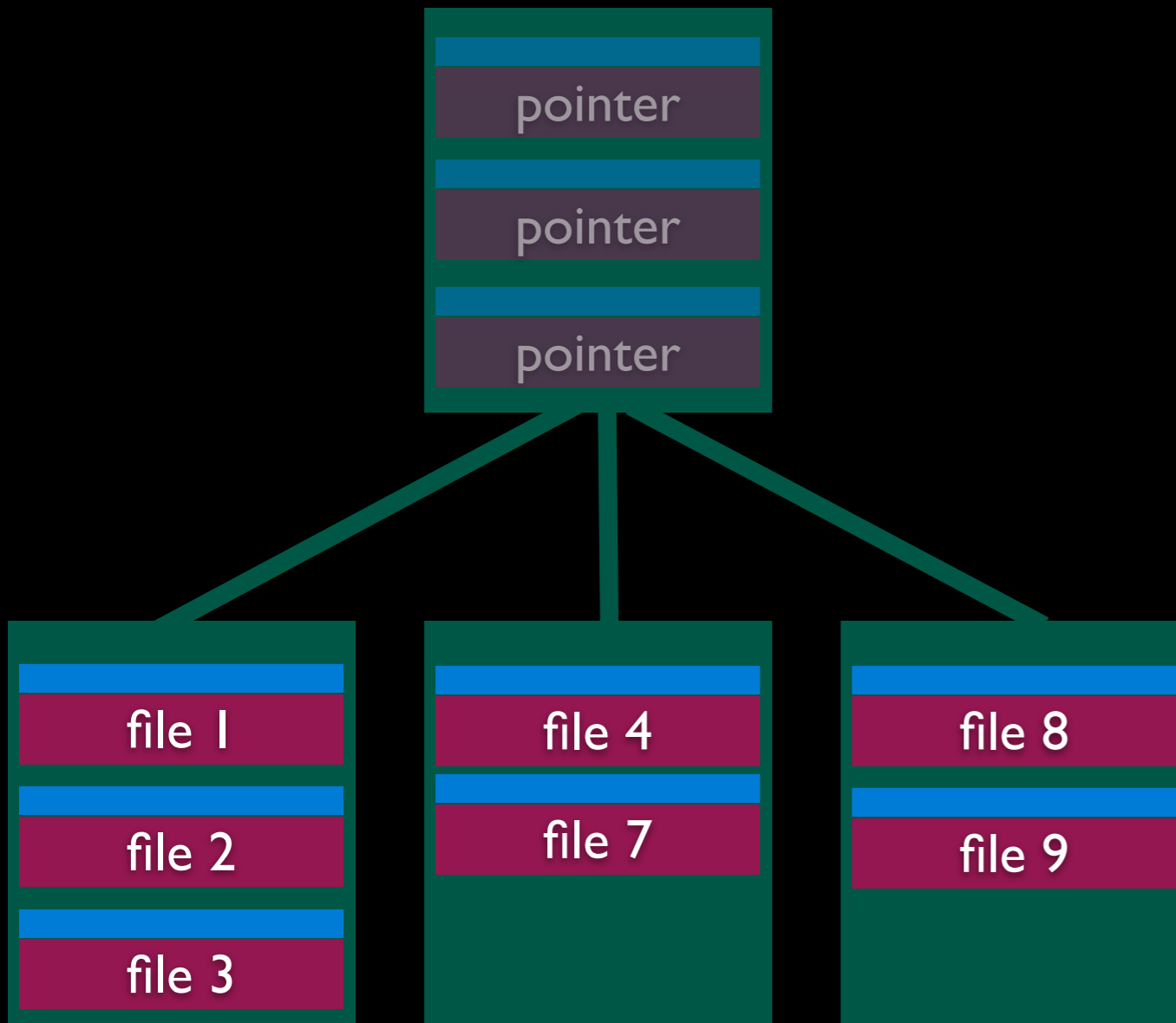
# B-tree Storage



- Nodes are organized in a tree
- Records always maintained in sorted order
- Creation and deletion of files causes records to rearrange

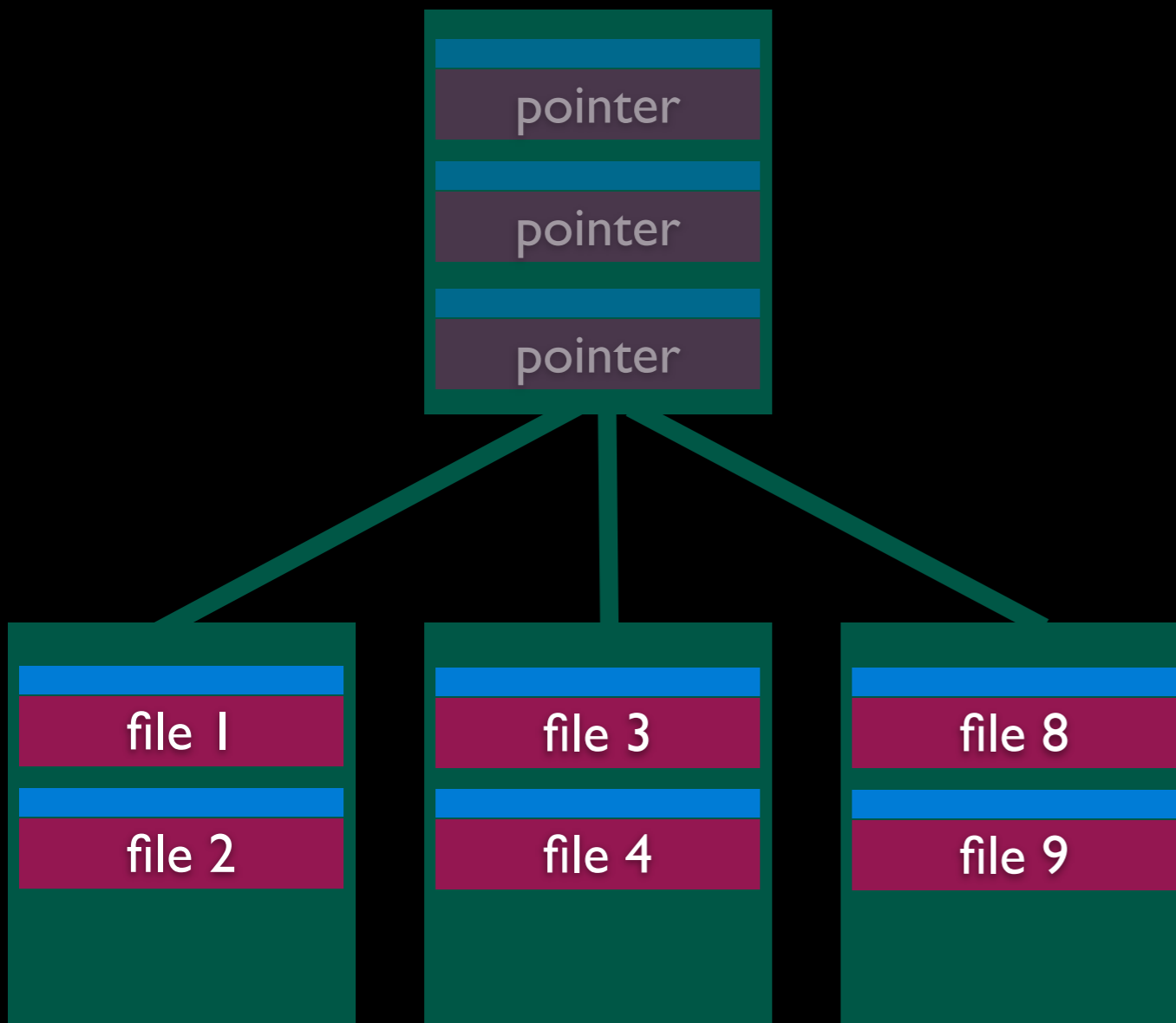


# B-tree Storage



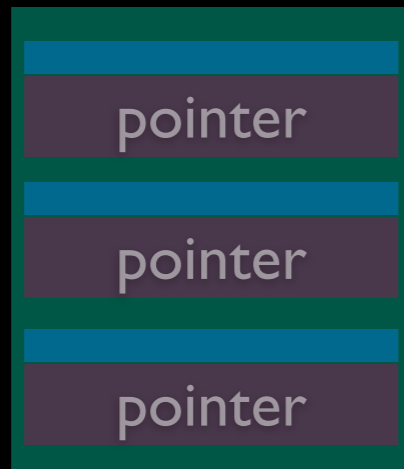
- Nodes are organized in a tree
- Records always maintained in sorted order
- Creation and deletion of files causes records to rearrange

# B-tree Storage

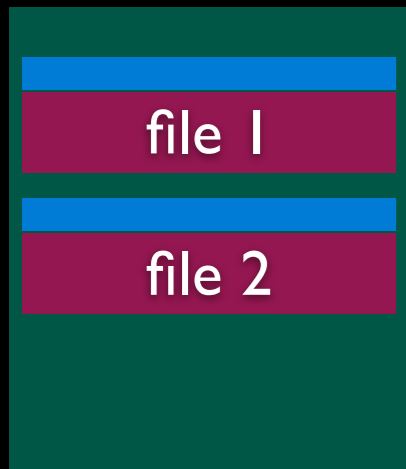


- Nodes are organized in a tree
- Records always maintained in sorted order
- Creation and deletion of files causes records to rearrange

# File System Files



- Volume Bitmap
- Catalog File
- Extents Overflow
- Extended Attributes



# File System Files

- Volume Bitmap
- Catalog File
- Extents Overflow
- Extended Attributes



Catalog File

# File System Files

Volume Bitmap

- Volume Bitmap

 Catalog File

- Catalog File

Extents Overflow

- Extents Overflow

Extended Attributes

- Extended Attributes

# Role of Journal

Volume Bitmap

Catalog File

Extents Overflow

Extended Attributes

# Role of Journal

Volume Bitmap

Catalog File

Extents Overflow

Extended Attributes

Journal File

- Introduced in Mac OS X 10.2
- Records pending changes to metadata
- Collects related changes in transactions
- Sector/Block-oriented
- Allocation: 8 MB + 8 MB per 100 GB vol. size

# Role of Journal



Volume Bitmap

Catalog File

Extents Overflow

Extended Attributes

Journal File

- Introduced in Mac OS X 10.2
- Records pending changes to metadata
- Collects related changes in transactions
- Sector/Block-oriented
- Allocation: 8 MB + 8 MB per 100 GB vol. size



# Key Points

Volume Bitmap

Catalog File

Extents Overflow

Extended Attributes

Journal File

# Key Points

Volume Bitmap

Catalog File

Extents Overflow

Extended Attributes

Journal File

- B-tree nodes are recorded as whole unit
- Catalog File, Extents Overflow, Extended Attributes are B-trees: must distinguish
- A Catalog File Record may appear in the journal due to unrelated changes

# Recovery Approach

1. Begin at logical start of journal file
2. Scan until a B-tree node is found
  - No header signature
  - Sanity checks used to validate
3. Iterate node records
  - a. Search the active Catalog File for each Catalog File Record
  - b. If not found, conclude it is a deleted file

# Recovery Approach (cont'd)

4. Cache the Catalog File Record in the deleted file cache:

- Replace duplicate (by CNID) record

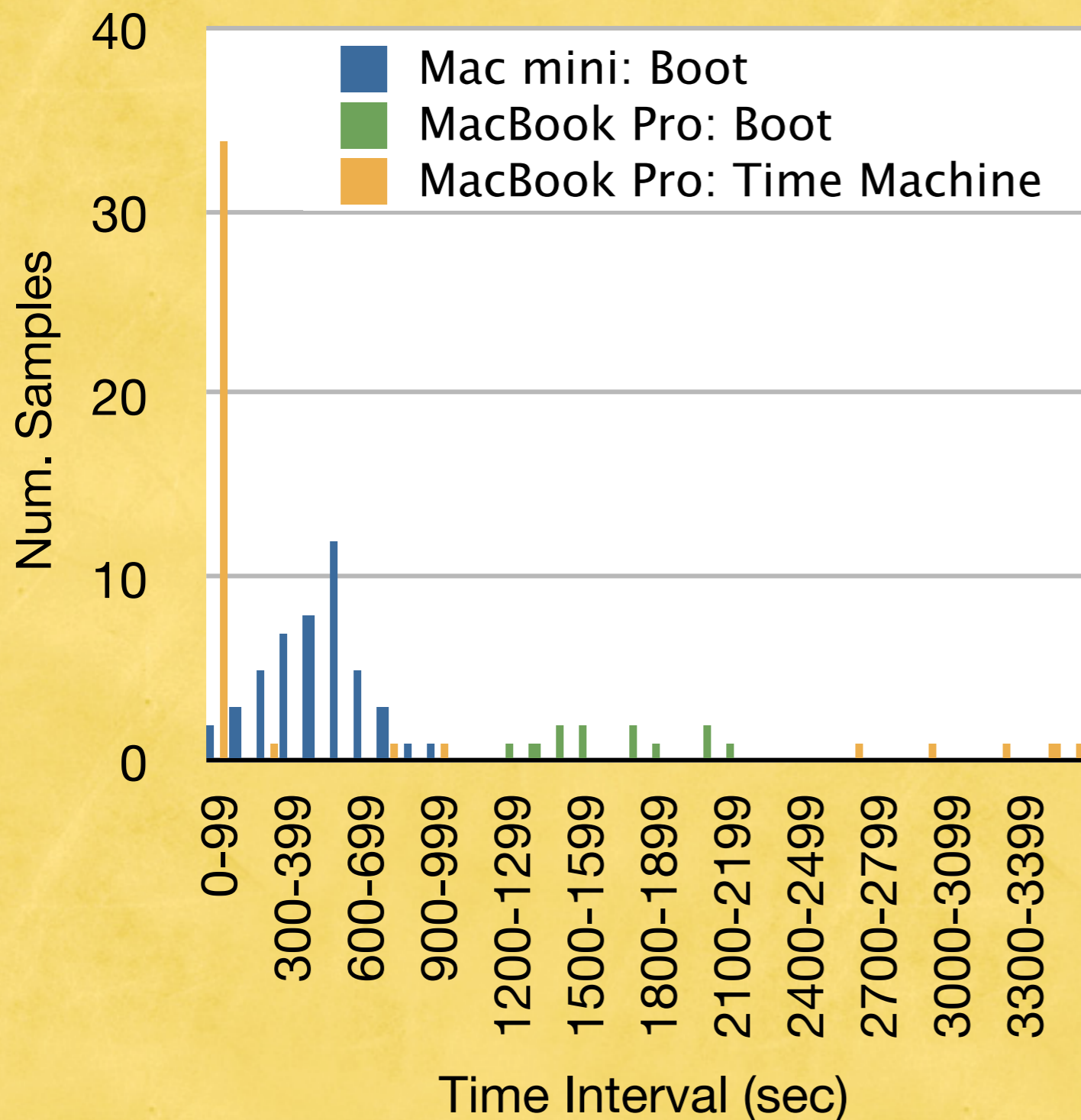
5. Score the recoverability:

- Check current in-use status of blocks
- Good: all blocks unused
- Partial: first block(s) not in use
- Poor: first block(s) in use

# Test Configurations

- Goal: establish typical “window of opportunity”
- Two test configurations:
  - Mac mini
  - MacBook Pro
- Mixture of use cases:
  - Boot volumes
  - Secondary volumes
  - Time Machine

# Lifetime of Data in Journal



- Boot volume: 5 min to 30 min
- Secondary: can be several hours or more
- Time Machine: idle between backups, approximately 30 sec during a backup

# Empirical Results

Volume	Good	Partial	Poor	Total
MBP: Boot	59	0	8	67
MBP: Time Mach	3	0	0	3
Mini: Boot	10	0	4	14
Mini: FireWire	32	0	87	119
Mini: FireWire	14	0	22	36
Mini: Flash	141	0	21	162

# Limitations

- Data in journal is short-lived
- Evidence of the file must be in the journal prior to it being deleted
  - Deleted status determined by deduction
  - Can't predict if a deleted file is detectable
- Path may not be recoverable
- Only has 8 extent records
- Time of deletion unknown



# Summary

- Effective for recently deleted files
- Recovers files and metadata with high accuracy
- Limited by short window of opportunity and the need for record to exist in journal prior to deletion
- Complementary to file carving

# Thank you

Aaron Burghardt  
burghardt\_aaron@bah.com

Adam Feldman  
feldman\_adam@bah.com

## Questions?