Deciding when to forget in the Elephant file system

Doug Santry, Mike Feeley, Norm Hutchinson,
Alistair Veitch*, Ross Carton, and Jacob Ofir

University of British Columbia

Hewlett-Packard Laboratories*

Protecting file system data

- System and media failure
  - Focus of file-system research for many years
- User and application failure
  - No protection
  - Delete and write cause data loss
  - Artifact of limited storage capacity

Storage is no longer limiting

- Disk capacity trends
  - 25 – 35 GB now
  - Increasing by 60% per year
  - 250 – 350 GB in 5 years
- Disks are now:
  - Big enough to keep some old versions
  - Not big enough to keep everything

Protecting data with big disks

- Key idea
  - Retain important old versions of files
  - System, not user, controls storage reclamation
- Key issues
  - Is versioning at granularity of file or file system?
  - How long are old versions retained?
  - How can users control retention safely?

Previous work

- File-system grain
  - Copy-on-write checkpoint of entire file system
  - Performed periodically
  - E.g., Plan-9, WAFL, AFS
- File grain
  - Copy-on-write of individual files
  - Performed continuously
  - E.g., Cedar, VMS
    - Retained last few versions
    - No protection from delete

Elephant overview

- Delete and write
  - Do not cause data loss immediately
- Storage reclamation
  - File-grain retention policies specified by users
  - Policies implemented by system cleaner
- User interface
  - Rollback to any point in the past
    - {open,cd,…} filename@yesterday:12:00

Talk outline

- Principles and retention policies
- Prototype implementation
  - Meta data
  - File and name histories
- Evaluation
  - Workload analysis
  - User experience

Protection depends on file type

- Read only
- System managed
  - Derived
  - Cached
  - Temporary
- User managed

Principles

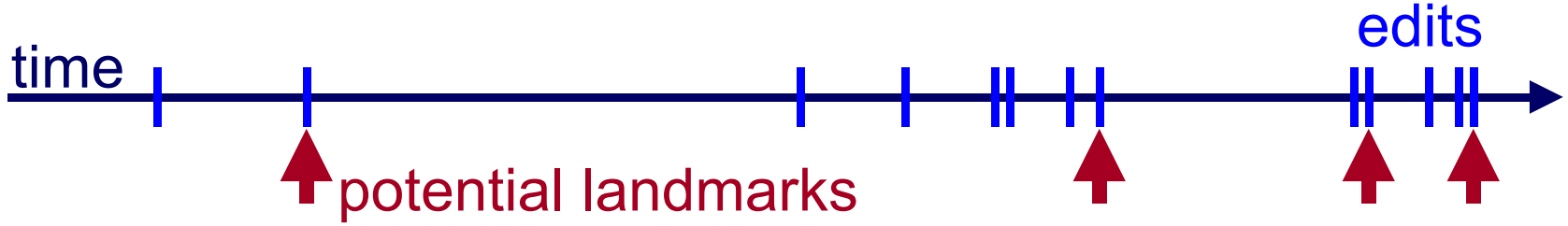
- Near-term reversibility
  - Of every operation on valuable data
  - For a limited period of time
- Long-term history
  - Of selected files
  - Including only selected *landmark* versions

File-grain retention policies

- Keep One
  - Update date in place and immediate delete
- Keep All
  - Retain all versions
- Keep Safe
  - Retain all versions for second-chance interval
- Keep Landmarks
  - Retain only landmark versions

Potential-landmark heuristic

- Key observations
  - Files are modified in barrages
  - Ability to differentiate edits degrades with time
- Strategy
  - Designate lead edit of barrage as landmark
  - Barrage “granularity” increases with time



History discontinuities

- Deleted versions
  - Discontinuity in file's history
  - System can report all discontinuities to user
- Grouping files
  - User groups related files
  - A landmark of any file is landmark for group

User implemented policies

- New policies
  - Written as user-level programs
  - Registered with kernel
  - Used in the same way as standard polices
- Cleaning
  - System cleaner execs user-policy program
  - Runs with privileges of file's owner

Elephant prototype

- Implementation
  - New VFS in FreeBSD 2.2.8
- Interface
  - Add time to any pathname “file@time”
  - Set process’s default time
  - Set file’s policy or group files
  - Make version a landmark
  - Read a file’s history
  - Tools including: tls, tgrep, tdiff, and tview

Versioning meta data

- Inode history
  - Inode log contains file's copy-on-write inodes
  - Inode added to log on first write after open
  - Non-versioned files stored by standard inode
- Name history
  - Directory lists name creation and deletion time
  - Name retained until all file versions are deleted
  - Old names periodically moved to history inode

Two views of history

- File (inode) history
  - All versions of a file independent of its name
  - Rename not reflected in file history
- Name history
  - Name can refer to different files at different times
  - Some applications rely on name history
    - Modify file by first renaming to backup (e.g., emacs)
- Elephant provides both views of history

Workload analysis

- Measured system
  - Workgroup server at HP Labs
  - Supporting 12 active researchers
  - Used for development, document prep., etc.
  - 15 GB, 360,000 files, 27,000 directories
- Analysis
  - File-type distribution
  - Write-traffic distribution

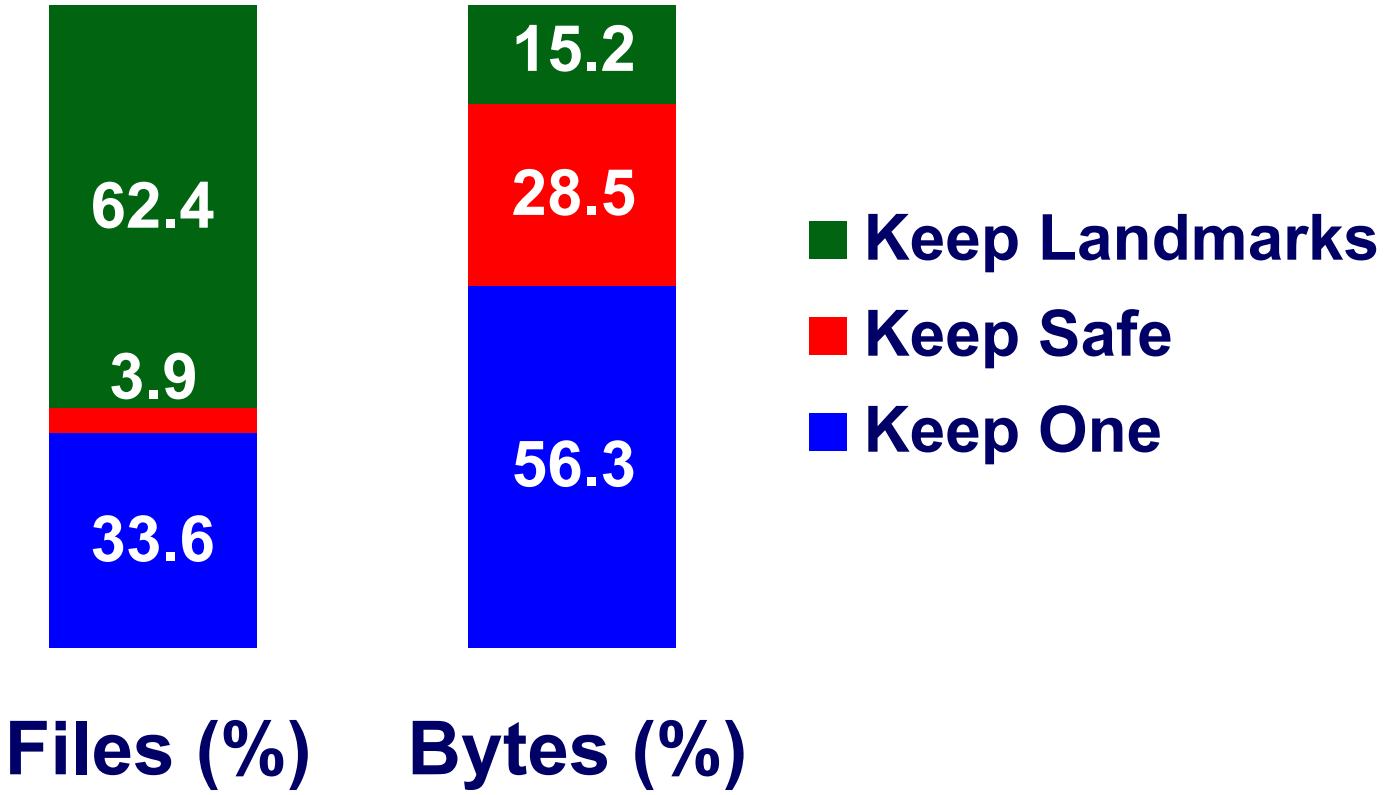
File-type taxonomy

- Source
  - C, C++, perl, shell scripts
- Documents
  - text, HTML, word processor, mail
- Derived
  - object, library, exec, postscript, PDF
- Archive
  - tar, compressed, data
- Temporary
  - *.tmp, web-browser caches

Allocating policies by file type

- Keep One
  - Derived
  - Temporary
- Keep Safe
  - Archive
- Keep Landmarks
  - Source
  - Documents
  - Other

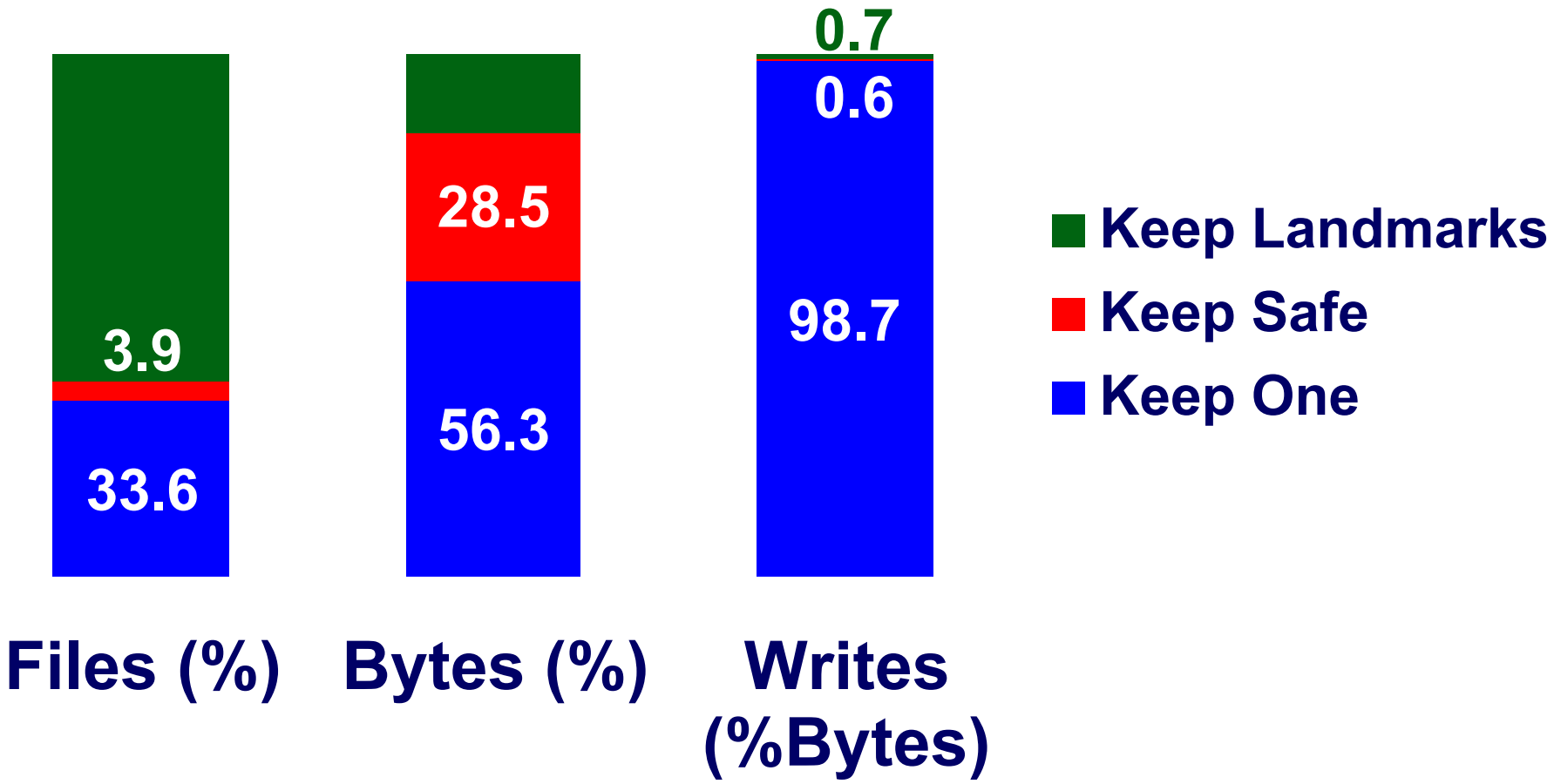
Storage by policy



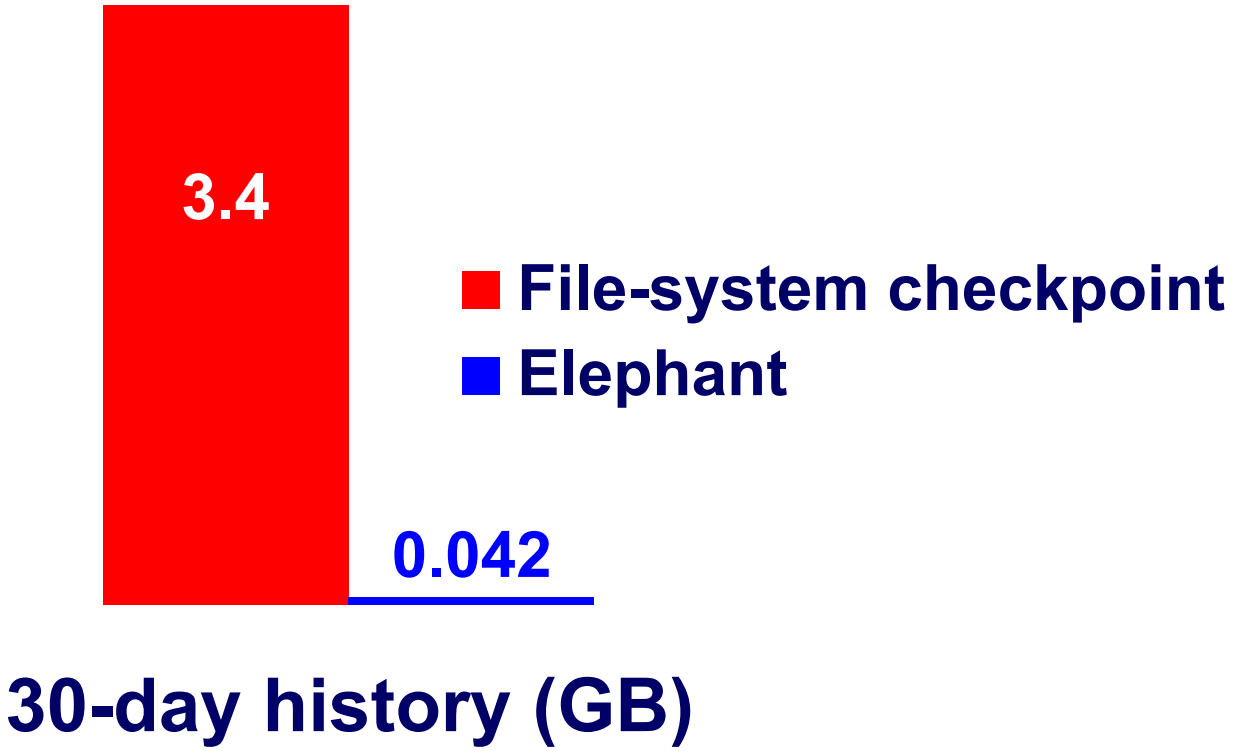
Write traffic

- Trace
  - Same HP-Labs workgroup server
  - Collected Aug 29 – Oct 8, used Sep 27 – Oct 1
  - Records all open, close, read, and write
  - Includes file name
- Summary
  - 112 MB / day written on average
  - 15 GB of total storage, 12 active users

Storage growth by policy



Importance of file-grain retention



NFS shadowing

- Problem
  - Would you trust your data to a research FS?
- Solution
  - Elephant prototype can shadow an NFS server
    - Snoops network for NFS packets
    - Updates shadow Elephant file system
  - Users
    - Create and update files via NFS
    - Read current and historic versions via Elephant

Conclusions

- Protecting data from users and applications
  - Files require different degrees of protection
    - Reversibility: all versions for limited period
    - History: landmark versions forever
  - Important versions are small fraction of disk
- Elephant
  - File-grain retention policies specified by users
  - Retains all important older versions
  - Rollback file, directory, or fs to any point in past