# Effective Use of CSAIL Storage

How to get the most out of your computing infrastructure

Garrett Wollman,
Jonathan Proulx,
and Jay Sekora
*The Infrastructure Group*

# Introduction

# Outline of this talk

1. Introductions
2. Existing storage situation
3. Where we are going
4. Filesystems are cheap…
5. …But backups are expensive
6. Storage usage anti-patterns
7. Before you ask for (or start using) CSAIL storage
8. Positive storage usage design patterns
9. Q&A

# Existing storage situation

- Three (four?) tiers of storage

  1. CSAIL AFS cell: slowest but most secure
  2. NFS on Linux servers: faster, no security
     - Back-end storage is XFS over Linux LVM over outboard RAID controller
  3. NFS on Hitachi (BlueArc) appliance
     - Back-end storage is proprietary
  4. Local disk on your workstation/server: fastest, not shared, no backups

- A few filesystems already on the new infrastructure

- Thanks to automounter, NFS storage server is mostly invisible

# Existing storage situation

- Backups: all data stored on file servers is backed up, except:
  - “scratch” filesystems
  - Anything you told us didn’t need it
- When files are changing a lot, backups require more resources
  - On file servers
  - On backup server
- The servers really do get slower when you have a deadline!

# Current utilization

| File server | # of filesystems | # of files | size of files | free space* |
|---|---|---|---|---|
| zim | 39 | 120,634,169 | 35,271 GB | 18,305 GB |
| gir | 15 | 274,658,812 | 42,546 GB | 6,716 GB |
| phobos† | 24 | 6,876,904,664 | 115,200 GB | 5,353 GB |

*nearly all free space is allocated to existing filesystems but not yet being used
†includes snapshots; file count may be erroneous

# Where we are going

- Single hardware and software platform for all NFS storage
- Two tiers: “scratch” and backed-up
- Backed-up tier has:
  - Online disk-to-disk first-level backup
  - Backups come from first-level backup server, not production server
  - Hourly (and weekly, and daily, and monthly) snapshots of everything
- Scratch tier is engineered for speed
- Both tiers have SSD caching and write-ahead log

# New hardware platform

- Donated by Quanta Computer

  Please thank them!

- Storage server
  - 12-core dual-socket Intel Xeon
  - dual 10-Gbit/s Ethernet
  - 96 GB RAM
  - dual 16-port LSI 6 Gbit/s external SAS

- Disk complement
  - 4 dual-pathed 24-drive shelves (1:6 SAS port expander)
  - 92 2-terabyte SAS drives, 2 8-megabyte RAM SSDs, 2 240-megabyte flash SSDs

# New software platform

- FreeBSD 9.1 with ZFS

  Please thank people who used to work for Sun!

- Single storage pool
- 11 RAID-Z2 groups of 8 drives each
  - Allows us to survive the failure of an entire disk shelf
  - 4 hot-spare drives
- Redundancy for log drives, extra speed and capacity for cache
- Compression, Unicode on by default

# Filesystems are cheap…

- Cannot emphasize this enough!
  - A filesystem is a logical entity, not a physical resource.
  - Creating one takes TIG about fifteen minutes, most of which is spent interacting with you.
  - *The physical resource (disk space)* ***is*** *limited.*
- If you think you might need a new filesystem, just ask
  - We'll tell you how and why later in this presentation.
- Filesystems are the fundamental unit of organization for storage
  - Used for naming and access control
  - Container for file server and storage backend settings
  - Container for backup server settings

# …Backups are expensive

- Backups are the largest ongoing cost related to storage
  - Backup system is licensed software; cost depends on number of simultaneous processes needed to complete backups in a reasonable time.
  - Backups are written to tape, and tapes are periodically sent to off-campus storage provider.
  - Cloud-based backup is even more expensive at this scale (and only replaces the tape, not the software)
- Important to identify data that does not need to be backed up
  - Temporary and intermediate files
  - Anything that can be easily recreated
- Backup tape cost is O(data size + churn); backup time is O(# files)

# Anti-patterns

- *Nearly everything we currently do at CSAIL is an anti-pattern*
- Organizing namespace by the user who first generated the data
- Building large datasets deep in the directory hierarchy
- Putting archival data in the same filesystem as rapidly churning cluster output
- Numbering filesystems rather than giving meaningful names
- Creating excessively large directories

# Before you ask for storage

- This part of the talk is covered on our wiki page on NFS storage: tig.csail.mit.edu/wiki/TIG/NFSStorage
- Think about the data you are going to be storing:
  - Does this data need to be backed up?
  - What is the access pattern: write-once, read-many (archival) or interleaved reads and writes?
  - Is the data compressible or not?
  - What is the structure of the data? Lots of little files? A few really big files? How many, or how big?
  - Are there special security requirements for this data? If so, NFS may not be appropriate.
  - *If any of these answers are different for parts of your data, those parts belong is a different filesystem.*
- How much space is required? How much will it grow over time?
- We can help answer some of these questions.

# Storage usage design patterns

- Talking specifically about normal Unix-style programming with NFS
  - GFS (either one), HDFS, S3 are  that way.
- Use meaningful names at the top level of the filesystem
- Introduce hierarchy to break up large directories of similar content
- Store data in archives when not in use
  - But consider carefully the appropriate level of the hierarchy at which to do this
- Don’t write anything in append mode (`"a"`, `">>"`, `O_APPEND`)
- Use local scratch space for intermediate outputs, then move to NFS

# Q&A

people.csail.mit.edu/wollman/2012-NFS-storage.pdf