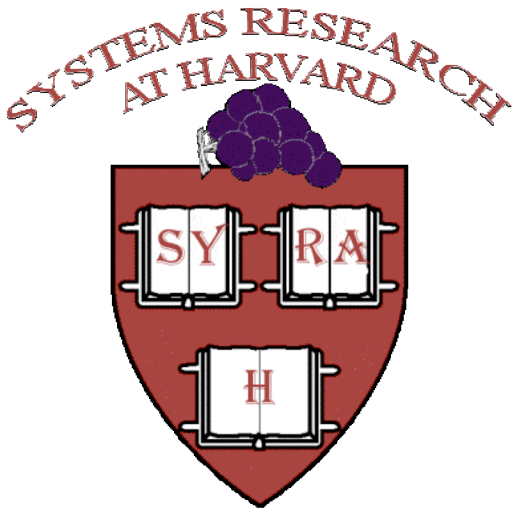
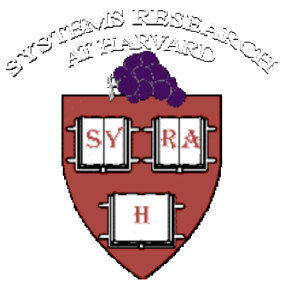
# Passive NFS Tracing of Email and Research Workloads



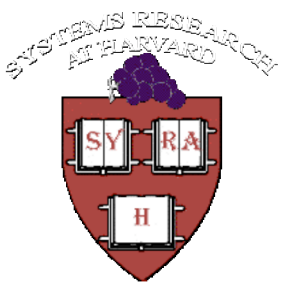

Daniel Ellard, Jonathan Ledlie, Pia Malkani, Margo Seltzer

FAST 2003 - April 1, 2003



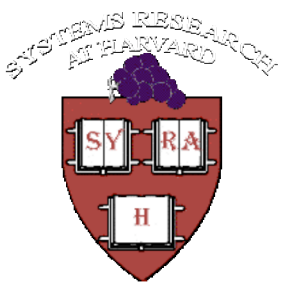

# Talk Outline

- Motivation
- Tracing Methodology
- Trace Summary
- New Findings
- Conclusion



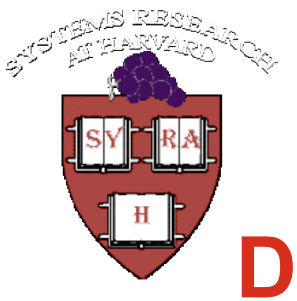

# Motivation - Why Gather Traces?

- Our research agenda: build file systems that
  - Tune themselves for their workloads
  - Can adapt to diverse workloads
- Underlying assumptions:
  - There is a significant variation between workloads.
  - There are workload-specific optimizations that we can apply on-the fly.
- *We must test whether these assumptions hold for contemporary workloads.*



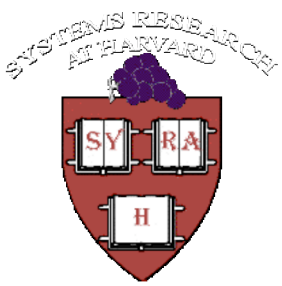

# Why Use Passive NFS Traces?

- Passive: no changes to the server or client
  - Sniff packets from the network
  - Non-invasive trace methods are necessary for real-world data collection
- NFS is ubiquitous and important
  - Many workloads to trace
  - Analysis is useful to real users
- Captures exactly what the server sees
  - Matches our research needs



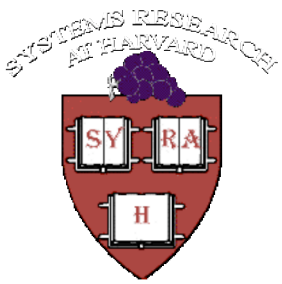

# Difficulties of Analyzing NFS Traces

- Underlying file system details are hidden
  - Disk activity
  - File layout
- The NFS interface is different from a native file system interface
  - No open/close, no seek
  - Client-side caching can skew the operation mix
- Some NFS calls and responses are lost
- NFS calls may arrive out-of-order




# Our Tracing Software

- Based on tcpdump and libpcap (a packet-capture library)
  - Captures more information than tcpdump
  - Handles RPC over TCP and jumbo frames
- Anonymizes the traces
  - Very important for real-world data collection
  - Tunable to remove/preserve specific information
- Open source, freely available



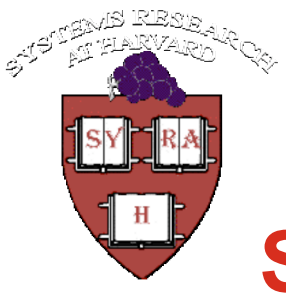

# Overview of the Traced Systems

## CAMPUS

- Central college facility
- Almost entirely email: SMTP, POP/IMAP, pine
- No R&D
- “Normal” users
- Digital UNIX
- 53G of storage (1 of 14 home directory disks)

## EECS

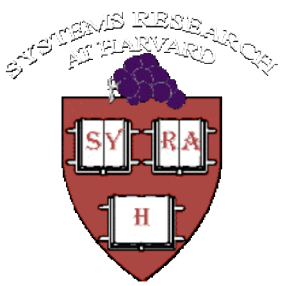
- EE/CS facility
- No email
- Research: software projects, experiments
- Research users
- Network Appliances filer
- 450G of storage




# Summary of Average Daily Activity

10/21/2001 - 10/27/2001

| | CAMPUS | EECS |
|---|---|---|
| Total Ops | 26.7 Million | 4.4 Million |
| Read Ops | 65% (119.6 GB) | 10% (5.1 GB) |
| Write Ops | 21% (44.6 GB) | 15% (9.1 GB) |
| Other | 14% | 75% - getattr, lookup, access |
| R/W Ops | 3.01 | 0.69 |



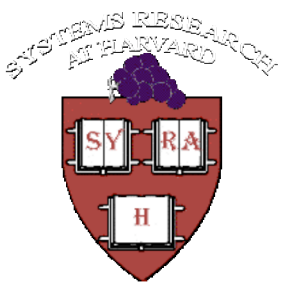

# Workload Characteristics

## CAMPUS

- Data-Oriented
- 95%+ of reads/writes are to large mailboxes
- For newly created files:
  - 96%+ are zero-length
  - Most of the remainder are < 16k
  - < 1% are “write-only”

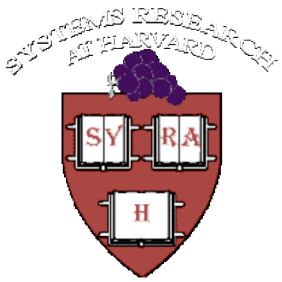
## EECS

- Metadata-Oriented
- Mix of applications, mix of file sizes
- For newly created files:
  - 5% are zero-length
  - Less than half of the remainder are < 16k
  - 57% are “write-only”



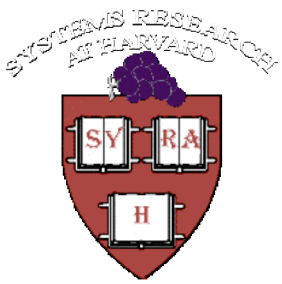

# How File Data Blocks Die

- CAMPUS:
  - 99.1% of the blocks die by overwriting
  - Most blocks live in “immortal” mailboxes
- EECS:
  - 42.4% of the blocks die by overwriting
  - 51.8% die because their file is deleted
- *Overwriting is common, and a potential opportunity to relocate/reorganize blocks on disk*



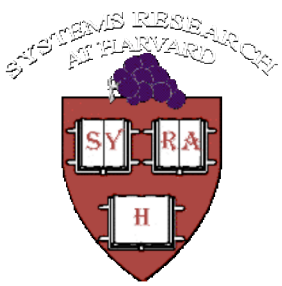

# File Data Block Life Expectancy

- CAMPUS:
  - More than 50% live longer than 15 minutes
- EECS:
  - Less than 50% live longer than 1 second
  - Of the rest, only 50% live longer than two minutes
- *Most blocks die in the cache on EECS, but on CAMPUS blocks are more likely to die on disk*



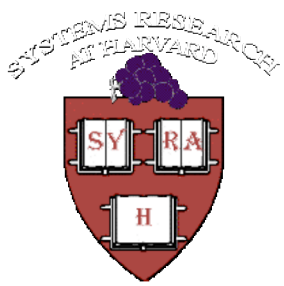

# Talk Outline

- Motivation
- Tracing Methodology
- Trace Summary
- **New Findings**
- Conclusion

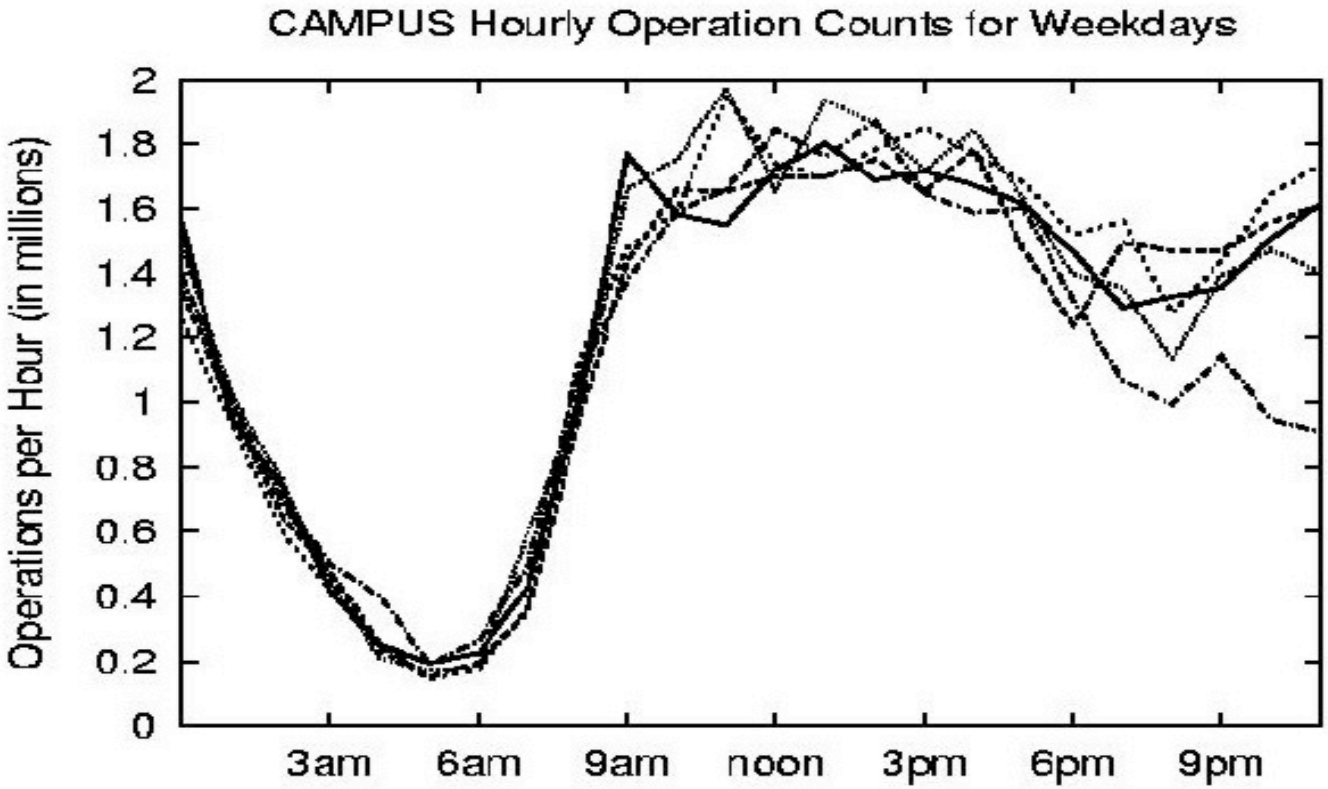


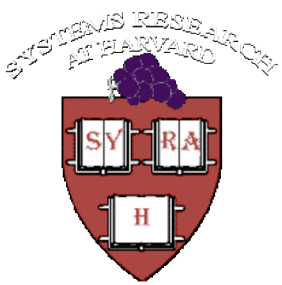

# Variation of Load Over Time

- EECS: load has detectable patterns
- CAMPUS: load is quite predictable
  - Busiest 9am-6pm and evenings Monday - Friday
  - Quiet in the late night / early morning
- *Each system has idle times, which could be used for file system tuning or reorganization.*
- *Analyses of workload must include time.*



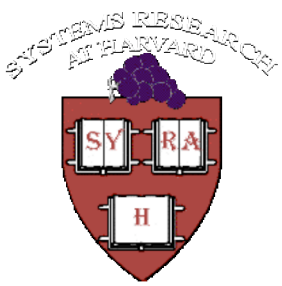

# The Daily Rhythm of CAMPUS





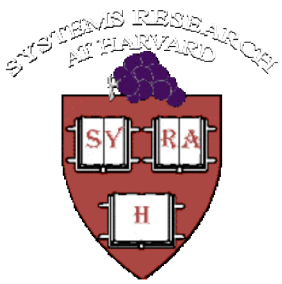

# New Finding: File Names Predict File Properties

- For most files, there is a strong relationship between the file name and its properties
  - Many filenames are chosen by applications
  - Applications are predictable
- The filename suffix is useful by itself, but the entire name is better
- The relationships between filenames and file properties vary from system to another



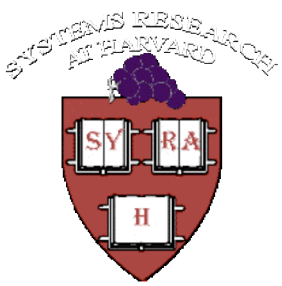

# Name-Based Hints for CAMPUS

- Files named “inbox” are large, live forever, are overwritten frequently, and read sequentially.
- Files with names starting with “inbox.lock” or ending with the name of the client host are zero-length lock files and live for a fraction of a second.
- Files with names starting with # are temporary composer files. They always contain data, but are usually short and are deleted after a few minutes.
- Dot files are read-only, except .history.




# Name-Based Hints for EECS

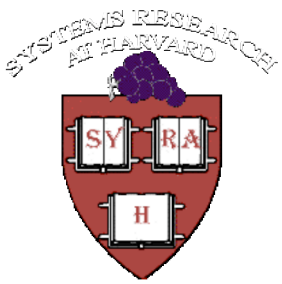
- On EECS the patterns are harder to see.
- We wrote a program to detect relationships between file names and properties, and make predictions based upon them
  - Developed this tool on CAMPUS and EECS data
  - Successful on other later traces as well
- *We can automatically build a model to accurately predict important attributes of a file based on its name.*




# Accuracy of the Models

Accuracy of predictions for EECS, for the model trained on 10/22/2001 for the trace from 10/23/2001.

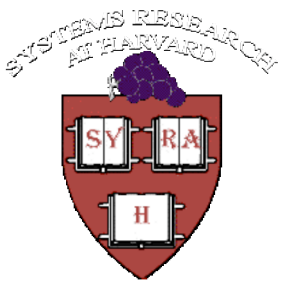
| | Prediction Accuracy | | | |
|---|---|---|---|---|
| Length = 0 | **99.4%** | | | |
| Length < 16K | **91.8%** | | | |




# Accuracy of the Models

Accuracy of predictions for EECS, for the model trained on 10/22/2001 for the trace from 10/23/2001.

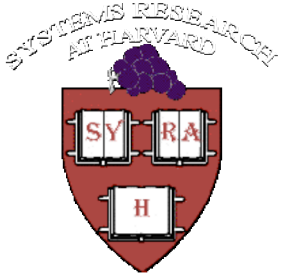
| | Prediction Accuracy | % of files | Accuracy w/o names | |
|---|---|---|---|---|
| Length = 0 | **99.4%** | **12.5%** | **87.5%** | |
| Length < 16K | **91.8%** | **35.2%** | **64.8%** | |




# Accuracy of the Models

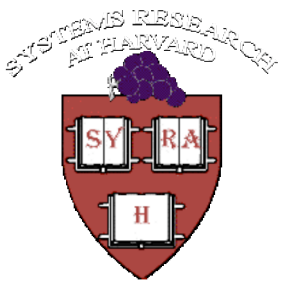
Accuracy of predictions for EECS, for the model trained on 10/22/2001 for the trace from 10/23/2001.

| | Prediction Accuracy | % of files | Accuracy w/o names | % Error Reduction |
|---|---|---|---|---|
| Length = 0 | **99.4%** | 12.5% | 87.5% | **94.9%** |
| Length < 16K | **91.8%** | 35.2% | 64.8% | **76.6%** |



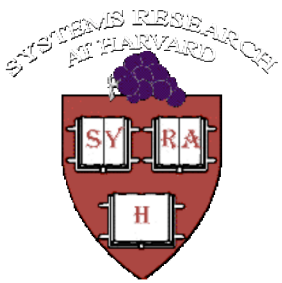

# New Finding: Out-of-Order Requests

- On busy networks, requests can be delivered to the server in a different order than they were generated by the client
  - nfsiods can re-order requests
  - Network effects can also contribute
- This can break fragile read-ahead heuristics on the server
- *We investigated this for FreeBSD and found that read-ahead was affected*




# Conclusions

- Workloads do vary, sometimes enormously
- New traces are valuable
  - We gain new insights from almost every trace
- We have identified several possible areas for future research:
  - Name-based file system heuristics
  - Handling out-of-order requests




# The Last Word

Please contact me if you are interested in exchanging traces or using our tracing software or anonymizer:

http://www.eecs.harvard.edu/sos

ellard@eecs.harvard.edu

Another resource:

www.snia.org