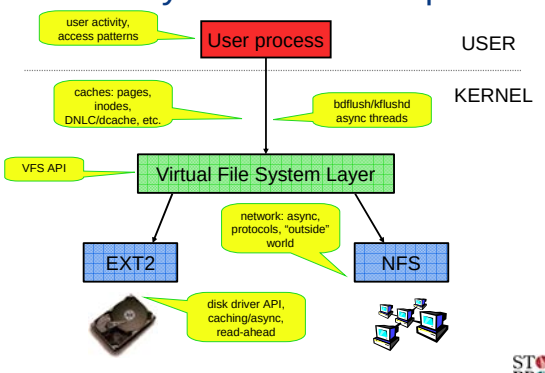
File System Benchmarking:
Fallacies, Pitfalls, and Beyond

Erez Zadok, Nikolai Joukov,
Avishay Traeger, and Charles P. Wright
Stony Brook University

http://www.fsl.cs.sunysb.edu/
Tech-Report FSL-05-04

File Systems are Complex



Survey

- Analyzed the benchmarking practices of 68 file systems research papers in:
  - Symposium on Operating System Principles (SOSP 1999, 2001, 2003)
  - Symposium on Operating Systems Design and Implementation (OSDI 2000, 2002, 2004)
  - USENIX Conference on File And Storage Technologies (FAST 2002,2003,2004)
  - USENIX Annual Technical Conference (including FREENIX track) (2002, 2003, 2004)

File System Benchmarking Practices Today

"We collected a one-hour NTFS file-system trace from a developer's machine in our research group."

"Primarily because it is customary to do so, we also ran a version of the Andrew benchmark."

File System Benchmarking Practices Today

"We repeated each experiment five times and took the best results, to eliminate the effects of other activity in the operating system. Generally, the best results are repeatable, with a few bad outliers that represent experiments that were interfered with by other activity affecting the processor and cache."

FAST Best Paper

Outline

- Motivation
- **Survey: Benchmarks Today**
  - **Macro-benchmarks**
  - Traces
  - Micro-benchmarks
  - Workload Generators
- Tools
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

PostMark (1997)

- Synthetic workload
- Workload typical of short-lived small files
  - E-mail
  - Netnews
  - Web commerce
- Has three phases:
  - Creates a pool of random text files with sizes in a specified range
  - Performs transactions consisting of a create or delete composed with a read or write
  - Deletes the files

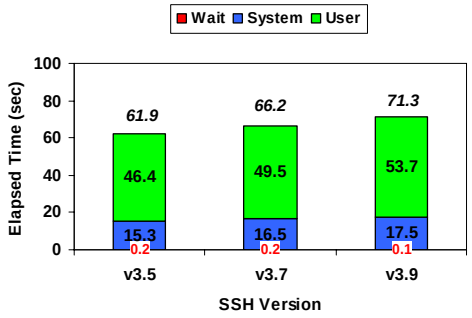
PostMark

- ☒ Does not scale (single thread)
- ☒ Default workload is no longer relevant: changes to configuration are necessary
- ☒ Times computationally-intensive code
- ☒ Does not have accurate timing
- ☑ Uses built-in pseudo random number generator (v1.5)

Compile Benchmarks

- Usually kernel, OpenSSH, Emacs, etc.
- ☒ Different packages do not produce the same workload
  - Not comparable between papers
- ☒ Same package does not produce the same workload
  - Not reproducible
- ☒ Difficult to understand
- ☒ Not scalable
- ☒ CPU intensive

OpenSSH Compile (make)



The Andrew File System Benchmark (1988)

- Synthetic workload ("common user workload").
- Operates on a source code directory
- Has five phases:
  - MakeDir: create a copy of the directory structure
  - Copy: copy the files to new location
  - ScanDir: `stat` every file
  - ReadAll: read all of the files
  - Make: compile and link the program

The Andrew File System Benchmark

- ☒ 'Make' phase dominates run time
  - All compile benchmark drawbacks
- ☒ It does not scale
  - The default data set will fit into the buffer cache of most systems
- ☒ Some used other source programs
  - Makes results incomparable
- ☒ Modified Andrew Benchmark (1990)
  - Uses same compiler for all machines

The Transaction Processing Performance Council (TPC)

- Non-profit corporation founded in 1988
- Creates standard transaction processing and database benchmarks
- Maintains several benchmarks
  - New versions released
  - Obsoletes older versions
- The benchmarks execute various types of transactions against a database

The Transaction Processing Performance Council (TPC)

- ☑ Council of database professionals
  - Adds credibility
  - Keeps benchmarks up to date
- ☒ Narrow representation of file system workloads
  - Many databases today run on top of F/S
  - Heavy write-load, random r/w
- ☒ Database adds extra complexity
  - Results are incomparable

SPEC SFS 3.0 (1997)

- SPEC: The Standard Performance Evaluation Corporation
- Creates and maintains standardized benchmarks
- SFS
  - Measures NFS server performance
  - Only SPEC benchmark that measures file system performance

SPEC SFS

- ☑ Reduces dependence on clients by crafting RPC packets from user-space
- ☑ Scales well
  - ☑ Multiple clients to one server
- ☑ Reports more than one number
- ☑ Is updated and maintained by SPEC
  - ☑ SPEC 4.0 (2006?)
- ☒ Has narrow scope
- ☒ Doesn't test NFS clients
- ☒ NFSv4? (FileBench)

Outline

- Motivation
- **Survey: Benchmarks Today**
  - Macro-benchmarks
  - **Traces**
  - Micro-benchmarks
  - Workload Generators
- Tools
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

Traces

- Logs of operations that are collected, and can later be replayed to generate the same workload
- Types:
  - Trace of another benchmark
    - TPC (esp. TPC-W)
  - Trace of some users performing normal tasks in some environment
    - Activity seen on a server

Traces (Cont.)

- Capturing:
  - No standard way to capture traces
  - Information and operations present in a trace depend on the capture method
  - The capture method is often unspecified
- Replaying
  - Timing: original speed, fastest?
  - Possible solution: normalize & calibrate

Traces (Cont.)

- How realistic is the trace?
  - Traces may become stale
  - Large sample sizes are needed
- Availability of traces
  - Re-using traces encouraged
  - … but, hard to get (large size)
  - …and, traces lost, people move on, etc.
  - Solution: authoritative trace repository
    - See www.snia.org working group (IOTTA)

Outline

- Motivation
- **Survey: Benchmarks Today**
  - Macro-benchmarks
  - Traces
  - **Micro-benchmarks**
  - Workload Generators
- Tools
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

Micro-Benchmarks

- Small number of operation types
- Highlight some aspect of the file system
- Three types:
  - De Facto Standard
    - Bonnie / Bonnie++
    - Sprite LFS benchmarks
  - System utilities
  - Ad-hoc

Bonnie

- Tests single file:
  - Writes file, reads file, random seek/rd/wr

☒ Uses the machine's pseudo-random number generator

☒ Options are hard-coded (only file size can be set from command line)

☒ One character reads/writes test buffered libc I/O, not the file system

☒ Does not scale

System Utilities

- Some have used standard utilities to measure performance (e.g., **`wc`**, **`grep`**, **`cp`**, **`diff`**, **`tar`**, **`gzip`**)

☑ They are more standardized than ad-hoc micro-benchmarks

☒ None of the papers specified the version of the program

☒ They do not scale

Ad-Hoc Micro-Benchmarks

- Written for a specific research paper
  - Not standardized, not reproducible, not comparable
  - Often inefficient
- Acceptable uses:
  - With macro-benchmarks and/or traces
  - To highlight unique aspects of the file system
- Unacceptable uses:
  - Only using micro-benchmarks to measure performance of a large project
  - Using them without any explanation of why they were used
- 33/68 papers, 119 ad-hoc B/Ms total

Outline

- Motivation
- **Survey: Benchmarks Today**
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - **Workload Generators**
- Tools
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

Workload Generators

- Iometer
  - ☑ Allows benchmark to reach steady state
  - ☒ Not enough customization
  - ☒ Linux and Windows versions vary
- Buttress
  - ☑ High accuracy
  - ☑ Event-based programming interface
  - ☒ More complex to specify exact workload
  - ☒ Captures syscalls, replays I/O operations
- FileBench: the future
  - ☑ Generate macro- and micro-benchmark workloads
  - ☑ Configure benchmarks with scripts
  - ☒ Alpha/Beta status (sourceforge), maturity, acceptance

Outline

- Motivation
- Survey: Benchmarks Today
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - Workload Generators
- **Tools**
  - **Auto-pilot [Usenix/Freenix '05]**
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

Benchmarking is Hard

- Iterative process
  - Bugs
  - Inefficient code
  - Unexplained results
- Accuracy
  - Reproducible
  - Stable
  - Fair
- Presentation
  - Need to understand the results

Our Approach

- Iterative process
  - Automate as much as possible
  - Extensibility
- Accuracy
  - Record everything
  - Reproduce machine state
- Presentation
  - Tabular reports
  - Graphs
  - Statistical analysis

Auto-pilot [Freenix 2005]

- A framework for running benchmarks
  - Not a suite of benchmarks or metric
- Language to define benchmarks
- Sample scripts to run file-system benchmarks
- Tools to analyze the results

Features

- Cold Caches
  - Read large file to flush cache
  - Unmount file system
  - `chill`
  - Checkpoint benchmark state and reboot
    - pre-test warmup phase?
- Scripts to execute and measure benchmarks
  - Samples for ext2/3, reiserfs, etc.
  - Postmark
  - Compile tests
  - Easy to add new scripts
  - Pre/post setup scripts
    - Stop/start system services (e.g., cron)

Features: Hooks

- Record free memory, network utilization, I/O operations, background CPU time
- Compilation command
- File system hooks
  - Mount, unmount, mkfs, tunefs
  - Stackable file systems hooks
  - NFS hooks including measurement of remote nfsd CPU time

Features: Getstats

- Input: Results logs, CSV, GNU time
  - Modular parser architecture provides for extension
- Output: Tabular reports, CSV, and more
- After parsing applies *transformations:*
  - Aggregates threads
  - Unifies commands
  - Adds Wait and CPU%
  - Computes overhead

Hypothesis Testing

- Two-sample t-test
- Assume *null* hypothesis
  - First configuration is faster than the second
  - First configuration is slower than the second
  - First configuration is equal to the second
- Compute the probability of observing the data given the null hypothesis
  - P-value
- If P-value is below a pre-defined significance level (e.g., 5%), then reject the null hypothesis

Detecting Anomalous Results

- Display half-widths and standard deviations
- Highlight outlying values
  - Display individual results with high z-scores
- Linear regression
  - Measured quantities should not have a trend as execution proceed
  - Detects memory leaks
    - Elapsed time has a positive slope
    - Free memory has a negative slope

Features (Cont.)

- Predicates
  ("$delta" < 0.05 * $mean) || ($count > 30)
- Inform when benchmark terminated
- Graphit
  - Input: Result logs, Getstats output, or CSV
  - Output: Bar and line Graphs
  - Uses Gnuplot as a backend
  - Automatically converts data into suitable format for plotting
- Status
  - 70 page manual
  - Released under GPL
  - Actively maintained (lists, bugzilla, etc.)
  - http://www.filesystems.org/project-autopilot.html

Outline

- Motivation
- Survey: Benchmarks Today
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - Workload Generators
- **Tools**
  - Auto-pilot [Usenix/Freenix '05]
  - **Tracefs [FAST '04]**
  - Replayfs [FAST '05]
  - FSprof
- Conclusions and Future Work

Tracefs Summary

- Stackable file system
  - Transparently trace any other file system
- Low overhead
  - Async trace-output writer
  - Reduce further by selecting what to trace
- Modular architecture (plugins)
- Transformation plugins
  - Encryption, compression, checksumming, anonymization, …
- Portable trace format
  - Records anything useful about system state
- Output filters: local file, remote socket, etc.

Outline

- Motivation
- Survey: Benchmarks Today
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - Workload Generators
- **Tools**
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - **Replayfs [FAST '05]**
  - FSprof
- Conclusions and Future Work

Replayfs Summary

- Trace compiler
  - Convert portable trace for OS-specific replaying
- Optimizations: "productive" pre-spin
  - flush objects/data, pre-fetch, zero-copy, etc.
  - No context switches needed
- Efficiency
  - <2% memory consumption
  - 32% faster replaying
    - 2.5x faster if skip copy-to-user

Outline

- Motivation
- Survey: Benchmarks Today
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - Workload Generators
- **Tools**
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - **FSprof**
- Conclusions and Future Work

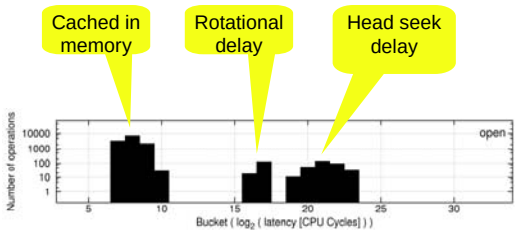
FSprof: Motivation

- What goes on inside a file system?
  - Where are the latencies?
- User/system/elapsed time too coarse
- User-level tools inaccurate
- Syscalls miss mmap ops
  - … and cannot be used on file servers
- Kernel profilers focused on CPU usage
  - …but file systems have significant I/O!

FSprof: Solution

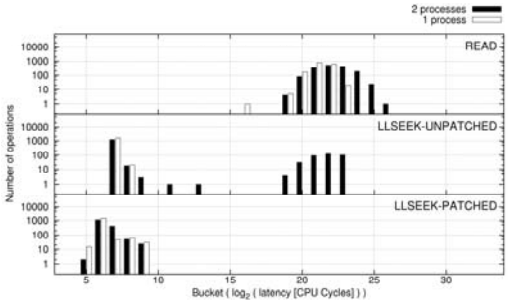
- Automatically instrument f/s to measure each operation's latency
  - No source: stackable f/s
- Automatically instrument SCSI/IDE drivers
- Use TSC register for accurate timing
- Record operation times in exponential buckets
  - Powers of two
- Efficient
  - <4% O/H CPU time (Postmark)

Tri-Modal Behavior



Linux 2.4.24, ext2, `grep -r` on source tree

Lock Contention



2.6.11, two processes reading same file randomly

Outline

- Motivation
- Survey: Benchmarks Today
  - Macro-benchmarks
  - Traces
  - Micro-benchmarks
  - Workload Generators
- Tools
  - Auto-pilot [Usenix/Freenix '05]
  - Tracefs [FAST '04]
  - Replayfs [FAST '05]
  - FSprof
- **Conclusions and Future Work**

Summary

- Improve file system benchmarking
  - Researchers
  - Reviewers, PCs, shepherds
- Need standards that evolve
  - TPC model
  - FileBench
- Need tools to help:
  - FileBench, Auto-pilot, tracefs/replayfs, FSprof,, and more
- Need centralized trace repository
  - FAST 2005 BoF @ 9pm
- You can help!

Future Work

- Normalizing traces for replaying on newer (faster) machines
- Predictive performance via micro-benchmarks and hardware modeling
  - Virtual replaying

Questions

File System Benchmarking:
Fallacies, Pitfalls, and Beyond

Erez Zadok, Nikolai Joukov,
Avishay Traeger, and Charles P. Wright
Stony Brook University

http://www.fsl.cs.sunysb.edu/
Tech-Report FSL-05-04