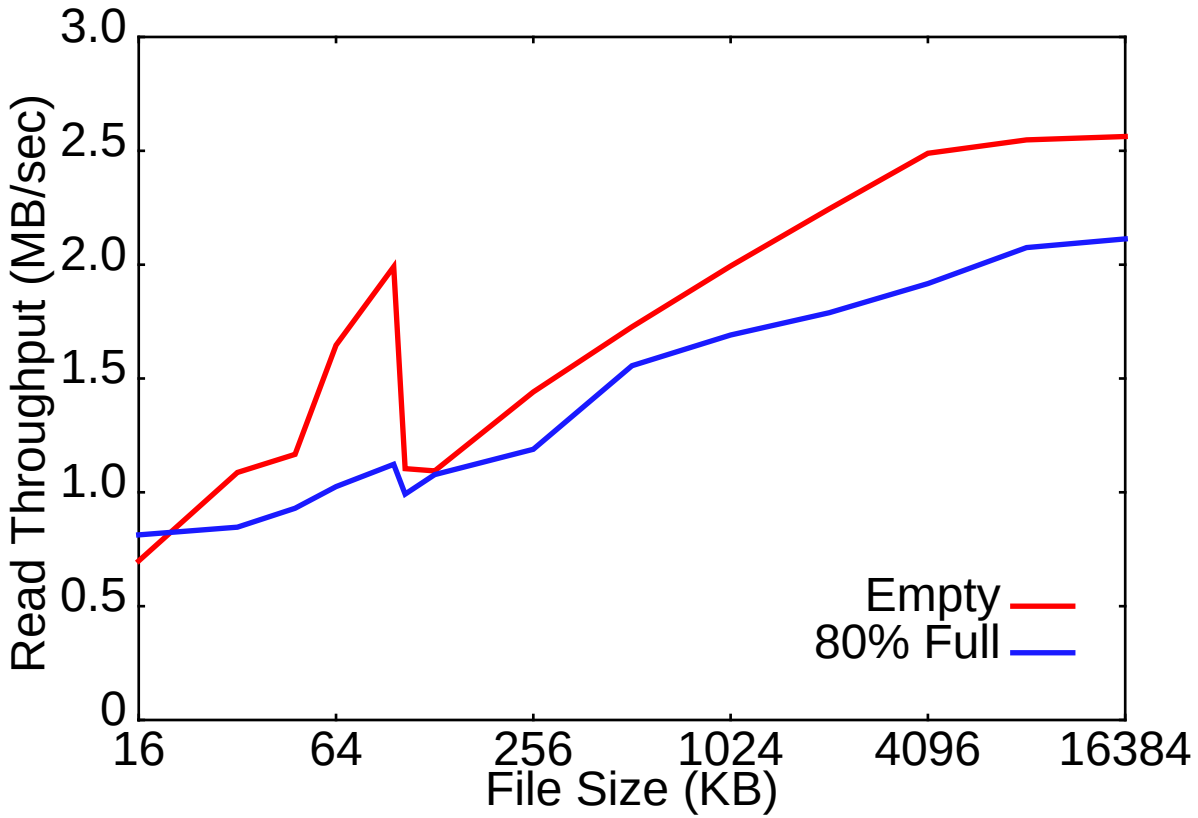
File System Aging: Increasing the Relevance of File System Benchmarks

Keith A. Smith
Margo I. Seltzer

Harvard University
Division of Engineering and Applied Sciences

File System Performance



Problem #1

- **Full and empty file systems perform differently.**
- **Most research uses empty file systems.**
- **Real world file systems are never empty.**

Don’t benchmark empty file systems!

Problem #2

- **Just filling a file system isn’t enough.**
- **The history of a file system determines its state.**
- **Design decisions may affect how state evolves over time.**
- **Most research uses empty file systems.**
- **Researchers ignore a large area of design space.**

Don’t benchmark empty file systems!

Our Solution

- **Use simulated workload to *age* file system.**

Overview

- **Problem**
- **File system aging**
  - Creating the workload
  - Verifying the workload
- **Example**
- **Conclusions**

File System Aging—Goals

- Examine state of file system after many months of activity.
- Support different workloads.
- Allow reproducibility.
- Be architecture independent.
- Make easy to use.

File System Aging—Method

- **Use real file system usage patterns to generate artificial *aging workload.***
  - Aging workload is sequence of file create, write, and delete operations.
- **Different workloads mimic different usage patterns.**
- **Reproducibility provided by reusing same workload.**
- **Workload parameterized in terms of POSIX interface.**

Source for Aging Workload

- **Long term trace was impractical.**
- **Data we had available:**

  1. Unix file system snapshots
     - Describes all files on file system.
     - Daily for one year
  2. NFS traces
     - All NFS requests to large file server.
     - Continuous for two weeks.

Generating Aging Workload

1. **Start with sequence of snapshots.**
2. **Populate file system.**
   - Create files present in first snapshot.
3. **Add inter-day file activity.**
   - Compare successive snapshots.
   - Identify created and deleted files.
   - Add corresponding create, write, and delete operations.

Generating Aging Workload

4. Add intra-day file activity.

- Use NFS traces to model short-lived file activity.
- Intersperse create, write, and delete operations based on model.

Sample Workload

- **Aging Workload:**
  - Seven months of activity
  - 1 GB file system
  - ~1.3 million file operations
  - Writes 87.3 GB to disk
  - Typical run time is 39 hours.

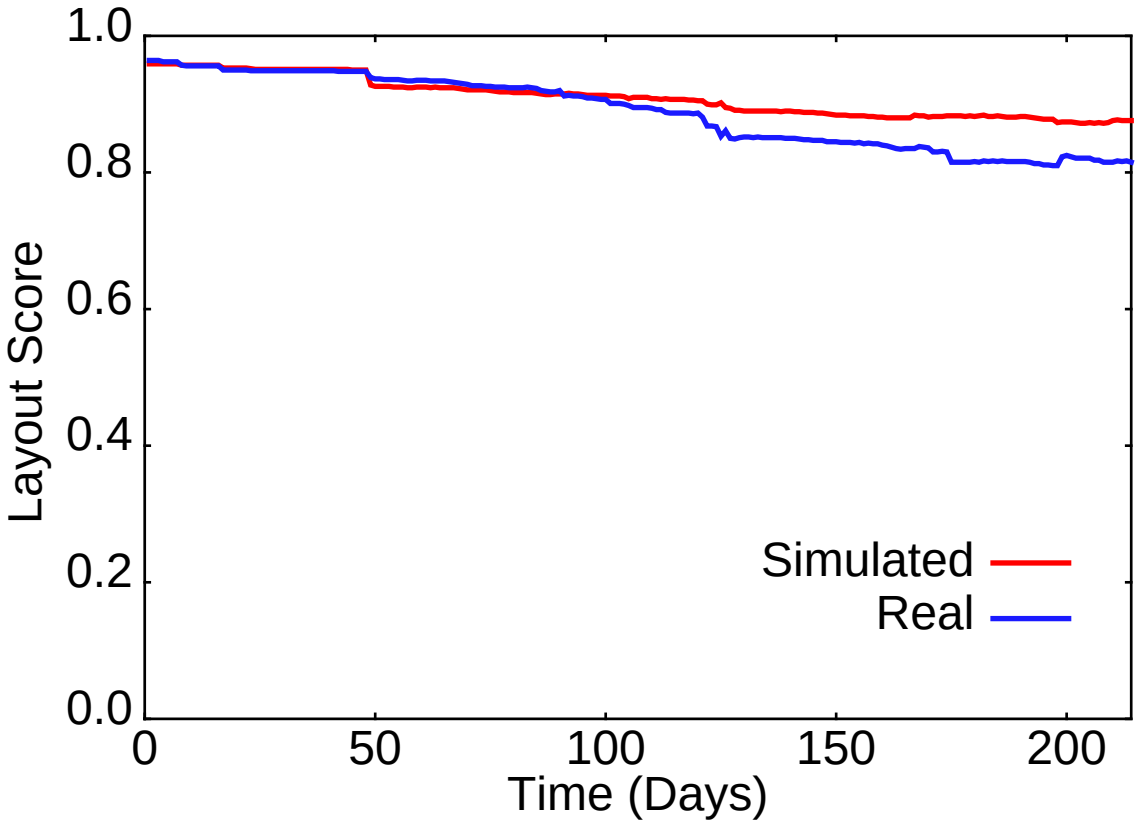
Verifying Workload

- **Start with empty file system.**
- **Age file system using workload.**
  - Execute file operations from workload on the test file system.
- **Compare file fragmentation on aged file system to last snapshot of file system from which workload was generated.**

Verification Metric

- **Layout Score**
  - Measures quality of file layout
  - Range: 0.0 – 1.0
  - Inversely proportional to file fragmentation
  - Score is percentage of file system blocks that are contiguous
  - 1.0 => All files are contiguously allocated
  - 0.0 => No contiguous allocation

Aging Verification



Example

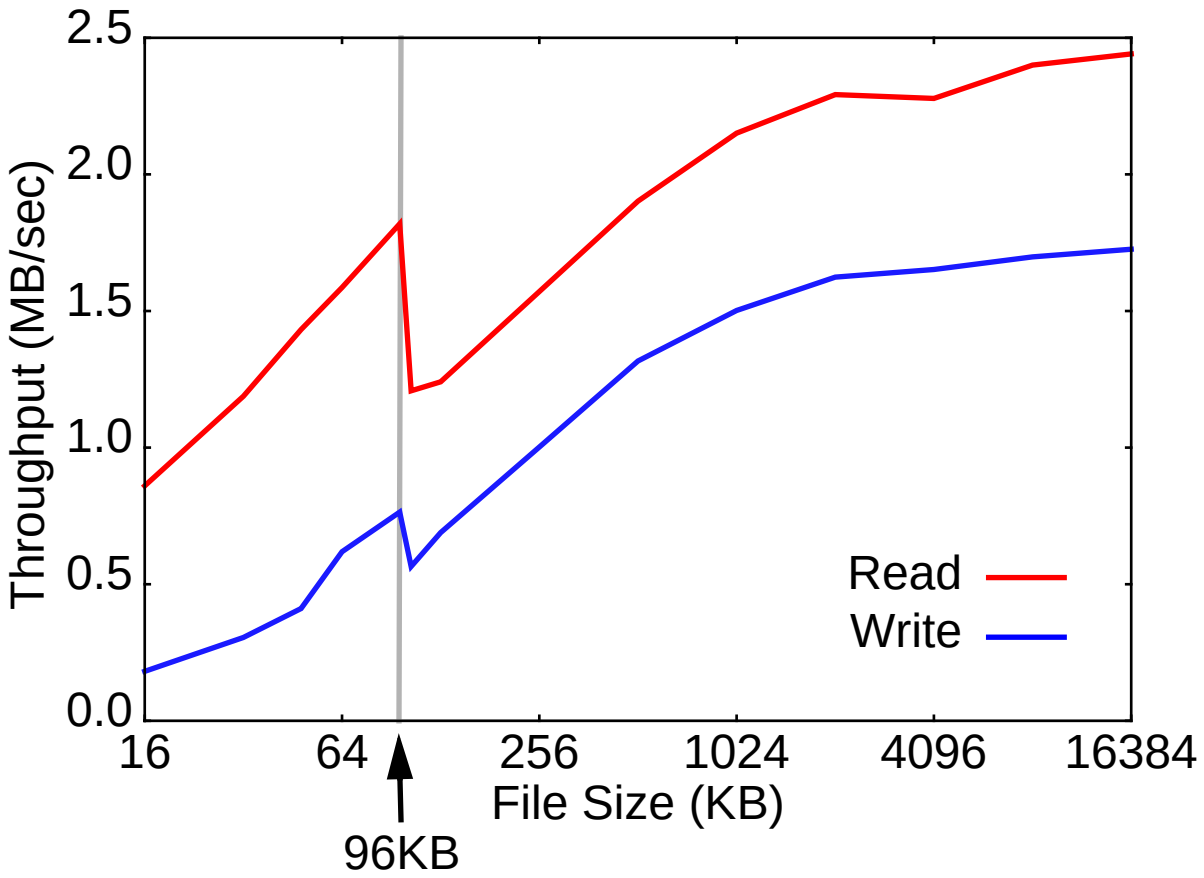
- **Modification to UNIX file system (FFS)**
- **Use aging to evaluate performance trade-offs.**

Test Platform

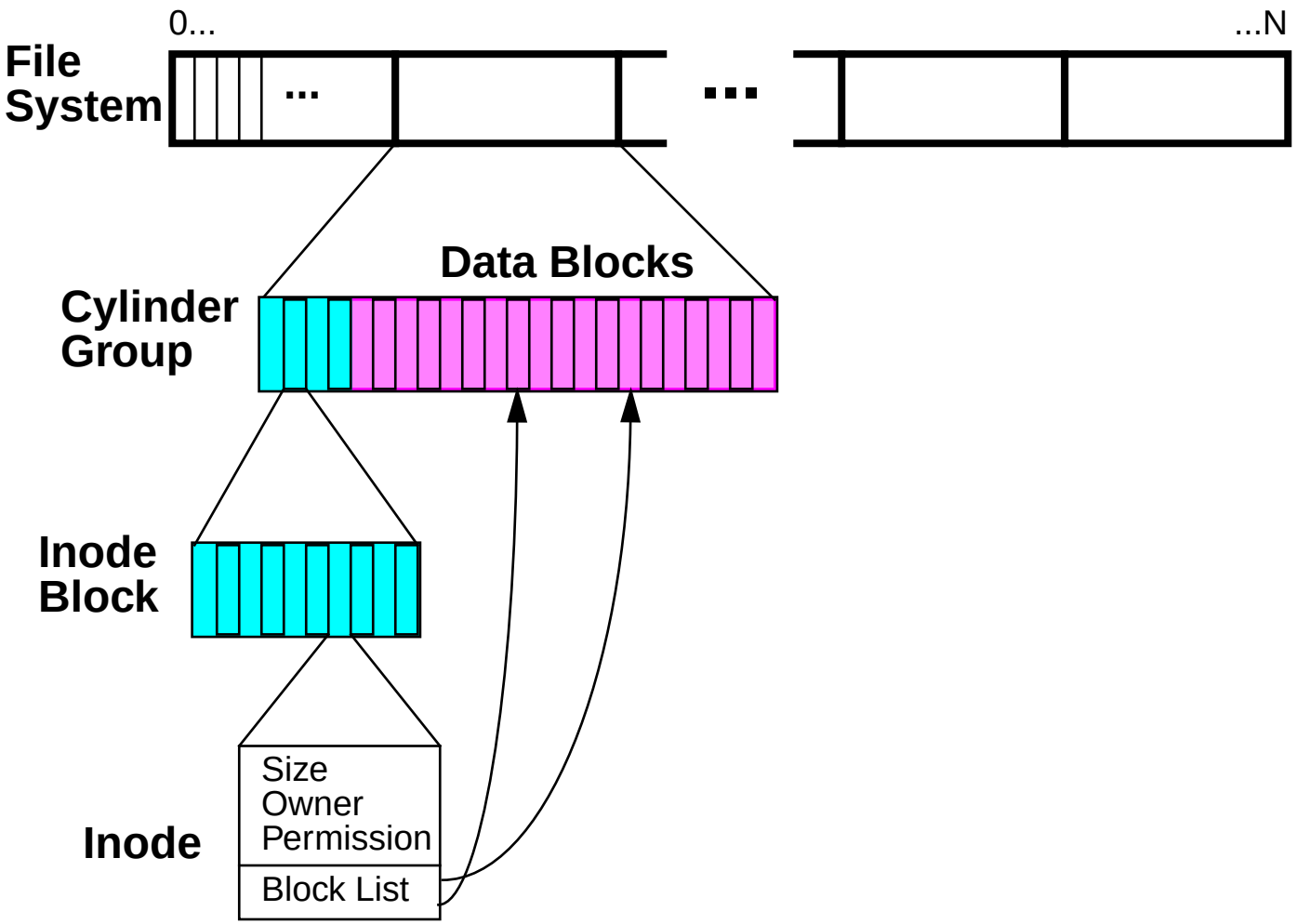
- **200 MHz Pentium Pro**
- **32 MB RAM**
- **PCI Bus**
- **NCR 53c825 SCSI controller**
- **Fujitsu M2694ES disk**
  - 1 GB, 5400 RPM, 15 Heads, 94 Sect./ Track (avg.), 1818 Cyl. 9.5 ms Avg. Seek
- **BSD/OS 2.1**
- **8 KB file system block size**
- ***maxcontig* = 7 blocks (56 KB)**

Baseline FFS Performance

(Aged file system)



The UNIX File System (FFS)



Cylinder Groups

- **Cylinder groups are allocation pools.**
- **They exploit locality of reference.**
- **Related data are collocated in same cylinder group.**
  - All files in a directory
  - Sequential blocks of a file

File Allocation

- First 12 file data blocks are allocated from same cylinder group as the file's directory.
- The 13th and subsequent blocks are allocated in a different cylinder group.
- All files have a large seek between 12th and 13th block.
- 12 blocks = 96 KB

Solution

- *NoSwitch* **file system**
- **Don’t switch cylinder groups after the 12th file block.**

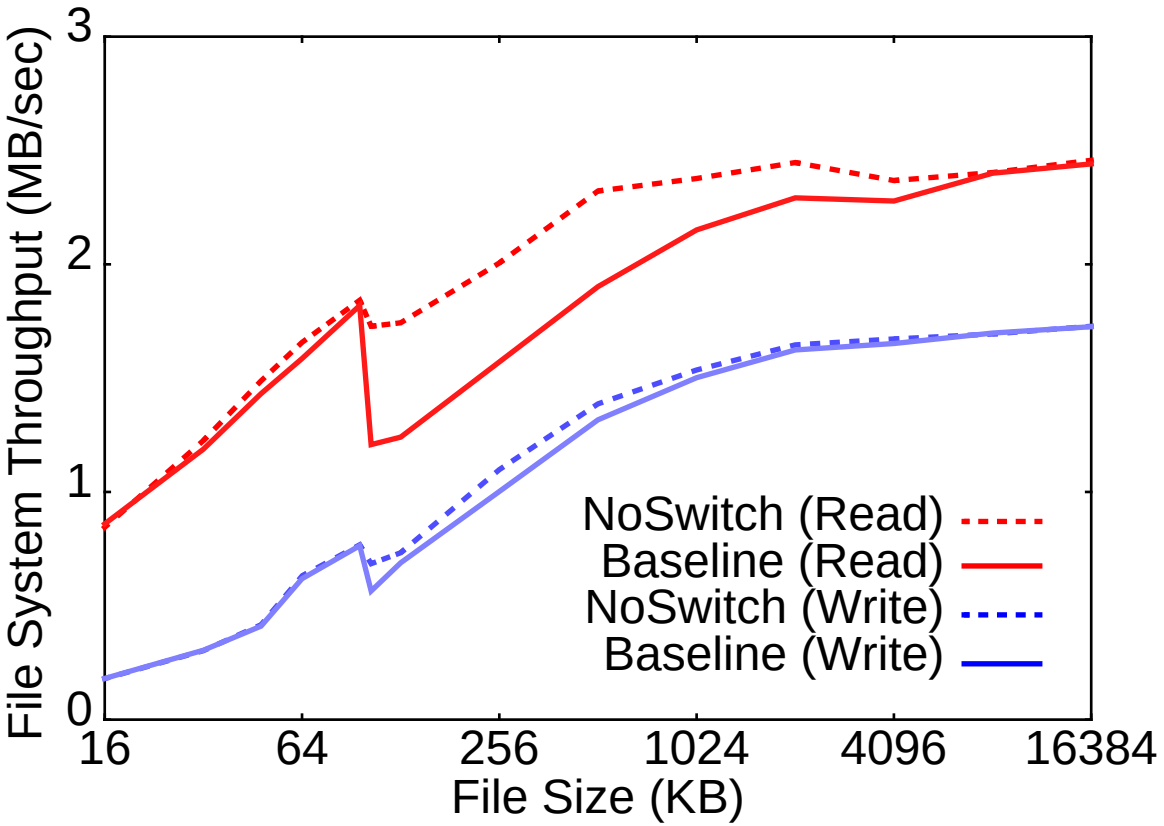
Potential Problem

- **Too many large files in one directory would cause cylinder group to run out of space.**
  - Creates *split files.*
  - Files in different cylinder group than their directory.
  - Extra seek to get from directory to file.
- **But does this happen?**
- **If so, how does it affect performance?**

Evaluation of NoSwitch

- **Age two file systems, one that switches cylinder groups, and one that doesn't**
- **Compare the resulting file systems**
  - Overall performance
  - Number of *split files.*

Performance



Number of Split Files

	Baseline	NoSwitch
Number of Files	33,797	33,797
Number of Split Files	4,312	9,155
Percentage of Split Files	13%	27%

Hot File Benchmark

- **Measure performance using files from aging workload**
  - Files modified during final 30 days
  - 92 MB (14.5% of allocated storage)
  - 3,207 files (9.5% of files)
  - 119 files large enough to benefit from NoSwitch
- **Two phase benchmark:**

  1. Read entire file set
  2. Overwrite entire file set

Hot File Performance

	Baseline	NoSwitch
Layout Score	0.928	0.931
Number of Split Files	327	594
Read Throughput	0.81 MB/sec	0.84 MB/sec
Write Throughput	0.49 MB/sec	0.50 MB/sec

Analysis

- **NoSwitch file system improves performance of medium and large files.**
- **NoSwitch file system increases the number of split files.**
- **Net effect is small performance improvement.**
- **Exact trade-off depends on workload!**

Conclusions

- **Benchmarking empty file systems is unrealistic.**
- **Benchmarking empty file systems can be misleading.**
- **File system aging is a technique for increasing the relevance of file system benchmarking.**

Don’t benchmark empty file systems!

File System Aging: Increasing the Relevance of File System Benchmarks

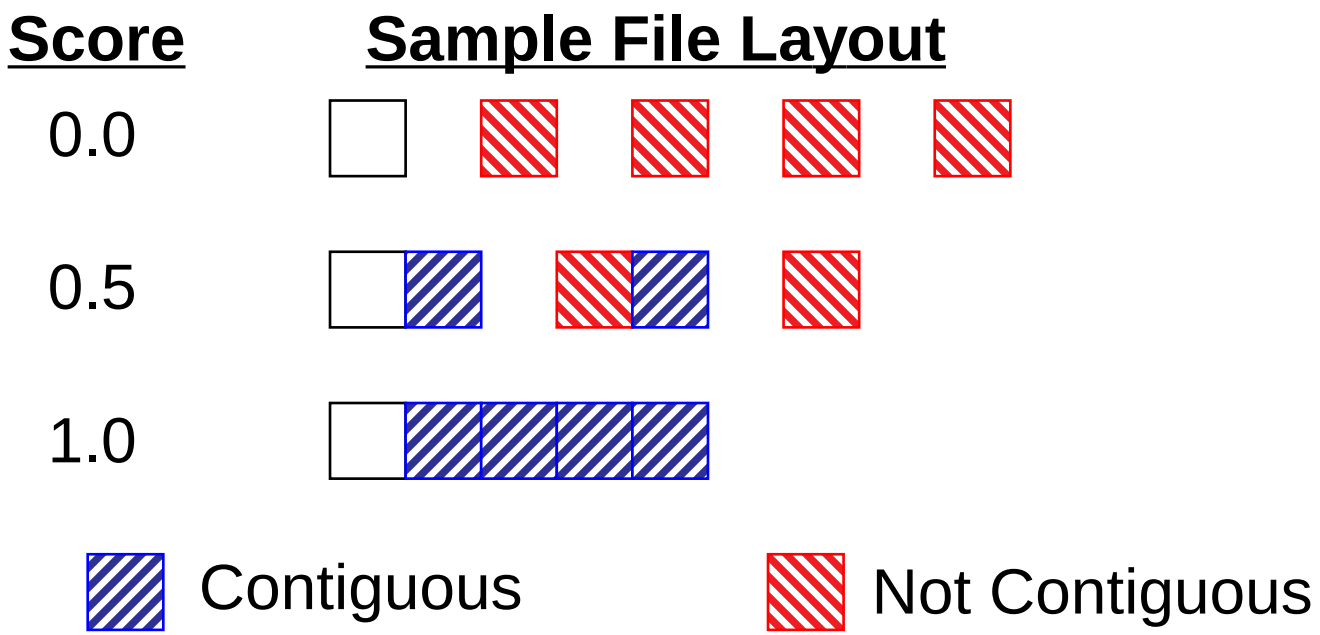
Keith A. Smith
Margo I. Seltzer

keith@eecs.harvard.edu
margo@eecs.harvard.edu

http://www.eecs.harvard.edu/~keith/sigmetrics97

Fragmentation Metric

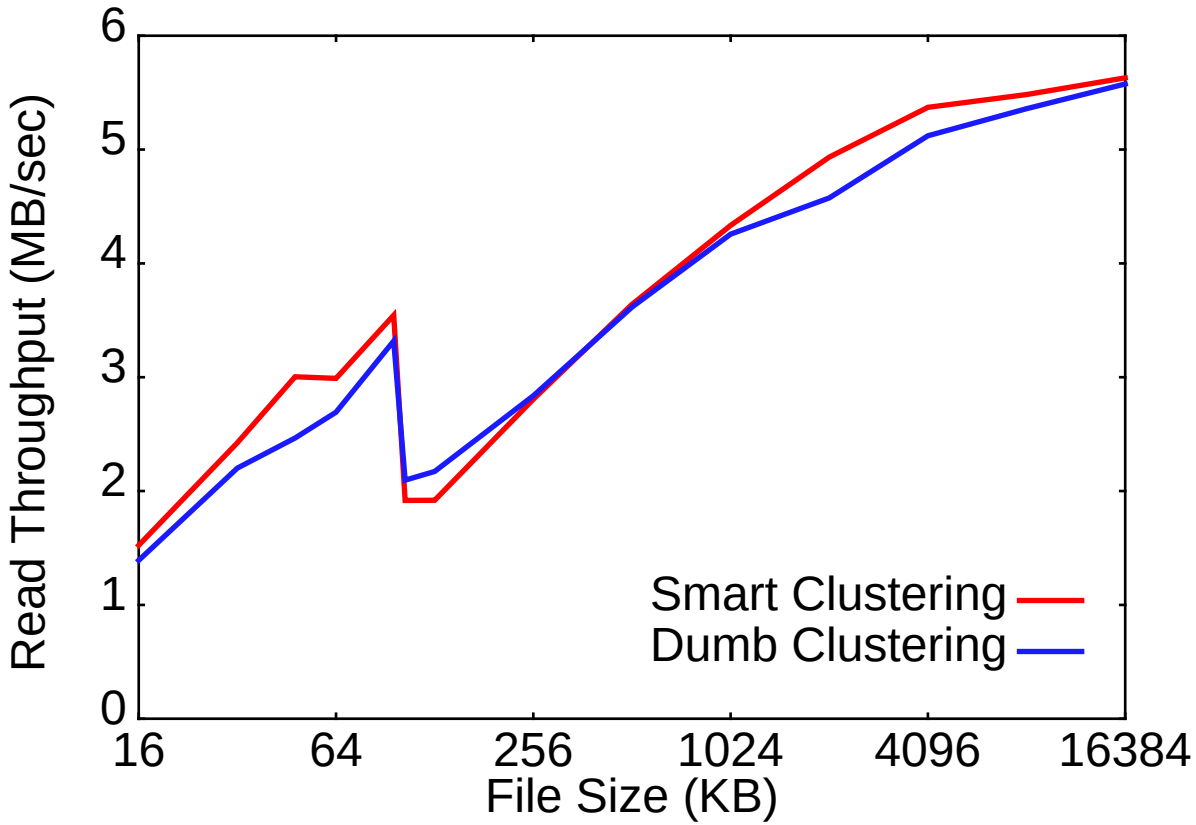
- ***Layout Score* measures fragmentation**
  - Fraction of blocks that are contiguous
  - Ignores first block of a file.



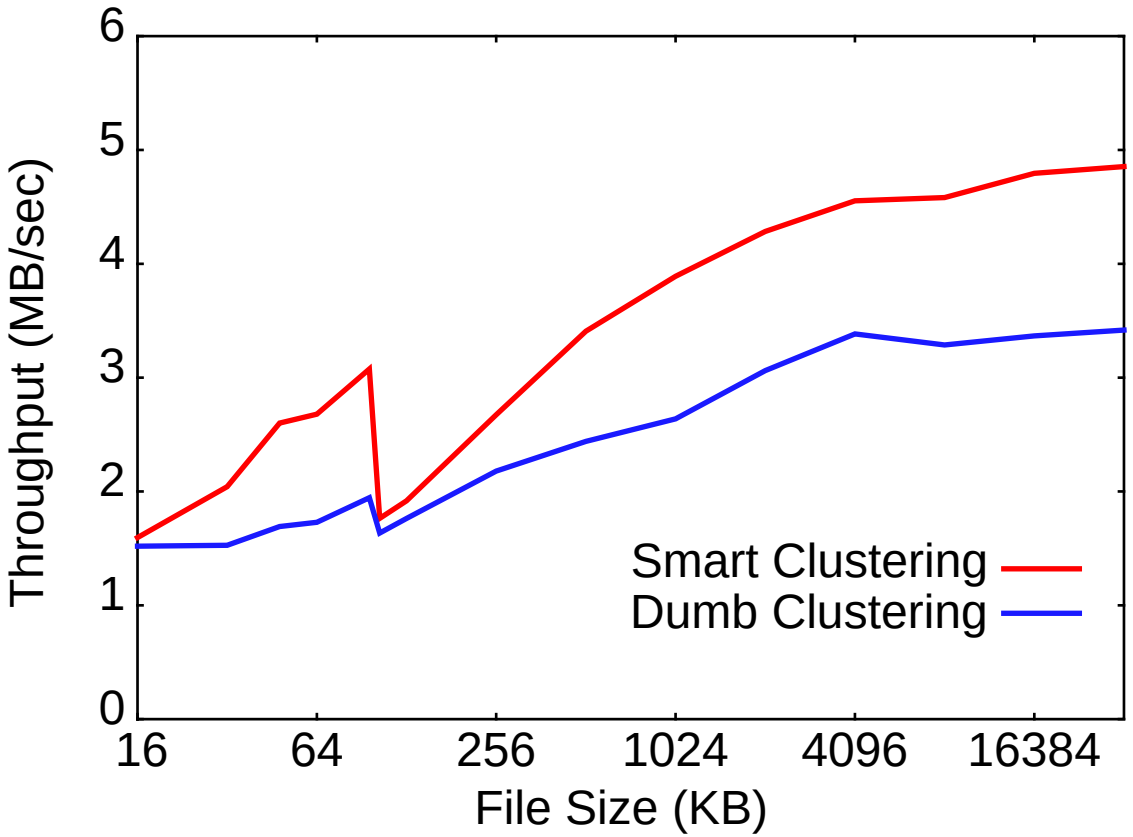
Sequential I/O Benchmark

- **32 MB data set**
- **Uniform file size (16 – 16,384 KB)**
- **25 files per directory**
- **Two Phases**
  - *Create Phase:* Create and write all files
  - *Read Phase:* Read all files

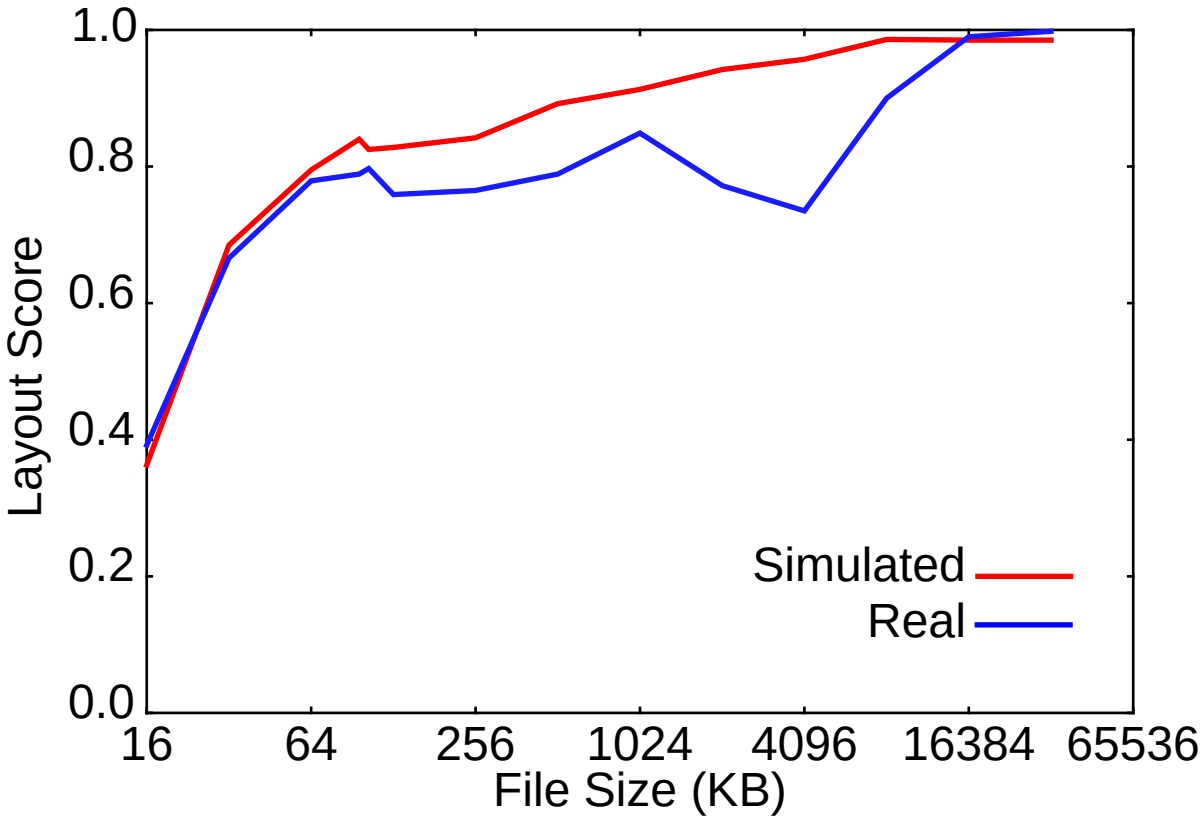
Comparison (empty)



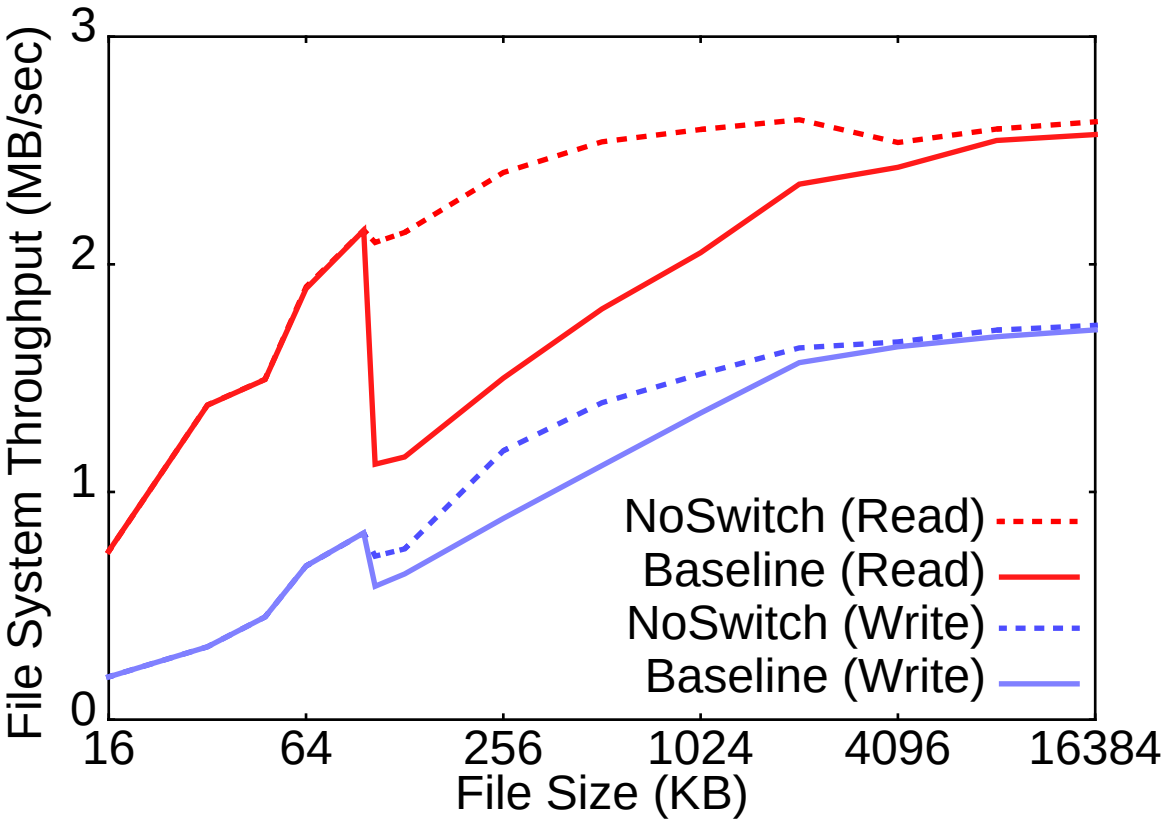
Comparison (aged)



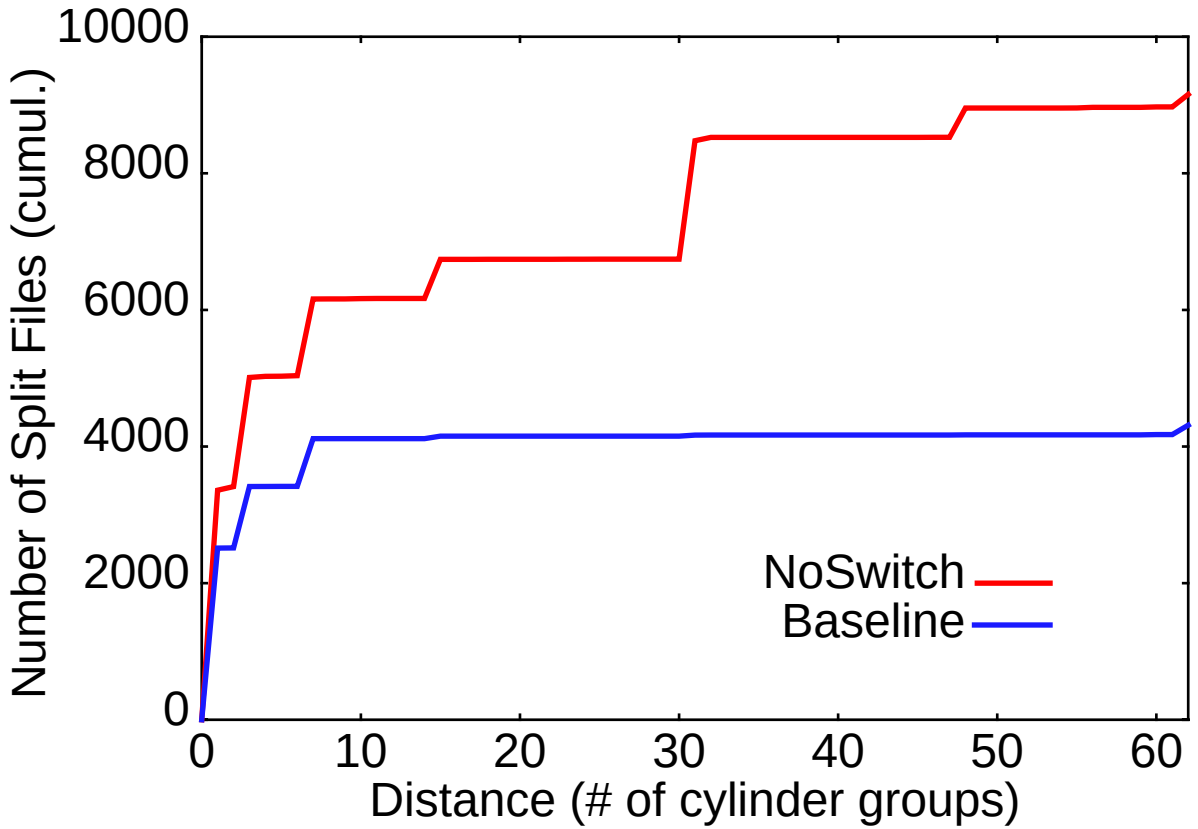
Aging Verification



Performance (empty)



Seek Distances in Split Files



Future Work

- **Improve aging algorithm**
- **Expand to cover more workloads.**
- **Parameterize for amount of aging or size of file system.**