File Append vs. Overwrite in a Sprite Cluster[1]

John H. Hartman

jhh@sprite.berkeley.edu

Sprite Project
University of California at Berkeley
January 21, 1992

1. The results in this talk were derived from the same set of file system traces used in "Measurements of a Distributed File System" by Mary G. Baker et al., in *Proceedings of the 13th Symposium on Operating System Principles*, Oct 1991. Better descriptions of the workload, environment, and tracing method can be found there.

Executive Summary

Environment

- Unix-like operating system
- Network file system (diskless clients)

Workload

- University environment
- OS development, architecture research, VLSI design, computer graphics research

Results

Fraction of bytes written

- Append: 96%
- Sequential overwrite: 1%
- Random overwrite: 3%

Environment

Hardware: network of workstations

- 40 10-MIPS diskless client workstations
- 4 file servers, 9 GBytes disk space
- Primary server was a Sun-4/280 with 5 GBytes of storage
- Single Ethernet network

Software: Sprite

- Unix compatible
- High-performance network file system
- File caches on both clients and servers
- Process migration

Workload

- University environment
- No restrictions on users
- 40 regular users, 40 occasional users
- OS researchers
- Architecture researchers
- VLSI circuit designers
- Graphics researchers
- Administrators

How we got the data

- Traced user-level file activity

  Logical file operations

  - Roughly equivalent to system calls
  - Results not affected by file cache
  - Results not affected by client/server protocol

  Required kernel modifications

  Collected on file servers

- 8 trace intervals each 24 hours long
- Trace intervals spread out over several months
- Lots of post-processing

Metrics

- % of bytes written that are appended to files
- % of bytes written that sequentially overwrite existing files
- % of bytes written that randomly overwrite existing files

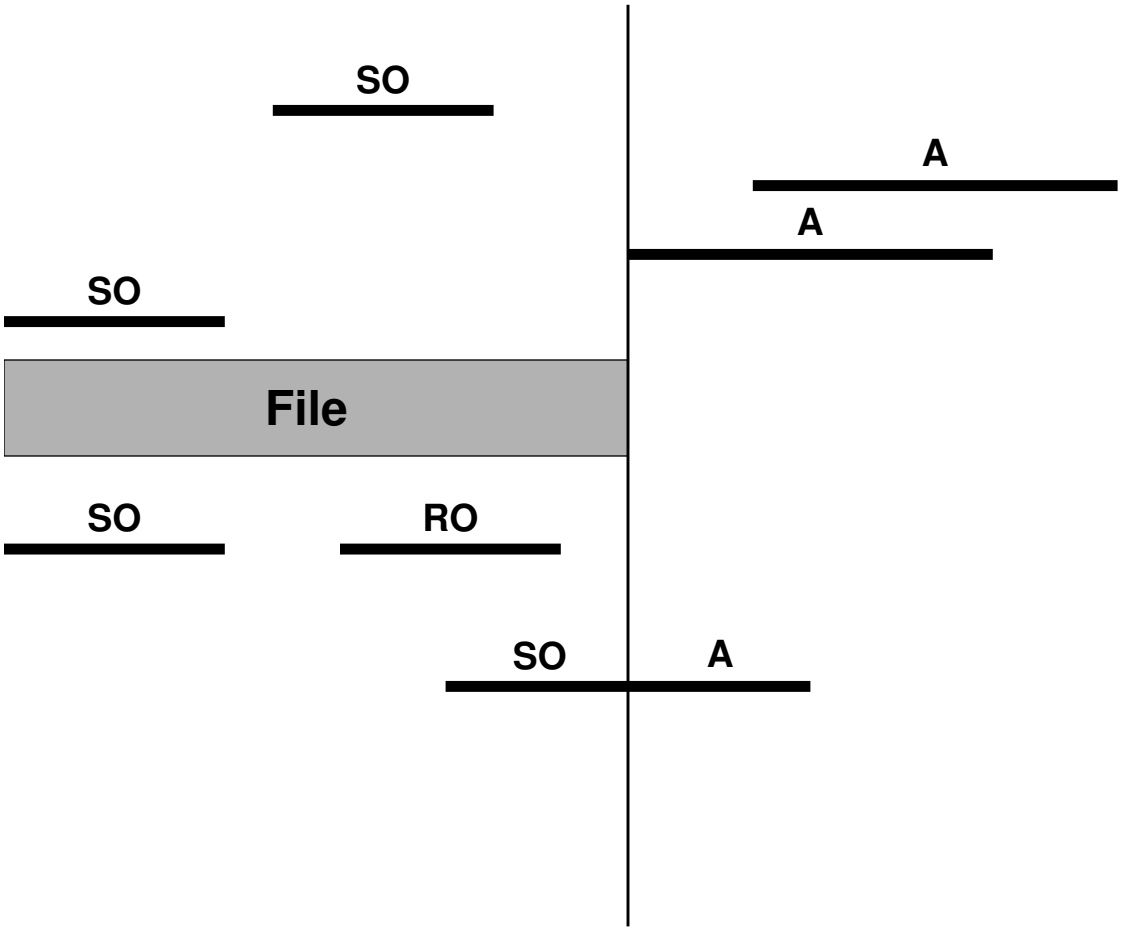
Caveats

- Bytes that fill a hole are considered to overwrite the file.
- Sequential vs. random is determined on a per-file basis.
- Overwrite of first byte is always sequential.
- First overwrite is always sequential.

Algorithm

```
if (start < fileSize)
   if ((start == previousEnd) ||
      (previousEnd == 0) ||
      (start == 0))
      type = sequentialOverwrite
   else
      type = randomOverwrite
   if (end <= fileSize)
      bytes = end - start
   else
      bytes = fileSize - start
      bytesAppended = end - fileSize
else
   bytesAppended = end - start
```

Overwrite vs. Append



Results

Trace	Append		Sequential Overwrite		Random Overwrite	
	MBytes	%	MBytes	%	MBytes	%
1	271	91.12	4	1.34	22	7.54
2	342	86.45	26	6.59	28	6.96
3	4564	98.73	13	0.29	46	0.98
4	1166	93.92	17	1.39	58	4.68
5	248	92.15	5	1.76	16	6.08
6	332	91.06	9	2.34	24	6.60
7	461	91.30	3	0.55	41	8.15
8	579	92.07	3	0.55	46	7.39
Totals	7964	95.66	80	0.96	282	3.38