File Systems Comparison: EXT4, ZFS, NTFS

Aribhit Mishra, Mickey Barboi and Mingi Kim

{aribhitm, barboi, mingi}@cs.wisc.edu

UNIVERSITY OF WISCONSIN–MADISON

Motivation

- Compare three of the most-widely used modern file systems : EXT4, ZFS and NTFS
- Lack of existing work that compares all three file systems
- Different features and implementations would mean varying performance over different loads

File system Comparison

Three different file systems provides different structures and features.

	EXT4	ZFS	NTFS
Max Filename Length	255 bytes	255 characters	255 characters
Max File Size	16 GiB to 16 TiB	16 EiB	16 EiB
Max Volume Size	1 EiB	256 ZiB	16 EiB
Checksum	×	✓	×
Block Journalling	✓	✓	×
File Changelog	×	×	✓
Internal Snapshotting	×	✓	Partial
Data Deduplication	×	✓	✓
Allocate-on-flush	✓	✓	×
Variabe Block Size	×	✓	×

Design

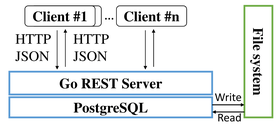
1. Macrobenchmark
   a) Serving images through Apache Server
      i. Serve image from static directory to n clients
      ii. Clients issue sequential requests with no intercommunication amongst clients
      iii. Images removed from drive, and recopied between tests
   b) Serving HTTP requests with Go and PostgreSQL
      i. Client issues HTTP requests trying with a JSON payload
      ii. Server creates a new record in database for each request
   c) Compiling Go
      - Download Go source code and compile it
2. Microbenchmark
   a) Single block size read/write operation test
   b) Large file I/O test
   c) Small files bulk I/O test

Test Environment

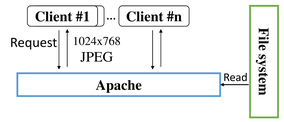
- Intel i7 6700k (8 threads at 4.2gHz, 8MB cache)
- 32GB DDR4 RAM
- System Drive : Samsung SM951 NVMe with 512GB of capacity
- Testing Drive : Western Digital 500GB HDD (7200 max RPM)
- Operating Systems:
  - NTFS run on Windows 10
  - EXT4 and ZFS run on Ubuntu 16.0.4

Benchmark Program Design at a Glance

1. Macrobenchmark

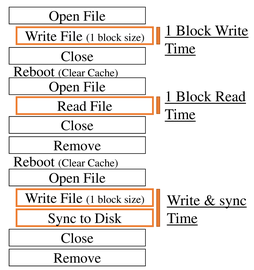


Serving HTTP requests with Go/PostgreSQL

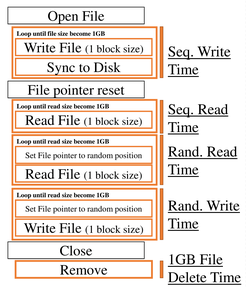


Serving images through Apache Server

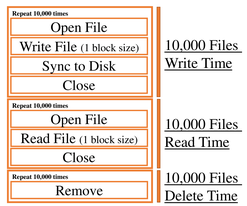
2. Microbenchmark



Single block size read/write operation test



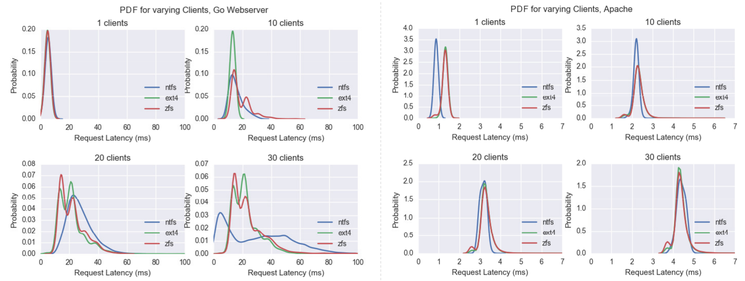
Large file I/O test



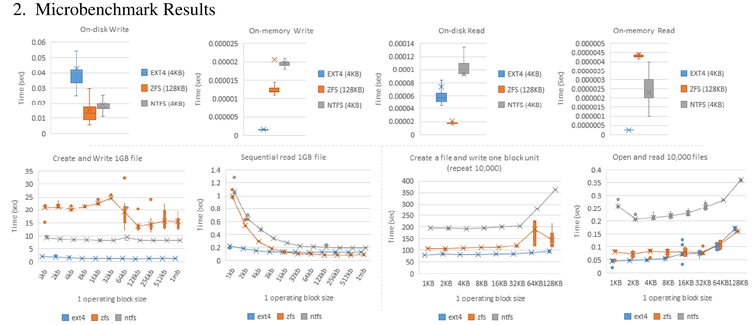
Small files bulk I/O test

Results

1. Macrobenchmark Results



2. Microbenchmark Results



Implementation

1. Macrobenchmark
   1. General Design
      1. n clients: [1, 10, 20, 30]
      2. Clients issue next request as soon as current returns
   2. Go REST HTTP Server
      1. HTTP POST with 52B JSON payload
      2. Clients do not share data
      3. Server
         1. Commits payload to database
         2. Reads it and returns to client
   3. Apache Webserver
      1. Requesting 1024x768 JPEG, 247KB
      2. Clients request unique images

2. Microbenchmark
   1. File I/O System calls
      - Used different system call libraries to see the performance in native operating system environments of each file system

	Open/Create	Read	Write	Move File Pointer
Linux	*open()*	*read()*	*write()*	*lseek()*
Windows	*CreateFile()*	*ReadFile()*	*WriteFile()*	*SetFilePointer()*

	Sync to Disk	Close	Remove
Linux	*fsync()*	*close()*	*unlink()*
Windows	*FlushFileBuffers()*	*CloseHandle()*	*DeleteFile()*

   2. Time measurement

	Get Time Function	Time Precision
Linux	*clock_gettime()* with *CLOCK_REALTIME*	25 nanosecond
Windows	*GetSystemTimePreciseAsFileTime()*	100 nanosecond

Conclusion

1. EXT4 slightly outperforms ZFS with NTFS lagging behind
2. EXT4 shows consistent performance across all metrics, while ZFS and NTFS have wider dispersion
3. ZFS outperforms when dealing with large files with at least 32KB access size in microbenchmark
4. Differences in Operating Systems and Application could have affected results
   - Apache, PostgreSQL and Go are native to Linux
   - Native home of ZFS is Solaris, not Linux

Source Code

1. Macrobenchmark
   - https://github.com/damouse/fs-bench-suite
2. Microbenchmark
   1. Single RW
      1. Linux : https://github.com/MingiK1m/cs736_rwbm
      2. Windows : https://github.com/MingiK1m/cs736_rwbm_windows
   2. Large/Small file(s) benchmark
      1. Linux : https://github.com/MingiK1m/cs736_fsbm
      2. Windows : https://github.com/MingiK1m/cs736_fsbm_windows