Application Based OS

Udit Jain

Dept. of Computer Science, IIT Delhi

14 Feb 2020

Problem Statement

- Different Kernels and OSes are efficient in different system calls and tasks.

Problem Statement

- Different Kernels and OSes are efficient in different system calls and tasks.
- Fundamental differences in abstraction & implementation.

Problem Statement

- Different Kernels and OSes are efficient in different system calls and tasks.
- Fundamental differences in abstraction & implementation.
- Quantitative analysis of system calls' efficiency with given architecture and which environment.

Problem Statement

- Different Kernels and OSes are efficient in different system calls and tasks.
- Fundamental differences in abstraction & implementation.
- Quantitative analysis of system calls' efficiency with given architecture and which environment.
- To make different kernel implementations for applications.

Objectives

The project can be divided into the following tasks.

1. Literary Analysis to identify the systems to compare and what standard applications will be used.

Objectives

The project can be divided into the following tasks.

1. Literary Analysis to identify the systems to compare and what standard applications will be used.
2. Quantitative comparison of function call stacks.

Objectives

The project can be divided into the following tasks.

1. Literary Analysis to identify the systems to compare and what standard applications will be used.
2. Quantitative comparison of function call stacks.
3. Compare why is there a significant difference in timing (>1.5x).

Objectives

The project can be divided into the following tasks.

1. Literary Analysis to identify the systems to compare and what standard applications will be used.
2. Quantitative comparison of function call stacks.
3. Compare why is there a significant difference in timing (>1.5x).
4. Analyse the code and data-structures of the system calls having significant difference in timings.

Objectives

The project can be divided into the following tasks.

1. Literary Analysis to identify the systems to compare and what standard applications will be used.
2. Quantitative comparison of function call stacks.
3. Compare why is there a significant difference in timing (>1.5x).
4. Analyse the code and data-structures of the system calls having significant difference in timings.
5. Integrate the code together and make a more efficient kernel for particular applications.

Literary Analysis

Existing papers compare performance of the two most popular systems: Linux and FreeBSD.

- Evolution of Linux and FreeBSD. [1].
- Performance Analysis and comparison of Implementation of Linux and FreeBSD systems. [2].

Literary Analysis

Existing papers compare performance of the two most popular systems: Linux and FreeBSD.

- Evolution of Linux and FreeBSD. [1].
- Performance Analysis and comparison of Implementation of Linux and FreeBSD systems. [2].

Research Papers and documentation which show the significant difference in implementation details of the Linux and FreeBSD systems:

- Quantitative Scheduler comparison. [3].
- Analyzing a decade of system calls [4].

Literary Analysis Cont.

Benchmarks suites to simulate application workload and invoke core system calls:

- Unixbench [5].
- Filebench [6].

Virtual Machine vs Native System

Trade-offs in doing the analysis on virtual machines vs installing the OSes on native hardware.

Virtual Machine vs Native System

Trade-offs in doing the analysis on virtual machines vs installing the OSes on native hardware.
Virtual machines, pros:

- Simpler and faster set-up each time. Connection of host machine with internet.
- Can set equal hardware specifications very easily.

Virtual Machine vs Native System

Trade-offs in doing the analysis on virtual machines vs installing the OSes on native hardware.
Virtual machines, pros:

- Simpler and faster set-up each time. Connection of host machine with internet.
- Can set equal hardware specifications very easily.

Virtual machines, cons:

- Due to the abstraction on another layer, have incorrect quantitative results and not true comparison.

Virtual Machine vs Native System Cont.

Native hardware, pros:

- True timing values comparison for the benchmarks.
- Can compare on bigger benchmarks, comparable to production architecture.

Virtual Machine vs Native System Cont.

Native hardware, pros:

- True timing values comparison for the benchmarks.
- Can compare on bigger benchmarks, comparable to production architecture.

Native Hardware, cons:

- Complex to (repeatedly) set up, including the partitioning tables and internet connections. (emperical evidence)
- Difficult to get exact same configuration systems.

Virtual Machine Installation

Installed Linux and FreeBSD on VMs and the values of individual benchmarks were different in comparison to the same configuration native system.

Native Machine Installation

I installed Linux and FreeBSD on two same configuration systems. Due to difference in partitioning schemes. Configured FreeBSD to use GUI and working internet behind proxy. Document [7].

Configuration

- x86_64 arch.
- 8 core i7-6700 CPU @ 3.40 GhZ CPU.
- 8 GB RAM.
- Caches - L1d cache:32K, L1i cache: 32K, L2 cache: 256K, L3 cache: 8192K.
- GCC-9.
- FreeBSD 12.1 STABLE.
- Ubuntu 18.04.40.
- Unixbench and Filebench.

Steps

- Profiling tools perf on Ubuntu and dtrace on FreeBSD to inspect the function call stacks.
- Use the Flamegraph script to visualize the output as flamegraphs.
- Do this for each of 20 Unixbench tests and 15 micro-file benchmarks.
- Compare the score and timing information on each test.

Automation

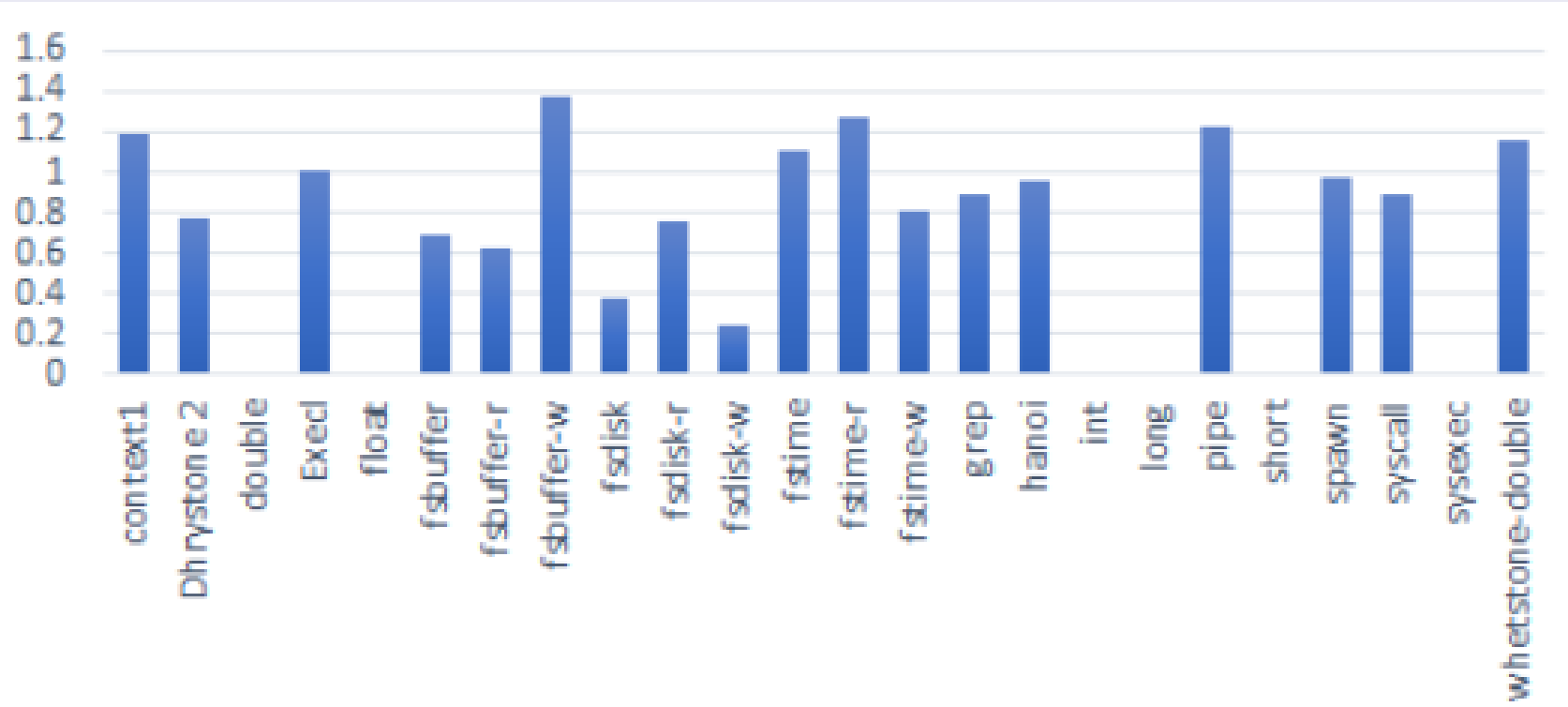
More than 20 tests in unixbench and 10-15 micro file benchmarks in filebench.
Sh scripts to automate the running of each test in :

- dhry2reg whetstone-double arithoh short int long float double hanoi context1 syscall sysexec pipe spawn execl fstime-w fstime-r fstime fsbuffer-w fsbuffer-r fsbuffer fsdisk-w fsdisk-r fsdisk grep
- filemicro_*

and drawing the flame-graphs as shown below. Full sh files here [9].

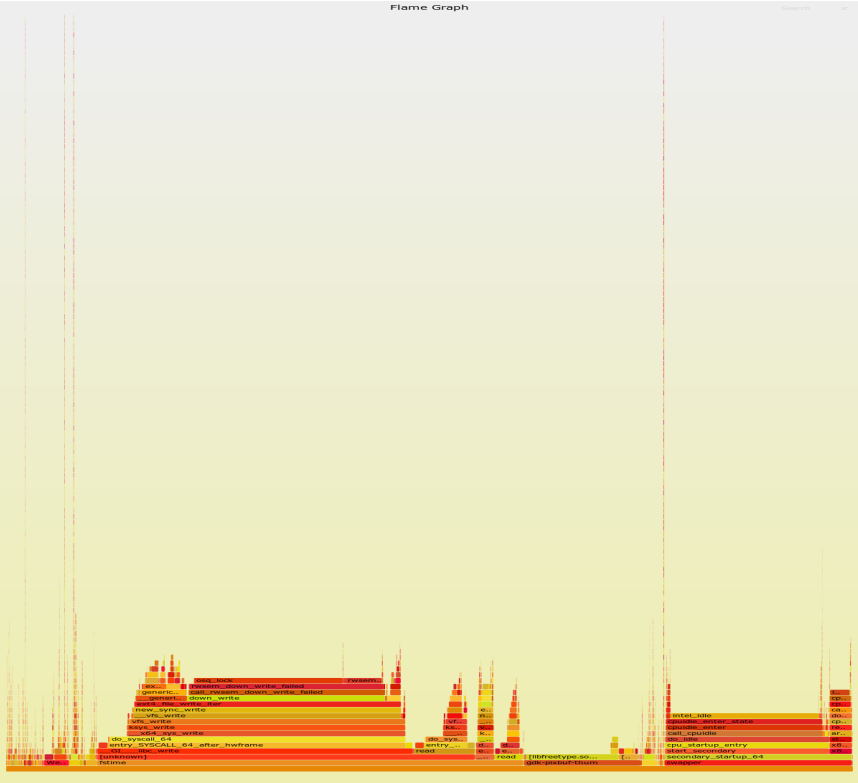
Timing Comparison

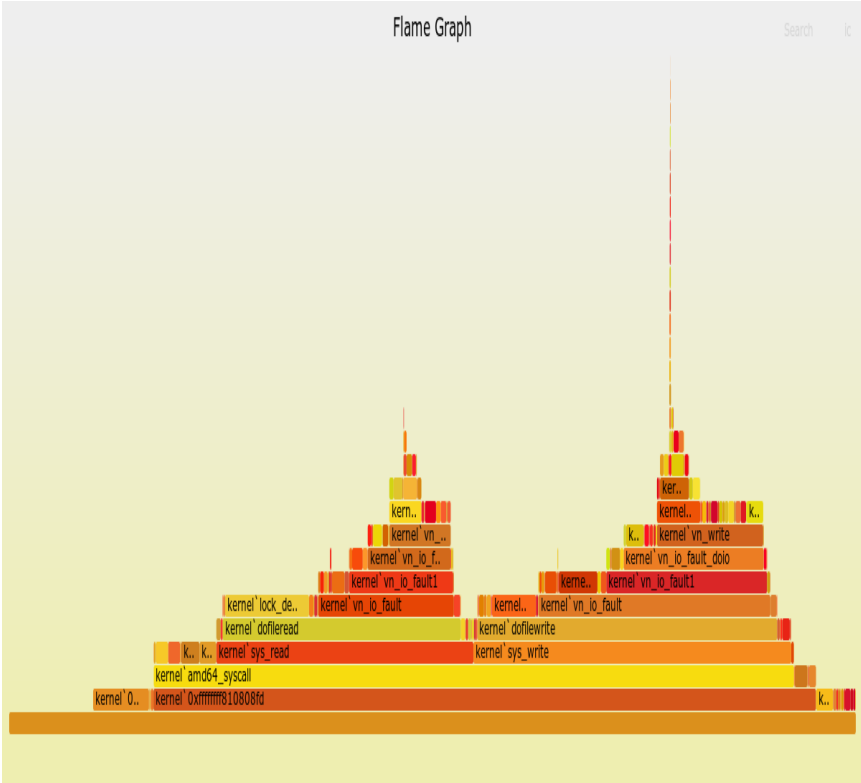
Performance FreeBSD / Ubuntu



Famegraphs

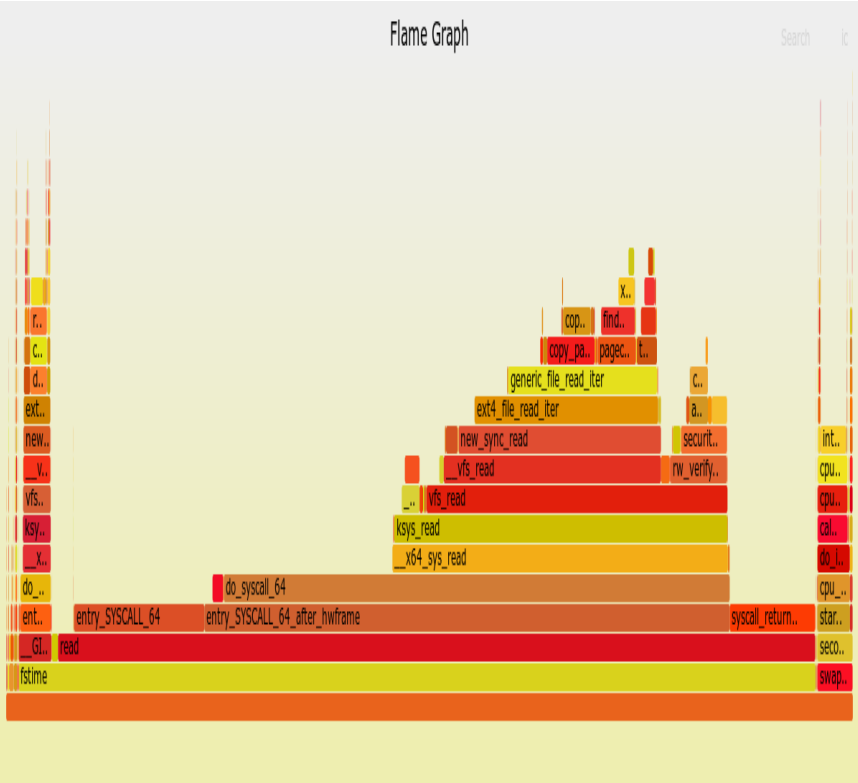
fsbuffer

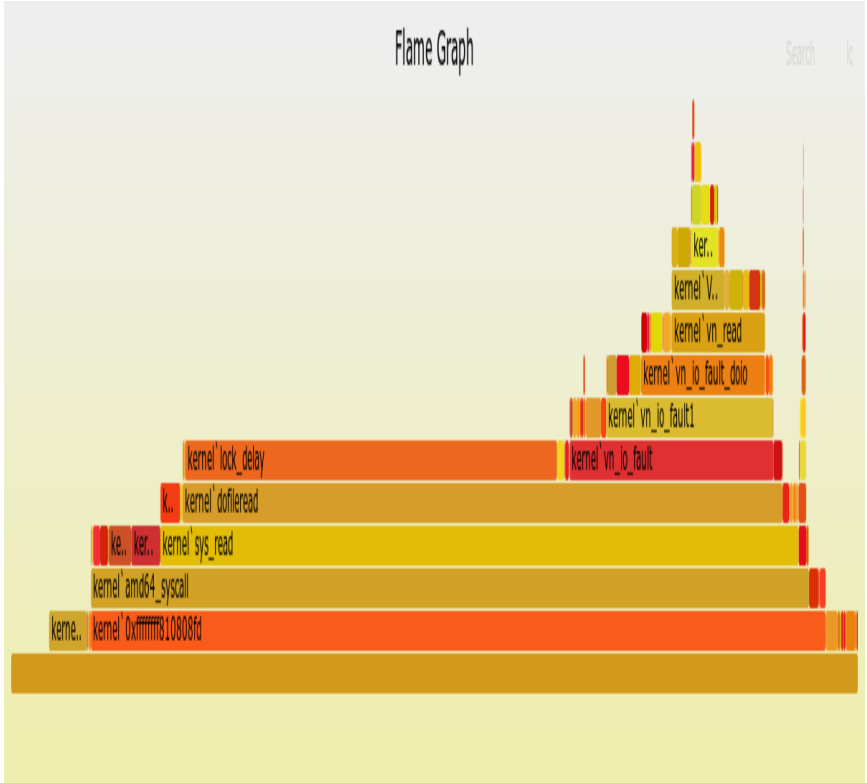




Famegraphs

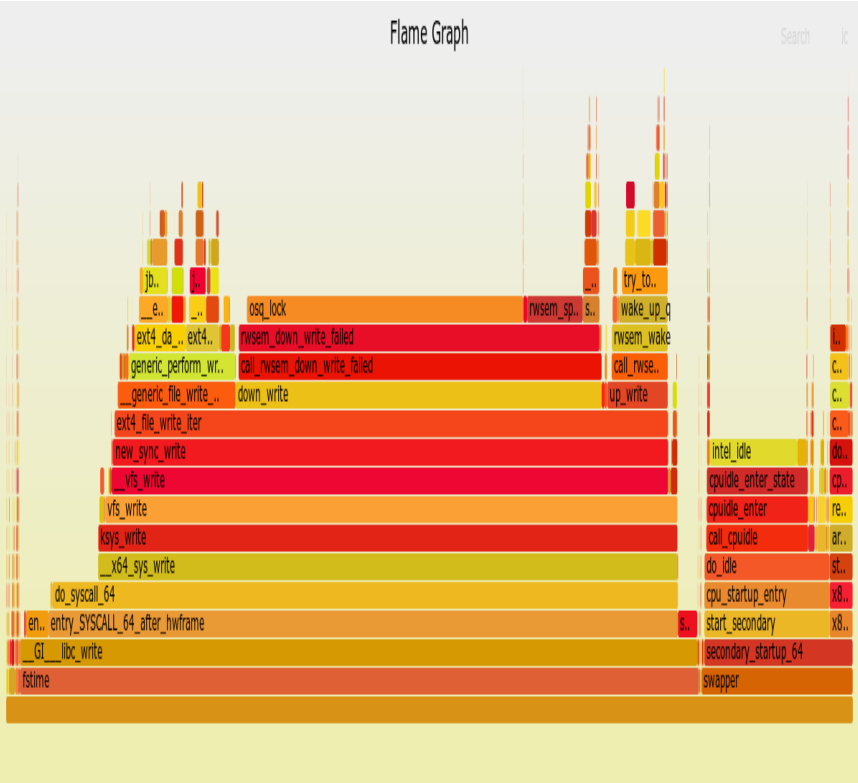
fsbuffer-r

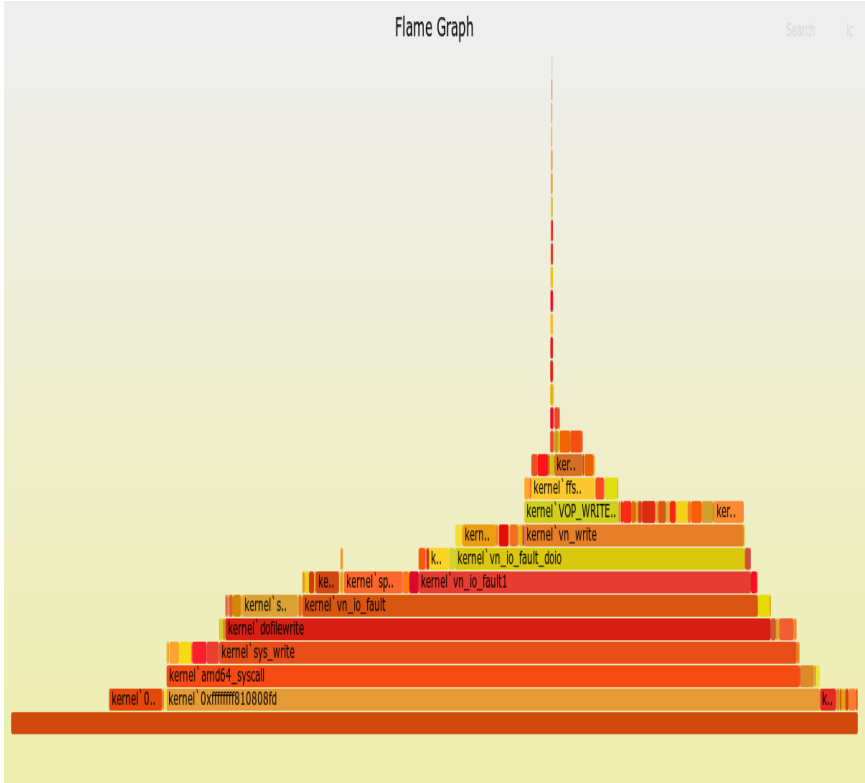




Famegraphs

fsbuffer-w





Key Points fsbuffer

Fsbuffer reads and writes from buffer to disk. Significant performance differences in FreeBSD and Linux. Involved System Calls:

- sys_read
- sys_write

Read faster in Ubuntu and write faster in FreeBSD.
In the write flamegraph we see, in Ubuntu, major time is spent on acquiring the spin lock (osq_lock) and that is not the case in FreeBSD which alternatively does it's locking procedure in read.

fstime, pipe and context

- The graph is analyzed for fstime group which is just opposite and copying from the disk file system to buffer.
- The pipe test has pipe_read and pipe_write as it's main calls.
- Context also uses the pipe_read and write calls.

fstime, pipe and context

- The graph is analyzed for fstime group which is just opposite and copying from the disk file system to buffer.
- The pipe test has pipe_read and pipe_write as it's main calls.
- Context also uses the pipe_read and write calls.

Main difference is due to the scheduling and locking mechanism and order for read and write. Detailed Report [8].

Conclusion and Next steps

Conclusion:

- Linux optimal for read-heavy operation applications and FreeBSD optimal for write-heavy applications.

Conclusion and Next steps

Conclusion:

- Linux optimal for read-heavy operation applications and FreeBSD optimal for write-heavy applications.

Next steps:

- Analyse the scheduling and locking mechanism code for both.

References

1. ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering. September 2006 Pages 204–211https://doi.org/10.1145/1159733.1159765
2. G. Lencse and S. Répás, "Performance Analysis and Comparison of Different DNS64 Implementations for Linux, OpenBSD and FreeBSD," 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, 2013, pp. 877-884.
3. Scheduler battle PDF.
4. Analyzing a decade of Linux system calls PDF.

References Cont.

5 UnixBench Github

6 FileBench Paper.

7 Installing FreeBSD on native system Google Doc.

8 Detail Flamegraph analysis Google Doc.

9 Automation Script Github