NFS-like system on FUSE

Adithya Bhat - adbhat@cs.wisc.edu

Setup

FUSE 2.9.7
Apache Thrift 0.9.3
Ubuntu VMs - 2 vCPUs, 7.5GB memory

Design

Layers

FUSE - RPCGateway - Thrift Client ==== Thrift Server - RPCGateway - NFS Server.
Pro - readability, maintainability, easy to extend.
Con - copying between layers, not sure if pass by reference mitigates this problem.

Connection Persistence

Pro - better performance
Con - harder crash handling, cannot handle multiple connections per thread
In a throughput > latency case, would argue that connections are not optimal.

Commit

Buffers on the server side
NOTE : I thought it would be more interesting to buffer all the data until and unless an fsync or a release is called, so implemented that. The complication here is discussed under issues.
Other Classes - WriteCache and WriteCacheEntry.

- Reason - to abstract out the collection buffering the write requests
- Why ? - make the insert, flush operations thread-safe. (Multi-threaded server)
- Con - costly in terms of time and space.
- Unless the application really knows what it's doing, seems sub-optimal.

Piece by piece lookup

Unnecessary / Implicitly handled by FUSE ?

Crash Consistency

Server Crashes -
Retry with exponential backoff, 10 tries, starts at 1 second.

File Handle

stat->dev_id, stat->ino
ioctl call to get generation number
Optionally, can use the path also here
NOTE : generation numbers are huge.. Instead of combining by adding, etc, can combine by selecting how many digits we think would be necessary. Very unlikely that one inode will see a million generations over one NFS session (right?)

Issues

Commit Semantics

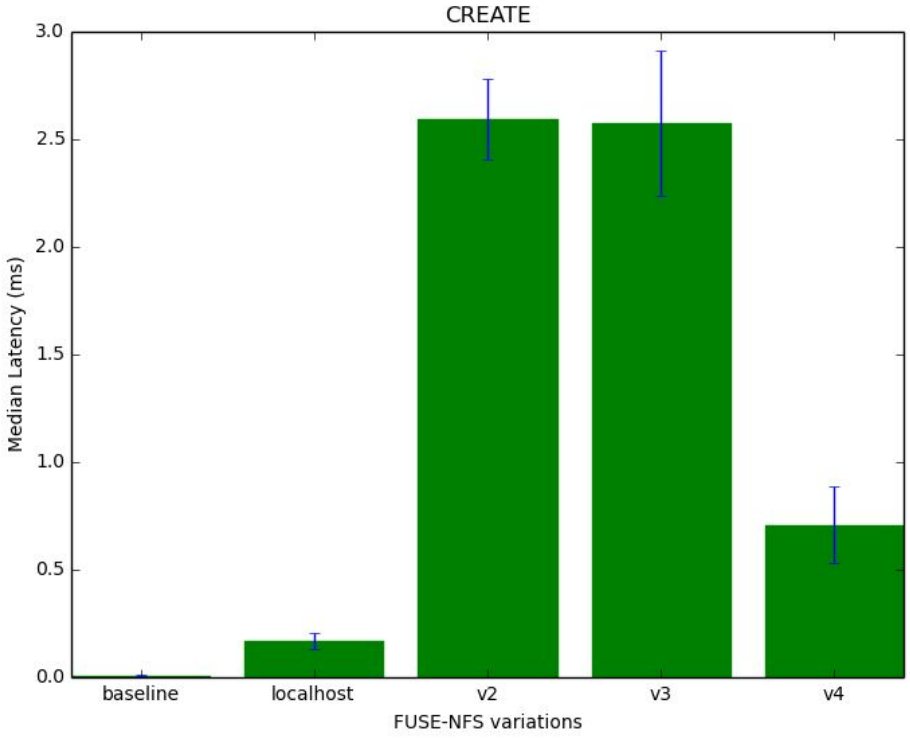
Situation : multiple clients accessing the same file.
Server is stateless, does not differentiate between clients.
Fsync called from any client will cause flushing of buffer
How can Client1 tell the difference between fsync by another client and server crash ?

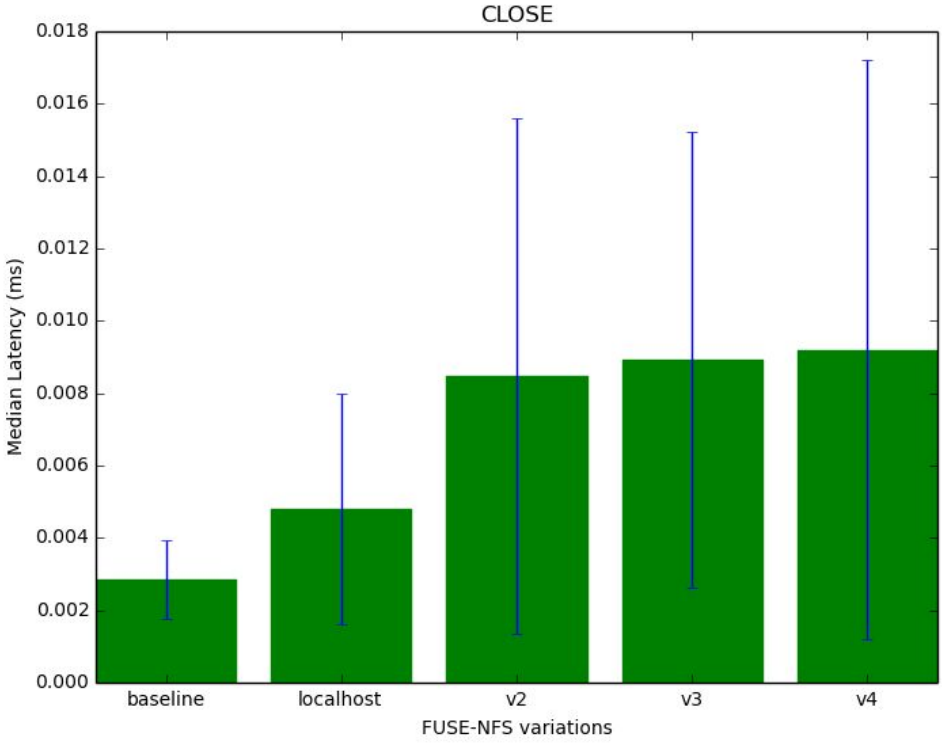
Versions

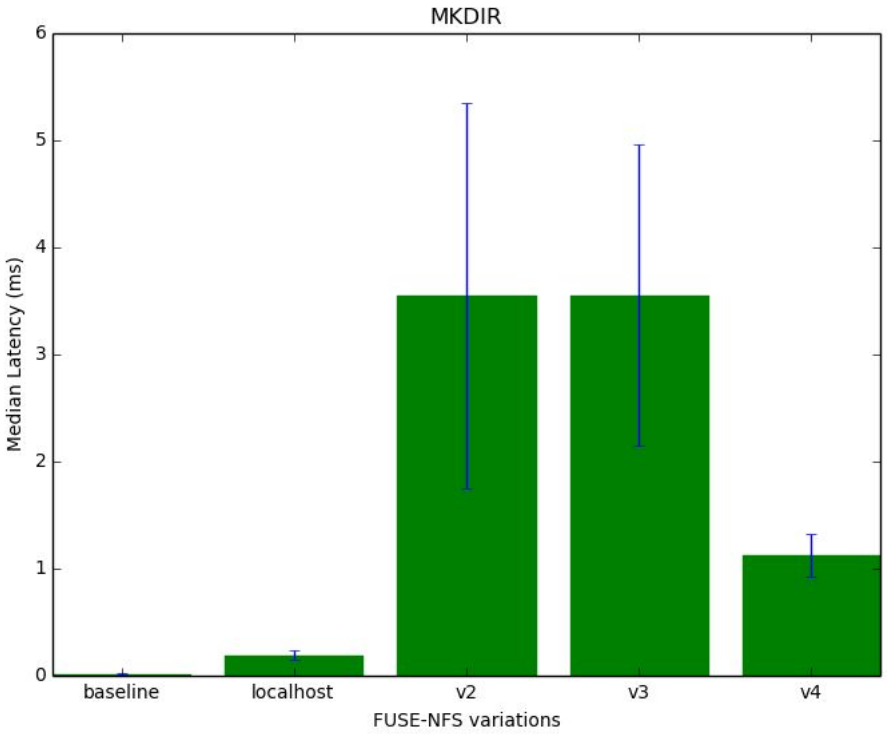
V2
Not a persistent connection
No explicit write caching

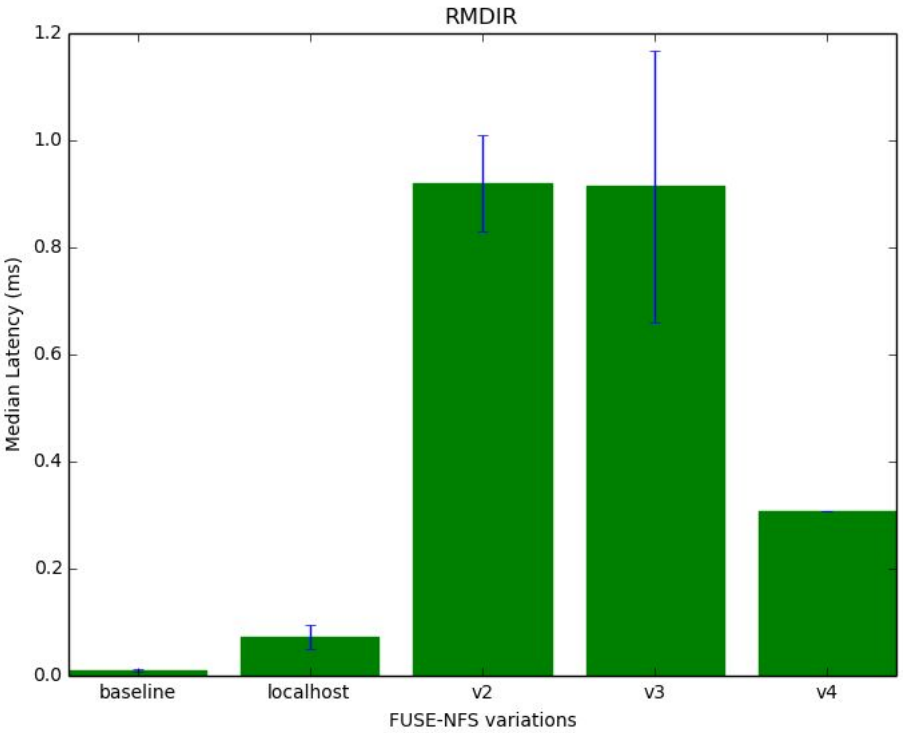
V3
Added COMMIT enhancement, and multithreaded server

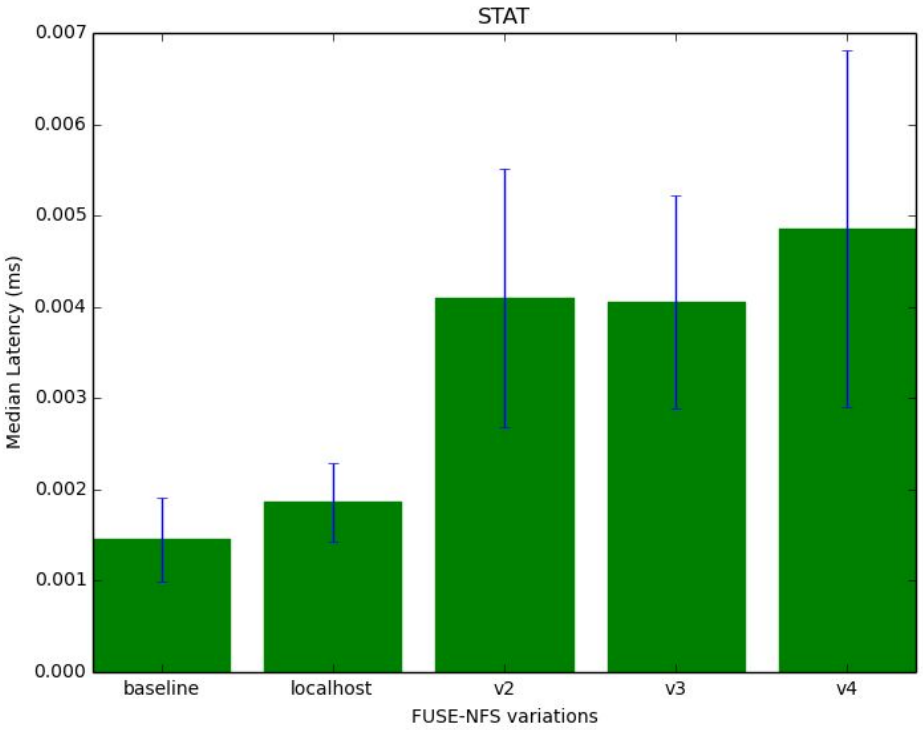
V4
Persistent connection + retry

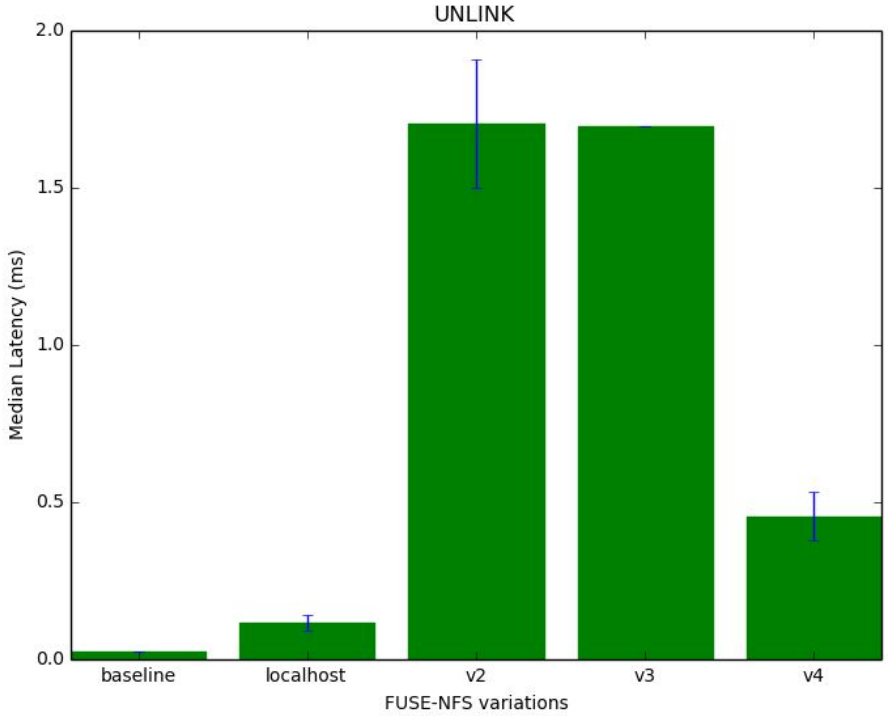

CREATE
Median Latency (ms)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
baseline
localhost
v2
v3
v4
FUSE-NFS variations

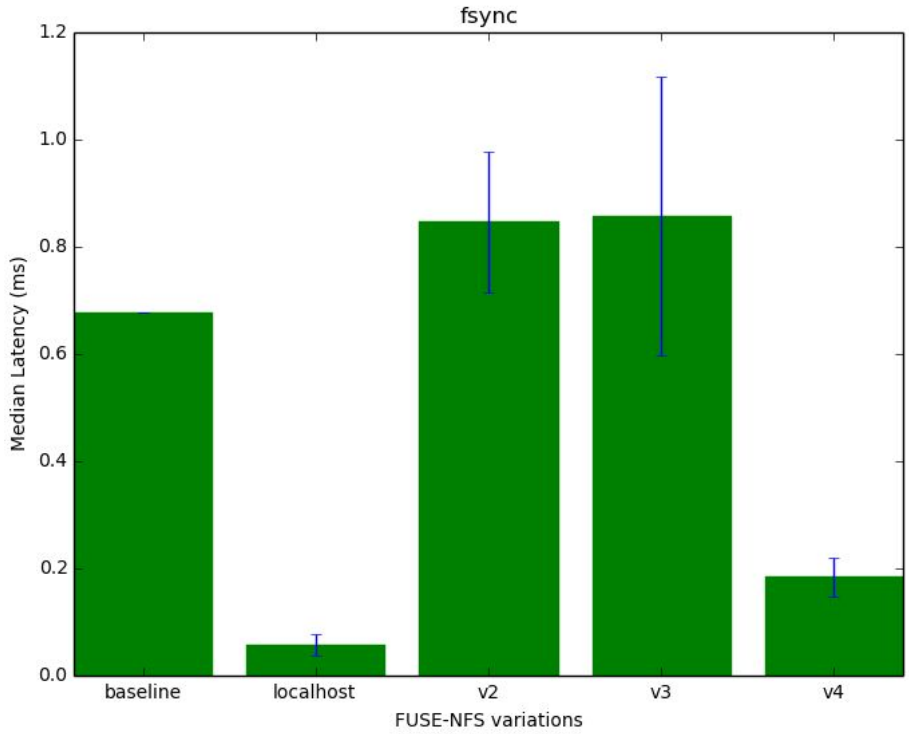

CLOSE
Median Latency (ms)
0.018
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
baseline
localhost
v2
v3
v4
FUSE-NFS variations

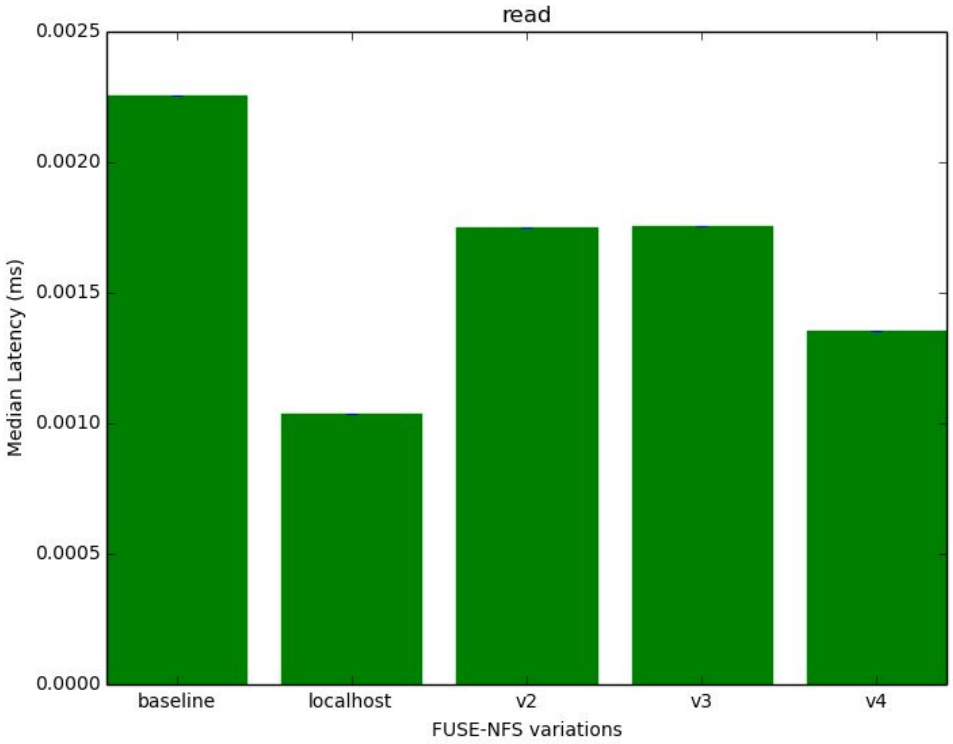

MKDIR
Median Latency (ms)
0
1
2
3
4
5
6
baseline
localhost
v2
v3
v4
FUSE-NFS variations

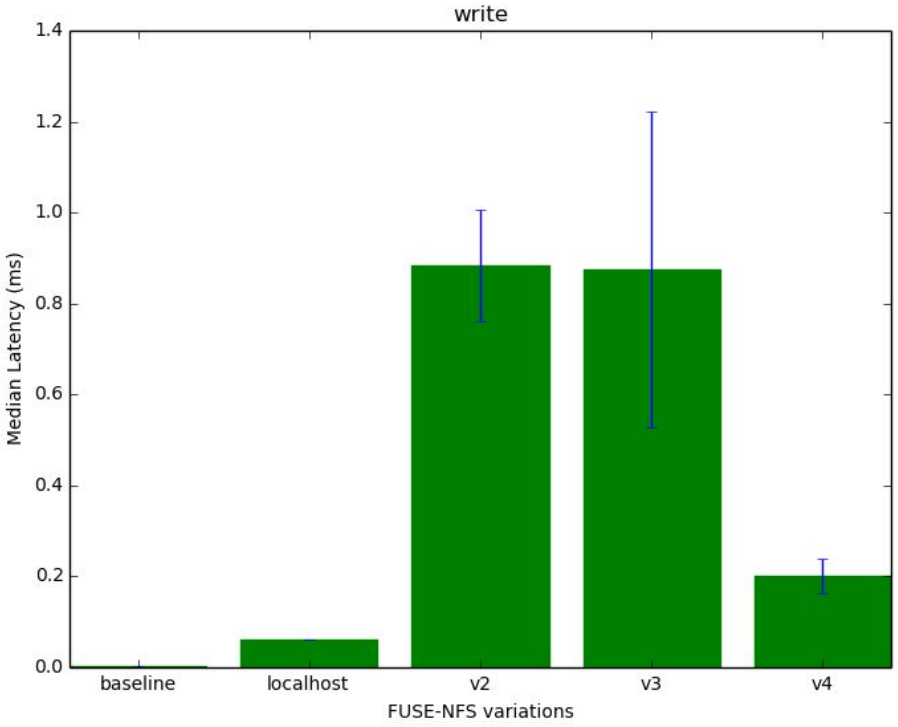

RMDIR
Median Latency (ms)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
baseline
localhost
v2
v3
v4
FUSE-NFS variations

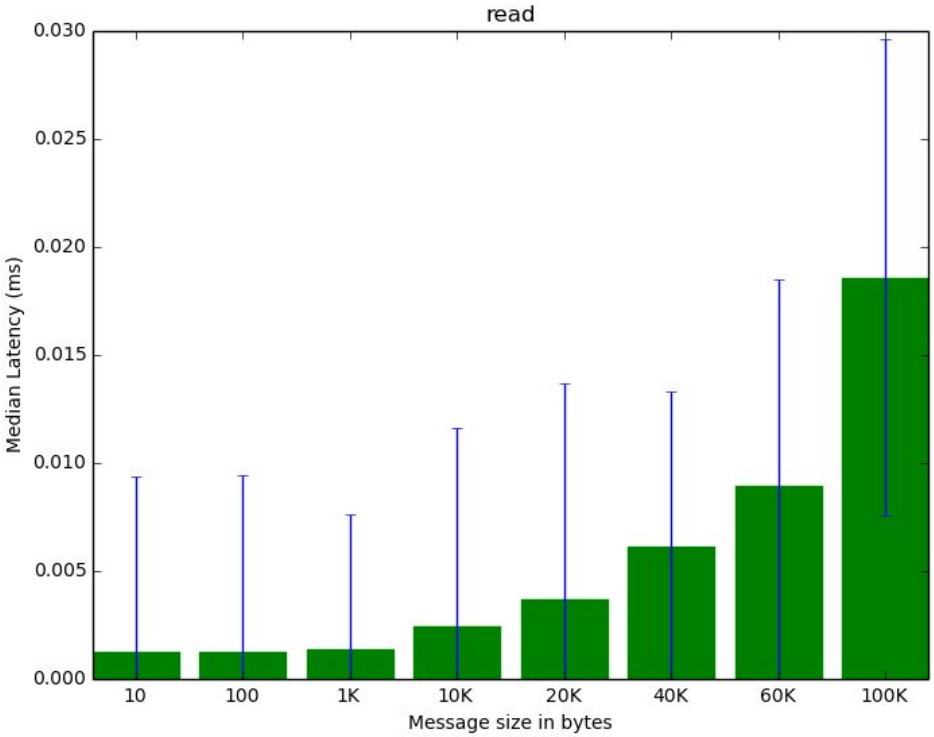

STAT
Median Latency (ms)
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
baseline
localhost
v2
v3
v4
FUSE-NFS variations

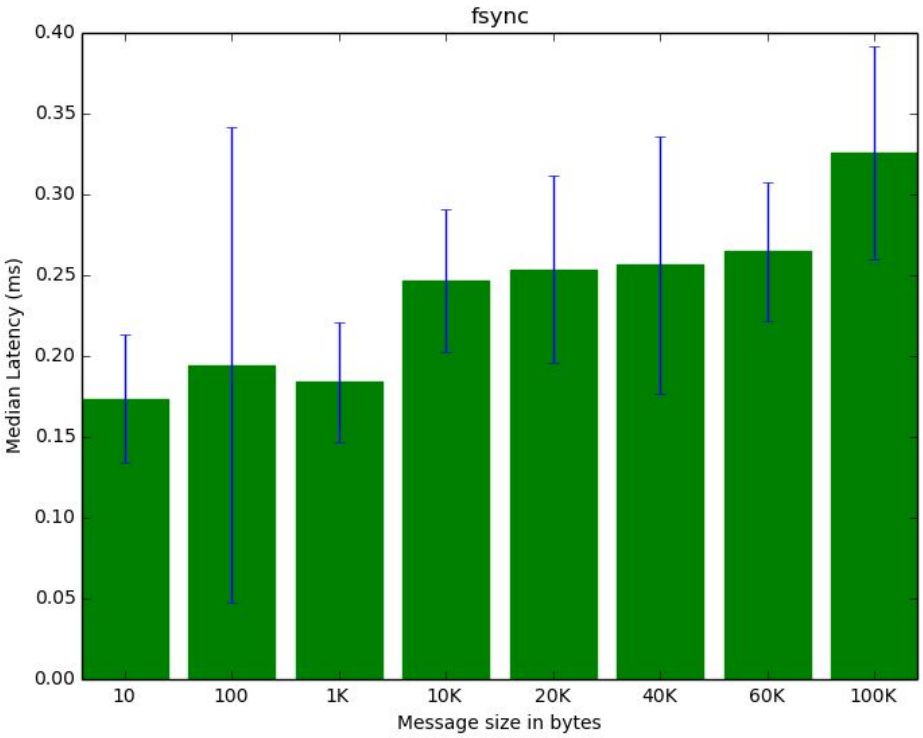

UNLINK
Median Latency (ms)
2.0
1.5
1.0
0.5
0.0
baseline
localhost
v2
v3
v4
FUSE-NFS variations

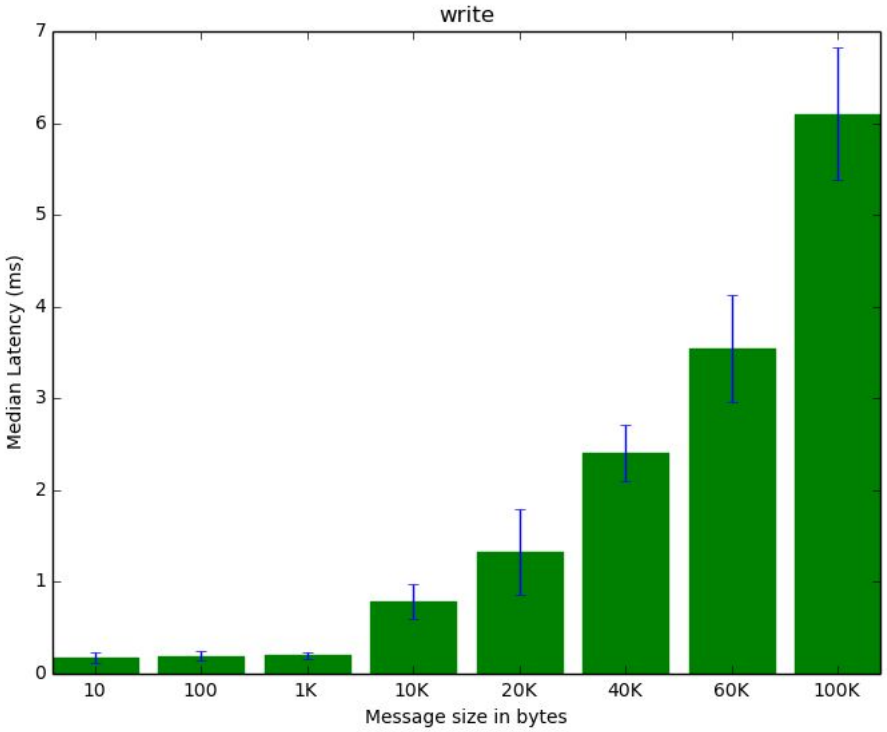

fsync
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Median Latency (ms)
baseline
localhost
v2
v3
v4
FUSE-NFS variations


read
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Median Latency (ms)
baseline
localhost
v2
v3
v4
FUSE-NFS variations


write
Median Latency (ms)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
baseline
localhost
v2
v3
v4
FUSE-NFS variations


read
Median Latency (ms)
0.030
0.025
0.020
0.015
0.010
0.005
0.000
10
100
1K
10K
20K
40K
60K
100K
Message size in bytes


fsync
Median Latency (ms)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
10
100
1K
10K
20K
40K
60K
100K
Message size in bytes


write
Median Latency (ms)
0
1
2
3
4
5
6
7
10
100
1K
10K
20K
40K
60K
100K
Message size in bytes