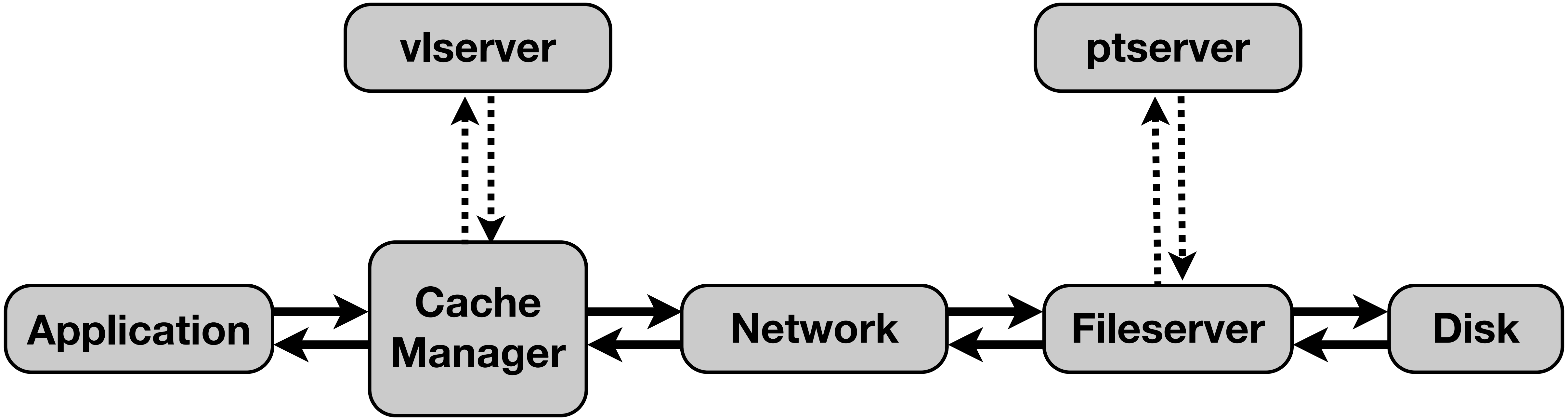
AFS Performance

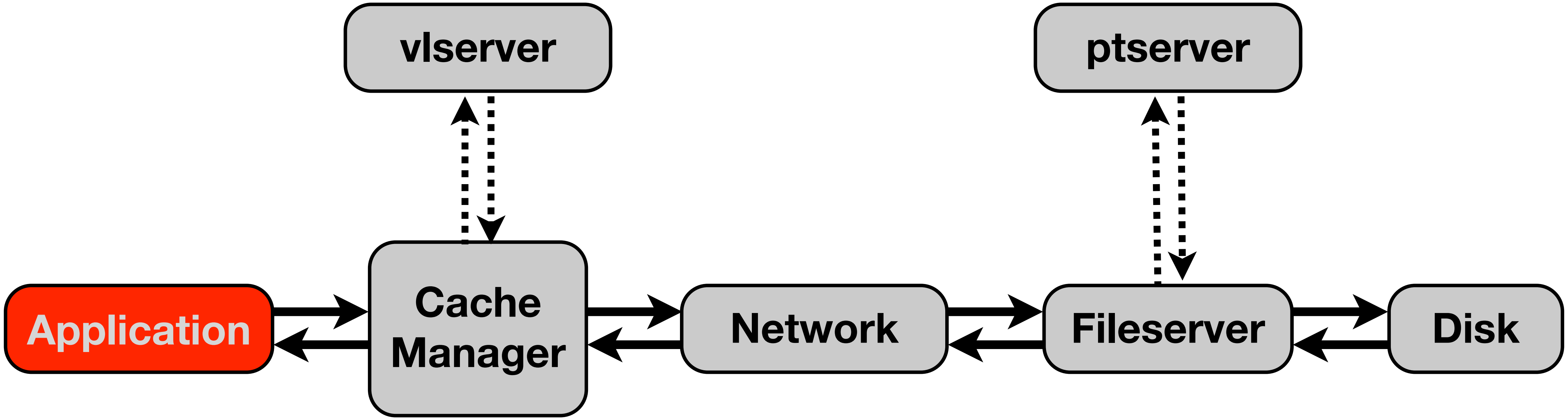
Simon Wilkinson
Your File System Ltd
sxw@your-file-system.com

The life of a request



YourFileSystem

The life of a request

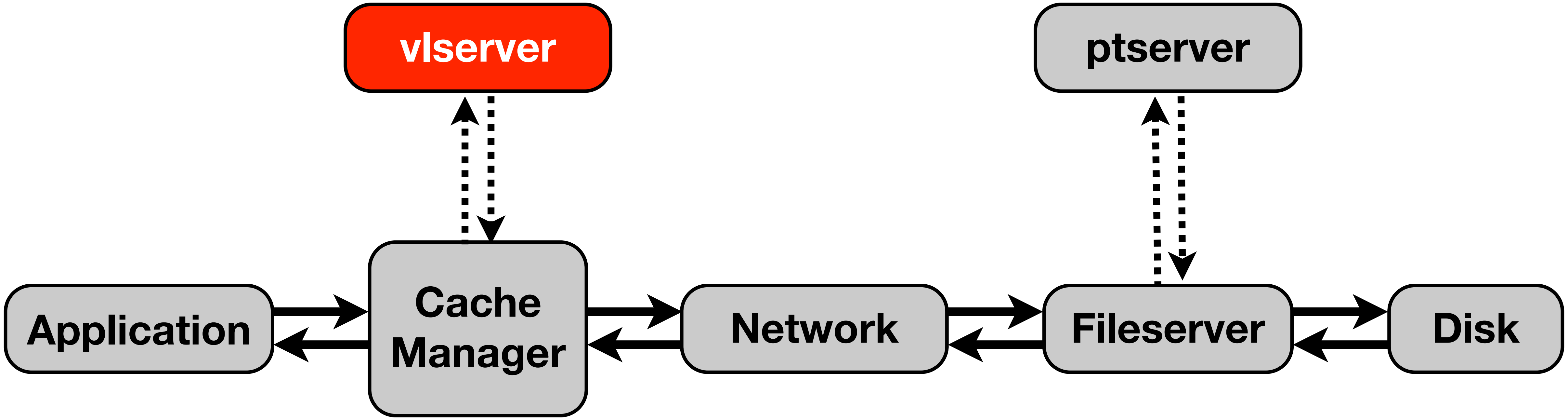


YourFileSystem

Application performance

- Constantly changing tokens is really expensive

The life of a request



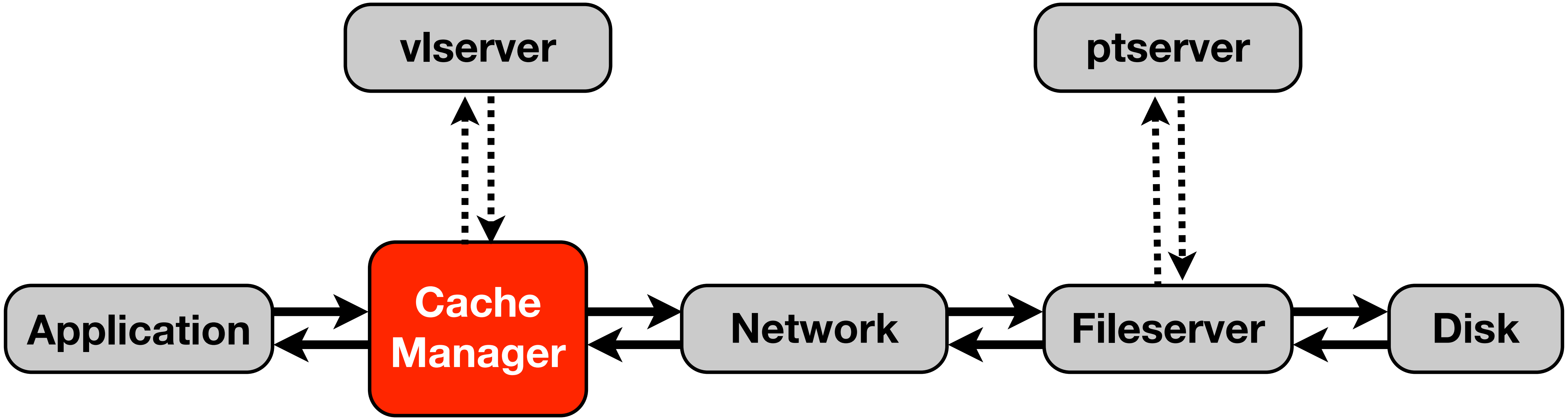
YourFileSystem

vlserver performance

- The client accesses the vlserver the first time a new volume is encountered

- Caches results for up to 2 hours

The life of a request

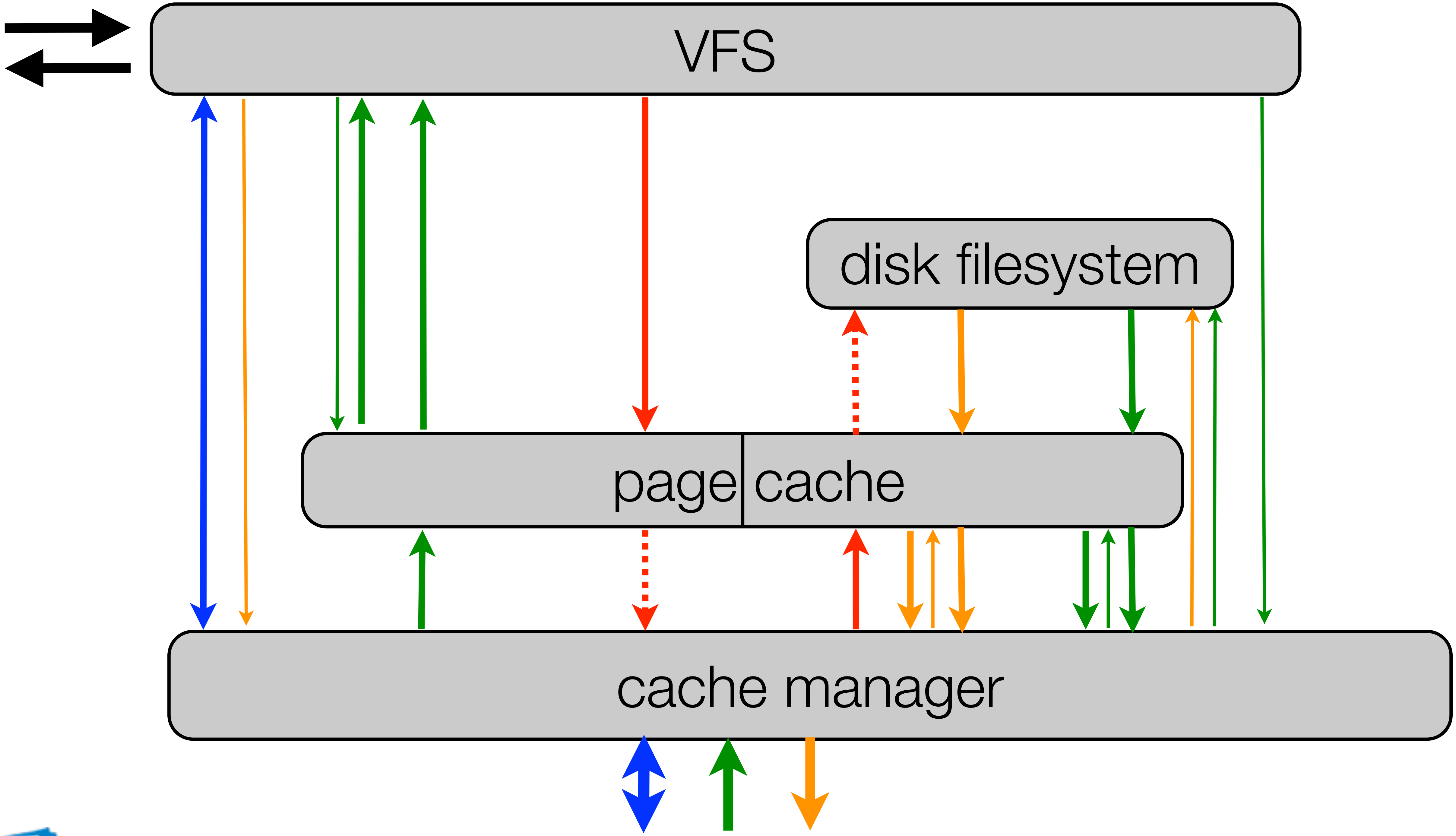


YourFileSystem

The Virtual File System

- Most modern OSes have a virtual filesystem
- Accepts POSIX system calls
- Implements common functionality
- Provides API which filesystems must implement

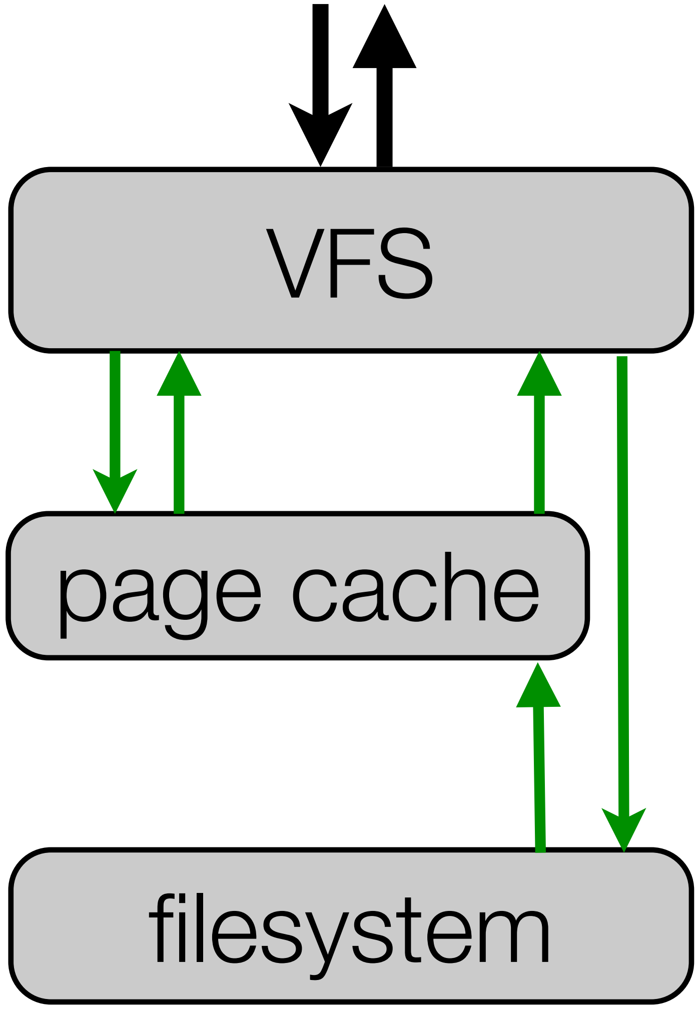
Cache Manager in more detail



YourFileSystem

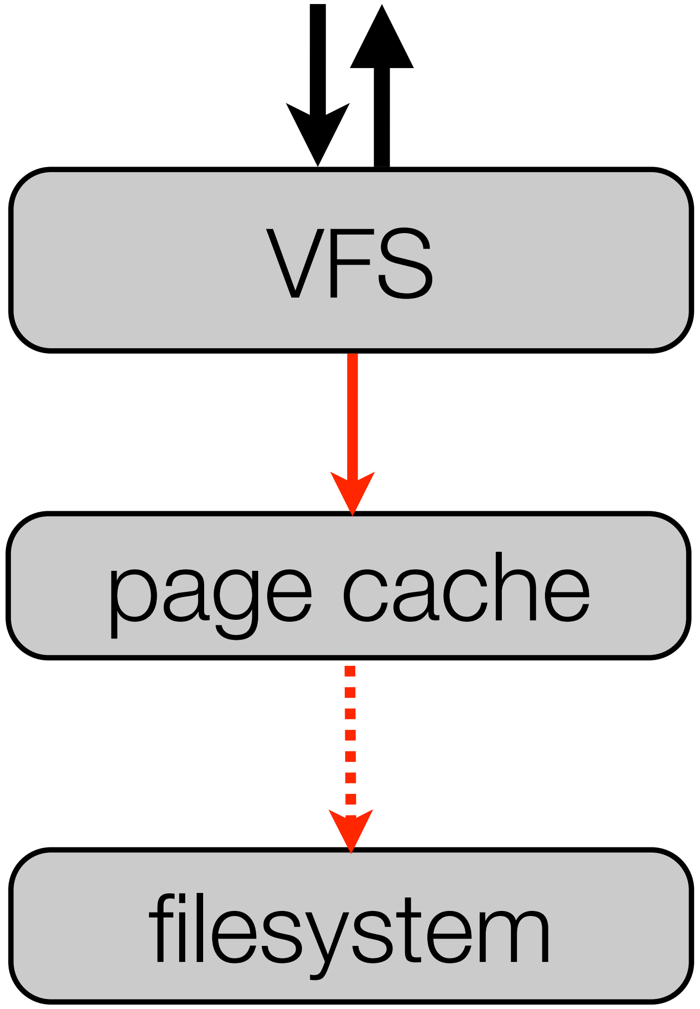
Virtual Memory Filesystems (read)

- All data operations are mediated by the page cache
- VFS checks first for up to date data in cache, and returns this to application
- Otherwise, filesystem is requested to fetch date into page cache
- Either way, caller's data comes from the cache

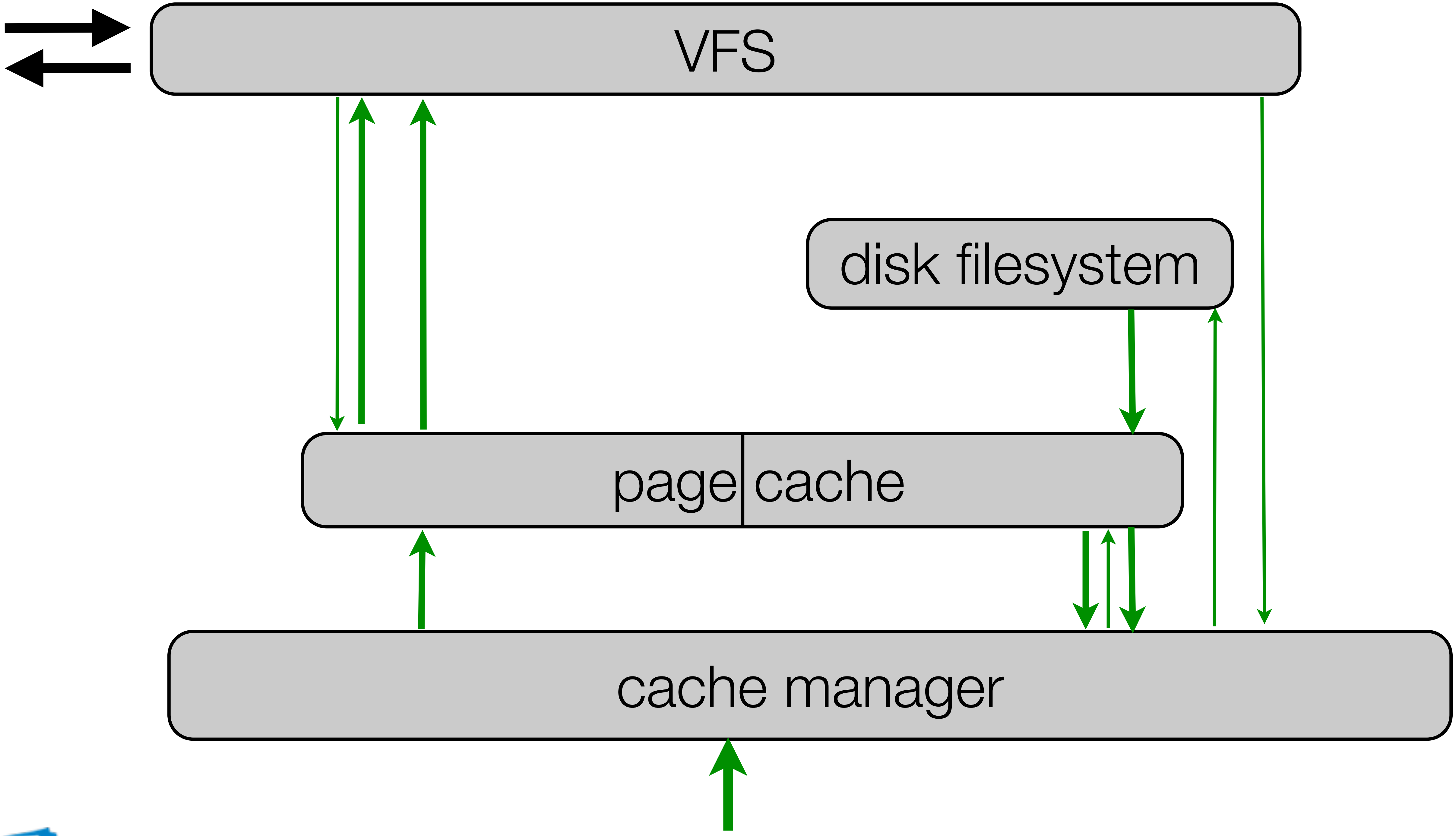


Virtual Memory Filesystems (write)

- Writes go first to the memory cache
- Written out to the filesystem in the background, or when requested
- Page cache is shared between all filesystems, and managed by the kernel
- Use of virtual memory essential for mmap() support

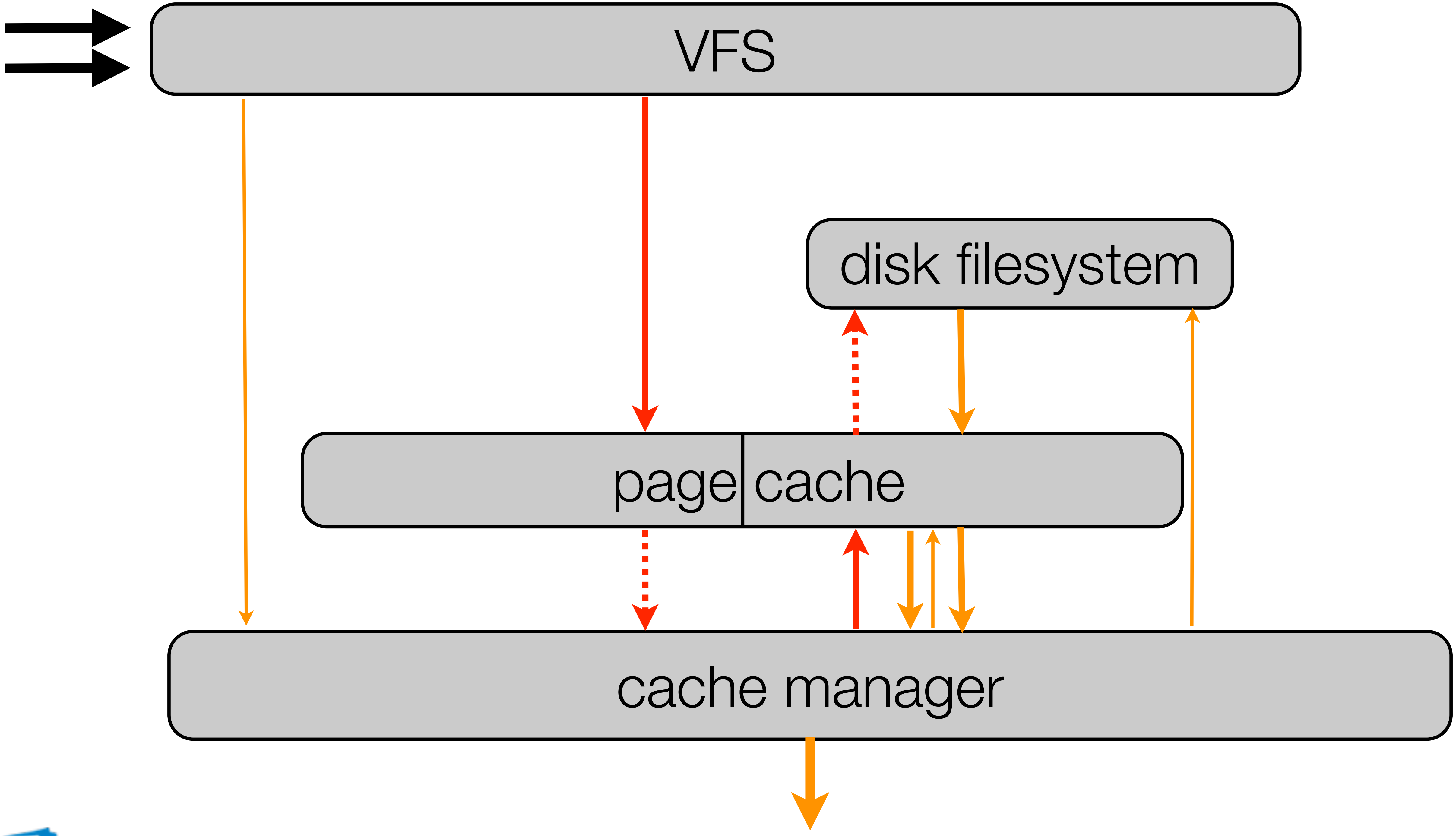


Cache Manager: Reads



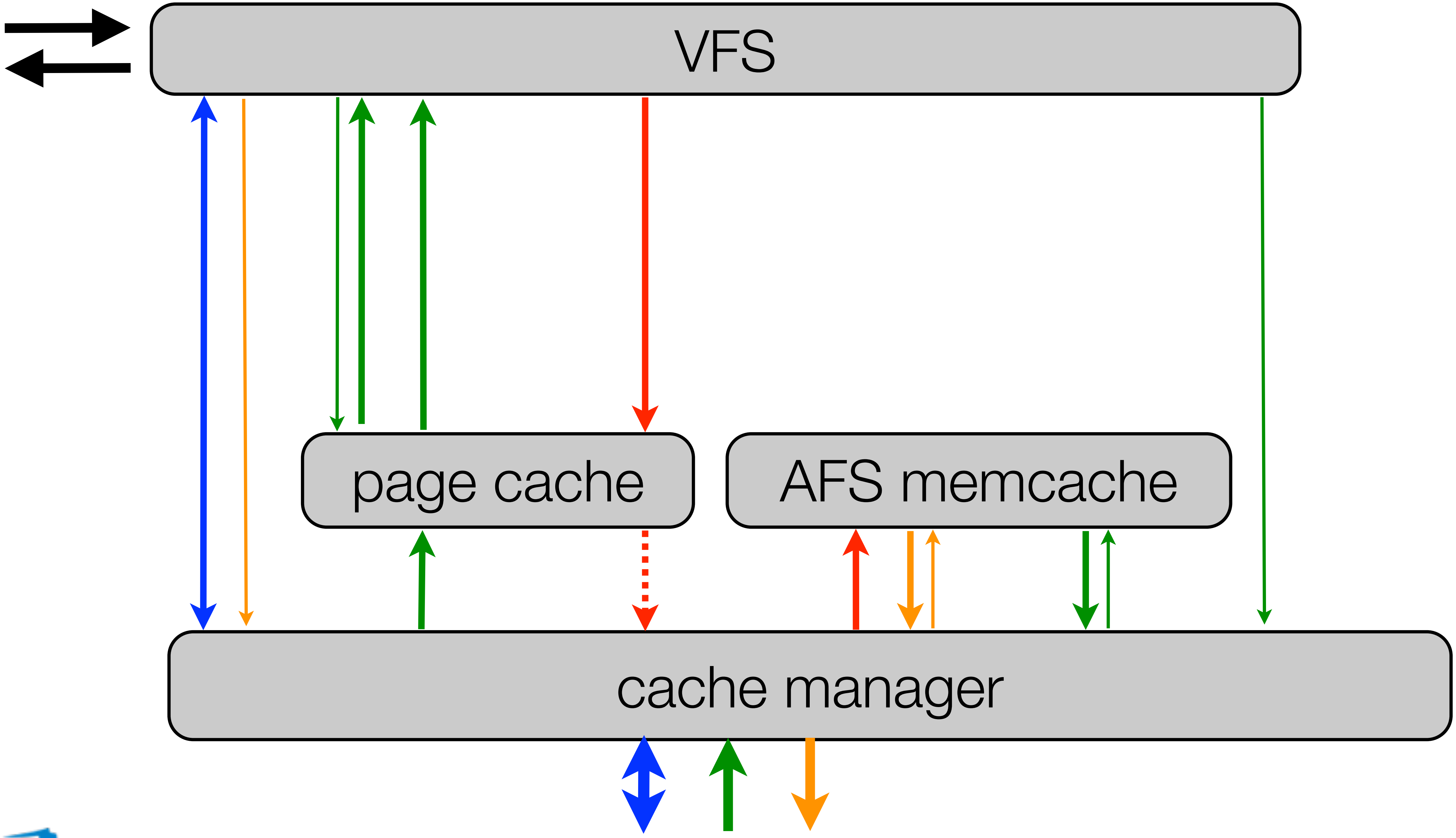
YourFileSystem

Cache Manager: Writes



YourFileSystem

Cache Manager: Memory cache



VFS

page cache

AFS memcache

cache manager

Memory cache: pros and cons

- Memory cache is faster than disk cache
- Cache size is limited to the memory on your machine
- Memory cache space cannot be used for other purposes
- Memory cache cannot partially use chunks

Chunksize

- The cache is split into a set of chunks

$$\mathit{dcache} \times \mathit{chunksize} = \mathit{blocks}$$

- Chunksize is 8k on memory cache, and autotuned on disk cache, according to the cache size

- Chunksize also determines the amount of data fetched with each read from the fileserver

Chunksize pros and cons

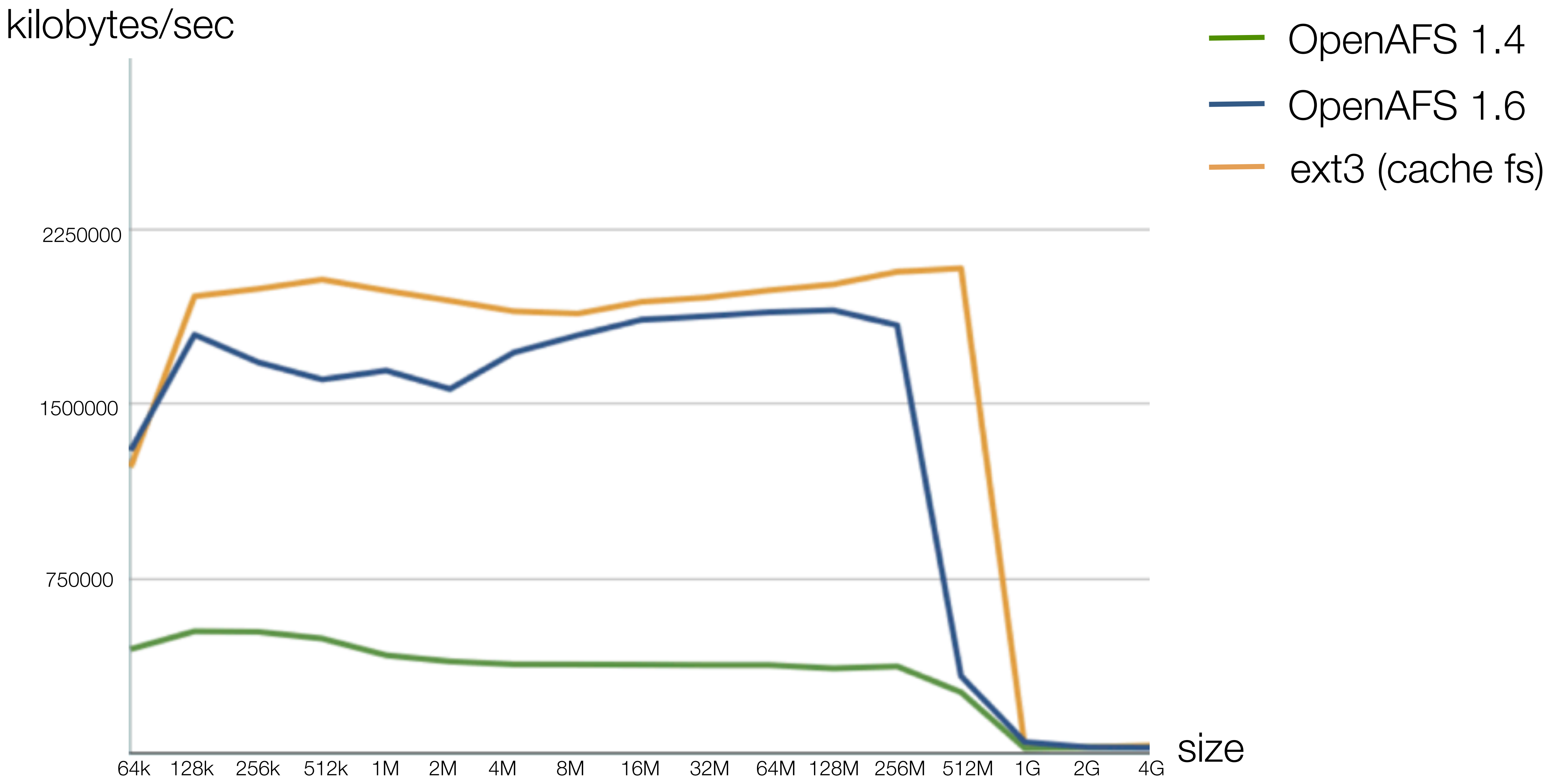
- Chunksize has a big impact on performance
  - Reading 1Mbyte when the application only wants one byte
  - Reading a byte a thousand times when the application reads 1Mbyte byte by byte
- If your application has no locality of access and a working set much larger than your cache, large chunk sizes really hurt performance

YourFileSystem

The Global Lock

- The OpenAFS cache manager was written when kernels only had interrupt and normal contexts
- SMP conversion was done by means of a global lock
- Parallel access speeds are poor

Linux page copying improvements



YourFileSystem

Avoiding the Cache

- fs bypassthreshold -size <filesize>
  - Any files larger than filesize will not be cached
- Use file options to allow application to bypass cache

```
open(“/afs/mycell/path/to/file”,
     O_RDONLY | O_DIRECT)
```

Using the cache more

- fs precache -size <filesize>
  - Controls a readahead size

Perception and storebehind

- On Unix, AFS is write on close
- Users (and applications) don’t expect close to take a long time!
- fs storebehind allows writes to the fileserver to happen in the background
- **BUT** if the write fails, no one will know

Bulkstat, fakestat and friends

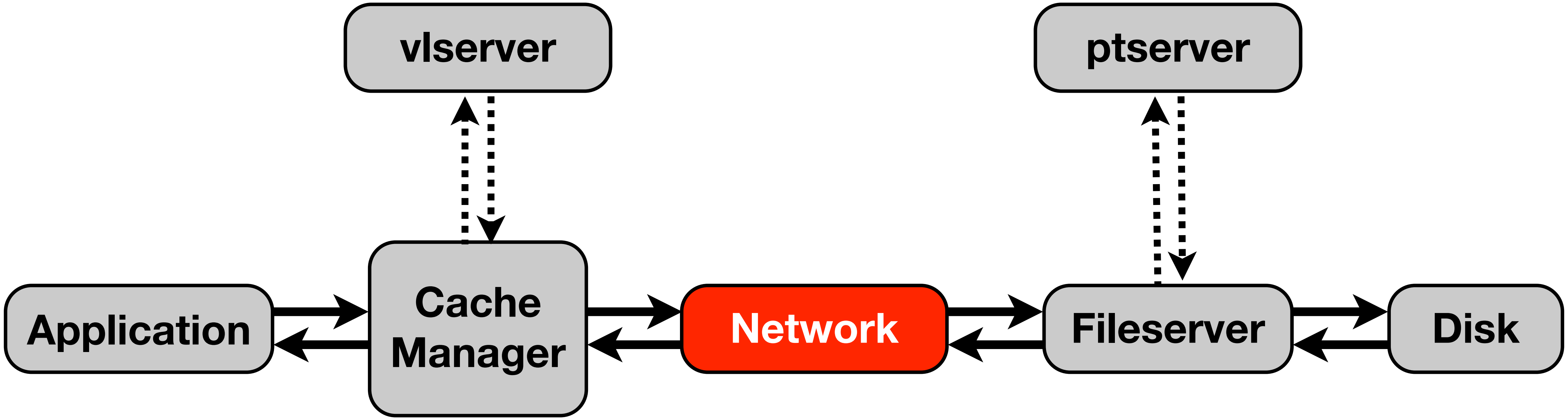
- Mainly concentrating on data operations, but metadata ops can have performance impact too
- The afsd option `-fakestat` avoids looking up the root.cell volume of every cell in /afs
- The option `-fakestat-all` blocks stat lookups of all mountpoints
- Bulkstat (not on Mac OS X) lumps multiple stat operations into a single RPC

YourFileSystem

Tuning the cache manager

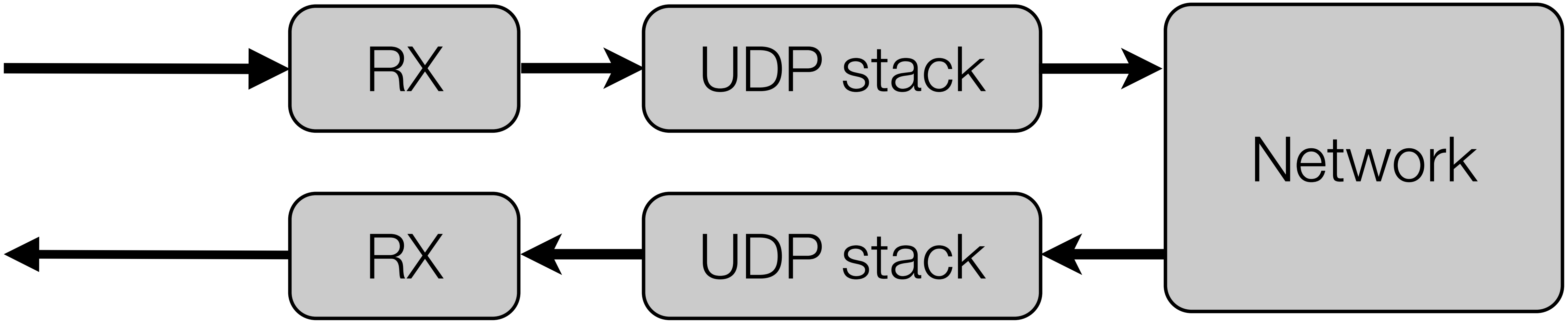
- Apart from the stuff already mentioned, auto tuning works well in 1.6
- Make sure your startup script doesn’t hard code in appropriate values!

The life of a request

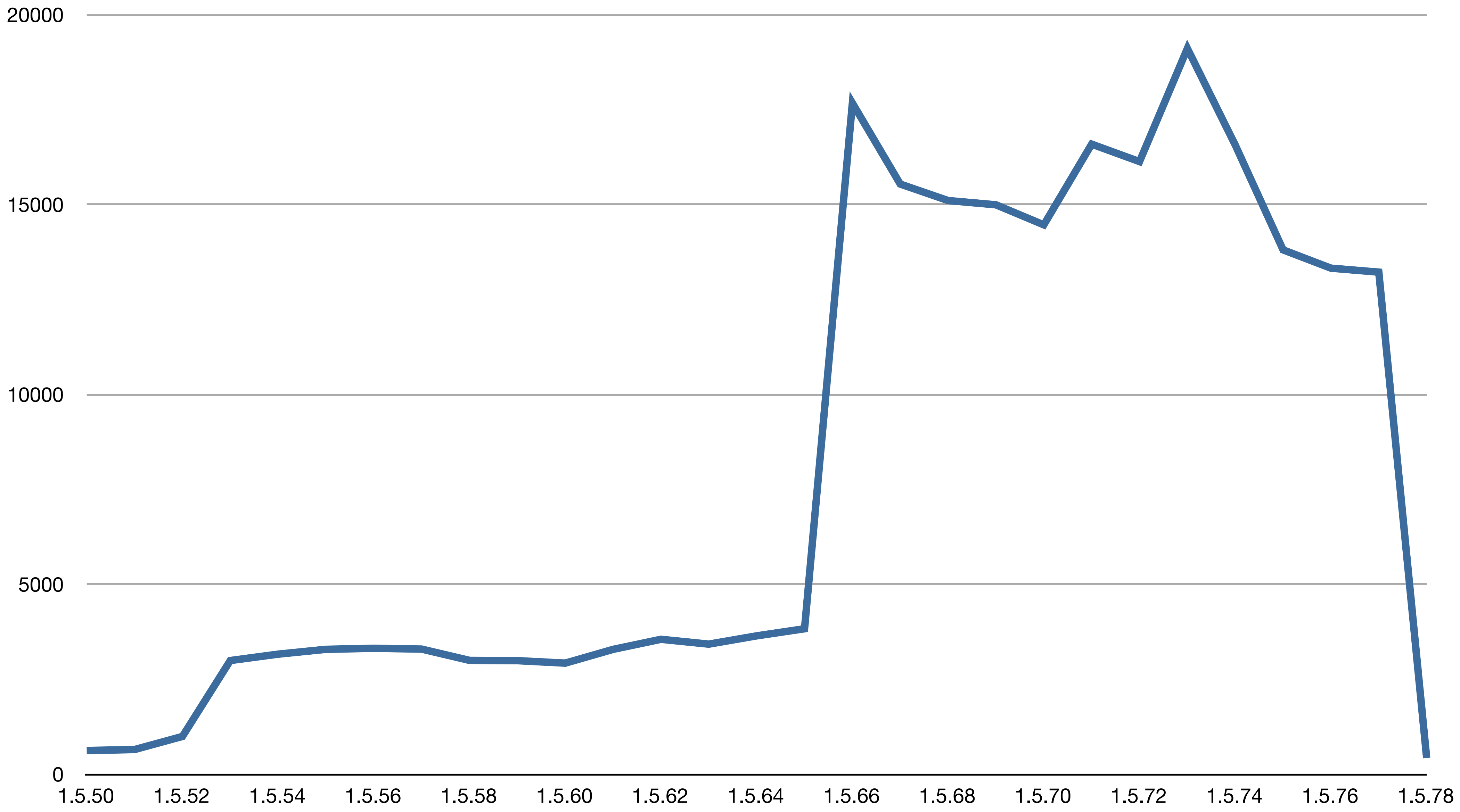


YourFileSystem

Network performance

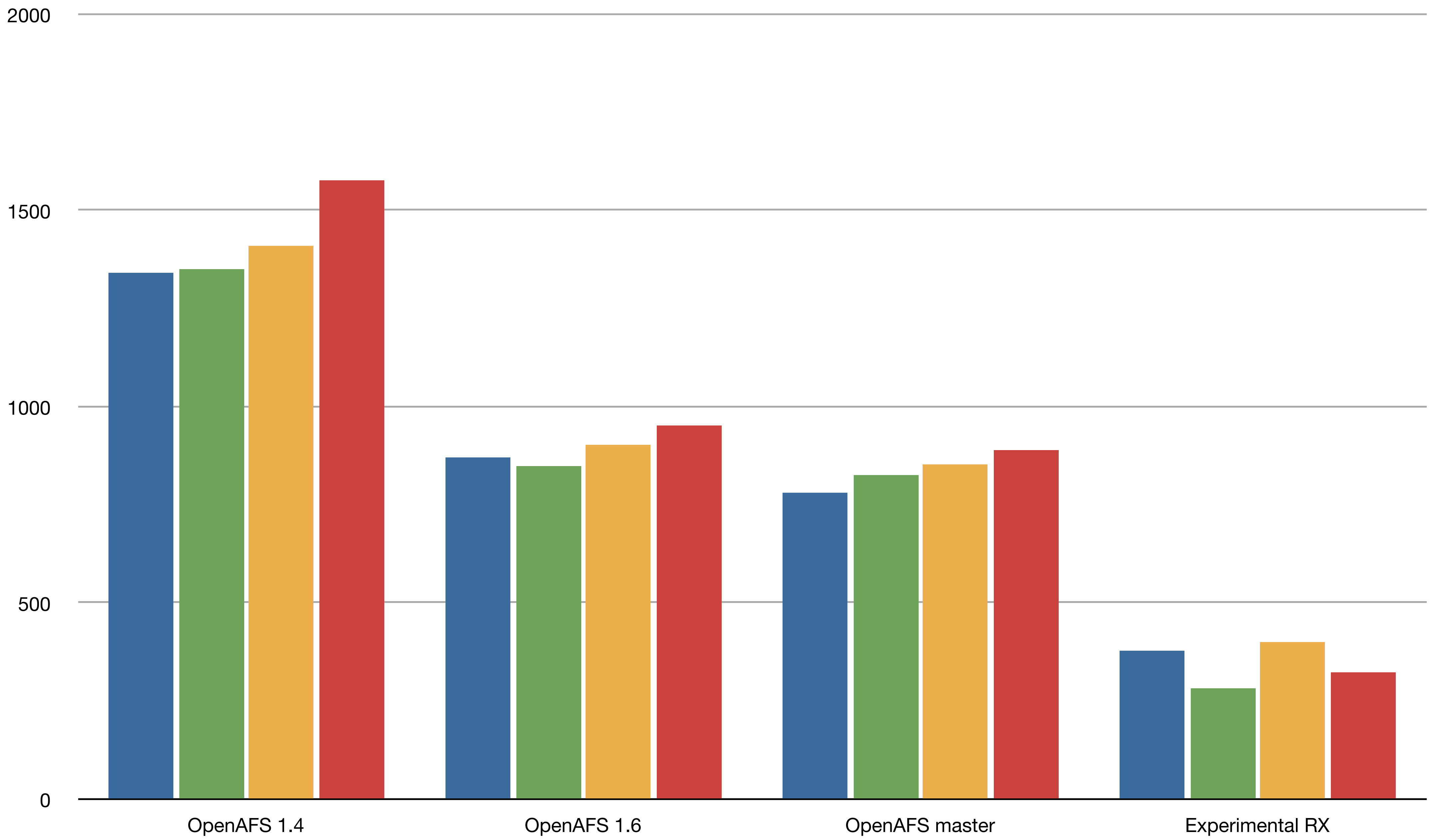


RX: Ooops



Time in ms to perform an RPC with 20Mbyte of data with each OpenAFS version

RX performance work



UDP Stack

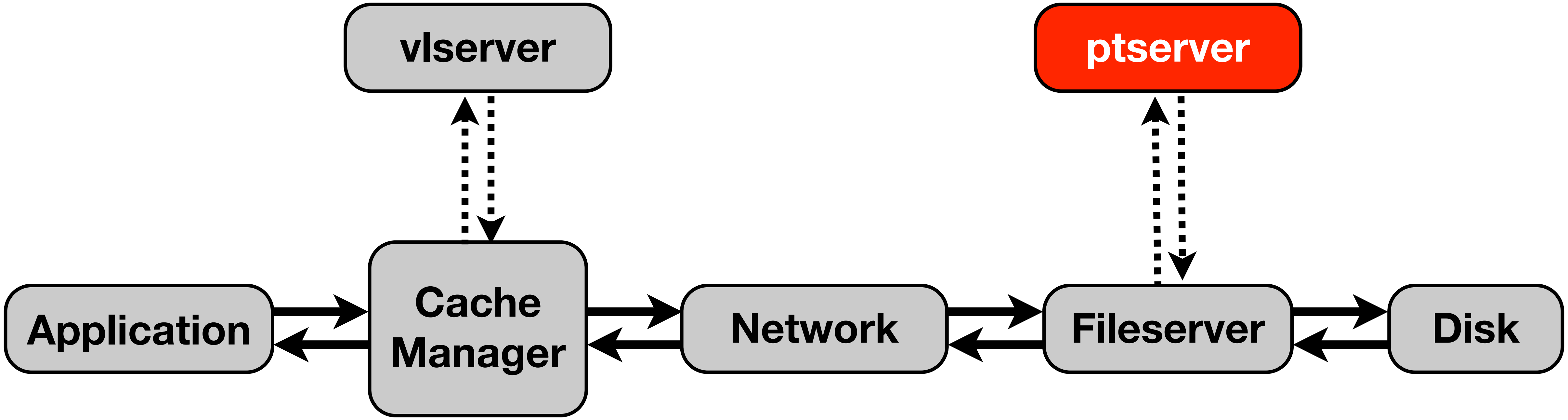
- Don’t run out of packets
- 30 simultaneous clients moving 1Mbyte of data each is enough to swamp Linux’s default UDP buffer size

```
[magrathea]sxw: netstat -su
Udp:
    7200071 packets received
    123 packets to unknown port received.
    3283 packet receive errors
    7194192 packets sent
    RcvbufErrors: 3283
```

Jumbograms

- Jumbograms send UDP payloads larger than the standard Ethernet MTU
- Now turned off by default - it broke on too many networks
- **BUT** - fragmented packets can actually be faster
- Also provides a way of exploiting larger MTUs

The life of a request



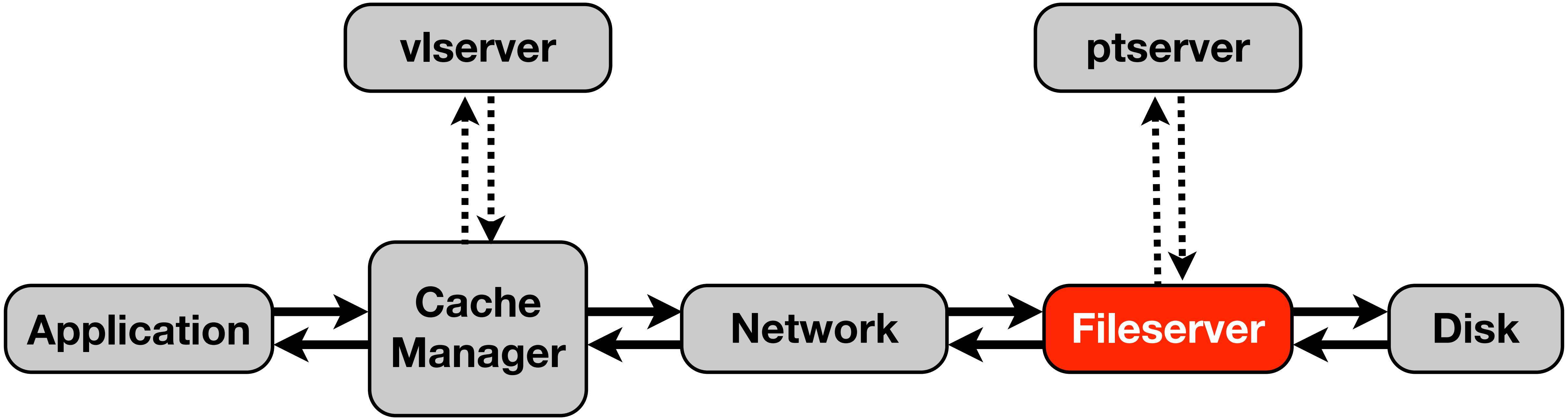
YourFileSystem

ptserver performance

- Fileserver contacts the ptserver with every new connection
- Until the ptserver responds, fileserver thread is blocked
- There aren’t many fileserver threads
- Be *very* careful about ptserver response time, and shutting down ptservers for maintenance

YourFileSystem

The life of a request



YourFileSystem

Fileserver tuning - threads

- Each simultaneous incoming call requires a thread

```
rxdebug  <server>
Trying 192.168.0.1 (port 7000):
Free packets: 3279, packet reclaims: 554, calls: 575230, used FDs: 64
not waiting for packets.
0 calls waiting for a thread
122 threads are idle
0 calls have waited for a thread
```

- OpenAFS 1.6 allows a maximum of 16384 threads (of which 16376 are available for calls)

Ensure you have sufficient callbacks

- Fileserver has a limited amount of space for callbacks, set at run time.
- Check whether you’re running out !

```
./xstat_fs_test -collID 3 -fsname lammasu.inf.ed.ac.uk

             0 DeleteFiles
          1517 DeleteCallBacks
             0 BreakCallBacks
        382707 AddCallBack
             0 GotSomeSpaces
         23265 DeleteAllCallBacks
            33 nFEs
           192 nCBs
        100000 nblks
          7327 CBsTimedOut
             0 nbreakers
             0 GSS1
             0 GSS2
             0 GSS3
             0 GSS4
             0 GSS5
```

UDP buffers

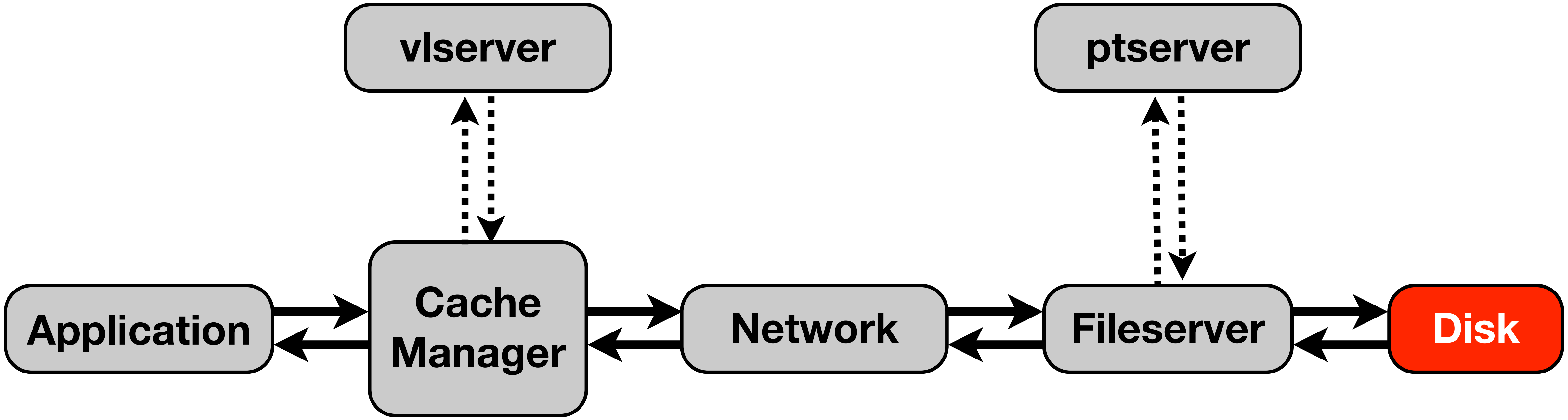
- Make sure that the `-udpsize` parameter is big enough!
- Don’t worry about `-rxpcks` - we autotune this now

Abort threshold

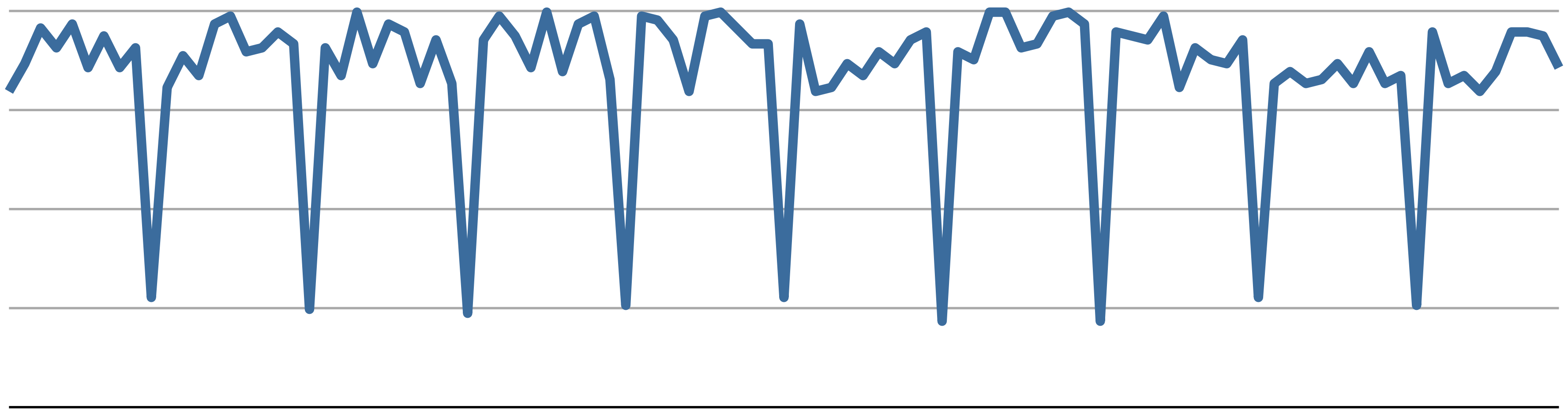
- Fileserver protects itself against misbehaving clients
- If a client sends more than a configured number of failed requests it is throttled
- Very easy to be throttled by doing, for example, ls on a directory you don’t have permission for
- -abortthreshold controls this

YourFileSystem

The life of a request



Journalling woes



Dodgy RAID

- Poor performance RAID arrays can have big effects on fileserver performance
- In particular, RAID 5 is evil

Tuning your OS

- Normal operating system tuning for high speed I/O applies
- Memory not used by the fileserver will be used for page cache - the more the merrier!

Questions