fs123

An open-source, network filesystem with pervasive caching

Michael Fenn
D. E. Shaw Research
LISA 2019
October 28, 2019

Roadmap

- **Motivation**
- Software Design
- Consistency
- Implementation
- Observations
- Extensions
- Future work
- Acknowledgements

Motivation

- DESRES has a big software repository invoked via a module-like system
  - Centralized software repos are common pattern for research computing environments
  - Ours is called “the garden”
  - CentOS 7 version is 1.1 TiB with 8.2 M files
- Traditionally we’ve used NFS for the repo
  - Distributed the garden to many NFS servers via rsync
- NFS has challenges
  - Poor performance for PYTHONPATH searches
  - Single server per mount instance
  - Poor user experience during maintenance

Motivation – NFS performance

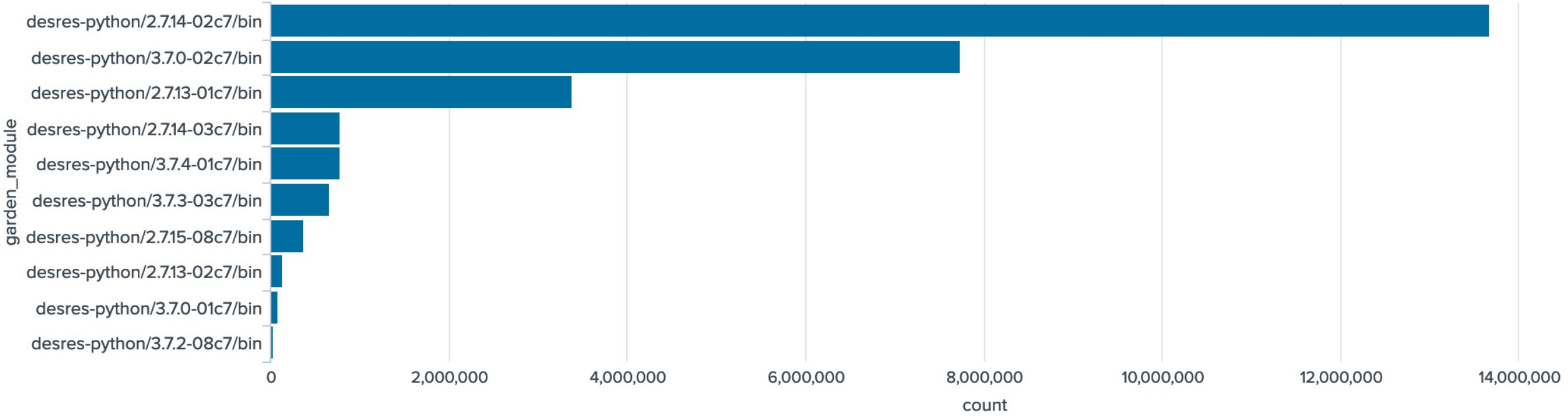
NFS

```
$ time garden with -m desres-python/3.7.4-03c7/bin python3 -c
'import scipy; import pandas'

real                0m7.224s
```

- NFS caches time out quickly
- Most interactive lookups are effectively uncached
- Therefore, PYTHONPATH searches are slow

Motivation – Is Python really that important?

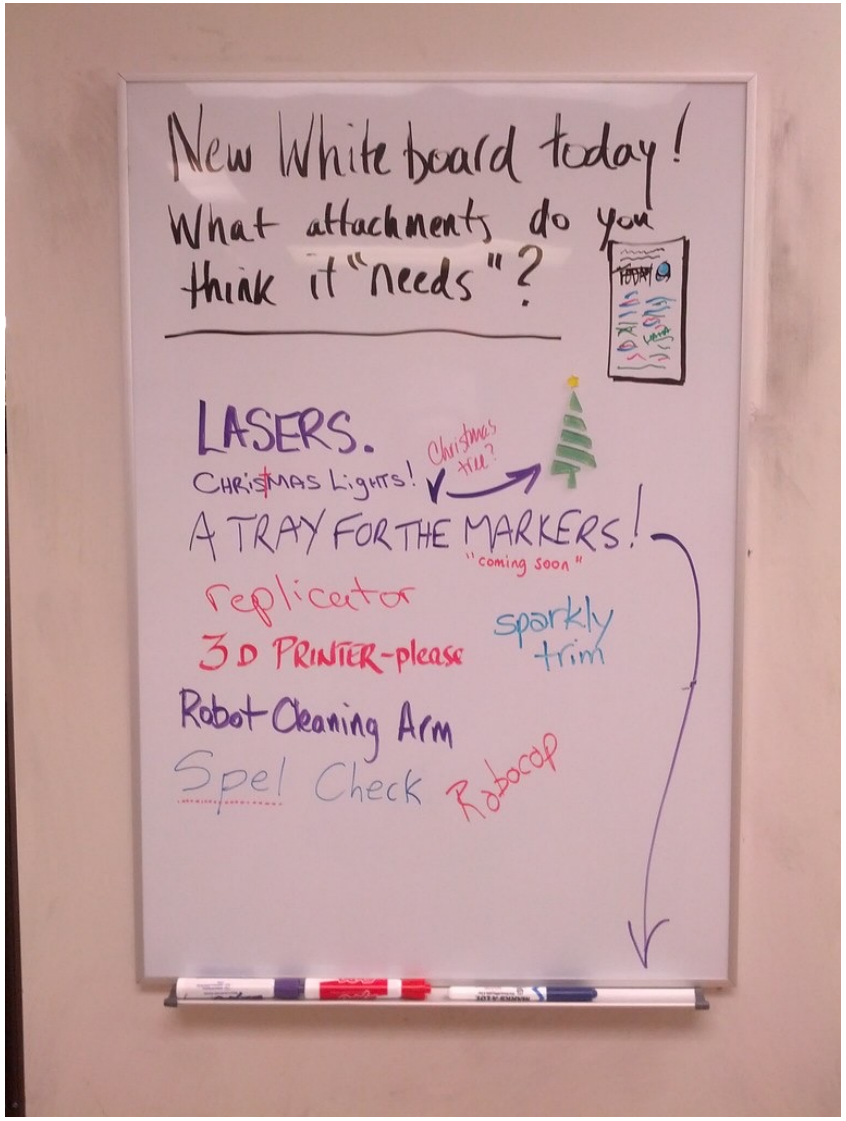
We've established that Python on NFS is slow, but does it matter?



With O(25M) invocations per week, slow Python startup adds up

Motivation – Requirements

- Needs
  - Distribute a software repository to clients
    - Clients are themselves distributed across US offices and data centers
    - O(10K) clients
  - A POSIX-like interface
    - Python, the loader, etc. only know how to read files
  - Some way to improve Python startup performance
    - Caching ENOENT is a good start
- Wants
  - Well-defined, configurable consistency semantics
  - HTTP-based protocol
    - Don't reinvent the wheel
    - Use high-quality open source proxies, load balancers, etc.
  - Pervasive caching support
    - Standard HTTP Cache-Control headers
  - Client-local and shared caches
  - Offline/disconnected operation



fs123 performance preview

NFS

```
real  0m7.224s
```

fs123 (remote-cache)

```
$ time garden with -m desres-python/3.7.4-03c7/bin python3 -c
'import scipy; import pandas'

real              0m2.428s
```

fs123 (fully-cached)

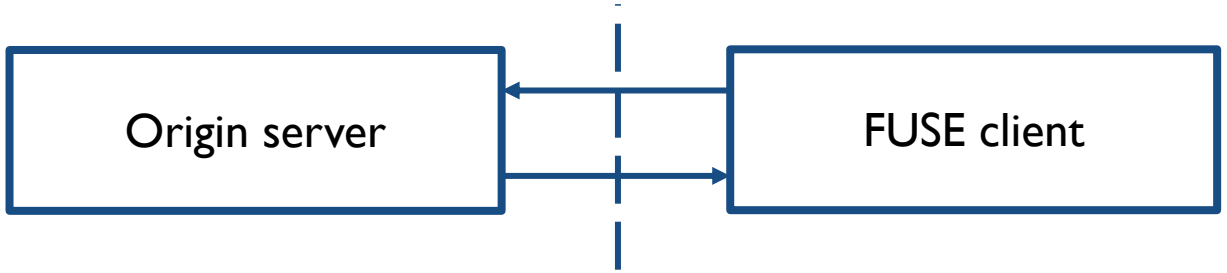
```
$ time garden with -m desres-python/3.7.4-03c7/bin python3 -c
'import scipy; import pandas'

real              0m0.549s
```

Roadmap

- Motivation
- **Software Design**
- Consistency
- Implementation
- Observations
- Extensions
- Future work
- Acknowledgements

Software Design – What is fs123?

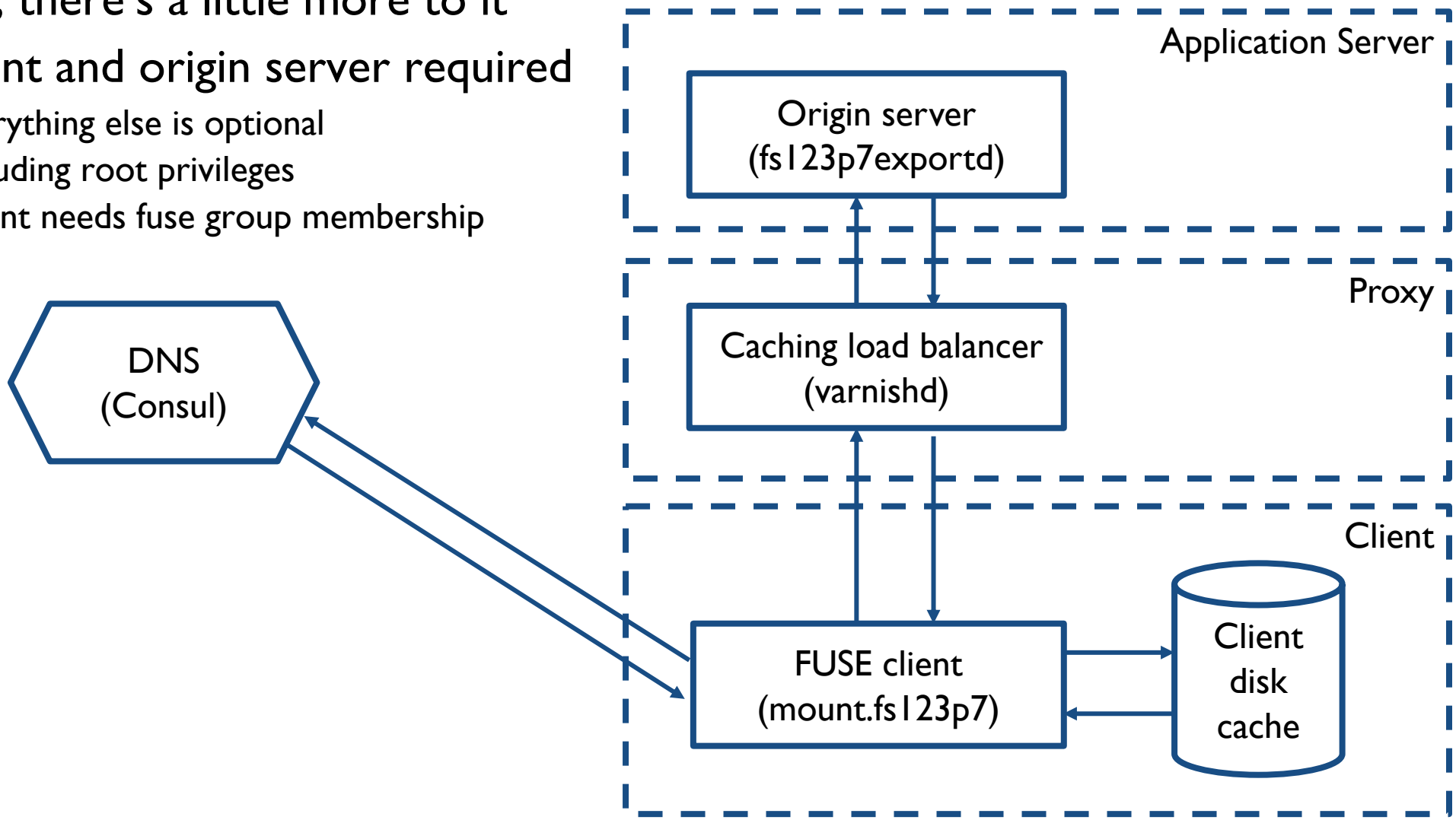
- What does fs123 do?
  - Read-only distributed POSIX filesystem
- How does it do it?
  - Loosely-coupled client-server protocol built on HTTP
  - Client implements a Filesystem in USErspace (FUSE) filesystem
  - HTTP origin server exports a backend POSIX filesystem



- That's it!

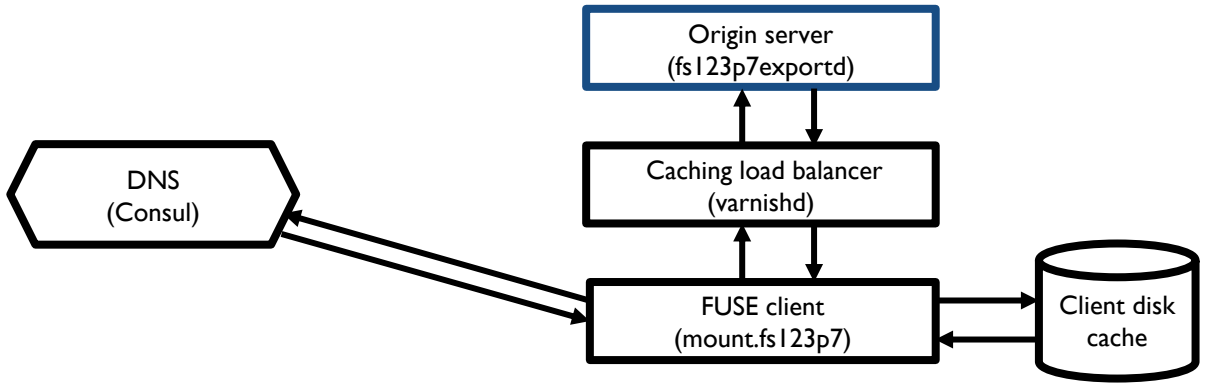
Software Design – Architecture

- OK, there's a little more to it
- Client and origin server required
  - Everything else is optional
  - Including root privileges
  - Client needs fuse group membership



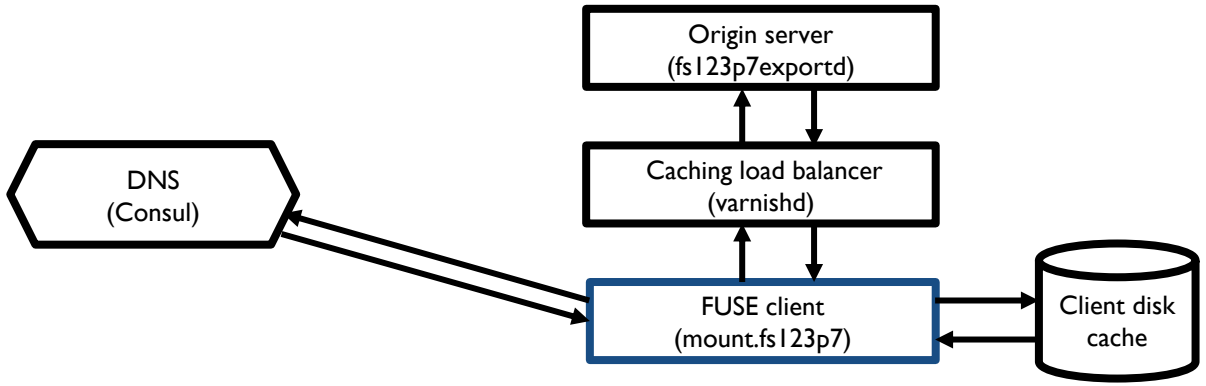
Software Design – fs123p7exportd

- Standard HTTP/1.1 server
- Presents an underlying POSIX filesystem as an fs123 export
  - 1:1 mapping between backend and exported files
- Adds Cache-Control headers based on metadata
- Written in C++ using libevent via evhttp
- Multi-threaded design
  - Each thread runs its own event loop
  - The kernel round-robin's across threads calling accept(2) on the same fd



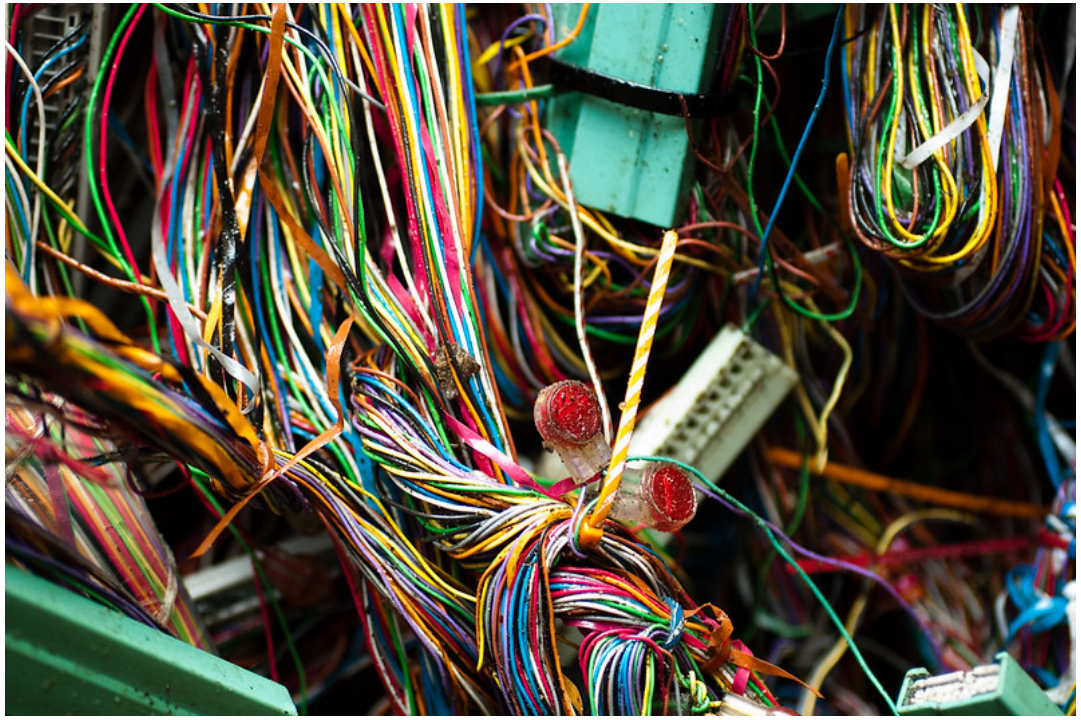
Software Design – mount.fs123p7

- mount.fs123p7 client runs on each compute node
- Written in C++
- Implements a FUSE to fs123 protocol adapter
- Has an (optional) client-local disk cache
- Takes care to pass as much cache timeout information to the kernel as possible
- Standard mount helper command-line interface
  - mount -t fs123p7 http://fs123server/myfs123export /mnt works
  - As does calling mount.fs123p7 directly
  - We start the client via autofs
  - Can be mounted statically or via systemd



Software Design – Well-defined protocol

- mount.fs123p7 – 7400 lines



- fs123p7exportd – 2900 lines
  - /a - attributes
  - /d - directory
  - /f - file read
  - /l - readlink
  - /n - statistics
  - /s - statfs
  - /x - xattr

Software Design – Versioned protocol

- Aside: what's all this "p7" stuff anyway?
- We didn't get the protocol quite right the first time
  - Or the second
  - Or the third
  - …
- It's important to encode the protocol version in the URL, e.g.
  - /myfs123export/7/2/fs123/a/path/to/file
  - 7 is the major protocol version
  - 2 is the minor protocol version
- Encoding the version in the URL allows a virtual hosting server (e.g. Apache httpd) to dispatch requests to different origin servers
- Version 6 might be (and was) served by a different binary than version 7

Roadmap

- Motivation
- Software Design
- **Consistency**
- Implementation
- Observations
- Extensions
- Future work
- Acknowledgements

Consistency – Loosely-coupled systems

- Loosely-coupled distributed file systems
  - Good for availability and partition tolerance
  - No single, coherent view of system state
- Need to define a consistency model describing when changes are visible
- Timeouts are the obvious way to do this in a loosely-coupled system
- NFS timeouts are per mountpoint, which have undesirable tradeoffs
  - Long timeouts
    - Reduce server load
    - More client cache hits, lower average latency
    - Clients are slow to notice changes
  - Short timeous
    - Increase server load
    - Fewer client cache hits, higher average latency
    - Clients quickly notice changes
- There's got to be a better way!

Consistency – Tunable consistency

- What if we tell the system more about how we update the hierarchy?
  - /gdn/desres-python/3.7.4-03c7/lib/python3.7/site-packages/pandas
- fs123 allows individual directory trees to have different cache timeouts
- Short timeouts for (meta)data which may change rapidly
  - E.g. Parent directories of garden modules
  - /gdn/desres-python/…
- Long timeouts for data we know will be static
  - E.g. Installed garden modules
  - /gdn/desres-python/3.7.4-03c7/lib/python3.7/site-packages/pandas

Consistency – High availability considerations

- In the NFS model, a mount is to a specific IP address
  - High-end storage systems allow the IP to fail over between nodes
  - However, typically limited to failing over within tightly-coupled cluster
- HTTP makes it easy to have independent origin servers
  - The mount "device" is a URL
  - HA and failover are the norm with HTTP-based services
  - For Internet-facing services, geo-failover is the norm
- Easy to make client requests transparently arrive at another origin server
  - But this is filesystem, and filesystems need to be consistent
  - How do we get independent origin servers to present the same hierarchy?
- Many attributes are easy to sync with rsync
  - Name
  - Contents
  - Mode, owner, group
  - Timestamps
- But what about the inode number?

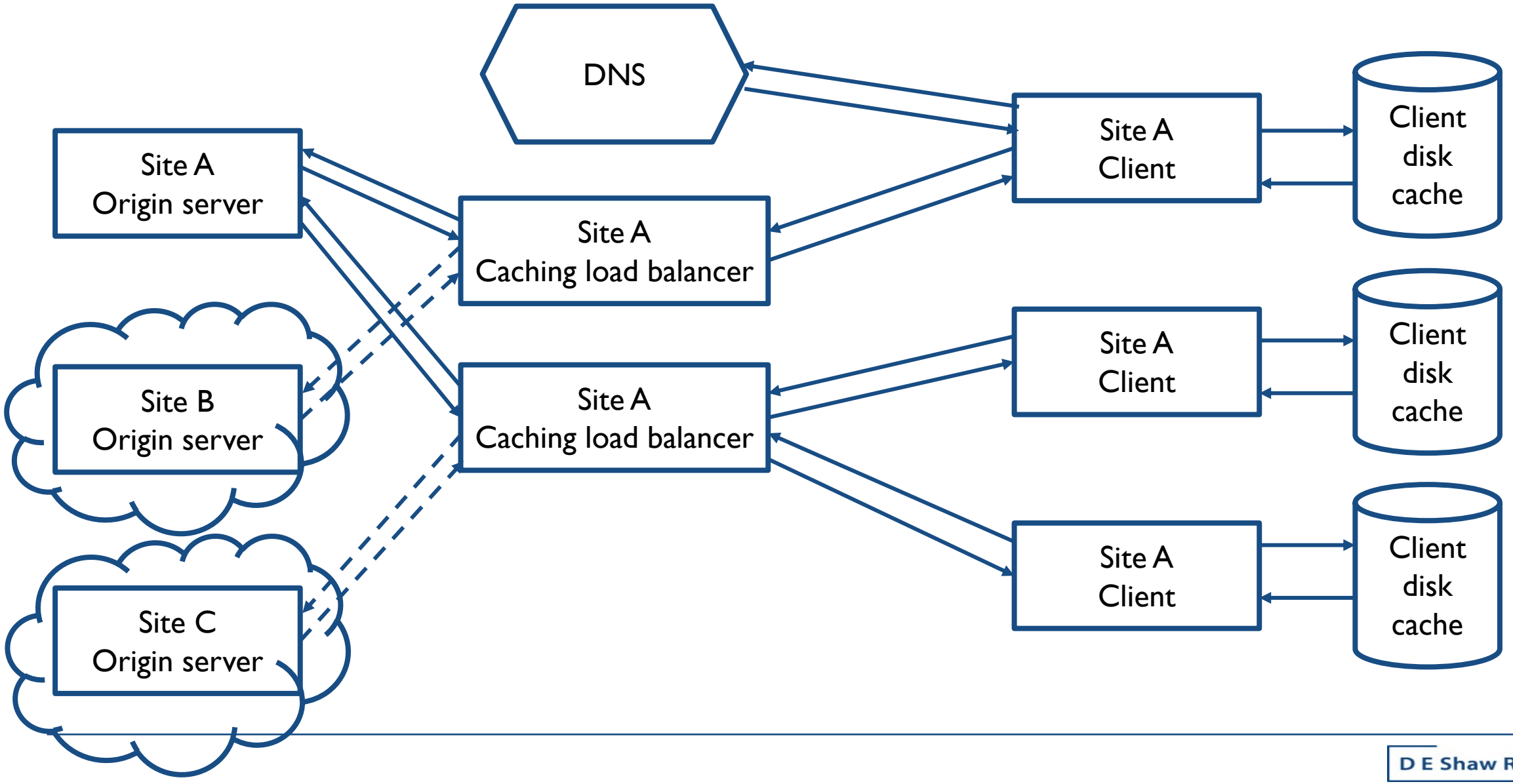
Consistency – Inode numbers

- We need fs123 clients to see the same inode number, even upon failover
- How does the client generate inodes for clients?
  - Use the inode from the backend filesystem? We can't choose that
  - Use a hash based on the name? Doesn't work for rename(2)
  - Fabricate it entirely? Requires the client to keep a record of inodes it used
- Settled on an extended attribute (xattr) on the backend files called estalecookie
- Example xattr
  - `user.fs123.estalecookie="15702125752060249 42"`
- The garden install process adds xattrs to each file written into the backend
- xattr makes its way into an HTTP header which the client uses to generate inodes
- xattrs can be rsync'ed so independent servers can generate the exact same response
- A non-HA or non-POSIX backend could use a different method for generating the estalecookie

Roadmap

- Motivation
- Software Design
- Consistency
- **Implementation**
- Observations
- Extensions
- Future work
- Acknowledgements

Implementation – Architecture



Implementation – Components

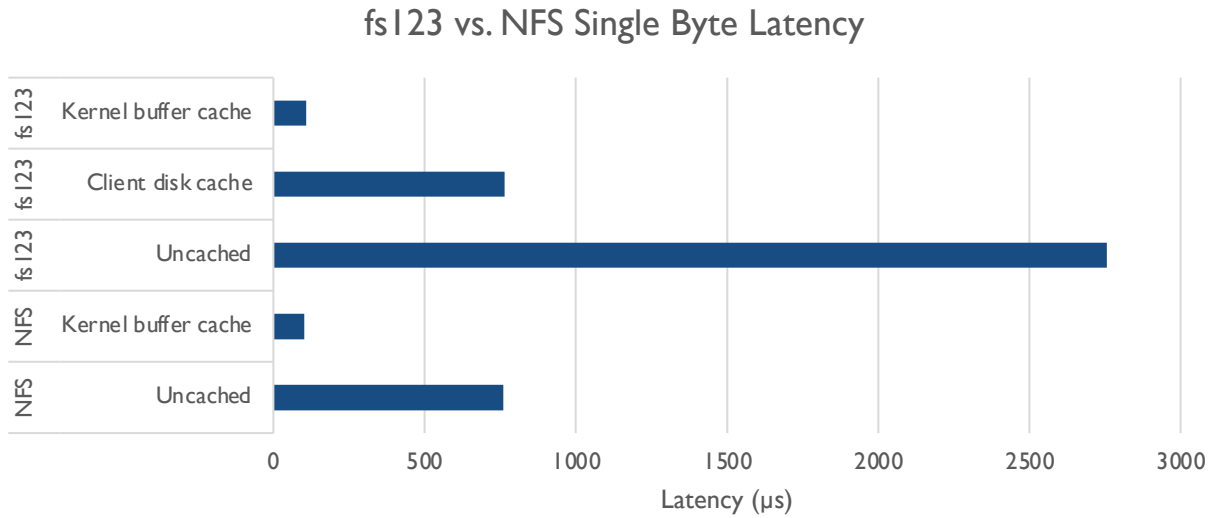
- Use open source components wherever practical
- mount.fs123p7 client
  - HTTP library – libcurl
  - Encryption library – libsodium
  - FUSE library – libfuse
- fs123p7exportd server
  - HTTP server library – evhttp (libevent)
  - Encryption library – libsodium
- External components
  - Load balancer/caching proxy – Varnish
  - URL namespacing – Apache httpd
  - Service discovery – Consul
  - Can be omitted or substituted with compatible tools (e.g. HAProxy, nginx, Route53)

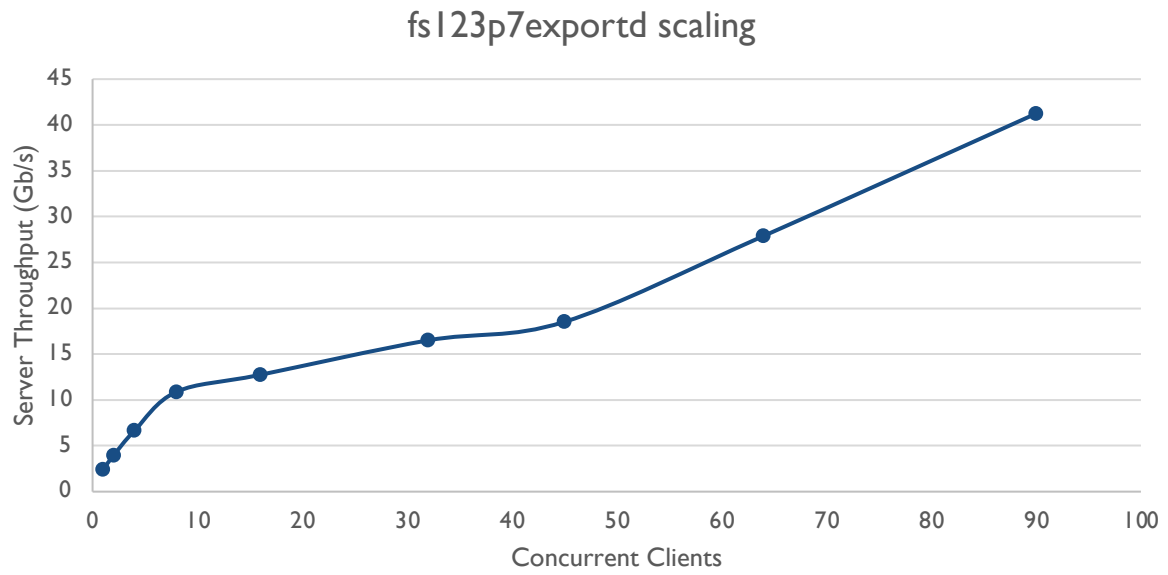
Roadmap

- Motivation
- Software Design
- Consistency
- Implementation
- **Observations**
- Extensions
- Future work
- Acknowledgements

Observations – Performance

- Single byte I/O latency
  - NFS
    - Uncached - 761 µs
    - Kernel buffer cache - 102 µs
  - fs123
    - Uncached - 2,756 µs
    - Client disk cache - 765 µs
    - Kernel buffer cache - 109 µs
- Throughput
  - mount.fs123p7 local caches reads ~700 MB/s
  - fs123p7exportd server network write ~40 Gb/s





Observations – Performance implications

- Context switching into the mount.fs123p7 FUSE client is negligible for most workloads
  - The cost of the actual I/O dominates
- Telling the kernel as much about cache timeouts as possible is a big win
  - The buffer cache is fast, so use it as much as possible
  - The best performance regimes are when mount.fs123p7 FUSE client rarely needs to service an I/O
- The libevent fs123p7exportd server is fast and the client-local cache is robust
  - Perhaps don't need a caching proxy at all
  - However, we still want a load balancer to fail over between origin servers
  - Caching is "free" when using varnish as a load balancer
- Performance and compatibility are good enough to serve /usr
  - Busybox Docker image that bootstraps into an fs123 /usr served from CentOS
  - Caching is good enough to access /usr disconnected most of the time

Observations – Architecture implications

- HTTP is a good foundation for highly-available services
  - We perform client and server upgrades without downtime
  - Routine server version upgrades are a non-event
  - 84 releases with no client impact*
- FUSE daemon is an unexpectedly good way to do non-disruptive client upgrades
  - Rather than needing to reboot or unmount for a new kernel module, we just install the new client binaries
  - New mounts automatically get the new version
  - Lazy unmount (umount -l) the mountpoint and let autofs mount the new version upon request
  - Old client terminates when last open file is closed

* From the upgrade process

Observations – Origin server implementations

- Well-defined fs123 protocol implies that the same client can make requests to multiple server implementations
  - It's one thing to say you have a well-defined protocol
  - But we intended to prove it
- fs123p7exportd server exports an underlying POSIX filesystem
  - There's no reason the export must be of physical files
  - Currently have two additional fs123 server implementations

Roadmap

- Motivation
- Software Design
- Consistency
- Implementation
- Observations
- **Extensions**
- Future work
- Acknowledgements

Extensions – Multi-Archive File System (MAFS)

- What if the presented hierarchy didn't correspond exactly to the backend filesystem?
- MAFS exports a virtual hierarchy of tar archives
- Tar archives are parsed on ingest and metadata stored in LevelDB
- Backend filesystem contains
  - tar archives
  - the LevelDB
  - journal of all write operations
- fs123-exported virtual hierarchy with expanded view of archives
- Underlying filesystem has large files
  - Reduced inode usage
  - Higher performance migration between servers and to tape

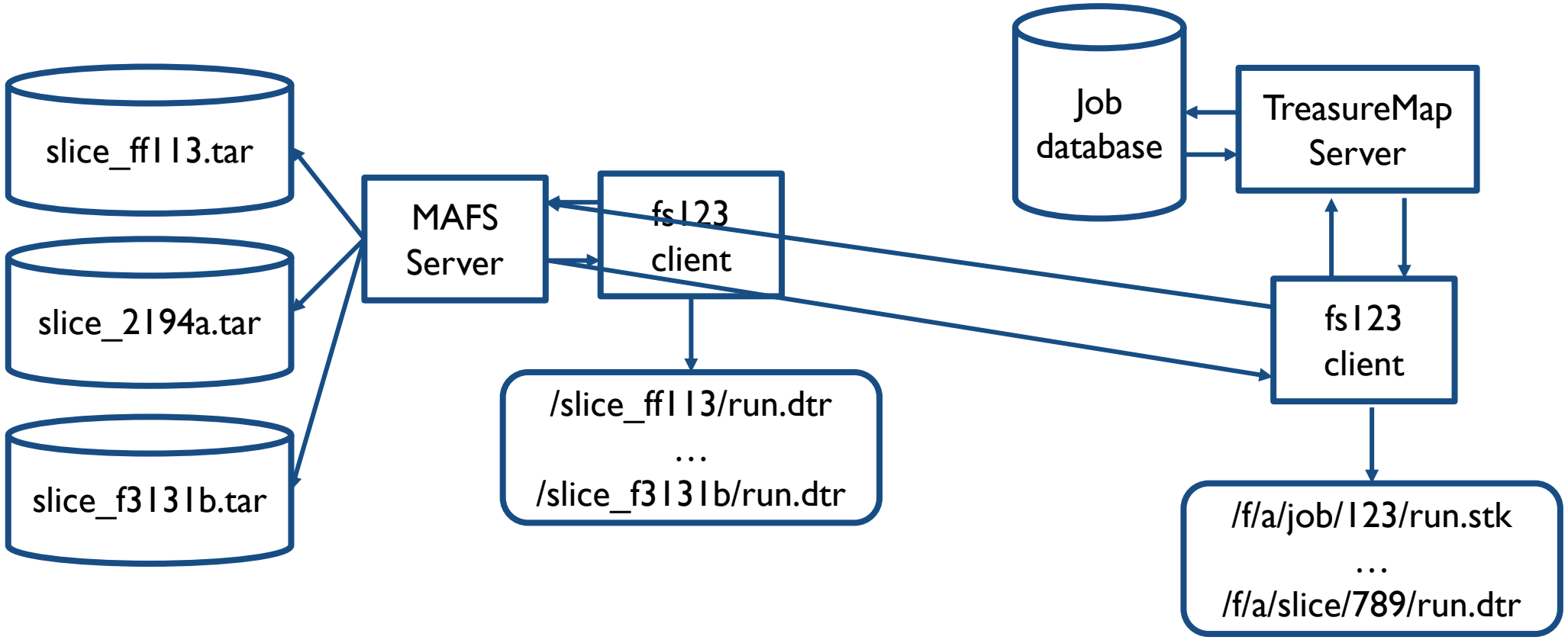
Extensions – TreasureMap

- What if the hierarchy presented by the server had no backend files?
- TreasureMap presents a virtualized view of compute jobs for clients
- View constructed by querying a PostgreSQL job tracking database
- Allows infrastructure to abstract the physical location of job outputs while providing a POSIX view
- Not in the data path in the general case
  - Issues of HTTP 302 redirects to ask the client to query the underlying MAFS server
  - The 302 "just worked" because the client uses libcurl

Extensions – Bringing it all together

Multi-Archive File System (MAFS)

TreasureMap



Roadmap

- Motivation
- Software Design
- Consistency
- Implementation
- Observations
- Extensions
- **Future work**
- **Acknowledgements**

Future work

- Cooperative client caching
  - In a batch compute cluster, clients typically all request the same data
  - Can reduce server load and north-south network bandwidth via cooperative client-side caching
  - Torrent or multicast-based approach?
- Improve offline/disconnected operation
  - For mobile clients, it would be useful to short-circuit a network operation we know will fail
  - Currently have a manual switch
  - Heuristics to detect long-term disconnection?
- Framework for writing origin servers
  - Each origin server currently uses a slightly different codebase, should be unified for simplicity
- Object store-backed origin server
  - Would make it easy to use S3 or an on-premises object store as a backend
- Write path in the protocol
  - Writes currently occur out of band via rsync or a simple upload server

fs123 is open source!

- https://github.com/DEShawResearch/fs123
- BSD 3-clause license
- We update it from our internal repository on a regular basis
- Contributions welcome!

Acknowledgements

- D. E. Shaw Research
  - John Salmon
  - Mark Moraes
  - Peter Skopp

Questions?

Backup