Speeding up Terraform caching with OverlayFS

Ricard Bejarano
Site Reliability Engineer, Cisco

Ricard Bejarano

Site Reliability Engineer Technical Leader
Cisco

Ricard Bejarano

Site Reliability Engineer Technical Leader
Cisco

120k+ Terraform resources

600+ engineers

100’s of operations per minute

bejarano.io/terraform-overlayfs

Speeding up Terraform caching with OverlayFS

Published Apr 13, 2025 by Ricard Bejarano

The Terraform plugin cache does not support↗ concurrent `terraform init` runs.

This is a **massive inefficiency** for Terraform users past a certain scale, since all we can do is **disable caching** or **serialize all inits**, both of which are suboptimal past a certain number of providers or concurrent plan/apply operations, respectively.

From what I could find in Terraform's circles, this is a known problem↗ and there's intent of fixing it↗, but it's complicated↗. And since it seems like we're a long way from a native solution, we had to **get creative**.

Introducing OverlayFS

OverlayFS↗ is a Linux filesystem which combines the contents of multiple read-only directories with a writable layer on top, into a single volume.

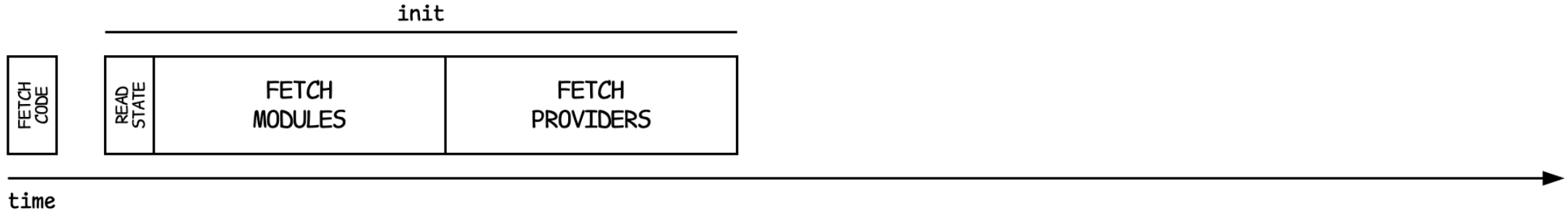

time

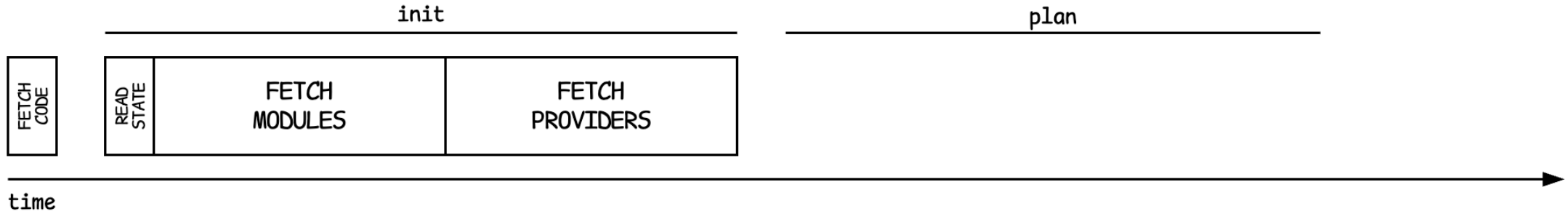

FETCH
CODE
time

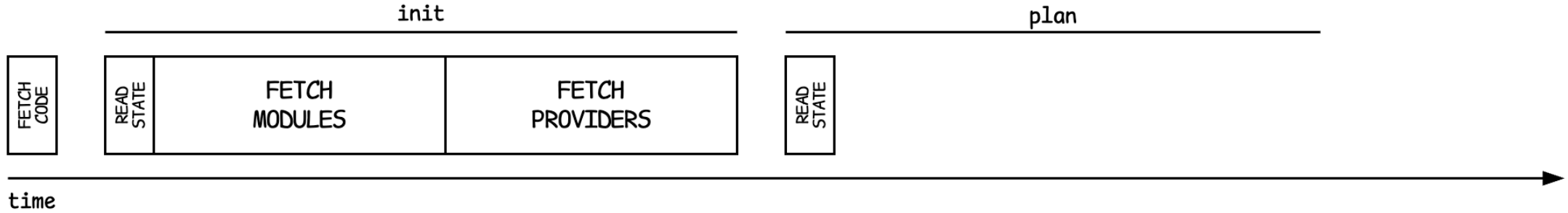

init
FETCH CODE
time

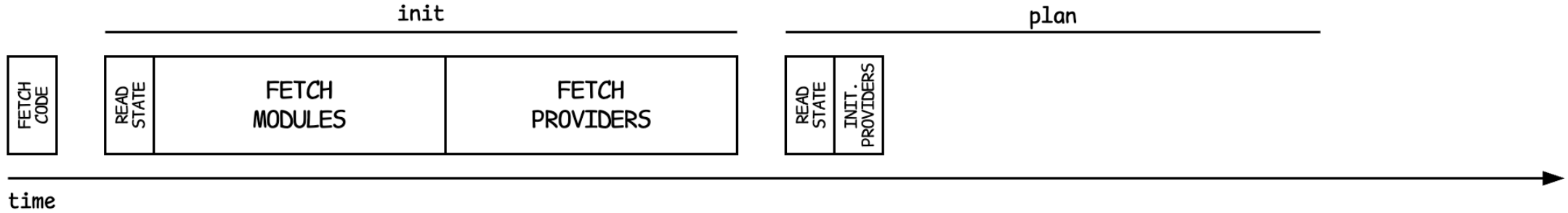

init
FETCH CODE
READ STATE
time

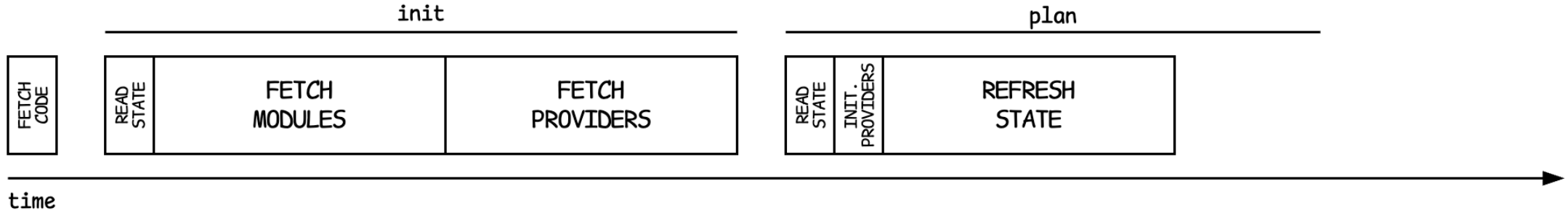

init
FETCH CODE
READ STATE
FETCH MODULES
time

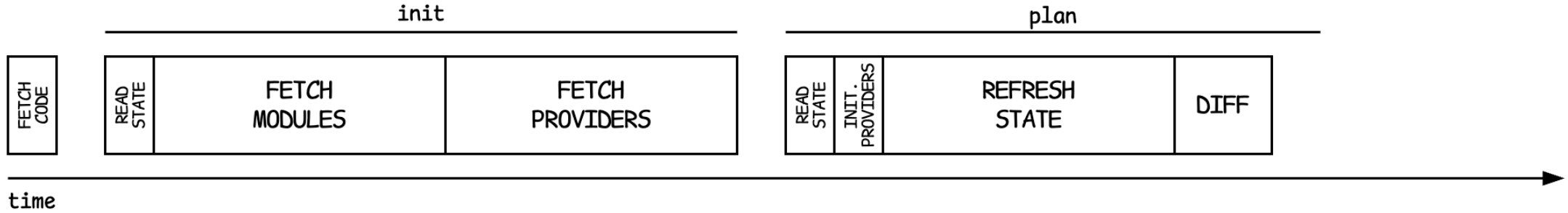

init
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
time

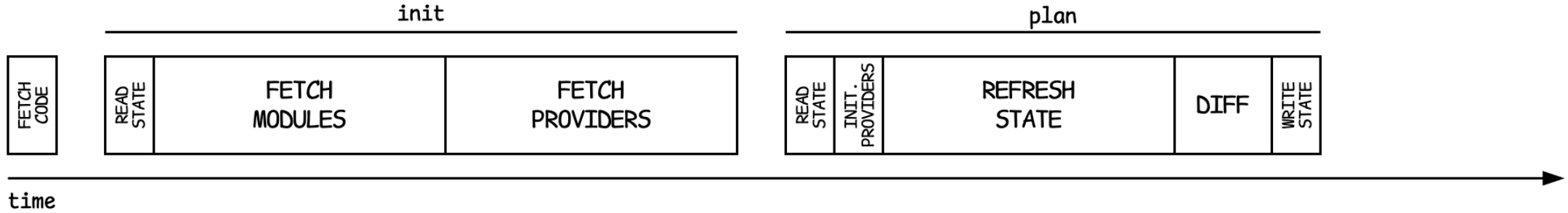

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
time

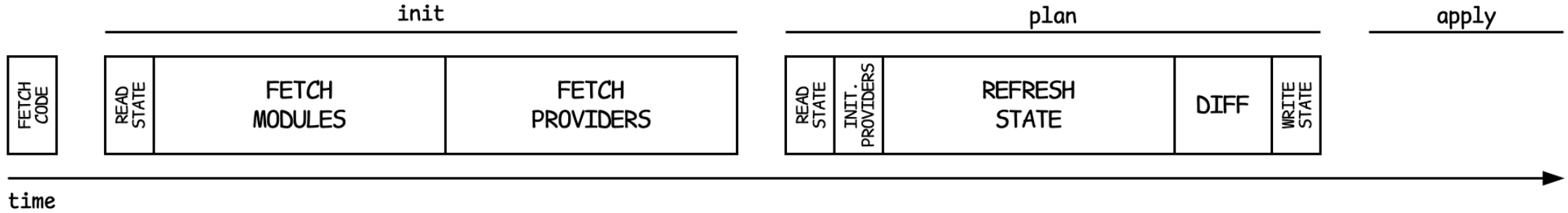

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
time

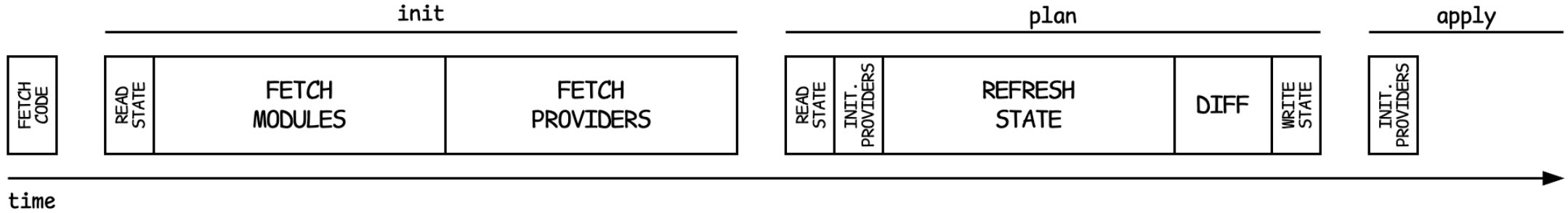

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
time

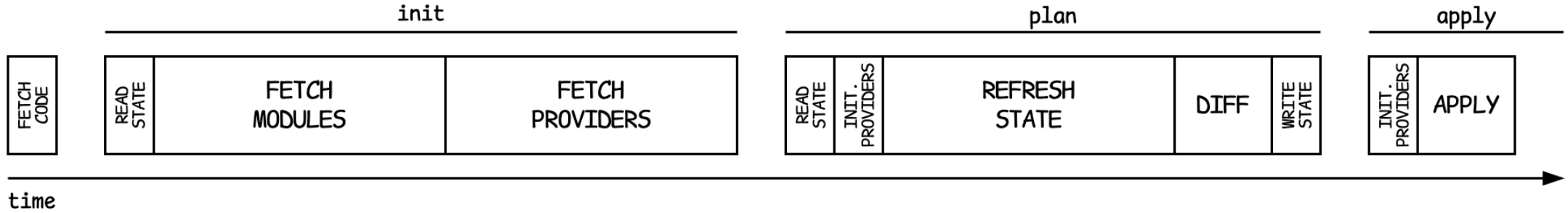

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
time

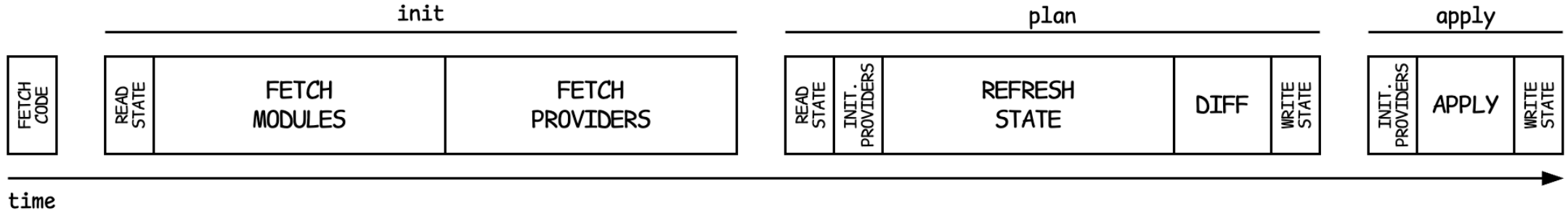

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
time

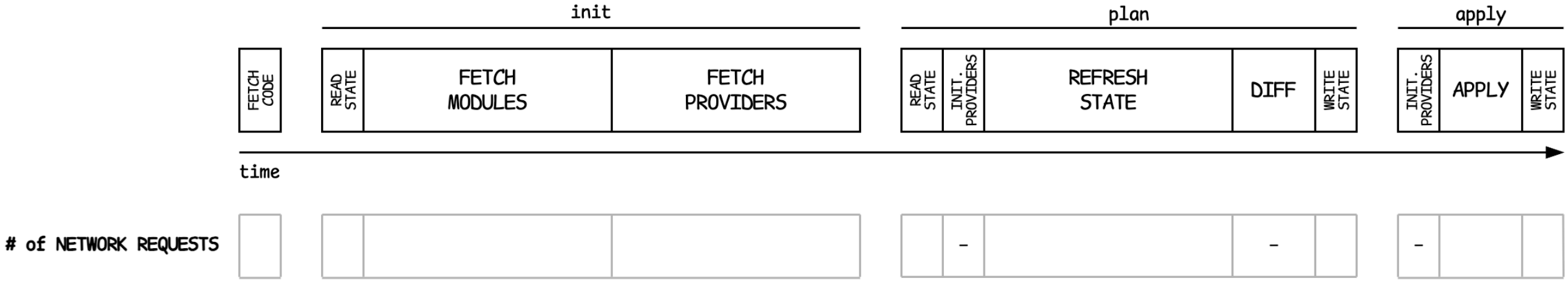

init
plan
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
time

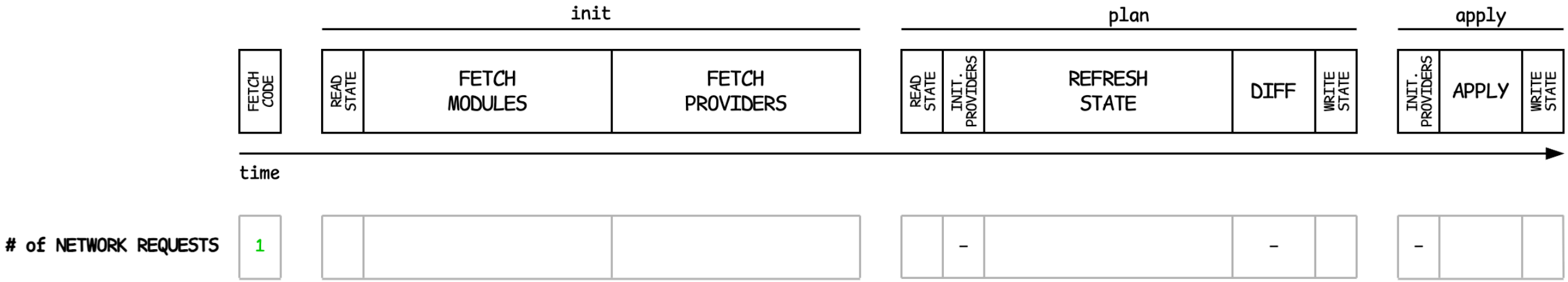

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
time

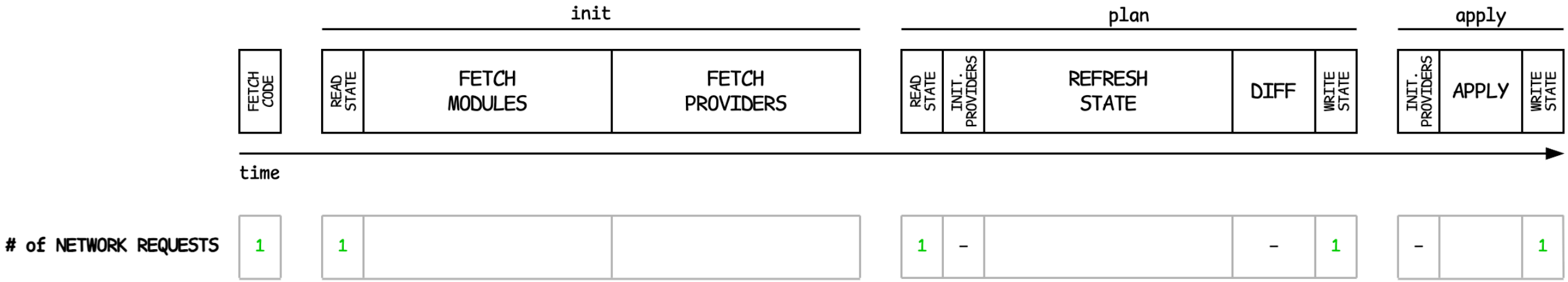

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
time

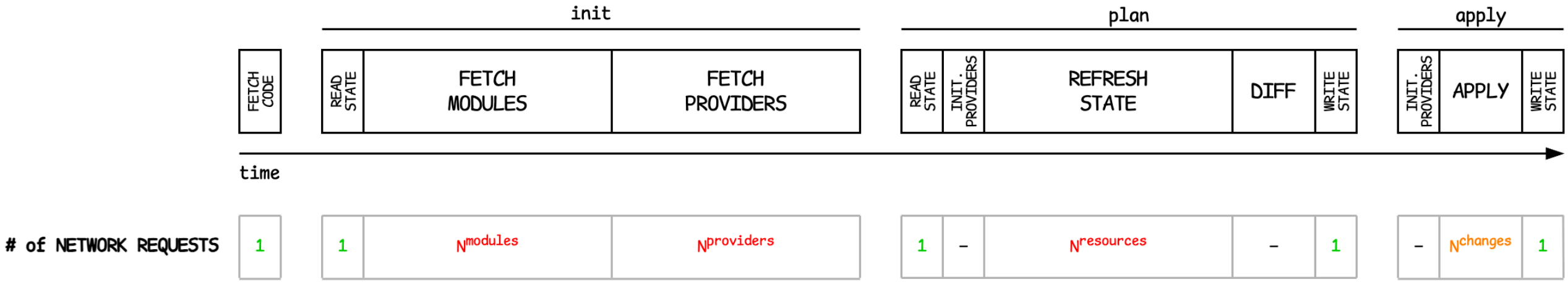

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
time

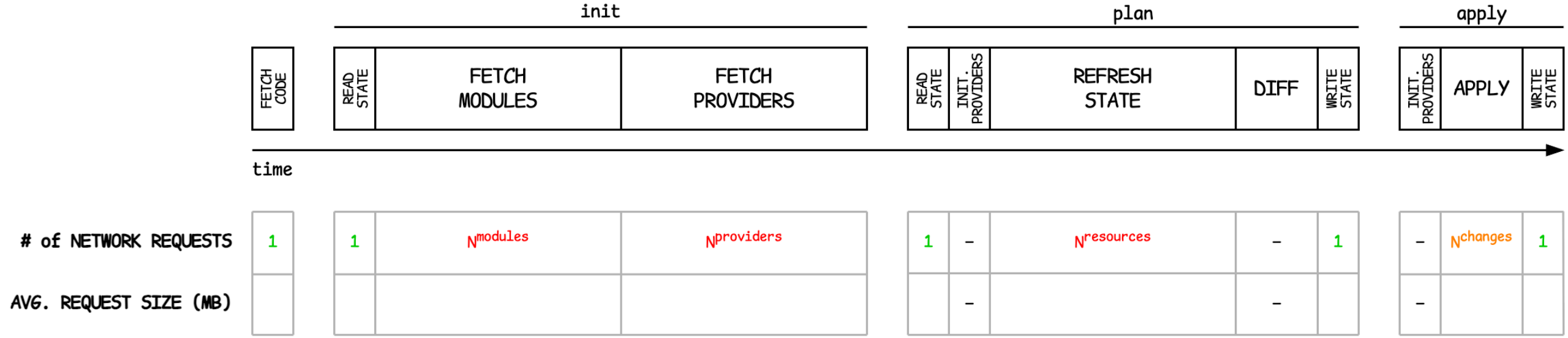

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time

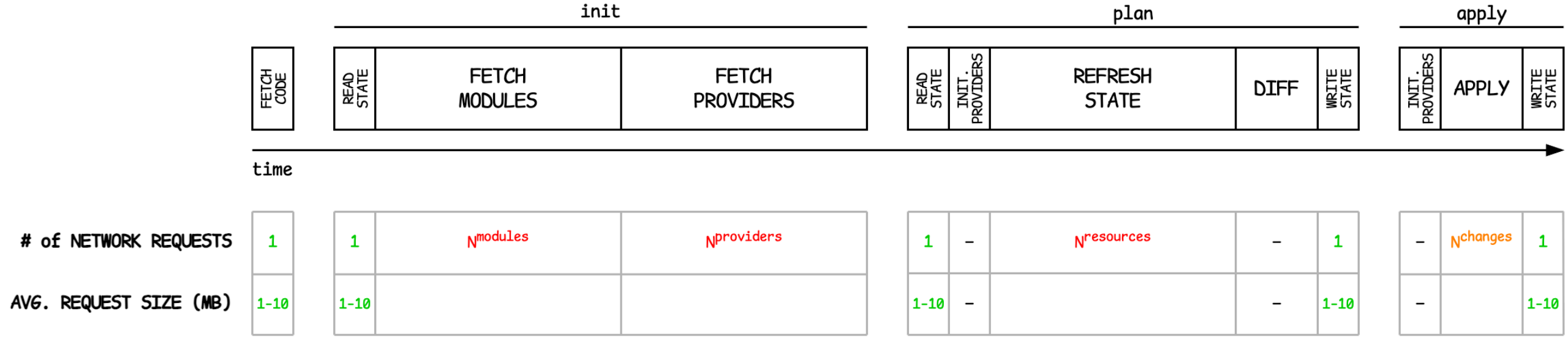

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
-
-
-

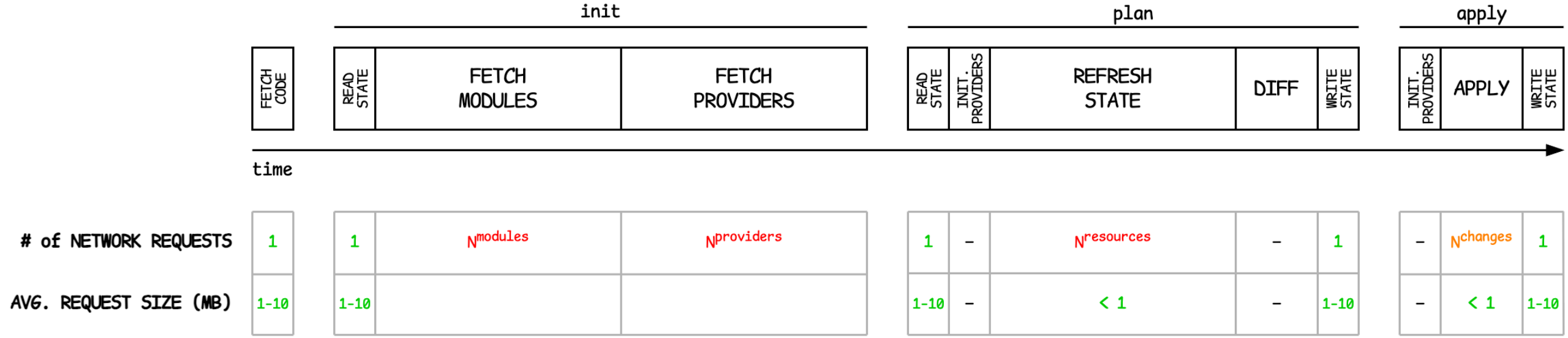

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
-
-
-

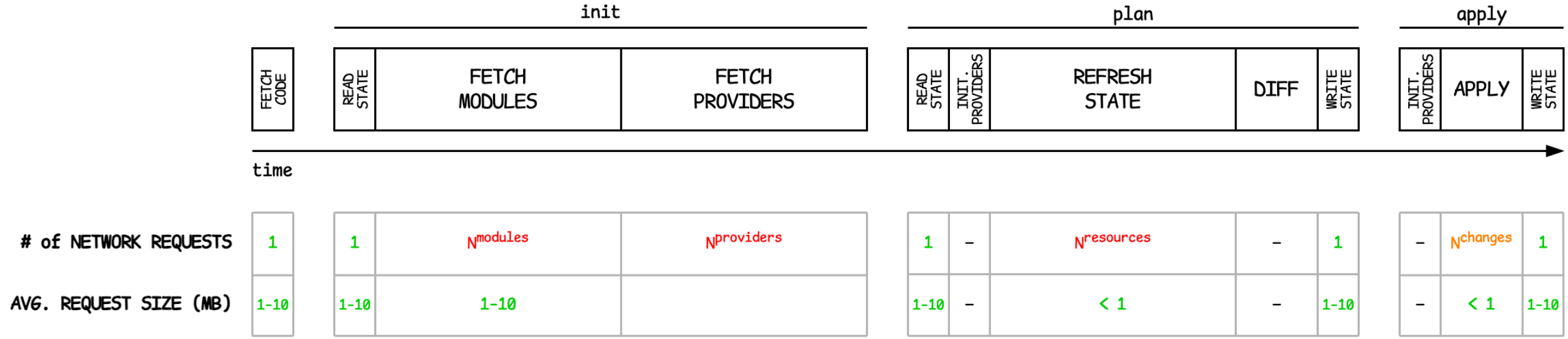

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
1
-
-
1
-
1


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1

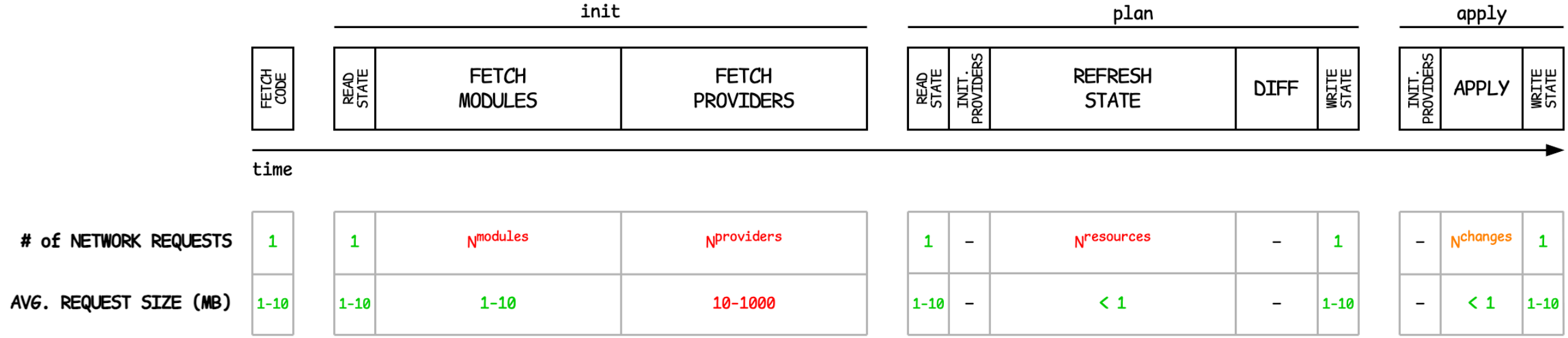

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
–
–
–


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
–
–
1-10
–
1-10

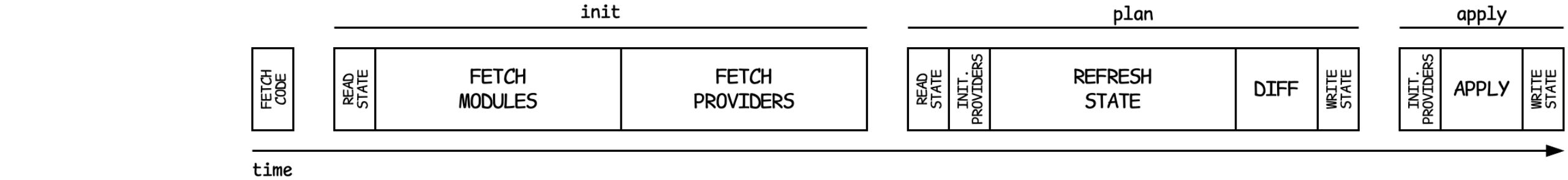

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
–
< 1
–
1-10
–
< 1
1-10



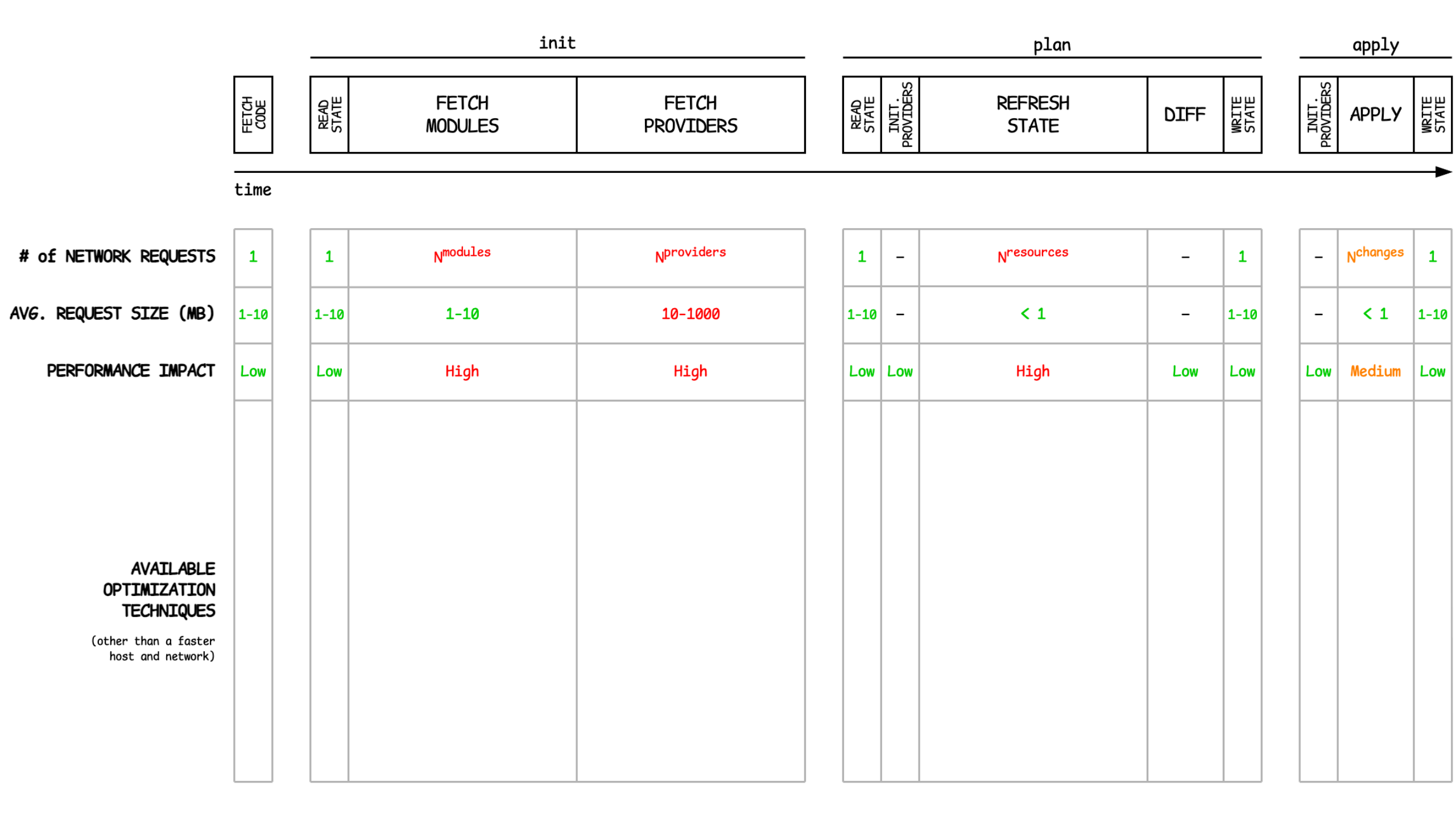
	FETCH CODE	READ STATE	FETCH MODULES	FETCH PROVIDERS	READ STATE	INIT. PROVIDERS	REFRESH STATE	DIFF	WRITE STATE	INIT. PROVIDERS	APPLY	WRITE STATE
# of NETWORK REQUESTS	1	1	$N^{modules}$	$N^{providers}$	1	–	$N^{resources}$	–	1	–	$N^{changes}$	1
AVG. REQUEST SIZE (MB)	1-10	1-10	1-10		1-10	–	< 1	–	1-10	–	< 1	1-10

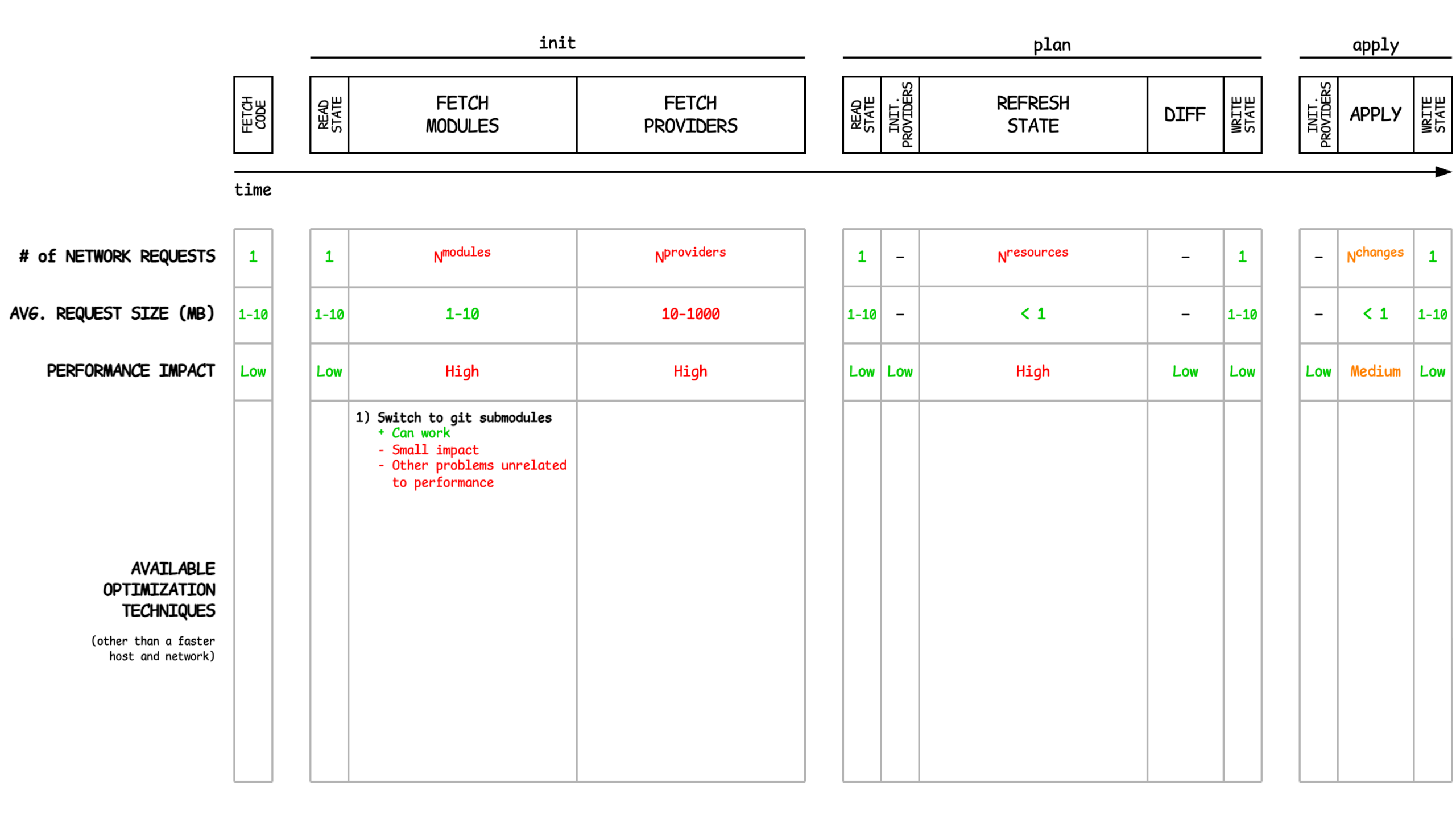


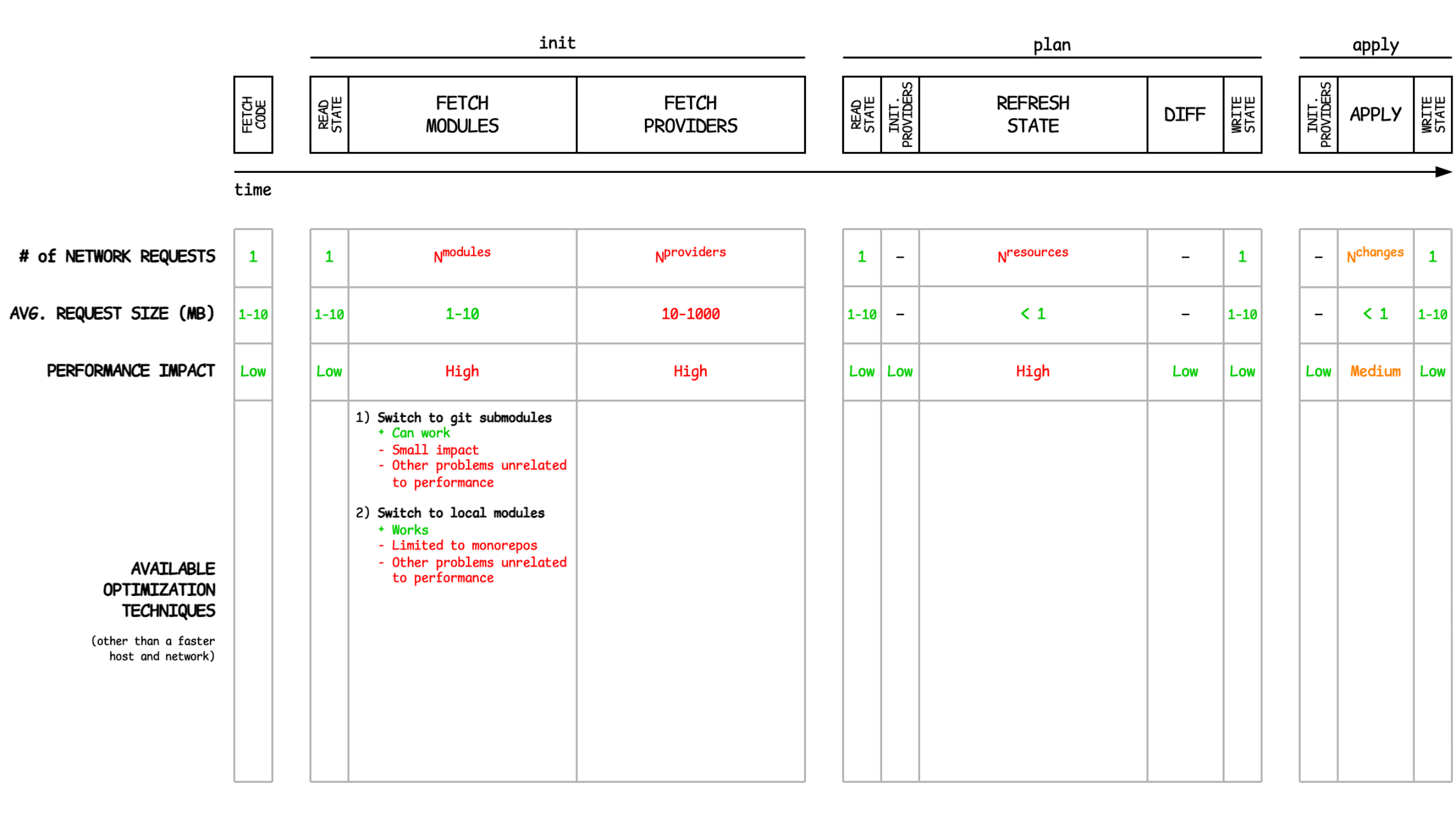
	FETCH CODE	READ STATE	FETCH MODULES	FETCH PROVIDERS	READ STATE	INIT. PROVIDERS	REFRESH STATE	DIFF	WRITE STATE	INIT. PROVIDERS	APPLY	WRITE STATE
# of NETWORK REQUESTS	1	1	$N^{modules}$	$N^{providers}$	1	–	$N^{resources}$	–	1	–	$N^{changes}$	1
AVG. REQUEST SIZE (MB)	1-10	1-10	1-10	10-1000	1-10	–	< 1	–	1-10	–	< 1	1-10

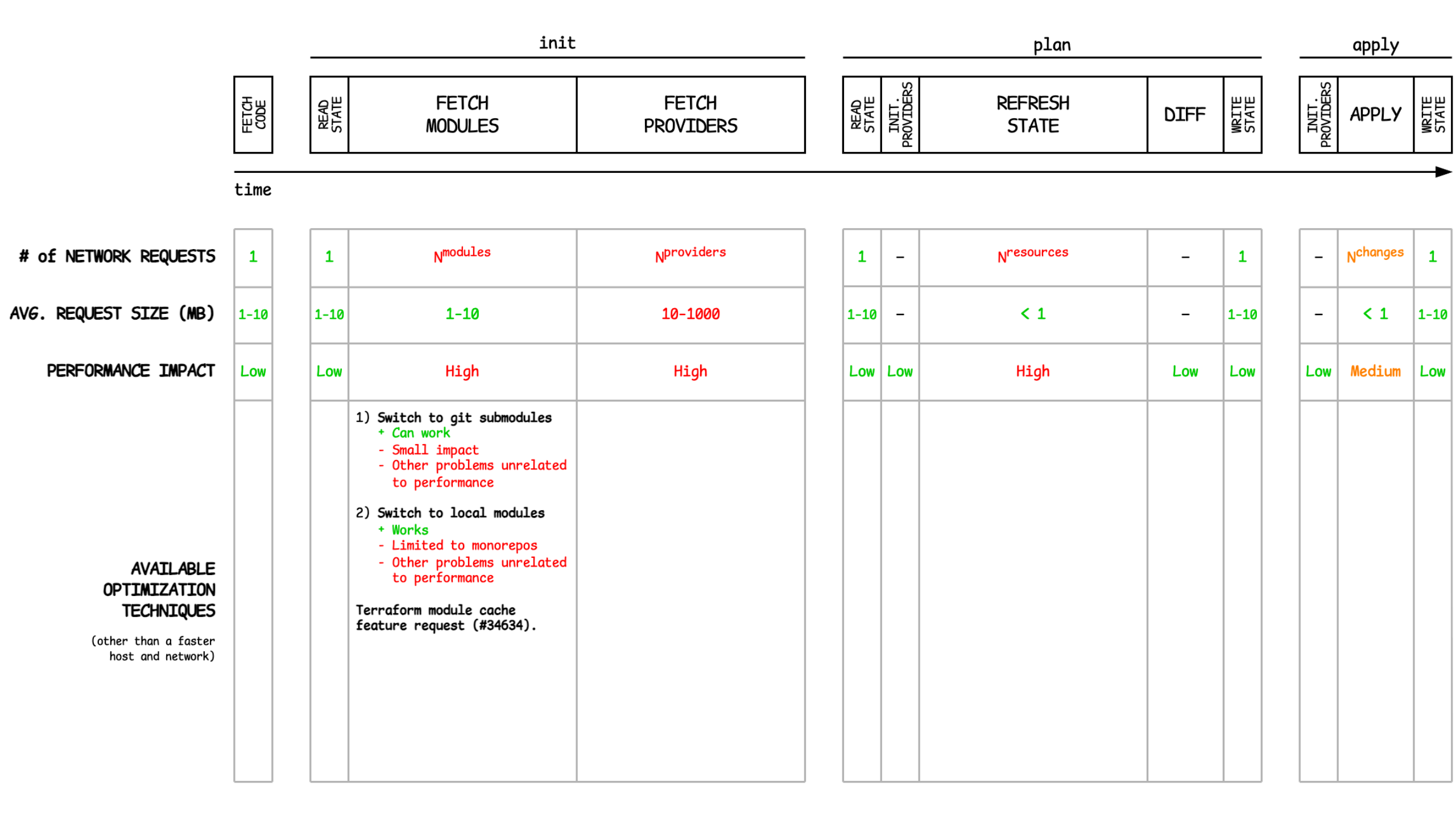


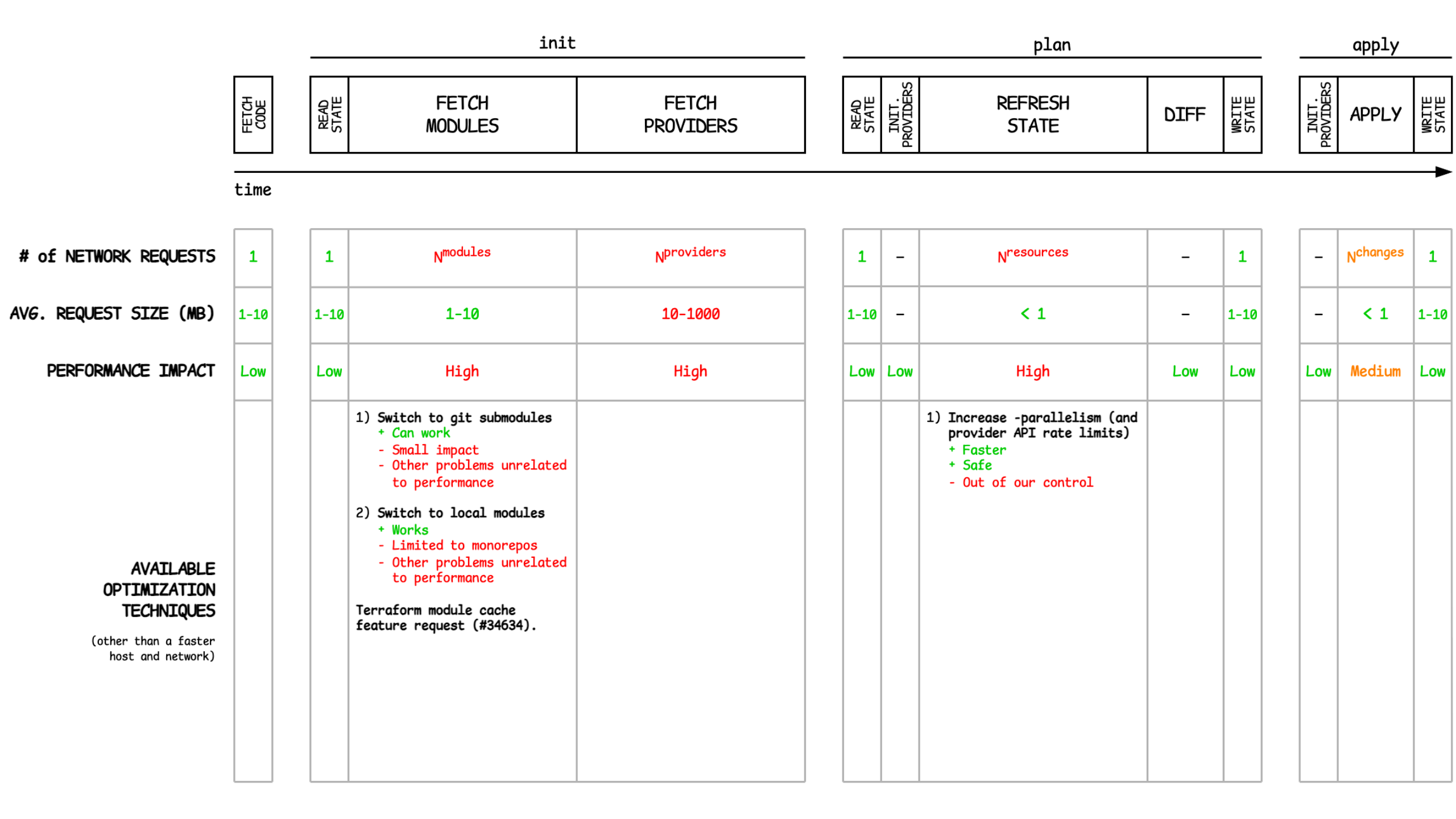
# of NETWORK REQUESTS	1	1	$N^{modules}$	$N^{providers}$	1	–	$N^{resources}$	–	1	–	$N^{changes}$	1
AVG. REQUEST SIZE (MB)	1-10	1-10	1-10	10-1000	1-10	–	< 1	–	1-10	–	< 1	1-10
PERFORMANCE IMPACT	Low	Low	High	High	Low	Low	High	Low	Low	Low	Medium	Low

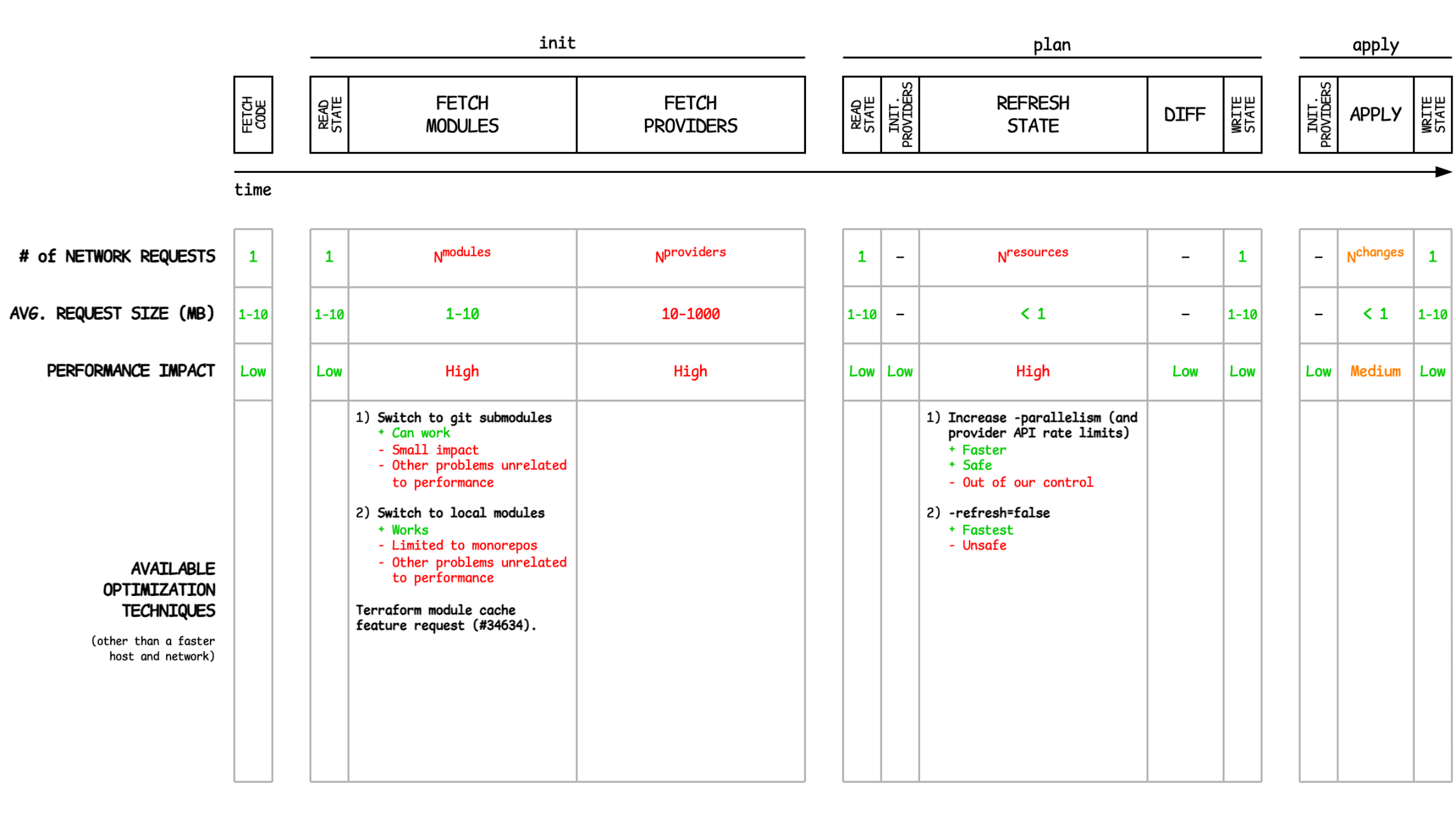

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)

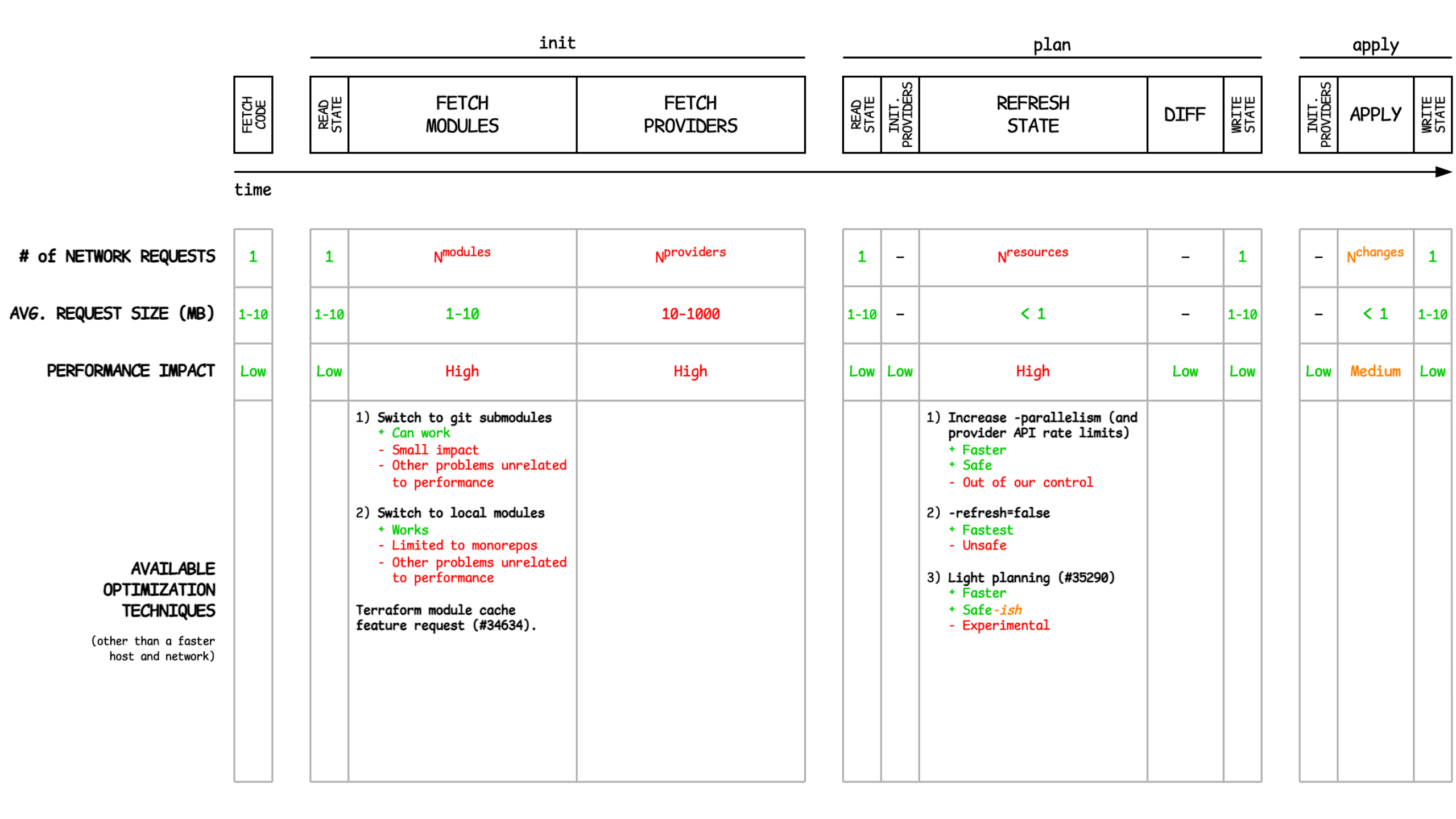

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance

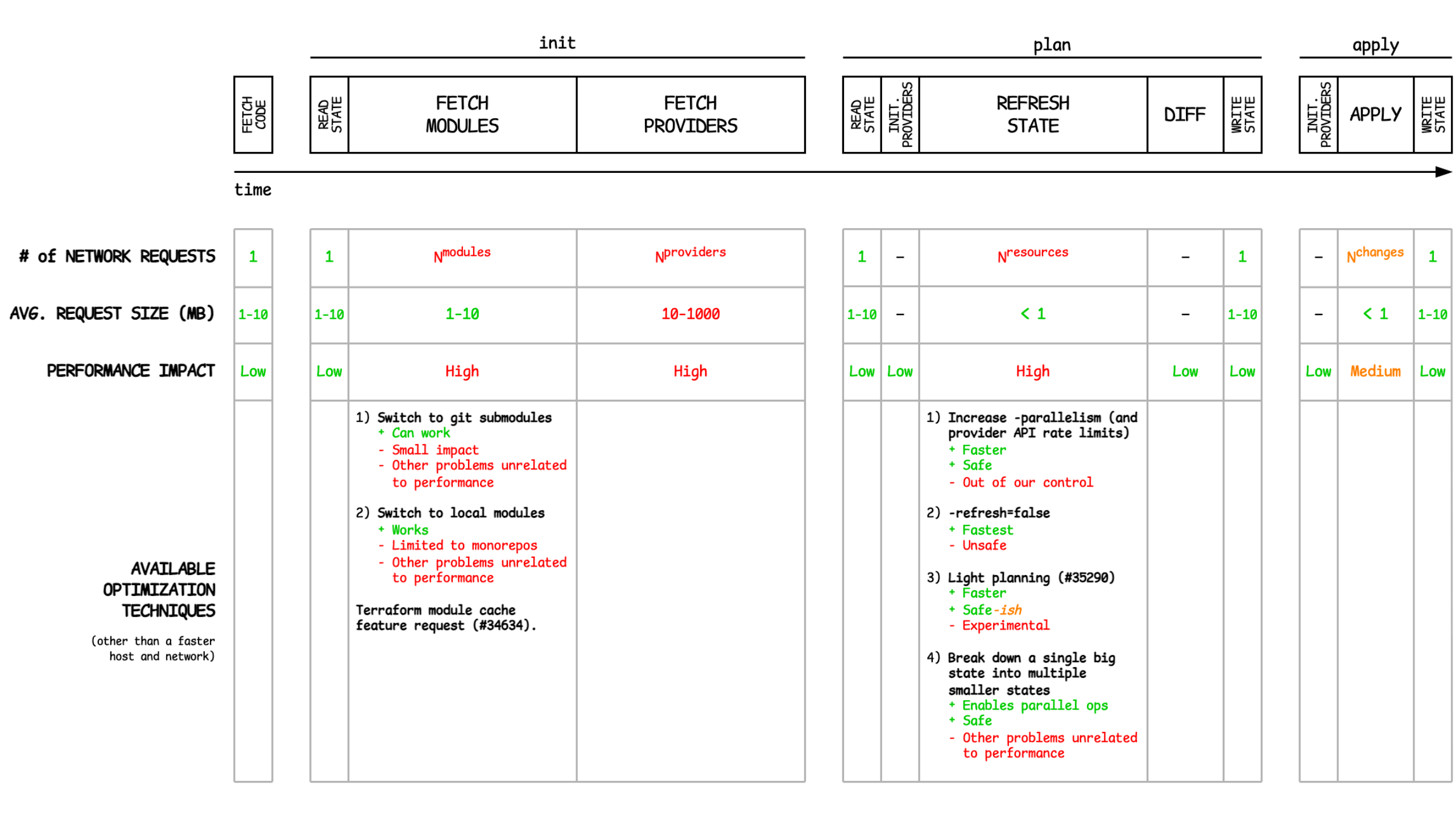

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance

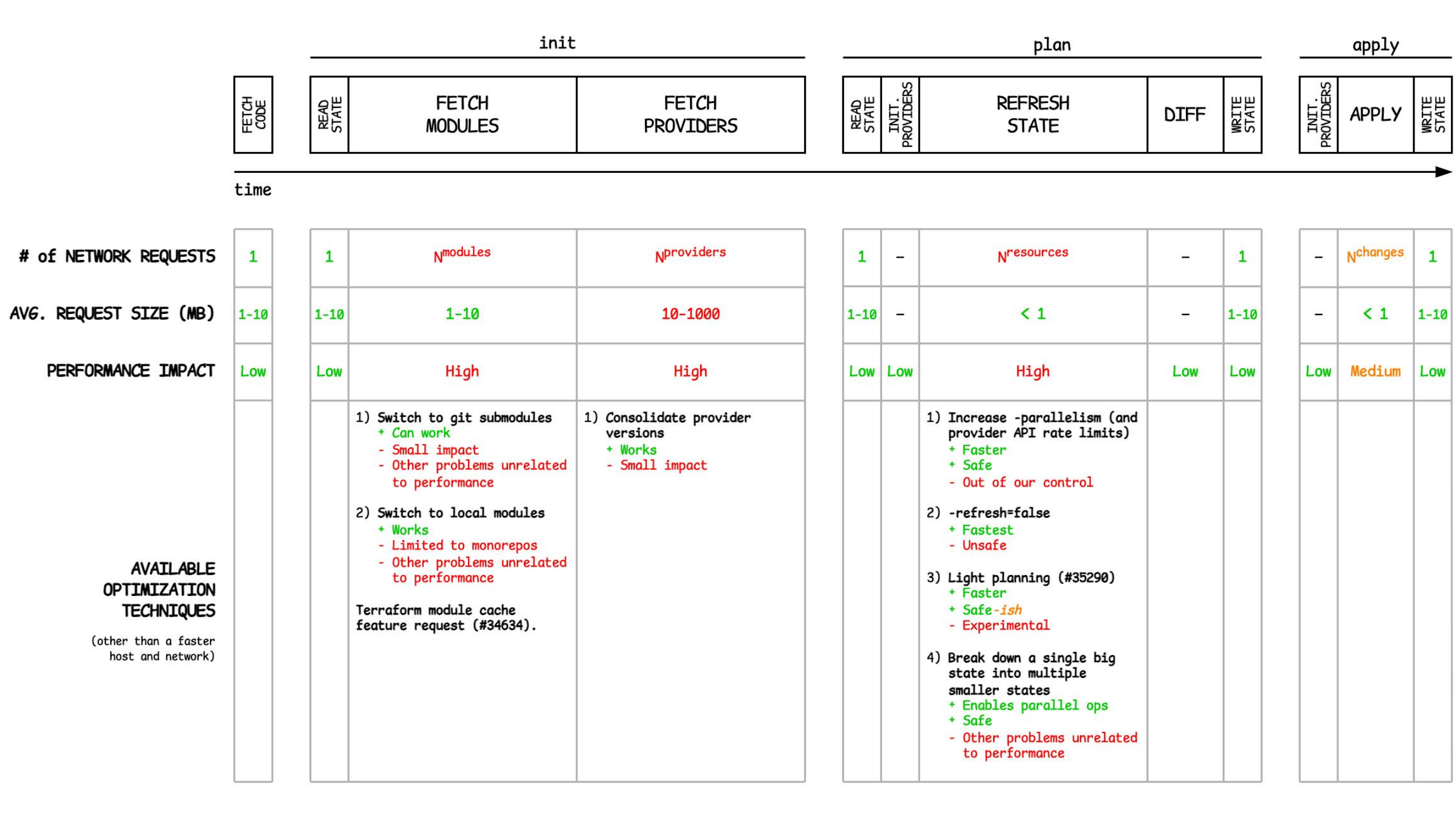

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).

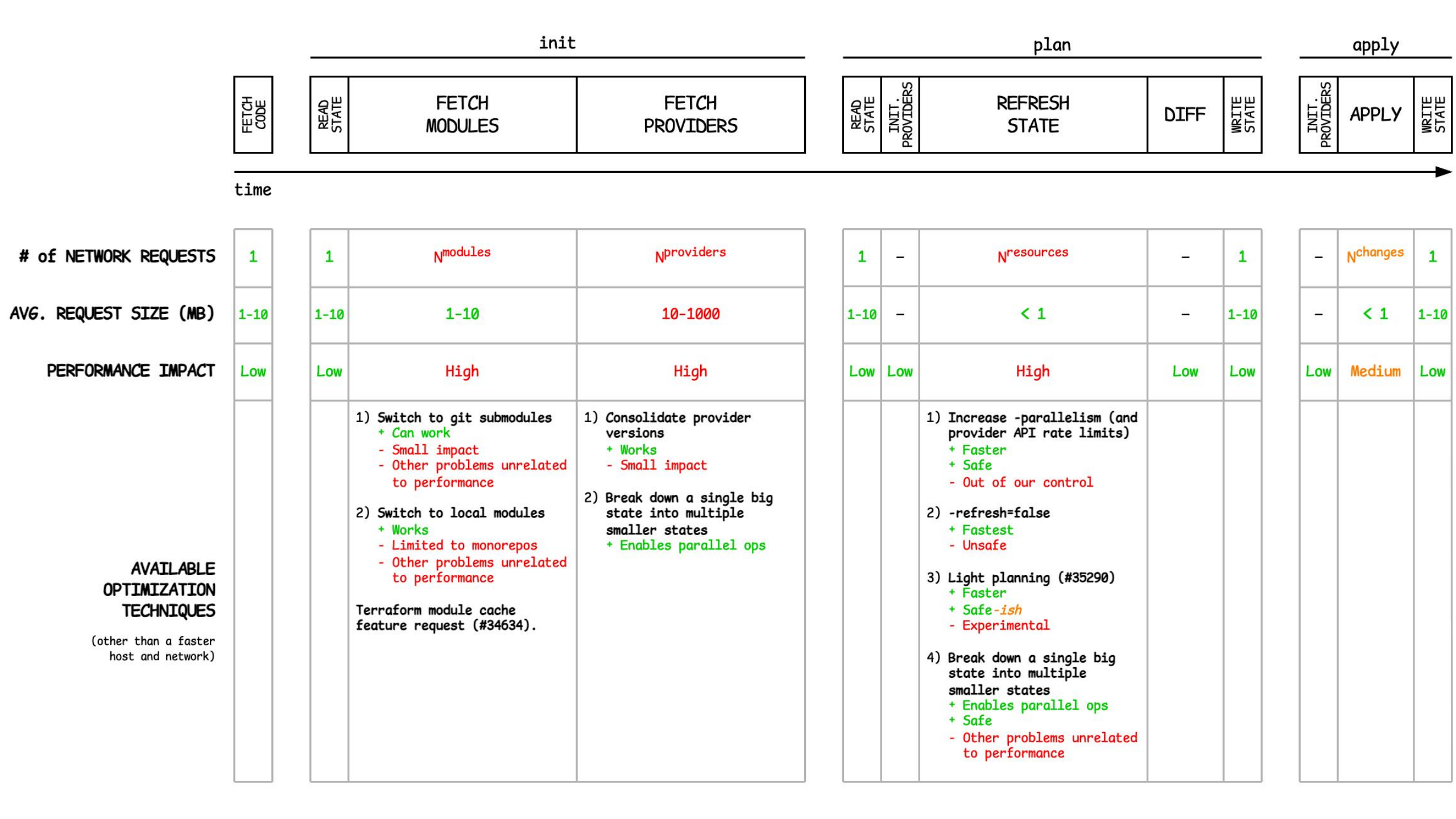

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control

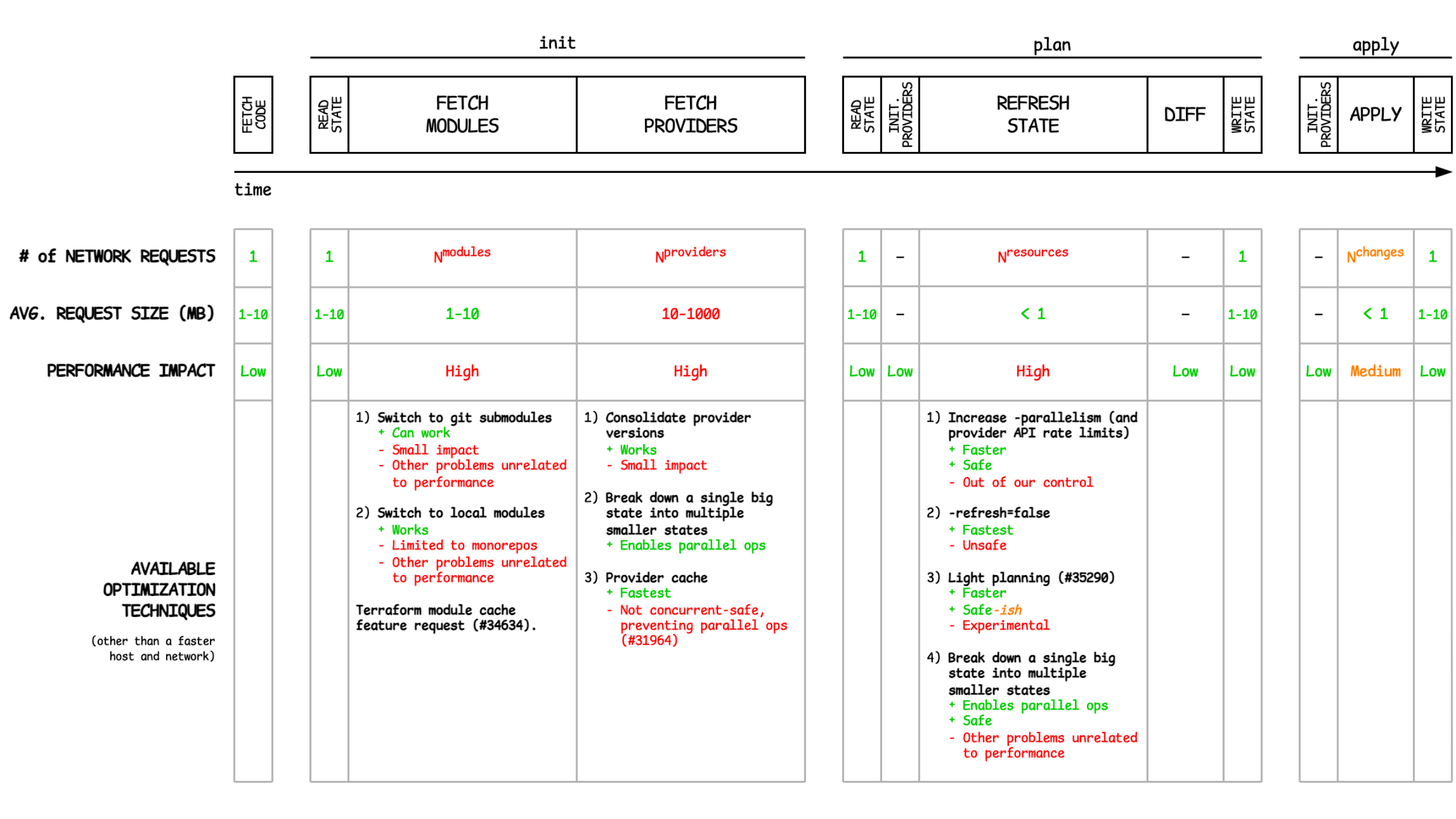

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe

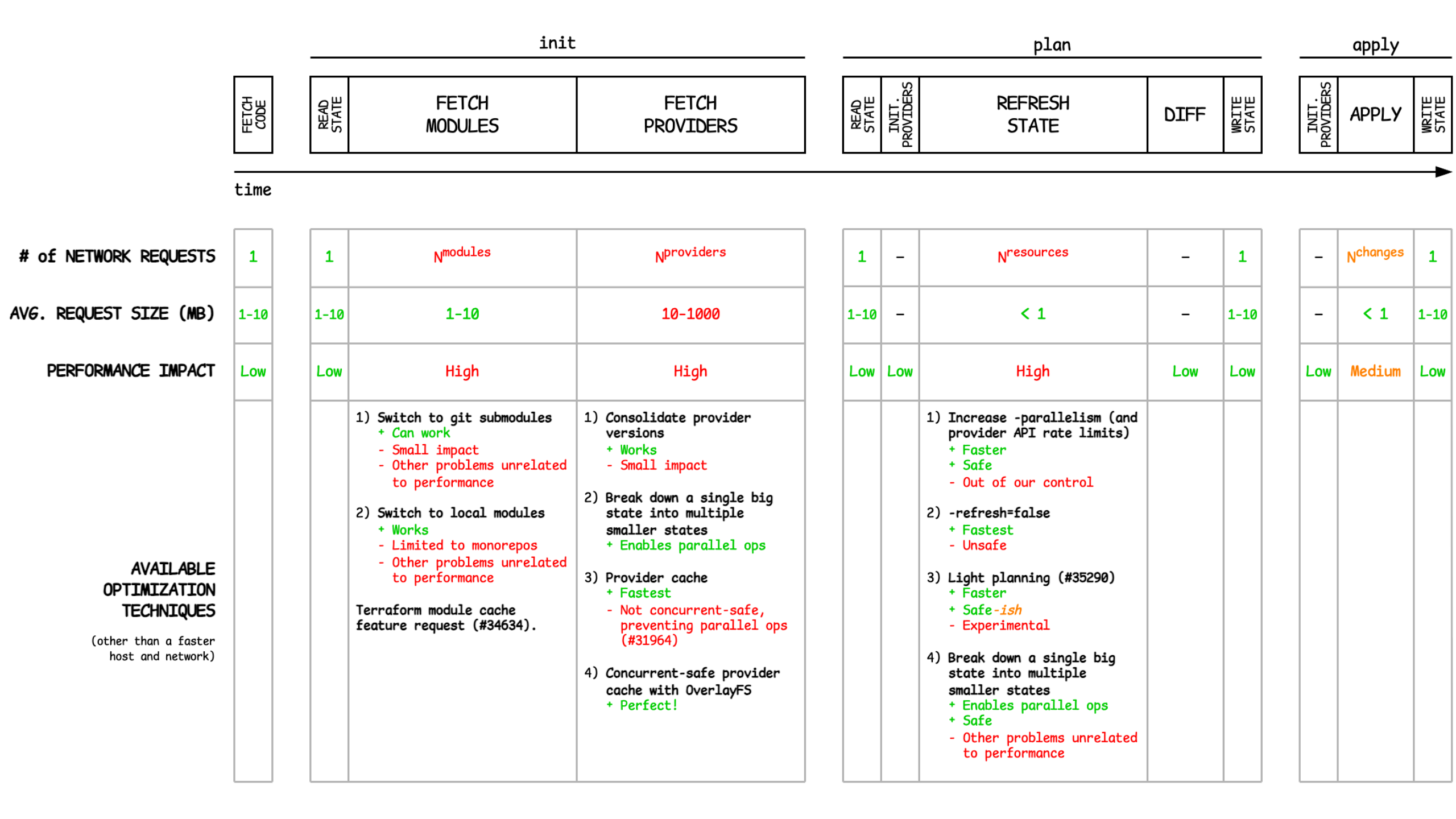

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental

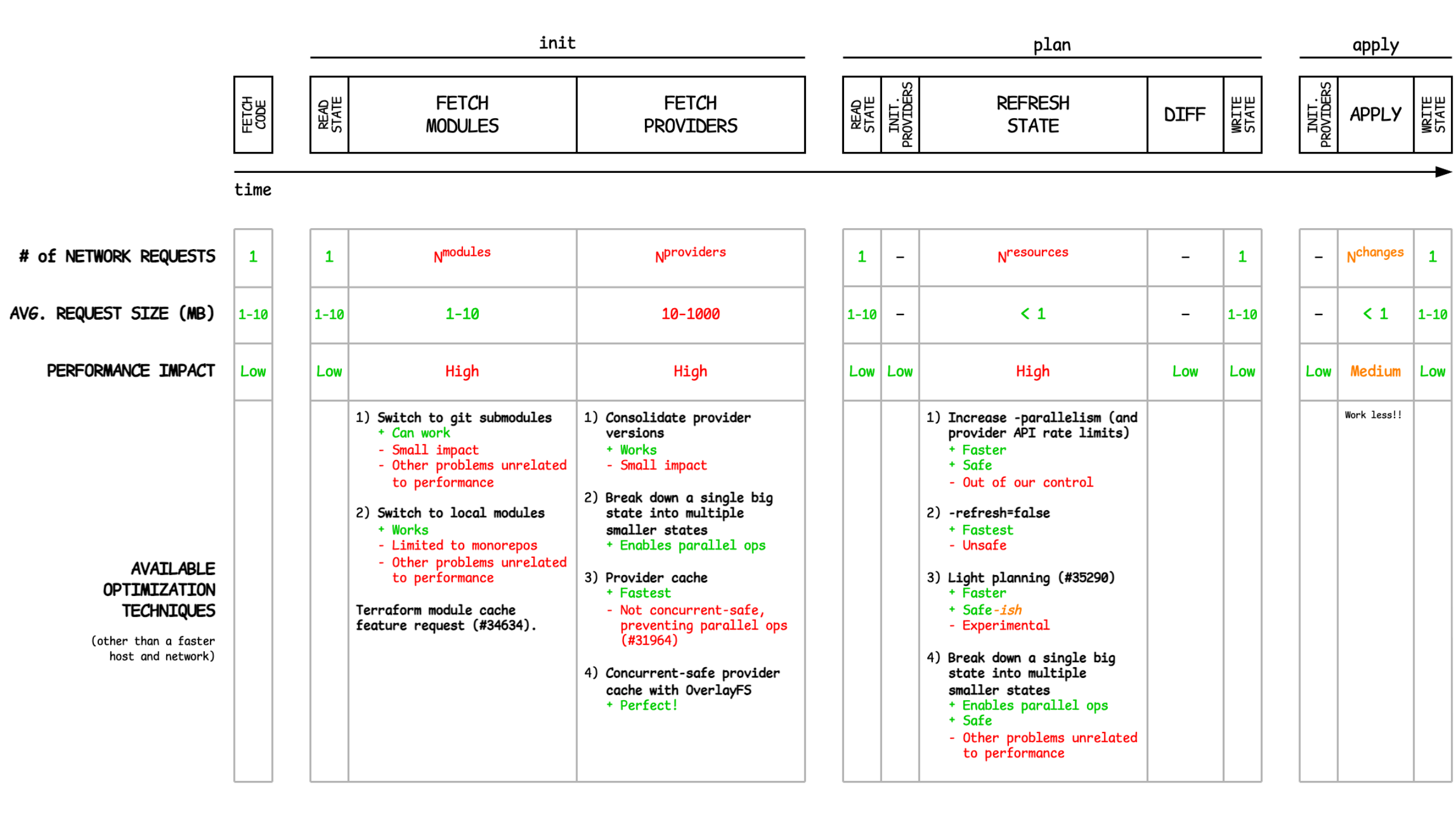

init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Consolidate provider versions
+ Works
- Small impact
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Consolidate provider versions
+ Works
- Small impact
2) Break down a single big state into multiple smaller states
+ Enables parallel ops
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Consolidate provider versions
+ Works
- Small impact
2) Break down a single big state into multiple smaller states
+ Enables parallel ops
3) Provider cache
+ Fastest
- Not concurrent-safe, preventing parallel ops (#31964)
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Consolidate provider versions
+ Works
- Small impact
2) Break down a single big state into multiple smaller states
+ Enables parallel ops
3) Provider cache
+ Fastest
- Not concurrent-safe, preventing parallel ops (#31964)
4) Concurrent-safe provider cache with OverlayFS
+ Perfect!
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance


init
plan
apply
FETCH CODE
READ STATE
FETCH MODULES
FETCH PROVIDERS
READ STATE
INIT. PROVIDERS
REFRESH STATE
DIFF
WRITE STATE
INIT. PROVIDERS
APPLY
WRITE STATE
time
# of NETWORK REQUESTS
1
1
N^modules
N^providers
1
–
N^resources
–
1
–
N^changes
1
AVG. REQUEST SIZE (MB)
1-10
1-10
1-10
10-1000
1-10
–
< 1
–
1-10
–
< 1
1-10
PERFORMANCE IMPACT
Low
Low
High
High
Low
Low
High
Low
Low
Low
Medium
Low
AVAILABLE OPTIMIZATION TECHNIQUES
(other than a faster host and network)
1) Switch to git submodules
+ Can work
- Small impact
- Other problems unrelated to performance
2) Switch to local modules
+ Works
- Limited to monorepos
- Other problems unrelated to performance
Terraform module cache feature request (#34634).
1) Consolidate provider versions
+ Works
- Small impact
2) Break down a single big state into multiple smaller states
+ Enables parallel ops
3) Provider cache
+ Fastest
- Not concurrent-safe, preventing parallel ops (#31964)
4) Concurrent-safe provider cache with OverlayFS
+ Perfect!
1) Increase -parallelism (and provider API rate limits)
+ Faster
+ Safe
- Out of our control
2) -refresh=false
+ Fastest
- Unsafe
3) Light planning (#35290)
+ Faster
+ Safe-ish
- Experimental
4) Break down a single big state into multiple smaller states
+ Enables parallel ops
+ Safe
- Other problems unrelated to performance
Work less!!

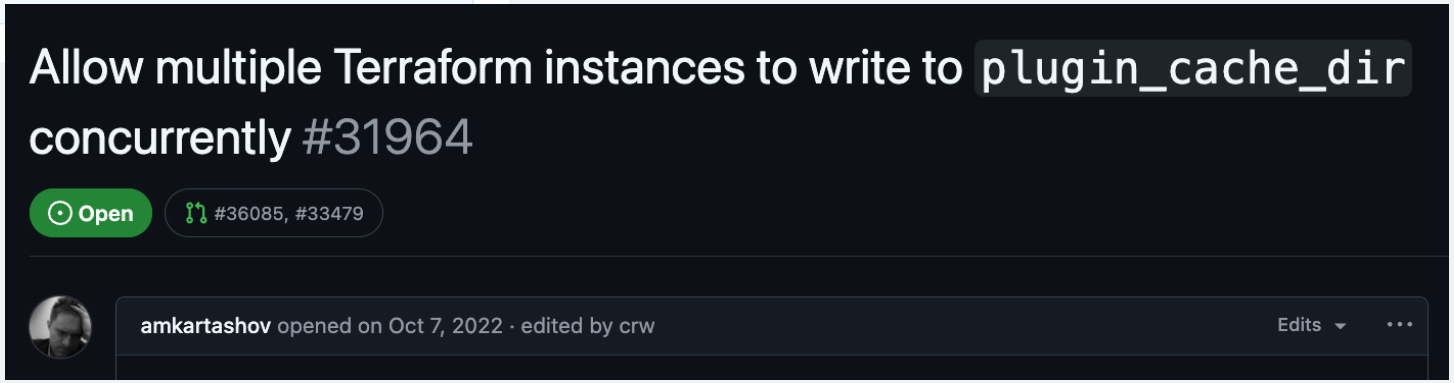
Terraform's provider cache is **not** concurrent-safe

Terraform’s provider cache is **not** concurrent-safe

> **Note:** The plugin cache directory is not guaranteed to be concurrency safe. The provider installer's behavior in environments with multiple `terraform init` calls is undefined.

Terraform's provider cache is **not** concurrent-safe

> **Note:** The plugin cache directory is not guaranteed to be concurrency safe. The provider installer's behavior in environments with multiple `terraform init` calls is undefined.



Terraform's provider cache is **not** concurrent-safe

> **Note:** The plugin cache directory is not guaranteed to be concurrency safe. The provider installer's behavior in environments with multiple `terraform init` calls is undefined.

Allow multiple Terraform instances to write to `plugin_cache_dir` concurrently #31964

Open · #36085, #33479

amkartashov opened on Oct 7, 2022 · edited by crw · Edits

apparentlymart on Oct 10, 2022 · Contributor

If we do want to change something to make this concurrency-safe then I think a key requirement for us to navigate is ensuring we don't break anyone who currently has their cache directory on a network filesystem where e.g. filesystem-level locking may not be available or may not be reliable.

We didn't explicitly document that the plugin cache directory must be on a filesystem that is only visible to the current kernel, and I've seen questions online in the past which imply that people are already doing that and so I think that existing behavior is de-facto covered by the v1.x Compatibility Promises because when our documentation was ambiguous about something we typically favor keeping existing usage patterns working unless there's a strong reason to break them.

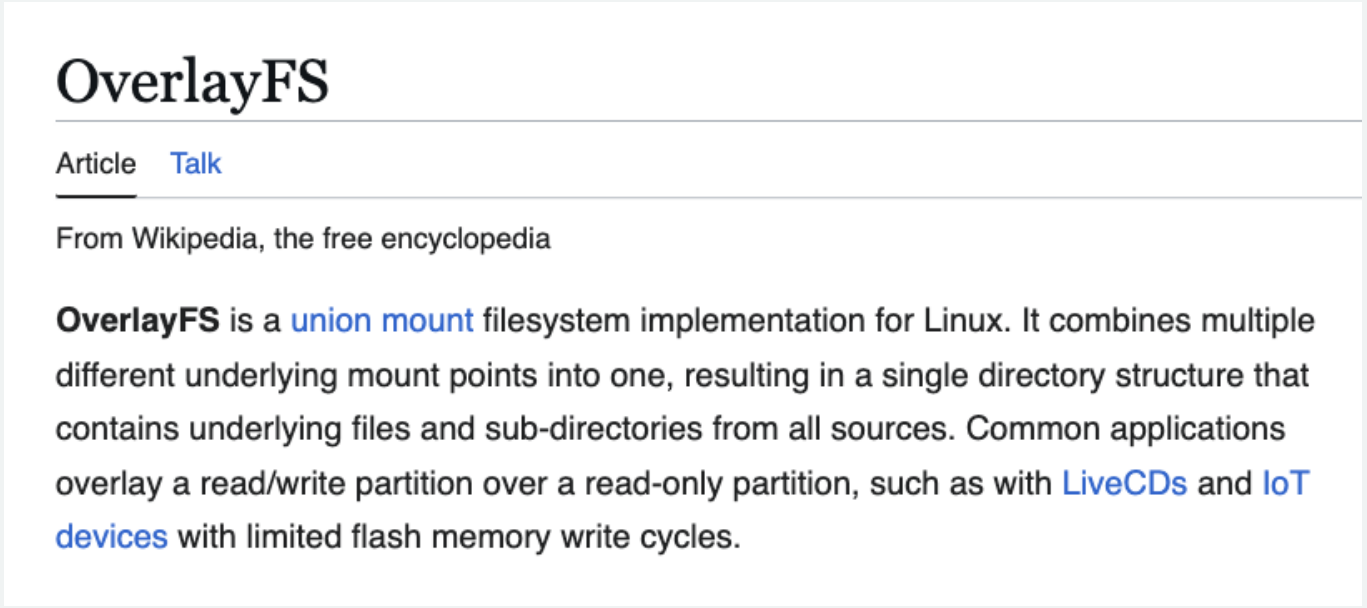
(I don't intend this to mean that we *absolutely cannot* add an additional restriction here, but if we wish to do that then I think we'll need to justify that the benefit outweighs the cost, and probably also provide a reasonable migration path or backward-compatibility mechanism for those who are relying on our current lack of global locking for the cache directory.)

Hopefully we can instead devise a solution which relies on the atomicity of certain filesystem operations rather than on explicit locking.

For example: perhaps Terraform could initially populate a directory named so that other Terraform processes won't find it, and then move it into its final location using an atomic move/rename system call. If the move/rename fails due to a conflicting

Introducing OverlayFS

wikipedia.org/wiki/OverlayFS



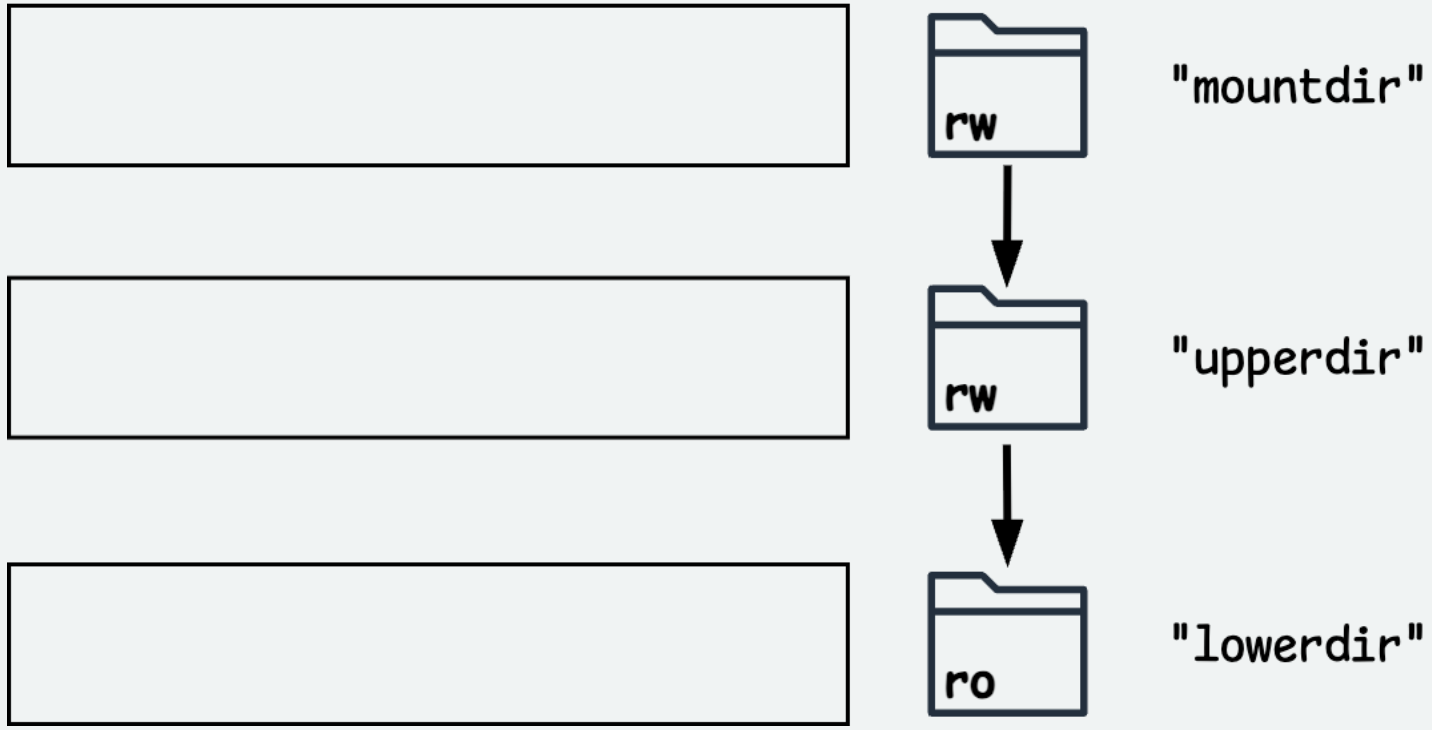
OverlayFS

Article Talk

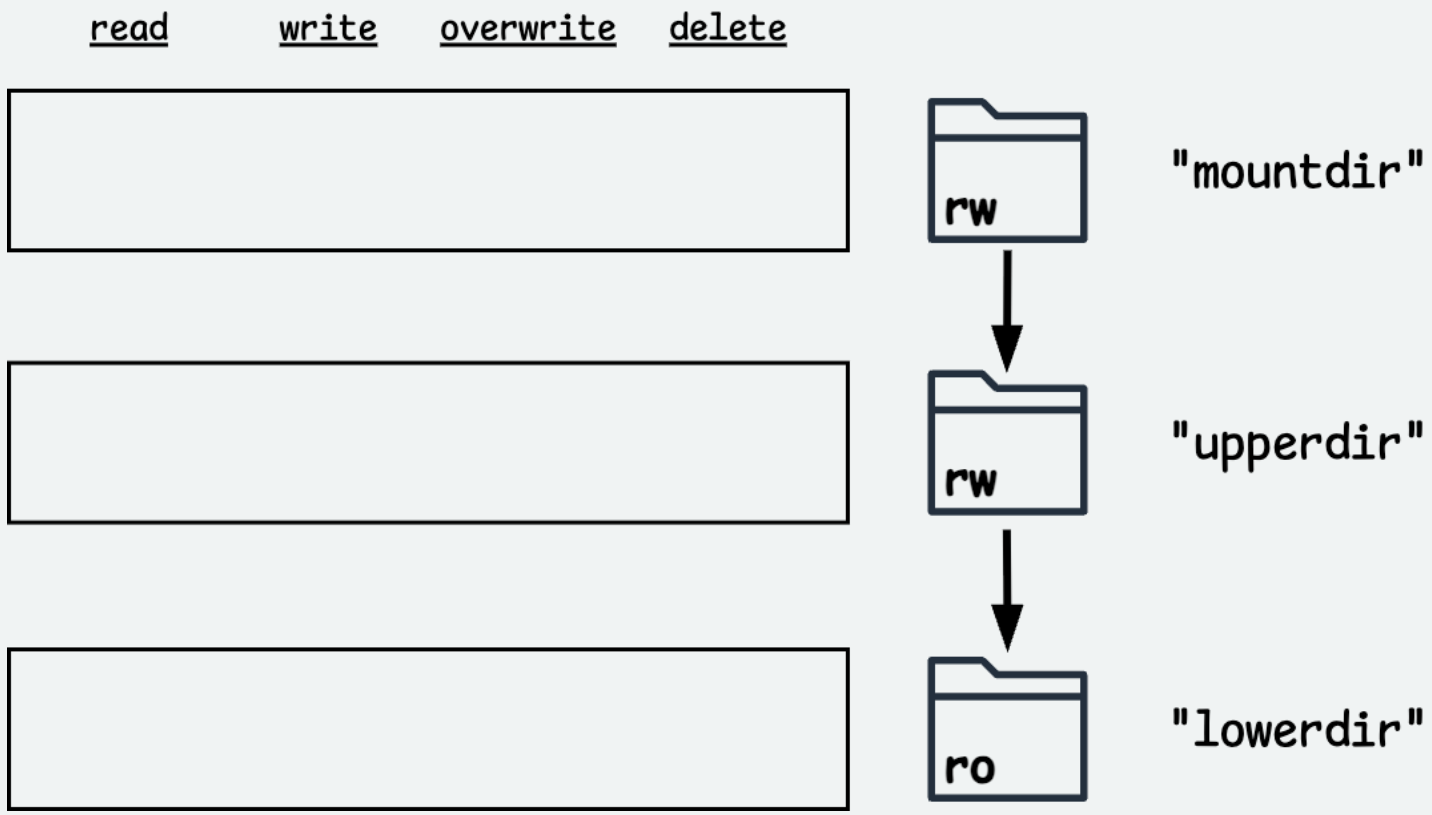
From Wikipedia, the free encyclopedia

OverlayFS is a union mount filesystem implementation for Linux. It combines multiple different underlying mount points into one, resulting in a single directory structure that contains underlying files and sub-directories from all sources. Common applications overlay a read/write partition over a read-only partition, such as with LiveCDs and IoT devices with limited flash memory write cycles.

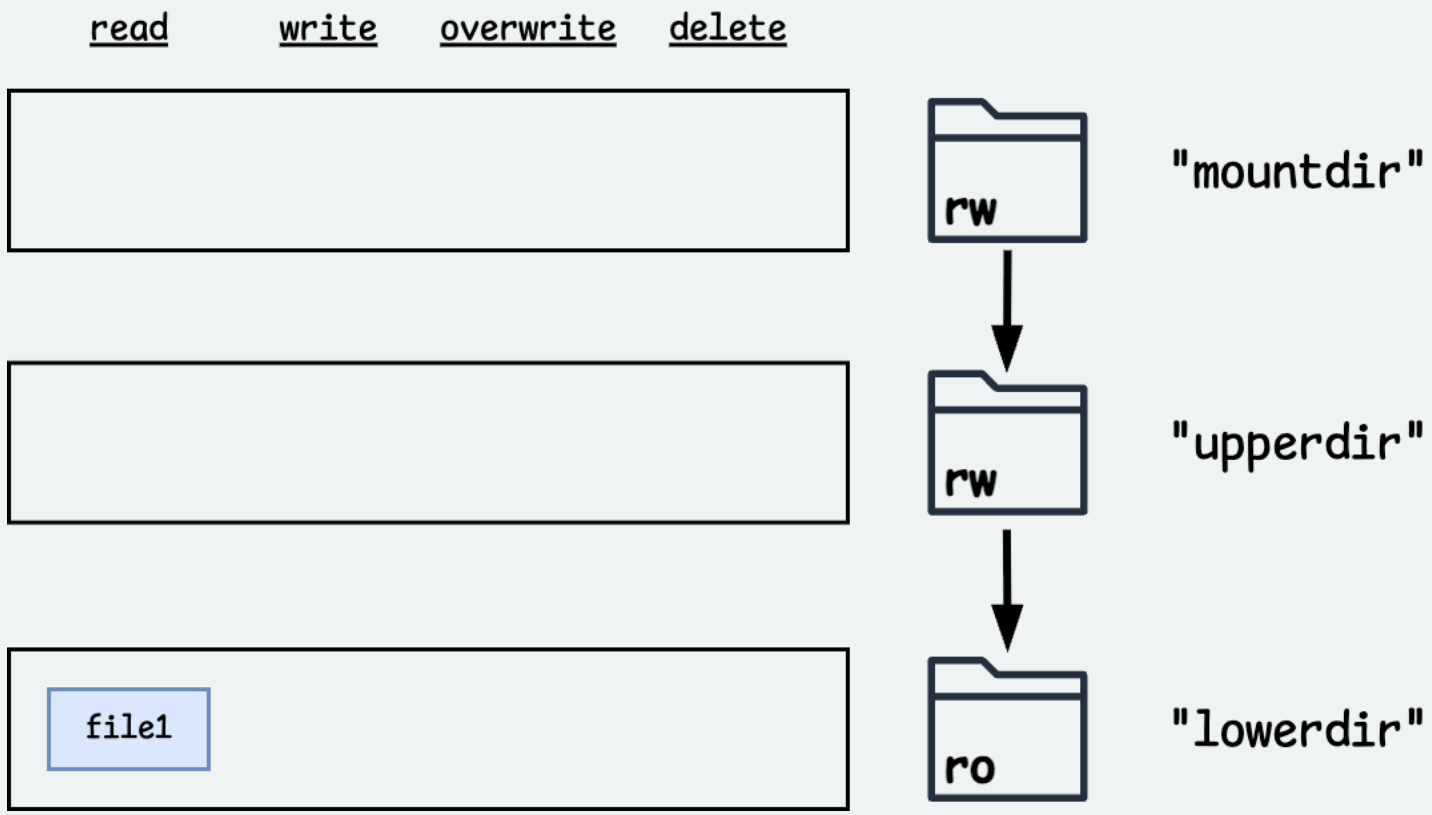
Introducing OverlayFS



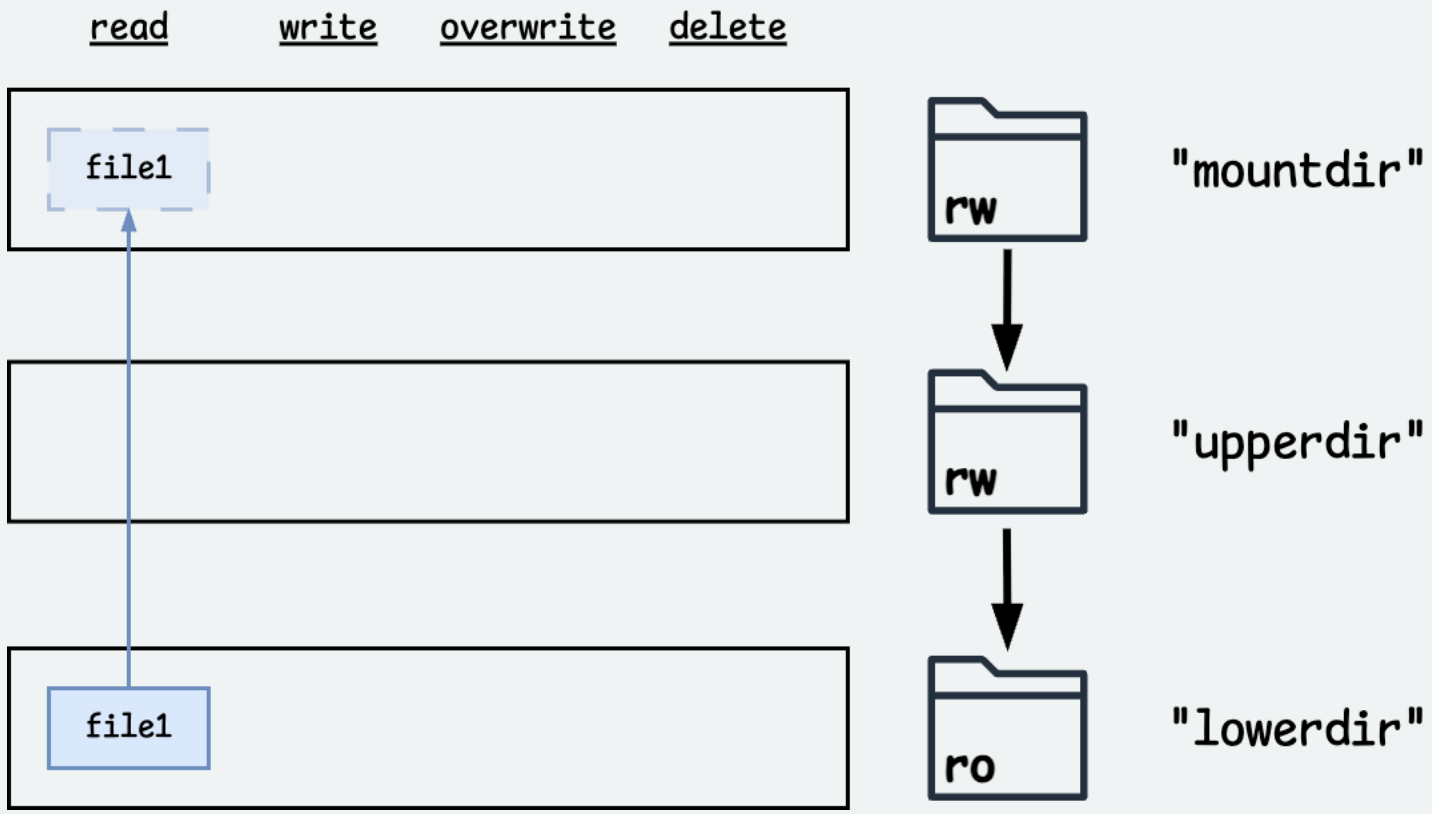
Introducing OverlayFS



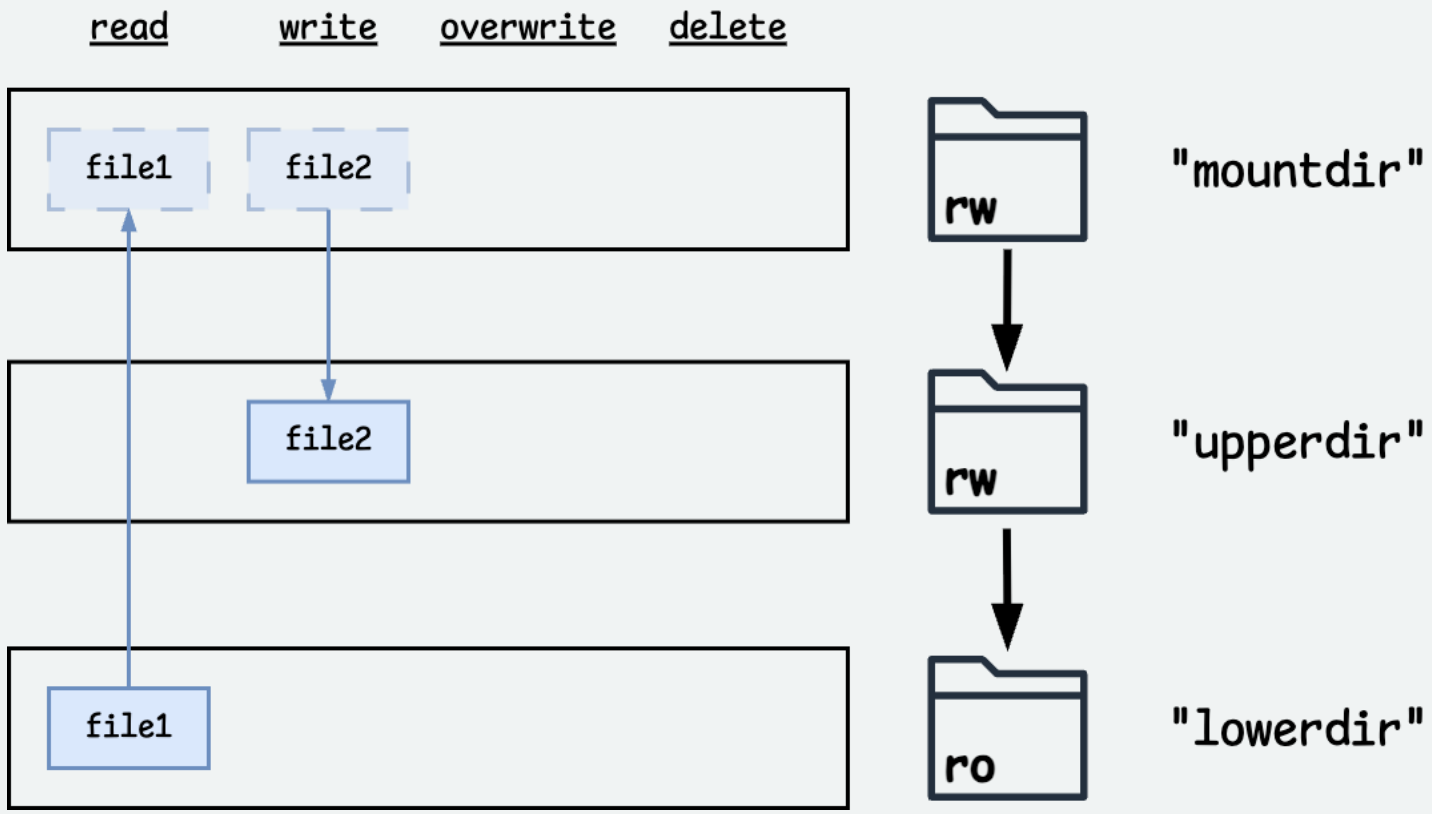
Introducing OverlayFS



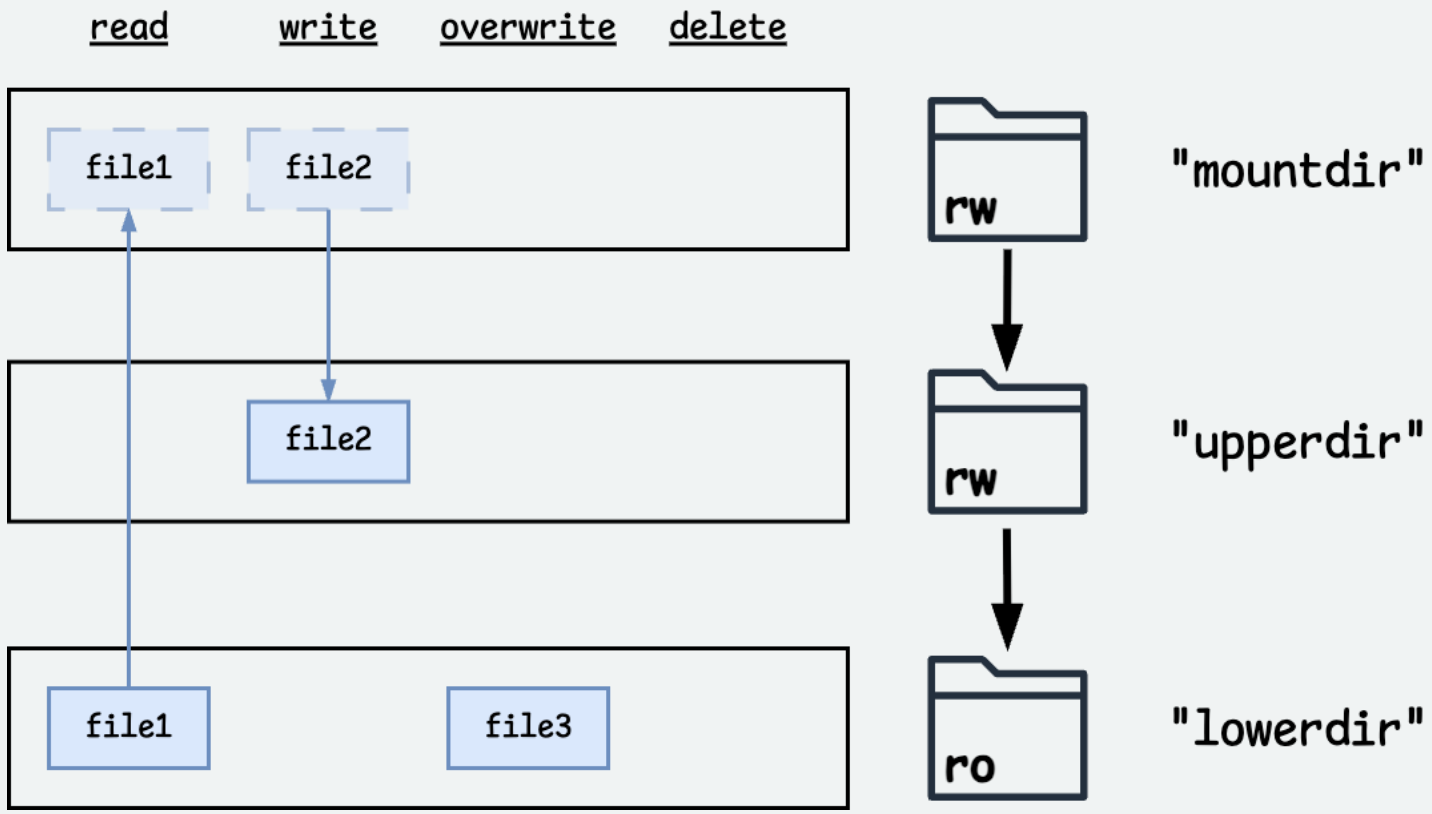
Introducing OverlayFS



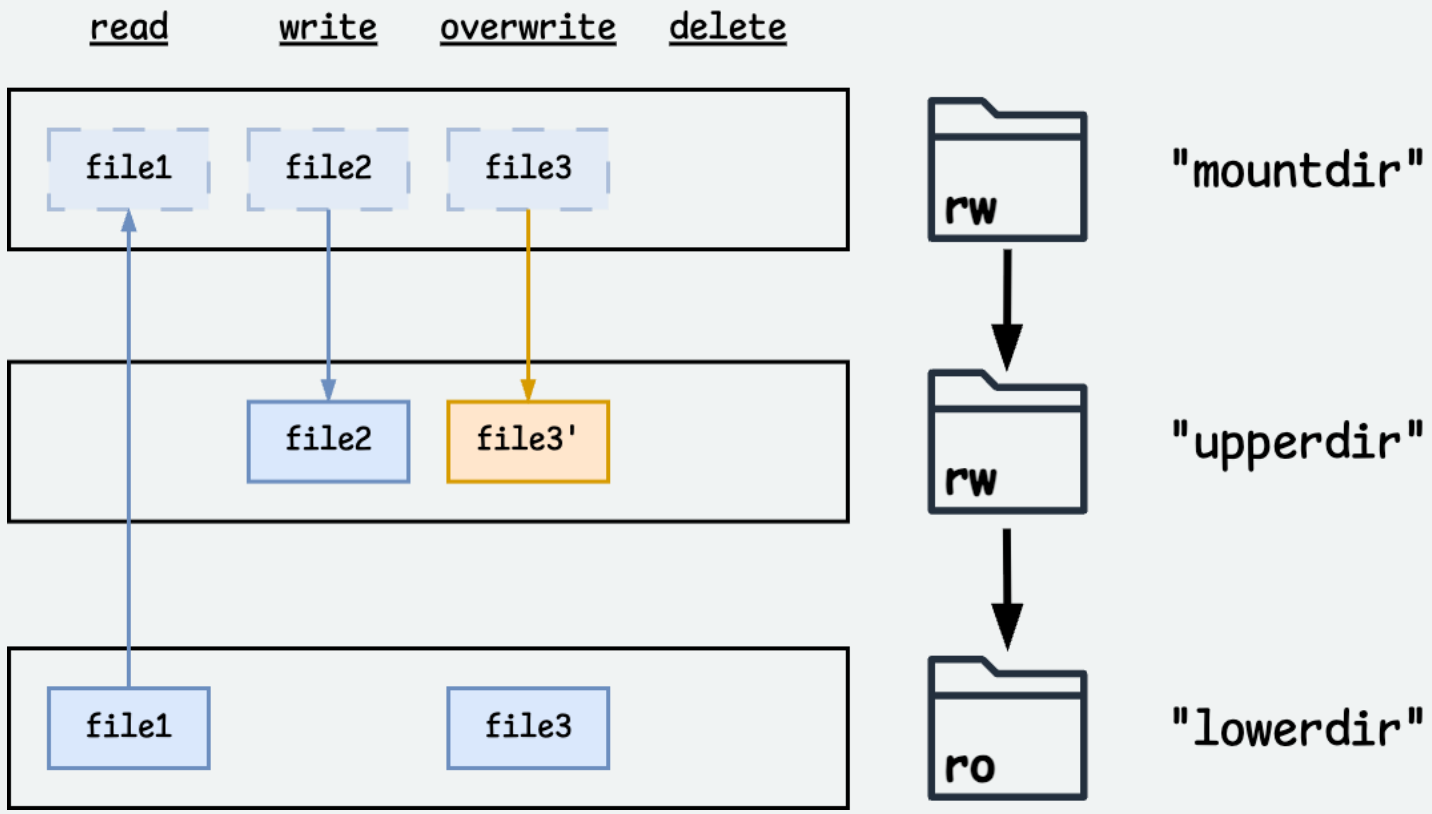
Introducing OverlayFS



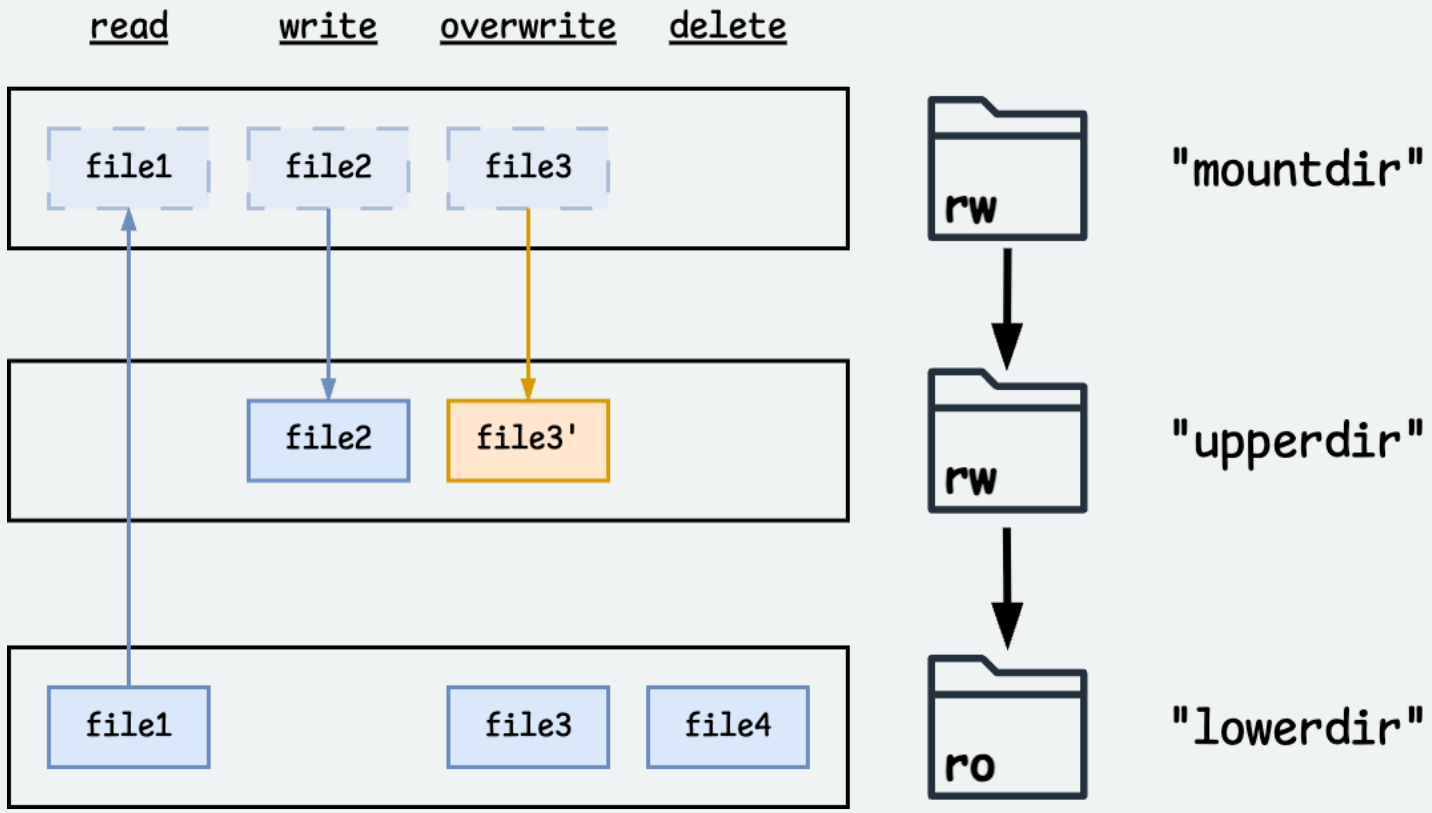
Introducing OverlayFS



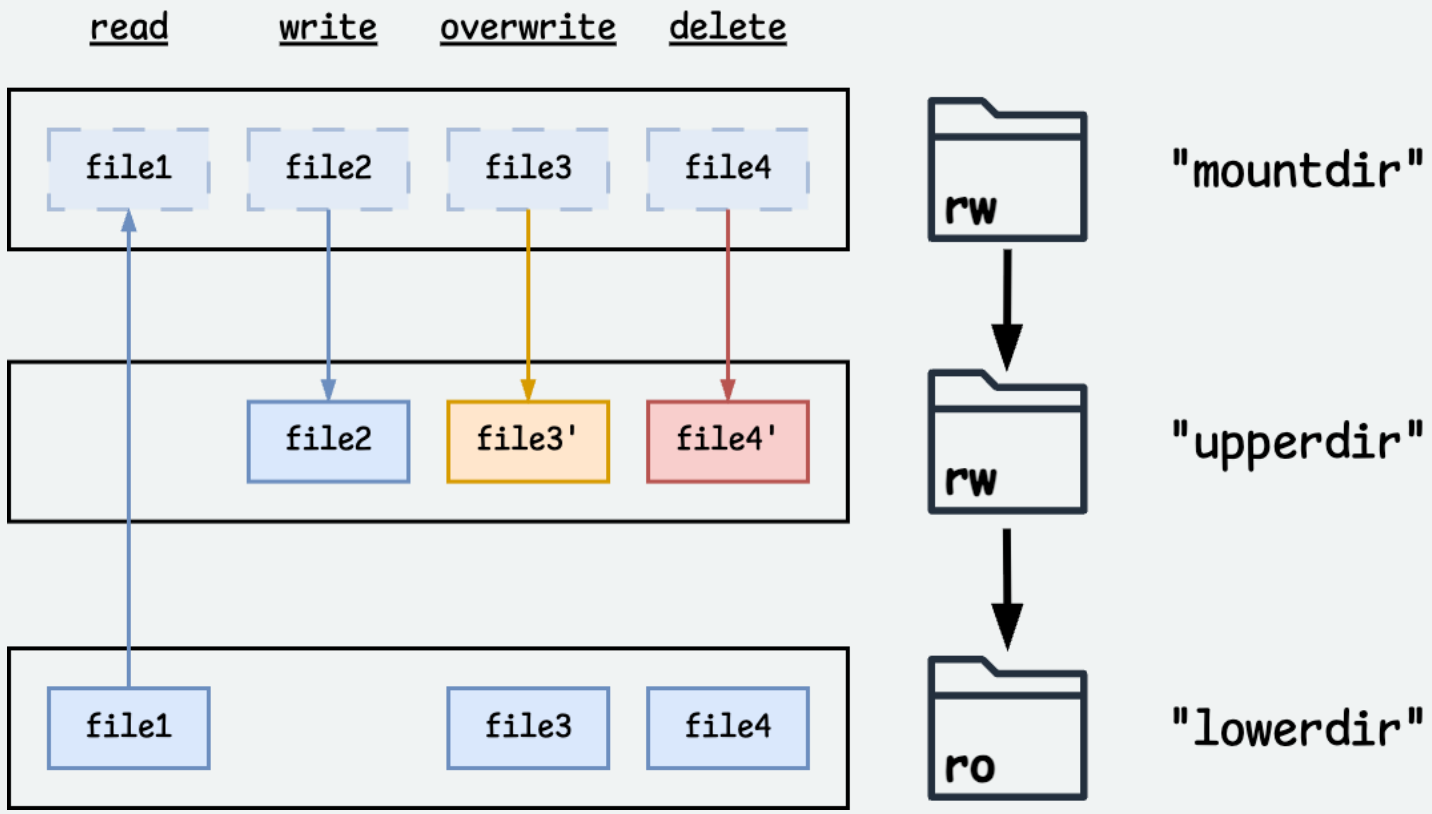
Introducing OverlayFS



Introducing OverlayFS



Introducing OverlayFS

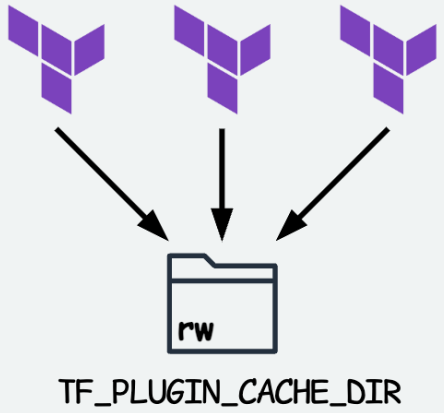


Terraform provider cache with OverlayFS

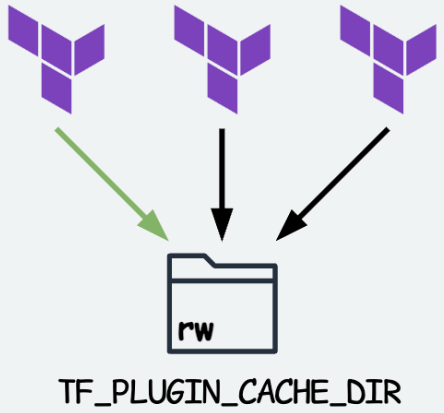
Terraform provider cache with OverlayFS

Terraform provider cache with OverlayFS

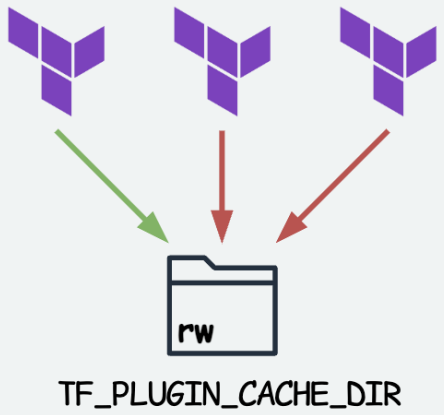
Terraform provider cache with OverlayFS



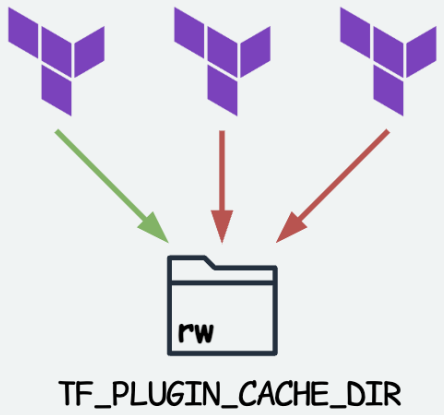
Terraform provider cache with OverlayFS



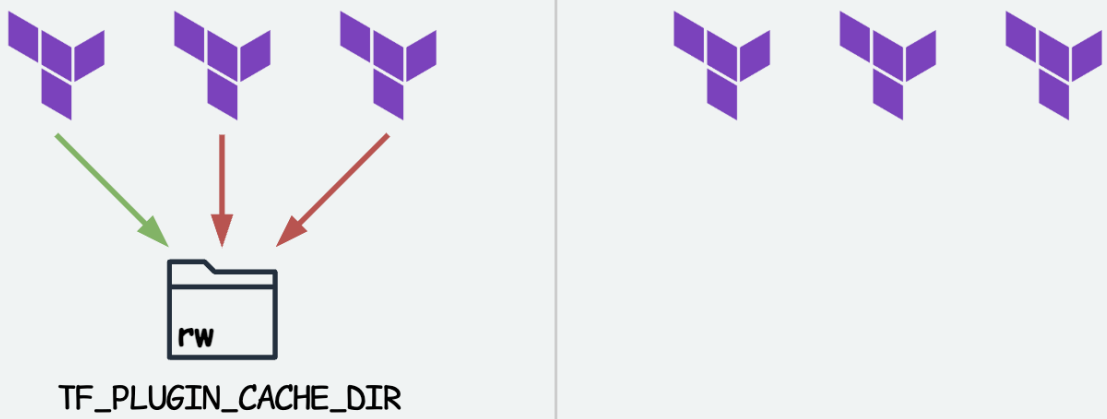
Terraform provider cache with OverlayFS



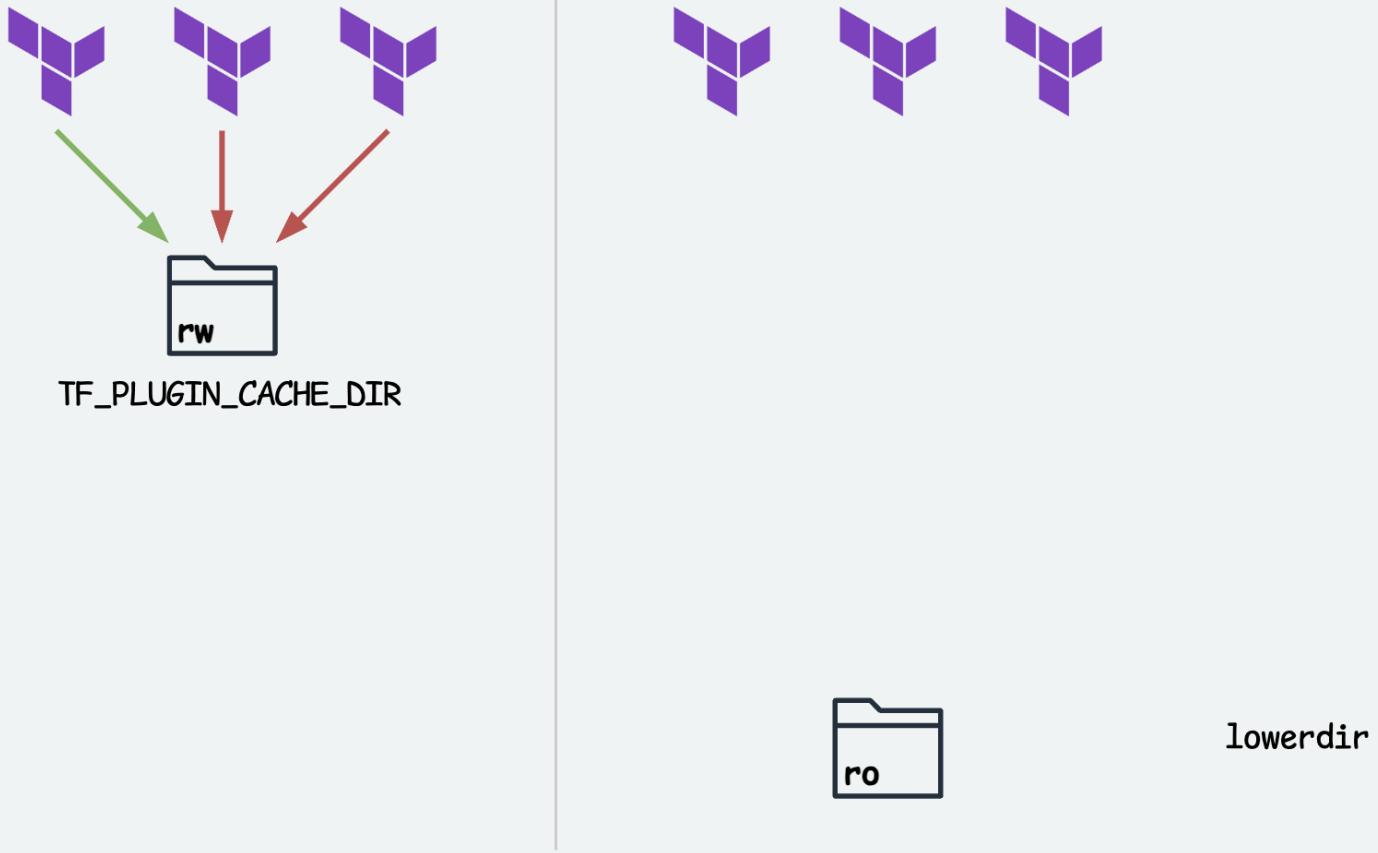
Terraform provider cache with OverlayFS



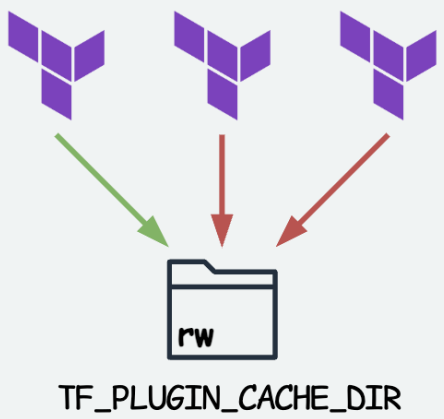
Terraform provider cache with OverlayFS

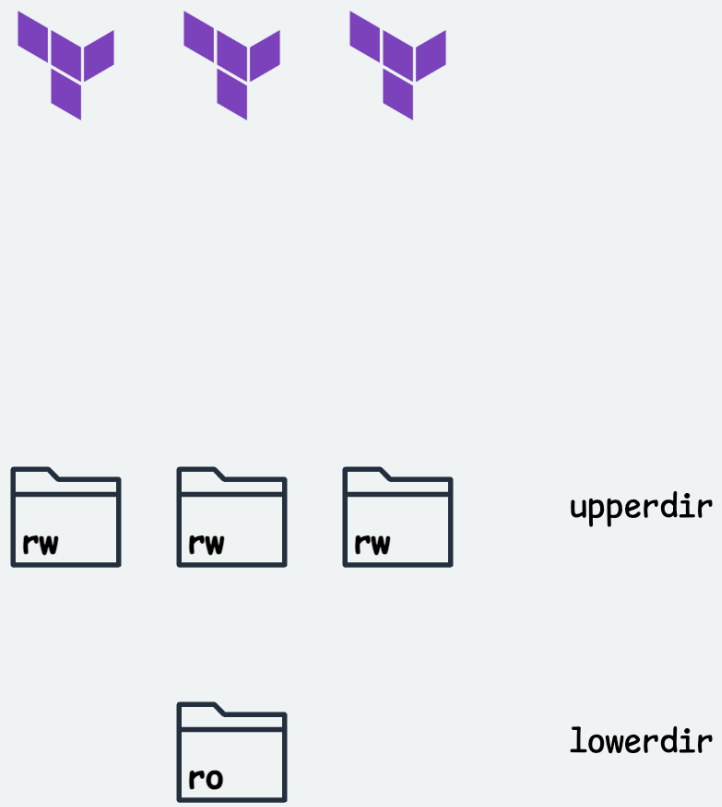


Terraform provider cache with OverlayFS

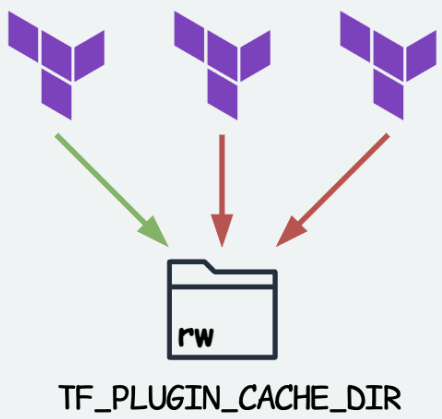


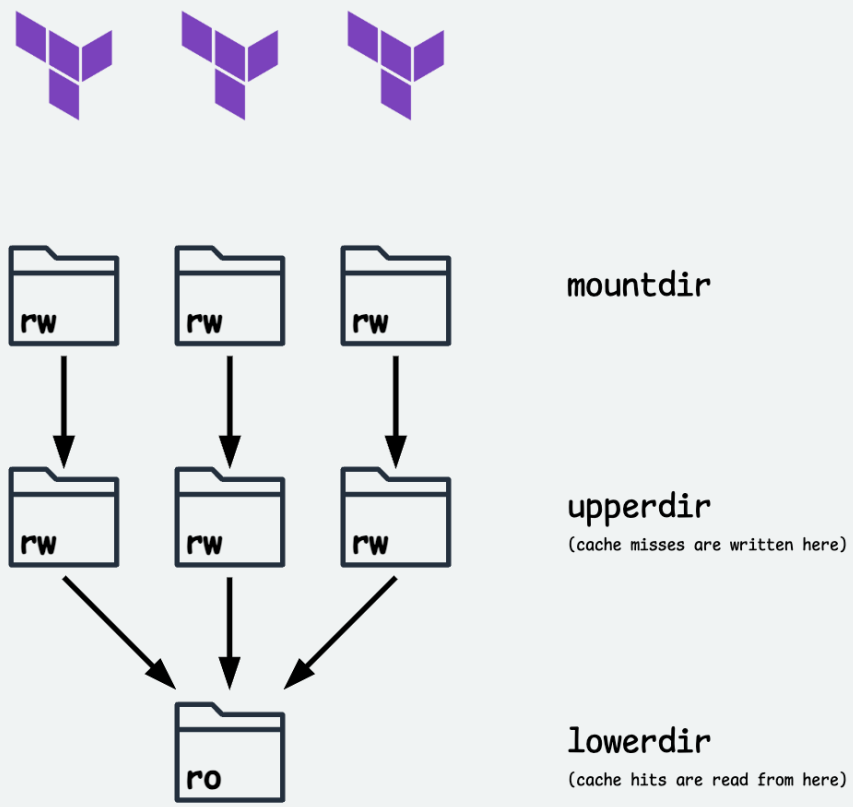
Terraform provider cache with OverlayFS



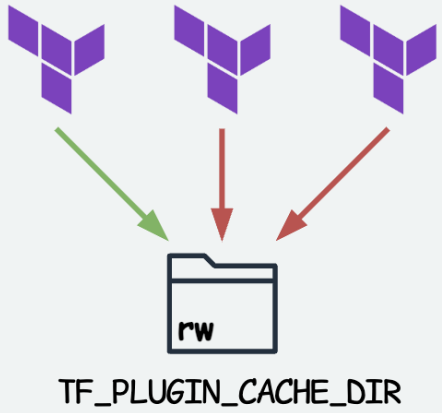


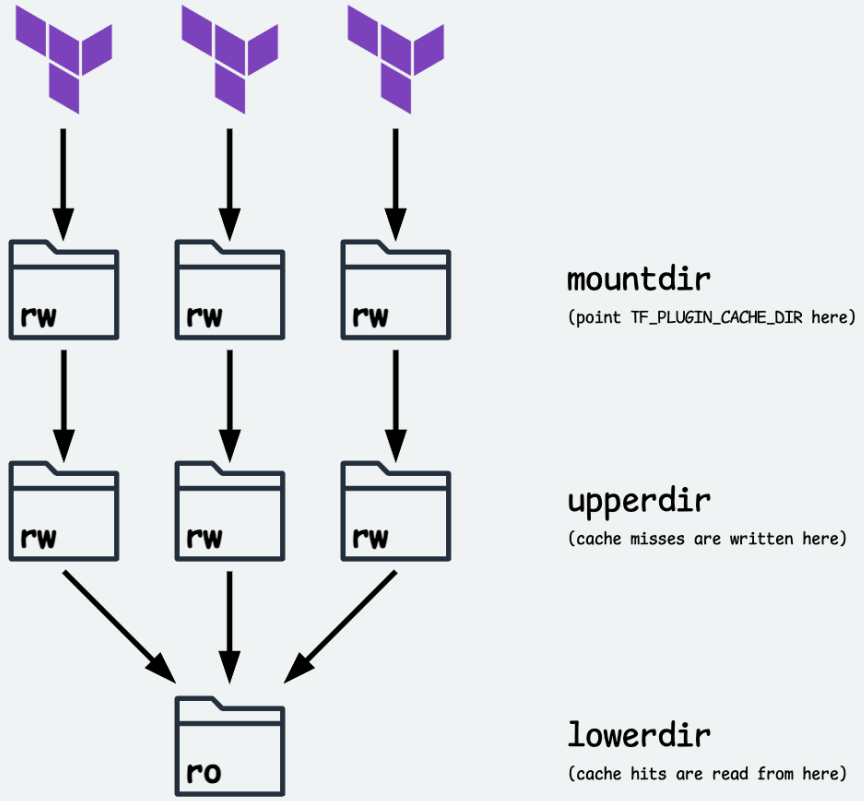
Terraform provider cache with OverlayFS



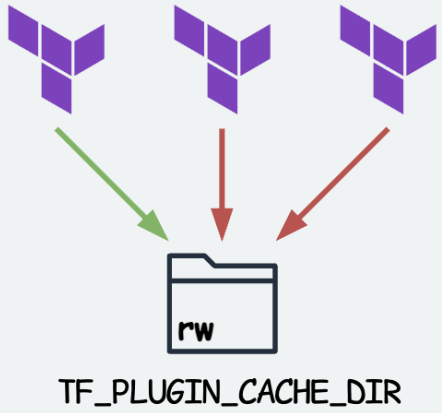


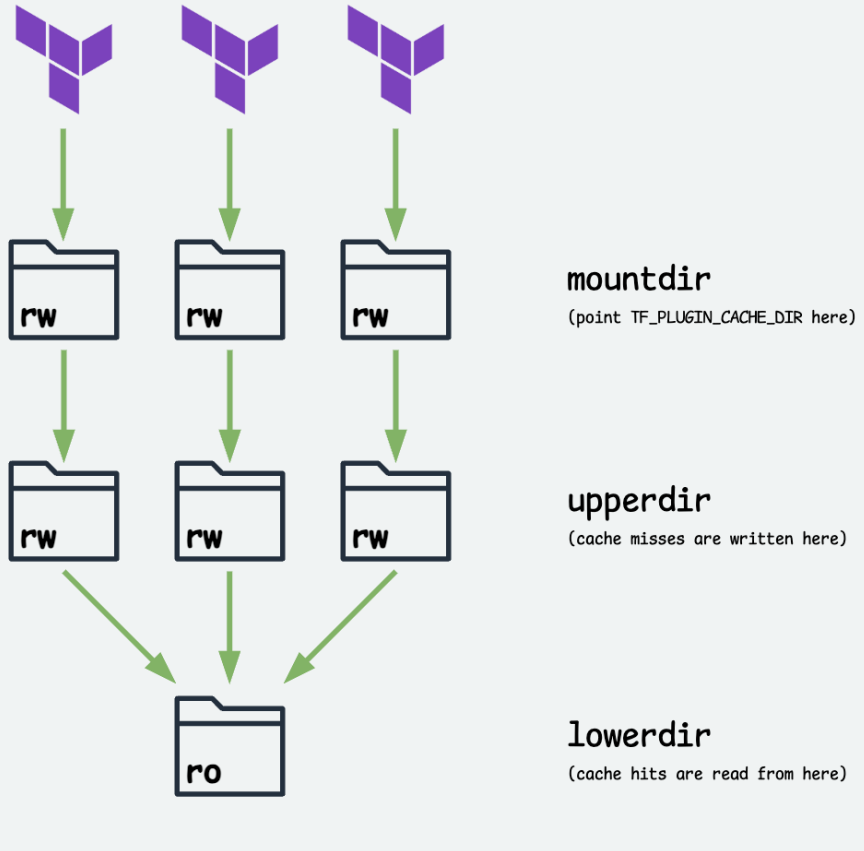
Terraform provider cache with OverlayFS



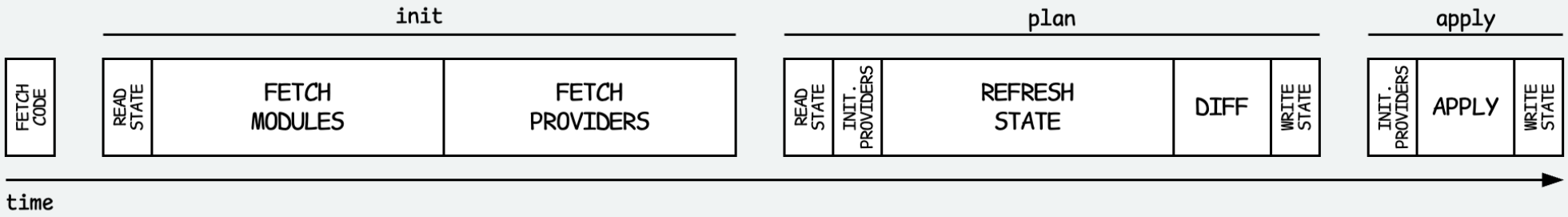


Terraform provider cache with OverlayFS

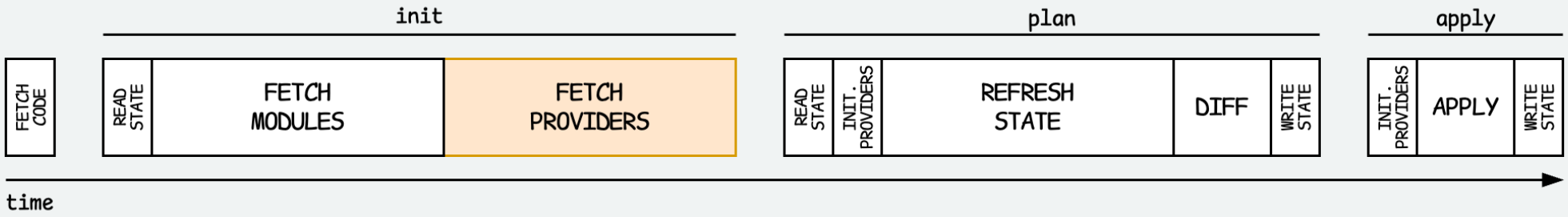




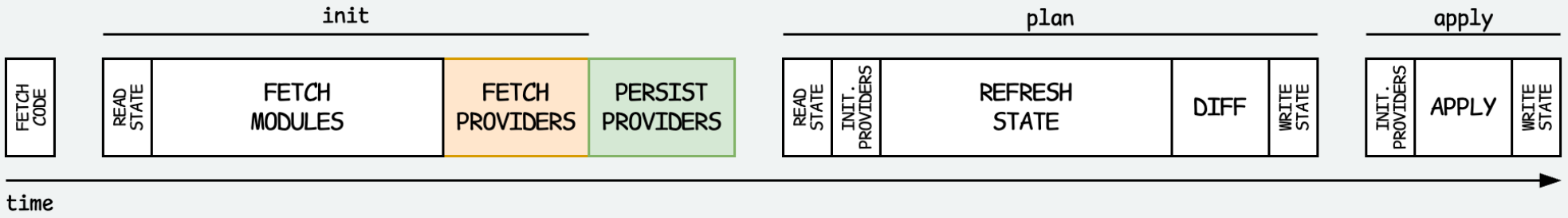
Terraform provider cache with OverlayFS



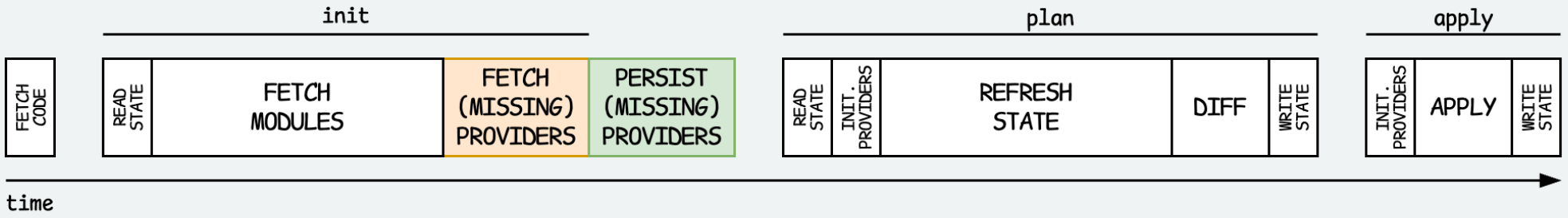
Terraform provider cache with OverlayFS



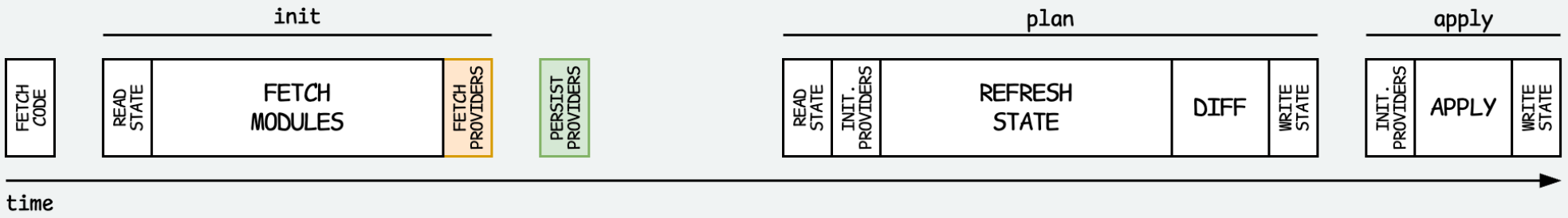
Terraform provider cache with OverlayFS



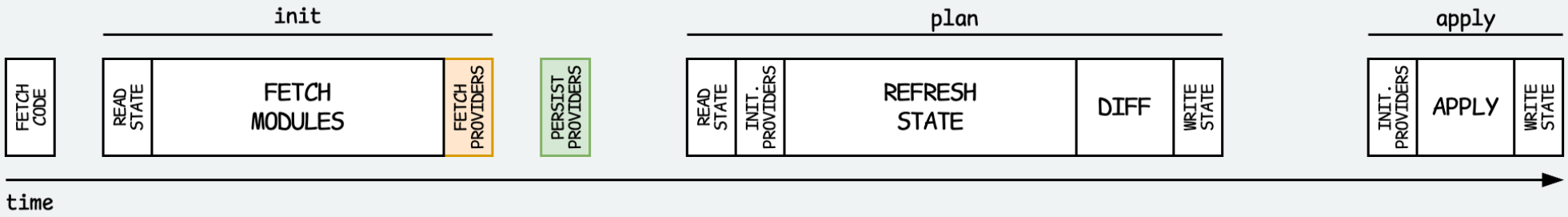
Terraform provider cache with OverlayFS



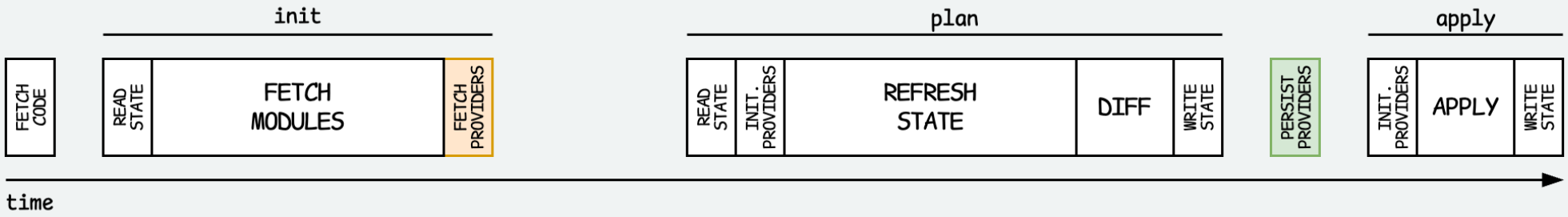
Terraform provider cache with OverlayFS



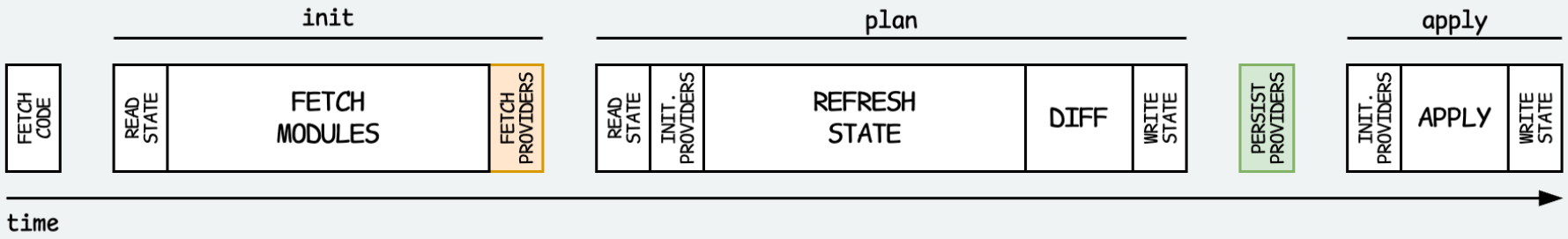
Terraform provider cache with OverlayFS



Terraform provider cache with OverlayFS



Terraform provider cache with OverlayFS



Results

- More than **90% average reduction** in plan times:
  - For our bigger states, we went from measuring in ***hours*** to ***minutes***.
  - For our smaller states, we went from measuring in ***minutes*** to ***seconds***!
- Restored the ability to **plan our entire infrastructure in less than an hour!**
  - Unlocking use cases such as **drift detection** and **cross-cutting infrastructure operations**.
- My squad **no longer gets pinged** for slow Terraform pipeline performance.

How can you get this?

If you use HCP Terraform:

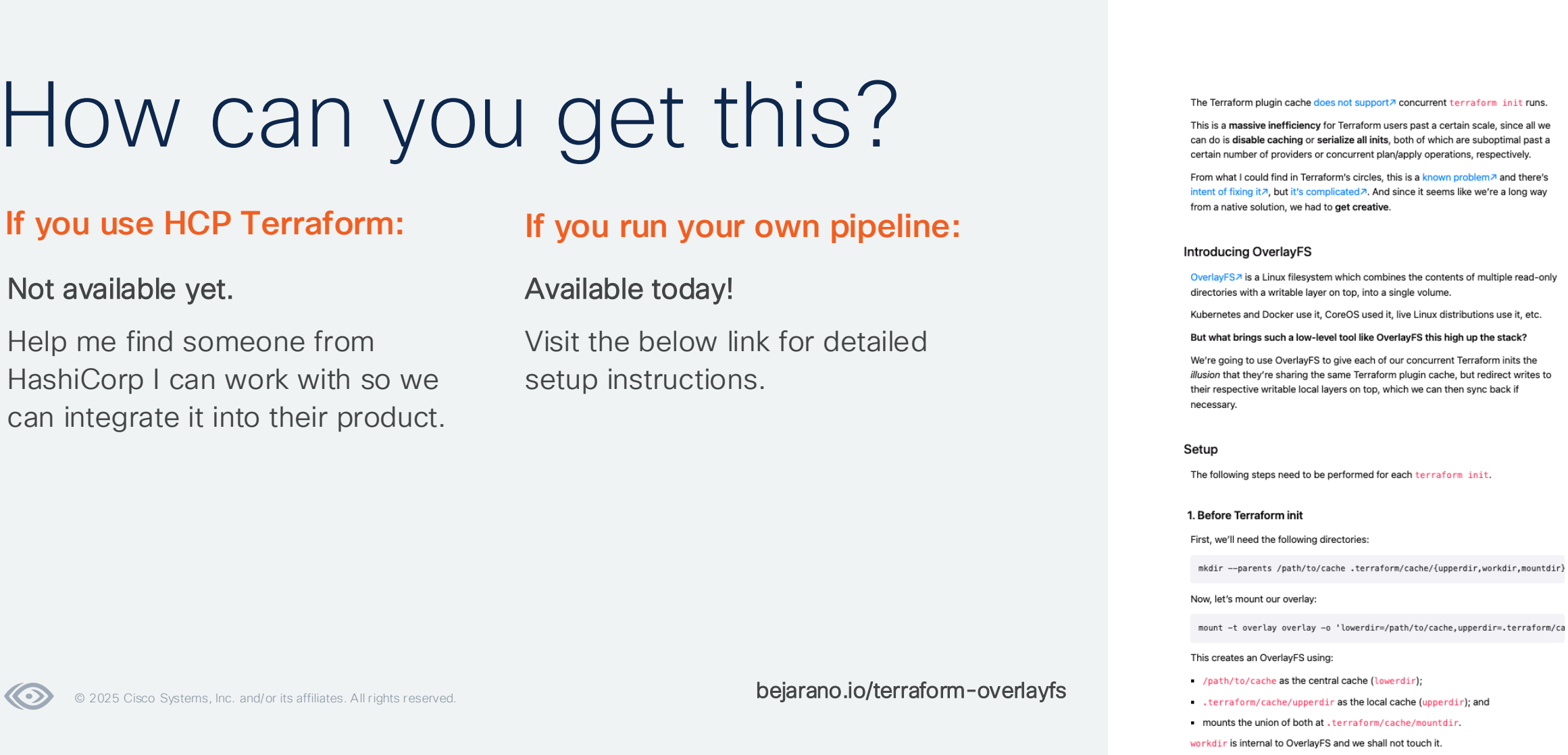
Not available yet.

Help me find someone from HashiCorp I can work with so we can integrate it into their product.

If you run your own pipeline:

Available today!

Visit the below link for detailed setup instructions.



Speeding up Terraform caching with OverlayFS

Published Apr 13, 2025 by Ricard Bejarano

The Terraform plugin cache does not support↗ concurrent `terraform init` runs.

This is a **massive inefficiency** for Terraform users past a certain scale, since all we can do is **disable caching** or **serialize all inits**, both of which are suboptimal past a certain number of providers or concurrent plan/apply operations, respectively.

From what I could find in Terraform's circles, this is a known problem↗ and there's intent of fixing it↗, but it's complicated↗. And since it seems like we're a long way from a native solution, we had to **get creative**.

Introducing OverlayFS

OverlayFS↗ is a Linux filesystem which combines the contents of multiple read-only directories with a writable layer on top, into a single volume.

Kubernetes and Docker use it, CoreOS used it, live Linux distributions use it, etc.

But what brings such a low-level tool like OverlayFS this high up the stack?

We're going to use OverlayFS to give each of our concurrent Terraform inits the *illusion* that they're sharing the same Terraform plugin cache, but redirect writes to their respective writable local layers on top, which we can then sync back if necessary.

Setup

The following steps need to be performed for each `terraform init`.

1. Before Terraform init

First, we'll need the following directories:

```
mkdir --parents /path/to/cache .terraform/cache/{upperdir,workdir,mountdir}
```

Now, let's mount our overlay:

```
mount -t overlay overlay -o 'lowerdir=/path/to/cache,upperdir=.terraform/ca
```

This creates an OverlayFS using:

- `/path/to/cache` as the central cache (`lowerdir`);
- `.terraform/cache/upperdir` as the local cache (`upperdir`); and
- mounts the union of both at `.terraform/cache/mountdir`.

`workdir` is internal to OverlayFS and we shall not touch it.

bejarano.io/terraform-overlayfs