Zero-Downtime Live Migration of Stateful VMs on Kubernetes

Felicitas Pojtinger
@pojntfx

Loophole Labs

Felicitas
Pojtinger

bluesky: @felicitas.pojtinger.com
mastodon: @pojntfx@mastodon.social
github/twitter: @pojntfx
linkedin: in/pojntfx
web: felicitas.pojtinger.com

Loophole Labs

VMs in the Container Age

High overhead

Resource-intensive

Slow to start


VIRTUAL MACHINE MANAGER


KVM


QEMU

So why is everything still powered by VMs?

(Or: Why doesn’t everyone just use containers?)

A "Container of Stateful Applications"?

Good networking support

Small image sizes

Low overhead

Fast to start

Easy to build & distribute

Easy to scale

What Could We Do With Such a New Compute Unit?

Simplifying deployment and debugging

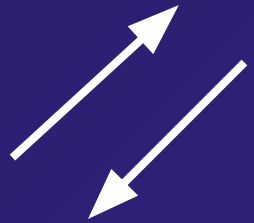


Image source: https://archive.ph/pBMM7, last accessed 2023-10-31

Moving between cloud providers without downtime

Moving long-running processes to more powerful servers (while continuing computing)

LLaMA C++

Image source: https://archive.ph/pL8ih, last accessed 2023-10-31

Migrating full desktop/mobile apps between devices (in milliseconds)

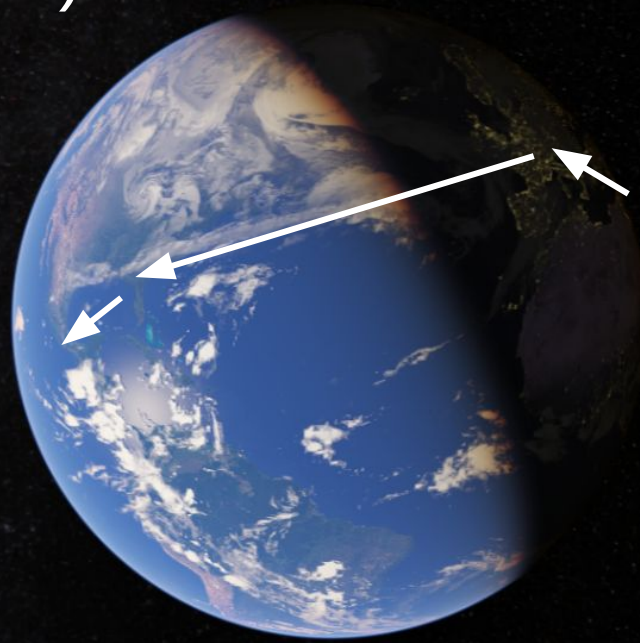
Waypipe
Project ID: 3237

Waypipe
Project ID: 3237

Waypipe
Project ID: 3237

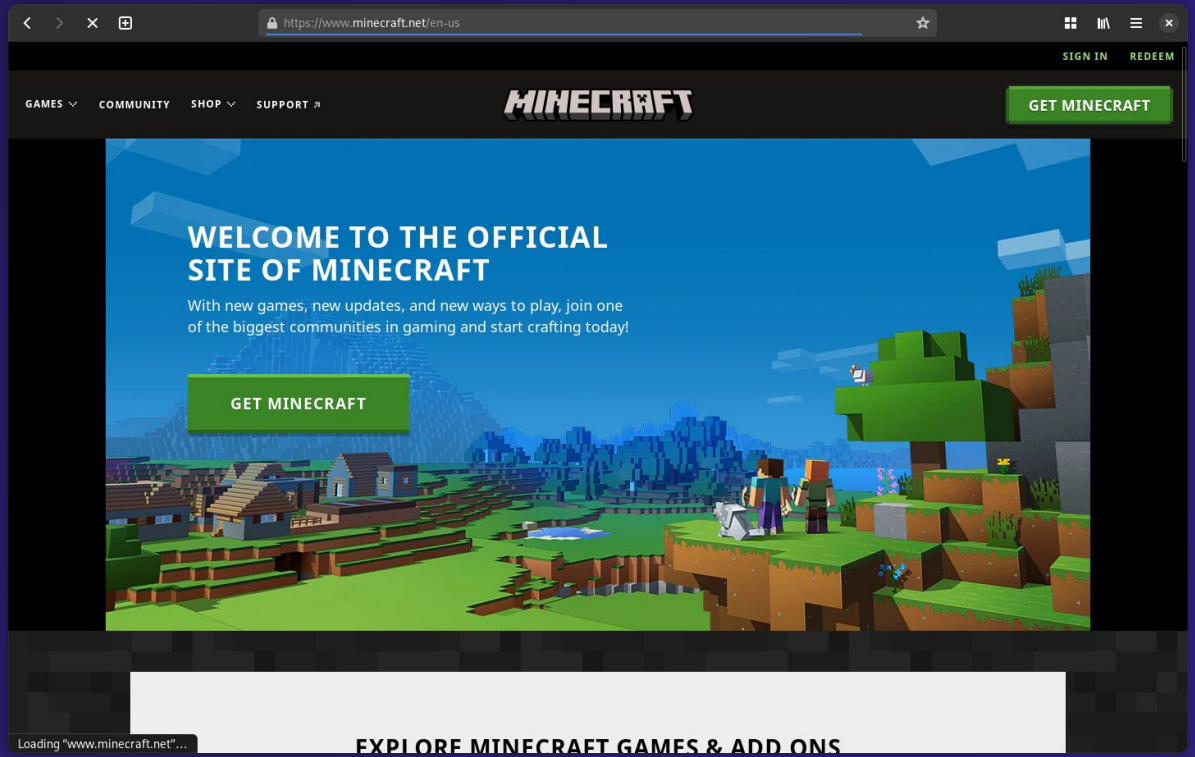
Image source: https://archive.ph/BWU0Q, last accessed 2023-10-31

Moving stateful services closer to your users as they access them (without downtime)



This sounds great

But it’s also impossible … right?



minecraft.net **(Version 1.12.2)**

Why Does This Not Exist Already?

Dated hypervisor ecosystems

Live migration is complex

VMs can’t be streamed in

IPs change after migrations

Outdated packaging tooling

Not designed for modern orchestrators

How Did We Implement It Anyway?

A (Very Brief) Overview of Existing Live Migration Techniques

Pre-Copy Migration

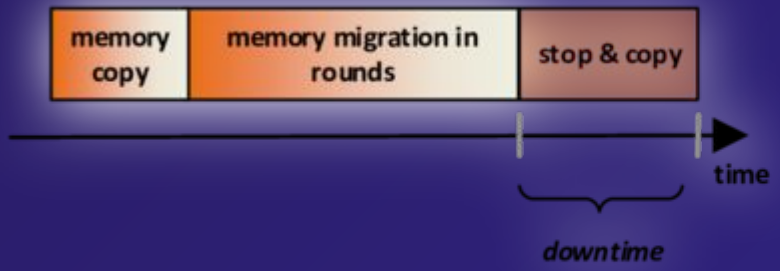


Image source: https://archive.ph/pBMM7, last accessed 2023-10-31

Post-Copy Migration

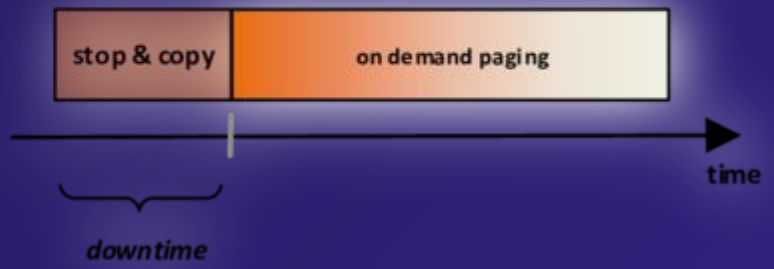


Image source: https://archive.ph/pBMM7, last accessed 2023-10-31

Remote Memory

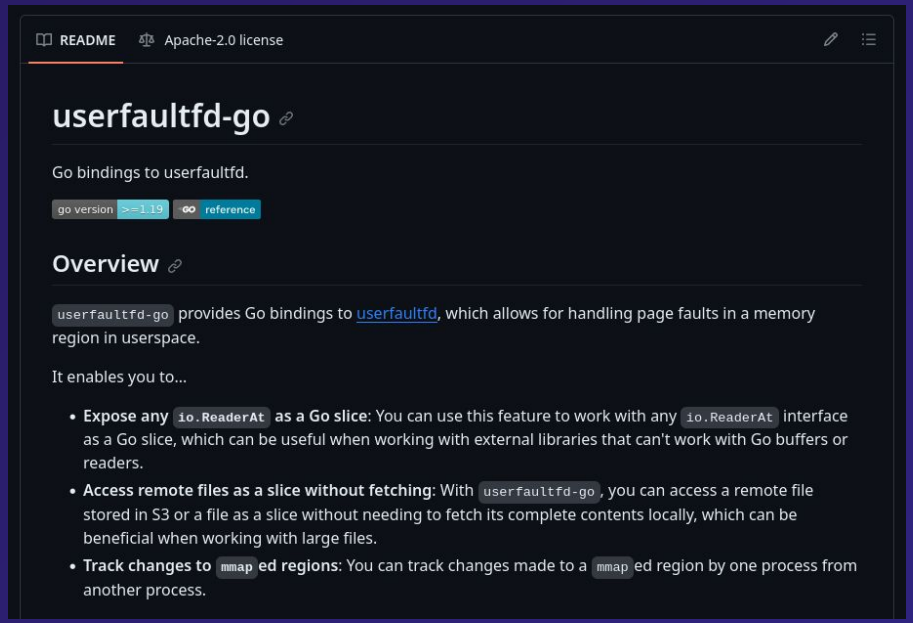
Highly application-specific

Very latency-sensitive

Dependent on specific transports

Not easily extensible

What about userfaultfd?



README Apache-2.0 license

userfaultfd-go

Go bindings to userfaultfd.

go version >=1.19 reference

Overview

`userfaultfd-go` provides Go bindings to userfaultfd, which allows for handling page faults in a memory region in userspace.

It enables you to...

- **Expose any `io.ReaderAt` as a Go slice**: You can use this feature to work with any `io.ReaderAt` interface as a Go slice, which can be useful when working with external libraries that can't work with Go buffers or readers.
- **Access remote files as a slice without fetching**: With `userfaultfd-go`, you can access a remote file stored in S3 or a file as a slice without needing to fetch its complete contents locally, which can be beneficial when working with large files.
- **Track changes to `mmap`ed regions**: You can track changes made to a `mmap`ed region by one process from another process.

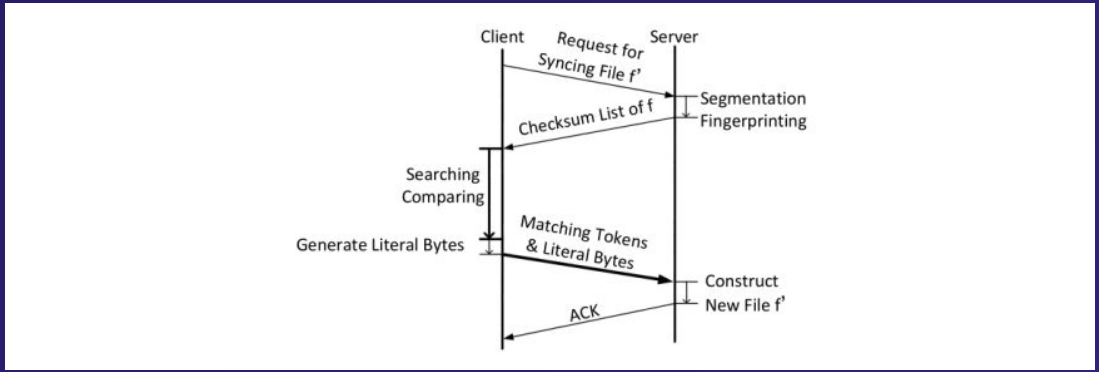
What about userfaultfd?

```
func (a abcReader) ReadAt(p []byte, off int64) (n int, err error) {
    n = copy(p, bytes.Repeat([]byte{'A' + byte(off%20)}, len(p)))

    return n, nil
}
```

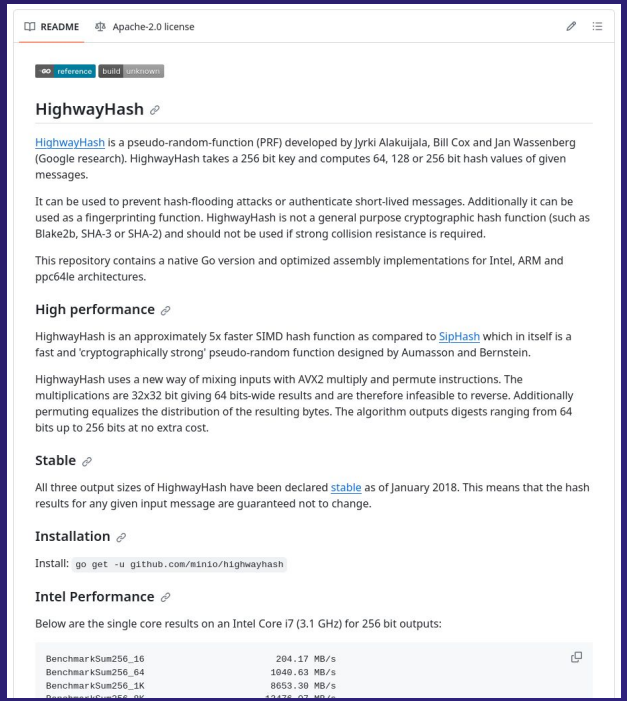
```
f, err := os.OpenFile(*file, os.O_RDONLY, os.ModePerm)
b, uffd, start, err := mapper.Register(int(s.Size()))
mapper.Handle(uffd, start, f)
```

```
// ...
f, err := mc.GetObject(ctx, *s3BucketName, *s3ObjectName, minio.GetObjectOptions{})
b, uffd, start, err := mapper.Register(int(s.Size()))
mapper.Handle(uffd, start, f)
```

File-based synchronization



File-based synchronization



README Apache-2.0 license

reference build unknown

HighwayHash

HighwayHash is a pseudo-random-function (PRF) developed by Jyrki Alakuijala, Bill Cox and Jan Wassenberg (Google research). HighwayHash takes a 256 bit key and computes 64, 128 or 256 bit hash values of given messages.

It can be used to prevent hash-flooding attacks or authenticate short-lived messages. Additionally it can be used as a fingerprinting function. HighwayHash is not a general purpose cryptographic hash function (such as Blake2b, SHA-3 or SHA-2) and should not be used if strong collision resistance is required.

This repository contains a native Go version and optimized assembly implementations for Intel, ARM and ppc64le architectures.

High performance

HighwayHash is an approximately 5x faster SIMD hash function as compared to SipHash which in itself is a fast and 'cryptographically strong' pseudo-random function designed by Aumasson and Bernstein.

HighwayHash uses a new way of mixing inputs with AVX2 multiply and permute instructions. The multiplications are 32x32 bit giving 64 bits-wide results and are therefore infeasible to reverse. Additionally permuting equalizes the distribution of the resulting bytes. The algorithm outputs digests ranging from 64 bits up to 256 bits at no extra cost.

Stable

All three output sizes of HighwayHash have been declared stable as of January 2018. This means that the hash results for any given input message are guaranteed not to change.

Installation

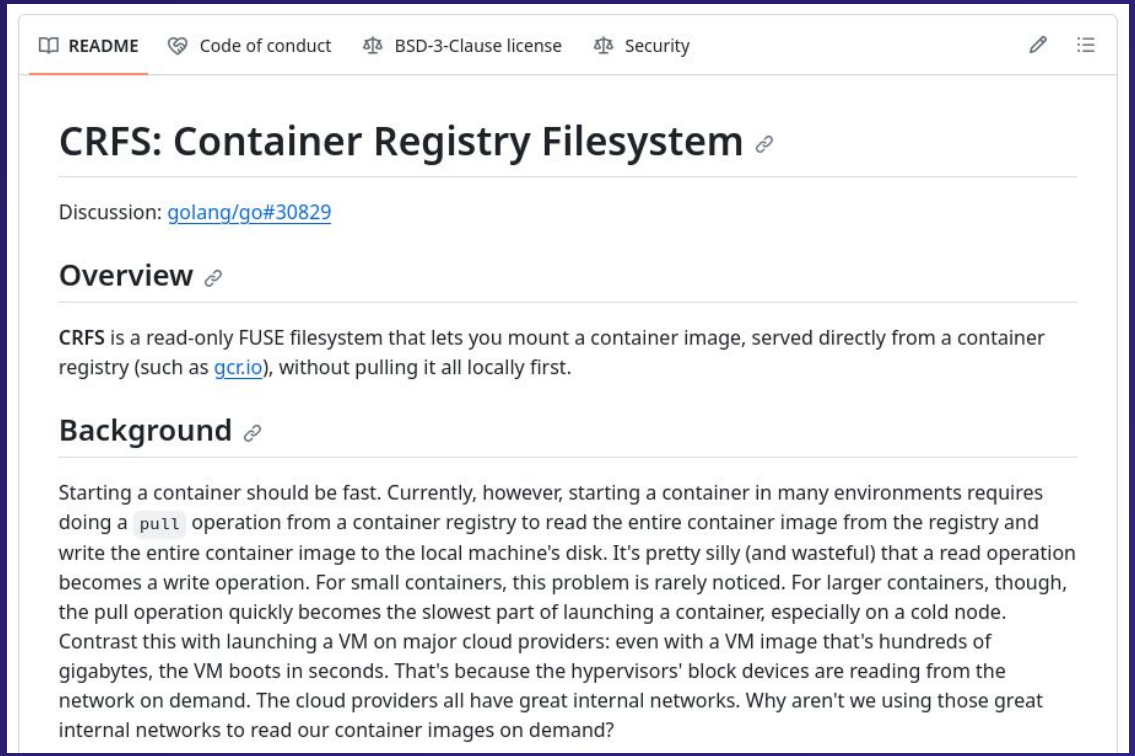
Install: `go get -u github.com/minio/highwayhash`

Intel Performance

Below are the single core results on an Intel Core i7 (3.1 GHz) for 256 bit outputs:

```
BenchmarkSum256_16                204.17 MB/s
BenchmarkSum256_64               1040.63 MB/s
BenchmarkSum256_1K               8653.30 MB/s
```

CRFS



README | Code of conduct | BSD-3-Clause license | Security

CRFS: Container Registry Filesystem

Discussion: golang/go#30829

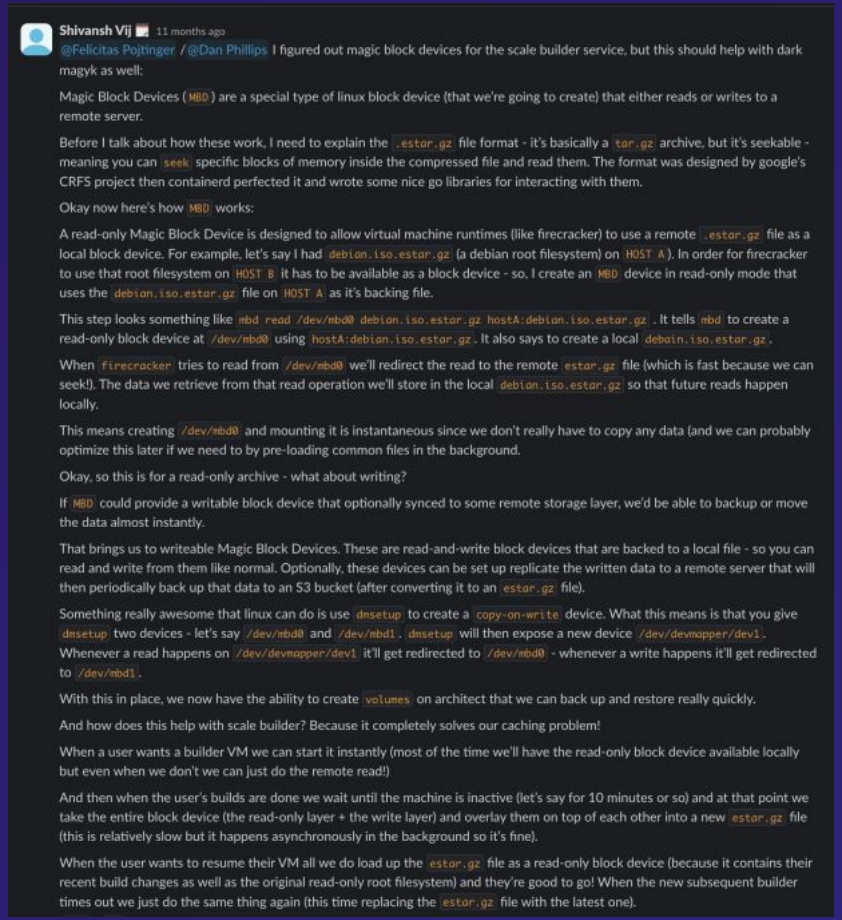
Overview

CRFS is a read-only FUSE filesystem that lets you mount a container image, served directly from a container registry (such as gcr.io), without pulling it all locally first.

Background

Starting a container should be fast. Currently, however, starting a container in many environments requires doing a `pull` operation from a container registry to read the entire container image from the registry and write the entire container image to the local machine's disk. It's pretty silly (and wasteful) that a read operation becomes a write operation. For small containers, this problem is rarely noticed. For larger containers, though, the pull operation quickly becomes the slowest part of launching a container, especially on a cold node. Contrast this with launching a VM on major cloud providers: even with a VM image that's hundreds of gigabytes, the VM boots in seconds. That's because the hypervisors' block devices are reading from the network on demand. The cloud providers all have great internal networks. Why aren't we using those great internal networks to read our container images on demand?

MBDs



Shivansh Vij 11 months ago

@Felicitas Pojtinger / @Dan Phillips I figured out magic block devices for the scale builder service, but this should help with dark magyk as well:

Magic Block Devices (`MBD`) are a special type of linux block device (that we're going to create) that either reads or writes to a remote server.

Before I talk about how these work, I need to explain the `.estar.gz` file format - it's basically a `tar.gz` archive, but it's seekable - meaning you can `seek` specific blocks of memory inside the compressed file and read them. The format was designed by google's CRFS project then containerd perfected it and wrote some nice go libraries for interacting with them.

Okay now here's how `MBD` works:

A read-only Magic Block Device is designed to allow virtual machine runtimes (like firecracker) to use a remote `.estar.gz` file as a local block device. For example, let's say I had `debian.iso.estar.gz` (a debian root filesystem) on `HOST A`). In order for firecracker to use that root filesystem on `HOST B` it has to be available as a block device - so, I create an `MBD` device in read-only mode that uses the `debian.iso.estar.gz` file on `HOST A` as it's backing file.

This step looks something like `mbd read /dev/mbd0 debian.iso.estar.gz hostA:debian.iso.estar.gz`. It tells `mbd` to create a read-only block device at `/dev/mbd0` using `hostA:debian.iso.estar.gz`. It also says to create a local `debain.iso.estar.gz`.

When `firecracker` tries to read from `/dev/mbd0` we'll redirect the read to the remote `estar.gz` file (which is fast because we can seek!). The data we retrieve from that read operation we'll store in the local `debian.iso.estar.gz` so that future reads happen locally.

This means creating `/dev/mbd0` and mounting it is instantaneous since we don't really have to copy any data (and we can probably optimize this later if we need to by pre-loading common files in the background.

Okay, so this is for a read-only archive - what about writing?

If `MBD` could provide a writable block device that optionally synced to some remote storage layer, we'd be able to backup or move the data almost instantly.

That brings us to writeable Magic Block Devices. These are read-and-write block devices that are backed to a local file - so you can read and write from them like normal. Optionally, these devices can be set up replicate the written data to a remote server that will then periodically back up that data to an S3 bucket (after converting it to an `estar.gz` file).

Something really awesome that linux can do is use `dmsetup` to create a `copy-on-write` device. What this means is that you give `dmsetup` two devices - let's say `/dev/mbd0` and `/dev/mbd1`. `dmsetup` will then expose a new device `/dev/devmapper/dev1`. Whenever a read happens on `/dev/devmapper/dev1` it'll get redirected to `/dev/mbd0` - whenever a write happens it'll get redirected to `/dev/mbd1`.

With this in place, we now have the ability to create `volumes` on architect that we can back up and restore really quickly.

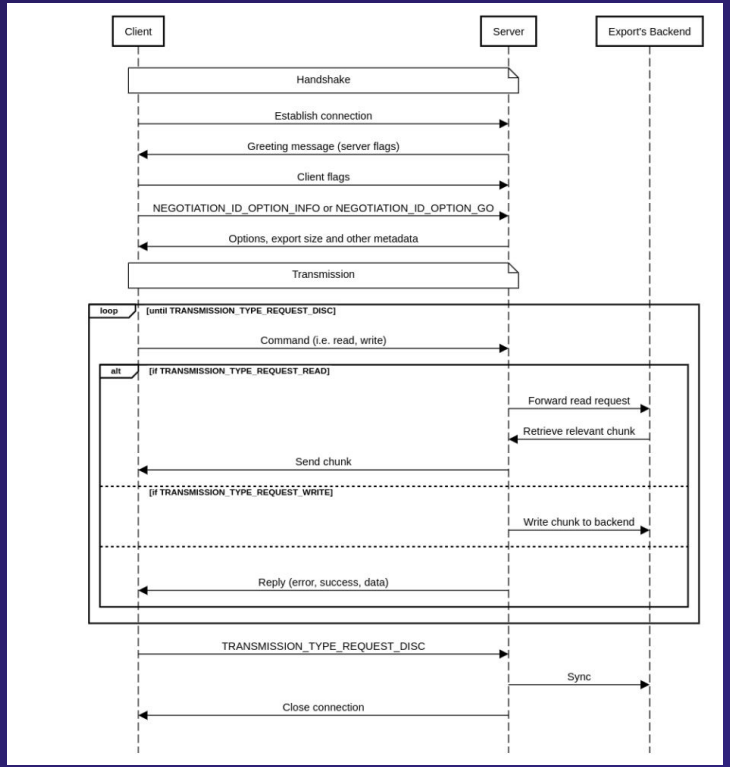
And how does this help with scale builder? Because it completely solves our caching problem!

When a user wants a builder VM we can start it instantly (most of the time we'll have the read-only block device available locally but even when we don't we can just do the remote read!)

And then when the user's builds are done we wait until the machine is inactive (let's say for 10 minutes or so) and at that point we take the entire block device (the read-only layer + the write layer) and overlay them on top of each other into a new `estar.gz` file (this is relatively slow but it happens asynchronously in the background so it's fine).

When the user wants to resume their VM all we do load up the `estar.gz` file as a read-only block device (because it contains their recent build changes as well as the original read-only root filesystem) and they're good to go! When the new subsequent builder times out we just do the same thing again (this time replacing the `estar.gz` file with the latest one).

NBD


Client
Server
Export's Backend
Handshake
Establish connection
Greeting message (server flags)
Client flags
NEGOTIATION_ID_OPTION_INFO or NEGOTIATION_ID_OPTION_GO
Options, export size and other metadata
Transmission
loop
[until TRANSMISSION_TYPE_REQUEST_DISC]
Command (i.e. read, write)
alt
[if TRANSMISSION_TYPE_REQUEST_READ]
Forward read request
Retrieve relevant chunk
Send chunk
[if TRANSMISSION_TYPE_REQUEST_WRITE]
Write chunk to backend
Reply (error, success, data)
TRANSMISSION_TYPE_REQUEST_DISC
Sync
Close connection

NBD with go-nbd

README Apache-2.0 license

go-nbd

Pure Go NBD server and client library.

hydrun CI passing | go version >=1.20 | reference | chat 1 users

Overview

go-nbd is a lean NBD server and client library supporting the baseline protocol.

It enables you to ...

- **Build NBD servers and clients in Go:** Develop Network Block Device servers and clients using the efficient and easy-to-understand Go programming language, without having to fallback to CGo.
- **Expose any `io.ReadWriter` as a block device:** Effortlessly turn a file, byte slice, S3 bucket or other `io.ReadWriter` into a fully-fledged block device.
- **Bridge with legacy services:** If you need to make your application's dynamic data available to a legacy system, providing a NBD interface can be the perfect solution.

NBD with go-nbd

```
type Backend interface {
    ReadAt(p []byte, off int64) (n int, err error)
    WriteAt(p []byte, off int64) (n int, err error)
    Size() (int64, error)
    Sync() error
}
```

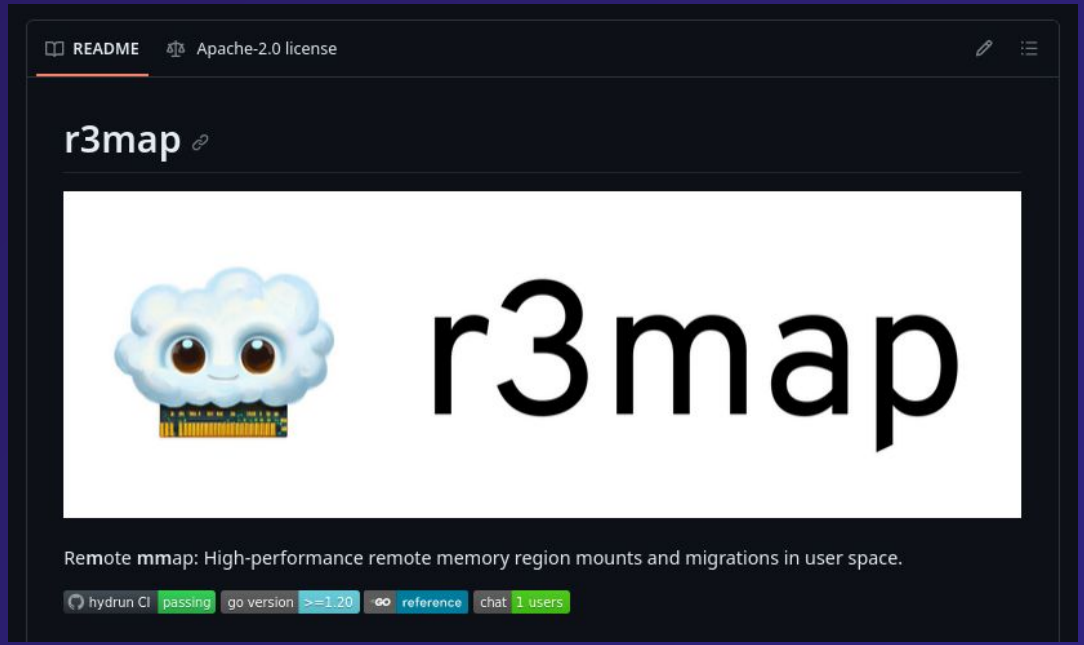
```
func Handle(conn net.Conn, exports []Export, options *Options) error
```

What about RTT?

r3map

A Universal Resource Mount and Migration Solution
(“re**m**ote **mm**ap”)

r3map


README
Apache-2.0 license
r3map
r3map
Remote mmap: High-performance remote memory region mounts and migrations in user space.
hydrun CI
passing
go version
>=1.20
reference
chat
1 users


Felicitas Pojtinger, 2023
Efficient Synchronization of Linux Memory Regions over a Network
A Comparative Study and Implementation

Concurrency

Lifecycle management/hooks

Pre-emptive pulls

Pull priority heuristics

Workload analysis

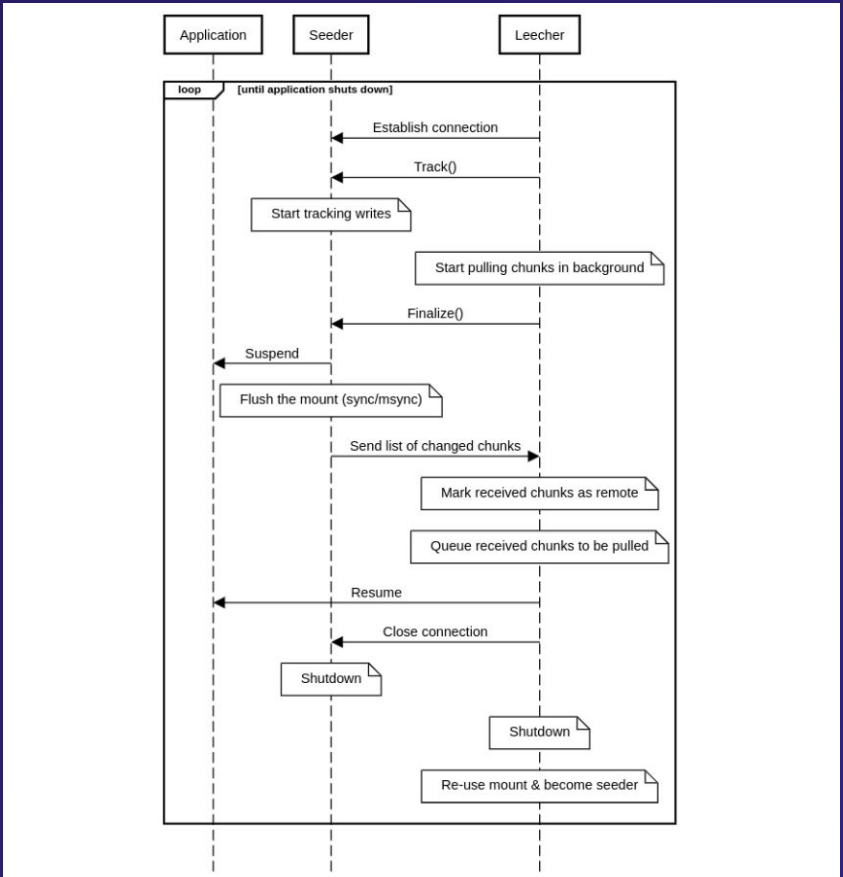
Network protocol

Caching

Mounts & migrations

Language-independent

Migration


Application
Seeder
Leecher
loop
[until application shuts down]
Establish connection
Track()
Start tracking writes
Start pulling chunks in background
Finalize()
Suspend
Flush the mount (sync/msync)
Send list of changed chunks
Mark received chunks as remote
Queue received chunks to be pulled
Resume
Close connection
Shutdown
Shutdown
Re-use mount & become seeder

Seeder

```go
defer mgr.Close()
file, svc, err := mgr.Seed()
if err != nil {
        panic(err)
}

log.Println("Starting app on", file.Name())
```

Leecher

```
conn, err := grpc.Dial(*raddr, grpc.WithTransportCredentials(insecure.NewCredentials()))
if err != nil {
        panic(err)
}
defer conn.Close()

log.Println("Leeching from", *raddr)

defer mgr.Close()
finalize, err := mgr.Leech(services.NewSeederRemoteGrpc(v1.NewSeederClient(conn)))
if err != nil {
        panic(err)
}
```

How Fast Is r3map?

Loophole Labs

Latency (0 ms RTT)

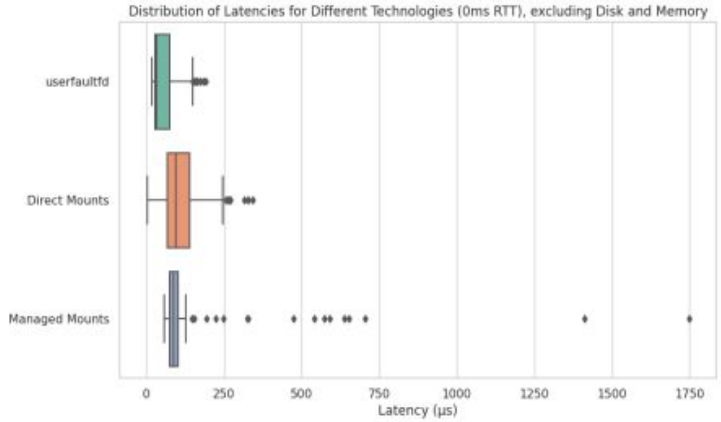


Figure 9: Box plot for the distribution of first chunk latency for userfaultfd, direct mounts and managed mounts (0ms RTT)

Latency (by RTT)

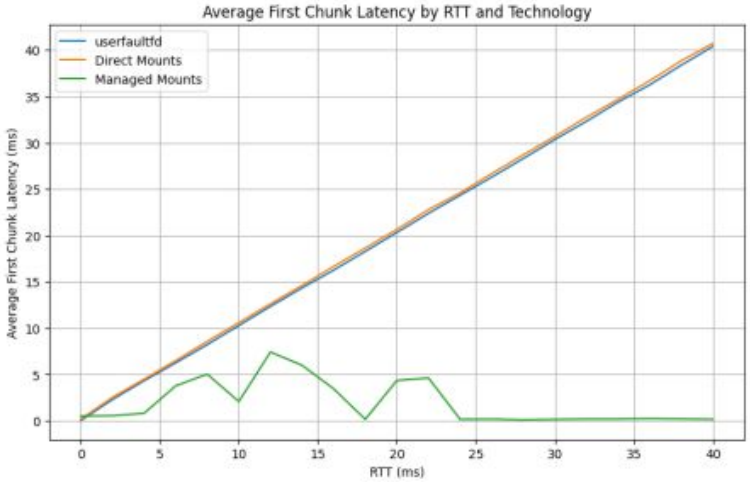


Figure 10: Average first chunk latency for userfaultfd, direct mounts and managed mounts by RTT

Read Throughput (0 ms RTT)

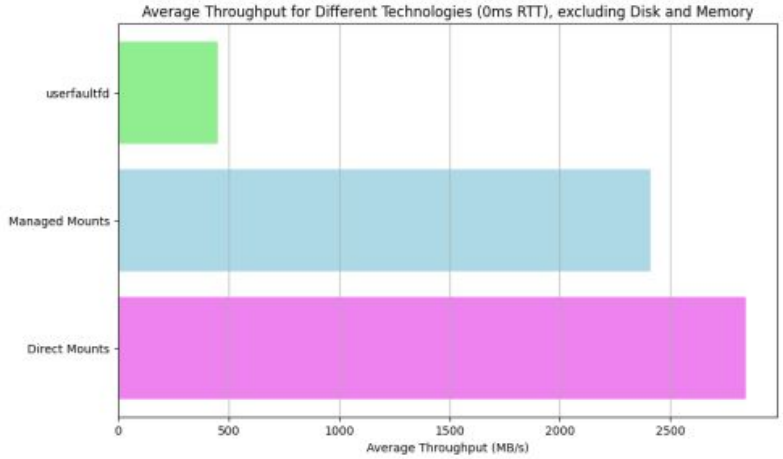


Figure 13: Average throughput for userfaultfd, direct mounts and managed mounts (0ms RTT)

Read Throughput (by RTT)

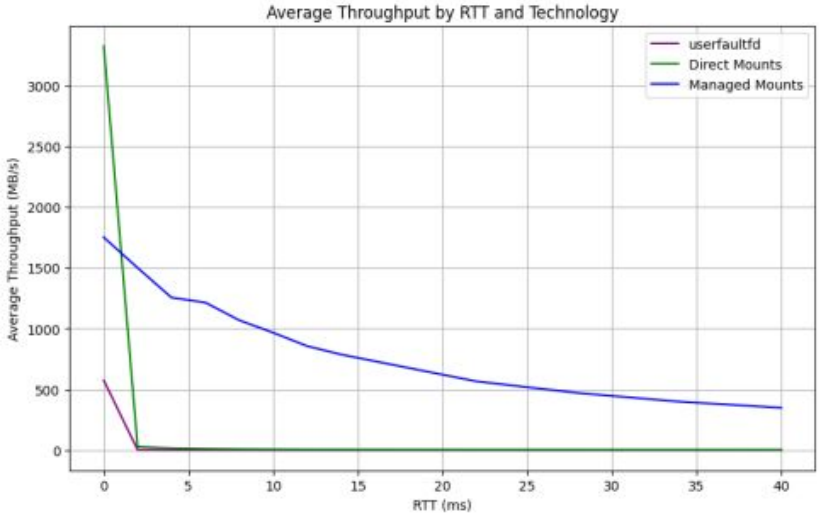


Figure 15: Average throughput for userfaultfd, direct mounts and managed mounts by RTT

Write Throughput (by RTT)

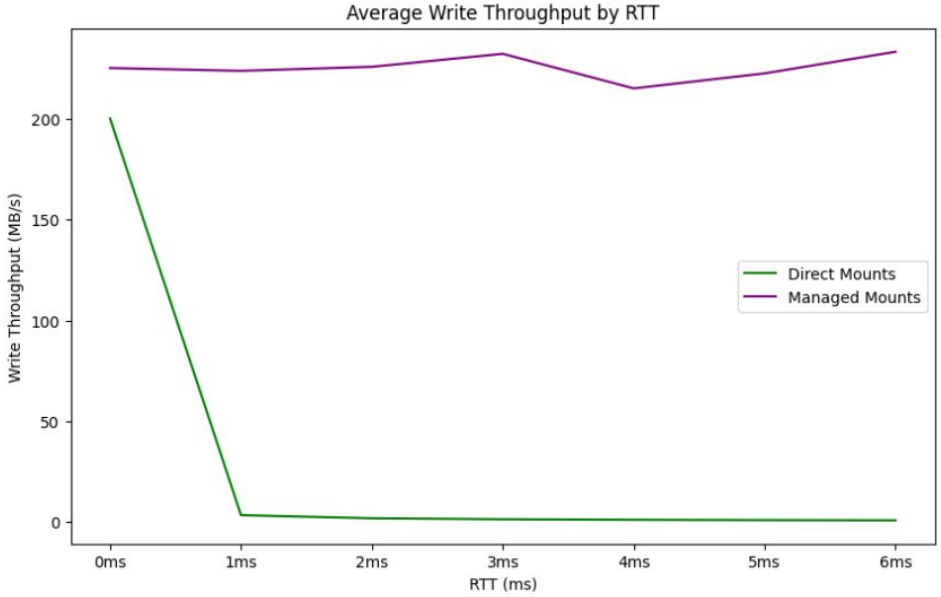
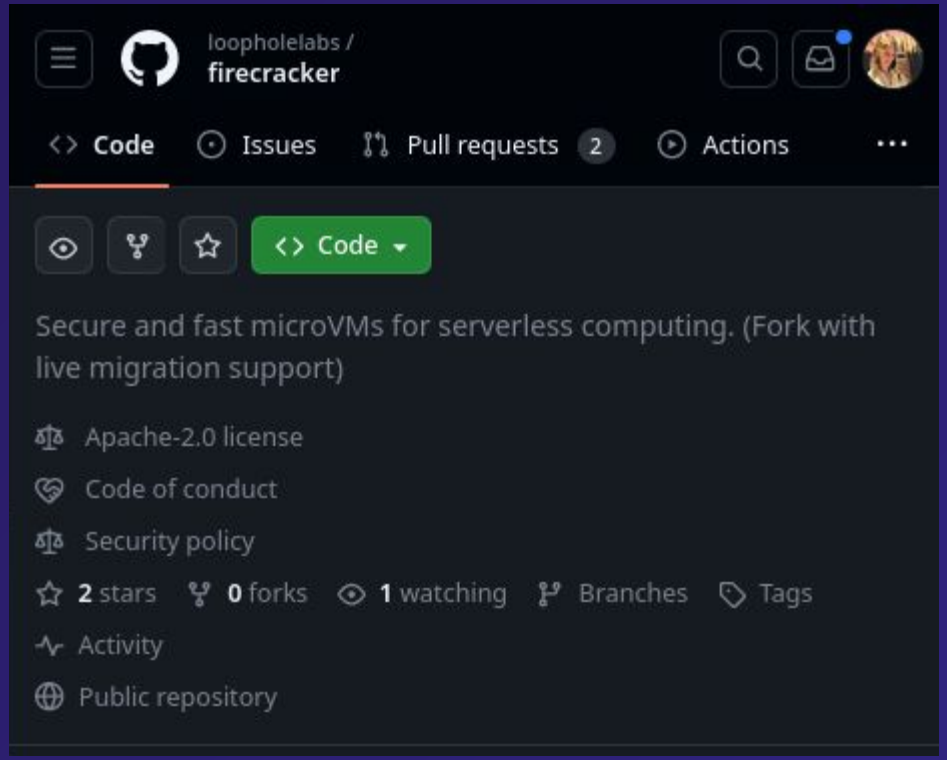


Figure 17: Average write throughput for direct and managed mounts by RTT

Using r3map with Firecracker


loopholelabs /
firecracker
Code
Issues
Pull requests 2
Actions
Code
Secure and fast microVMs for serverless computing. (Fork with live migration support)
Apache-2.0 license
Code of conduct
Security policy
2 stars
0 forks
1 watching
Branches
Tags
Activity
Public repository


4 src/utils/src/vm_memory.rs
@@ -119,8 +119,8 @@ pub fn create_guest_memory(
119
120 for region in regions {
121 let flags = match region.0 {
122 - None => libc::MAP_NORESERVE | libc::MAP_PRIVATE | libc::MAP_ANONYMOUS,
123 - Some(_) => libc::MAP_NORESERVE | libc::MAP_PRIVATE,
124 };
125
126 let mmap_region =
119
120 for region in regions {
121 let flags = match region.0 {
122 + None => libc::MAP_NORESERVE | libc::MAP_SHARED | libc::MAP_ANONYMOUS,
123 + Some(_) => libc::MAP_NORESERVE | libc::MAP_SHARED,
124 };
125
126 let mmap_region =


feat: Apply live migration patches for v1.5
live-migration-1.5
pojntfx committed 2 days ago
Unverified
Showing 10 changed files with 185 additions and 9 deletions.

Architekt

Tooling and orchestration for live migration with r3map and Firecracker

README | AGPL-3.0 license

Architekt

Zero-Downtime Live Migration of Stateful VMs on Kubernetes

Overview

🚧 This project is a work-in-progress! Instructions will be added as soon as it is usable. 🚧

Contributing

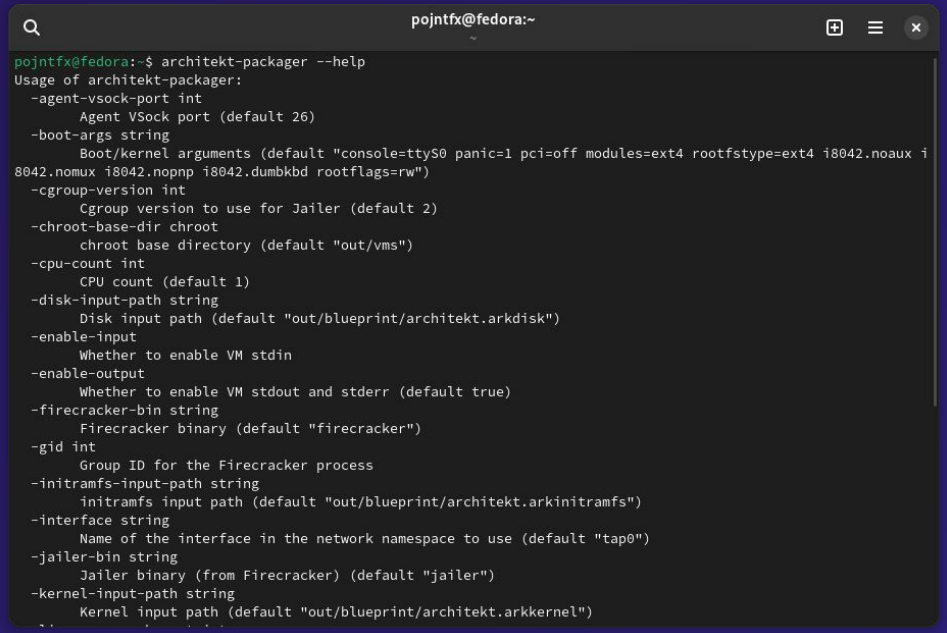
Bug reports and pull requests are welcome on GitHub at https://github.com/loopholelabs/architekt. For more contribution information check out the contribution guide.

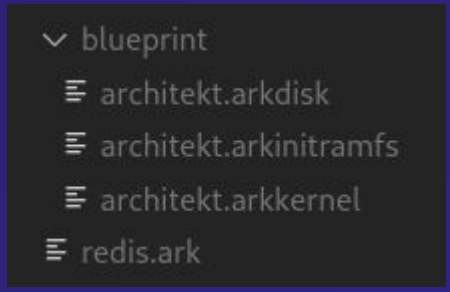
License

The Architekt project is available as open source under the terms of the GNU Affero General Public License, Version 3.

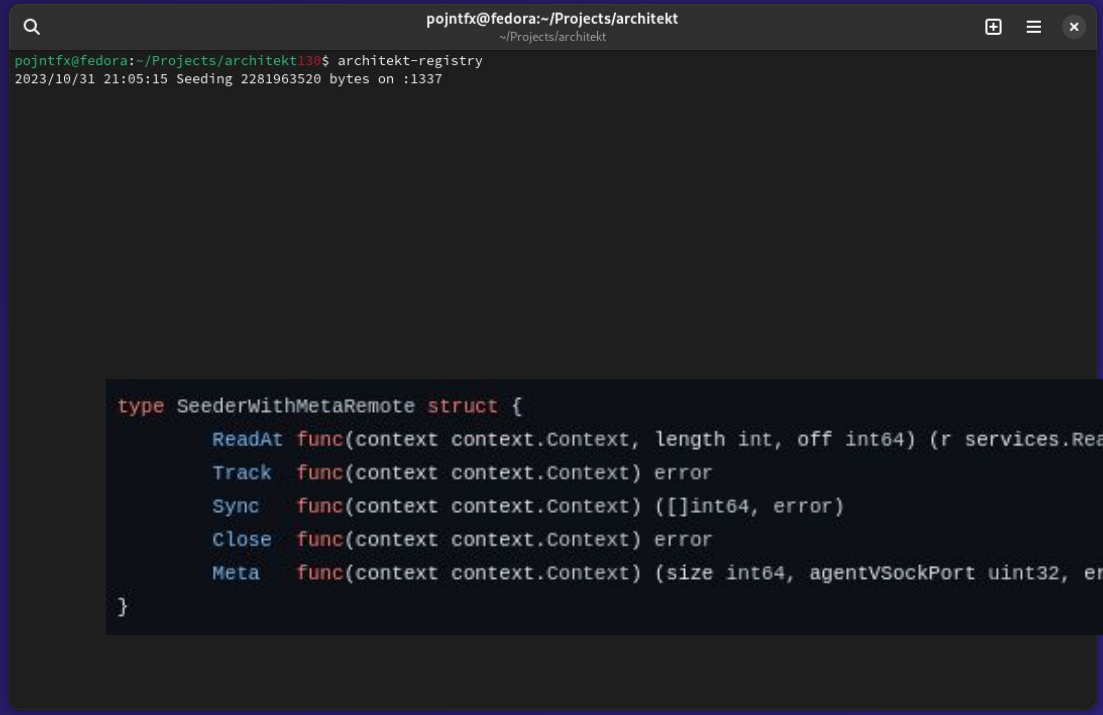
- architekt-agent
- architekt-daemon
- architekt-liveness
- architekt-manager
- architekt-operator
- architekt-packager
- architekt-peer
- architekt-registry
- architekt-runner
- architekt-worker

Architekt Packager

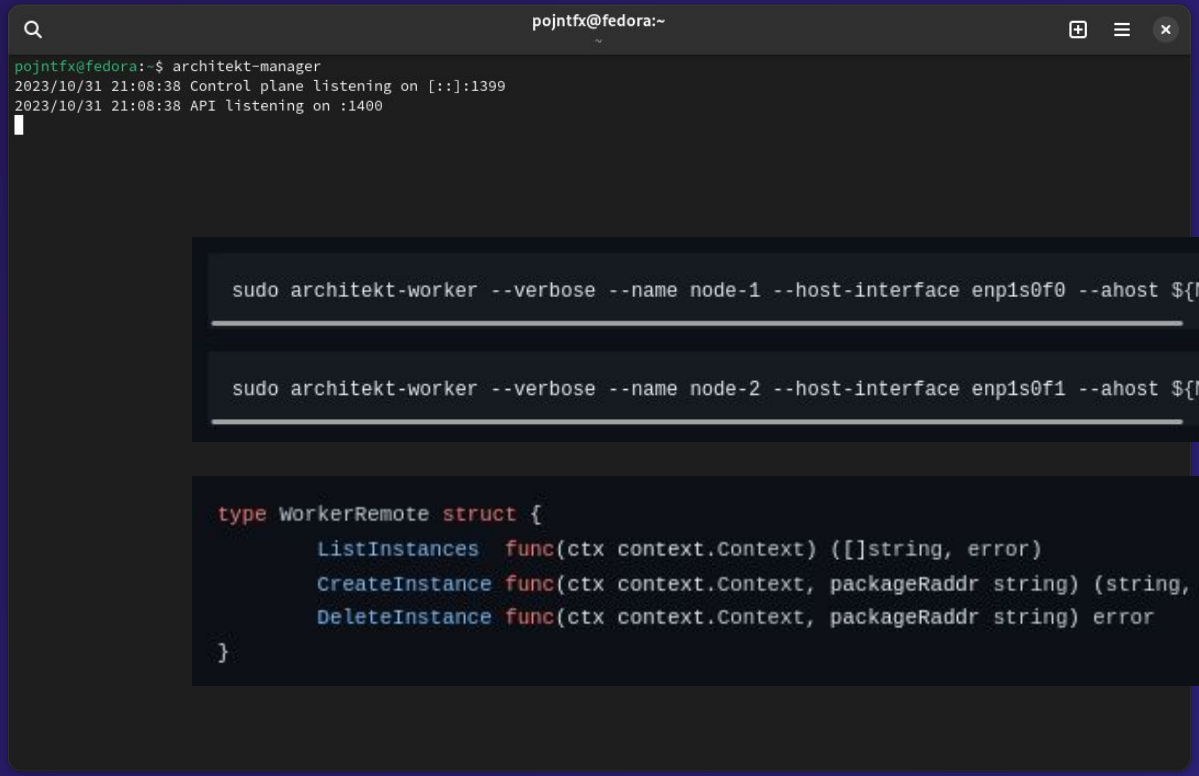




Architekt Registry



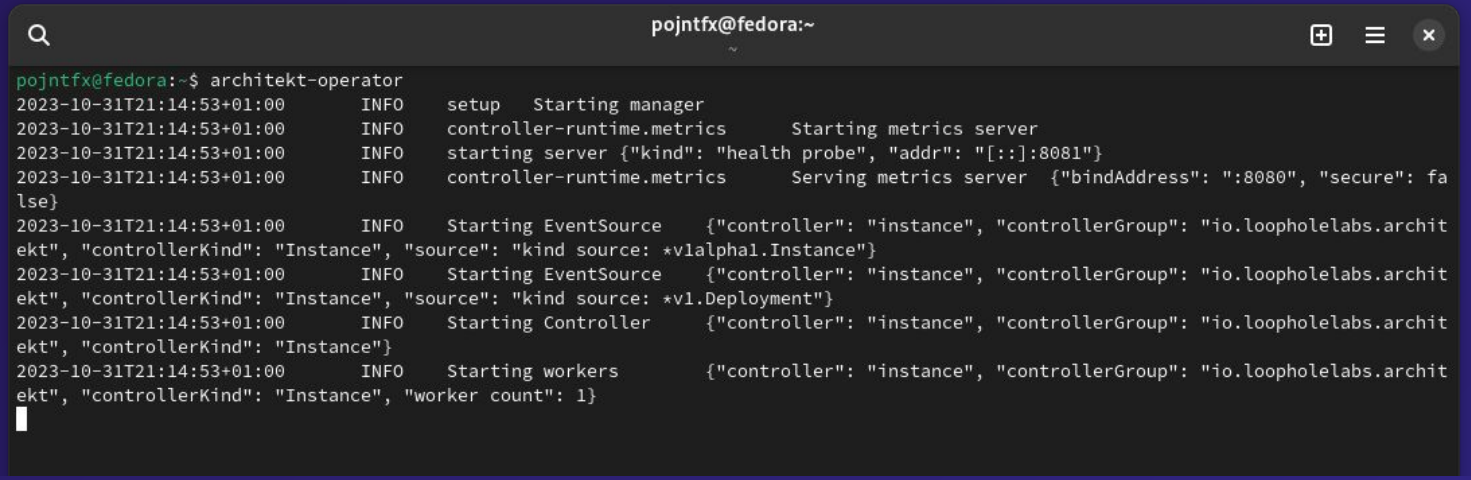
Architekt Control Plane



Architekt Control Plane API

```
curl -v http://${CONTROL_PLANE_IP}:1400/nodes | jq

curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances | jq
curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances | jq

export PACKAGE_RADDR=$(curl -v -X POST http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances/${REGISTRY_IP}:1337 | jq -r) # Create VM on node 1

curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances | jq
curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances | jq

export PACKAGE_RADDR=$(curl -v -X POST http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances/${PACKAGE_RADDR} | jq -r) # Migrate VM from node 1 to node 2

curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances | jq
curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances | jq

export PACKAGE_RADDR=$(curl -v -X POST http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances/${PACKAGE_RADDR} | jq -r) # Migrate VM from node 2 to node 1

curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances | jq
curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances | jq

curl -v -X DELETE http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances/${PACKAGE_RADDR} # Delete VM from node 2

curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-1/instances | jq
curl -v http://${CONTROL_PLANE_IP}:1400/nodes/node-2/instances | jq
```

Architekt Kubernetes Operator



Architekt Instance CRD

```yaml
apiVersion: io.loopholelabs.architekt/v1alpha1
kind: Instance
metadata:
  name: redis
spec:
  packageRaddr: localhost:1337 # If you change this, the VM is recreated from the new `packageRaddr`
  nodeName: minikube # If you change this, the VM is migrated
```

```
kubectl apply -f config/samples/architekt_v1alpha1_instance.yaml # Crea
kubectl get -o yaml instance.io.loopholelabs.architekt/redis # List VMs
kubectl apply -f config/samples/architekt_v1alpha1_instance.yaml # Migr
kubectl apply -f config/samples/architekt_v1alpha1_instance.yaml # Migr
kubectl delete -f config/samples/architekt_v1alpha1_instance.yaml # De
```

VM Network Interfaces

```
func NewNamespace(
        id string,

        hostInterface string,
        namespaceInterface string,

        namespaceInterfaceGateway string,
        namespaceInterfaceNetmask uint32,

        hostVethInternalIP string,
        hostVethExternalIP string,

        namespaceInterfaceIP string,
        namespaceVethIP string,

        blockedSubnet string,

        namespaceInterfaceMAC string,
) *Namespace {
```

```
if err := CreateNAT(d.hostInterface); err != nil {
        return err
}

hostVethIPs := NewIPTable(d.hostVethCIDR)
if err := hostVethIPs.Open(ctx); err != nil {
        return err
}

if err := d.namespaceVethIPs.Open(ctx); err != nil {
        return err
}
```

IP 1 → IP 2

Lynk

A new approach to networking

Independent of network topology

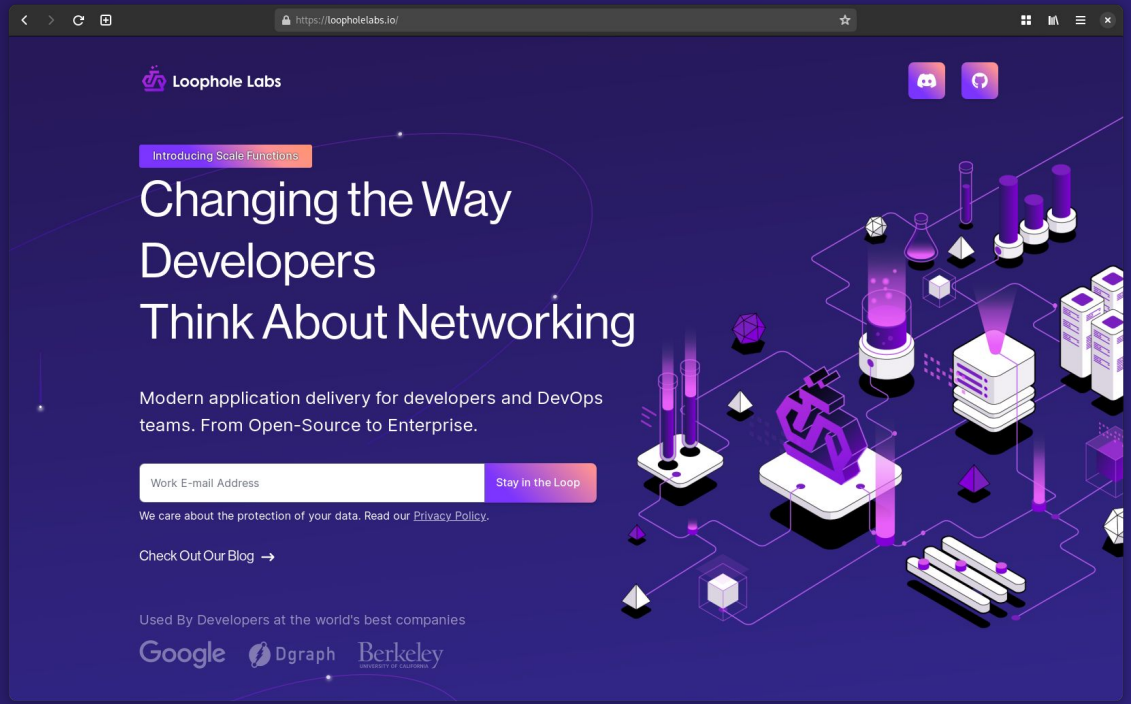
Can be deployed behind firewalls

Nodes do not require a public IP

Automatically handles TLS encryption in flight

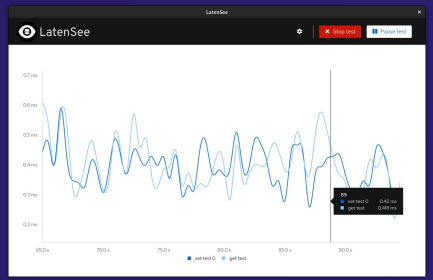
Can keep connections alive during migration

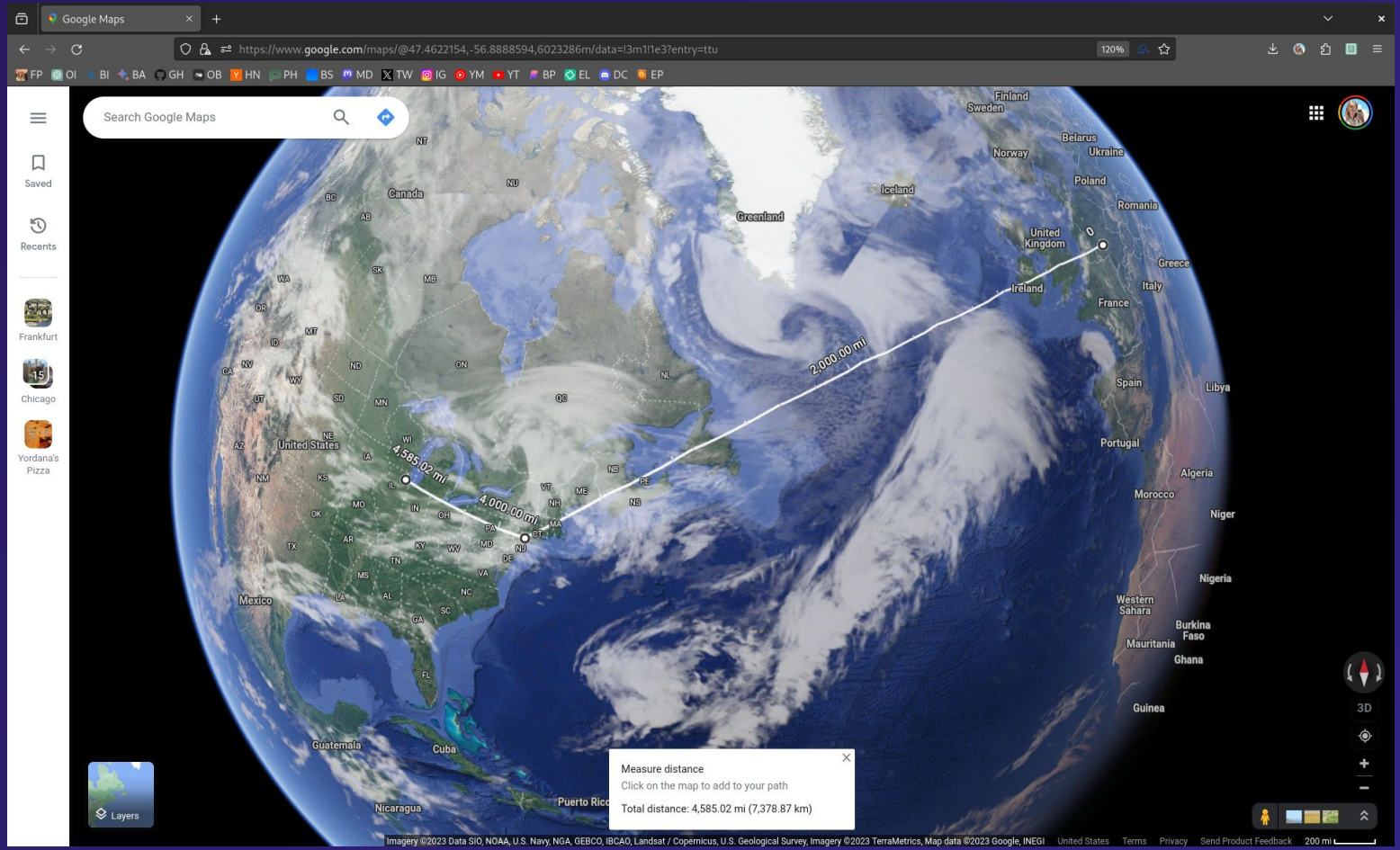
Tightly integrated with Architekt



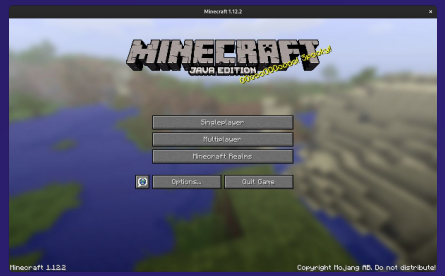
loopholelabs.io

But Does This Actually Work?


LatenSee
Stop test
Pause test
0.7 ms
0.6 ms
0.5 ms
0.4 ms
0.3 ms
0.2 ms
65.0 s
70.0 s
75.0 s
80.0 s
85.0 s
90.0 s
set test:0
get test


Google Maps
https://www.google.com/maps/@47.4622154,-56.8888594,6023286m/data=!3m1!1e3?entry=ttu
Search Google Maps
Saved
Recents
Frankfurt
Chicago
Yordana's Pizza
4,585.02 mi
4,000.00 mi
2,000.00 mi
Measure distance
Click on the map to add to your path
Total distance: 4,585.02 mi (7,378.87 km)
Layers
3D
Imagery ©2023 Data SIO, NOAA, U.S. Navy, NGA, GEBCO, IBCAO, Landsat / Copernicus, U.S. Geological Survey, Imagery ©2023 TerraMetrics, Map data ©2023 Google, INEGI
United States
Terms
Privacy
Send Product Feedback
200 mi

Live Demo


Minecraft 1.12.2
MINECRAFT
JAVA EDITION
Singleplayer
Multiplayer
Minecraft Realms
Options...
Quit Game
Minecraft 1.12.2
Copyright Mojang AB. Do not distribute!

Live Demo

Connect to:

minecraft.architect.run
Minecraft **1.12.2**

Recap: We Can We Do With This New Compute Unit?

Simplifying deployment and debugging

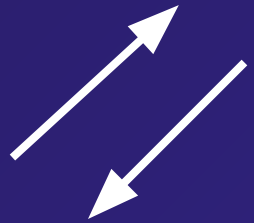


Image source: https://archive.ph/pBMM7, last accessed 2023-10-31

Moving between cloud providers without downtime

Moving long-running processes to more powerful servers (while continuing computing)

Image source: https://archive.ph/pL8ih, last accessed 2023-10-31

Migrating full desktop/mobile apps between devices (in milliseconds)

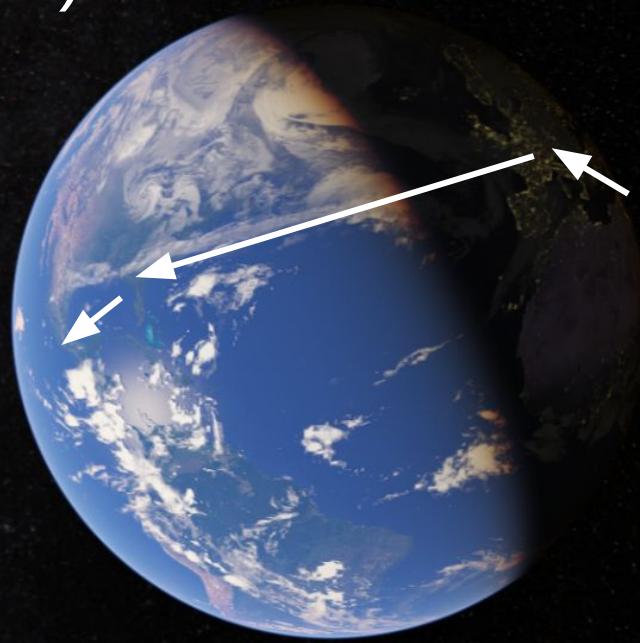
Waypipe
Project ID: 3237

Waypipe
Project ID: 3237

Waypipe
Project ID: 3237

Image source: https://archive.ph/BWU0Q, last accessed 2023-10-31

Moving stateful services closer to your users as they access them (without downtime)



pojntfx/networked-linux-memsync

Efficient Synchronization of Linux Memory Regions over a Network: A Comparative Study ...

7 0

Bachelor's Thesis

Efficient Synchronization of Linux Memory Regions over a Network

A Comparative Study and Implementation

Author: **Felicitas Pojtinger**
University: **Hochschule der Medien Stuttgart**
Course of Study: **Media Informatics**
Date: **2023-08-03**
Academic Degree: **Bachelor of Science**
Primary Supervisor: **Prof. Dr. Martin Goik**
Secondary Supervisor: **M.Sc. Philip Betzler**

Contents

Felicitas Pojtinger, 2023

Efficient Synchronization of Linux Memory Regions over a Network

A Comparative Study and Implementation

pojntfx/r3map

High-performance remote memory region mounts and migrations in user space.

★ 31 0

pojntfx/go-nbd

Pure Go NBD server and client library.

★ 305 15

pojntfx/networked-linux-memsync

Efficient Synchronization of Linux Memory Regions over a Network: A Comparative Study ...

★ 7 0

loopholelabs/architekt

Zero-Downtime Live Migration of Stateful VMs on Kubernetes

★ 5 0

loopholelabs/firecracker

Secure and fast microVMs for serverless computing. (Fork with live migration support)

★ 2 0

Come talk to us!

Loophole Labs

Felicitas
Pojtinger

bluesky: @felicitas.pojtinger.com
mastodon: @pojntfx@mastodon.social
github/twitter: @pojntfx
linkedin: in/pojntfx
web: felicitas.pojtinger.com

Loophole Labs

Loophole Labs

github/twitter:	@LoopholeLabs
discord:	loopholelabs.io/discord
web:	loopholelabs.io