Container Native FS Interposer

Julia Hua, Jiawei Xiang, Hilario Gonzalez, Juncheng Cao
Mentors: Vasily Tarasov, Alex Merenstein

October 9, 2024

Recap

Last sprint

- Project scaffolding (layout, CI, etc.)
- Dummy CSI driver
- Passthrough FUSE filesystem

This sprint

- Support CSI volume stacking
- CSI benchmark in CI
- Tracing FUSE filesytem

CSI volume stacking

CSI plugin for Kubernetes that allows to mount a **stackable** *FUSE-based file system over* **another file system**

Obstacle

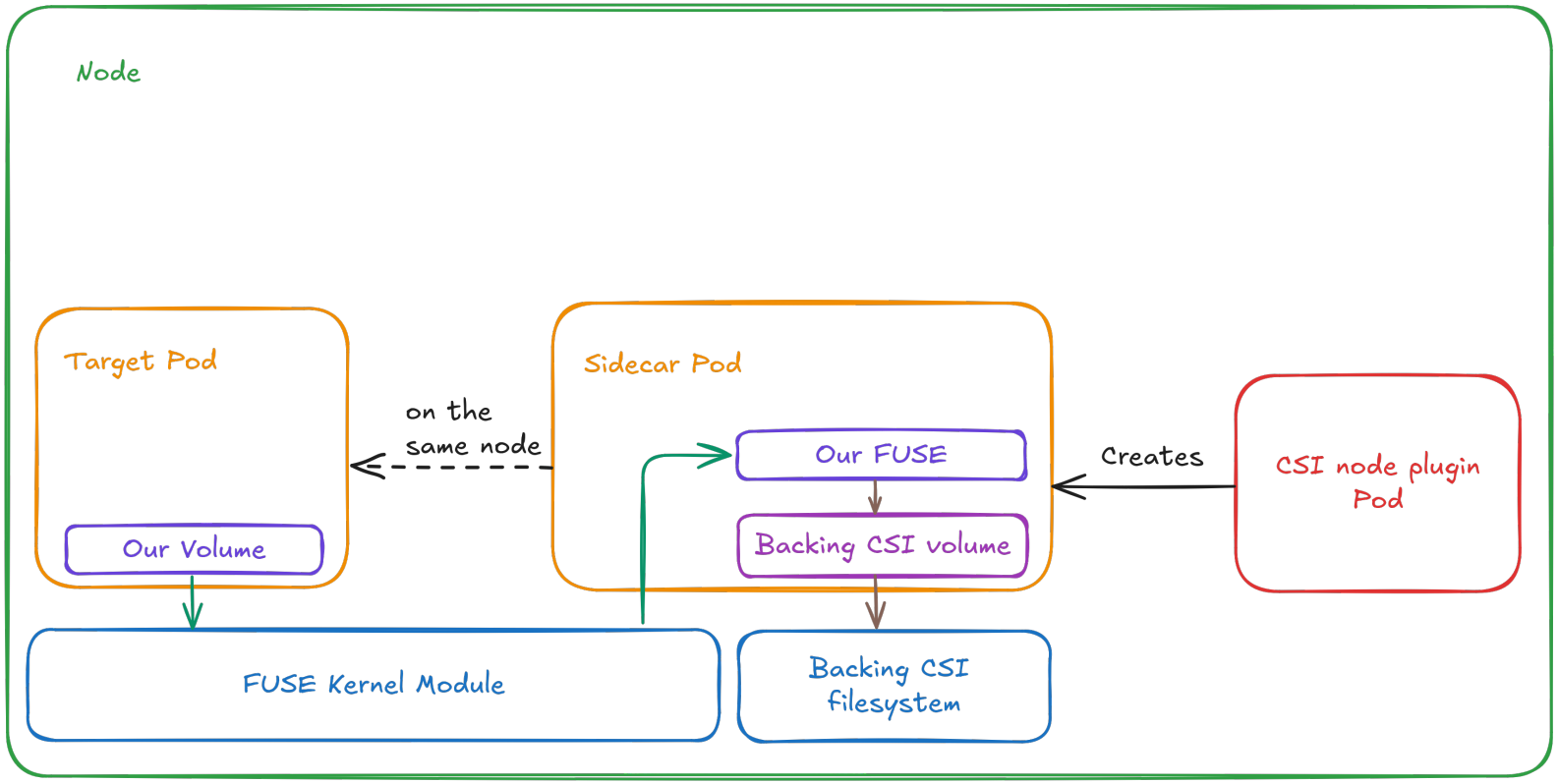
- CSI plugins don't have direct access to other CSI plugins

Solution

- Mount the backing CSI volume in an sidecar[1] pod
- Mount our FUSE filesystem in the sidecar pod
- Propagate the FUSE mount to the host

[1] on the same node as the target pod

CSI volume stacking



CSI benchmark

MinIO

MinIO is a high performance, distributed object storage system widely used with machine learning, analytics and cloud-native application workloads.

Warp

Warp is MinIO's official benchmarking tool with support for all S3 compatible storage systems and the ability to simulate a variety of workload.

CSI benchmark


```
initContainers:
  - name: minio
    image: quay.io/minio/minio:latest
    args: [ "server", "/data" ]
    volumeMounts:
      - mountPath: "/data"
        name: data
containers:
  - name: warp
    image: quay.io/minio/warp:latest
volumes:
  - name: data
    csi:
      driver: interposer.csi.example.com
      volumeAttributes:
        persistentVolumeClaimName: <backing volume>
```

Tracing

What

Tracking the activities of application under a specific workload

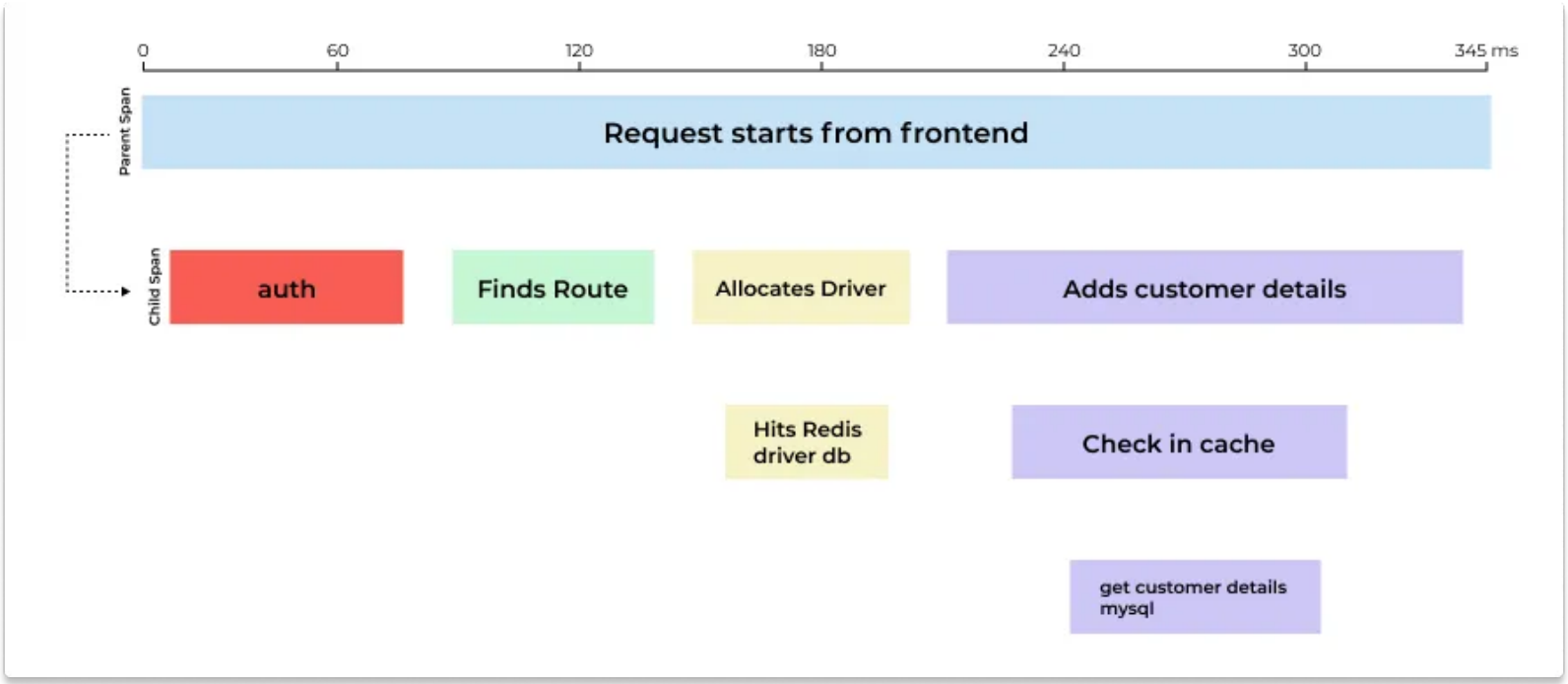
- file operation
- start and end time
- # of bytes read/written

Why

- Find performance bottlenecks
- Debug application
- Analyze system resources used

Tracing - OpenTelemetry

High-quality, ubiquitous, and portable telemetry to enable effective observability



IO Throughput Throttling

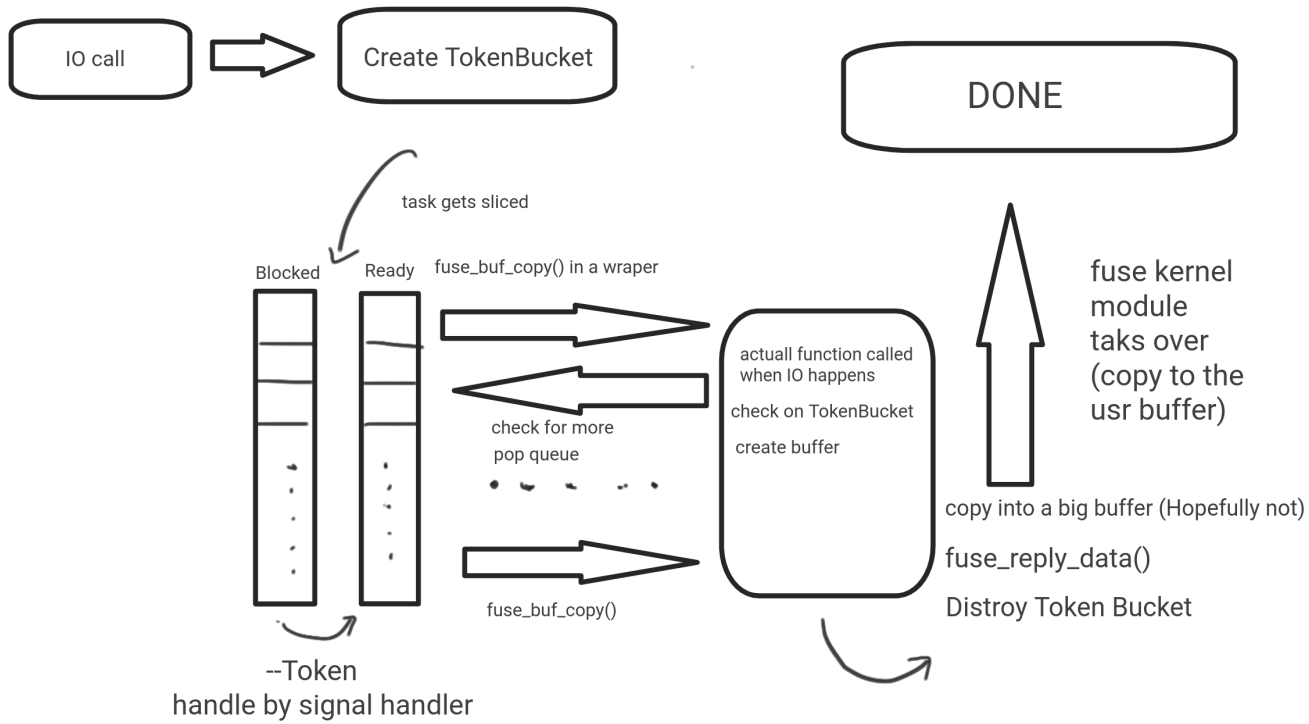
Done

- TokenBucket class
- Token replenishment timer

Working on

- rewrite read/write logic

Solution to new IO



Faulty and Delayed I/O

What

Introducing random faults in the file system to simulate hazards that can occur

- I/O truncation
- I/O failure
- Delay

Why

Developers can interpose this file system to test how applications will handle failure from the container

- Test fault tolerance of Application
- Emulate conditions like latency or hardware issues
- Identify performance bottlenecks

Faulty and Delayed I/O Logging

Developer can trace simulate faults in log

- The file system will log instances of simulated error
- timestamps are included
- developers can compare this log to see if application is catching all the errors

Future Plans

- Persist the data in timeseries database
- Metric collection with otel (in addition to tracing)
- Integrate the logging with otel

Burndown

Title	Date	Moved to Done Estimated Hours	Moved to Done Actual Hours	Ideal Hours	Burndown Estimated	Burndown Actual	Team #	Hours/Sprint
Sidecar pod	09/24/2024	6	6	80	74	74	*4*	*20*
Persistent volume	09/25/2024	2	2	74	72	72		
Lowlevel FUSE and benchmark	09/26/2024	8	8	69	61	61		
	09/27/2024	0	0	63	63	63		
	09/28/2024	0	0	57	57	57		
OpenTelemetry and TokenBucket	09/29/2024	10	10	51	41	41		
Virtiofs and Token replenish	09/30/2024	10	10	46	36	36		
C -> C++	10/01/2024	3	3	40	37	37		
Fault injection	10/02/2024	4	4	34	30	30		
Delay injection	10/03/2024	4	4	29	25	25		
	10/04/2024	0	0	23	23	23		
	10/05/2024	0	0	17	17	17		
	10/06/2024	0	0	11	11	11		
Prepare for demo	10/07/2024	9	9	6	-3	-3		
	10/08/2024	0	0	0	0	0