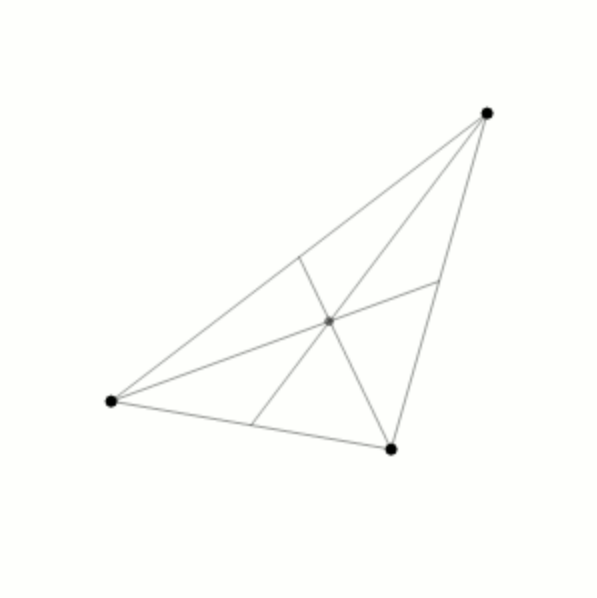
The Three-Tier Problem of CNG

2 Lessons + 1 Idea from 20 PoCs



Yui Matsumura / yuiseki
Geolonia, Inc.
CNG Japan 2026

The main message

CNG is not only a storage problem.

It is also a compute problem.

Cloud-native geospatial storage has enabled something important:

request-time geospatial computation

What I mean by “Three-Tier Problem”

There are three execution environments:

Cloud / Edge / Browser or App

And we must decide where to place:

Storage / Compute / Representation

This placement problem is the Three-Tier Problem of CNG.

The design space

	Cloud	Edge	Browser / App
Storage			
Compute			
Representation			

There is no single correct placement.

The right design depends on storage size, transfer time, runtime constraints, update frequency and progress of technology.

How I got here

I built about 20 small PoCs around CNG data.

Not to benchmark everything.

Not to propose a complete architecture.

But to understand one question:

What becomes possible when

CNG data is treated as computable input?

Two lessons + one idea

Lesson 1: Computability

A tile can be a request-time computation result.

Lesson 2: Antigravity

Cloud-native geospatial data has both gravity and antigravity.

One idea: Symmetry

Cloud and Edge may share portable geospatial data products.

<Lesson_1>

Lesson 1: Computability

A vector tile is not only a pre-generated file.

A vector tile can be a function result.

Why vector tiles mattered

Starting from vector tile responses forced three constraints:

1. computation must be bounded by tile coordinates
2. output must be compact and renderable
3. the function must be fast enough to serve interactively

So the tile becomes a useful unit of geospatial computation.

Introduction to the Demo

`https://yuiseki.dev`

Demo 1: Dynamic vector tile on `yuiseki.dev`

https://yuiseki.dev/poc-cng-taroverture-openmaptiles/


0:00

</Lesson_1>

<Lesson_2>

Lesson 2: Antigravity

`Data gravity` says:

large data tends to stay in the cloud.

But CNG data can also have `antigravity`.

When data is:

- chunked
- indexed
- range-readable
- self-describing

useful subsets can escape into portable products.

Examples of Data Antigravity

You can extract from...

- PMTile
- STAC and COG
- GeoPackage
- GeoParquet

CLI tool for data antigravity

- https://github.com/yuiseki/cng-data-antigravity

data moving across tiers:

cloud-scale source → portable asset → edge-deployable service

The data can move outward from cloud-scale storage.

Antigravity is not magic

Ideally, the subset preserves native format and provenance.

But the core idea is simpler:

CNG data can be large enough to stay in the cloud,

yet addressable enough to escape the cloud.

Practical concerns remain:

- provenance
- licensing
- update strategy
- runtime-specific indexes

</Lesson_2>

From Antigravity to Symmetry

If data can move from Cloud to Edge,
then we can ask a harder question:

Can Cloud and Edge share the same portable geospatial data product?

This is the idea I call:

Cloud-Edge Symmetric Geospatial

Edge Native Geospatial

In the real world,

There are already geospatial data formats that can be handled entirely on smartphones.

Example 1: OsmAnd

Example 2: Organic Maps

<One_Idea>

One idea: Cloud-Edge Symmetric Geospatial

This does not mean:

“everything runs everywhere.”

It means:

reduce rebuilds and reformatting between Cloud and Edge

by sharing a portable geospatial data product.

Symmetry across tiers

Capability	Cloud	Edge	Browser
search	scalable service	portable local API	local / cached query
routing	managed backend	local routing graph	lightweight interaction
tiling	FaaS / tile API	local tile API	protocol / rendering
analysis	batch / serverless	bounded local compute	constrained client compute

Demo 2.1: POI search on `yuiseki.dev`

https://yuiseki.dev/poc-cesg-poi-search/


0:00

There is no `search engine` anywhere

There are only files and functions

Demo 2.2: Route search on `yuiseki.dev`

https://yuiseki.dev/poc-cesg-route-search/


0:00

There is no `routing engine` anywhere

There are only files and functions

Now, I’d like to share a shocking fact with you

Please be careful of heart attacks

`yuiseki.dev` is a `k3s` cluster on 6 Raspberry Pis

The demo you just saw is running on this cluster.

You can own your cloud in your room

OK, Let's take a deep breath

Why this is still only an idea

Edge-native geospatial apps are already exists.

Examples include smartphone map apps with:

- regional files
- POI search
- offline routing
- runtime-optimized formats

But these formats evolved independently.

The hard part may not be technical.

It may be economic and institutional.

The third tier: Browser

Browser is not only a display surface.

It has its own storage and compute model:

- Service Worker
- Cache Storage
- OPFS
- Wasm

Example PoC:
https://github.com/yuiseki/poc-opfs-sw-tile

Today I focus mainly on Cloud → Edge,
but Browser is the third body in the same problem.

</One_Idea>

Final takeaway

The Three-Tier Problem of CNG

is the placement problem of:

```
Storage / Compute / Representation
```

across:

```
Cloud / Edge / Browser or App
```

2 Lessons + 1 Idea

2 Lessons

1. Computability
   tile = request-time computation result
2. Antigravity
   CNG data can escape from cloud-scale storage

1 Idea

3. Symmetry
   geospatial capabilities can move across Cloud / Edge / Browser or Apps

I look forward to working with all of you

To build the future of CNG.