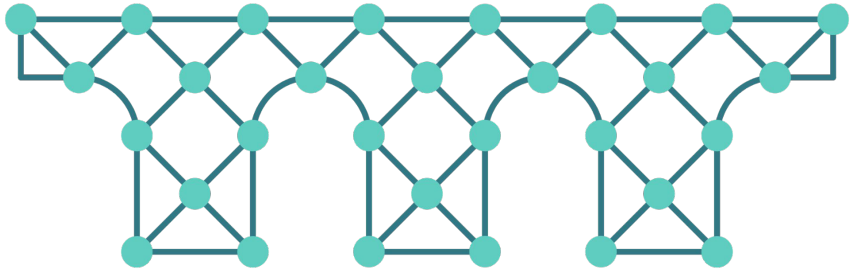
Sharding a GraphQL Gateway for Blast Radius Reduction

Linquan Zhang & Cetin Sahin

GraphQLConf 2026

airbnb



Viaduct

Airbnb's GraphQL API Gateway

Viaduct, by design

Unified Schema

One unified GraphQL schema serving the entire organization.

Tenant Ownership

Tenants own schema and resolvers.

Shared Infrastructure

A monolithic execution environment running on one shared JVM.

Scale & Velocity

~600 tenants with updates deployed daily across the platform.

~600
tenant modules

>70%
of Airbnb's API traffic

1
shared fate

10 - 15 s

One bad query. The whole gateway goes down.

Resource isolation becomes **more and more critical** as the platform continues to scale

Traffic Sharding for GraphQL

improved resiliency and isolation

Goal

Reduce the blast radius through **better resource isolation** during the instances of faulty code execution

Three decisions defined our sharding journey

Policy

Routing

Infrastructure

Basic Policy: Sharding

Hash based sharding on **operation name**

```
query UserProfile {
  user(id: $id) {
    name
  }
}
```

Basic Policy: Sharding

Hash based sharding on **operation name**

```
query UserProfile {
  user(id: $id) {
    name
  }
}
```

Basic Policy: Sharding

Hash based sharding on **operation name**

```
shardingKey = hash(UserProfile)
```

```
query UserProfile {
  user(id: $id) {
    name
  }
}
```

Basic Policy: Sharding

Hash based sharding on **operation name**

```
shardingKey = hash(UserProfile)
```

```
query UserProfile {
  user(id: $id) {
    name
  }
}
```

Two operations with the ***same shardingKey*** end up on the same shard

Enhanced Policy: Shuffle Sharding

Reduce the probability of a single bad operation taking out “**neighboring**” operations
An operation is assigned to **multiple shards randomly**.

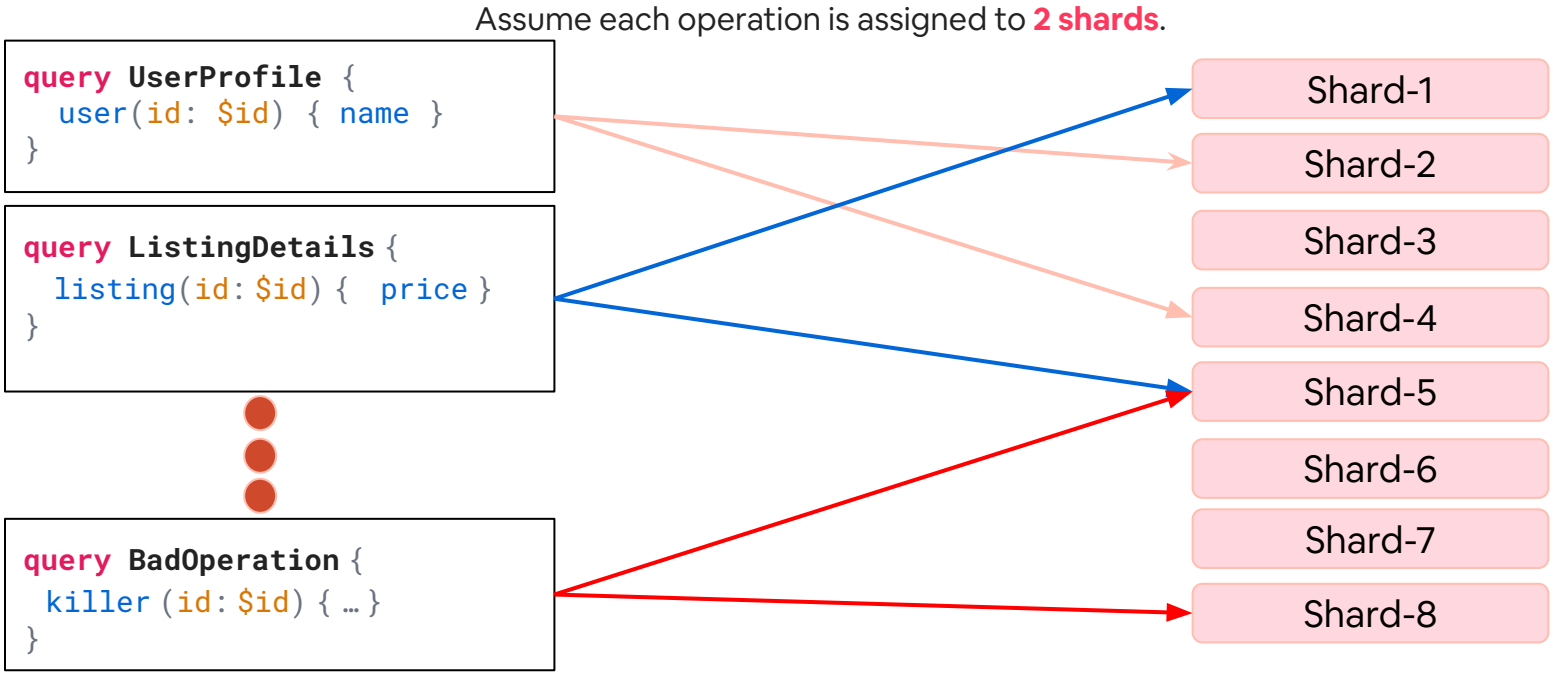
```
query UserProfile {
  user(id: $id) { name }
}
```

```
query ListingDetails {
  listing(id: $id) {  price }
}
```

```
query BadOperation {
  killer (id: $id) { … }
}
```

Enhanced Policy: Shuffle Sharding

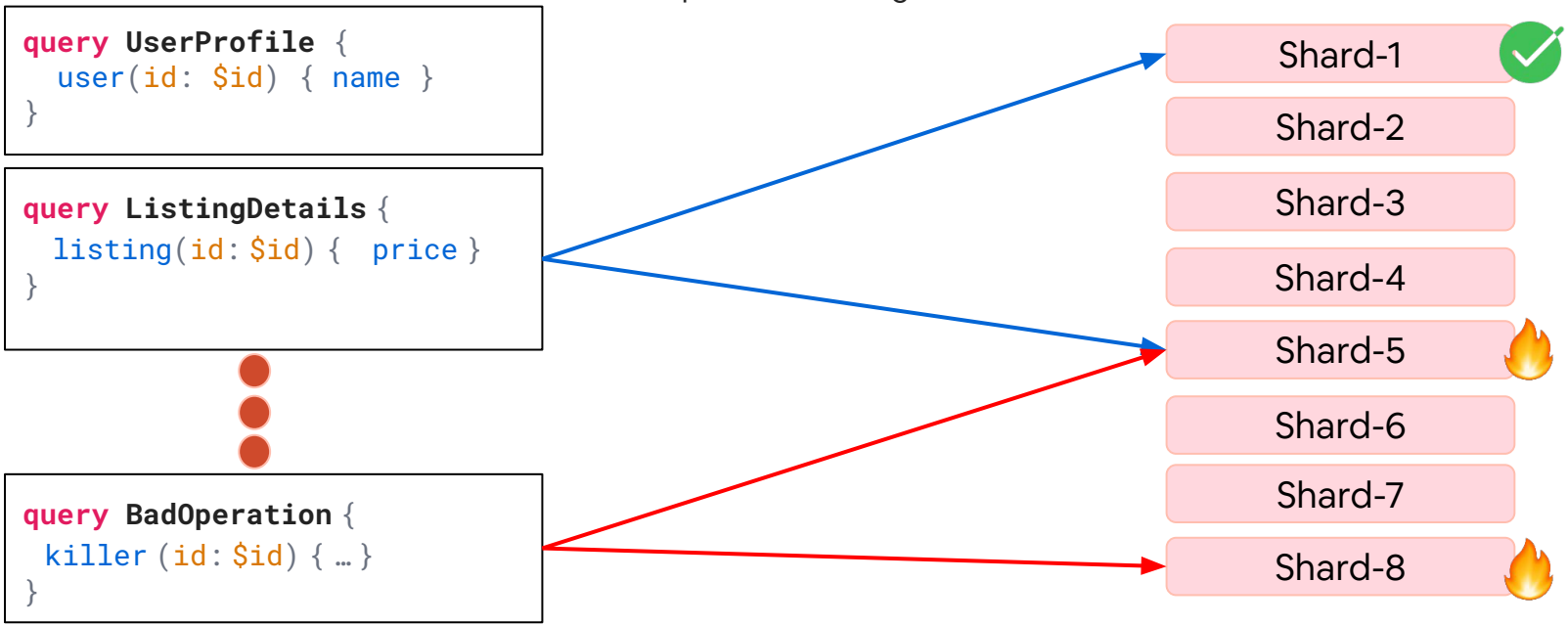
Reduce the probability of a single bad operation taking out "**neighboring**" operations
An operation is assigned to **multiple shards randomly**.



Enhanced Policy: Shuffle Sharding

Reduce the probability of a single bad operation taking out "**neighboring**" operations
An operation is assigned to **multiple shards randomly**.

Assume each operation is assigned to **2 shards**.



Routing

Sharding Dispatcher

A dispatcher in front of the gateway

Cost

Extra hop, extra component to operate.

Win

Change policy locally

Enabled rapid feature development

Infrastructure

Deployment Strategy

Enumerated Kubernetes deployments, not StatefulSet, not custom control plane.

Scalability Decisions

Each deployment (shard) scales independently.

Control Plane Philosophy

A custom control plane would have been more toil than value.

Architecture

Clients hit one endpoint. The dispatcher routes each operation to its shard.
Viaduct currently operates on 30 shards in production.

Each operation has 2 candidate shards, full collision drops to ~0.2%.

Remaining Gaps

Traffic Isolation

Online and offline traffic mixed in every shard.

Mitigation Speed

Slow mitigation: debugging and fixing identified issues takes hours

Experimentation Tier

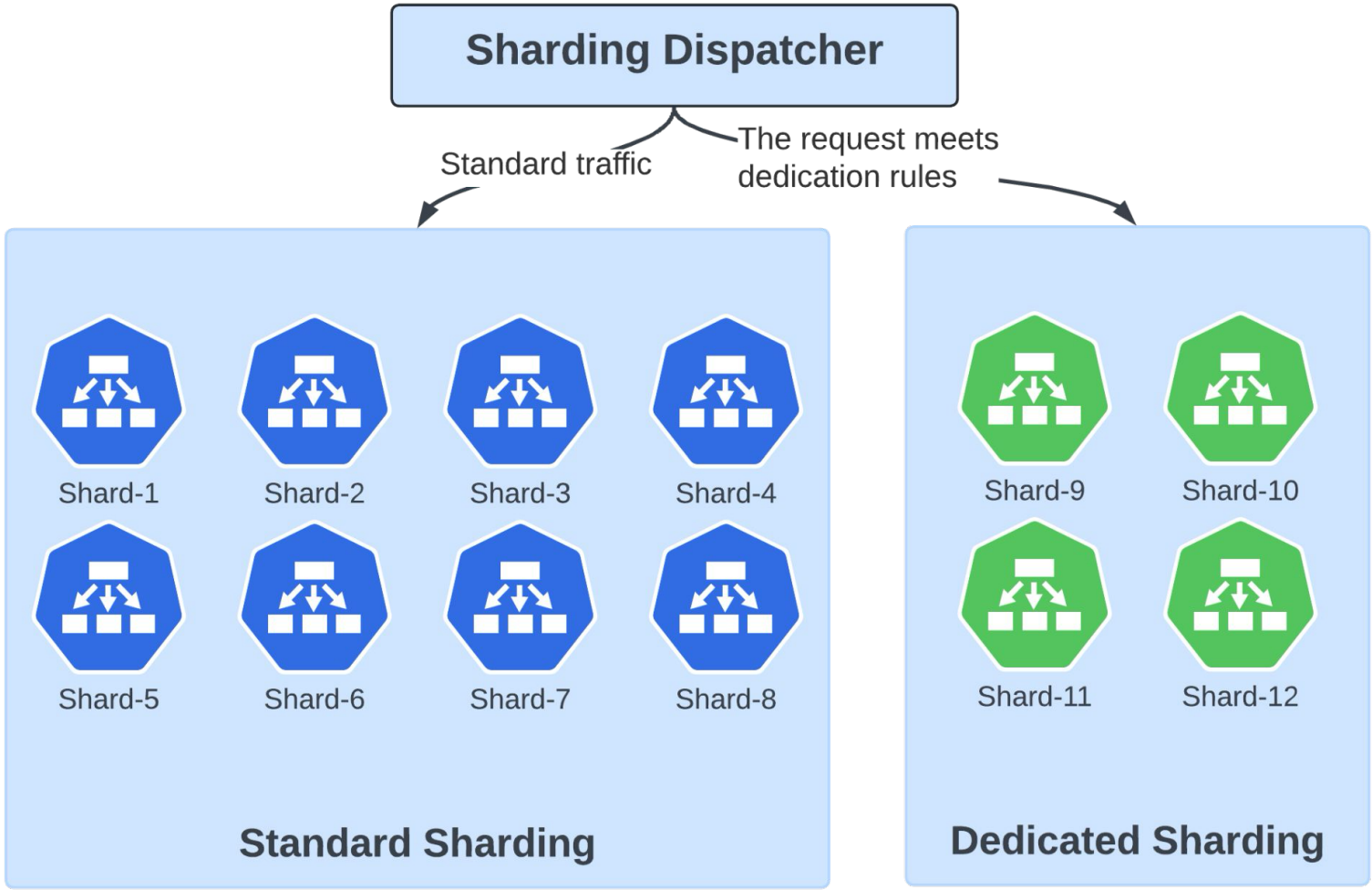
Lost the experimentation tier when shuffle replaced it.

Full Tenant Isolation

Preventing code execution issues from impacting neighboring tenants.

Dedicated Sharding

Shuffle sharding still runs. A rule layer goes in front.


Sharding Dispatcher
Standard traffic
The request meets dedication rules
Shard-1
Shard-2
Shard-3
Shard-4
Shard-5
Shard-6
Shard-7
Shard-8
Standard Sharding
Shard-9
Shard-10
Shard-11
Shard-12
Dedicated Sharding

Operational Rules

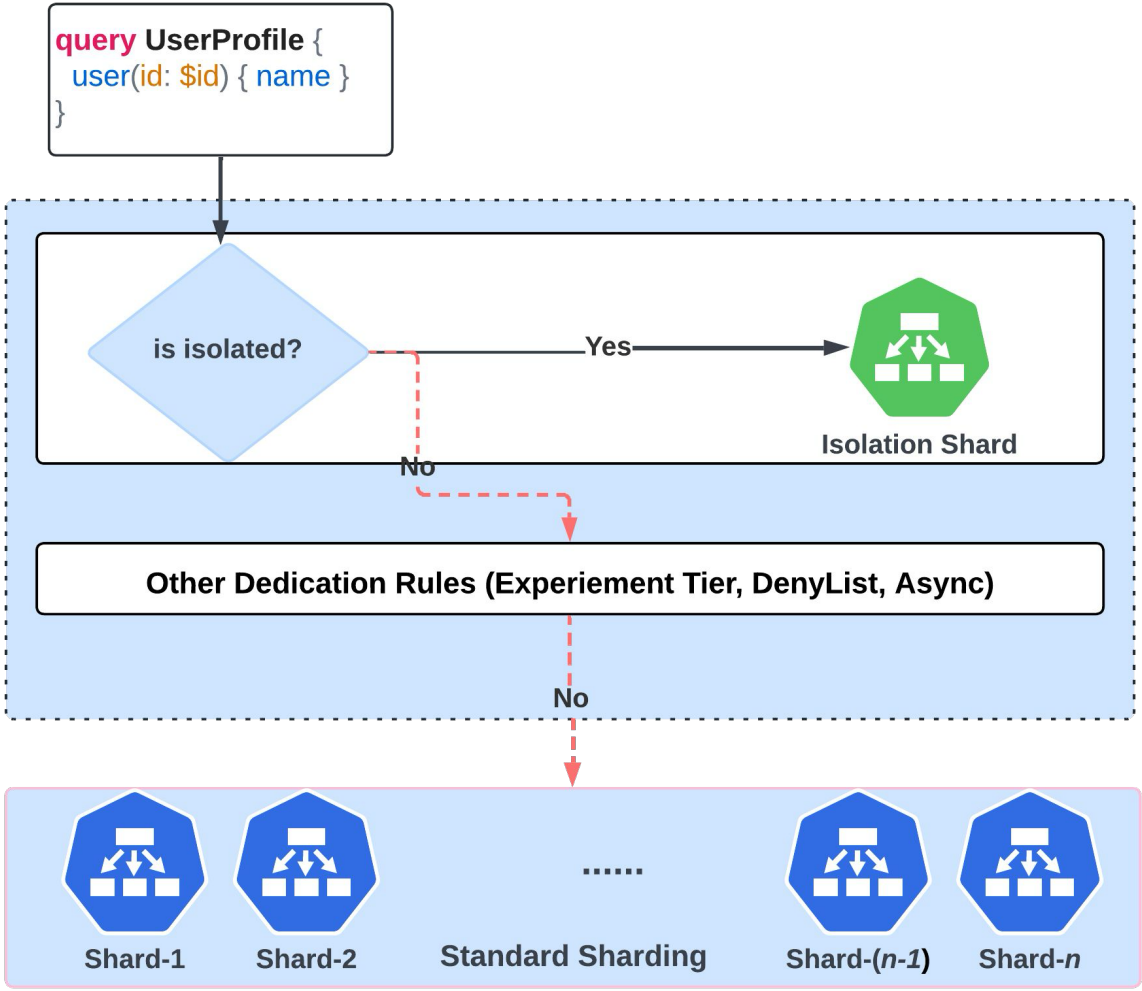
DenyList
Kills a misbehaving operation which shouldn't run at all.

Isolation Shards
Contains noisy operations on a dedicated shard without killing them.

Async & Subscription Shards
Separates asynchronous / subscription traffic from online traffic.

Experimental Shards
Runs designated traffic patterns for releasing new features.

Routing with Dedicated Sharding



Results on Mitigation Speed

Before

~3 hours

↘

After

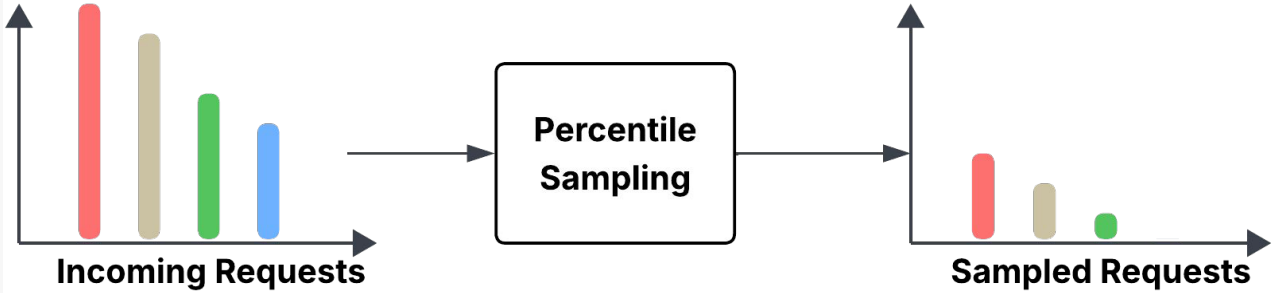
~10 mins

Powerful, flexible
and unlocks new opportunities

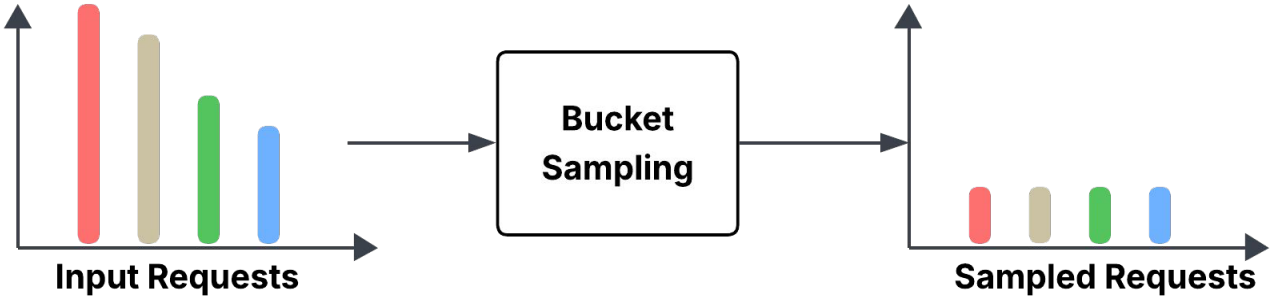
Sample Size Normalization

Catch low-QPS critical operations before they fan out;
Reduce the blast radius before it happens

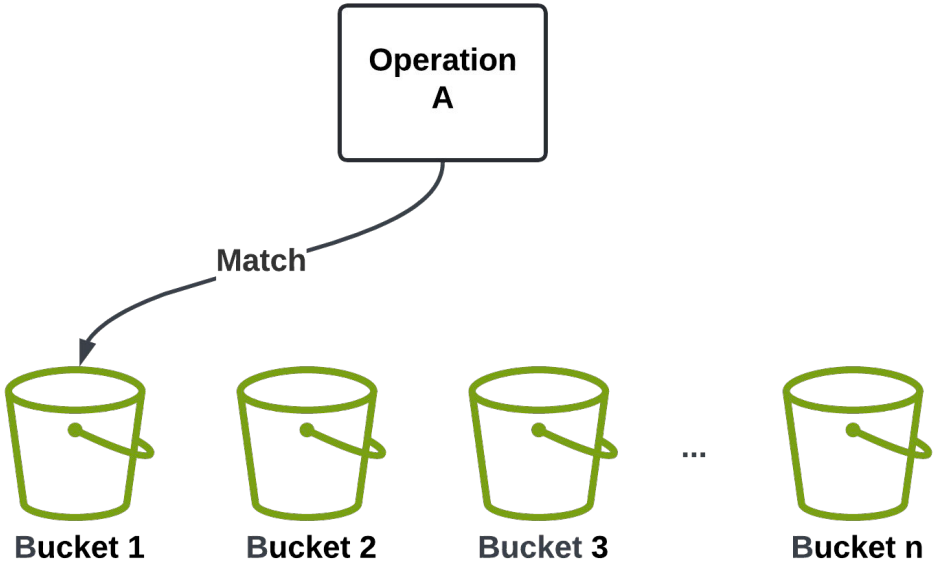
Old Sampling for Canary



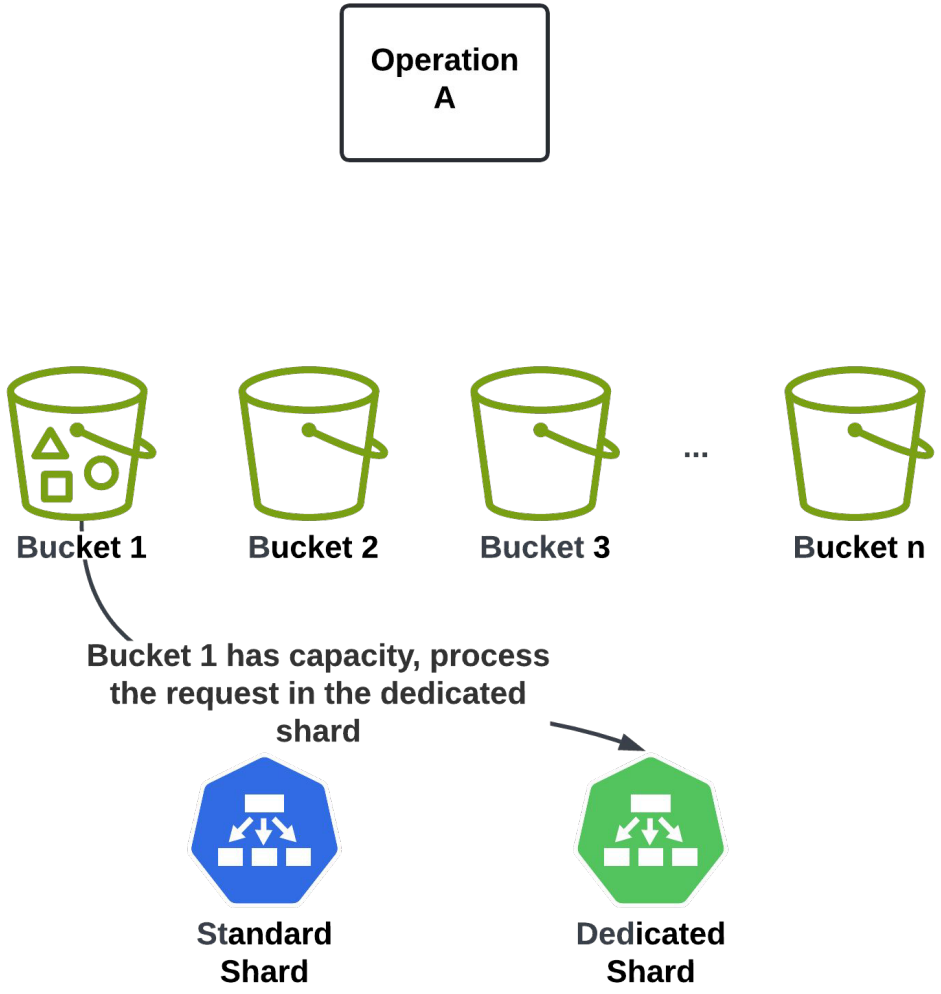
New Bucket Sampling



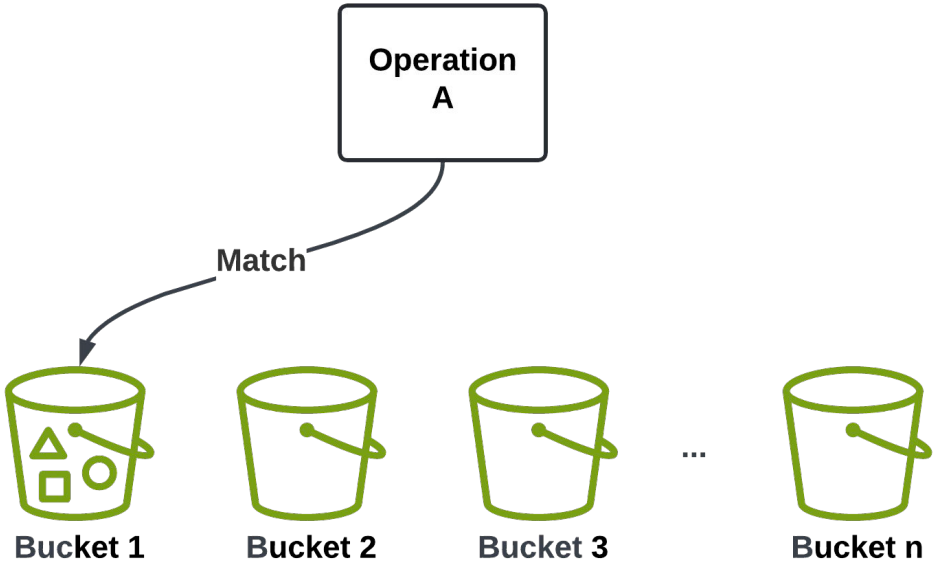
Bucket Sampling



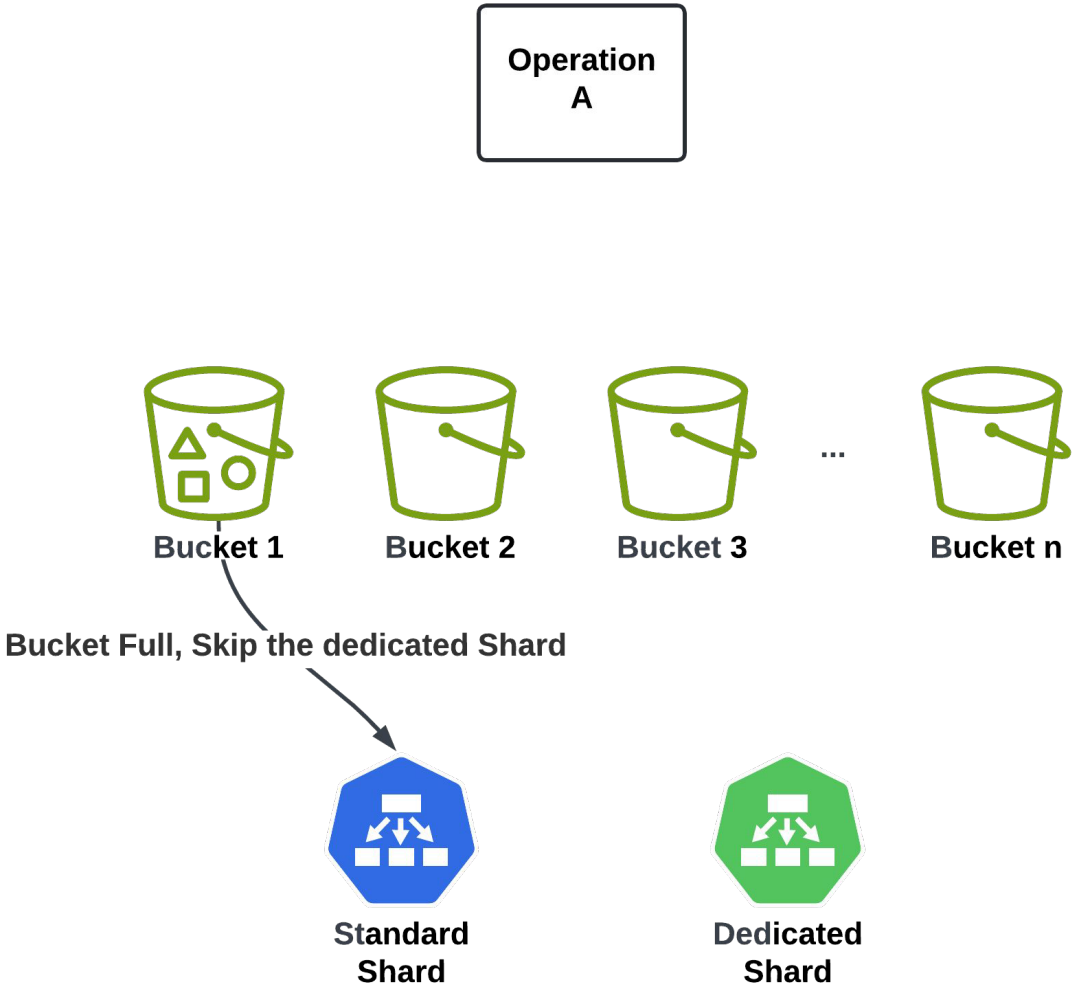
Bucket Sampling



Bucket Sampling



Bucket Sampling



~1000× signal amplification

Remaining Gaps

Traffic Isolation

Online and offline traffic mixed in every shard.

Mitigation Speed

Slow mitigation: debugging and fixing identified issues takes hours

Experimentation Tier

Lost the experimentation tier when shuffle replaced it.

Full Tenant Isolation

Preventing code execution issues from impacting neighboring tenants.

Evolve toward isolation without fragmenting the API surface.

Thanks!