# 2022.05.10 APOLLO DR 方案

## 目标

- 多个可用区多活架构，确保一个可用区发生故障时可以快速统一切换到另一个可用区，确保业务在新可用区正常获取配置

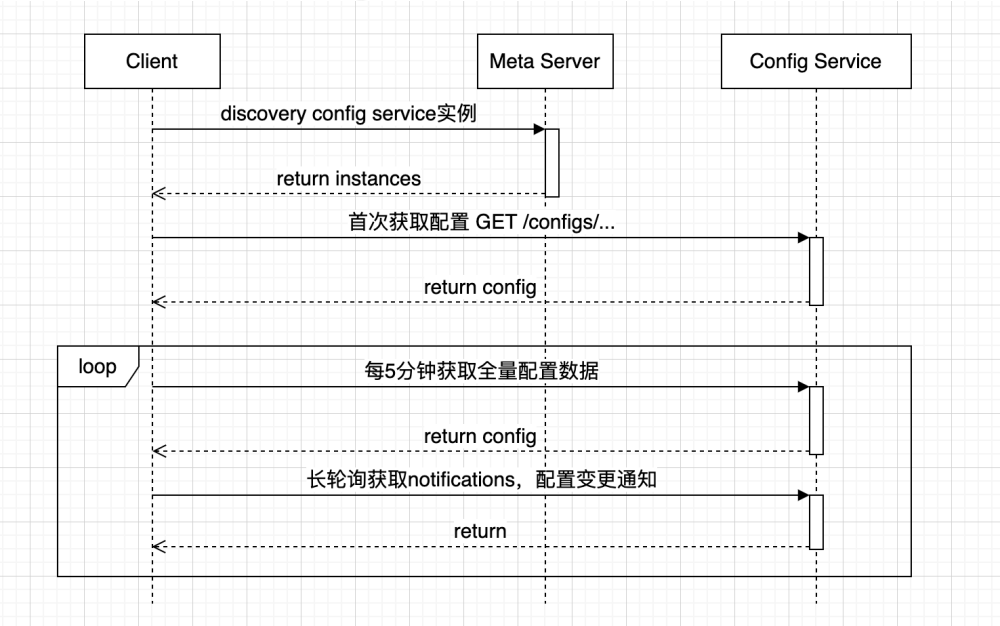
## 相关域名和服务

- https://config.ssc.shopeemobile.com 配置portal界面，主要是开发或QA在办公室使用
- https://mate.ssc.shopeemobile.com 用于发现Config服务（主要用于客户端获取订阅配置），和Admin服务（主要用于Portal写入配置）

## 涉及用例说明

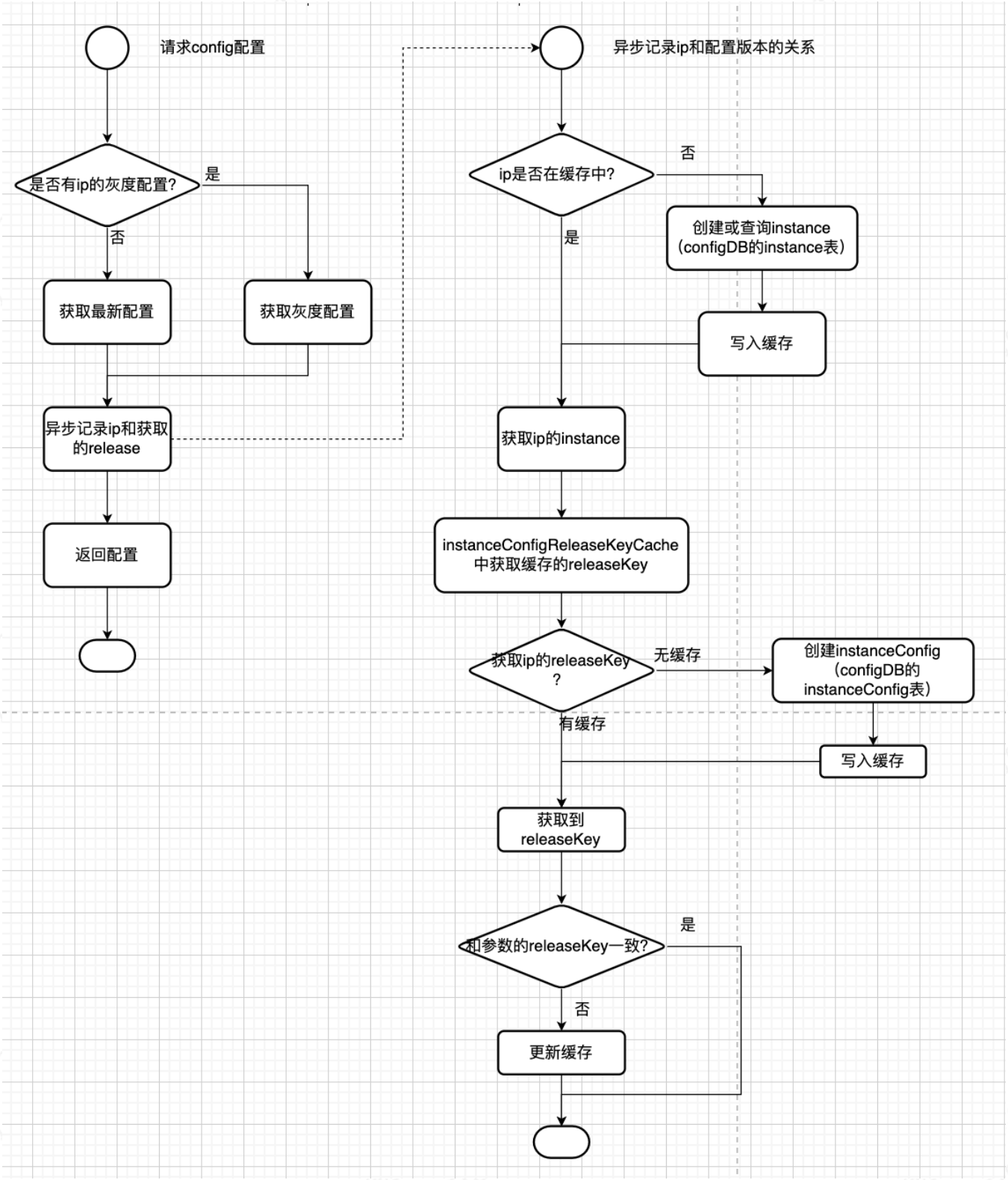
- 从portal发起的修改配置的请求会经过admin service 到达config db，这部分涉及操作配置appid，namespace等的读写操作，只能走主库
- client客户端读取配置主要涉及两个操作：**1.启动时获取配置（每5分钟） 2.长轮询获取变更信息**，以下对此做详细说明

### 客户端获取配置




#### 全量配置接口

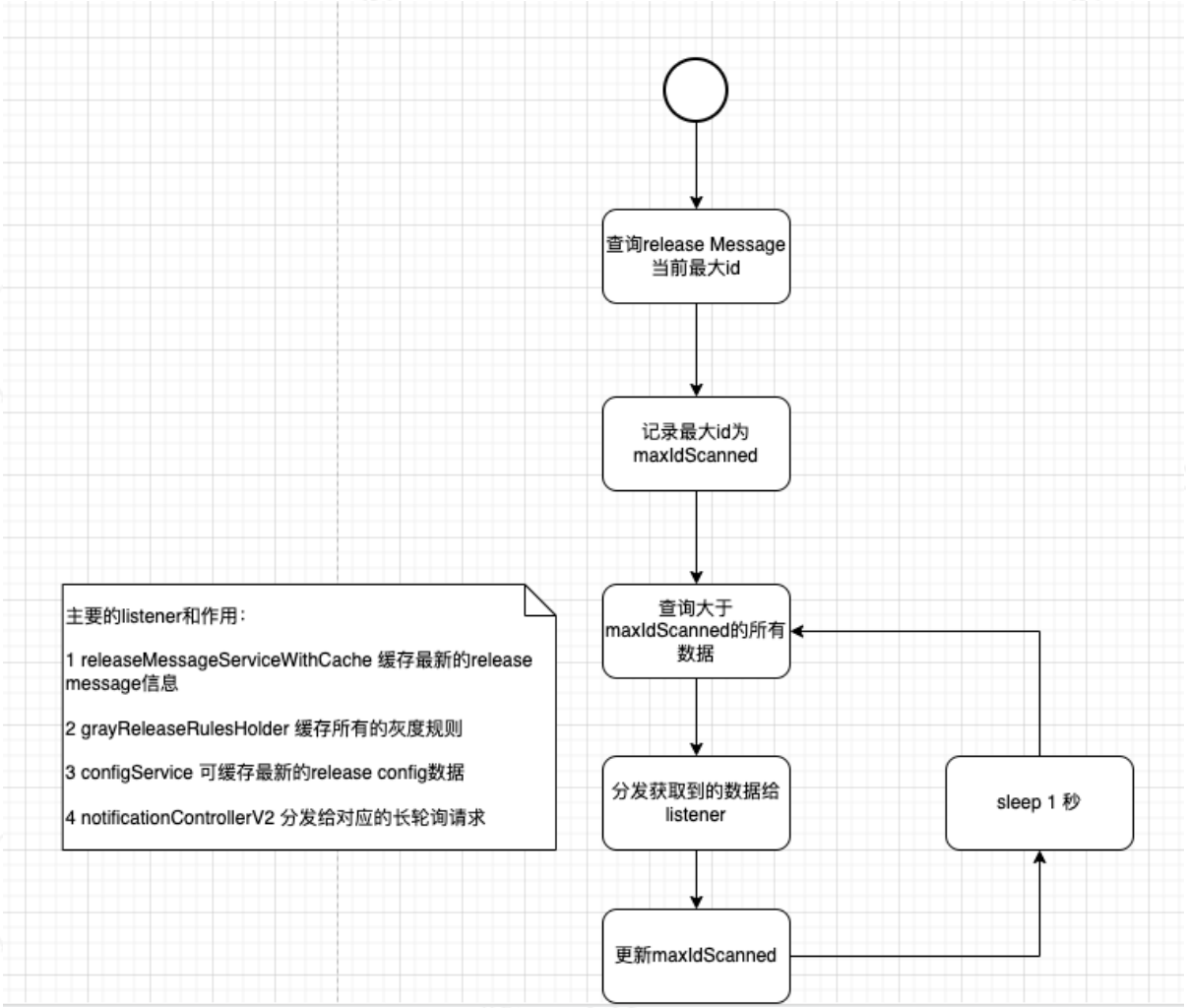
其中涉及写入ConfigDB的操作是全量获取配置的接口"GET configs/:appid/:cluster/:namespace?releaseKey=:releaseKey&ip=:ip"，写入为异步的操作，不影响




## 拉取notification接口

这里分为两部分

1. 轮询拉取新的release message，并分发给订阅的listener，这里的listener
2. 客户端长轮询获取最新的notification ( 超时50s)




## 问题记录

Q: 目前Eurake在apollo中怎么使用，部署在哪里？

A: eurake内嵌在config service里，和config service 同属于同一个进程，

通过打包时的 live_meta=https://mate.ssc.shopeemobile.com/ test_meta=https://mate-ssc.test.shopeemobile.com/ 等对应环境参数来使得config service 和 admin service使用它来注册

Q: config service中主要有哪些缓存，有什么作用？

A：ConfigServiceWithCache缓存最新的config release，key的格式为 :appid+:cluster+:namespace，**目前没有启用**，开关在serverConfig表中的配置项 "config-service.cache.enabled"；

ReleaseMessageServiceWithCache缓存最新的release message， key的格式为 :appid+:cluster+:namespace;

Q：ConfigServiceWithCache缓存是否有缓存穿透的风险，是否会过期？

A：ConfigServiceWithCache 目前默认没有启用，也就是当前服务查询全量配置都会走到config DB；

ConfigServiceWithCache有针对没有查询到的场景做了处理（存了nullConfigCacheEntry，没有发现会穿透的行为；

过期行为为：

1. 空闲1小时
2. 通过客户端请求时带上的message信息来比对版本，如果比客户端请求带过来的版本小，则过期掉

3. 查询新的配置产生

Q：config service是否和admin service存在交互？

A：之间没有交互

Q：config service 目前没有打开缓存的原因是什么？

A：假设存在config1 和 config2 两个服务端，

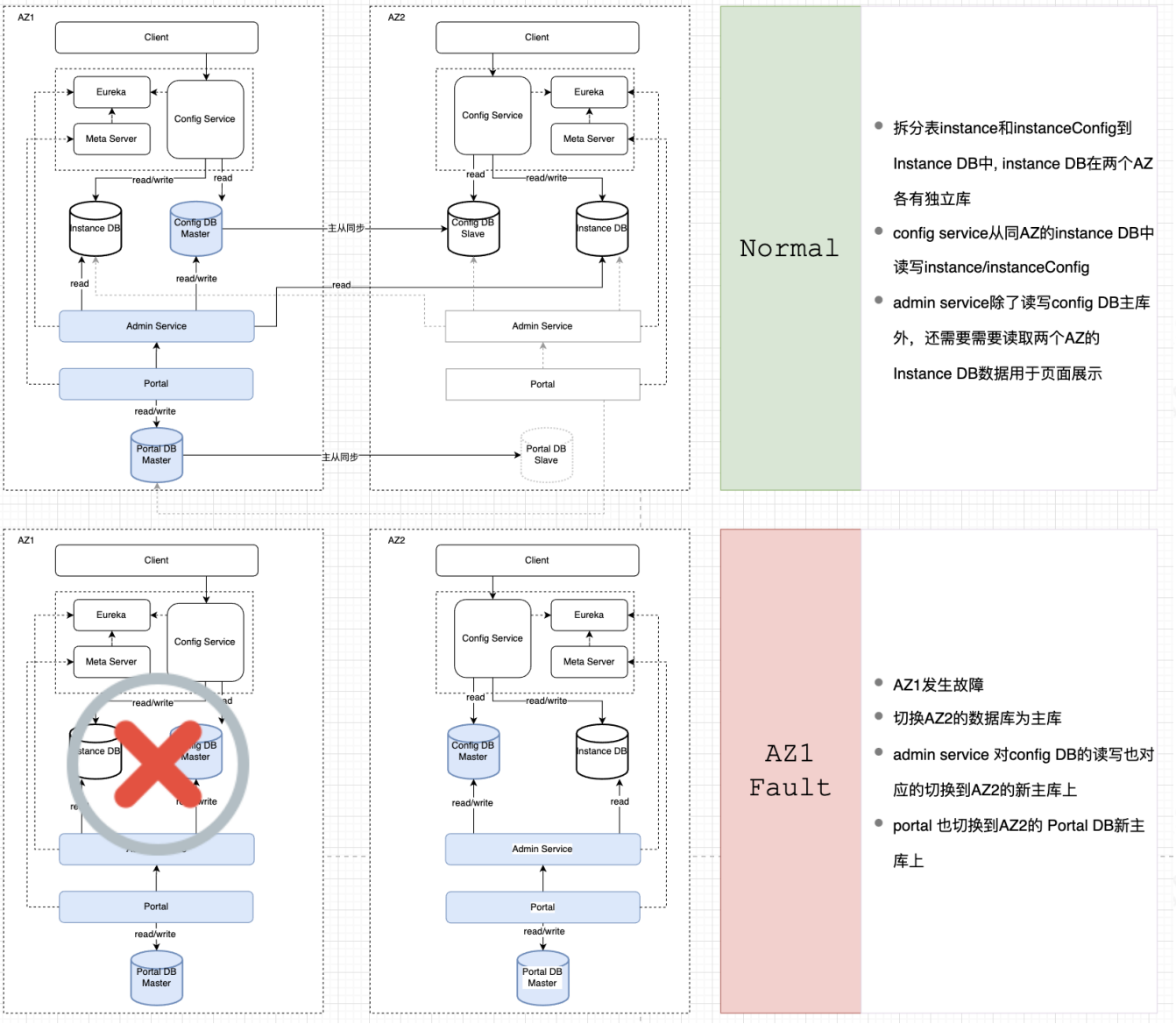
python的客户端可能存在，请求config1获取到最新的notification id，但没有带上notification id去请求，这是如果请求分配到了config2，config2如果还没来得及加载最新的配置到缓存，导致client读到的可能是旧配置；

而目前golang客户端请求configs接口时有带上客户端侧缓存的最新的notification id，这时服务端会有比对notification id的过程来确保是否需要重新从数据库加载数据，避免了python 客户端的问题；

由于python客户端的问题，目前不打开缓存配置，configs直接穿透到数据查询，以保证最新的配置；

# 方案设计

## 方案一（选定）




- 从Config DB数据库中拆出Instance DB（用于记录客户端实例ip，客户实例目前已经接收到的release配置版本），每个AZ有自己独立的Instance数据库
- AZ1和AZ2都可用的情况下
  - Portal和Admin Service相关的数据库读写均使用AZ1的主库
  - Config Service的读操作（配置读取和订阅）走同AZ的数据库，写操作（记录订阅的客户端实例）写入到同AZ的instance DB中
- 当AZ1发生故障时，AZ2的从库变更为主库，请求也自然会打到AZ2的数据库上，而Instance DB无需更改

## 优缺点

- 优点
  - AZ2的客户端的所有操作都不需要跨AZ
- 缺点
  - 需要修改涉及拆出的InstanceDB的相关表（Instance，InstanceConfig）的代码逻辑

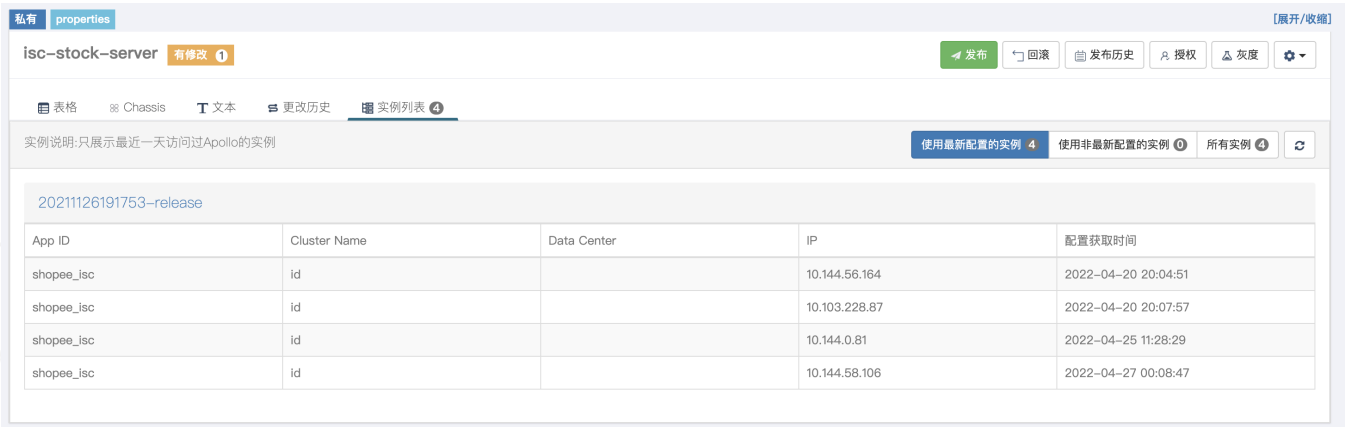
## 引入Instance DB修改说明

由于client获取配置时会写入两个表，除此之外其他操作均为读操作，所以可以考虑将两个表拆出成单独的库

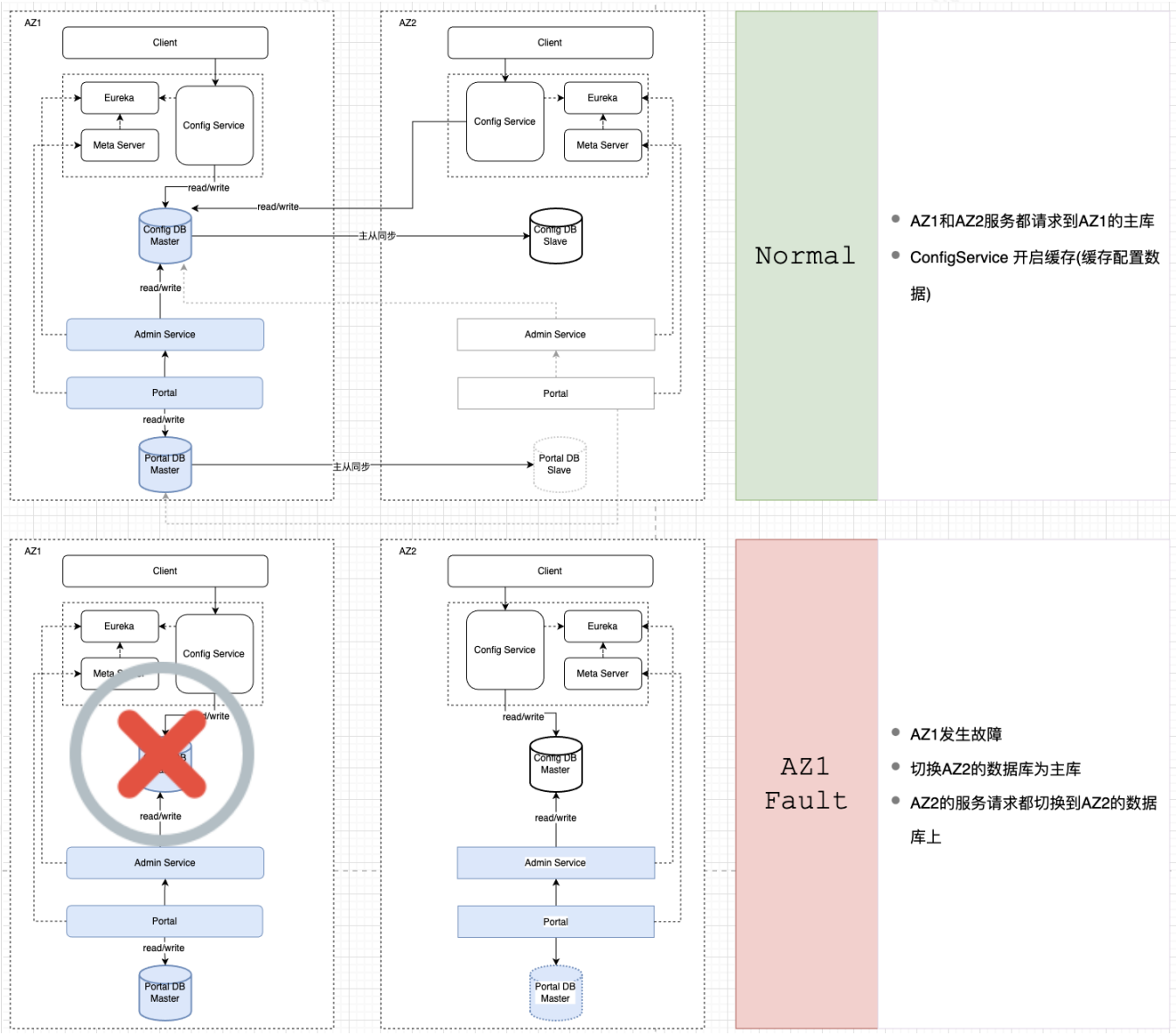
- Instance
- InstanceConfig

而拆出成单独的库，除了影响这里的写入操作外，还会影响portal界面上instance列表的显示，以及灰度规则的配置，

因此admin service读取instance信息提供给portal展示时需要查询所有instance DB的数据并做聚合




## 方案二




- 正常情况下，AZ1和AZ2都使用在AZ1中的主库进行读写
- AZ1发生故障时，AZ2的从库变更为主库，请求也将切换到AZ2的数据库上

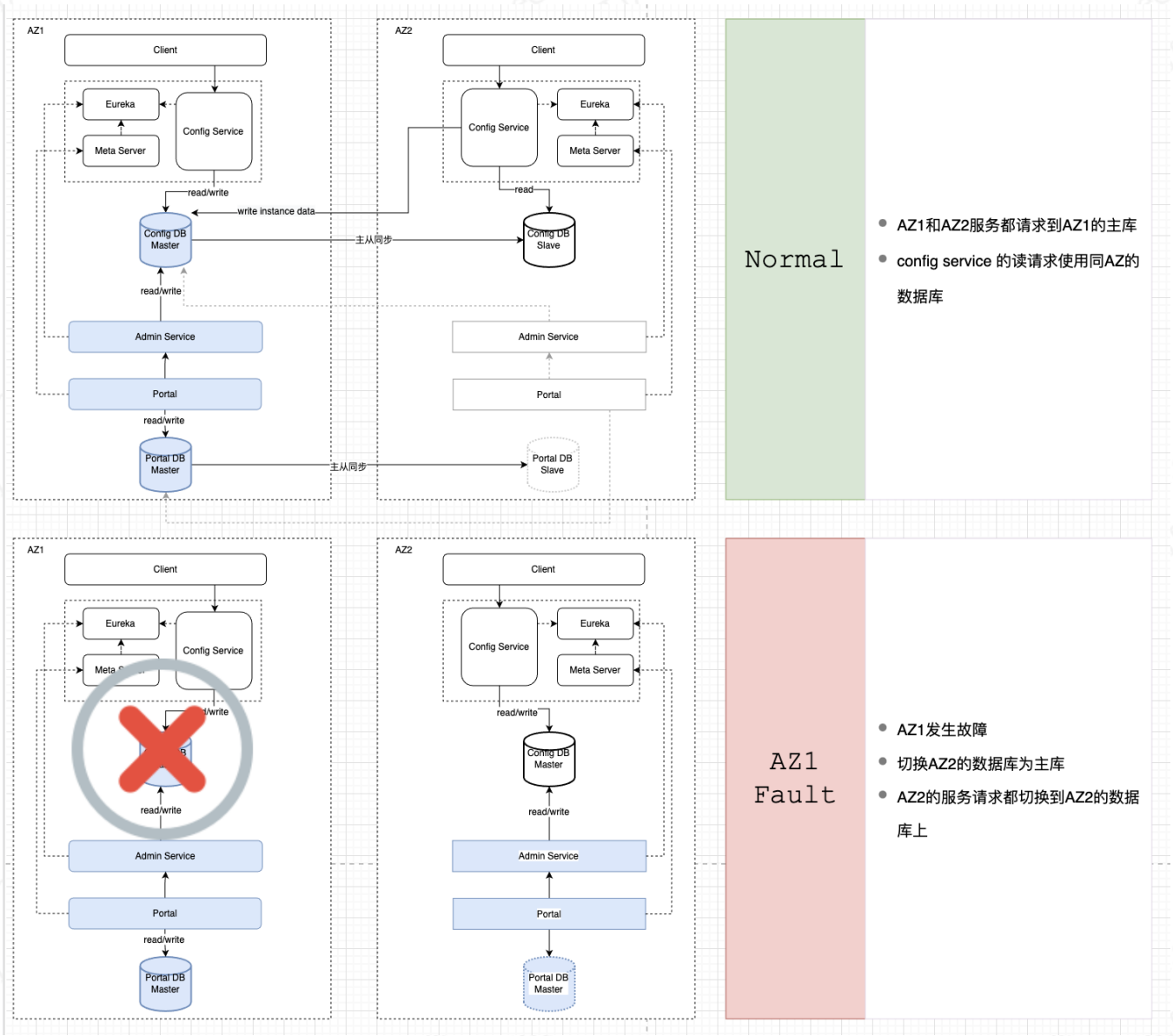
### 切换步骤

- 修改portal db和config db 的主库，并改变master域名指向AZ2的数据库

### 优缺点

- 优点
  - 架构简单无需对服务代码做处理
- 缺点
  - AZ2的客户端访问会受到AZ2到AZ1的访问延迟影响，以目前sg到mx的情况为例，大概是300ms
  - 数据库延迟问题可考虑用缓存来解决，但由于存在python客户端的问题（why cache disabled），不能直接打开

**改进AZ2的config service的读操作，可以将config service读写分离成，写主库，读从库（在相同AZ）**，由于instance和instanceConfig只在客户端发起的“Get configs/...” 请求中异步创建修改，写的延迟在毫秒级可以接受




- 这样变化后仍需要修改代码使得config service读取和写入config DB分离，其中需要配置默认AZ1（目前是sg）读写都是主库，而其他AZ读同AZ从库，写主库
- 当发生AZ1的崩溃时，需要切换config DB AZ2的从库为主库，并将AZ2的读写都指向新的主库（即AZ2的config DB）

## DR总结

- 方案一，由于instance的写入和读取发生了改变，需要进行的改动和验证点比较多，暂时不考虑；

- 方案二，config service 要缓存跨AZ的数据库config DB主库的读取到的数据(打开"config-service.cache.enabled", 要验证有无缓存穿透的问题);
- 由于前文提到的**python客户端的问题**（ why cache disabled ）对于方案二实行有两个方向可选
    - 选项1: 修改掉python客户端，确保configs请求都带上客户端当前获取到的最大notification id，并告知业务及时更新客户端
    - 选项2: 采用方案二中的改进方案，将AZ2中的config service 改为读写分离，写操作跨AZ到主库，读操作读同AZ的从库
- 基于目前的情况，**倾向于使用方案二改进后的读写分离方案**

# 具体功能修改

## config service 读写分离

参考实现 https://calm_java.gitee.io/blog/2020/02/13/labs-springboot2-dynamic-datasource/

## 使用datacenter字段标识AZ

```
CREATE TABLE `Instance` (
  `Id` int(11) unsigned NOT NULL AUTO_INCREMENT COMMENT 'Id',
  `AppId` varchar(32) NOT NULL DEFAULT 'default' COMMENT 'AppID',
  `ClusterName` varchar(32) NOT NULL DEFAULT 'default' COMMENT 'ClusterName',
  `DataCenter` varchar(64) NOT NULL DEFAULT 'default' COMMENT 'Data Center Name',
  `Ip` varchar(32) NOT NULL DEFAULT '' COMMENT 'instance ip',
  `DataChange_CreatedTime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '',
  `DataChange_LastTime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '',
  PRIMARY KEY (`Id`),
  UNIQUE KEY `IX_UNIQUE_KEY` (`AppId`,`ClusterName`,`Ip`,`DataCenter`),
  KEY `IX_IP` (`Ip`),
  KEY `IX_DataChange_LastTime` (`DataChange_LastTime`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COMMENT=''
```

可以看到instance 表中存在`DataCenter`字段，目前逻辑是通过"Get configs/..." 请求时带上信息来插入到instance表中，但这样的改动会涉及客户端。

基于目前的情况，由于client和其请求的config service一定在同一个AZ中，**所以服务端也可以默认设置一个AZ信息，作为信息填入到instance表中的这个字段中**。

# 相关

- 图示链接 https://drive.google.com/file/d/1k0GkjTCcUU5VReOq6rOYWytnmX4qV95i/view?usp=sharing