Pivotal

# *Spring Cloud Netflix Eureka*

李刚



Pivotal | IT大咖说

# 目录

- Eureka概述
- 数据结构
- Register机制
- Renew机制
- Cancel机制
- Evict机制
- Eureka Server缓存机制
- Eureka Server节点复制机制
- Eureka Client获取注册信息
- Q&A

***Eureka***，古希腊词语。含义为我找到了！ 我发现了！
相传阿基米德发现浮力原理时说出了这个词。

***Spring Cloud***架构中充当着注册中心的角色

***GitHub***地址：
https://github.com/Netflix/eureka

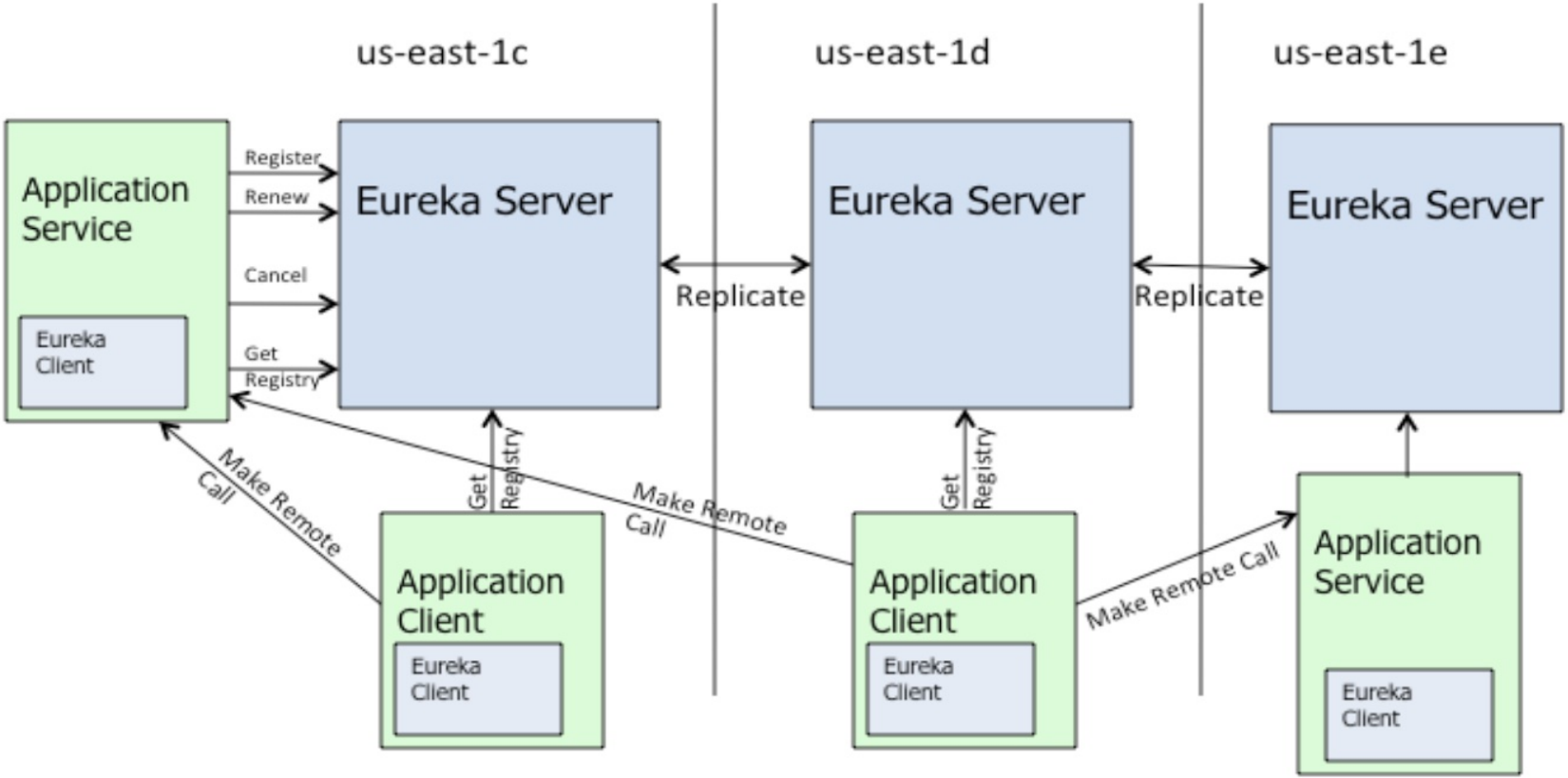
***1.9.3 Release*** 2018年6月26日

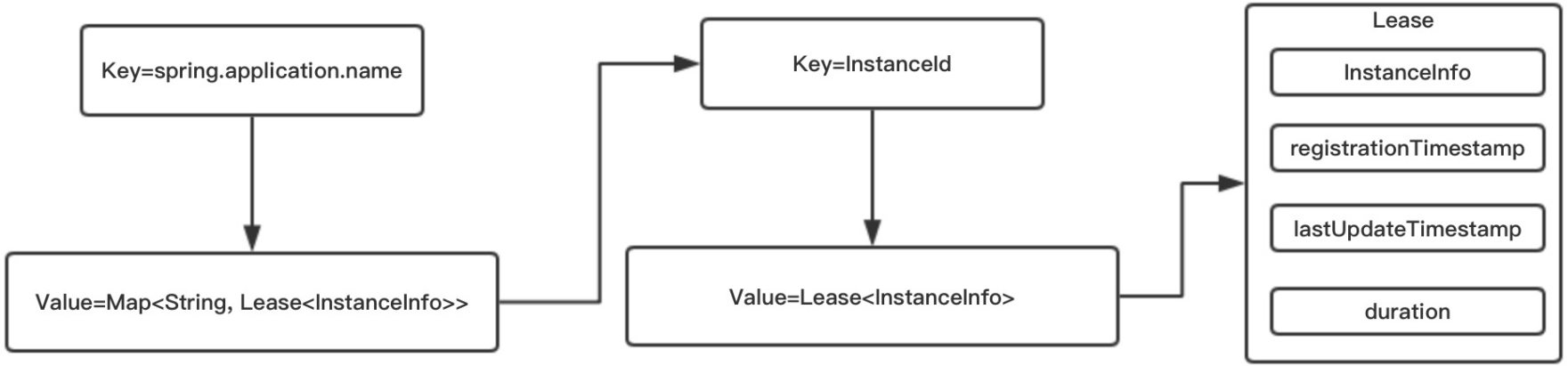
***1.9.2 Release*** 2018年6月2日

***1.9.1 Release*** 2018年6月1日

***1.9.0 Release*** 2018年4月26日

***1.8.8 Release*** 2018年4月10日



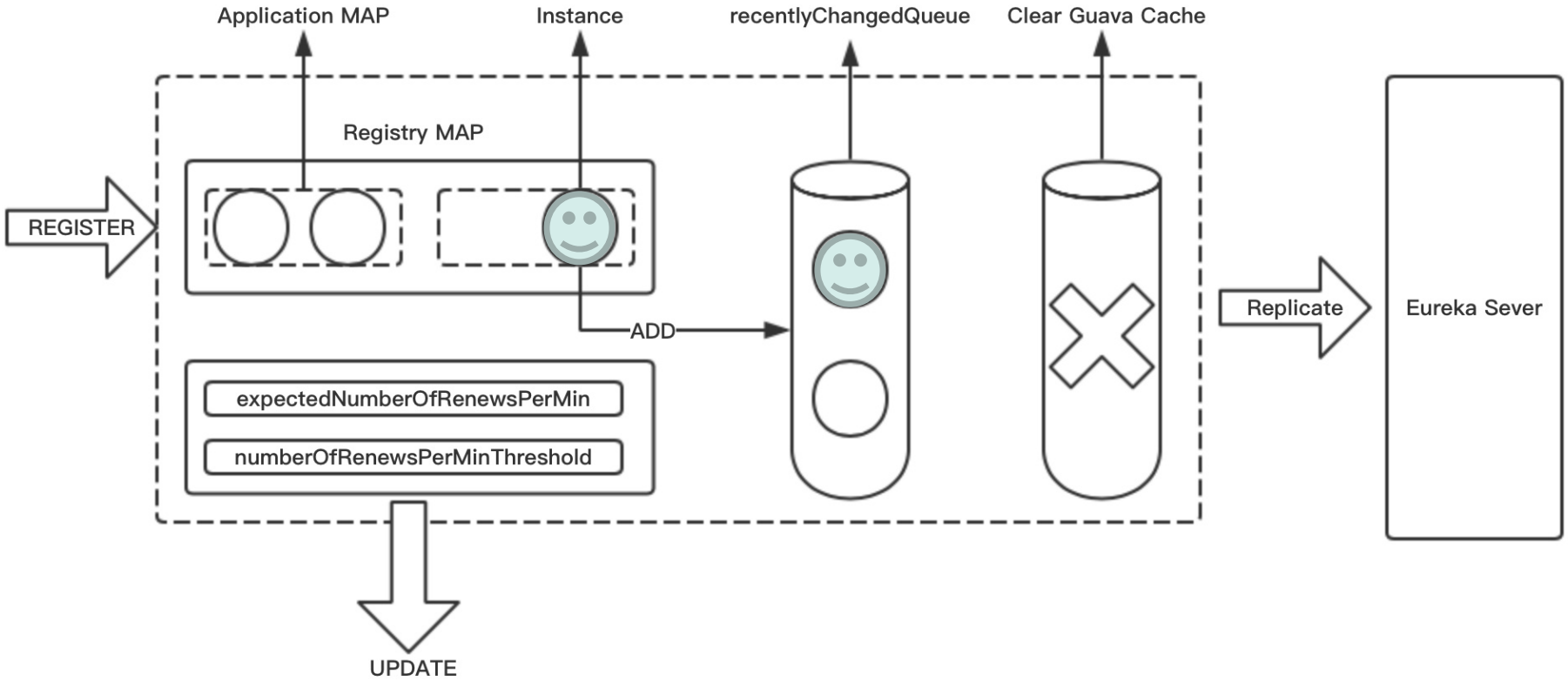



***Eureka Server***作为注册中心，保存注册信息的数据结构是双层***MAP***.

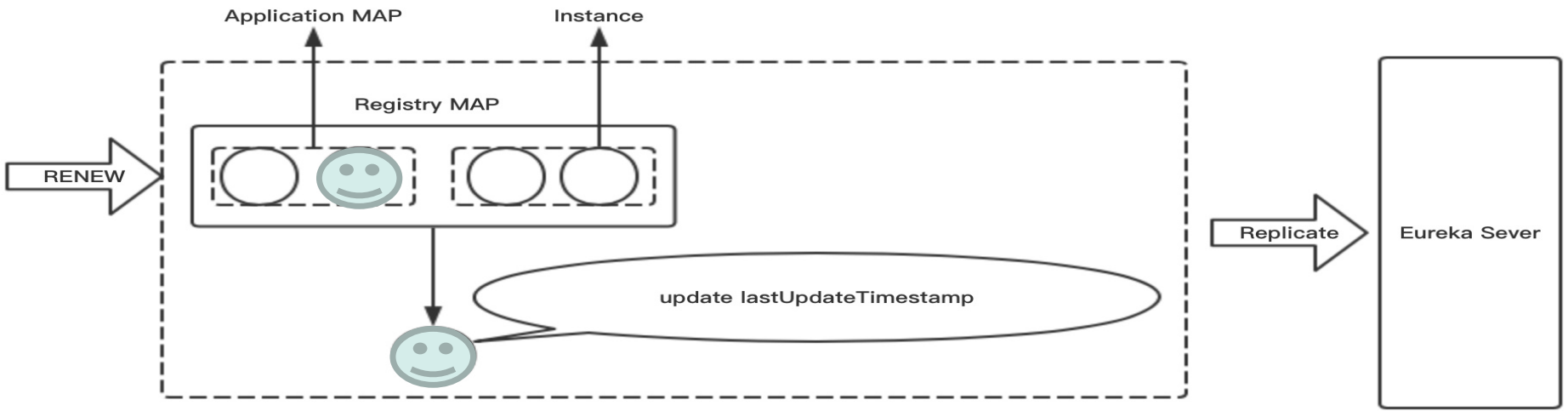
- 第一层***ConcurrentHashMap***：
  ***key***值为***spring.application.name***
  ***value***值为***MAP***

- 第二层***ConcurrentHashMap***:
  ***key***值为***InstanceId***
  ***value***值为***Lease***

```
public Lease(T r, int durationInSecs) {
    holder = r;
    registrationTimestamp = System.currentTimeMillis();
    lastUpdateTimestamp = registrationTimestamp;
    duration = (durationInSecs * 1000);

}
```



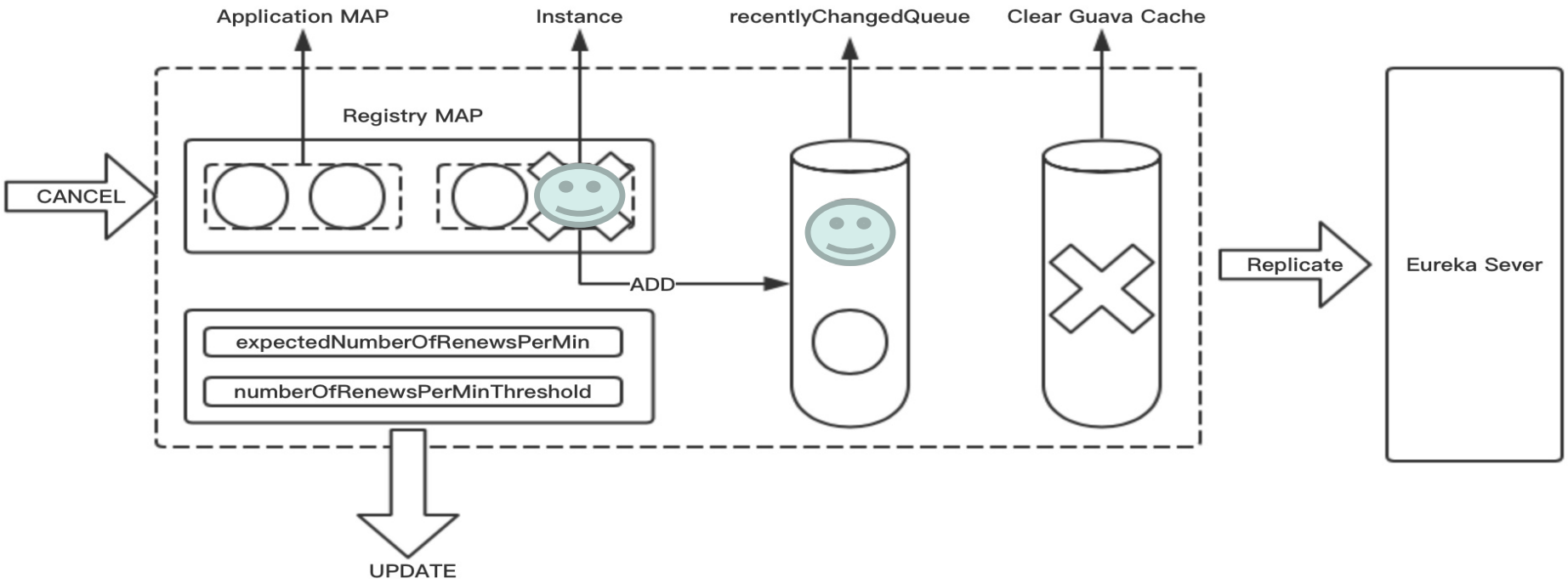

- ❑ 保存注册信息
- ❑ 更新阈值
- ❑ 将新增的实例保存到***Queue***中
- ❑ 清空缓存
- ❑ 复制给其他***Eureka Server***节点




- 找到续约实例
- 更新续约时间
- 复制给其他***Eureka Server***节点

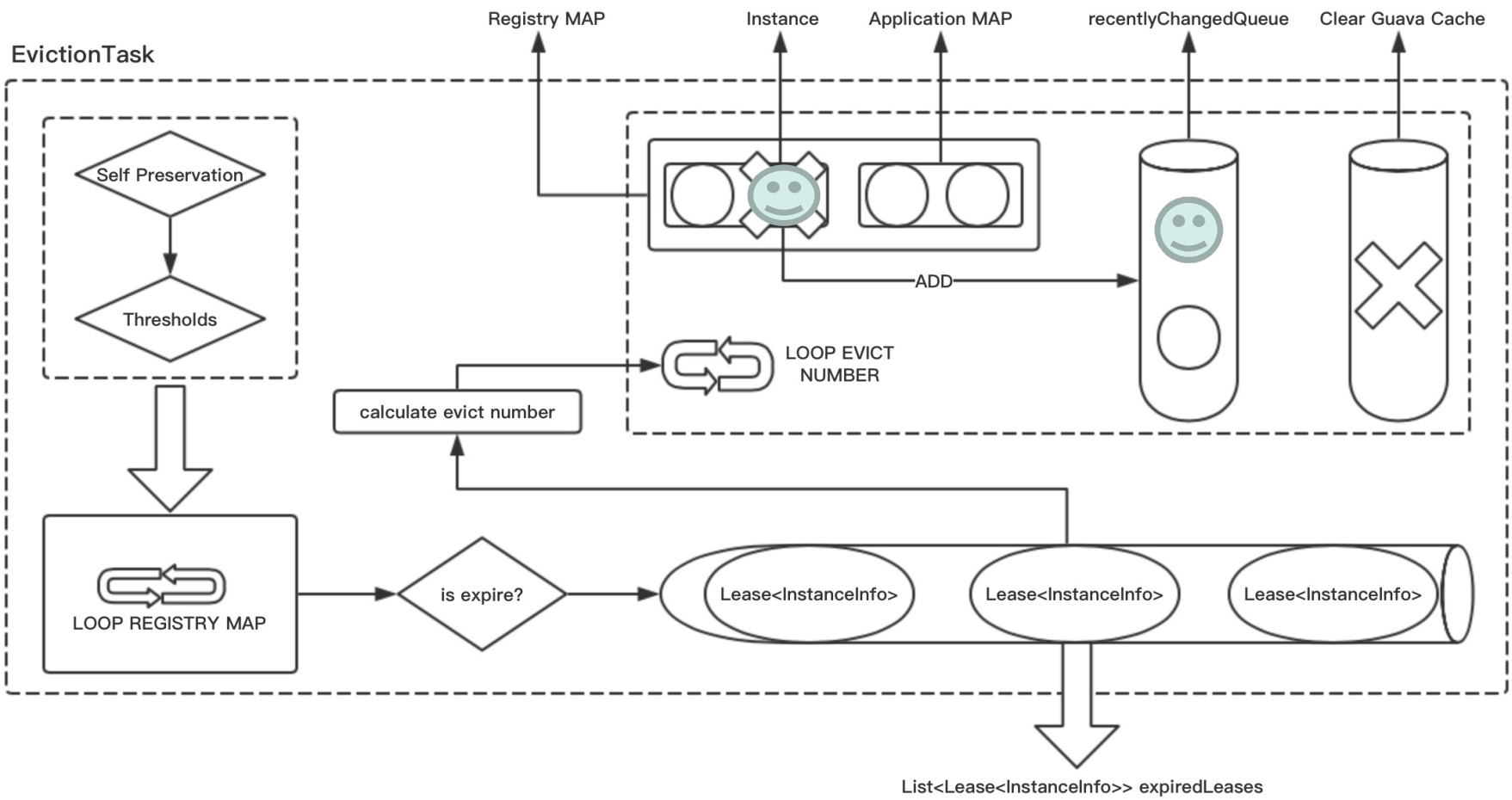
```
Map<String, Lease<InstanceInfo>> gMap = registry.get(appName);
Lease<InstanceInfo> leaseToRenew = gMap.get(id);
renewsLastMin.increment();
leaseToRenew.renew();
```

```
public void renew() {
    lastUpdateTimestamp = System.currentTimeMillis() + duration;

}
```




- 删除具体实例
- 将删除的实例保存到***Queue***中
- 清空缓存
- 复制给其他***Eureka Server***节点
- 更新属性

```
try {
    read.lock();
    Map<String, Lease<InstanceInfo>> gMap = registry.get(appName);
    Lease<InstanceInfo> leaseToCancel = gMap.remove(id);
    leaseToCancel.cancel();
    InstanceInfo instanceInfo = leaseToCancel.getHolder();
    if (instanceInfo != null) {
        instanceInfo.setActionType(ActionType.DELETED);
        recentlyChangedQueue.add(new RecentlyChangedItem(leaseToCancel));
    }

} finally {
    read.unlock();
}
```


Registry MAP
Instance
Application MAP
recentlyChangedQueue
Clear Guava Cache
EvictionTask
Self Preservation
Thresholds
ADD
LOOP EVICT NUMBER
calculate evict number
LOOP REGISTRY MAP
is expire?
Lease<InstanceInfo>
Lease<InstanceInfo>
Lease<InstanceInfo>
List<Lease<InstanceInfo>> expiredLeases

# Eureka Server缓存机制

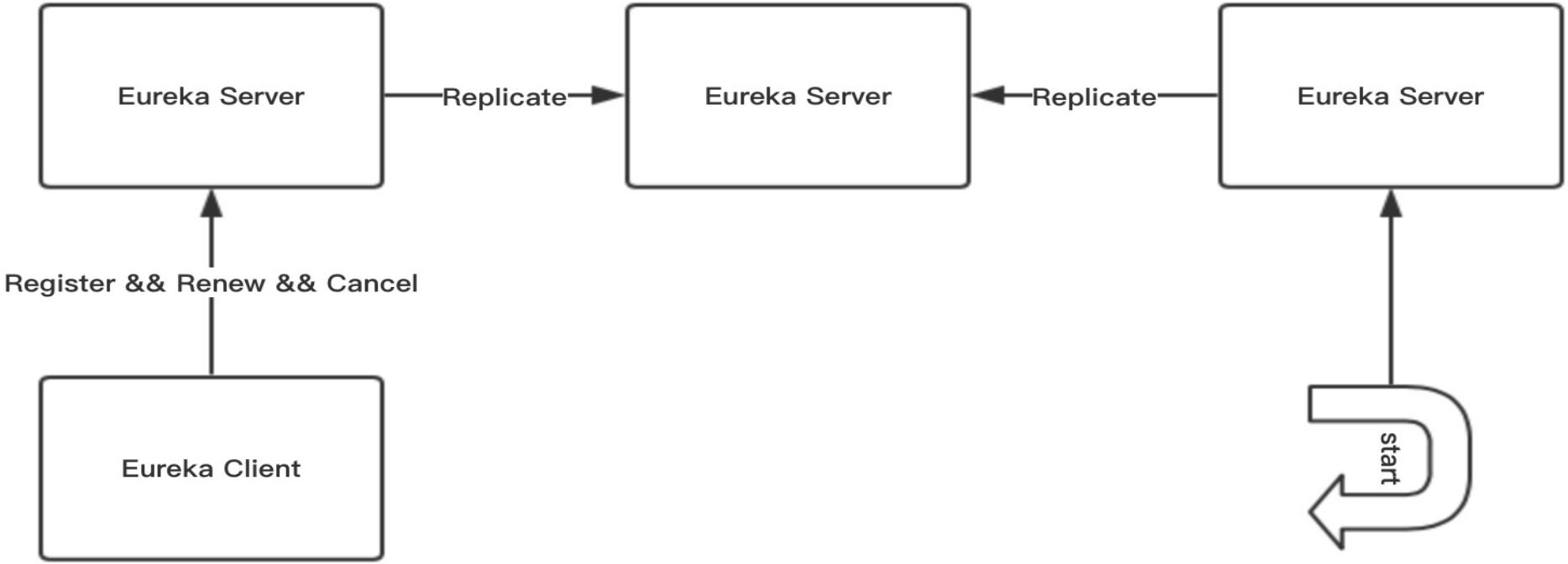
## *com.netflix.eureka.registry.ResponseCacheImpl*


***Eureka Server***内置两层缓存

- ***readOnlyCacheMap***本质***MAP***,无过期时间
- ***readWriteCacheMap***本质***Guava***缓存，存在过期时间
- 通过参数可决定是否启用***MAP***缓存

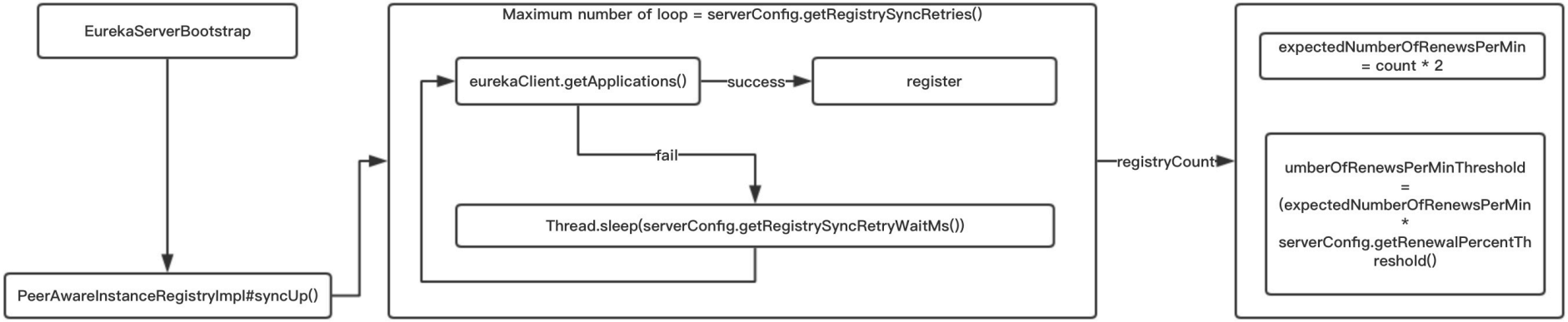
```
if (shouldUseReadOnlyResponseCache) {
    timer.schedule(getCacheUpdateTask(),
            new Date(((System.currentTimeMillis() / responseCacheUpdateIntervalMs) * responseCacheUpdateIntervalMs)
                    + responseCacheUpdateIntervalMs),
            responseCacheUpdateIntervalMs);
}
```

# Eureka Server节点复制机制



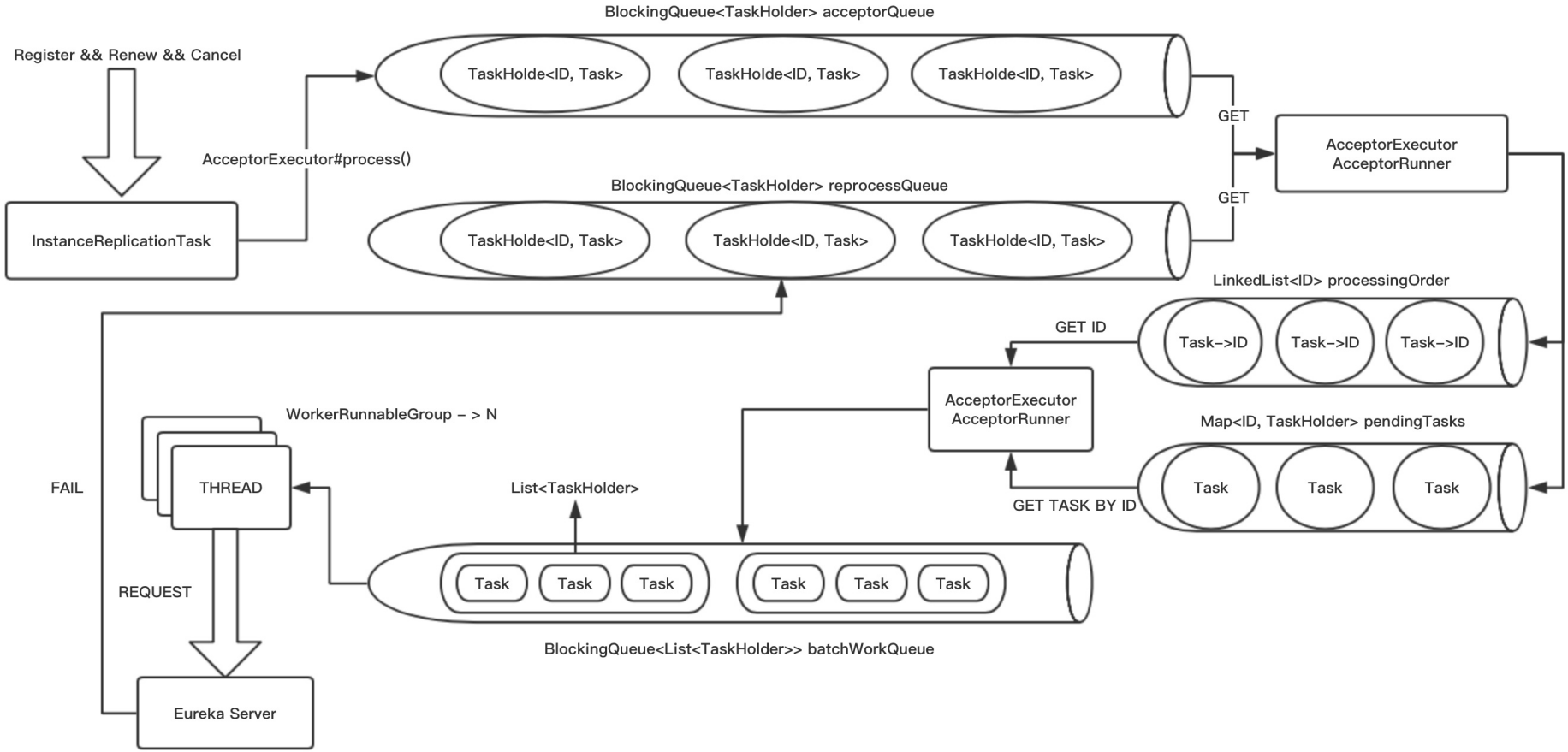

- ***Eureka Server***启动时从相邻节点同步已有注册信息
- ***Eureka Server***接收***register***、***renew***、***cancel***事件时会将此信息同步给相邻节点

# *Eureka Server*启动时从相邻节点同步已有注册信息




- 循环次数
  ***serverConfig.getRegistrySynRetries()***
- 循环间隔
  ***serverConfig.ferRegistrySyncRetryWaitMs()***
- 更新阈值
  ***expectedNumberOfRenewsPerMin = registerConut * 2***
  ***numberOfRenewsPerMinThreshold = expectedNumberOfRenewsPerMin * config***
  ***config = config.getRenewalPercentThreshold()***

# 向相邻*Eureka Server*节点同步*register*、*renew*、*cancel*事件



# Eureka Client获取注册信息


```
@Override
public boolean shouldDisableDelta() {
    return configInstance.getBooleanProperty(
            propName: namespace + "disableDelta",
            defaultValue: false)
            .get();
}
```

全量

- 首次
- 增量同步失败，获取全量
- 开启增量与全量差异的日志信息

```
@Override
public boolean shouldLogDeltaDiff() {
    return configInstance.getBooleanProperty(
            propName: namespace + SHOULD_LOG_DELTA_DIFF_KEY,
            defaultValue: false)
            .get();
}
```

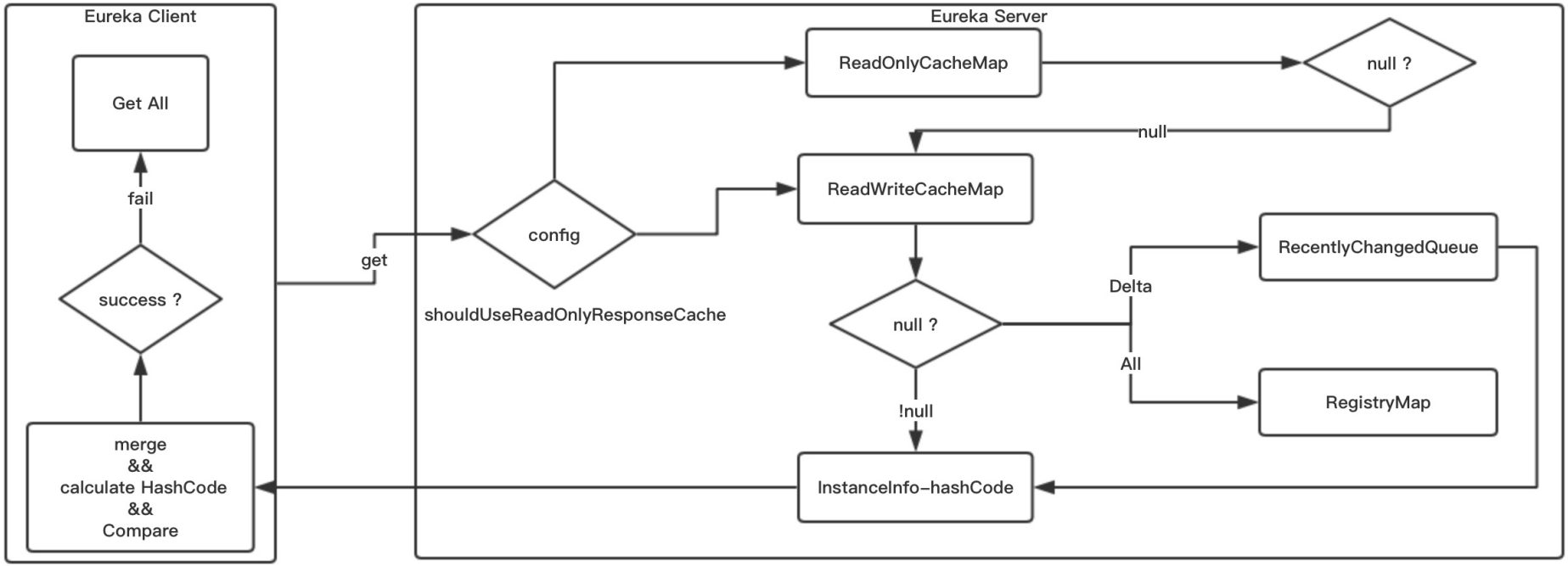
增量

- 通过参数限制，是否开启增量模式

缓存

- 缓存获取数据的逻辑

```
Value payload = null;
try {
    if (useReadOnlyCache) {
        final Value currentPayload = readOnlyCacheMap.get(key);
        if (currentPayload != null) {
            payload = currentPayload;
        } else {
            payload = readWriteCacheMap.get(key);
            readOnlyCacheMap.put(key, payload);
        }
    } else {
        payload = readWriteCacheMap.get(key);
    }
} catch (Throwable t) {
    logger.error("Cannot get value for key : {}", key, t);
}
return payload;
```


Eureka Client
Get All
fail
success ?
merge
&&
calculate HashCode
&&
Compare
get
Eureka Server
config
shouldUseReadOnlyResponseCache
ReadOnlyCacheMap
null ?
null
ReadWriteCacheMap
null ?
Delta
All
!null
RecentlyChangedQueue
RegistryMap
InstanceInfo-hashCode

# THANK YOU!

# Q&A


# Transforming How The World Builds Software