注册中心优化

优化点

1. 服务下线机制。如果服务下线需要即时通知，以防消费者调用下线服务
2. 性能优化。消费者每次查询注册中心改成查缓存
3. 心跳检测
4. 多种方式实现注册中心

心跳检测，续期

1. 给服务节点的key设置过期时间，然后再给服务节点设置定时任务续期。如果没有定时续期，key过期了，说明服务节点没有了；
2. 维护一个local注册中心，将本节点注册的服务信息在local里放一份。定时任务循环续期这个local中心里的就可以；

服务节点主动下线，被动下线

被动下线

心跳检测中已经实现

主动下线

在服务节点停止时主动下线

使用JVM提供的shutdownHook机制。允许在jvm停止前执行清理或其他必要的工作。

Java

```java
public void destroy() {
    System.out.println("当前节点下线");
    // 下线节点
    // 遍历本节点所有的 key
    for (String key : localRegisterNodeKeySet) {
        try {
            kvClient.delete(ByteSequence.from(key, StandardCharsets.UTF_8)).get();
        } catch (Exception e) {
            throw new RuntimeException(key + "节点下线失败");
        }
    }

    // 释放资源
    if (kvClient != null) {
        kvClient.close();
    }
    if (client != null) {
        client.close();
    }
}

```

```java
public static void init(RpcConfig newRpcConfig) {
    rpcConfig = newRpcConfig;
    log.info("rpc init, config = {}", newRpcConfig.toString());
    // 注册中心初始化
    RegistryConfig registryConfig = rpcConfig.getRegistryConfig();
    Registry registry = RegistryFactory.getInstance(registryConfig.getRegistry());
    registry.init(registryConfig);
    log.info("registry init, config = {}", registryConfig);

    // 创建并注册 Shutdown Hook，JVM 退出时执行操作
    Runtime.getRuntime().addShutdownHook(new Thread(registry::destroy));
}

```

消费端服务缓存

1. 添加一个缓存操作类
2. 获取服务列表时先判断缓存中有没有，如果没有再请求注册中心获取，然后添加到缓存。

监听机制

1. 使用etcd的监听方法
2. 在消费端服务发现里实现
3. 新建一个set，将需要监听的放到set里
4. 当监听到delete事件后，清除本地缓存