**任务调度系统设计**

# 1 前言

## 1.1名词解释

* confnamesrv提供HTTP服务 , 服务返回当前可用的配置中心/注册中心的地址
* registryclient提供注册中心sdk供各个业务方调用
* registryserver提供TCP调用注册服务
* registryworker 注册中心任务服务
* platformweb提供web服务,查看配置信息,修改配置信息，查看注册中心服务

# 2 注册中心

## 2.1整体架构

## 2.2注册中心表设计

### 2.2.1生产者实例表

表名：t\_registry\_instance

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Comment** |
| **Id** | **bigint(20) NOT NULL** | **自增Id主键** |
| **instance \_uniquekey** | **varchar(128) NOT NULL** | **唯一主键（ip;port;alias;group）** |
| **ip** | **varchar(16) NOT NULL** |  |
| **port** | **SMALLINT(5) NOT NULL** | **8080:httprpc , 11999:tcprpc , 12999：taskrpc** |
| **app\_name** | **VARCHAR(16)** | **项目名称** |
| **pid** | **SMALLINT(5) NOT NULL** |  |
| **alias** | **VARCHAR(32) NOT NULL** | **项目名称+版本号** |
| **service\_group** | **VARCHAR(32)** | **1 : httprpc 2: tcprpc 3:taskrpc** |
| **context\_path** | **VARCHAR(128)** | **上下文** |
| **server\_timeout** | **INT(8)** | **超时时间** |
| **status** | **TINYINT(4)** | **状态 0：不可用 1：可用** |
| **start\_time** | **datetime NOT NULL** | **启动时间** |
| **create\_time** | **datetime NOT NULL** | **创建时间** |
| **update\_time** | **datetime NOT NULL** | **更新时间** |

## 2.3客户端设计

### 2.3.1存储模块

实例缓存存储本地缓存,文件系统也存储一份实例信息。

BerkeleyDb 单个实例多个数据库，如下：

|  |  |
| --- | --- |
| 数据库名 | 说明 |
| BASE | 注册中心基本信息 |
| INSTANCE | 应用实例信息 |
| CLIENT(CONSUMER) | 消费者信息 |
| SERVER(PROVIDER) | 服务者信息 |

### 2.3.2链接管理

1. 客户端启动
2. 从名字服务获取注册中心地址（hshc-platform-core里有相关方法）
3. 从注册中心地址中,尝试与所有的注册中心建立连接
4. 在连接中，需要标识是不是允许通知需要加
5. 启动异步线程, 定时将心跳包（10秒）发到所有注册中心
6. 定时从注册中心（已连接中的某一台随机）30s获取全量数据
7. 当注册中心server其中一台发生变化时,通知客户端发生变化
8. 当注册中心server通知client删除实例时, 修改实例的状态为不可用

### 2.3.3实例发布

通过心跳包发送到注册中心，注册包里有实例信息

### 2.3.4安全管理

配置文件：

hshcCommonPlatform.properties

hshc\_common.properties

hshc-platform-core核心包中解析配置文件

应用传输需要在网络层传输：

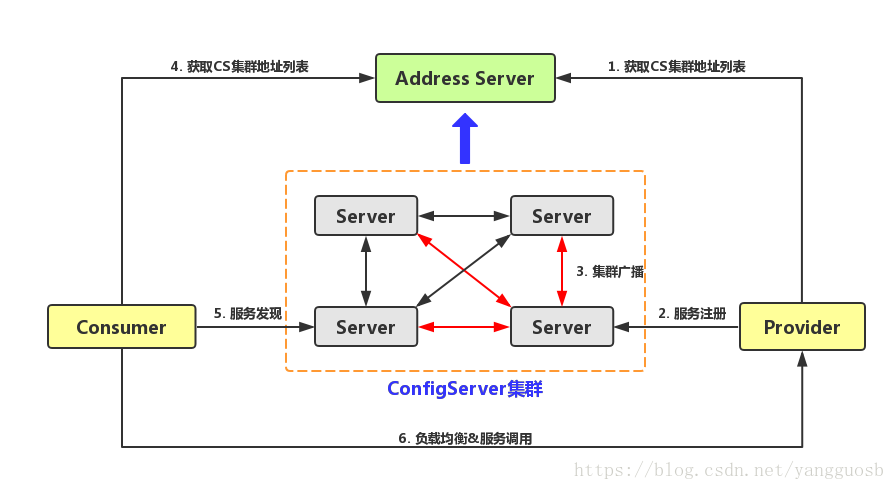
header头中需要传输三个字段：

|  |  |
| --- | --- |
| 字段名 | 说明 |
| appId | 项目编号 hshcCommonPlatform.properties中获取 |
| appName | 项目名 hshc\_common.properties中获取 |
| appKey | 项目key hshcCommonPlatform.properties中获取 |

## 2.4服务端设计

### 2.4.1业界方案

ConfigServer2.0架构



Consumer/Provider启动时，通过统一域名从Address Server获取ConfigServer集群的地址列表；

Provider随机选取一台机器建立长连接，并进行服务注册；

ConfigServer集群中的机器接受到服务注册信息后，广播到集群其它机器（异步写）；

ConfigServer集群中的机器推送对应的服务注册信息到连接的Consumer;

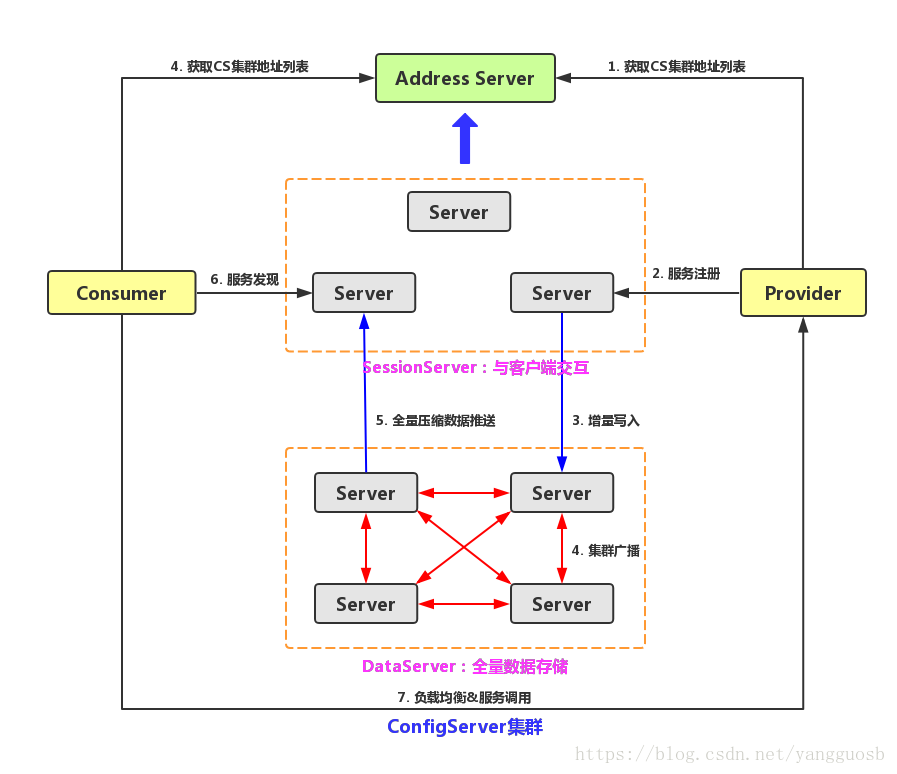
Consumer得到Provider的地址信息后，按照一定的负载均衡策略，进行远程服务调用；

适用场景

　　ConfigServer集群规模为中小型规模场景。原因是ConfigServer集群中是通过两两互联的方式进行数据同步的，当集群规模变大后，两两互写的竞争越来越大，单机的内存、网卡和连接数等都会成为瓶颈；

ConfigServer3.0架构

ConfigServer集群被垂直拆分成两部分：DataServer和SesionServer。SesionServer负责与客户端的交互，可以随着客户端数据量水平扩展；DataServer负责全量数据的存储，只有固定数量的少量机器，天然具备Failover机制，一台机器宕机不会影响到其它机器，如下图所示：



### 2.4.2花生方案

