昵称：穿林度水
园龄：2年4个月
粉丝：2
关注：62
+加关注

< 2017年7月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签
asm cglib(1)
cat(1)
cxf webService(1)
diamond(1)
excel(1)
JavaMelody(1)
java常用 api(1)
jdk(1)
jvmtop 监控(1)
liunx 安装文件(1)
更多

随笔分类
Btrace(1)
cat(1)
diamond(7)
Disruptor(2)
dubbo
eclipse(5)
Esper(1)
findbugs(2)
hadoop 生态(3)
jafka(1)
java util(9)
java监控(3)
js(3)
jvm(4)
liunx(7)

淘宝Diamond架构分析

http://blog.csdn.net/szwandcj/article/details/51165954

背景知识

早期的应用都是单体的，配置修改后，只要通过预留的管理界面刷新reload即可。后来，应用开始拆分，从单一系统拆分成多个子系统，每个子系统还会对应多个运行实例，就开始面临一些问题：
1. 配置分散在多个业务子系统里，对同一配置的翻译在多个子系统里经常不一致。比如订单和购物车都有货币类型的配置，如果购物车上了一种新的货币类型而订单却没有相应同步增加配置项就会造成程序错误。
2. 将配置收敛成一个公有服务，可以有效改善，但是又会带来其他问题。在复杂应用里，修改一个配置项，无法确切的知道需要刷新哪些相关子系统。最终只能做全量刷新，甚至是停机发布。这对于一些停机敏感的应用例如电商几乎是无法接受的。
3. 配置收敛后，配置中心成了应用中的单点，配置如果挂了，应用也会跟着产生异常甚至挂掉。

Diamond就是为了解决这些问题，它是个高可用的配置中心。

Diamond的配置类型

配置是Diamond的核心域，也是Diamond致力于去解决的问题。Diamond有两个主要配置类型– single和aggr。二者结构如下：

ConfigInfo	ConfigInfoAggr
-dataId	-dataId
-group	-group
-content	-content
-md5	-datumId
-appName	-appName

Aggr和single相比，少md5多datumId。DatumId是aggr的逻辑主键，aggr下dataId和datumId是1对多的关系， 也就是说多条aggr会聚合成一条single，diamond通过merge任务对aggr合并最终生成一条single。

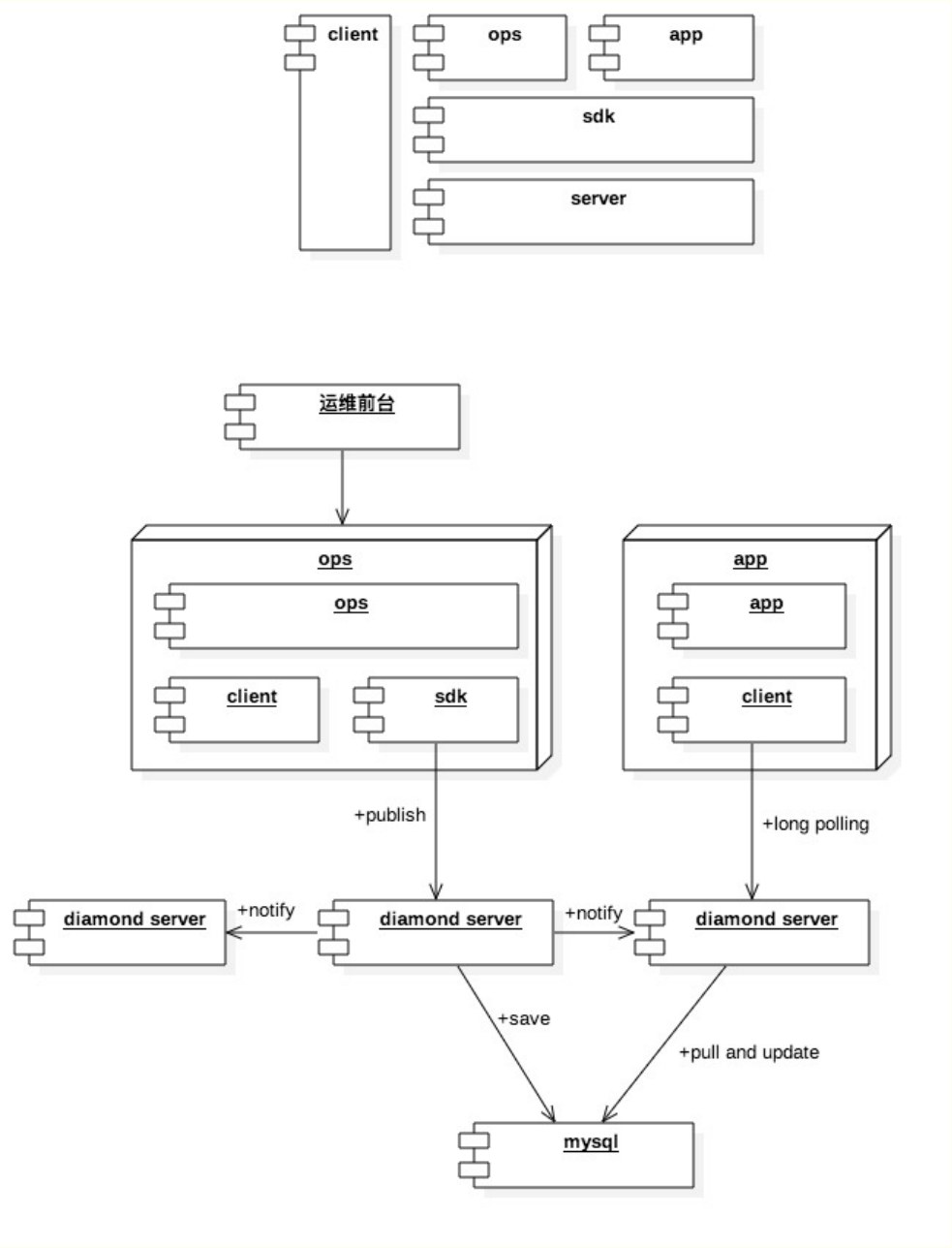
Md5是对content md5编码生成的字符串，用于判断缓存数据相比数据库数据是否不同，缓存数据必须严格与数据库数据一致，diamond并没有数据版本，默认数据库数据是最新的，也就是说如果数据库数据发生回退，即使缓存数据更新也会跟着回退。

Single才有md5，aggr其实并不算是完整的配置（多条aggr一起才是一个完整的配置），所以不需要校验数据是否改变。

整体架构设计

下图是Diamond的组件视图。Diamond主要有ops, sdk, client和server 4个组件。Ops是运维用的配置工具，主要用于下发以及查询配置等；server则是Diamond的后台，处理配置的一些逻辑；sdk则是提供给ops 或者其他第三方应用的开发工具包；client则是编程api，它和sdk乍看有点像，其实差别很大，sdk是用于构建前台运维配置程序的，本质是对数据 的维护，所有的访问和操作都是直接面向数据库的；而client则是这些数据的消费者，

事实上准确的说是diamond的消费者们（各子系统）都是通过 client组件对server访问。



Diamond server是无中心节点的逻辑集群，读请求都是访问local file，而写请求则会先进入数据库，接着再更新各节点缓存。注意：ops或者其他第三方运维系统（其实就是sdk模块）读取和写入的都是数据库，这很容 易理解，缓存会有lag，配置系统必须面向的是实时数据。

Diamond的数据库是单点的，这就可以利用数据库特性保证数据的原子性，一致性和持久性，也就不需要实现类似zk的集群协议，也就不存在 leader/follower以及observer等节点角色，它是去中心化的，所有节点都可以接受任意请求。Diamond是典型的读多写少，写一般 都来自运维系统例如ops，这种请求量会很小，即使峰值期对数据库的冲击也不会太大。实际上它就是数据库之上的一个保护壳，数据库的数据通过它透出来，也 通过它渗进去。

Diamond的同质节点之间会相互通信以保证数据的一致性，每个节点都有其它节点的地址信息，其中一个节点收到变更请求后，首先写入数据库，再通知所有同质节点更新缓存，保证数据的一致性。

为了保证高可用，client会在app端以本地文件形式缓存数据的snapshot，保证即使server不可用时app也可用，这一点和dubbo很相似，所以也完全可以使用diamond搭建dubbo注册中心。

内存缓存

- Client端使用的内存cache是一个AtomicReference
  - 它并不是通常理解的内存缓存，而只是一个事件源，只有被监听的配置才会有cache。Cache内聚了group，dataId，md5，content和listener等。
  - 客户端的长轮询任务（下一节将会重点介绍）只轮询被监听的配置，也就是cache的数据。客户端在pull到新数据后首先会更新 snapshot，再更新cache，接着全量对比所有cache和它关联的listener的md5信息从而知道配置更新有没有被通知，没有则以 cache中的内容作为消息载体通知，通知完成后更新listener的md5。
  - 没被监听的数据不需要轮询，因为diamond提供的读数据api默认会先从服务节点获取实时数据。

mysql(2)
oracle(2)
spring 生态(4)
TBSchedule(2)
TSharding(2)
web(2)
zookeeper(1)
限流(5)
字节码(1)

随笔档案

2017年4月 (2)
2016年12月 (1)
2016年11月 (3)
2016年10月 (18)
2016年9月 (10)
2016年8月 (8)
2016年7月 (6)
2016年6月 (9)
2016年5月 (8)
2016年4月 (4)
2016年3月 (4)
2016年2月 (1)
2016年1月 (2)
2015年12月 (12)
2015年11月 (22)
2015年10月 (4)

最新评论

1. Re:diamond
@淫贼github 上面搜一个...
--穿林度水
2. Re:diamond
检出源码需要输入SVN账号密码，账号密码是什么啊？
--淫贼

阅读排行榜

1. Oracle中的 UPDATE FROM 解决方法(2951)
2. 接口限流算法总结(1816)
3. eclipse 导入maven 父子项目(1108)
4. log4j日志异步化大幅提升系统性能(1009)
5. 取消eclipse js验证(979)

评论排行榜

1. diamond(2)

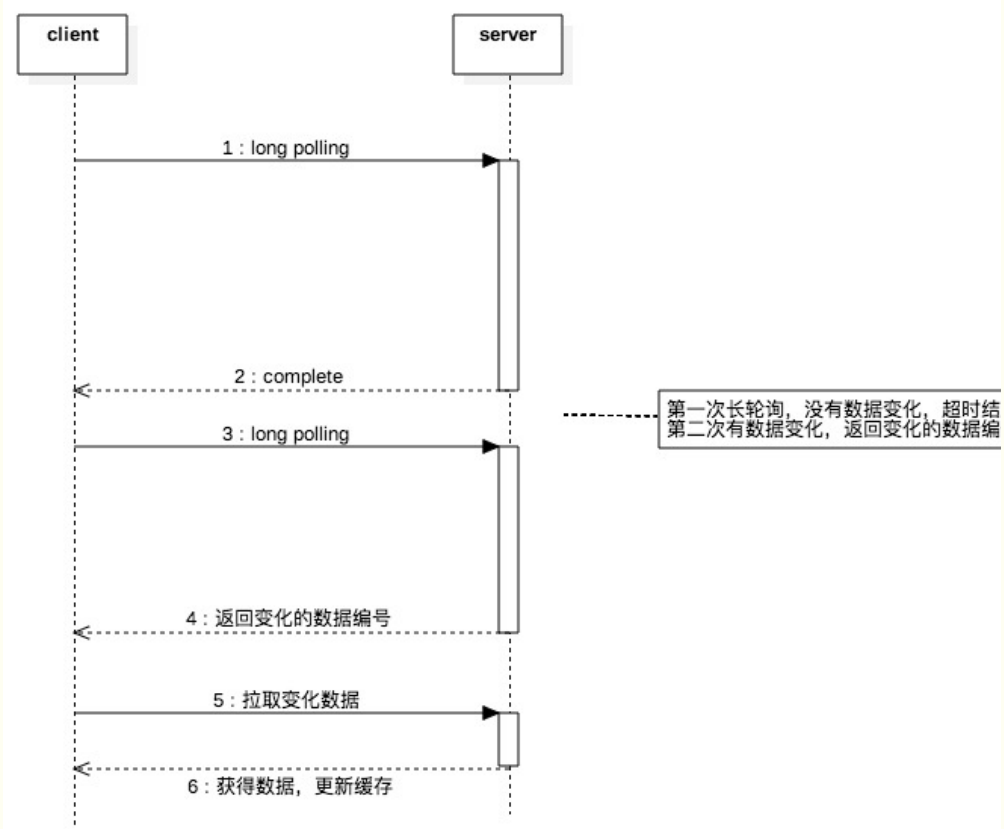
推荐排行榜

1. TSharding源码阅读-MapperShardingInitializer(1)

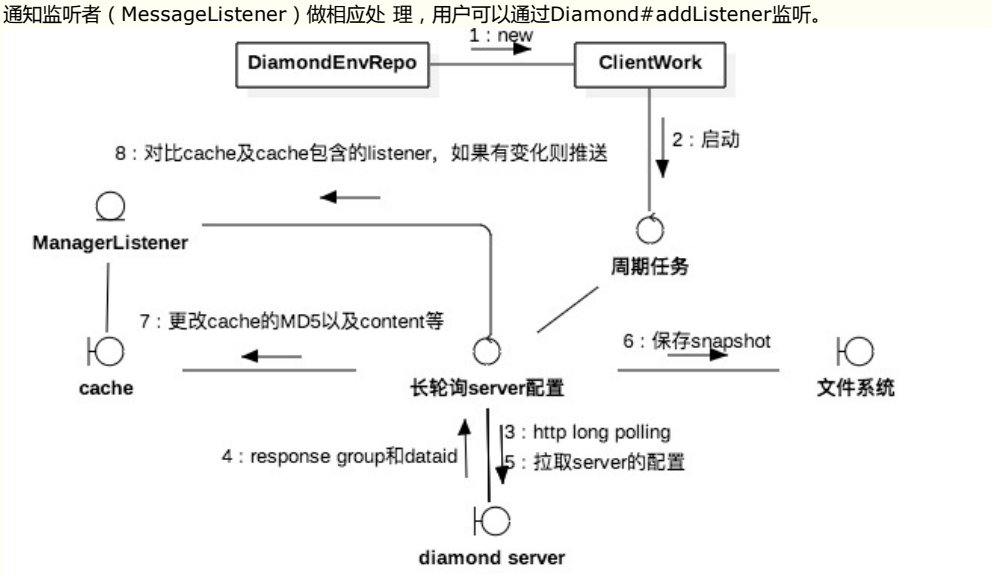
- 在客户端发起长轮询或者服务节点做dump时，都需要对比md5信息以确定是否要推送或者dump。Server端缓存全量缓存了所有配置的md5信息，并会第一时间得到更新，得到更新同时还会推送LocalDataChangeEvent。
- 无论客户端还是服务端，内存缓存仅仅是为了满足某种功能需求，并不作为读的数据源（客户端只缓存部分数据，服务端不缓存配置内容）。这是基 于产品本身定位而来的，产品定位本身就是牺牲一部分速度以降低成本，并且同时提供长轮询机制为时效性要求高的配置做到准实时的变更推送。但在客户端，每个 应用的兴趣点都是分散的，平均下来每个应用感兴趣的配置数据并不大。

长轮询

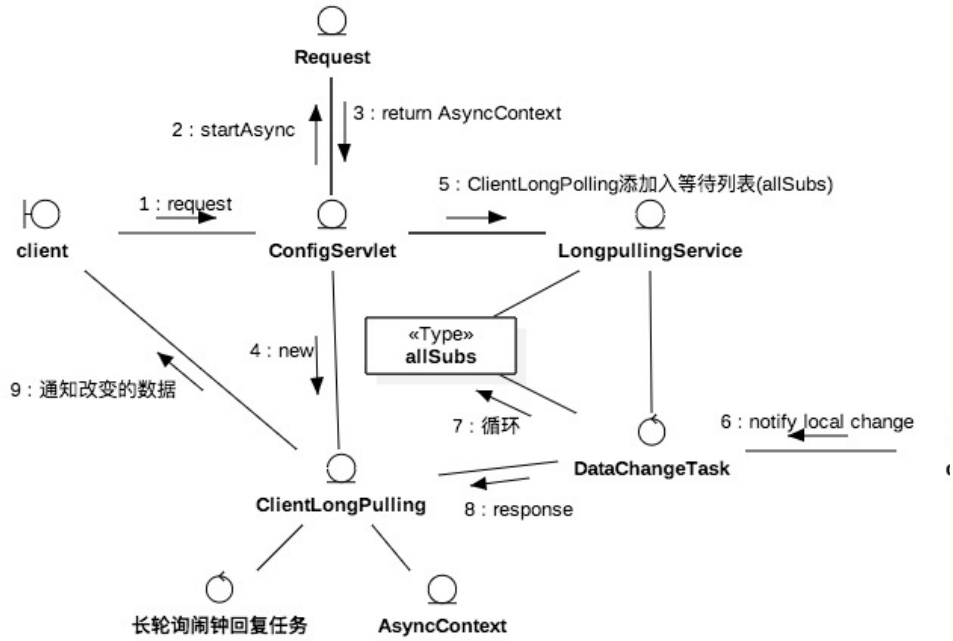
Client会不断长轮询server，获取最新的配置推送，尽量保证本地数据的时效性。



Client默认启动周期任务对server进行长轮询感知server的配置变化，server感知到配置变化就发送变更的数据编号，客户端通过 数据编号再去拉取最新配置数据；否则超时结束请求（默认10秒）。拉取到新配置后，client会通知监听者（MessageListener）做相应处 理，用户可以通过Diamond#addListener监听。



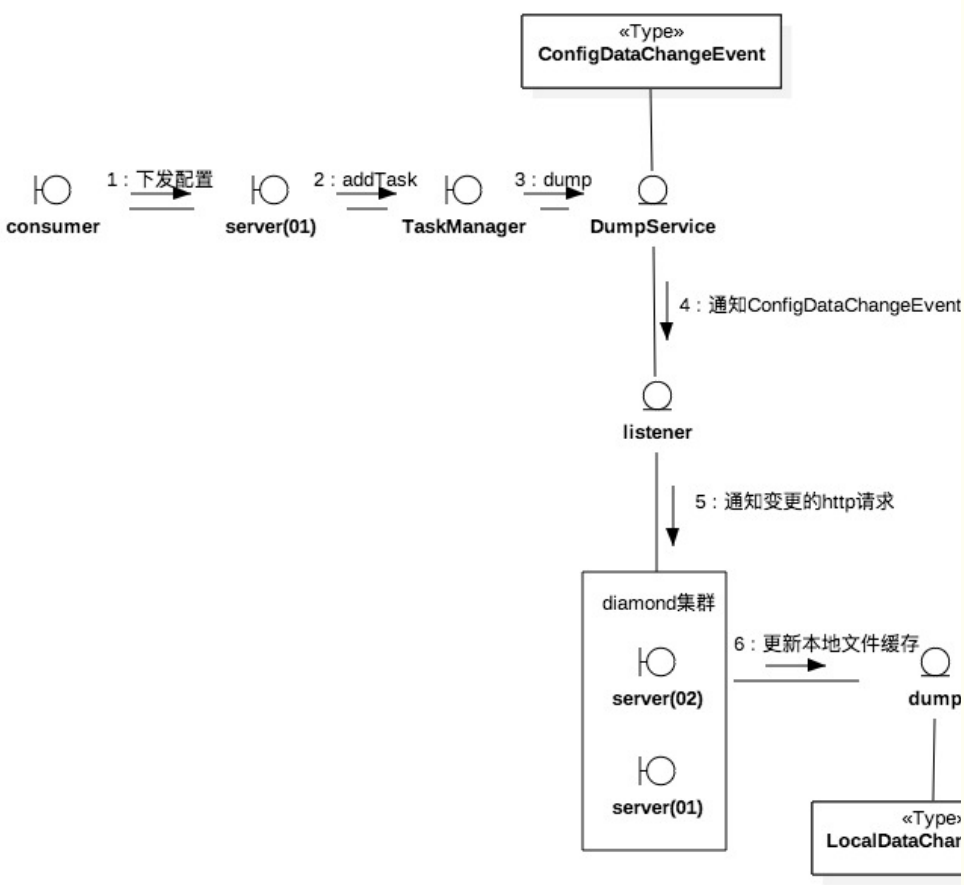
服务端通过AsyncContext对请求做非阻塞处理，通过定时任务感知配置变化。



详细描述下上图，
1. server收到请求后启动AsyncContext，并基于它构造ClientLongPulling。ClientLongPulling除了AsyncContext还有一个超时回复任务对长轮询请求做超时处理
2. 之后ClientLongPulling被加入等待列表。LongPullingService关联一个感知数据变化的定时任务，当有数据变化时（收到 LocalDataChangeEvent），就会循环等待列表里的ClientLongPulling，推送数据变化回客户端。
4. Dump是抽象出来的一块儿概念，server的数据变化都会触发响应的dump task，并会发送相应的事件，由server感知，DataChangeTask也是一个事件监听者，能感知local file的数据变化。

服务端架构设计

Diamond集群是去中心化的，使用通知机制保证集群各节点数据一致。

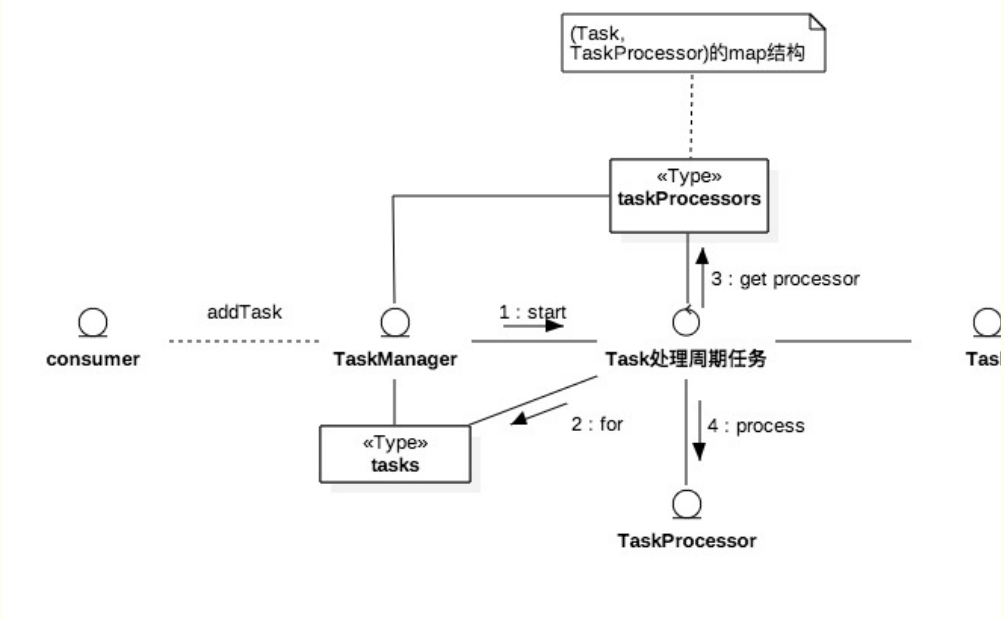


1. Diamond集群内每个节点都有其他节点的地址信息，当一台节点收到写请求后首先写数据库，接着就会发送ConfigDataChangeEvent， 监听者收到该事件后通知所有节点做数据变更。通知的所有节点也包括自己，下发配

置的请求只会更新数据库，并不会更新本地文件缓存。
2. 通知发送到所有节点后，通过dump更新local file。Dump是将配置对象dump进本地文件的过程，dump完毕后发送LocalDataChangeEvent，见上节。

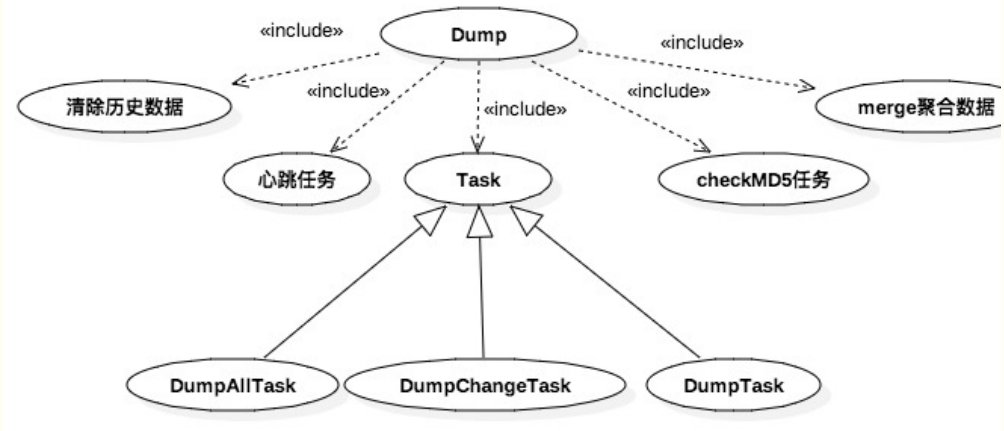
任务管理

Diamond收到配置请求，先执行数据库操作，然后再向任务管理器的任务栈里插入一条任务，任务管理器感知到新任务后，pick相应的 TaskProcessor处理。TaskProcessor和Task的关系通过 TaskManager#addProcessor（或 setDefaultProcessor）设置。
Diamond的任务管理器是FIFO的，这会造成长延时任务阻塞其他任务的执行。为解决这个问题，Diamond的开发者为每个Task都定制 TaskManager。用户可以做些优化，参考hadoop的公平算法，针对应用场景（比如长延时，离线，实时等等）定制TaskProcessor。

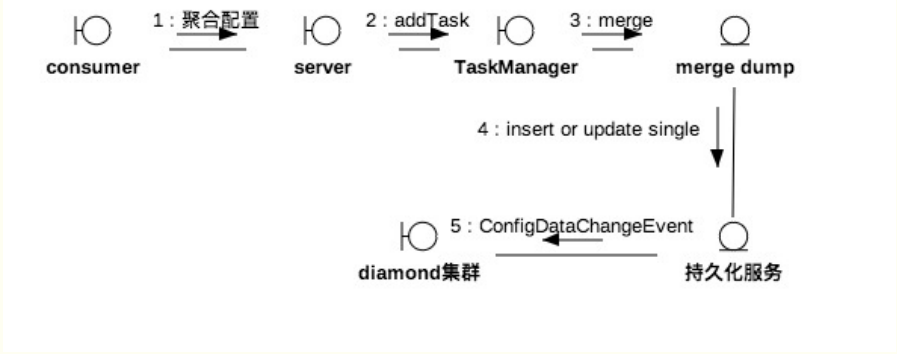


其他一些任务

为了保证Diamond的数据一致，除了以上介绍的两类任务，还有其他一些task，见下图：



1. Merge任务用于合并aggr。App下发聚合配置到diamond后，会触发合并任务生成single，同时广播配置变更事件（ConfigDataChangeEvent），各节点收到通知后拉取数据库相应single数据并更新local file。这与上面描述的server数据同步基本类似，只是事件源头换成了merge。



2. DumpAllTask会每6小时run一次做全量dump，全量删除老的缓存数据，生成新缓存。
2. DumpChangeTask做增量dump。通过和数据库的配置做md5对比，删除被移除配置的文件缓存，更新md5

不一致的文件缓存等等。
3. 清除历史数据用于清除1周前的数据库his表数据。
4. 心跳任务用于记录心跳时间，节点服务重启时会判断距离上次心跳时间是否已经超过一小时，超过一小时则做全量dump，否则做增量。

剩下的都很好理解，不再一一介绍，需要注意的是，这些任务并不是都是定时做，有些是事件触发的，例如DumpTask和merge；还有些在Diamond服务启动时会触发，如merge，DumpChangeTask等。

分类: diamond