# 反向代理or按需分发

**V 0.3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** | **评审** |
| 2013.07.05 | 0.1 | 任务分发的模式讨论，选择推模式还是拉模式，反向代理还是分级cache。 | 郭强 | 已审 |
| 2013.07.09 | 0.2 | 按照陈超给出的评审意见，修改。 | 郭强 | 已审 |
| 2013.07.12 | 0.3 | 按照评审会议给出的评审意见，修改；增加了改进方向，和实现方案。 | 郭强 | 待审 |
|  |  |  |  |  |

## 推模式vs拉模式

### 推模式

又称分发模式，是现网运行的模式。分发完成，然后发布播放的web页面。

* **优点**
  + 简单，可靠。
  + 对及时性要求不高。
  + 网络出现拥堵，或者瞬断，不会有很大影响。
* **缺点**
  + 分发时间较长。
  + 分发期间，带宽占用会比较集中。
  + 不管用户是否需要，都会推送。

### 拉模式

或者叫反向代理模式，是我们实验中的模式。发片时，可以先发布播放的web页面；有用户点击时，才代理用户请求，边下载边服务。

* **设计目的**
  + 缩短发片时间，可以直接发布播放页面。
  + 减少磁盘存储，只有用户点击的影片才会被拉到前端。
  + 同样的理由，可以减少分发时的网络带宽占用。
* **潜在问题**
  + 对网络健康状况依赖较高，网络发生拥堵，或者瞬断时，代理下载不及时，影响用户播放。
  + 对调度服务器tracker/tuner的要求较高。
    - 需要区分MS还是客户端，如果是客户端的请求，那么就近调度；如果是MS的请求，那么调度指向上级资源。
    - 调度服务器要实现负载均衡，否则会出现MS同时代理下载的数量，超出并发限制。
  + 限速策略，实现复杂度高。
    - 影片码率动态多变。
    - 多个用户同时下载同一部影片的不同range。
    - 多路下载代理，会出现抢占，可能会出现某一路下载速率不足。

初步结论：**在推模式的基础上改进，实现按需分发，按照影片的热度分发，后文称为按需分发方案。**

## 按需分发

存储分为三级：公司的中心片库，一级存储，二级存储。中心片库，并不对外服务。发片时，一级存储，固定分发；二级存储，按需分发，并定时回收。这样做的结果是：一级存储，存储了全量影片，二级存储，仅存储了热门影片。

中心片库

一级存储

一级存储

一级存储

固定分发

按需分发

按需回收

* **优点**
  + 在现有推模式之上的一次改进，保留了现有推模式的优点。
  + 同时，按需分发，可以及时应对影片的热度上升。
  + 同时，按需回收，可以减少二级存储的磁盘空间。
  + 二级cache配置，跟一级cache的影片配置可以区别对待。二级cache可以配置较多内存，一级cache可以配置较多磁盘。

### 影片热度

影片热度，由影片的点击量体现。按照点击量，对影片进行热度分级。计算公式为：热度等级 = 点击量/每级点击量。

* **谁的点击量？**
  + media\_id，代表一部电视剧/一部电影/一档综艺节目/一部动漫等等。
  + serialid，代表了media\_id中的某一集电视剧/某一期综艺。
  + hashid，同一个serailid，对应不同码率/不同文件格式，会生成不同的dat文件，每个dat文件，对应一个hashid。

因为分发的最小单位是dat文件，所以最好是统计hashid的点击量。

* **区分平台**

因为MediaServer服务PC平台，还是服务移动平台，是分开的；所以统计点击量，也应分开，区分是来自PC平台，还是来自移动平台。

* **实时统计**

统计点击量，最好能够实时汇总，或者接近实时汇总。

* **现状**

数据库corsair\_0，表fs\_fsplay\_stats，存储了media\_id每天的点击量。

* **需求**
  + 统计hashid的点击量。
  + 区分PC平台和移动平台。
  + 区分来自哪个运营商（逻辑）。
  + 实时统计。
* **难点**
  + 如何规定每级热度的点击量。

### 热门任务分发策略

* **热度每上升一个等级，多分发一份。**
* **分发及时性保证：**
  + 影片上线，预分发两份。这两份，可以支撑点击量2L。
  + 影片点击量在0~L的时候，不需要动作。
  + 在影片点击量突破L的时候，多分发一份。只要分发过程的耗时t1，小于点击量由L涨到2L的耗时t2，即可。
  + 要保证分发及时，目标是，降低t1>t2出现的概率。

### 冷门任务回收策略

* **热度每下降一个等级，回收一份。**
* **每天凌晨，回收昨天的降级。**

### 冷门任务回收造成的文件频繁删除问题

影片频繁升降级，造成dat文件频繁下载删除，可能会出现震荡现象。

* **解决办法**

删除dat文件时，仅标记为删除，例如改名为hashid.dat.rm；文件再次下发时，可以查找待删除文件，如果有，直接恢复改名，即可。等磁盘将满时，才真正删除dat文件。

## 整体部署结构

### 系统结构图

点播日志分析服务器

分发策略服务器

macross分发服务器

增/减分发

获取热度

下发任务

* **功能单元**
  + 点播日志分析服务器，分析用户上传的日志，实时统计影片热度信息。
  + 分发策略服务器，生成策略，实现按需分发，或者按需回收。
  + macross分发服务器，将任务下发给二级存储的MediaServer。
* **分发流程**
  + 分发策略服务器，与点播日志分析服务器，共享数据库。
  + 点播日志分析服务器，实时统计，将hashid的热度信息写入数据库。
  + 分发策略服务器，定时查询数据库，发现影片热度升级了，根据配置的策略决策，看是否增加分发。如果增加分发，则通知macross分发服务器。
  + macross分发服务器，收到增加分发指令，选择某个MS，下发任务。
  + MediaServer收到任务，下载该任务。
* **回收流程**
  + 每天凌晨，分发策略服务器，执行定时任务，通过查询数据库，盘点昨天影片热度是否比前天降级；如果降级，通知macross分发服务器，回收影片。
  + macross分发服务器，收到回收任务指令，选择某个MS，下发任务回收指令。
  + MediaServer将该影片标记为删除。
  + 磁盘空间满时，MediaServer真正删除影片。

### 两个要点

* 接近实时统计

影片的热度，最合适的参数应该是同时在线用户数。统计同时在线用户数，可能会有一些困难，会存在一些误差，所以我们决定近似：

* 统计一天的点击量，作为影片的热度表示。
* 统计一小时的点击量，作为影片的热度表示。

总之，越接近实时统计越好。

* 磁盘用时回收

冷门任务的回收是为了减少磁盘存储，如果磁盘存储够用，没有必要回收。到磁盘全满时，回收，可能会来不及。所以，设定磁盘存储到达80%时，回收。

## 实现方案

两个概念定义。

* 热度和热度等级
  + 热度，定义为一个时段内用户点击量，越接近实时越好；
  + 热度等级，定义为热度/一级热度点击量，是决定分发几份dat文件的人为分级。
* 冷度，针对热度为0的影片，需要做区分，所以制定如下指标：
  + 冷度指标1，定义为最近连续未观看的天数，如果为N天，那么冷度为-N；
  + 冷度指标2，定义为未观看天数与上线天数的占比；
  + 冷度指标3，定义为上线之后的总点击量。

方案实现需要三个服务器，日志分析服务器，分发策略服务器（更名为热度调度服务器），macross分发服务器。

### 日志分析服务器

* 程序输入

用户点击播放过程中，上传的日志。

* 程序逻辑

统计dat文件的点击量。

按照dat文件的hashid，分平台（PC平台/移动平台），分运营商，分时段（一天内/一小时内）统计点击量。得到的结果类似如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列1** | **列2** | **列3** | **列4** | **列5** |
| hash\_id | 运营商id isp\_id | PC平台点击量 play\_num@pc | 移动平台点击量 play\_num@mobile | 开始统计时间 start\_time |
|  |  |  |  |  |
|  |  |  |  |  |

* 程序输出

按时段统计完成后，结果写入一个文件，放在指定路径下。

* 配置参数

时段长度（一天还是一小时），作为日志分析服务器的配置参数。

### 热度调度服务器

* 程序输入
  + hashid的列表，来自macross数据库。
  + hashid的点击量，来自日志分析服务器的分析结果。
* 程序逻辑
  + 数据库
    - 表1：table\_hashid

|  |  |  |
| --- | --- | --- |
| **列1** | **列2** | **列3** |
| hash\_id | 上线时间 online\_time | 状态: 是否下线 status |
|  |  |  |
|  |  |  |

* + - 表2：table\_hashid\_playnum

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列1** | **列2** | **列3** | **列4** | **列5** |
| hash\_id | 运营商id isp\_id | PC平台点击量 play\_num@pc | 移动平台点击量 play\_num@mobile | 开始统计时间 start\_time |
|  |  |  |  |  |
|  |  |  |  |  |

* + - 表3：table\_hashid\_cold

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列1** | **列2** | **列3** | **列4** | **列5** |
| hash\_id | 指标1 param1 | 指标2 param2 | 指标3 param3 | 开始计算时间 start\_time |
|  |  |  |  |  |
|  |  |  |  |  |

* + 模块划分
    - 热度升级管理器，只负责热度升级，实时运行。
    - 热度降级管理器，只负责热度降级，定时执行。（暂不需要）
    - 冷度计算器，只负责定时计算冷度三个指标。
    - 冷门选择器，只负责选择最冷的N个hashid，通知macross删除任务。
  + 处理逻辑
    - 程序启动时，从macross数据库读取hashid列表，写入table\_hashid。
    - 定时查询macross数据库，看是否有最新的hashid，如有，写入table\_hashid。
    - 定时从日志分析服务器获取到hashid的点击量统计，写入table\_hashid\_playnum。
    - 写入数据库时, 处理热度升级：
* 如果是按天统计，那么，今天截止到现在的点击量，与昨天全天的点击量作对比。
* 如果是按小时统计，那么，本小时内截止到现在的点击量，与昨天全天到现在的峰值做对比。
  + - 定时计算冷度，每个时段结束后，找到点击量为0的hashid（表1中没有标记为删除的hashid，而表2中不存在的hashid）并为之计算冷度的三个指标，写入table\_hashid\_cold，并写入更新时间。
    - 定时选择最冷的N个任务，通知macross删除任务。
  + 读写互斥问题

批量数据写入一半时，执行查询操作；查询到的数据，可能一半是新的，一半是旧的。

通过设置一个特殊行（hashid=0），如果有该行，那么这个时段已写完，否则，状态是写入进行中。

* 程序输出
  + 提供hashid热度的查询接口

http://heat.funshion.com/film\_heat?hashid=xxxx&ispid=yyyy

返回：hashid=xxxx，ispid=yyyy，playnum=xxx，heat\_level=N。

报文格式可选择json，xml，或者行式文本。

http:// heat.funshion.com/film\_heat?hashid=xxxx

返回：hashid=xxxx: ispid=1, playnum=xxx, heat\_level=N; ispid=2, playnum=yyy, heat\_level=M.

* + 提供hashid冷度的查询接口

http://heat.funshion.com/film\_cold?hashid=xxxx

返回：hashid=xxxx, param1=-100, param2=10%, total\_play\_num=xxxx

* + 通知macross分发影片

如果热度等级发生变化，通知macross该运营商应该有几份dat文件。

通知macross的接口定义如下：

http://www.macross.com/notify\_distribute?hashid=xxxx&ispid=yyyy&distribute=N

url参数说明：

* + - hashid=xxxx，代表dat文件的info\_hash；
    - ispid=yyyy，代表运营商的id；
    - distribute=N，表示该运营商内应该有N份dat文件；
* 配置参数

一级热度对应的点击量，作为分发策略服务器的配置参数，随时可以调整。

热度非常高之后，可能会分发太多份数。最大分发量作为分发策略服务器的配置参数。

* 数据库访问速率优化

建议在数据库前面放置memcached。

### macross分发服务器

* 程序输入

hashid在运营商内应该有N份的通知消息。

* 程序逻辑

macross分发服务器，收到消息，查询该运营商内现有几份，应该增加分发，还是减少分发。

然后，在该运营商内选择MS服务器，通知MS服务器，下发或者回收任务。

* 程序输出

通知MS服务器分发dat文件，或者回收dat文件。

## 后续改进

* 减少通信量
  + 日志分析服务器与热度调度服务器之间的通信量

日志分析服务器，实时统计到hashid的点击量，热度服务器收到点击量，要做两件事情：入库，决定是否增加分发。这个功能能否实现为一个reduce函数，被map/reduce调用。

* + 热度调度服务器与macross分发服务器之间的通信量
    - 初次运行时，热度调度服务器要去从macross数据库同步hashid列表；
    - 新的hashid产生后，热度调度服务器需要通过多次查询，才能得到。

能否改为热度调度服务器，直接使用macross数据库中的表。

* 热度调度服务器，对外提供服务：
  + 提供最冷的N个hashid列表，接口形式如下：

http://heat.funshion.com/coldest\_hashid?N=100

返回：json或者xml形式，内容为：

hashid，指标1，指标2， 指标3；

macross可以使用该接口删除冷门任务。

* + 提供最热的N个hashid列表，接口形式如下：

http://heat.funshion.com/heatest\_hashid?N=100

返回：json或者xml形式，内容为：

hashid，指标1，指标2， 指标3；

macross可以使用该接口决定优先分发哪些任务。

* + 对hashid列表进行按照从冷到热排序，接口形式如下：

http://heat.funshion.com/sort\_by\_cold?id1:id2:id3...

返回：json或者xml形式，内容为：

hashid，指标1，指标2， 指标3；

MediaServer可以提供列表，热度调度服务器进行排序，MediaServer进行冷门任务删除。

* + 对hashid列表进行按照从热到冷排序，接口形式如下：

http://heat.funshion.com/sort\_by\_heat?id1:id2:id3...

返回：json或者xml形式，内容为：

hashid，指标1，指标2， 指标3；

MediaServer可以提供列表，热度调度服务器进行排序，MediaServer进行热门任务的均衡，优先保证热门任务的服务。

* 热度调度服务器的进化

调度服务器，目前是为自动分发/自动回收提供策略；将来可能为调度服务器tracker/tuner提供实时的影片热度信息，可能会更名为热度中心。

## 几个讨论问题

* 删除任务要不要人工干预，还是要做到完全自动。
* 影片上线2个月内，属于试运营阶段，不能删除。
* 建立影片保护列表，一些特别的影片，不能删除。
* 小运营商内，磁盘存储容量有限，优先分发TopN。
* 评判热度时，网站前几页的影片热度需要加大权重。

## 会议评审意见

* 运营商id改为ip分组id
* http命令修改如下：
  1. // ?cmd=get\_hotest&N=10
  2. // ?cmd=get\_coldest&N=10
  3. // ?cmd=get\_hot&hashid=xxx
  4. // ?cmd=get\_hot&hashid=xxx&ip\_group\_id=y
  5. // ?cmd=get\_cold&hashid=xxx
  6. // ?cmd=get\_cold&hashid=xxx&ip\_group\_id=y