**公共服务平台**

**静态资源架构方案**

|  |
| --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 1.0 | 刘岗 |  | 2014-12-22 | 初稿 |
| 2.0 | 刘岗 | 刘刚 | 2014-12-29 | 根据架构师刘刚的建议完善相关内容 |

**版权声明**

版权所有 © 科大讯飞股份有限公司，保留所有权利。

**商标声明**

安徽科大讯飞信息科技股份有限公司的产品是安徽科大讯飞信息科技股份有限公司专有。在提及其他公司及其产品时将使用各自公司所拥有的商标，这种使用的目的仅限于引用。本文档可能涉及安徽科大讯飞信息科技股份有限公司的专利（或正在申请的专利）、商标、版权或其他知识产权，除非得到安徽科大讯飞信息科技股份有限公司的明确书面许可协议，本文档不授予使用这些专利（或正在申请的专利）、商标、版权或其他知识产权的任何许可协议。

**不作保证声明**

安徽科大讯飞信息科技股份有限公司不对此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。本手册内容若有变动，恕不另行通知。本手册例子中所用的公司、人名和数据若非特别声明，均属虚构。未得到安徽科大讯飞信息科技股份有限公司明确的书面许可，不得为任何目的、以任何形式或手段（电子的或机械的）复制或传播手册的任何部分。

**保密声明**

本文档（包括任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，除用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

本软件产品受最终用户许可协议（EULA）中所述条款和条件的约束，该协议位于产品文档和/或软件产品的联机文档中，使用本产品，表明您已阅读并接受了EULA的条款。

目 录

[1 编写目的 4](#_Toc407796565)

[2 使用对象 4](#_Toc407796566)

[3 需求演进 4](#_Toc407796567)

[4 架构现状 5](#_Toc407796568)

[5 存在问题 7](#_Toc407796569)

[6 问题分析 7](#_Toc407796570)

[7 方案假设 8](#_Toc407796571)

[8 方案制定 8](#_Toc407796572)

[9 技术选型 9](#_Toc407796573)

[10 组件介绍 11](#_Toc407796574)

[11 详细设计 13](#_Toc407796575)

[11.1 整体逻辑结构 13](#_Toc407796576)

[11.2 实际部署结构 13](#_Toc407796577)

[11.3 接口API规范 14](#_Toc407796578)

[11.4 接口API 18](#_Toc407796579)

# 编写目的

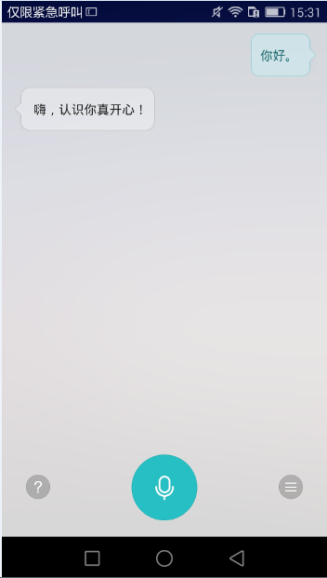
本文档详细介绍了公共服务平台静态资源管理平台的架构设计过程以及统一接口的使用规范。

# 使用对象

本文档可以作为架构人员的评审材料，开发人员的统一接口开发指导，以及运维人员对静态资源管理平台的部署优化指导文档。

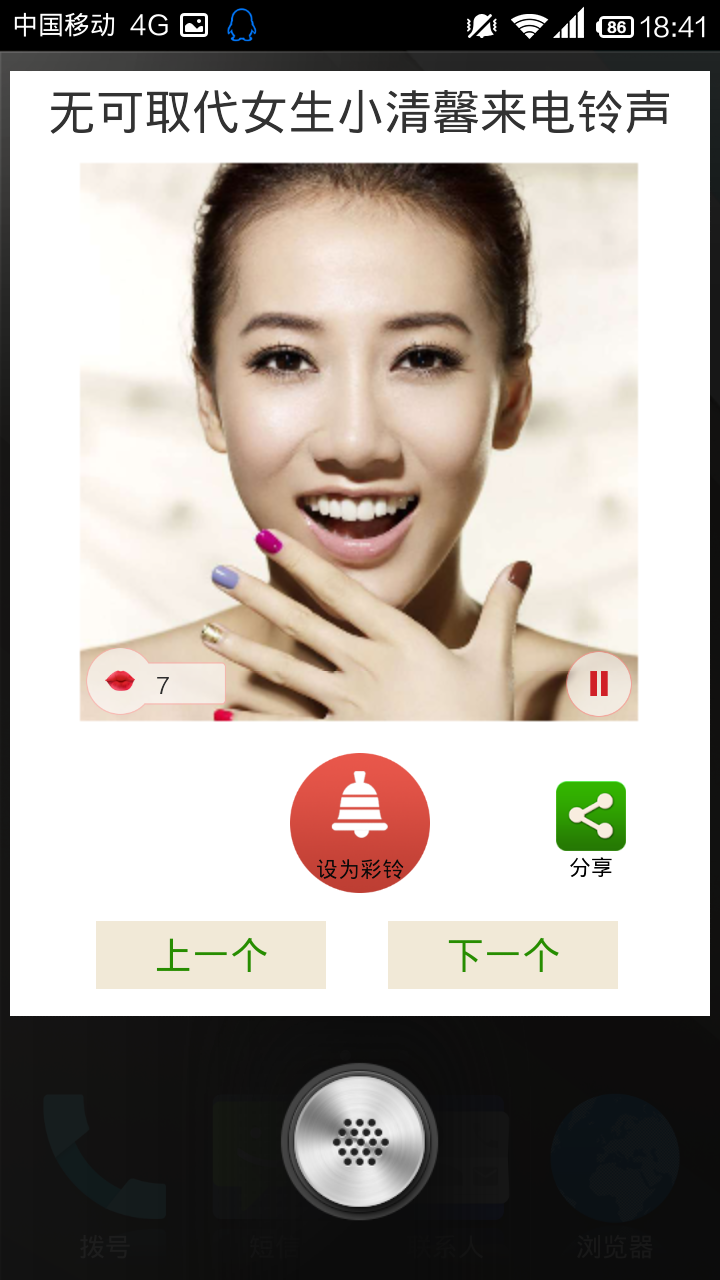
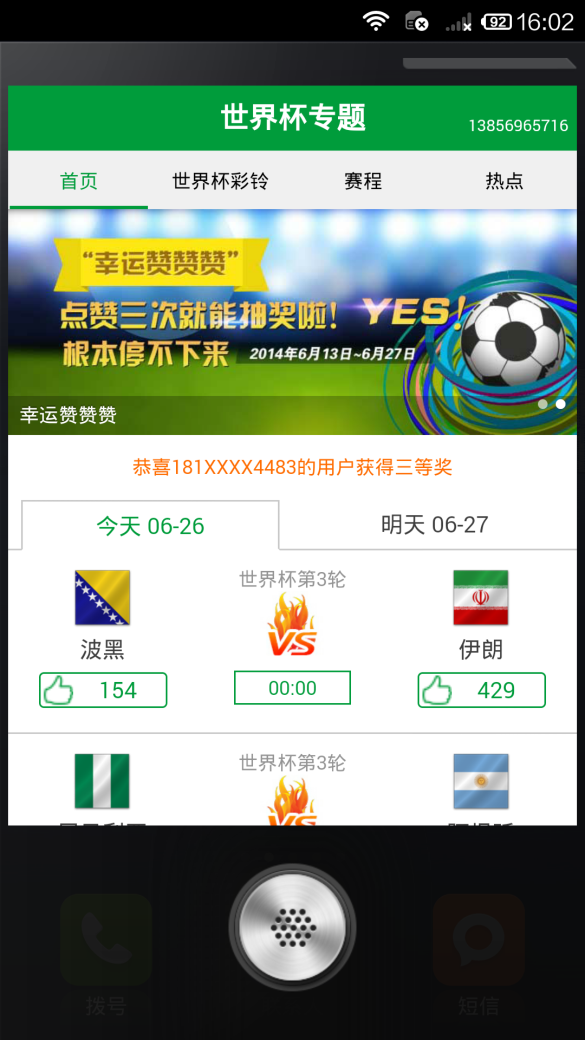
# 需求演进

公共服务平台一期，手机定制语点对于各厂商定制语点的语音结果返回大多采用文本的方式，如下图所示。

图片1 小米语音助手 图片2 OPPO语音助手 图片3华为语音助手

公共服务平台二期，由于我们掌握了用户的属性和行为数据信息，同时在部分H5页面的聚合上有一定的自主权，也就意味着我们可以直接通过广告的方式来获取一部分运营收益,因此在二期项目中开始在重点厂商逐步验证H5页面化的运营效果。



图片三 小米闲聊推送 图片四 小米世界杯活动 图片五 小米酷音铃声

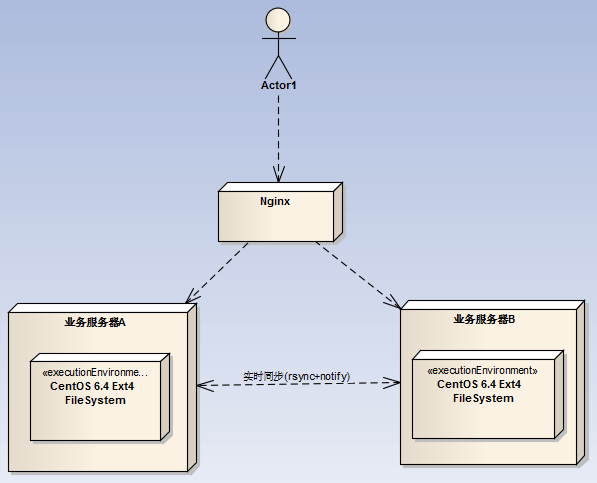
公共服务平台三期，通过二期H5的验证和技术的积累，三期开始在小米，OPPO，华为等各厂商开始大量使用H5页面结果返回。



图片六 小米的H5 图片七 OPPO的H5 图片八 华为的H5

# 架构现状

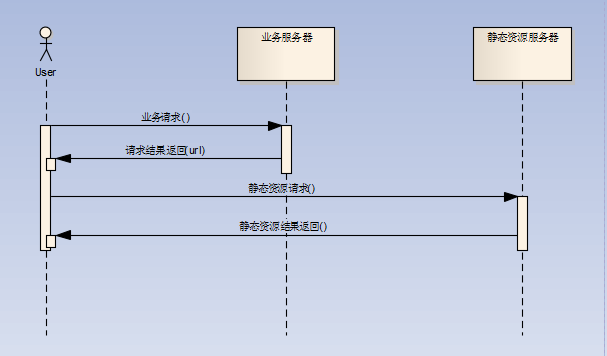
公共服务平台二期，为满足返回结果页面化的需求，在技术实现上亟待解决返回结果页面上的静态资源的集中存放和访问问题，考虑二期处于验证阶段，静态资源的访问量不是很大，于是在业务服务器上划分出一部分磁盘空间作为静态资源的集中存放区域并提供外部访问接口，考虑为避免单点故障问题，采用多台业务服务器的静态资源实时同步，前端负载均衡。



图片九 架构概览

架构介绍：

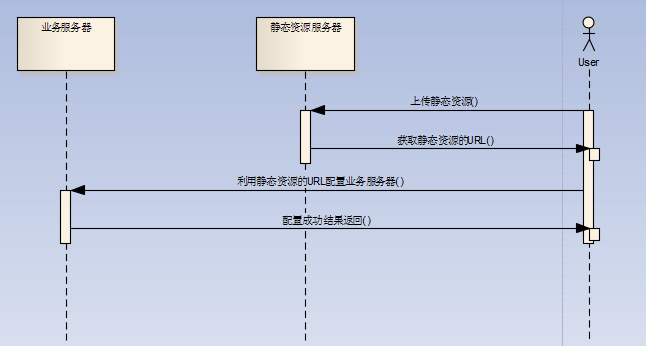
1. 前端采用负载均衡，避免单点故障问题，提高服务器扛并发能力。
2. 业务服务器OS采用CentOS6.4，文件系统采用的是Ext4，在OS的文件系统上存放静态资源。
3. 采用负载均衡的两台业务服务器采用实时同步的技术，保证两台服务器之间的静态资源对等。



图片十 用户交互时序图

用户交互时序介绍：

1. 用户通过语音请求业务服务器，业务服务器根据请求返回页面访问的URL。
2. 用户根据获取的URL直接访问静态资源。



图片十一 运营配置人员交互时序图

运营人员交互时序介绍：

1. 运营人员每次新的业务上线，首先将静态资源上传到静态服务器上，获取URL
2. 将静态资源的URL再根据业务逻辑配置到业务服务器上。

# 存在问题

随着页面化的需求也来越多，目前的静态资源架构现状存在以下几方面主要问题。

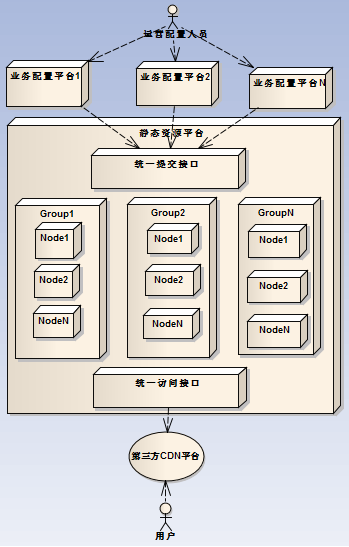
1. 没有静态资源管理平台，新业务上线，运营人员将静态资源上传到指定的文件路径下，按照原先设置的规则生成URL，导致指定的路径下文件存放凌乱复杂难以管理，文件节点越来越多且无法知道哪些文件可以删除，服务器访问效率低下。
2. 由于新功能上线需要运营人员先上传静态资源再配置业务，增加运营人员的工作量，系统的易用性差。
3. 为节省服务器资源，之前静态资源存放于业务服务器上，导致静态资源访问量增大占用带宽现象严重，影响正常的业务服务器工作。
4. 扩展性差，服务器随着静态资源的增多，很难扩展到其它服务器。
5. 随着静态资源的访问量的增大，对于带宽的要求越来越高，在带宽资源难以协调情况下，用户的体验急剧下降。

# 问题分析

针对以上存在问题，需求条件假设

1. 运营配置人员关注的是静态资源的访问URL，并不关心文件实际存放于何处。假设存在一个平台能够屏蔽静态资源文件的实际存放位置，运营配置人员只需输入静态资源文件，直接获取静态资源的访问URL，平台根据输入的静态资源文件按照策略进行合理的存放，避免文件存放凌乱的问题，从而提高服务器的访问效率。
2. 针对新功能的上线，运营配置人员关心具体业务，最好能在具体业务的配置页面进行静态资源的提交，而避免之前的二次交互操作。业务配置系统根据具体的上传文件和业务绑定后提交静态资源平台，提高系统的易用性。
3. 针对静态资源访问占用带宽影响正常业务访问问题，将静态资源与业务剥离，单独的部署静态资源平台。
4. 假设存在一个平台能够根据资源量的大小，具有可扩展性。
5. 针对用户访问速度慢的问题，平台能够很好的与CDN加速平台很好的结合。

# 方案假设



图片十二 方案假设概览图

根据存在的问题分析，假设静态资源平台应具备如下特点：

1. 优化文件的存放路径，按照一定规则进行存放，避免操作系统单文件夹下面存放过多的文件，导致访问效率低下，且对于上层配置应用屏蔽这些存放策略。
2. 有统一的接口可以允许其它的应用进行访问，输入静态文件资源，输出实际可访问的URL，且根据厂商提供CDN加速的接口和非CDN加速的接口。
3. 平台架构易于扩展，可根据实际的文件存放量进行节点数的控制。小批量存放可以单机搭建，大批量存放可扩展节点。
4. 平台具有很高的稳定性，对于节点可以有相应的备份节点，如图中的Node1，Node2互为容灾备份节点。
5. 平台对于存放的文件存有文件的描述信息，例如，文件的大小、关联的业务等，便于后期的维护。
6. 平台对外需要提供统一的访问接口，同时也支持用户的分散访问。

# 方案制定

根据分布式架构原理范型设计，项目组采用分布式文件系统作为平台实现的最终方案。底层用分布式文件系统的开源组件，在此基础之上根据实际的业务需求对组件进行二次接口开发或者调用接口的封装，从而达到最终的方案目的。

分布式文件系统基本要素：

1. 从文件系统的客户使用的角度来看，它就是一个标准的文件系统，对外提供一系列的API，由此进行文件或目录的创建、移动、删除、以及对文件的读写等操作。
2. 文件系统的内部，分布式文件系统不是管理本地磁盘，它的文件内容和目录结构都不是存储在本地磁盘上，而是通过网络传输到远端系统上。并且，同一个文件存储不只是在一台机器上，而是在一簇机器上分部署存储，协同提供服务，从而提高系统的容灾能力和可扩展性。

# 技术选型

分布式文件系统的发展概况

在分布式文件系统研究领域国外起步较早，目前业内应用成熟的系统较多，包括GFS，HDFS，Lusture,MogileFS,MooseFS等等。其中以GFS模式为代表的分布式文件系统最为成熟，且应用广泛。这些国外主流分布式文件系统普遍代码规模较大，且各自有各自的诞生背景，项目耦合度较强。若针对特定需求做定制化性能优化，工作量较大。

跟国外的分布式文件系统发展比起来，国内分布式文件系统的起步较晚。但由于淘宝，京东等国内互联网公司的兴起和业务的不断扩大，对专有的业务海亮数据存储需求不断提高，催生了一批性能优异的分布式文件系统。其中典型的代表有淘宝的TFS和轻量级开源的FastDFS等。

TFS主要针对淘宝中海量的商品描述图片等小文件存储的需求，以及海量的非结构化数据，他构筑在普通的Linux机器集群上，可为外部提供高可靠和高并发的存储访问。为淘宝提供海量的小文件存储，通常文件大小不超过1M，被广泛地应用于淘宝各项应用中。它采用了HA架构和平滑扩容，保证了整个文件系统的可用性和扩展性。同事扁化了数据组织结构，可将文件名映射到文件的物理地址，简化了文件的访问流程，一定程度上为TFS提供了良好的读写功能。是一个高可扩展、高可用、高性能面向互联网服务的分布式文件系统。

FastDFS是一个完全由国人开发的轻量级分布式文件系统，整体框架采用C语言编写，支持Linux，FreeBSD，AIX等Unix系统，并全部开源。它对文件进行统一管理，实现功能有：文件存储，文件同步，文件上传与下载，并具有动态扩容功能。该文件系统特别适合以文件为载体的在线服务网站。

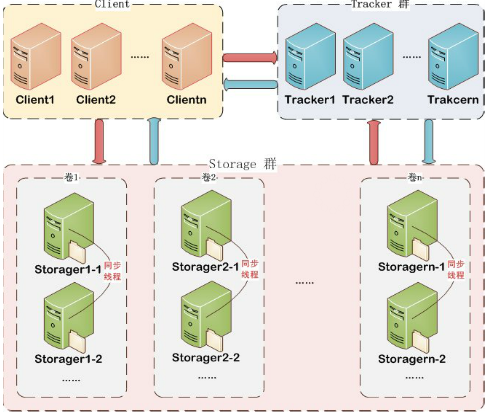
|  |
| --- |
| 决策事项1 |
| 决策事项 | 选择通用式的分布式文件系统还是选择专用式分布式文件系统 |
| 备选方案 | 1. 通用式分布式文件系统 |
| 1. 专用式分布式文件系统 |
| 决策选取方案 | 2 |
| 选取理由 | 从系统分类上讲，目前的分布式文件系统分为两类，一类是通用式文件系统，从使用角度来讲类似传统的文件系统，应用端可以对应的mount到一个挂载点上使用。典型的代表有前文中介绍的lusture,MooseFS等等。另一类是专用分布式文件系统，这类系统文件上传后不能修改，需要专门的API对其进行访问，典型代表有MogileFS、HDFS，TFS，FastDFS等等。下表从开发者友好性、系统复杂性和系统性能对两大类系统进行比较：    从表格特性中我们可以看出，专用分布式文件系统系统的有性能较高，系统的复杂性较低等优点，符合我们当前的需求。但是同时也存在开发者友好性较低，这就需要我们对其API进行良好的封装，以方便开发者使用 |
| 决策事项2 |
| 决策事项 | 通用式分布式文件系统选型 |
|  | 1. HDFS |
| 技术备选方案 | 1. TFS |
| 1. FastDFS |
| 决策选取方案 | 3 |
| 选取理由 | 1. 对于三款最主流的专用分布式文件系统HDFS，TFS和FastDFS，下面表格以是否存在索引服务器，海量小文件支持性能，大文件并行存储性，代码量级等角度进行横向对比。     从上表可以看出，相对于HDFS和TFS，FastDFS具有摆脱索引服务器束缚的优良架构，同时对小文件性能支持较好，整个系统是轻量级代码。   1. 安装配置对比   TFS的体验不如FastDFS  TFS较为复杂，尤其在稍微高版本的GCC下就编译不通过，如我们现网所使用的Centos6.4。FastDFS不存在这种问题。   1. 版本更新对比   TFS版本更新速度奇慢，部分模块如TFS Nginx有两年没有更新了。   1. 资源消耗对比   TFS比FastDFS多。   1. 空间分配对比   TFS采用了块，需要预先分配所以存储空间不能动态的增加占用磁盘  FastDFS没有块的概念，不存在一开始空间占用高问题。   1. 社区活跃度对比   FastDFS社区活跃度高，用户众多，实战案例丰富。  TFS 社区较冷清，相关学习资料较少。  对于使用中调优有帮助。  总得来说，相比于TFS，FastDFS更轻巧，就目前平台需求来看更合适。 |

# 组件介绍

● **简介**

FastDFS是由阿里巴巴公司资深架构师余庆设计并完成的一个开源的轻量级[分布式文件系统](http://baike.baidu.com/view/771589.htm)，它是由纯C语言实现的开源轻量级分布式文件系统，可以运行在Linux，FreeBSD等UNIX架构操作系统之上，它对[文件](http://baike.baidu.com/view/345685.htm)进行管理，[功能](http://baike.baidu.com/view/587727.htm)包括：文件存储、文件同步、文件访问（文件上传、文件下载）等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服务，如相册网站、视频网站等等。对于文件的访问采用专用的API完成，目前API支持C，JAVA，PYTHON，PHP等语言。它是典型的基于文件的键值存储系统，特别适合互联网应用，具有较高的性能和扩展性。国内互联网公司使用较多，有京东商城，支付宝，UC，飞信，赶集网，迅雷，58同城，51CTO等等二十余家，其中最大的一家某网盘服务商部署了400个group，总体文件存储量达到了6PB，文件数量超过1亿。由此可见FastDFS的商业成熟性和健壮性。

* **体系架构**

****

图片十三 FastDFS的逻辑架构图

如图所示，FastDFS集群中主要有两种角色，一种是跟踪服务器，另一种是存储服务器。其中跟踪服务器起到全局调度作用，它在内存中记录集群中所有的存储服务器的状态信息。它是连接客户端的存储服务器的桥梁。由于设计的精巧性，跟踪服务器的性能非常高，在上百台存储服务器的大型集群中，配置三个跟踪服务器就足以满足系统的需求。存储服务器的主要作用是存储文件和文件的相关信息。

FastDFS的服务器集群中不需要存储文件的索引信息，这是其最大的优势。所有的服务器角色都是对等的，并不会出现主从关系。同一个组内的所有存储服务器中存储的文件都是一样的，可以保证数据的冗余备份性。不同组之间的服务器不会相互通信，而且组内的存储服务器直接和跟踪服务器之间沟通，在日常工作中，跟踪服务器之间也不会相互通信。这种垂直结构可以在很大程度上节省网络资源，并方便系统的维护。

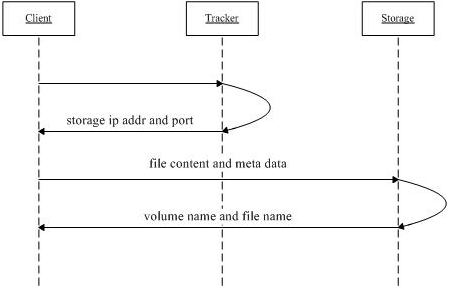
* **工作过程**

[文件上传](http://www.icultivator.com/tag/%e6%96%87%e4%bb%b6%e4%b8%8a%e4%bc%a0)**：**

1. Client询问Tracker server上传到的Storage server；

2. Tracker server返回一台可用的Storage server，返回的数据为该Storage server的IP地址和端口；

3. Client直接和该Storage server建立连接，进行文件上传，Storage server返回新生成的文件ID，文件上传结束。

****

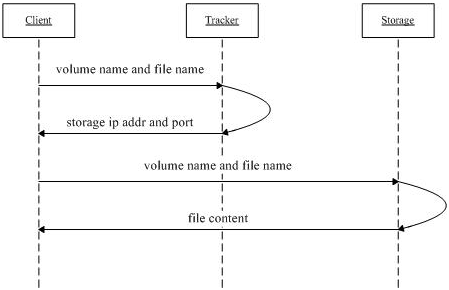
图片十四 文件上传时序图

[文件下载](http://www.icultivator.com/tag/%e6%96%87%e4%bb%b6%e4%b8%8b%e8%bd%bd)：

1. Client询问Tracker server可以下载指定文件的Storage server，参数为文件ID（包含组名和文件名）；

2. Tracker server返回一台可用的Storage server；

3. Client直接和该Storage server建立连接，完成文件下载。

****

**图片十五 文件下载时序图**

# 详细设计

## 整体逻辑结构

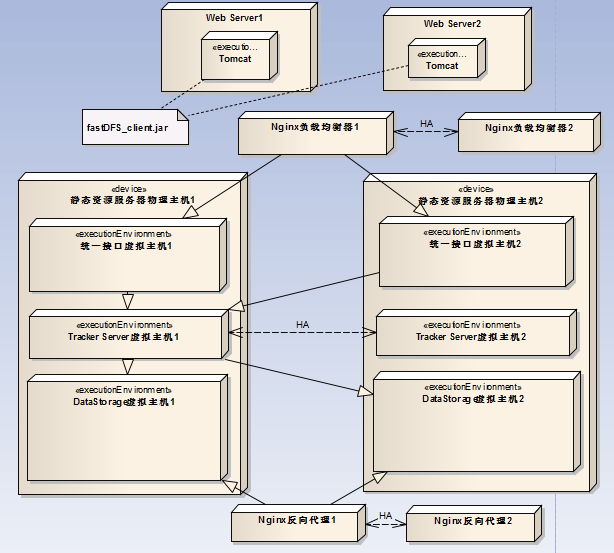


图片十六 整体规划图

1. 从整个系统的逻辑来看，系统有如下节点：
2. CDN加速节点，当重点厂商的用户访问我们的静态资源时，会首先获取最近的CDN节点的缓存，这样可以提高了重点厂商用户的页面展示效率，提高了用户的体验。同时由于CDN节点对于使用频度高的静态资源进行了缓存，源服务器的访问压力将会有所降低，解决了源服务器带宽窄的难题。
3. Nginx反向代理采用HA架构，提高了Nginx的稳定性。同时Nginx反向代理可以根据实际访问URL来区分出静态资源是否需要缓存。
4. FastDFS的存储节点，可以根据实际的文件存储量进行调整，可以做单机也可以做集群，避免了服务器资源的空置浪费。
5. FastDFS采用HA提高可用性，负责将配置平台提交的静态资源文件上传到存储节点上。由于TrackerServer启动需要耗费的资源少，所以直接和DataStorage直接部署于一台机器上，同样避免服务器资源的闲置占用。

## 实际部署结构

由于H5需求的不断变化，静态资源量大小也在不断变化，系统在设计时考虑易用性、扩展性、高效性等因素，采用相对轻巧的架构，避免服务器资源的闲置浪费。当前阶段采用精简的部署结构，同时为后期留有灵活扩展的能力。



图片十七 现阶段实际部署元素

1. 为考虑目前静态资源文件的实际数量以及保证服务的高可靠性，用两台物理主机来部署平台的所有元素，包括统一接口节点元素、跟踪服务器节点元素(Tracker Server)、存储节点元素(DataStorage)等，根据FastDFS的特性，后期静态资源量加大可以不断的动态添加存储节点。
2. 元素节点之间的关系，Nginx服务器节点和TrackerServer目前都是采用HA架构，但是具体的HA架构选型带确认。
3. 统一接口、存储节点（DataStorage）采用LB（负载均衡策略）来维持系统的稳定性。
4. 各业务的配置平台访问提交静态文件资源可通过我们二次开发封装的接口API导入后，直接接口调用，jar包基本不依赖其它的三方jar，使用简介方便，不对原有应用程序构成冲击。

## 接口API规范

* **配置分布式文件系统连接信息**

/\*\*

\* 配置分布式文件系统连接信息

\*

\* @param connectTimeOut

\* 连接超时的时限，单位为毫秒

\* @param networkTimeOut

\* 网络超时的时限，单位为毫秒

\* @param antiStealToken

\* 防盗链Token

\* @param charSet

\* 字符集

\* @param secretKey

\* 密钥

\* @param trackerHttpPort

\* Tracker服务器HTTP服务端口

\* @param trackerServers

\* Tracker服务器IP端口列表

\*/

public FastDFSManager(int connectTimeOut, int networkTimeOut,

boolean antiStealToken, String charSet, String secretKey,

int trackerHttpPort, InetSocketAddress[] trackerServers);

* **开启数据库记录开关**

/\*\*

\* @description 开启数据库记录开关

\* @author ranxu3

\* @create 2014年12月18日上午11:09:49

\* @version 1.0

\* @param url

\* 数据库地址

\* @param user

\* 数据库用户名

\* @param password

\* 数据库密码

\* @param dbDriver

\* 数据库驱动

\* @return boolean true:开启成功 false:开启失败

\*/

public boolean openRecordSwitch(String url, String user, String password,

String dbDriver);

* **关闭数据库记录开关**

/\*\*

\* @description 关闭数据库记录开关

\* @author ranxu3

\* @create 2014年12月18日上午9:06:27

\* @version 1.0

\*/

public void closeRecordSwitch();

* **根据本地文件路径上传文件**

/\*\*

\* @description 根据本地文件路径上传文件

\* @author ranxu3

\* @create 2014年12月15日下午4:09:15

\* @version 1.0

\* @param fileLocal

\* 本地文件绝对路径

\* @param fileName

\* 文件名称

\* @param fileExtName

\* 文件扩展名

\* @return String 文件上传后的远程地址，由文件组名称+文件名称组成。

\*/

public String saveFile(String fileLocal, String fileName, String fileExtName);

* **根据文件字节码上传文件**

/\*\*

\* @description 根据文件字节码上传文件

\* @author ranxu3

\* @create 2014年12月15日下午4:09:15

\* @version 1.0

\* @param fileByte

\* 文件字节码

\* @param fileName

\* 文件名称

\* @param fileExtName

\* 文件扩展名

\* @return String 文件上传后的远程地址，由文件组名称+文件名称组成。

\*/

public String saveFile(byte[] fileByte, String fileName, String fileExtName);

* **根据文件字节码裁剪上传文件**

/\*\*

\* @description 根据文件字节码裁剪上传文件

\* @author ranxu3

\* @create 2014年12月15日下午4:09:15

\* @version 1.0

\* @param fileByte

\* 文件字节码

\* @param offset

\* 截取开始索引

\* @param length

\* 截取长度

\* @param fileName

\* 文件名称

\* @param fileExtName

\* 文件扩展名

\* @return String 文件上传后的远程地址，由文件组名称+文件名称组成。

\*/

public String saveFile(byte[] fileByte, int offset, int length,

String fileName, String fileExtName)

* **根据远程文件路径获取文件字节码**

/\*\*

\* @description 根据远程文件路径获取文件字节码

\* @author ranxu3

\* @create 2014年12月11日上午10:56:37

\* @version 1.0

\* @param remoteFileLocal

\* 远程文件路径

\* @return byte[] 远程文件字节码

\*/

public byte[] getFileByte(String remoteFileLocal);

* **根据远程文件路径获取文件信息**

/\*\*

\* @description 根据远程文件路径获取文件信息

\* @author ranxu3

\* @create 2014年12月11日上午10:56:37

\* @version 1.0

\* @param remoteFileLocal

\* 远程文件路径

\* @return FileDetail 远程文件信息

\*/

public FileDetail getFileInfo(String remoteFileLocal);

* **远程文件写入本地磁盘**

/\*\*

\* @description 远程文件写入本地磁盘

\* @author ranxu3

\* @create 2014年12月17日上午9:26:52

\* @version 1.0

\* @param remoteFileLocal

\* 远程文件路径

\* @param fileLocal

\* 本地文件路径

\* @return boolean true:写入成功 false:写入失败

\*/

public boolean fetchFile(String remoteFileLocal, String fileLocal);

* **远程文件写入输出流**

/\*\*

\* @description 远程文件写入输出流

\* @author ranxu3

\* @create 2014年12月17日上午9:26:56

\* @version 1.0

\* @param remoteFileLocal

\* 远程文件路径

\* @param output

\* 输出流

\* @return boolean true:保存成功 false:保存失败

\*/

public boolean fetchFile(String remoteFileLocal, OutputStream output);

* **根据文件路径删除远程文件**

/\*\*

\* @description 根据文件路径删除远程文件

\* @author ranxu3

\* @create 2014年12月11日上午10:56:59

\* @version 1.0

\* @param remoteFileLocal

\* 远程文件路径

\* @return boolean true:删除成功 false:删除失败

\*/

public boolean unlinkFile(String remoteFileLocal);

## 接口API

