凤凰新媒体
Phoenix New Media

基础技术体系规划

2011 年

目录

本文档修订记录

版本	修订时间	修订人	修订内容
V1.0	2011-10	金明岩、李毅	初稿

1. 目的和要求

本文在对凤凰新媒体现有的 IT 技术体系进行梳理的基础上，明确一个中远期的基础技术体系规划，进而保障长期技术演进符合整体业务发展的需要。在明确规划的前提下规避开发行为中不符合架构趋势的方向错误，减少重复性的开发工作;明确软件系统的分类和最终部署关系；明确软件系统的边界和相关接口方式，进而建立一个规范统一的技术体系。

1.1 开发敏捷性要求

互联网业务和用户规模在不断的发展，这种发展有时已经超过了产品经理和架构师的预期，于是越来越多“敏捷”、“轻量级”等快速可以提交结果的开发方式被引入产品开发的过程。这些思路在需要高效协同的开发组织的同时，也对基础技术体系，尤其是基础 IT 架构提出了更高的要求。如何通过架构的合理以及基础产品体系的完善提供一个便于高效开发的场景，成为了越来越多互联网公司系统设计人员的主要工作任务。

提供高效的技术架构，可以使得业务系统的技术架构更“单纯”，即只需要考虑某些方面（Aspect）的问题，减少设计过程中设计人员所面对场景的思维复杂度；使得设计过程可以充分忠实于 SOC（关注点分离）的软件设计原则；同时技术架构可以将一些公共的业务抽象为线上已有的服务，采用 SaaS（Software-as-a-service）的方式构建线上的服务平台，使得业务系统的开发可以直接使用这些能力。

1.2 运维高效性要求

随着业务和用户规模的快速扩展，面对大量的服务器的运维场景，将超出纯粹人工维护可以接受的成本要求和复杂程度。

面对这些可能的困难，运维活动要求统一的系统和技术架构应该具有以下的属性：

1 服务器有效的分类。

2 每个分类的服务器所需要的运行环境要尽量的单纯。

3 有效的负载均衡无单点故障的调度体系。

4 几乎全部的故障都可以自动完成故障切换，运维人员只需要在事后修复硬件故障。

5 自动的安装部署体系，无需频繁的登录服务器，而是通过配置管理软件来完成系统的安装、维护和升级。

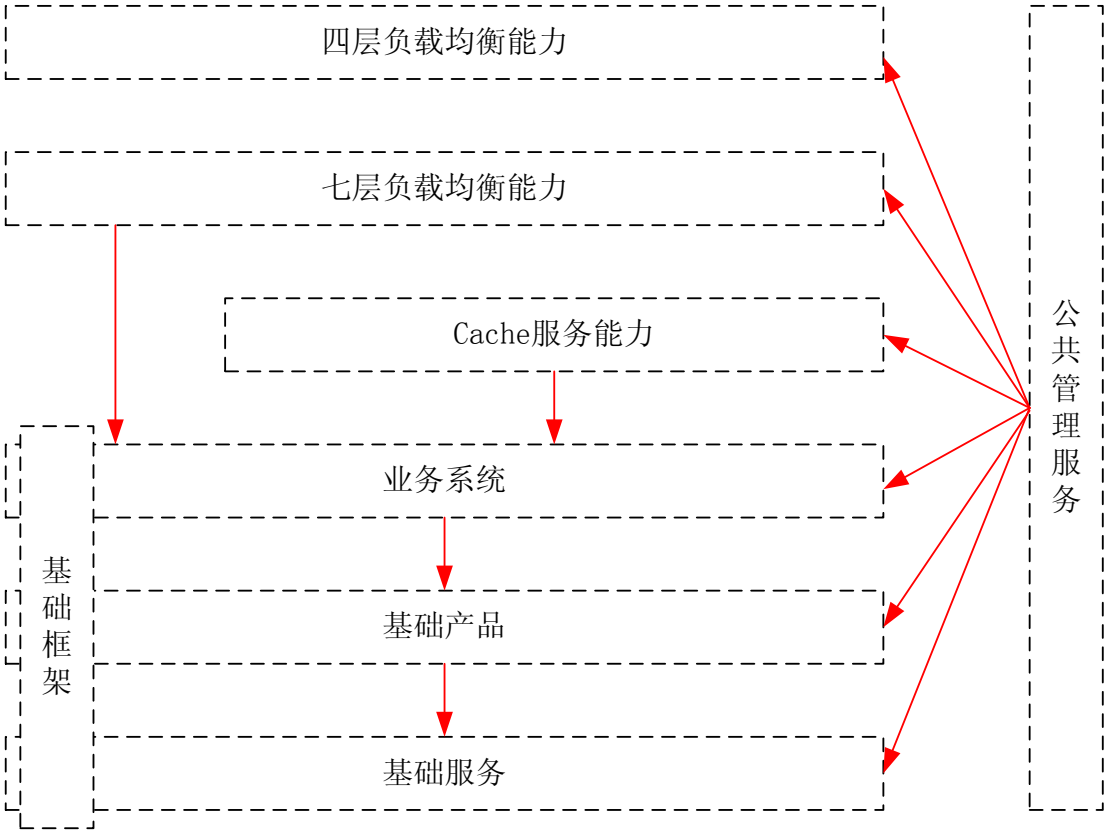
6 有效的监控体系，让故障自动呈现出来，无需定期关注和检查各种服务。

以上的这些要求，无论是靠行政执行，管理流程，还是软件服务去解决，都需要在技术架构的角度给予有效的规划，明确各种行为所负责的边界。

2. 基础技术体系蓝图

本章节将针对凤凰新媒体的技术体系做一个明确的分类，进而得到软件产品和项目的分类，明确其所负责的边界，明确不同分类之间的接口和调用方式。

2.1 图示



2.1 附图1 - 技术体系蓝图

2.2 说明

在上图中，我们明确了以下几种类别的软件服务及其部署能力：

1 基础服务。

2 基础产品。

3 公共管理服务

4 业务系统。

5 《基础框架》。

2.2.1 基础服务

基础服务是指提供基础能力的服务平台，其本身是一个部署态应用服务，但一般不提供某种业务能力，而是为完成某种应用提供某一个层次和某一方面问题的解决能力。基础服务及组件是提高开发效率的最有效手段，它可以让开发一个业务产品时不需要，或者很少的需要考虑某一个层次或方面的问题，使得业务系统能更关注本身的业务实现。

目前在互联网应用中，数据和内容的存储是迫切需要的基础服务，在下面的章节中将针对基础服务展开，明确基础服务所包含的内容。

2.2.2 基础框架

基础框架是指一些基础开发工具，本身是一个编码级的框架，在业务系统的设计和编码阶段引入业务系统中，与业务系统的程序一并部署并完成业务系统的业务能力，主要的目的是降低编码的复杂度和结构复杂性。

基础框架针对的并非一个部署场景，只是会影响开发活动和一个软件产品本身的结构，所以在本文中不会影响线上的部署能力。

2.2.3 基础产品

基础产品是指一些公共的业务产品，以业务服务的形式部署在线上，供需要此能力的其他业务系统调用。

基础产品本身将被部署为一个服务（Service），该服务提供某种业务能力，业务系统可以使用此服务来完成业务系统的相关功能。

2.2.4 公共管理服务

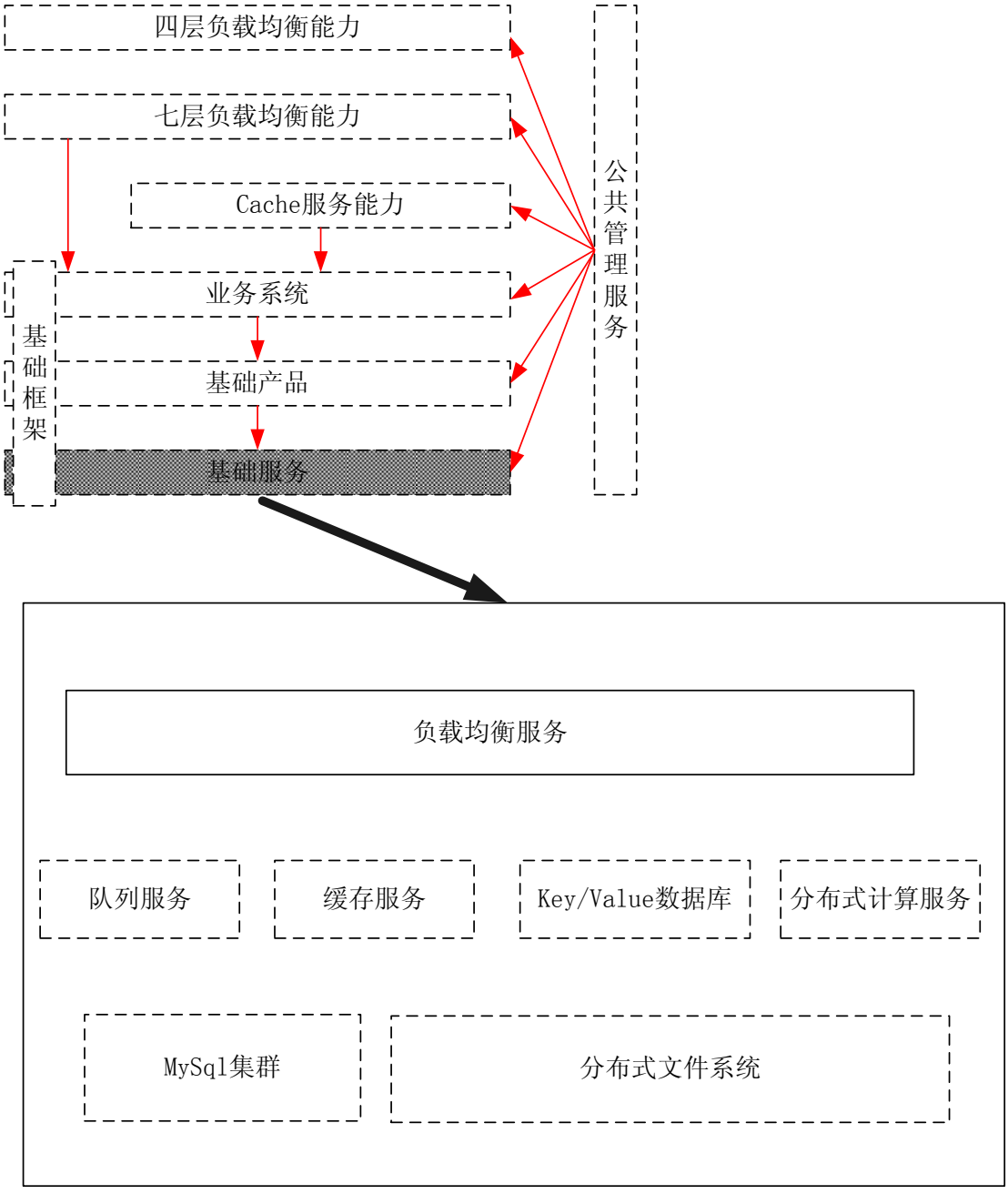
公共管理服务是指一些面向内部管理和维护的服务能力，这些服务提供了统计分析、监控、管理等相关能力，一般被内部的业务和运维人员使用，用来了解系统运行的情况和各种数据，用以决策和分析。

公共管理服务将提供一个全网管理的平台能力，所有的软件服务都需要向公共管理服务负责，进而保障软件服务本身可部署、可统计、可监控、可管理。

3. 基础服务规划

3.1 蓝图

3.1.1 图示



3.1.1 附图1 基础服务蓝图

3.1.2 子系统说明

基础服务着力于提供各种存储方面的能力。其中：

a) 分布式文件系统

主要用于存储海量文件，业务系统可以在分布式文件系统存储用户数据文件，也可以存储自身需要分布式能力的配置文件。

b) MySql集群

主要用户所有需要关系数据库存储数据的场景，将提供一个统一的数据库集群，业务系统将不再需要安装独立的数据库，而是用数据库集群提供的数据存储能力。

c) 分布式计算服务

提供公共的 Map/Reduce 能力，业务系统可以使用此能力完成需要的分布式计算而不需要独立安装和部署类似的服务。

d) Key/Value数据库

提供统一的 Key/Value 存储及相关域管理能力，完成业务系统需要的相应存储能力。

e) 缓存服务

提供一个有结构数据的缓存能力，以便业务系统可以管理一个分布式缓存，使得部署于不同服务器上的业务系统可以使用一个公共的缓存服务来完成相互之间的数据交换，提供一个类数据总线能力。

f) 队列服务

提供一个公共队列能力，使用 put get 的接口方式完成不同组件或不同服务之间的数据通讯和传递能力。

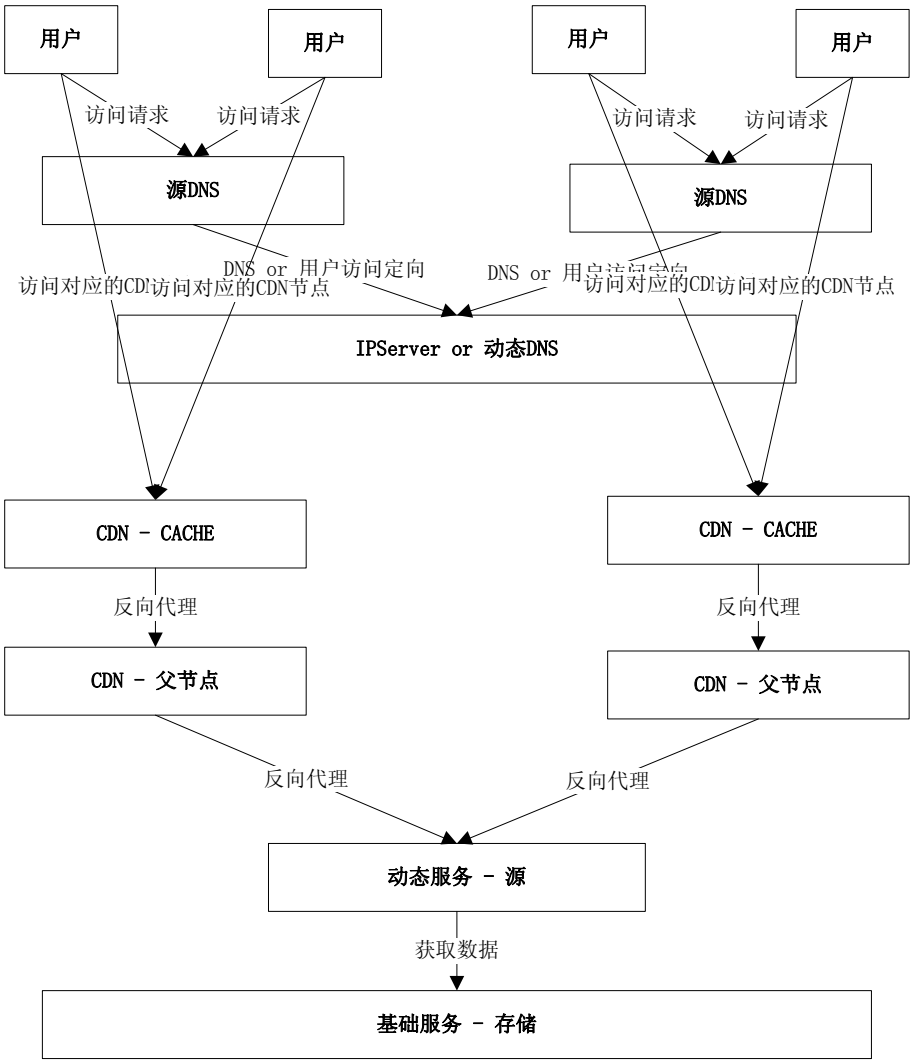
3.2 跨机房部署

因为基础服务中存储了业务系统的数据，故其需要考虑业务系统部署的变化，进而提供适应这种变化的部署能力。

3.2.1 可静态化的跨机房能力

对于静态服务可以考虑直接使用 CDN 系统,利用其负载均衡能力实现用户体验的最佳化。

如下图所示：



3.2.1 附图1 跨机房部署静态方式访问示例

步骤如下：

1 用户访问首先到达本地的 DNS.

2 如果采用 DNS 负载均衡,则到达 CDN 系统中的动态 DNS 服务，动态 DNS 根据用户的源 DNS 确认其位置，并将解析后的距离用户最近的 Node 对应的 Cache – VIP 地址.

如果采用 IPSERVER 完成负载均衡，则用户直接会访问 ipserver，然后被重定向到距离用户最近的 Cache-VIP 地址。

3 用户访问对应的 Cache 服务。

4 Cache 本地存在对应的数据时，则直接为用户提供服务，否则反向代理到父节点。

5 父节点存在数据则直接为用户提供服务，否则反响代理到源的动态服务。

6 源动态服务从基础服务中取得数据，然后返回给父节点。

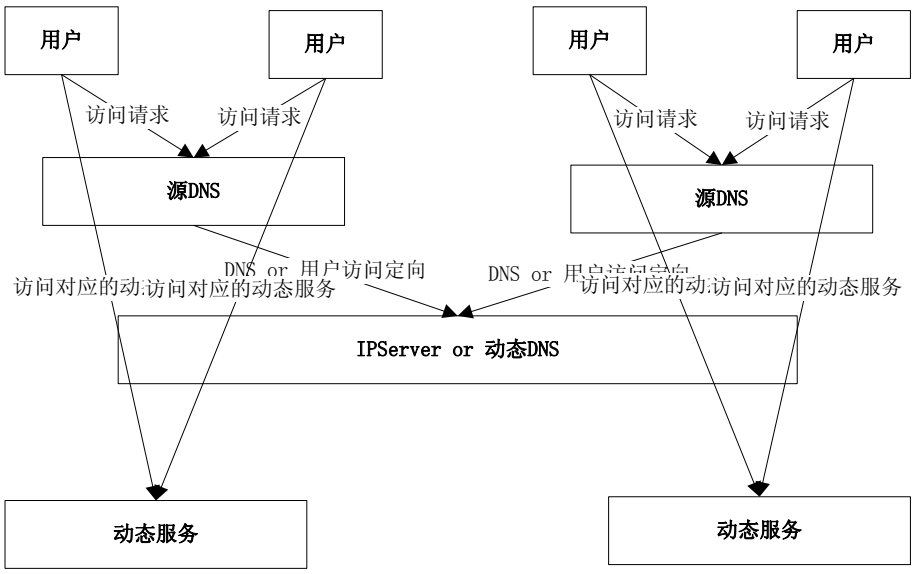
适用于静态内容的场景常见于较大数据量的用户访问行为，如访问一个图片、视频等。这些动态服务本身由于最终返回的链接可以转化为静态的，所以可以采用 CDN 完成用户请求的负载均衡，

故动态服务本身的压力可控且不会很大，所以可以部署在一个中心的互通性较好的机房中，然后由中心机房的一套基础服务为其提供数据存储服务。

3.2.2 动态化的跨机房能力

业务系统另外一种常见的部署场景由于要考虑其对应服务用户的网络体验，将业务系统的动态服务部署于距离用户较近的位置。由于这种服务提供的大多是一些动态能力，比如社交服务、用户校验、数据校验等，这些能力无法静态化运行在 CDN 中，所以就需要将这些动态服务部署在距离用户较近的机房中。

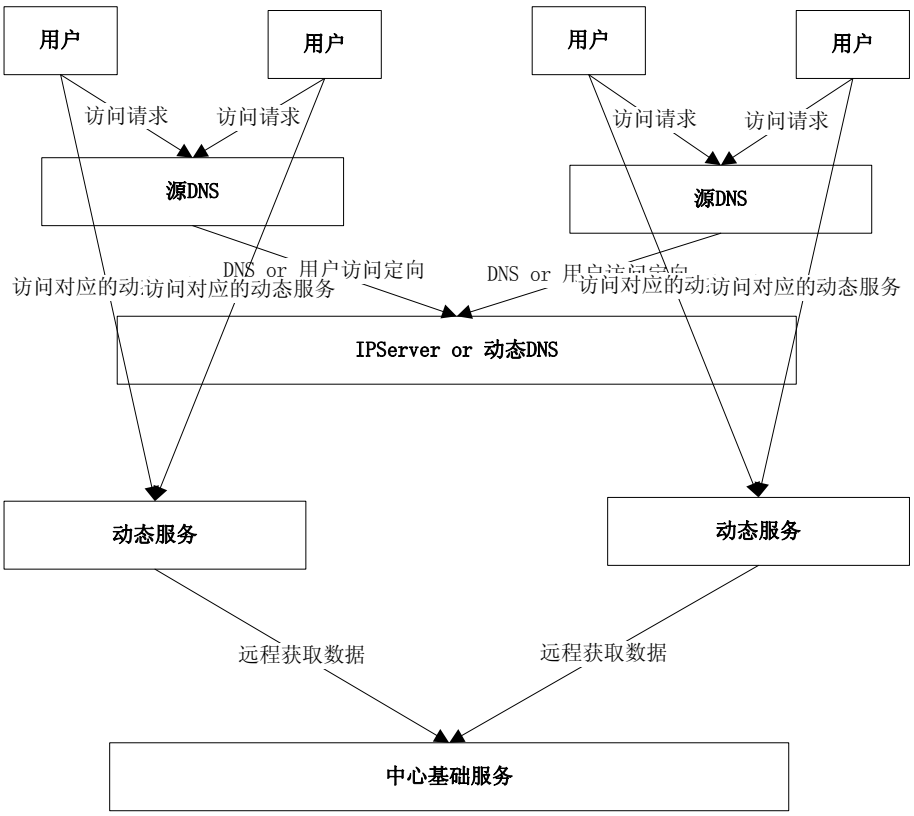
如下图：



3.2.2 附图1 跨机房部署动态方式业务系统访问示例

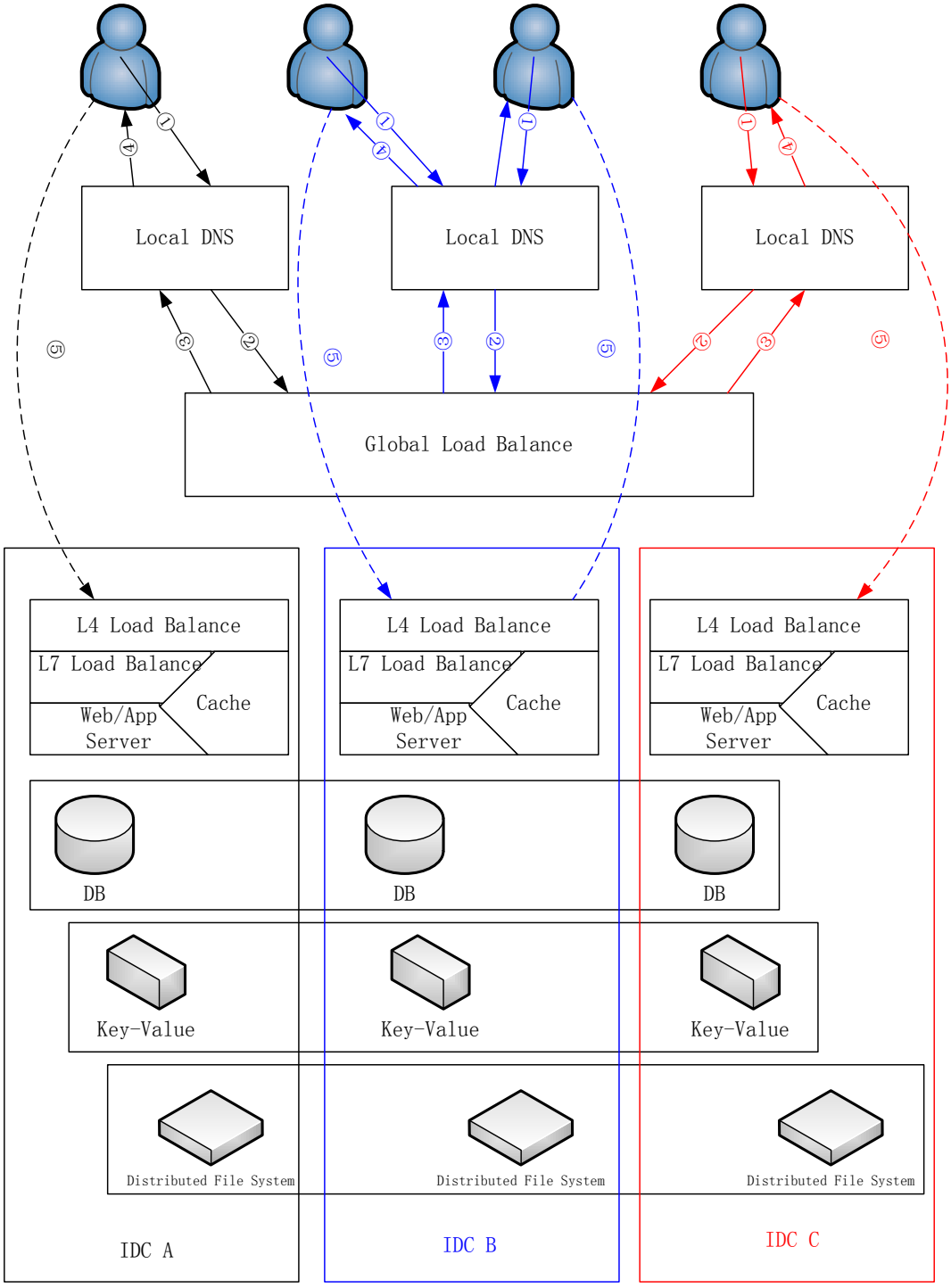
在上述场景中，如果动态服务采用基础服务提供的数据存储能力，在数据量和访问量不大的情况下，也可以将基础服务部署在中心的一个互通性较好的机房中，提供统一的存储能力。

如下图：



3.2.2 附图2 跨机房部署动态方式中心服务能力

在数据量或者访问量较大时，动态服务本身的体验要求就需要基础服务距离自己的位置更近，可以使用“本地化”的基础数据存储为用户提供可靠的服务质量。这就需要基础服务考虑跨机房部署问题。

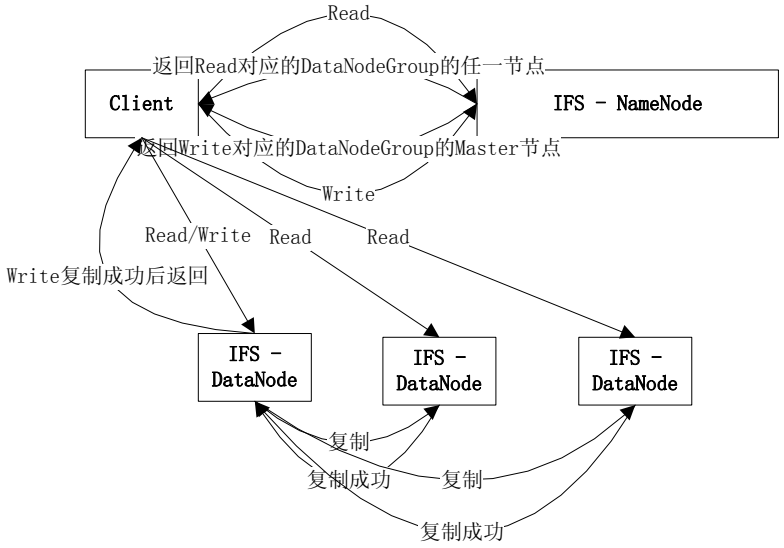


3.2.2 附图3 跨机房部署动态方式

我们在基础服务跨机房时，引入针对所有基础服务类型的统一的负载均衡服务。

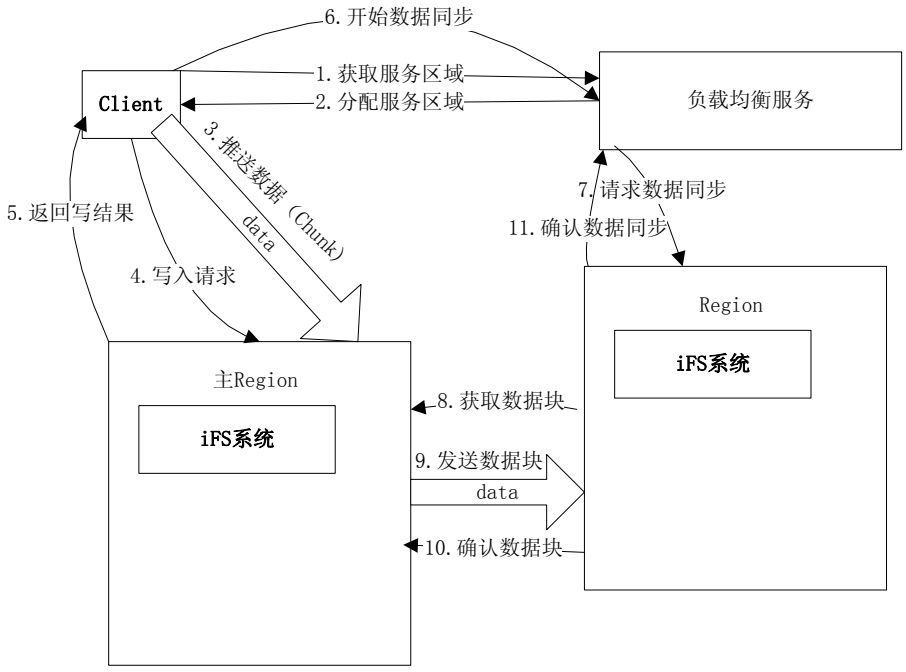
针对数据负载均衡和复制的问题，我们将其分为两个场景。

以分布式文件系统为例。分布式文件系统本身提供一个强一致性的负载均衡及数据复制体验：



3.2.2 附图4 基础服务内部数据复制（IFS）示例

负载均衡服务在现有的分布式文件系统本地强一致性的前提下，提供一个跨部署的远程负载均衡和数据复制能力，但数据复制提供的一致性为“最终一致性”。



3.2.2 附图5 跨机房最终一致性数据复制示例

如上图，步骤为：

1 client 调用负载均衡服务，询问其对应的本地 IFS 系统。

2 负载均衡服务计算其位置，返回其对应的服务区域调用地址。这个地址在 client 端可以长期

缓存，因为 client 端为一个动态服务，其部署的位置是固定的。

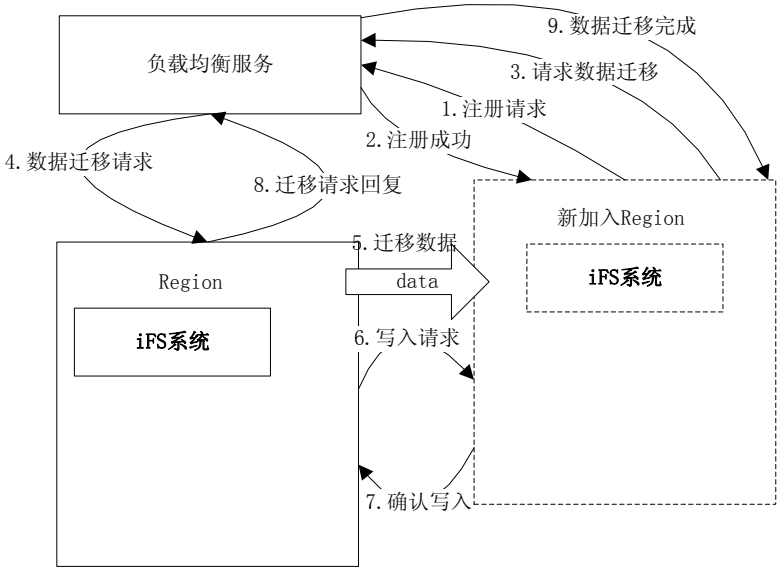
3 如图 3 4 5 步骤所示，client 完成向本地系统的数据写操作（读操作无需数据同步）。

4 如图 6 步骤所示，完成以后 client 需要报告负载均衡服务一个数据同步的消息。

5 负载均衡服务发起数据同步，要求远程需要同步的其他子系统向对应最优路径的系统请求数据，完成数据同步操作，如图 8 9 10 11 步。

6 当数据同步失败时，负载均衡服务只能尝试重试，如果最终重试失败，需要将已经同步完成的数据设置为失效的。以便业务系统访问时可以感知。

当在一个新的位置部署新的基础服务子系统时，需要完成一个数据同步操作，其大致的流程如下图：



3.2.2 附图6 跨机房新增基础服务部署数据复制示例

3.3 分组及分域

3.3.1 分组及分域的要求

在基础服务提供的服务能力规划中，所有的业务系统会共用了一套基础服务。这样基础服务本身就需要考虑如下几个问题：

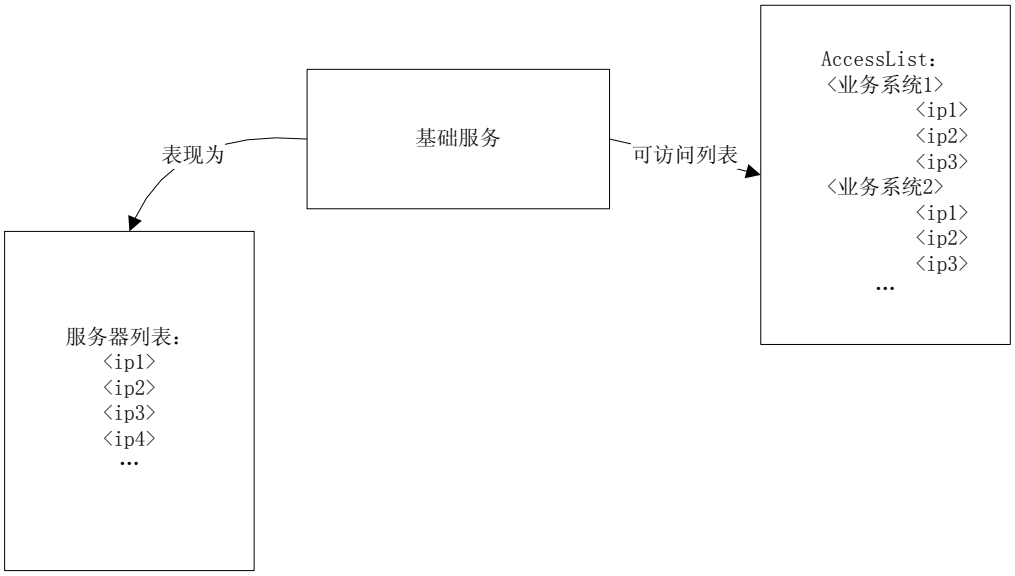
1 如何做到基础服务数据访问的安全性，即只允许经过授权的机器访问基础服务。

2 在基础服务内部数据的访问互斥性，即某一个业务只能访问业务自己的数据，而不能访问其他业务的数据。

3 业务系统内部以及业务之间不同的数据定义者定义的数据最大限度的避免可能的各种“碰撞”，如 Key/Value 中的不同业务系统生成一样的 key。

3.3.2 访问安全性

由公共管理服务-配置管理系统提供统一的管理数据，该数据描述针对某一个基础服务系统（最终细化到部署态的服务器上），都有哪些 ip 地址可以访问基础服务能力对应的通信端口。

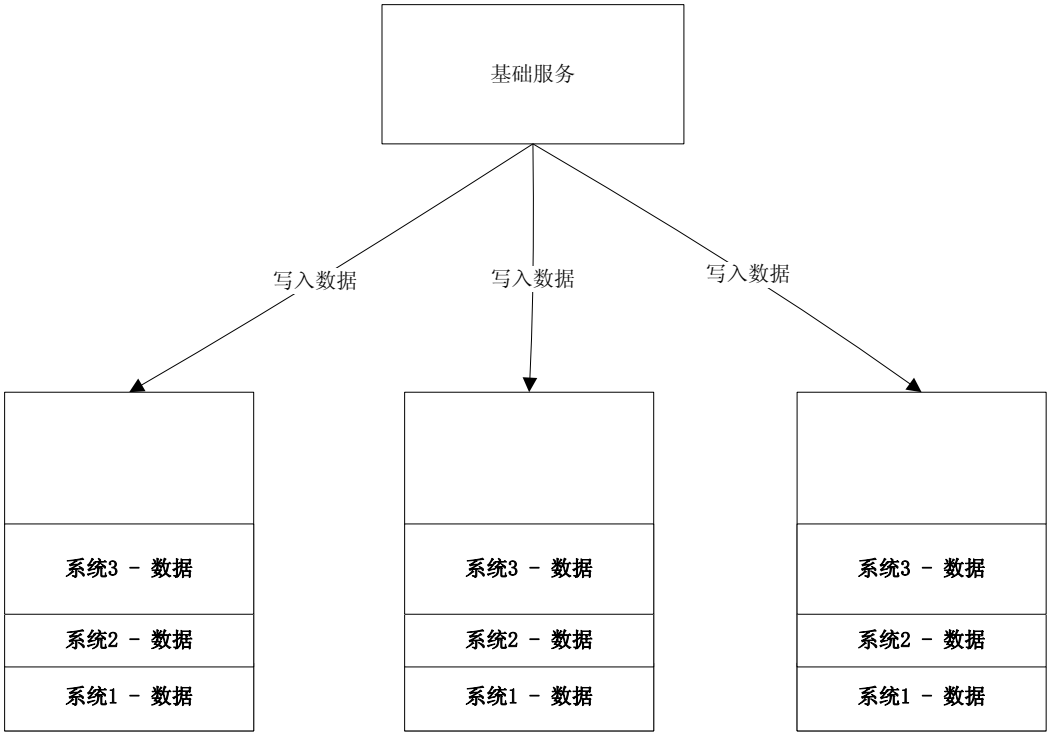


3.3.2 附图1 访问安全性配置表

配置创建或变更后，配置系统会下发此列表到基础服务对应的服务器上，基础服务服务器上的防火墙将按照此列表配置。这样就可以保障只有经过授权的服务器才可以访问对应的基础服务。

3.3.3 数据互斥性

对于一个业务系统来说，其数据会被基础服务分散存储到基础服务归属的各个服务器中（MySql 集群除外）。这样对于基础服务的一个存储空间来看，其存储的数据将属于不同的各种的业务系统，最终通过数学期望来保障所有的基础服务对应的存储单元的存储空间消耗基本一致。

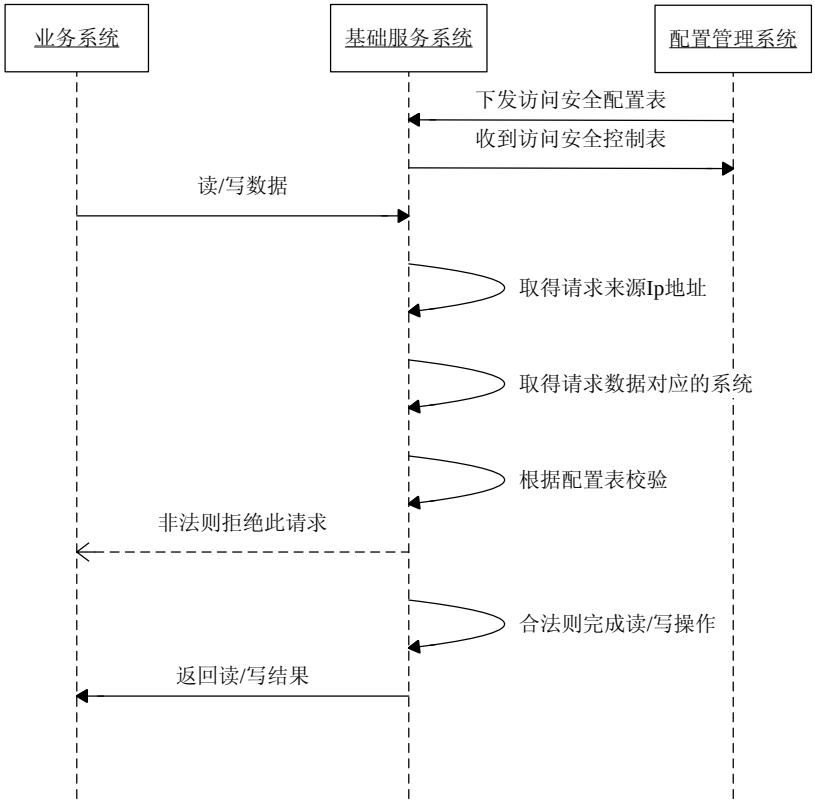


3.3.3 附图1 数据分布示例

在上文描述的业务逻辑的前提下，基础服务本身需要根据 3.3.2 中的“访问安全配置表”来控制业务系统的访问行为即：

只允许对应业务系统 ip 地址来源的请求访问对应的业务系统数据。

如何约束和识别不同的业务系统将在 3.3.3 中说明。在可识别业务系统的前提下，完成下面的时序：

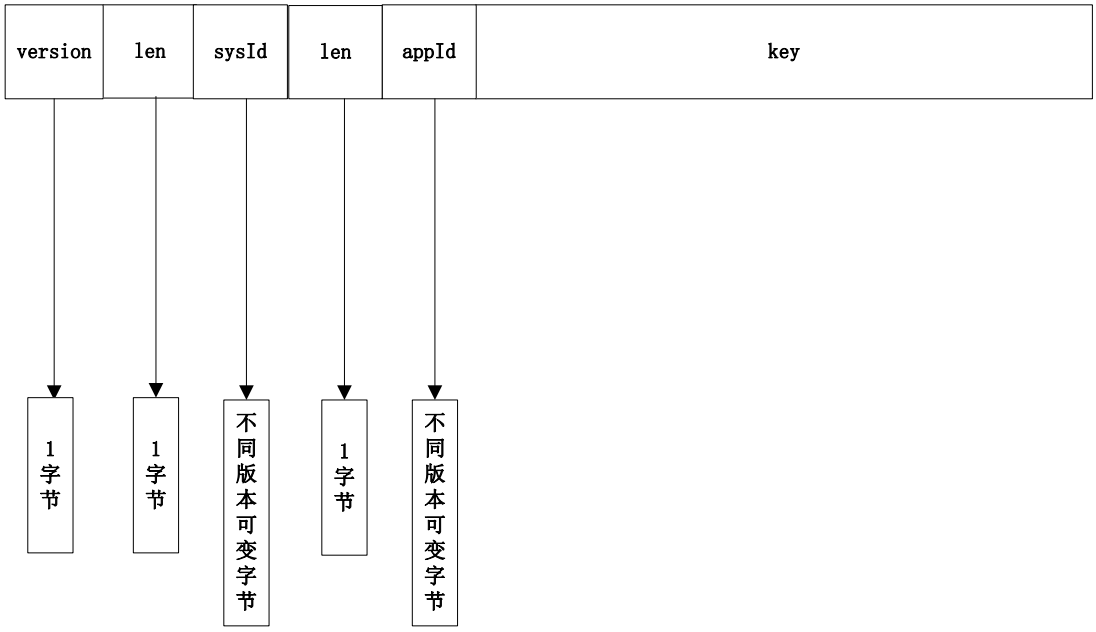


3.3.3 附图2 数据互斥时序

3.3.4 数据编码约束

为了最大限度的避免不同业务系统或者业务系统内部的不同组件在在使用基础服务时出现数据冲突，我们对基础服务涉及到的数据做出一些编码的约束。

对于所有可以隐含为 Key 值的访问数据（如分布式文件系统中的文件名称、Key/Value 存储中的 Key 值、队列服务中的队列标识）均采用以下编码：



3.3.4 附图1 数据编码规则

其中：

Vesion 为 1 个字节，存放此编码约束对应的版本，从 1 开始。每次变更+1.

首个 len 字段长度为 1 个字节，放入 sysId 的长度，在版本中给予长度固定的约束，如在版本 1 中 sysId 长度为 2.

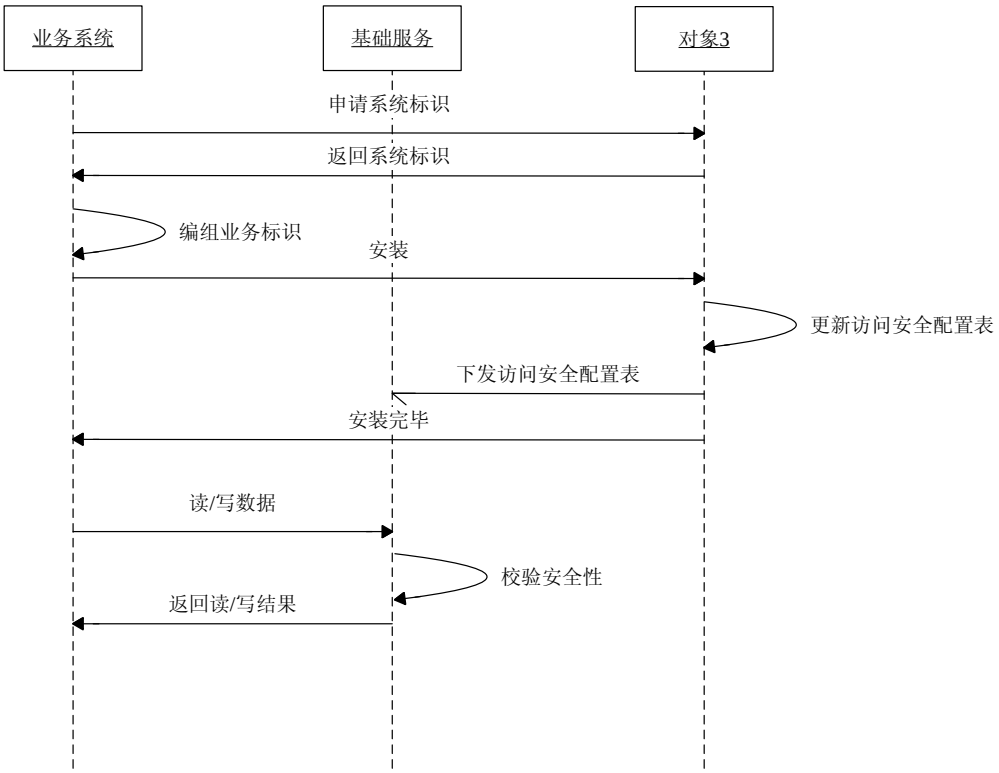
sysId 放入系统标志，每个业务系统均有一个固定的系统标志。在配置管理系统中申请获得。

第二个 len 字段长度也为 1 个字节，放入 sysId 对应的业务组件的 id，由业务系统自己定义，但长度为版本本身的约束。

AppId 字段放入组件标志，该标志长度固定，由业务系统自行确定。

Key 字段放入真正的业务 key 值。

数据编码具体的使用时序如下：



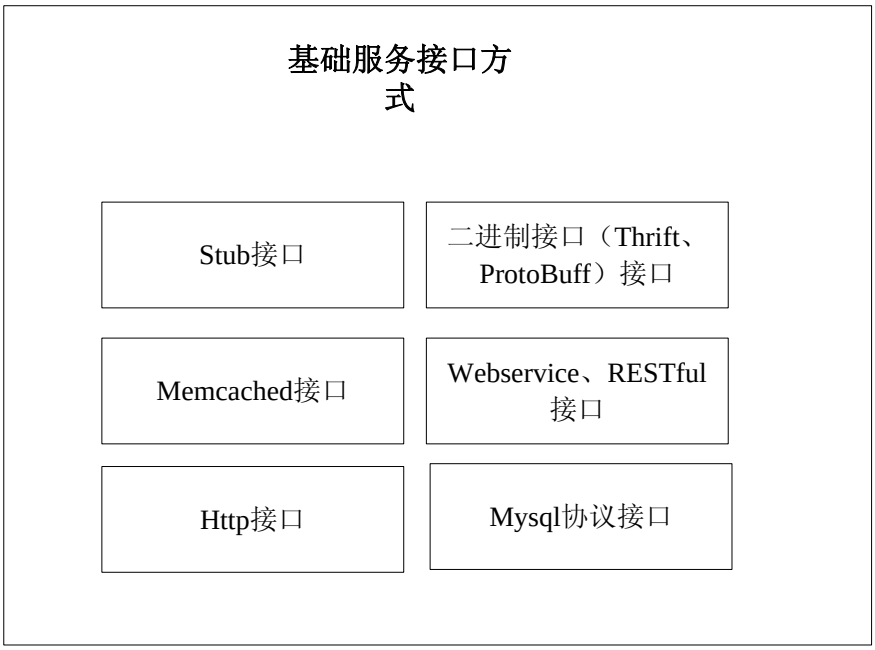
3.3.4 附图2 数据编码使用时序

在上图的“读/写数据”调用中，业务系统传递数据编码给基础服务系统，基础服务通过数据编码中的信息校验安全性。

3.4 接口能力

基础服务要最终被上层的基础产品、业务产品以及公共管理服务调用，故基础服务需要开发相应的接口能力。

基础服务可以提供如下几种接口能力：



3.4 附图1 基础服务接口类别

Stub 接口：Stub 接口使用在分布式文件存储、Key/Value 存储等服务系统；支持 RMI 等远程调用协议，需要在调用端引用基础服务相应的 Stub 库进行远程访问或调用。

Memcached 接口：可以使用在缓存系统和 Key/Value 存储中，调用者可以通过 Memcached 调用直接访问基础服务。

Http 接口：基础服务提供 Http 接口，它具有易于访问、方便调试等优点，但访问效率较低。

二进制（ProtoBuff，Thrift）接口：跨语言级的二进制访问协议接口，使用在分布式文件存储、键值存储等服务系统中。会在基础框架中采用类似 Gearman 等调用机制进行包装。

Webservice、RESTful 接口：通用的 SasS 调用方式，由于序列化和反序列化代价较大，访问效率不如上述的接口方式。

Mysql 协议接口：用于 Mysql 数据库访问以及 Mysql 代理访问的接口方式。

3.5 平滑扩展能力

3.5.1 分布式文件、键值系统等横向扩展

跨机房部署参考 3.2，在同一个区域里系统的扩展方式为增加服务器，注册到 Master 时将采用自有的负载均衡策略和容错机制把其他服务器的“数据”同步到新加入的服务器。

具体的内部负载均衡策略和分组机制不做约束。

3.5.2 Mysql集群横向扩展

a) Mysql的分区策略（Partition）

■ 简介

数据库划分把一个大的数据库分割为更小的数据库集合，每个分区遍及多个服务节点，用户能够在分区上实现本地事务处理。

划分方式：

数据库表、索引、事务日志和数据元素。

水平划分（Shard）：

把不同的行放入不同的数据库表中。

垂直划分：

少量的列和剩余的列分开存储在不同的数据库表中。

■ 划分的统一标准

范围划分：

划分主键值在一定范围之内，例如邮编在 70000 和 79999 之间作为一个划分标准。

队列划分：

划分主键值在一个数值集合内，例如把 Iceland, Norway, Sweden, Finland or Denmark 作为北冰洋国家的一个划分。

哈希划分：

划分主键值为一个哈希函数的值。

复合划分：

上述 3 种方式的组合。

b) Mysql的碎片化策略（Sharding）

■ 简介

水平扩展(Scale Out，亦或横向扩展、向外扩展)的解决方案，其主要目的是为突破单节点数据库服务器的 I/O 能力限制，解决数据库扩展性问题。

数据 Sharding 的策略与分区表的方式有很多类似的地方，有基于表、ID 范围、数据产生的时间或是 SOA 下理念下的基于服务等众多方式可选择。而与传统的表分区方式不同的是，Sharding 策略和业务结合的更为紧密，成功的 Sharding 必须对自己的业务足够熟悉，进行众多可行性分析的基础上进行，"业务逻辑驱动"。

■ Sharding 策略

冷热数据分离：根据业务特点，根据时间范围对主要的业务数据做 Sharding，把不到 10% 的"热"数据有效隔离开来，同时对这部分数据用以更好的硬件，提供更好的用户体验。而另外 90% 的数据因用户很少访问，所以尽管访问速度稍慢一点，对用户来说，影响也很小。

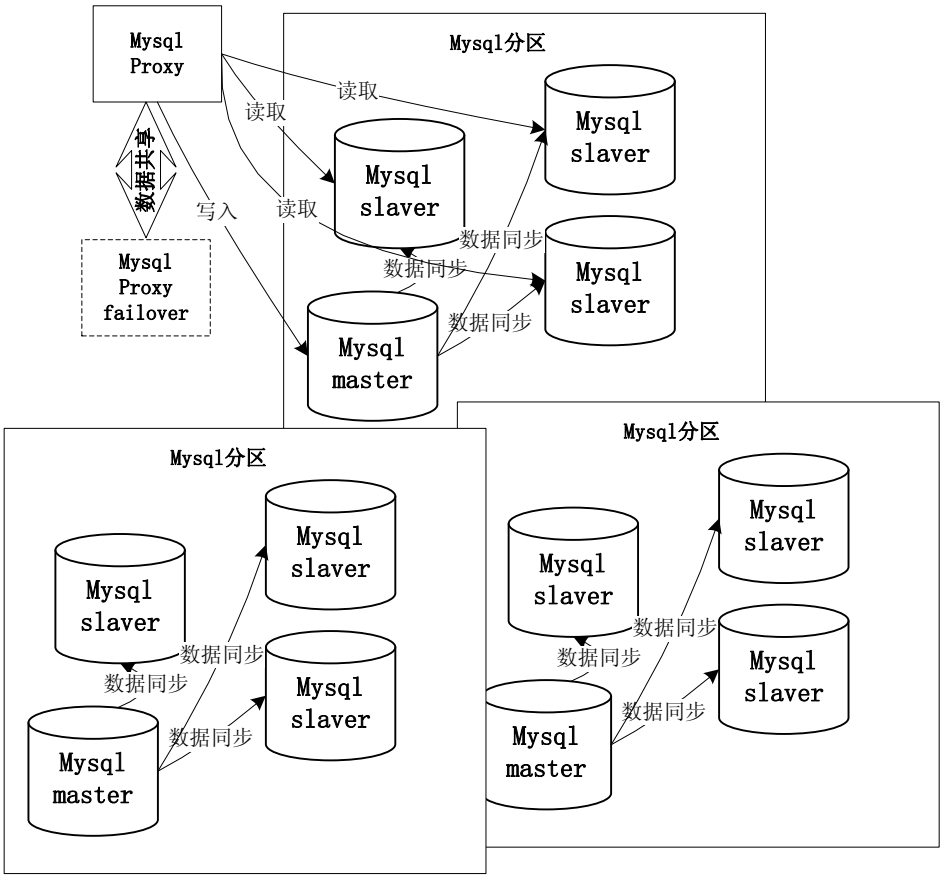
关联数据抽象：采用一定策略抽象数据，抽象出来的数据对象之间的关联数据很小，关联数据尽可能保持到同一分区。

c) Mysql代理

MySQL-Proxy 是处在你的 MySQL 数据库客户和服务端之间的程序，它还支持嵌入性脚本语言 Lua。这个代理可以用来分析、监控和变换（transform）通信数据，它支持非常广泛的使用场景：

■ 负载平衡和故障转移处理

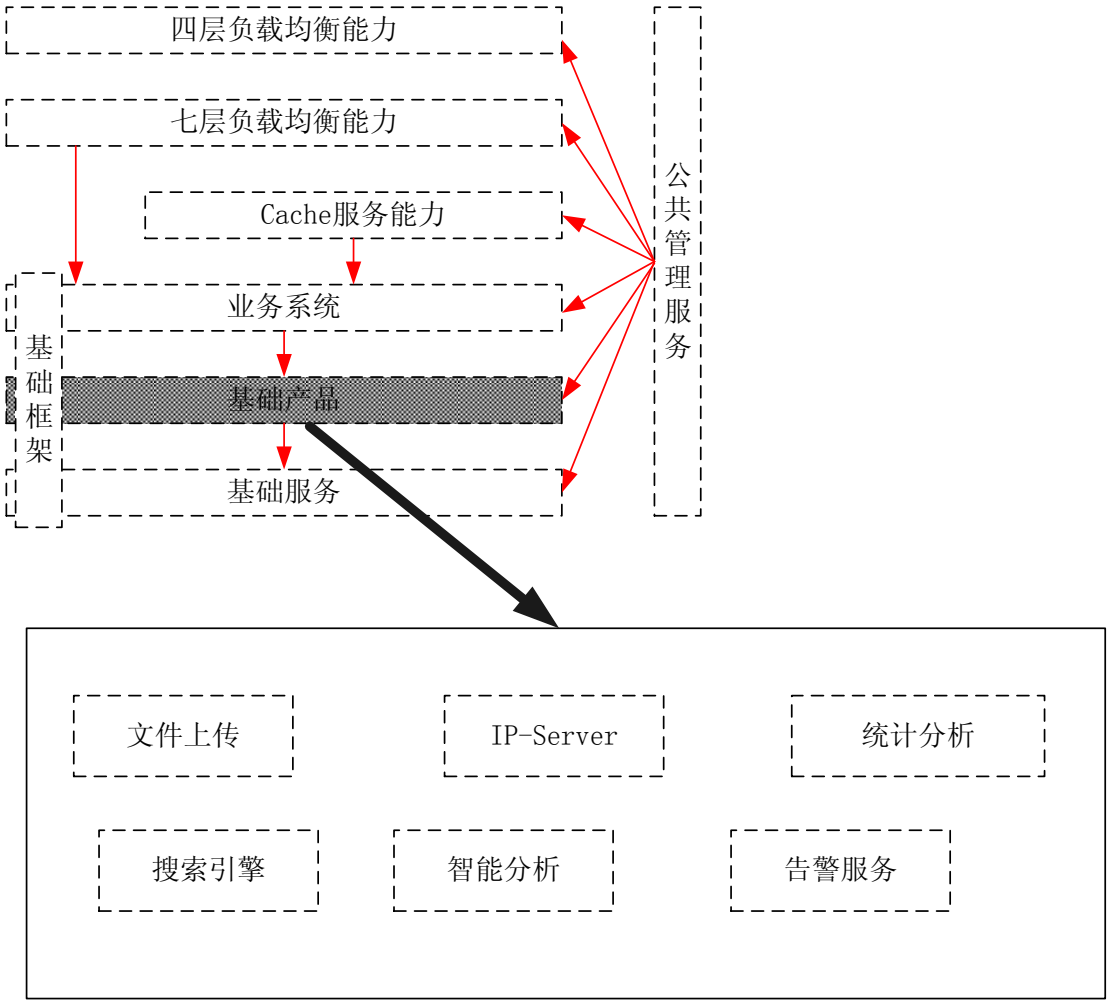
- 读写分离
- 查询分析和日志
- SQL 宏（SQL macros）
- 查询重写（query rewriting）
- 执行 shell 命令



3.5.2 附图 1 MysqlProxy 读写分离

4. 基础产品规划

4.1 蓝图



基础产品主要处理的方面是一些公共的业务能力，这些业务能力将被其他系统引用并完成业务系统的业务能力，也可能独立为一个业务能力，但是独立的业务能力基本不应该面对最终的互联网用户，而是由内部系统或者内部人员来使用。

基础产品会随着业务系统的开发活动需要而扩展，所以在本文不做过多描述。

4.2 负载均衡能力

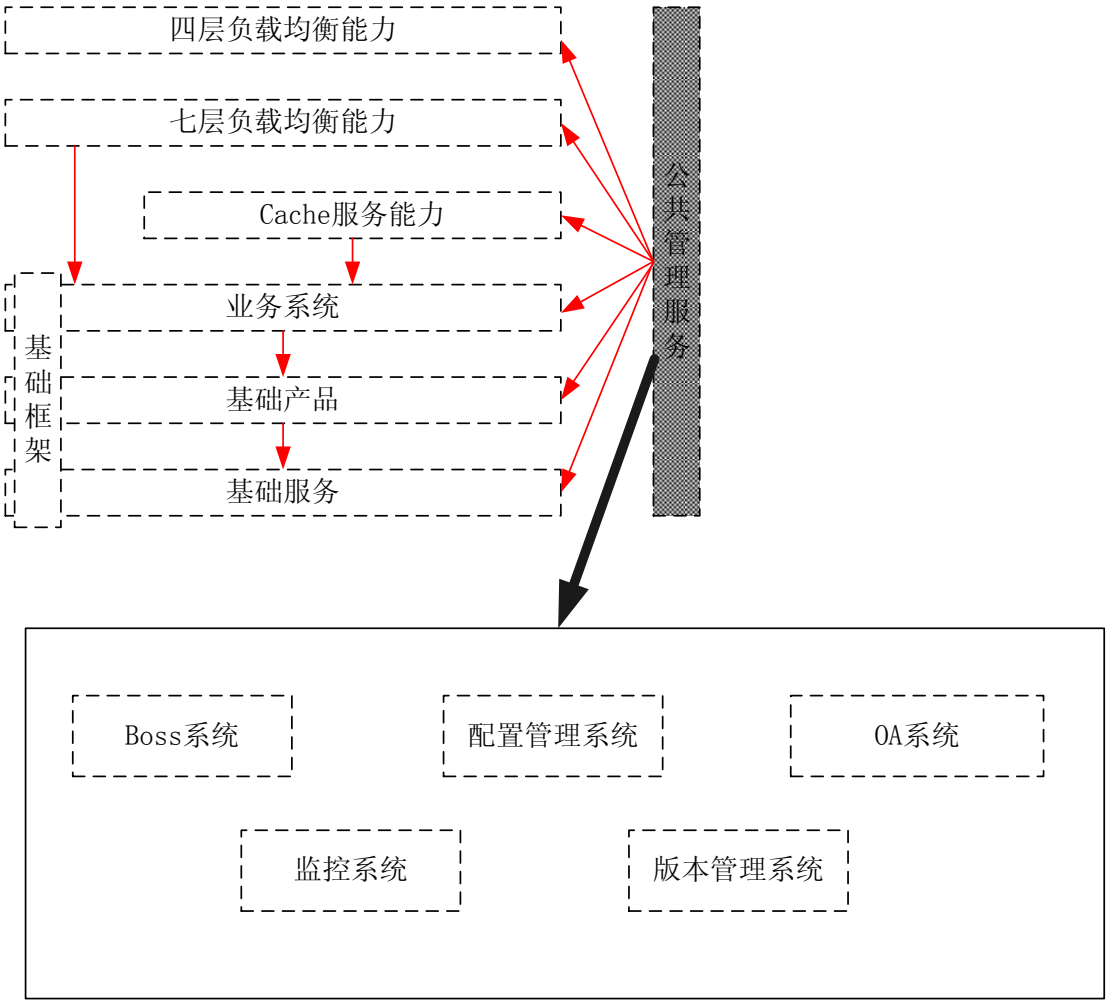
所有的基础产品必须考虑负载均衡模型，达到如下要求：

1 任何组件均无无单点故障。

2 所有的用户访问需要的组件均可平滑扩展。

5. 公共管理服务规划

5.1 蓝图



5.1 附图1 公共管理服务蓝图

5.2 子系统边界

5.2.1 Boss系统

Boss 系统即业务运营支持系统，它融合了业务支撑系统(BSS)与运营支撑系统(OSS)，是一个综合的业务运营和管理平台。我们目前的 Boss 系统中集成了一些基础运营系统，并包含了一部分权限管理能力，未来需要着力于为全网各个子系统提供统一的权限管理的能力。

5.2.2 配置管理系统

配置管理系统管理整个产品开发、测试、安装、运维以及后续的升级流程。在配置管理系统中各个流程环节可追踪；把机器自动化作业与人机交互有效的结合起来。作为一个系统运维平台，主要功能是发布、自动编译、上线及后续的升级；它还对线上所有机器进行状态维护和远程控制。

5.2.3 OA系统

内部员工登录OA 系统，在工作流引擎的支持下，处理各种OA 工作流事务，能够直观的从待办事项和已办事项看到自己的工作内容。OA 系统还能看到自己工作中消息提醒和系统公告。

5.2.4 版本管理系统

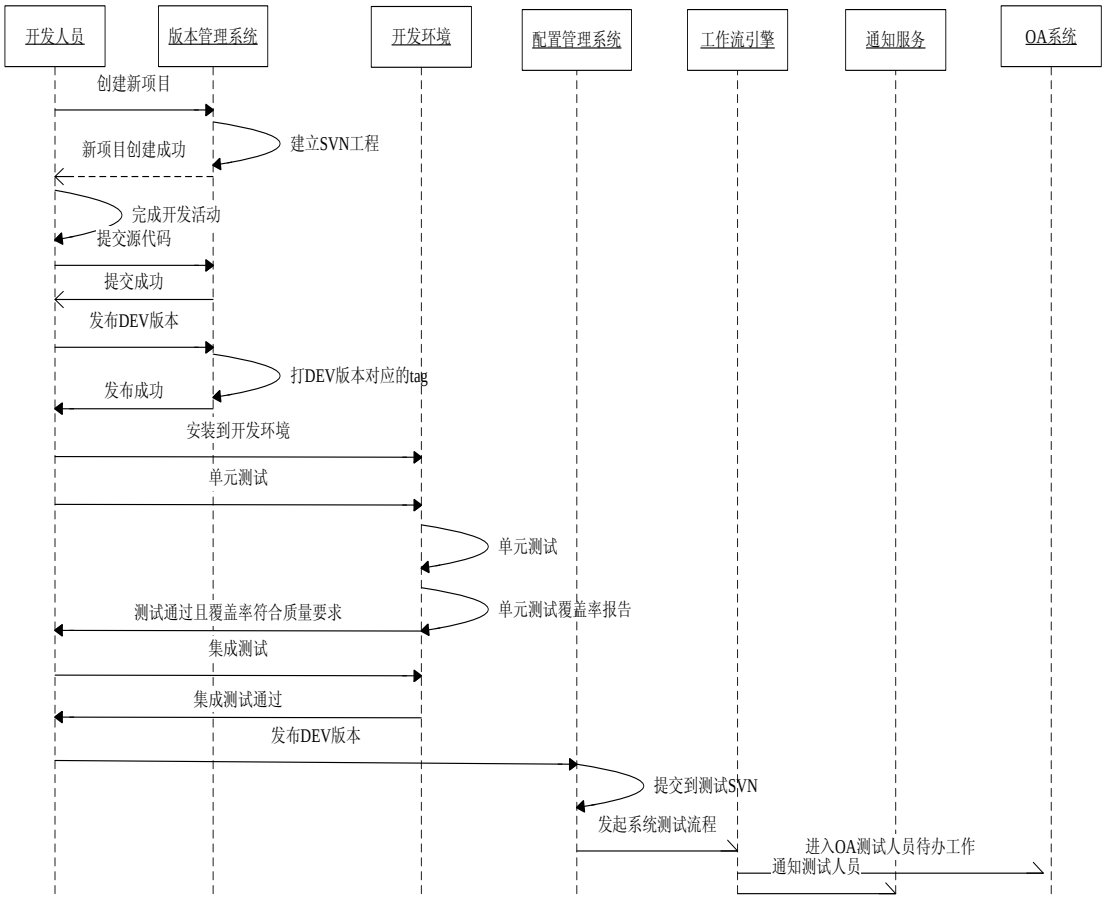
版本管理系统将结合 svn 以及 boss 系统具有的认证能力，通过有效的流程整合软件开发、测试以及上线的各个流程，保障各个流程的交付物可追踪可控制。

5.2.5 监控系统

监控系统是我们对线上服务主机和上线系统的运行状况实时监控分析的运维管理系统。它定义监控指标；监控线上服务器的运行状态；并对线上机器的异常行为进行监控报警和逻辑控制。

5.3 子系统交互示例

5.3.1 开发阶段



5.3.1 附图1 开发范例

在具有通过审核的开发计划的前提下：

1. 开发人员首先通过版本管理系统创建新的开发项目，然后在此项目中完成符合开发计划的开发活动。

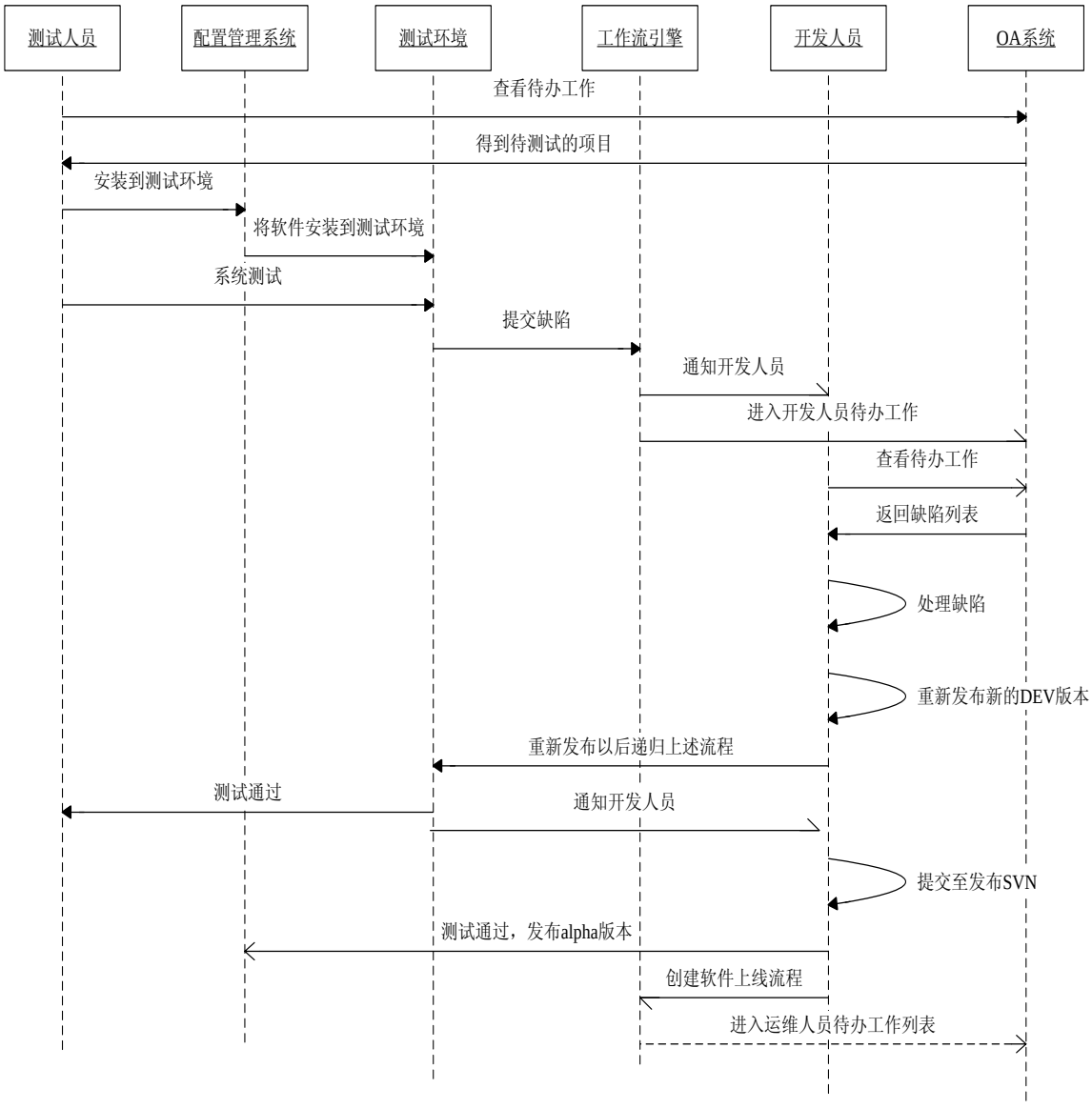
2．完成开发活动后，发布对应的版本计划对应的 dev 版本，dev 版本的命名规则参见**《IFENG-2011-TS-PR-01(凤凰新媒体-2011-技术规范-流程管理规范-No.01_版本发布管理规范)》。**

3. 发布 DEV 版本后，开发人员需要将此版本安装到开发活动内部的开发环境，并进行单元测试和集成测试，其中单元测试需要达到规定的覆盖率目标。

4. 测试通过后，将 DEV 版本对应的安装文件提交到配置管理系统，配置管理系统将此发布物提交到测试的 SVN 上，并通过工作流引擎发起一个测试流程，测试人员将收到通知，并可以在 oa 系统的待办工作列表中查看到对应的测试任务。

5 进入测试阶段。

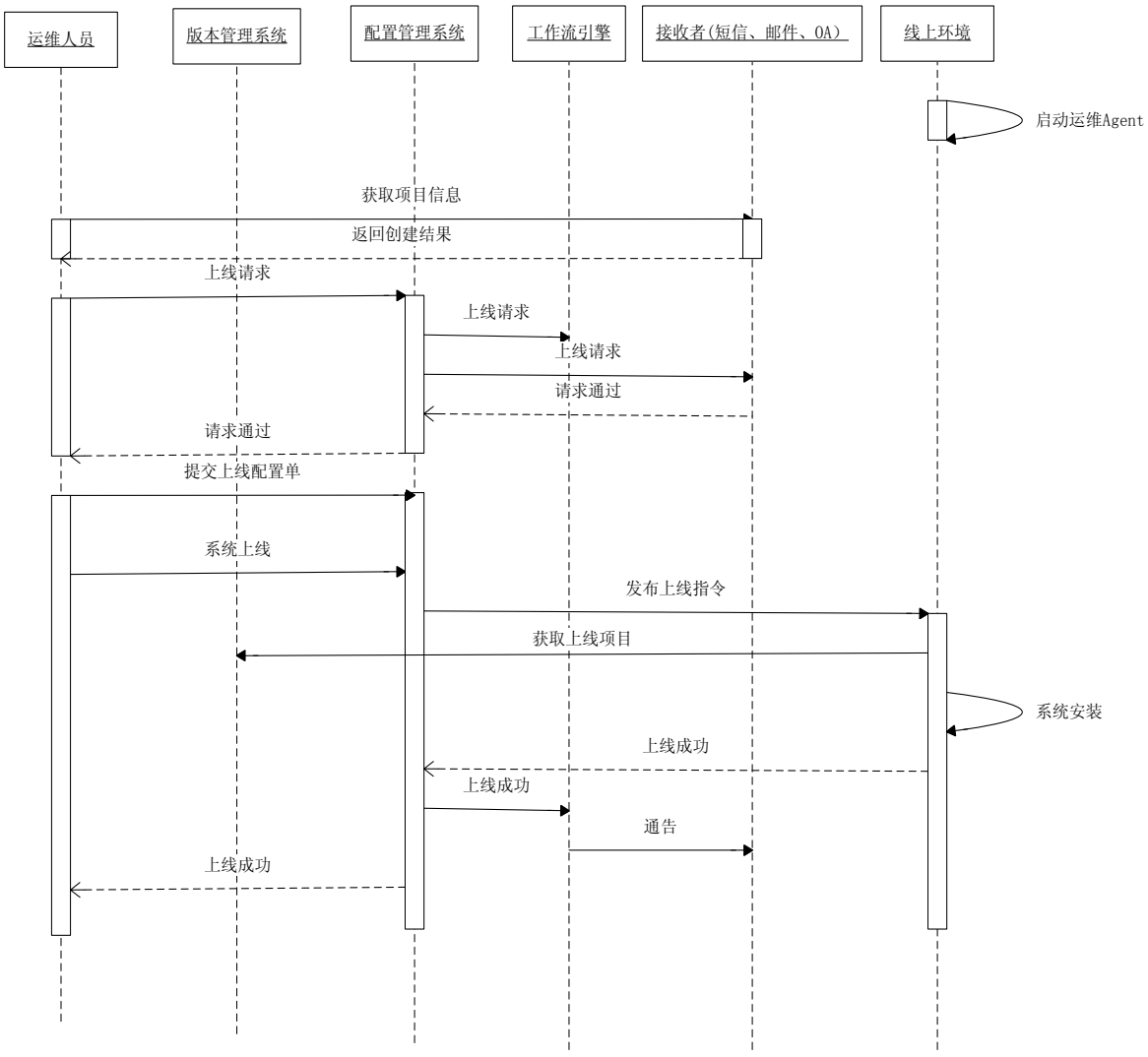
5.3.2 测试阶段



5.3.2 附图1 测试范例

1. 测试人员查询到自己的待办事项，发现一个新的待测试版本。
2. 测试人员通过配置管理系统将软件安装到测试环境。配置管理系统通知部署在测试服务器上的 agent 检出测试 SVN 对应的软件版本，并完成安装和启动的过程。
3. 测试人员在测试环境上进行测试，期间如果发生缺陷，将提交缺陷至缺陷跟踪系统（bugfree）并通过工作流引擎将缺陷放入开发人员的待办事项中。开发人员通过检查自己的待办事项处理属于自己的缺陷，处理完毕以后重新发布 dev 版本，回归上述流程。
4. 测试通过以后，测试人员完成测试工作，并确认测试通过的同时通过工作流引擎将此消息传递给开发人员，进入其待办事项。
5. 开发人员发布 alpha 版本，并创建一个软件上线流程。
6. 软件上线事件进入运维人员待办事项。

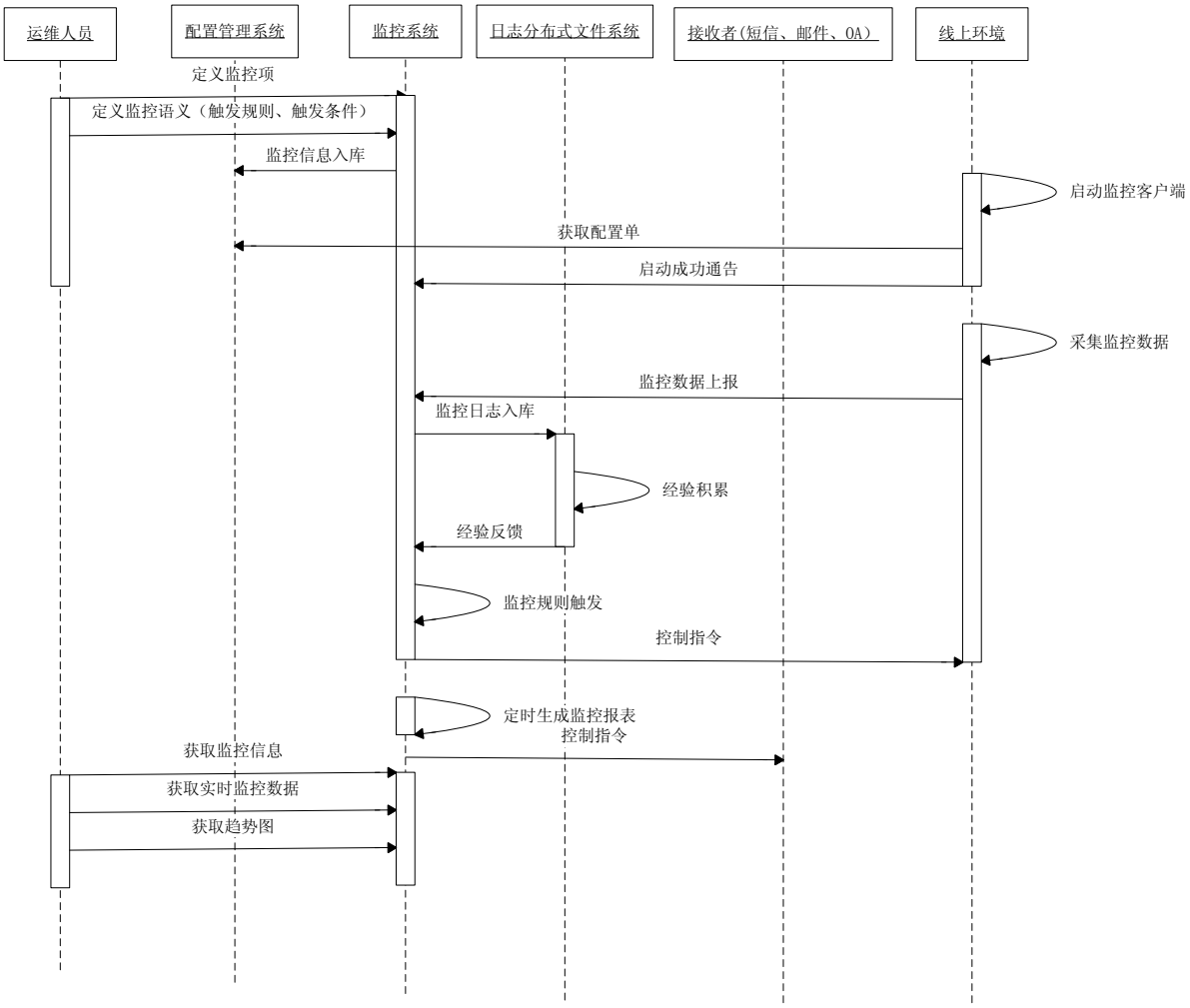
5.3.3 上线阶段



5.3.3 附图1 上线范例

1. 运维从 oa 系统获取待上线的项目名称。
2. 在配置管理系统发送上线请求，待相关领导从 oa 系统中审批完成后提交上线配置到配置管理系统中。
3. 提交系统上线申请至配置管理系统，配置管理系统向线上服务器发送上线指令，线上服务器从版本管理系统中对应的运维 svn 中获取资源软件安装文件，并安装。
4. 安装完毕以后系统上线完毕，通知运维人员，并通过 oa 通知相关人员。

5.3.4 系统监控

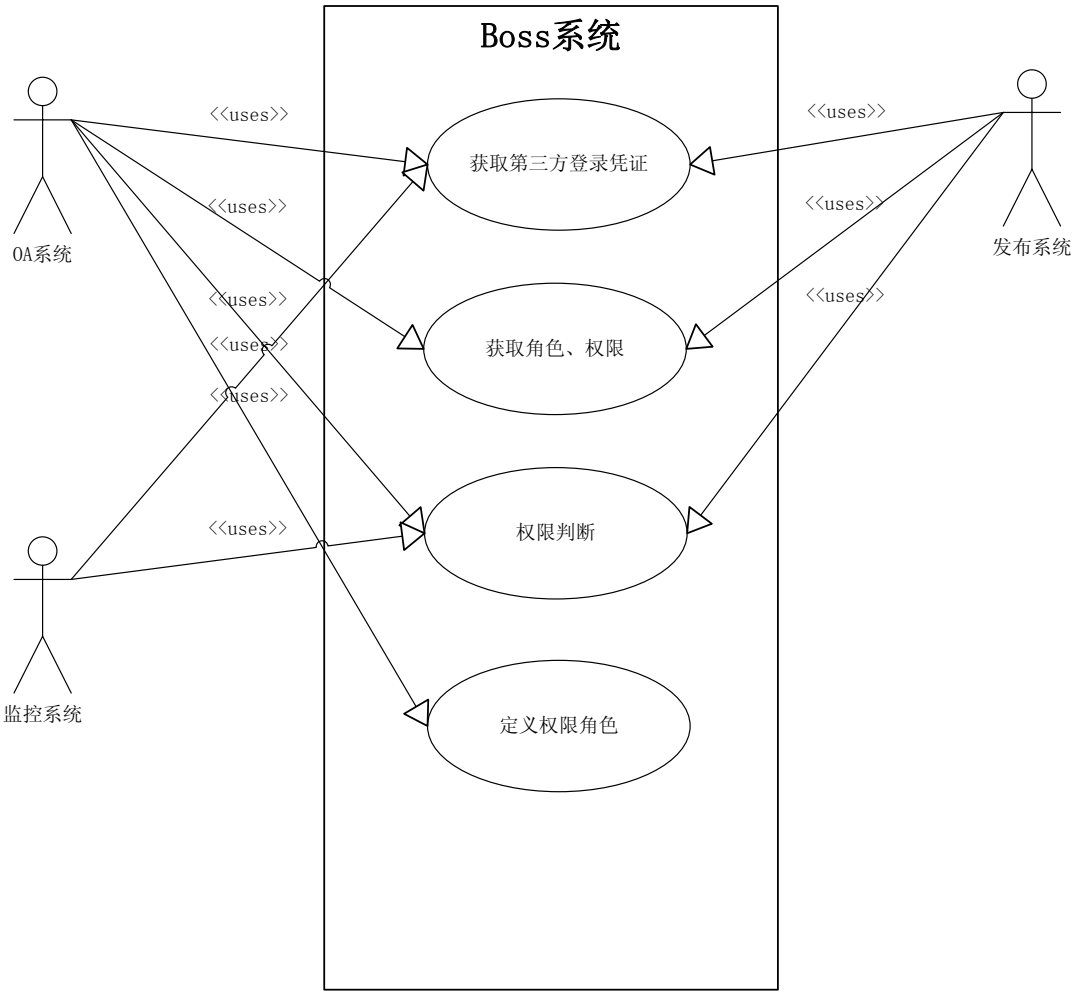


5.3.4 附图1 监控系统范例

1. 运维人员通过监控系统定义监控指标。并将监控需要的采集探针上传到配置管理系统中。
2. 通过配置管理系统的安装能力，监控系统将探针部署到线上环境中。
3. 监控探针定期或者根据事件反馈监控数据。监控系统分析及汇总数据。
4. 运维人员可以通过监控系统查看各种数据运行的状况和趋势。

5. 监控系统也可以根据配置的规则将各种异常或者其他需反馈的信息通过 oa 系统的待办事项或告警服务的能力通知给相关人员。

5.3.5 与Boss系统交互能力



5.3.5 附图1 各系统与BOSS 交互

6. 部分子系统概述

6.1 基础服务部分

6.1.1 分布式文件系统

a) 可参考的GFS模型

分布式文件系统提供一个分布式的、无单点故障的、高性能的文件存储服务。

业内可以参考的文件系统有很多，其结构大多类似，下图是 GFS 的结构图：

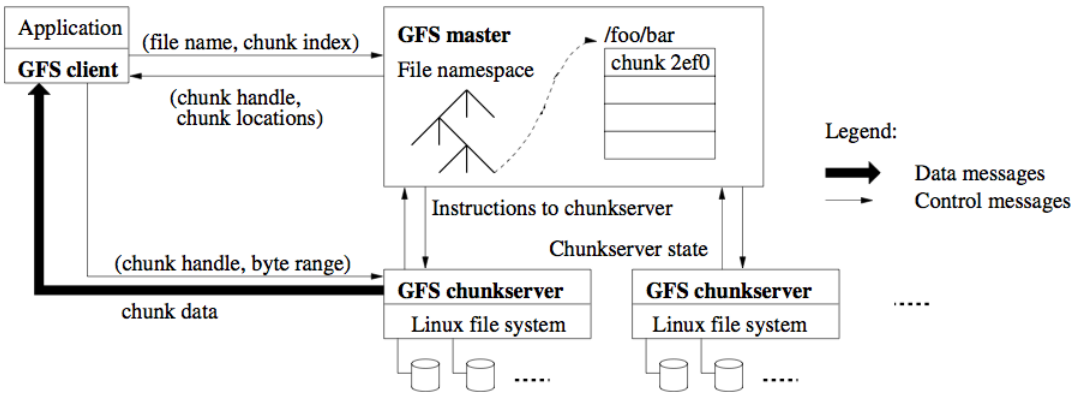


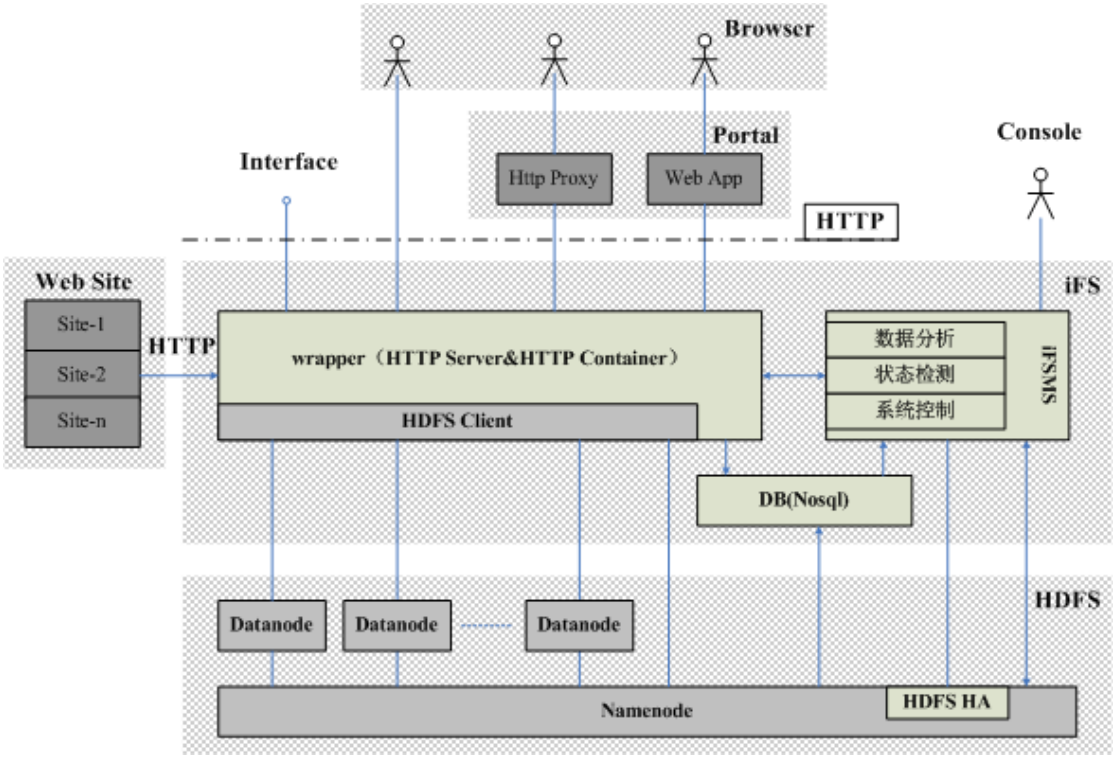
Figure 1: GFS Architecture

6.1.1 附图1 GFS 结构示意

b) IFS现状

在本文定义以前，技术部-基础研发组曾经开发过 IFS 系统。IFS 本身以 HDFS 为核心。并封装了客户端部分，提供内部路由体验，提供 http 和 memorycache 接口。

IFS 现有结构如下：



6.1.1 附图2 IFS 结构示意

其中 Datanode = GFS-chunkserver

Namenode = GFS-master

HDFSclient = GFS-client

IFS 的主要能力为封装了一个 wrapper ，进而屏蔽客户端和句柄缓存对真正业务系统的影响。

另外，考虑到 HDFS 的接口适应度，IFS 增加了一些接口能力。

c) 改进方向

结合业内经验和 IFS 系统现状，我们将一个部署态的分布式文件系统大致分为如下几个组件：

1 客户端。

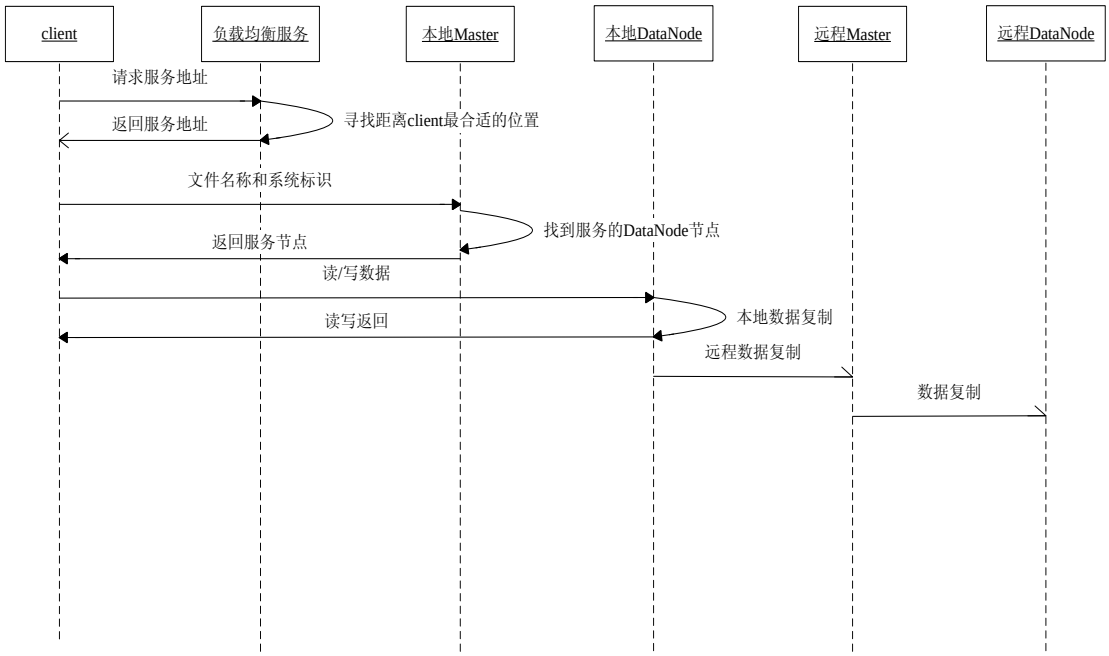
2 负载均衡服务。

3 master。

4 chunkserver.

增加的负载均衡服务将解决跨机房部署问题。

大致的时序图如下：



6.1.1 附图3 分布式文件系统数据复制时序

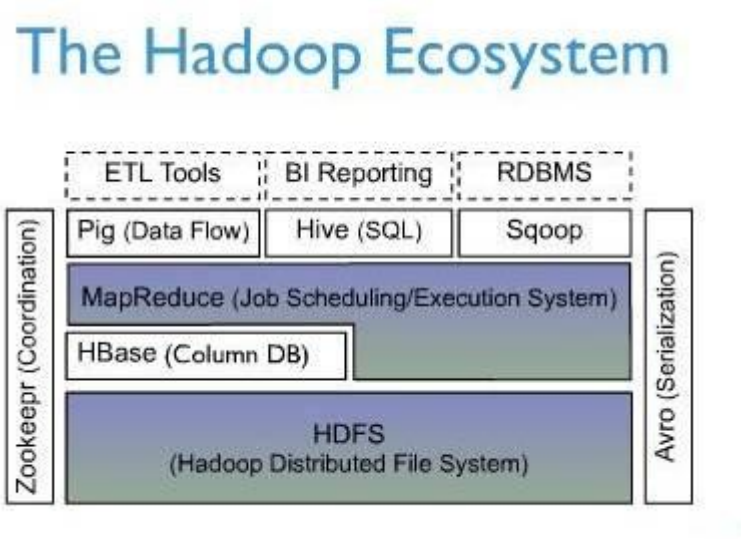
在现有的 ifs 基础上，新增一个用于跨机房的负载均衡服务，负载均衡服务为一个单独的服务。

6.1.2 MySql集群

待补充

6.1.3 Map/reduce 及 分布式计算

这两个基础服务，目前主要的应用场景在统计和搜索系统中，可以采用 HBase + zookeeper 来完成。



6.1.3 附图1 HBASE 结构示意

类似 Google Bigtable 利用 GFS 作为其文件存储系统，HBase 利用 Hadoop HDFS 作为其文件存储系统；Google 运行 MapReduce 来处理 Bigtable 中的海量数据，HBase 同样利用 HadoopMapReduce 来处理 HBase 中的海量数据；Google Bigtable 利用 Chubby 作为协同服务，HBase 利用 Zookeeper 作为对应。

6.1.4 NOSQL数据库

我们需要的是一个 NOSQL 数据库集群，提供一个统一的 NOSQL 能力，集群内部完成不同业务系统的域划分及管理即可。

目前我们的原有产品为 inetdb.

短期内主要的任务是明确 Nosql 存储的设计目标，针对 Inetdb 的实现，做改进或者重构。

6.1.5 通用队列服务

通用队列服务目前我们的自有产品为 icqueue,提高一个标准队列体验，但距离一个部署级的队列集群还有一定的差异。

6.1.6 通用缓存服务

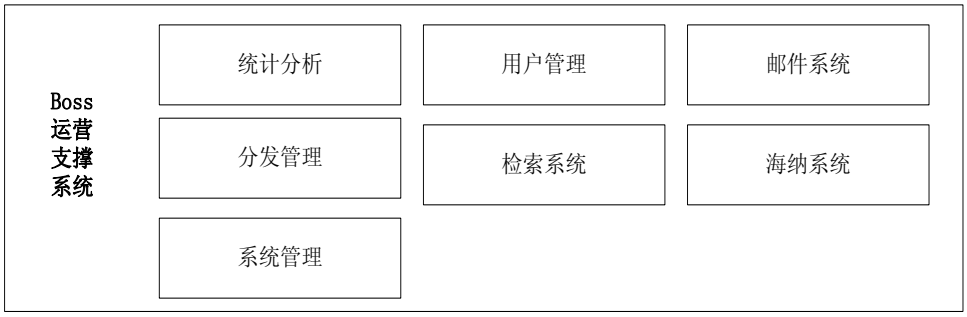
目前我们没有通用的缓存服务产品，下一步的重点是选型，比较倾向于 Redis，具体的还要看最后的比较结果，缓存服务自主开发的部分主要在分布式和分域。

6.2 基础产品部分

待补充

6.3 公共管理服务部分

6.3.1 Boss系统



6.3.1 附图1 Boss 系统结构

统计系统：对于站内文章访问趋势的统计和专题数据的统计。基于现有的统计系统不断改进完善。

用户管理：用户注册审核，重置密码；关键词访问，生成用户统计报表。部分为现有的 SSO 服务能力。

邮件系统：邮件任务管理；邮件用户、邮件组管理；邮件发送统计。部分为现有的 EDM 系统能力。

分发系统：管理从主站向各个省分站进行数据、视频的复制分发，部分为现有的 INMS 系统能力，关于 INMS 与监控系统的定位和分工，有待开发改进阶段再进行讨论和设计。

检索系统：现有的站内搜索系统能力。

海纳系统：现有的海量系统能力。

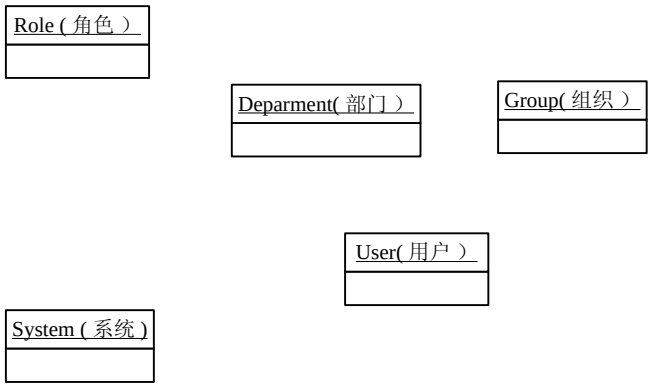
系统管理：包括编辑频道管理、用户管理、权限管理、权限组管理和资源管理。

a) 关于权限管理

我们将使用统一的权限管理，管理全网各个管理系统和业务系统需要的操作权限。

统一的权限管理将作为 BOSS 系统的一个组件存在，并由 BOSS 系统管理和维护。

统一权限管理中的数据模型如下：



6.3.1 附图2 权限管理数据模型

其中：

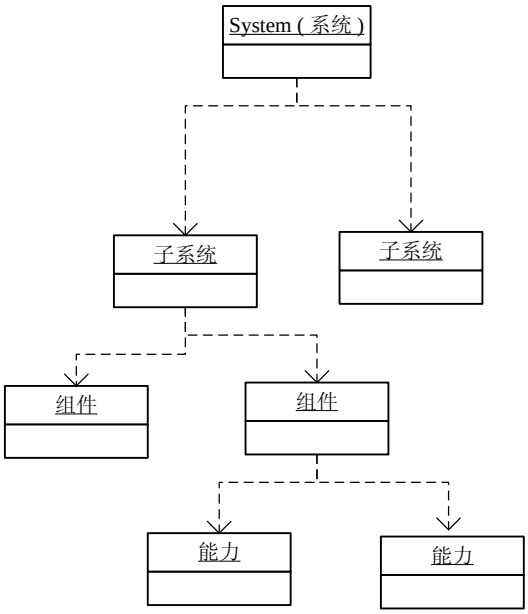
用户应对应一个自然人，一般为公司内部的一个人员。用户具有的最基本属性为：

1 登录能力：用户名、密码。

2 通知能力：手机号码、邮件地址。

系统为一个抽象或者实际存在的软件系统，对应我们的一个线上服务的部署态。

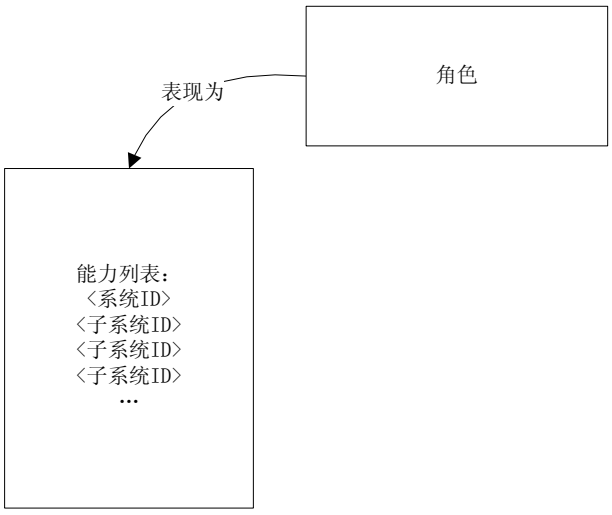
系统存在如下继承关系：



6.3.1 附图3 System 数据模型

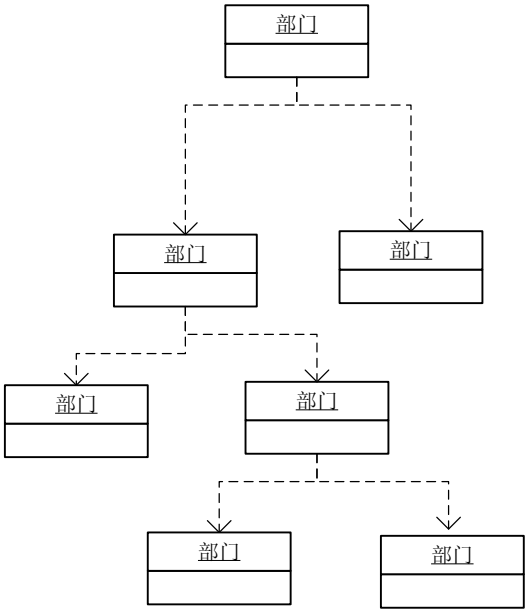
即系统内部存在一个树状结构的继承关系，使得系统可以一层一层的描述为一些具体的业务能力的聚合。

角色为一个能力的集合，其主要的含义为该角色将具有某些能力，能力为系统或者系统的子节点的抽象定义。也就是说一个角色将具有一些系统、子系统、组件或者能力的操作权限。



6.3.1 附图4 角色数据结构

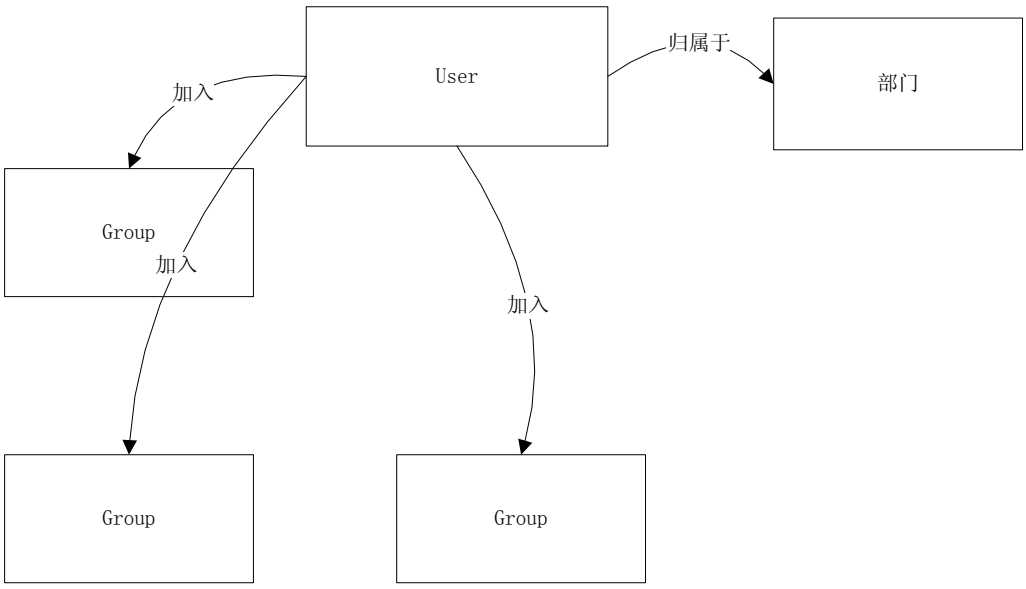
Department 部门将直接映射公司的组织结构，为一个典型的树状组织结构。员工会具体属于某一个部门。部门数据将直接对应管理职能，一般某一个部门内的人员应该具有同样能力属性。



6.3.1 附图5 Department 数据模型

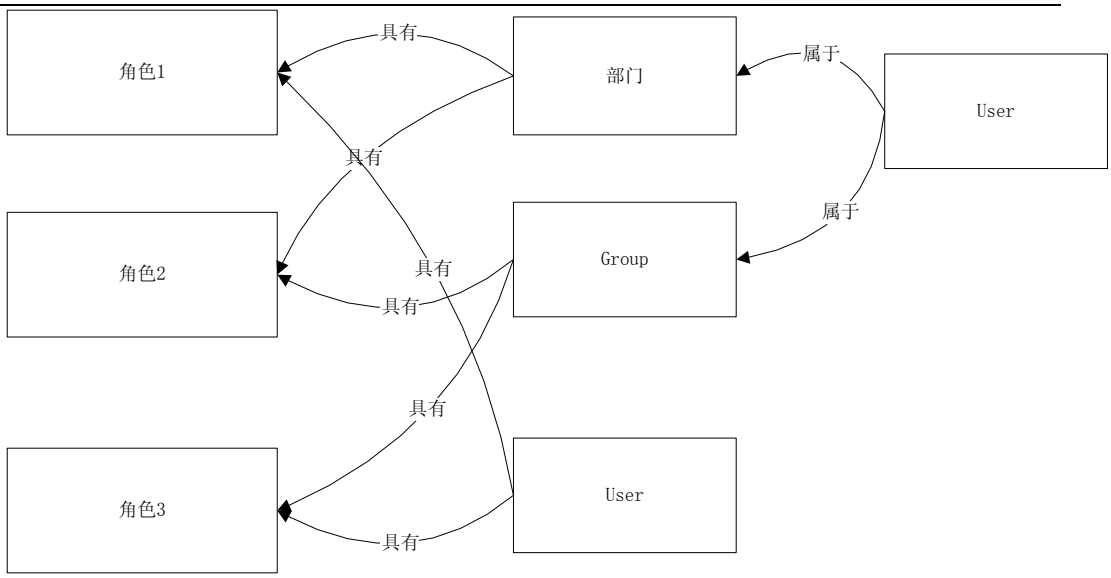
Group 组织本身不存在从属关系，为一个散列的数据，每一个组织由一些人员组成，一般情况下一个组织将向某一个业务行为负责，比如视频业务组中可能包含视频部门的编辑人员、技术部门的技术支持人员、技术部门的产品研发人员、技术部门的运维人员。

一个 User 只能属于一个部门，但可以属于多个组织。



6.3.1 附图6 User 从属关系

任何一个人、组织或者部门都可以具有某一个角色。同一个组织、部门或者人可以对应多个角色。



6.3.1 附图7 角色映射关系

当一个部门具有某一个角色时，其子部门和部门及子部门归属的人员也同时具有该角色对应的能力。

当一个组织具有某一个角色时，属于这个组织的人员也同时具有这个角色的能力。

依赖于上面的模型，在 Boss 系统中构造一个统一的权限管理模块，达到如下的几个能力。

1. 部门组织结构管理。
2. 业务组织管理。
3. 人员管理。
4. 角色管理。
5. 系统功能管理。

   在上述能力中，将完成人员的入职离职，人员的权限分配；系统上下线的权限分配，系统线上的统一登录和权限管理等能力和接口。

6.3.2 配置管理系统

6.3.2 附图1 配置管理系统结构

配置管理系统整个软件架构分为业务层、事务层和传输层。业务层实现了配置管理系统通过Web 方式对外提供的服务功能；业务层还实现了控制台服务，它接收并处理运维人员从虚拟控制台传递的命令行请求；事务层主要实现了上线任务的事务控制和会话状态管理；传输层实现了基本消息转发、与各个子系统通讯、并向运维人员提供 Linux 标准的虚拟控制台(PTY)接口。

■ 业务层

项目管理：创建、发布新系统；系统的统一业务交互管理。

发布上线：部署系统安装包到线上环境并安装执行。

控制台服务：接收并处理运维人员从虚拟控制台传递的命令行请求。

■ 事务层

任务管理：指令执行单元，通过异常捕获和一致性处理等方式正确执行上层的业务请求。

工作流模块：上线过程的流程管理和状态流转。

命令解释器：控制台命令的翻译，并传递指令到上层的控制台服务。

接口调用单元：各个接口单元的事务处理和会话控制，处理调用过程的超时和异常错误。

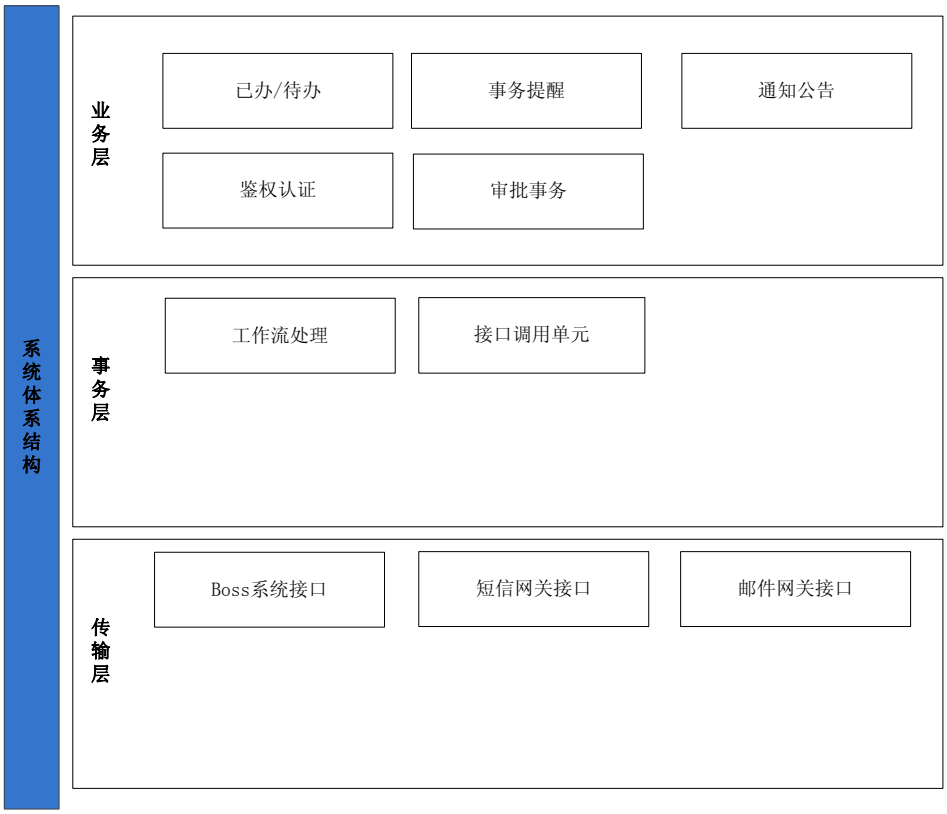
■ 传输层

消息转发：接收线上环境或版本服务器的消息数据、结构化处理，传递到上层；上层的任务请求和数据通过消息转发模块发送到线上环境或版本服务器。

虚拟控制台接口：向运维人员提供的 Linux 标准虚拟控制台(PTY)接口，运维人员能够使用比较熟

悉的命令行方式查询线上系统的运行状况或上线活动。

6.3.3 OA系统



6.3.3 附图1 OA 系统结构

OA 系统整个软件架构分为业务层、事务层和传输层。业务层实现了 OA 系统过 Web 方式对外提供的服务功能；事务层主要实现了工作流程的事务控制和会话管理；传输层实现了与各个子系统通讯。

■ 业务层

已办、代办：获取起草、发起或流转本用户的工作事务列表和明细。

事务提醒：通过消息、短信或展示页面等方式醒目的告诉用户代办事宜或消息。

通知公告：管理员给已知人群发送的系统消息和公告。

鉴权认证：用户登录通过第三方 Boss 系统进行身份认证和操作权限判定。

审批事务：对起草或流入的文档、信息进行审批。

■ 事务层

工作流模块：系统流程管理和状态流转。

接口调用单元：各个接口单元的事务处理和会话控制，处理调用过程的超时和异常错误。

■ 传输层

略。

6.3.4 监控系统

6.3.4 附图1 监控系统结构

监控系统整个软件架构分为业务层、事务层和传输层。业务层实现了监控系统对外提供的监控服务功能集合；业务层还实现了控制台服务，它接收并处理运维人员从虚拟控制台传递的命令行请求。事务层主要实现了事务控制和会话状态管理，并实现了监控任务的语义分析、控制执行。传输层实现了基本消息转发、与各个子系统通讯、并向运维人员提供 Linux 标准的虚拟控制台(PTY)接口。

■ 业务层

监控项定义：定义需要采集的监控项，监控项不仅包括上线主机的指标（例如 CPU 占用、内存占用、IO 占用等等），也包括各个上线系统的运行情况指标项。

监控语义：定义了监控动作、触发规则、报警规则等等。

监控数据收集：接收各个上线主机的监控数据或主动向各个主机发送监控数据请求。

监控报表：创建、生成各个监控主机的历史数据报表和趋势图。

控制台服务：接收并处理运维人员从虚拟控制台传递的命令行请求。

■ 事务层

任务管理：包含监控决策部件和监控动作执行单元，通过异常捕获和一致性处理等方式正确执行上层的业务请求。

命令解释器：控制台命令的翻译，并传递指令到上层的控制台服务。

接口调用单元：各个接口单元的事务处理和会话控制，处理调用过程的超时和异常错误。

系统报警：接收任务管理模块的报警请求，调用底层接口发送报警短信、邮件等等。

日志处理：监控数据存储到分布式文件系统，日志分析。

- 传输层

消息转发：接收线上环境监控消息、数据，传递到上层；上层的监控请求和数据通过消息转发接口发送到线上环境。

虚拟控制台接口：向运维人员提供的 Linux 标准虚拟控制台(PTY)接口，运维人员能够使用比较熟悉的命令行方式查询线上系统的运行状况和监控数据。

7. 实施计划

待补充

8. 遗留问题

1. 基础产品的具体产品列表需要待基础服务和管理系统完善以后逐渐定义。
2. 部分章节需等待初稿通过审核以后补充完善。
3. 基础框架部分由于和部署无关，未作描述，后续待补充。