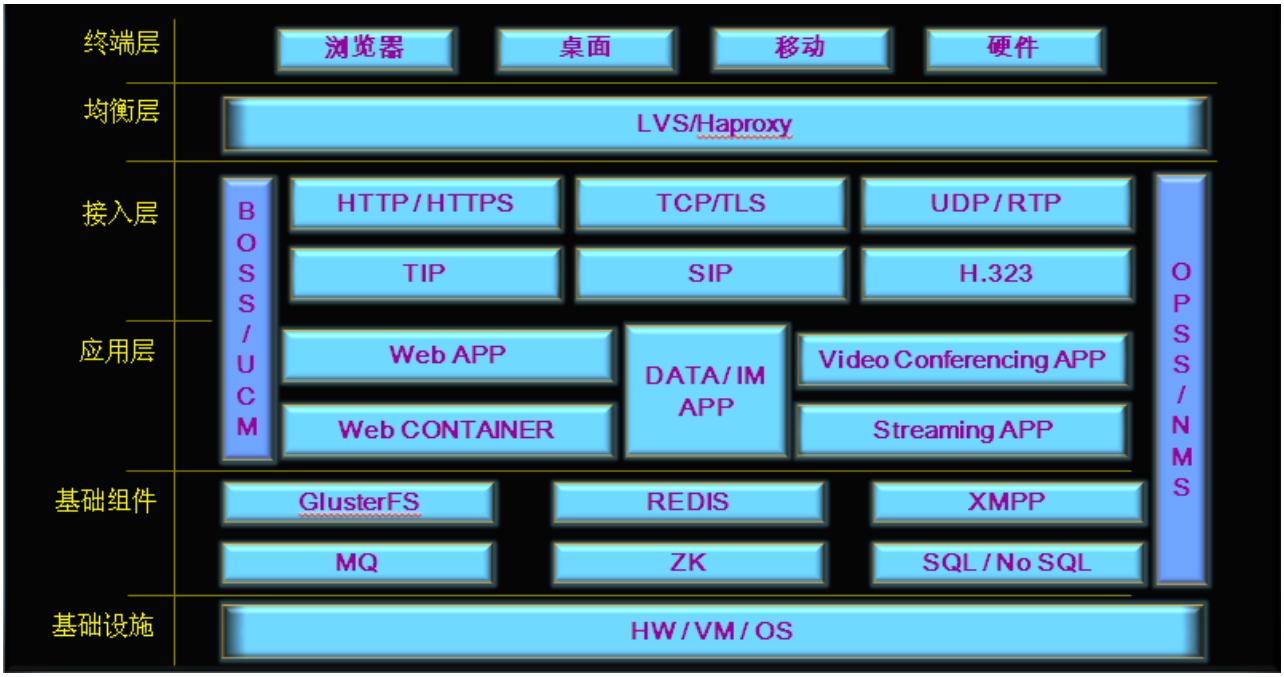
Le .5.0 会议平台架构

1 平台技术概览



2 数据词典

缩略语 KEDACOM® Jedi		
命名规则： 1、命名长度尽量控制在 3 个字母； 2、以英文全称首字母作为命名； 3、板卡命名以 U（Unit）为结尾； 4、外置服务器/业务模块以 S 结尾，分别代表 Server/Service； 5、客户端软件以 T 结尾； 6、子系统以及多业务产品以 S 结尾，代表 System； 7、控制台统一叫 Console,例如取名 CMC,区分 CMC Web，CMC Desktop，CMC for iPad。		
核心域服务		
NMS	Net Management Server	网管服务器分为 NMS_Webserver（网管 web 服务器）、NMS_starter（网管数据分析处理）、NMS_collector（网管接入与数据收集）

BMC	Business Management Console	业务管理中心
AMC	Account Management Console	账号管理中心
SSO	Single Sign On	单点登录
UPS	Unified Presence Server/Service	统一出席服务
平台域服务		
终端接入层		
APS	Acess Point Server/Service	接入点服务
协议适配层		
PAS	Protocol Access Server/Service	协议接入服务
状态通知层		
UPS	Unified Presence Server/Service	统一出席服务
媒体资源分配		
CSS	Conference Schedule Server/Service	会议调度服务
CMS	Conference Management Server/Service	会议管理服务
媒体处理层		
MRS	Media Relay Server/Service	媒体转发服务
MPU/MPS	Media Processing Unit/Server/Service	媒体处理服务
平台应用		
SSO	Single Sign On	单点登录
IVR	Interactive Voice Response	互动式语音应答
CMC	Conference Management Console	会管系统含 web mcc
基础服务层		
ZK	Zookeeper	Zookeeper
MQ	MQ	消息队列
Mysql	Mysql	数据库
REDIS	REDIS	key-value 存储系统
MODB	MODB	MODB
Keepalived	Keepalived	Keepalived
Nginx	Nginx	Nginx
Haproxy	Haproxy	Haproxy
GFS	Gluster File Server	GFS 是一个可扩展的分布式文件系统，用于大型的、分布式的、对大量数据进行访问的应用。
扩展业务		
XNU	Extended Notification Server/Service	扩展消息通知单元(即 XMPP 服务器)
SNS	Social Networking Services	社会性网络服务
LGS	Log Server/Service	日志服务器
NTP	Network Time Protocol	网络时间同步服务

SUS	Software Update Server/Service	软件升级服务器（sus 分根节点 sus-m 和子节点）
SDS	Scalable Deployment Server/Service	平台部署服务
NDS	Network Detect Server/Service	网络分析服务器(提供给 NDT 桌面客户端使用）
均衡负载		
LBS	Load Balance Server/Service	均衡负载服务器
DRM	Dynamic Resource Management	动态资源管理
桌面软件		
AST	Automatic Search Toolkit	自动搜索工具
NDT	Network Detect Toolkit	网络分析工具
板卡以及服务器		
UMU	Unified Management Unit	统一管理单元（x86 板卡）
UMU-E	Extended Unified Management Unit	增强版的统一管理单元（见板卡网口的区别）
SMU	Station Management Unit	机箱管理单元
CEU	Chief Executive Unit	主控单元（板卡）
CEU-E	Chief Executive Unit - Extreme	增强版的主控单元（见板卡网口的区别）
XMPU	Extended Media Processing Unit	超级媒体处理单元（板卡）
VDU	Video Dispatch Unit	视讯调度单元
VRS	Video Recording System	会议录播系统
DCS	Data Conferencing System	数据会议系统
BGW	Border Gateway	边界网关
平台部署工具		
DM	Domain/Deploy Manager（域/部署管理器）	负责节点间的部署逻辑
NM	Node Manager（节点管理器）	负责在节点内执行部署相关命令
Node	Node（节点）	Linux 操作系统
Ent	Environment（环境）	对应一个平台域
Rrecipe	Rrecipe（配方包）	代表一个应用。为 NM 提供此应用的具体部署实现
协议		
XMPP	The Extensible Messaging and Presence Protocol	可扩展通讯和表示协议
SIP	Session Initiation Protocol	会话发起协议
H.323	H.323	基于分组网络 PBN（packet Based Networks）上的多媒体通信系统标准
HTTP	Hypertext transfer protocol	超文本传送协议

URI	Uniform Resource Identifier	统一资源标识
DTMF	Dual Tone Multi Frequency	双音多频信号
GB28181	GB28181	安全防范视频监控联网系统信息传输、交换、控制技术要求。公安部科技信息化局提出，由全国安全防范报警系统标准化技术委员会（SAC/TC100)归口，公安部一所等多家单位共同起草的一部国家标准。
专业术语		
BOSS	Business & Operation Support System	综合业务和运营支撑系统
BSS	Business Support System	业务支撑系统
OSS	Operation Support System	操作支撑系统
CSS	Customer service System	客户服务系统
SSO	Single Sign On	单点登录
OA	Office Automation	办公自动化
登录账号	5.0 登录账号含：注册账号、邮箱、手机、号码这四类。	
号码	即视频会议的 E.164 号码	
域	核心域、服务域、平台域、用户域	

3 域

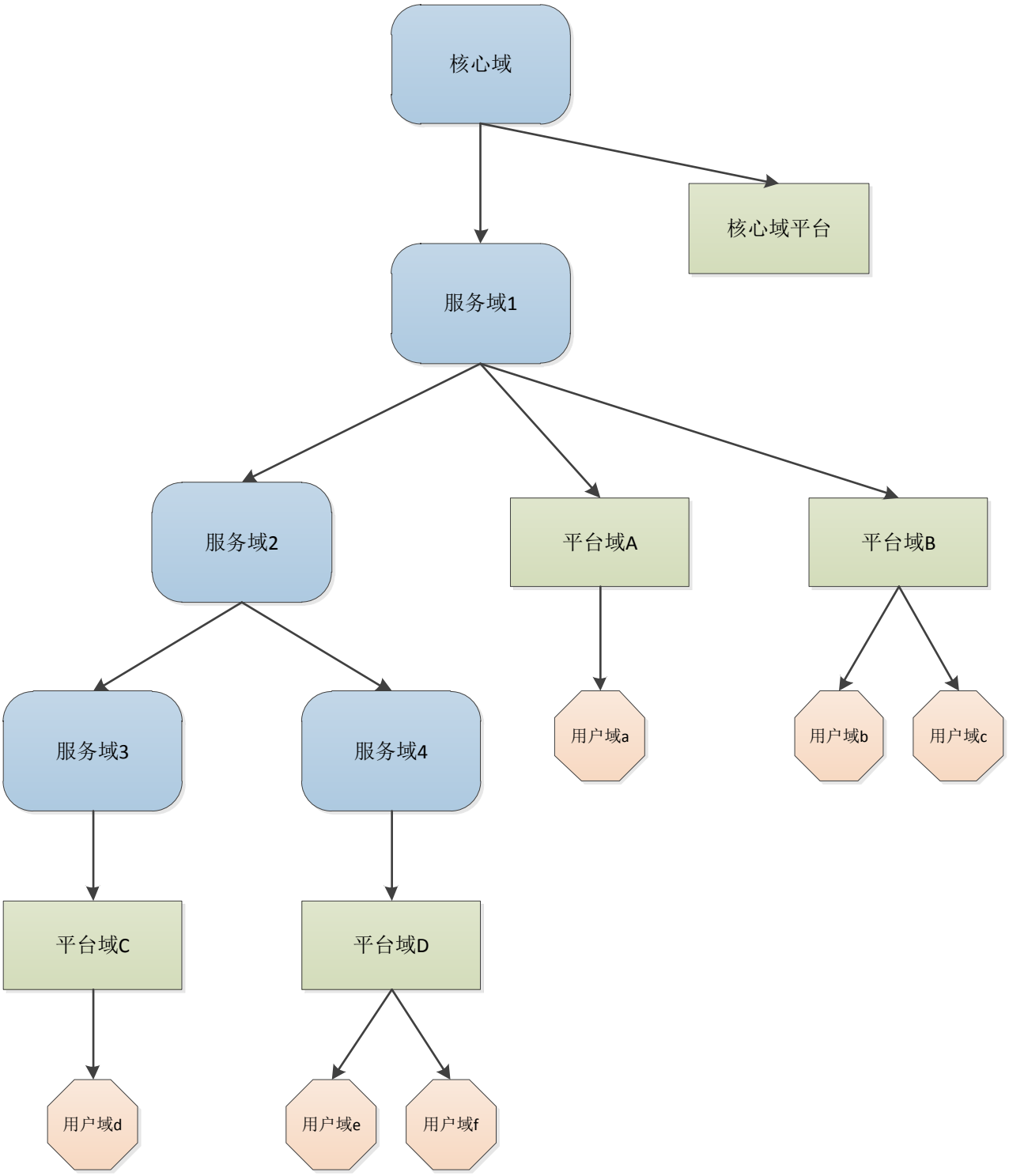
3.1 域划分的背景

V5.0 的域划分规则来源于 Motrix V2.6 的基础思想，并充分考虑原有设计在使用场景上的局限性并加以改善，首先追溯一下 V2.6 核心域、公有云、用户域三级划分的缺陷：

- 代理商、渠道拓展独立业务的使用场景：针对此类运营商，V2.6 系统无法支持受限范围内进行账号分配，所有的企业开通、放号都需集中在我司进行，也不利于运营商对客户信息的保护；
- 同一运营商异地部署平台的使用场景：V2.6 的设计里，所有的公有云都是独立存在相互之前无任何关联，服务没有办法共享，同样的原因，也无法实现平台的异地灾备；
- 行业应用场景多层级树形架构的使用场景：V2.6 的设计里，所有的公有云都是平级关系，无法进行层级的嵌套和关联；
- 跨用户域管理的使用场景：一方面希望各用户域独立用户信息，一方面部分高级权限管理员又有跨域管理的需求，此类场景在大型集团或者行业应用中都普遍存在。

针对上述使用场景，V5.0 对域进行了重新划分，，首先保留了核心域和用户域的概念，其次将公有云之间的逻辑关系具象成服务域，将其原来所代表的物理设备集合称之为平台域。通过上述划分，从而保证了系统部署架构上的充分自由度。

3.2 服务域、平台域、用户域的关系



核心域：核心域为整个 JDV5.0 平台的中心节点，保存着整个平台最为核心的数据，由 SSO、BMC、AMC、NMS、UPU 逻辑服务组成，功能涵盖了统一鉴权登录、逻辑域的划分以及权限控制、管理员账号分配、平台维护支撑等，并以此为基础为平台域各类应用提供数据下发功能

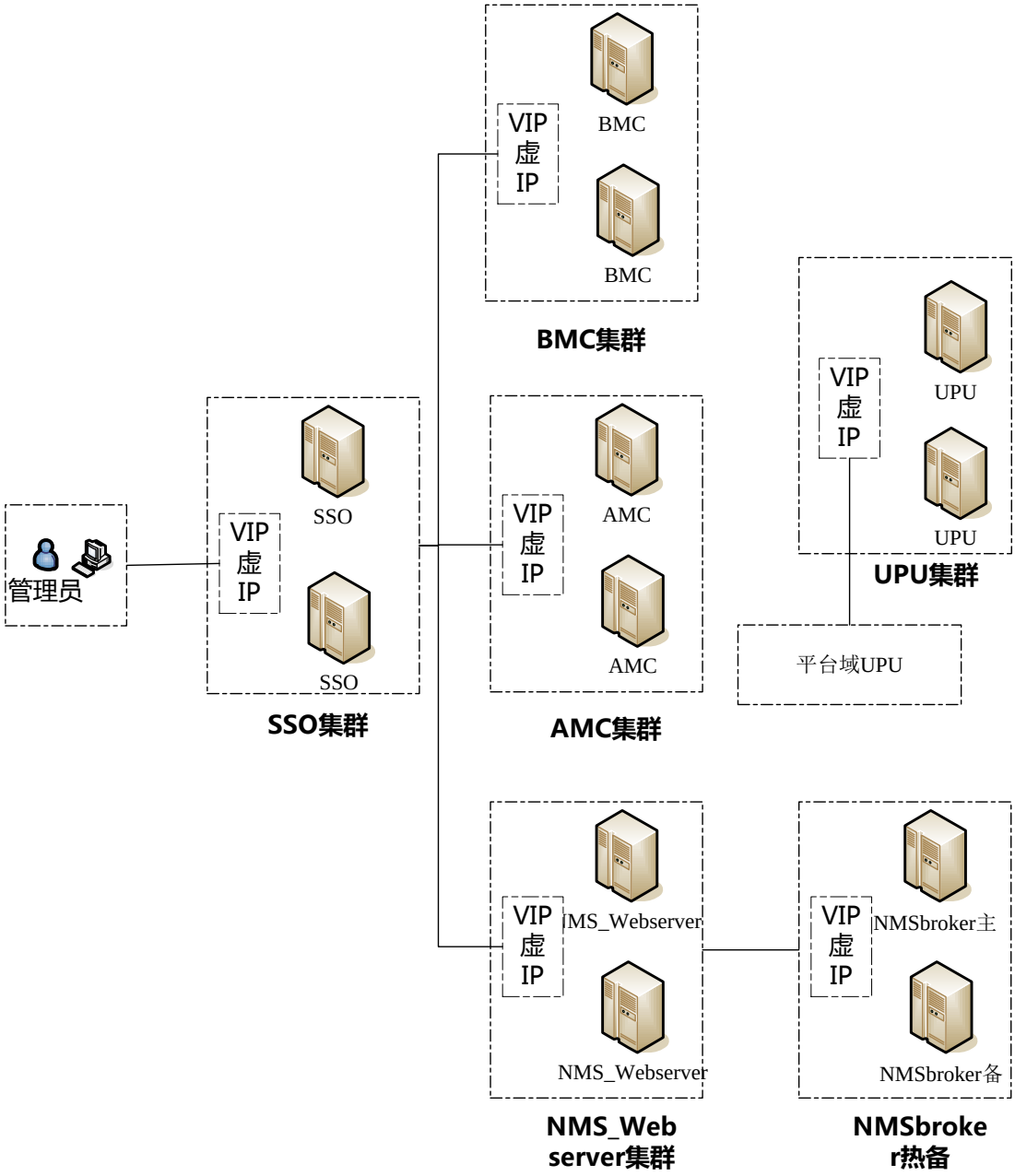
服务域： 服务域不含任何物理设备仅是逻辑上的概念，它可以被认为是平台域和用户域的集合并用来表示它们的归属关系。另外由于服务域可以嵌套，这嵌套关系又可用来表示层级结构。

平台域： 平台域是用户所获取服务的生产者，是一类物理服务器或专用硬件的集合。

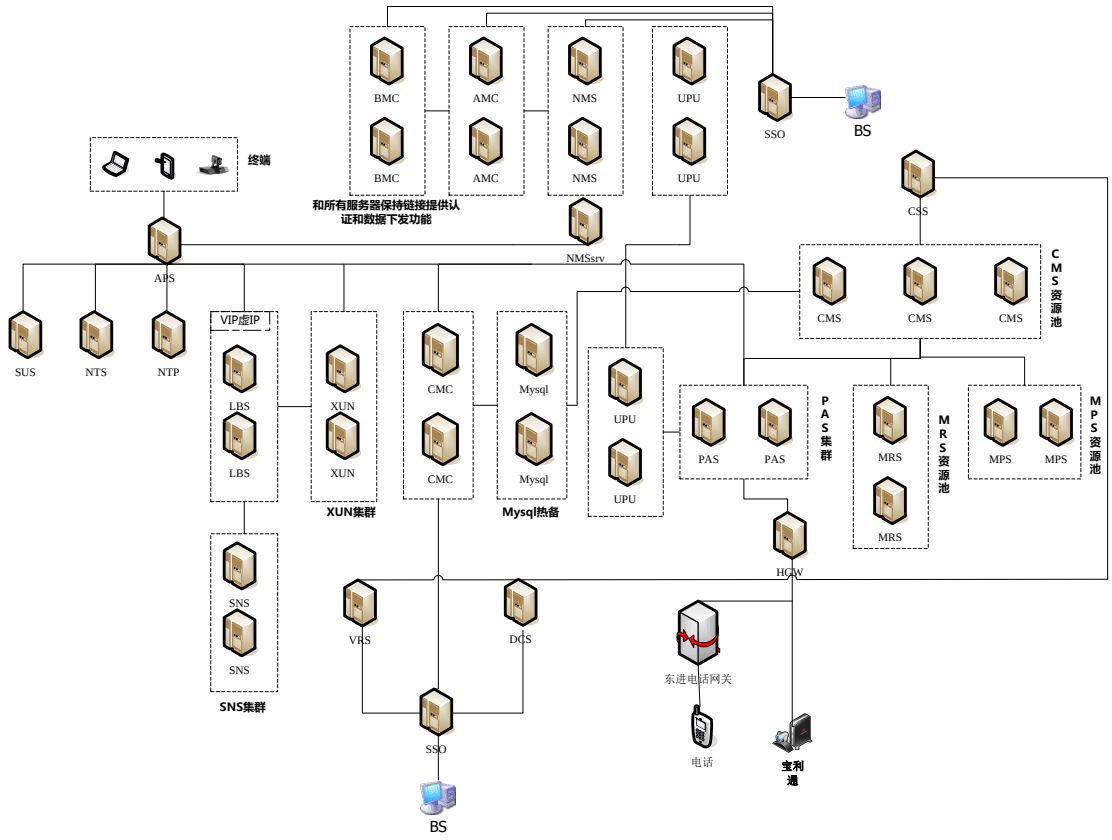
用户域： 用户域是云平台提供服务的消费者，终端、账号的集合组成了用户域。不同的用户域之间业务是相互隔离的，保证视频会议、会议管理、组织架构等私密性。

归属关系：
服务域可下辖 0 到 N 个平台域，当某级服务域无平台域时，可借用上级的平台域资源；
一个平台域只能从属于一个服务域，同一服务域下的平台域实现账号信息共享；
一个用户域归属于某个服务域，同时绑定该服务域下的某个平台域，作为优先登陆选项。

3.3 核心域组网示意图


VIP
虚
IP
BMC
BMC
BMC集群
VIP
虚
IP
UPU
UPU
UPU集群
管理员
VIP
虚
IP
SSO
SSO
SSO集群
VIP
虚
IP
AMC
AMC
AMC集群
平台域UPU
VIP
虚
IP
IMS_Webserver
NMS_Webserver
NMS_Web
server集群
VIP
虚
IP
NMSbroker主
NMSbroker备
NMSbroke
r热备

3.4 平台域组网示意图

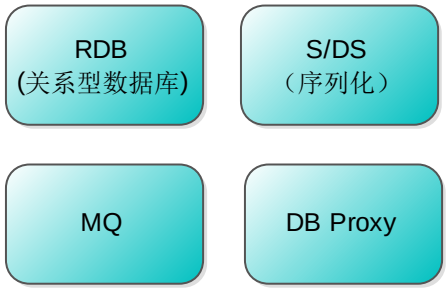


3.5 MODB

MODB 作为 5.0 架构的关键部分，在多级服务域中提供了上下级服务域之间信息的同步功能。

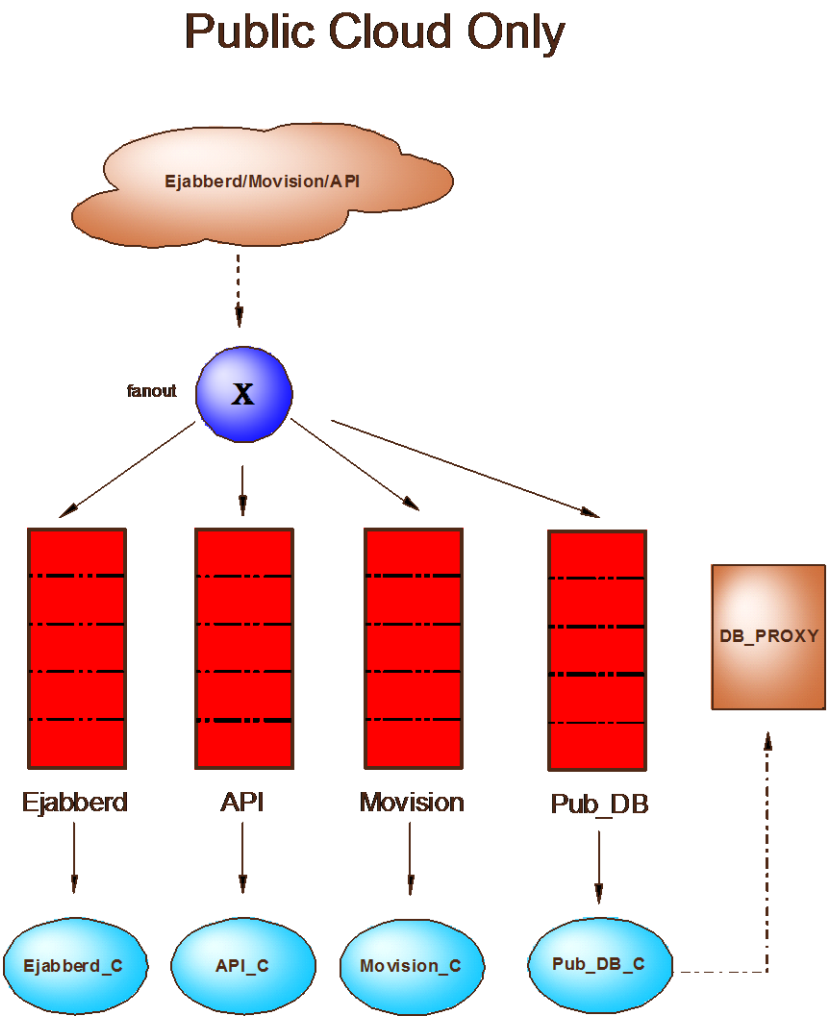
3.5.1 MODB 体系架构

MODB 由四部分组成：关系型数据库(MYSQL)、消息服务器（RabbitMQ）、序列化模块（JSON/XML/ProtocolBuffer）以及同步服务器 DB Proxy。

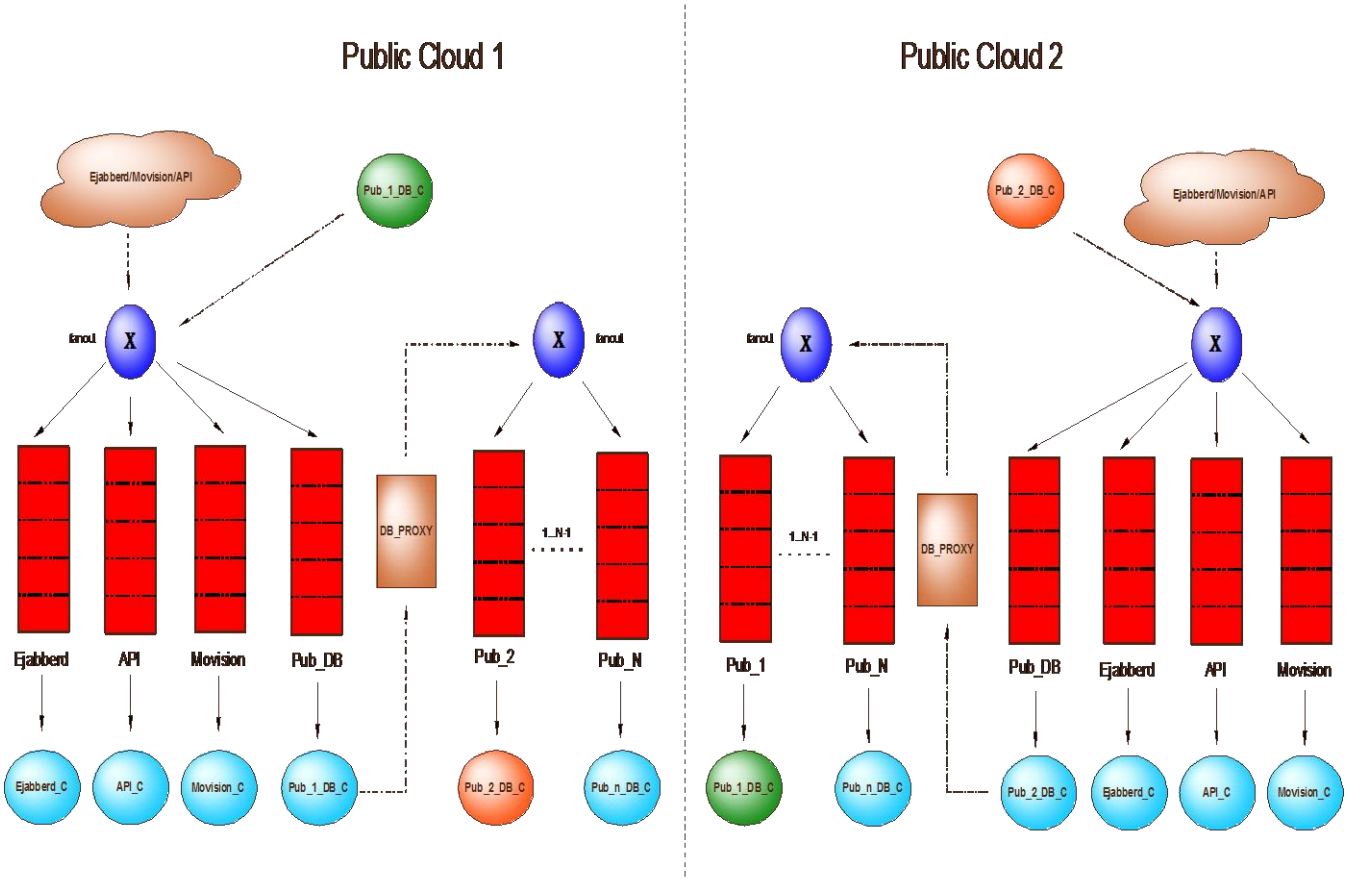


MODB 支持单公有云、多公有云、混合云模式。

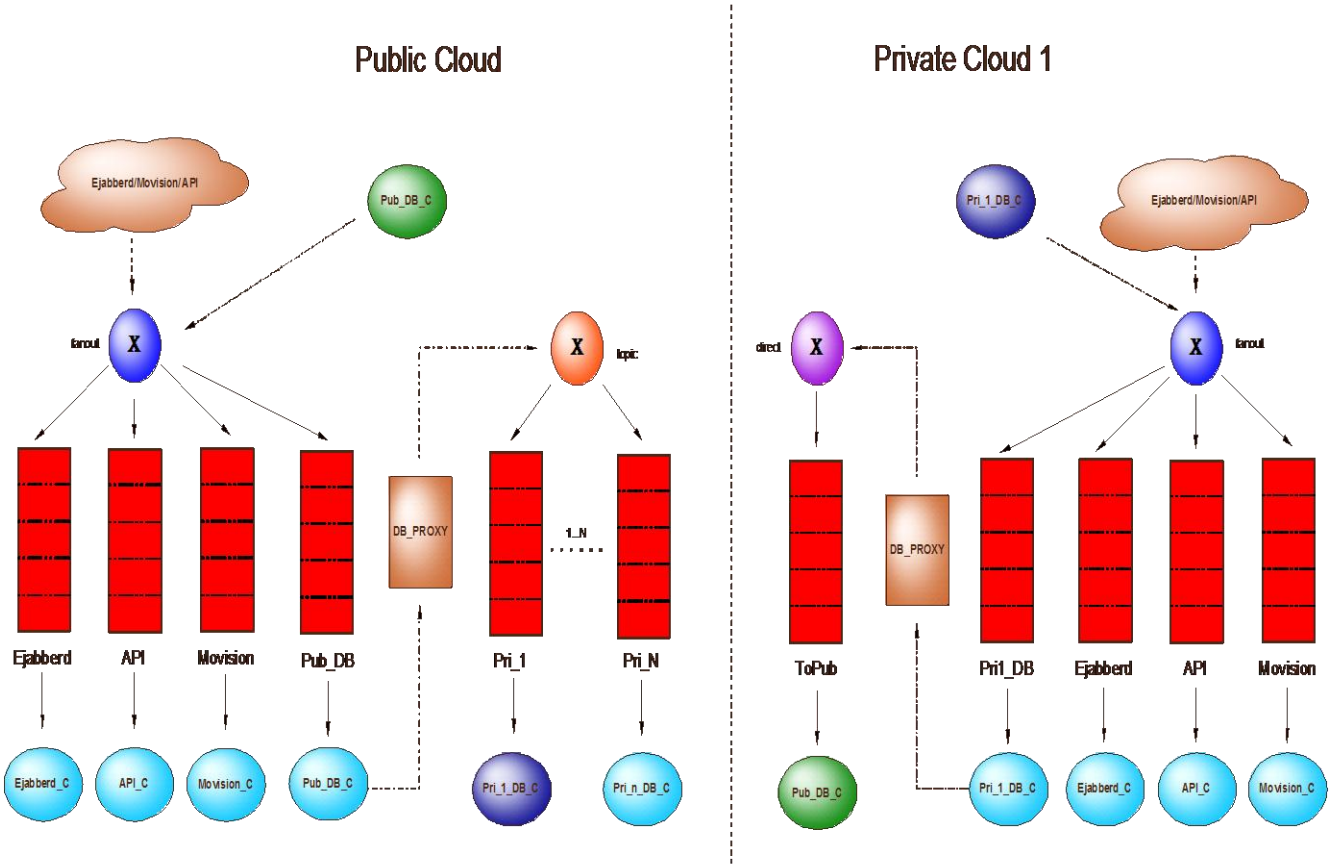
3.5.2 单公有云技术架构



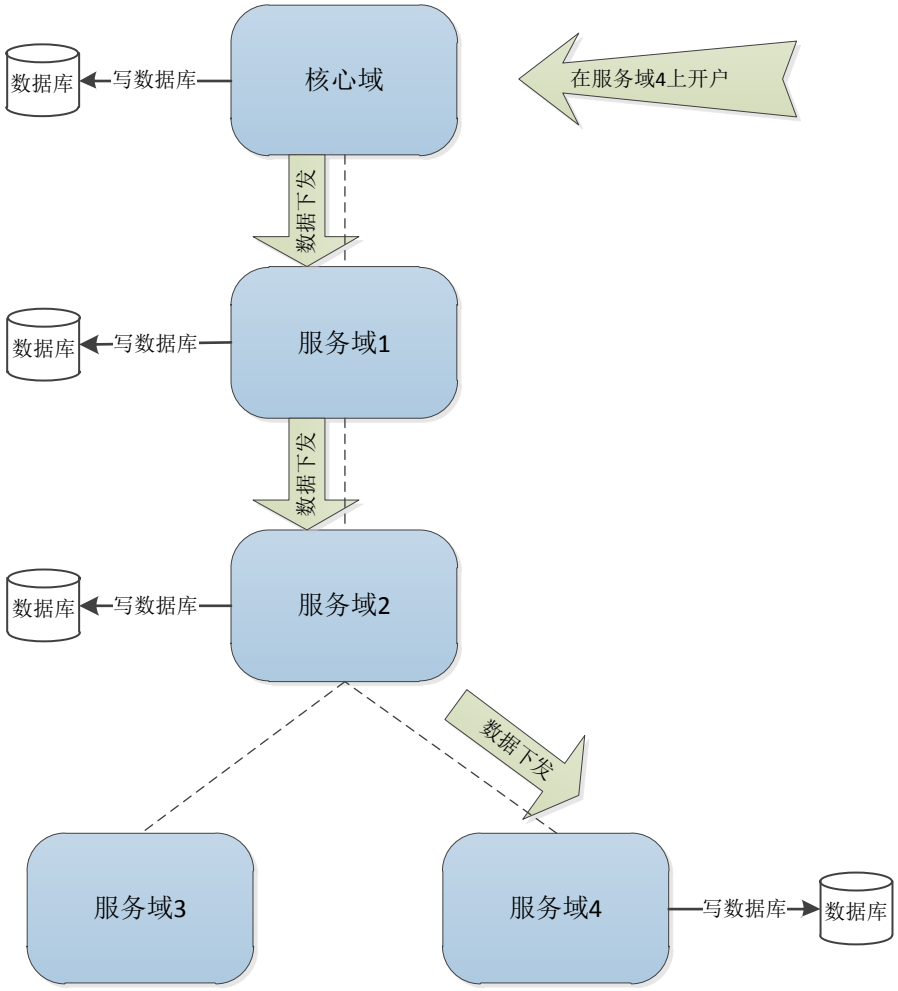
3.5.3 多公有云技术架构



3.5.4 混合云（单公有云多私有云）技术架构



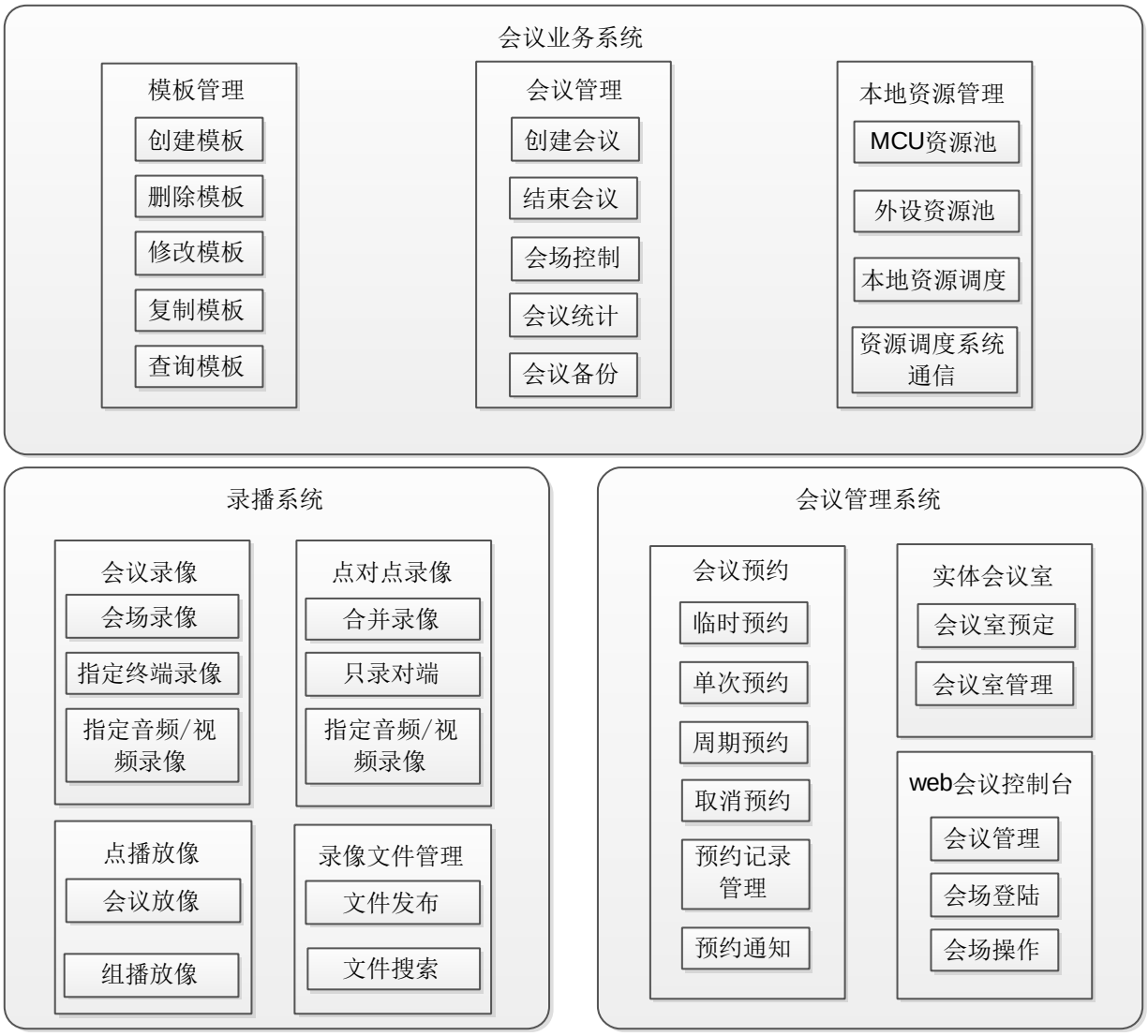
3.6 开户信息同步

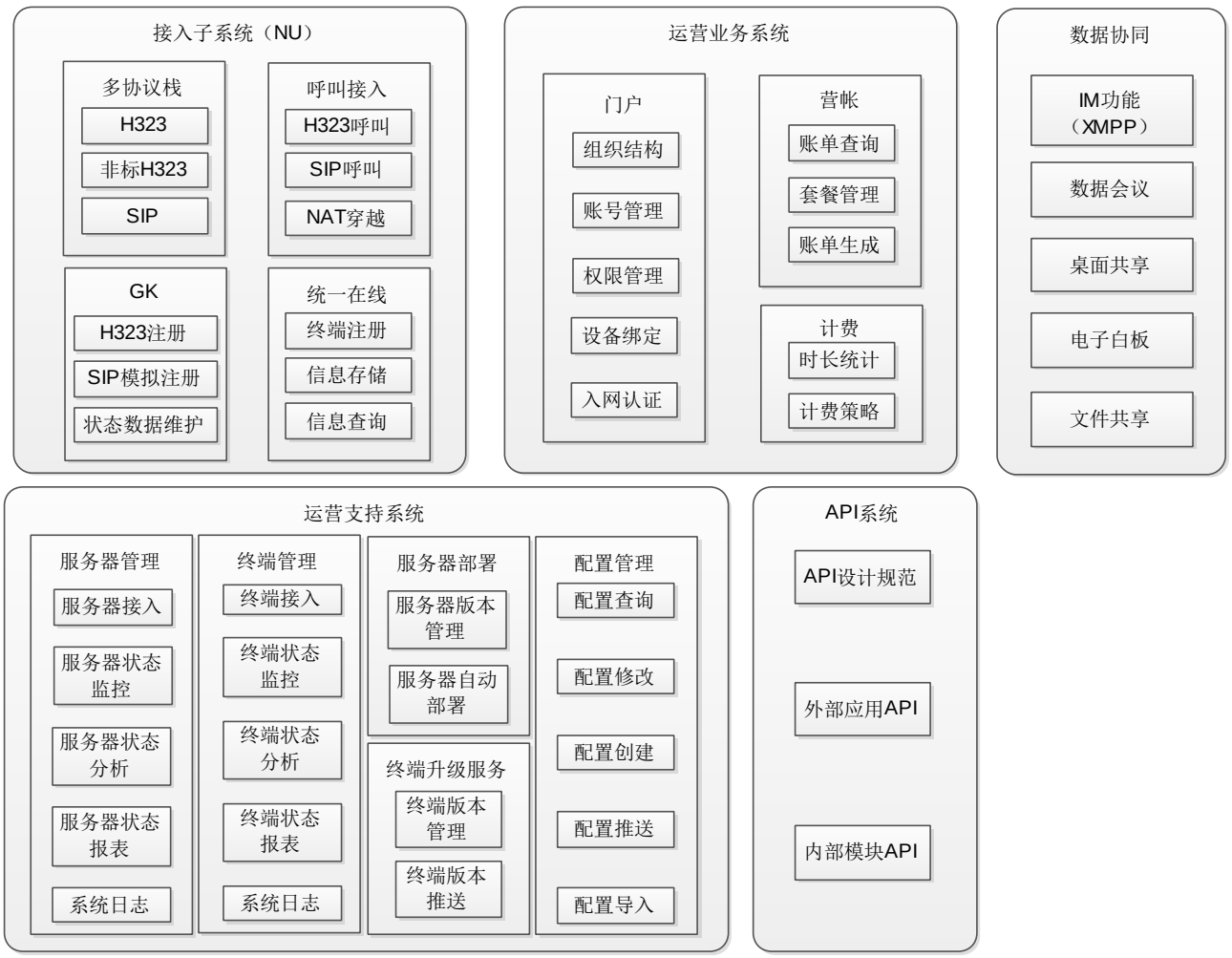


所有开户信息在核心域中操作，成功后根据从属关系逐级下发，到最终目的服务域为止。

4 子系统

4.1 子系统功能图示





4.2 子系统功能描述

(一) 会议业务子系统：提供即时视频会议的各项业务服务。包括模板管理，召开、结束、延时及各类视频会议业务，管理各类外设。

(二) 录播子系统：提供对多点及点对点会议的录放像服务，及非会议场景的录像点播功能。

(三) 接入子系统：负责各类型各厂商终端的协议接入，支持多协议，支持 NAT 穿越。

(四) 数据协同子系统：提供 IM,数据会议，桌面共享，电子白板等数据业务。

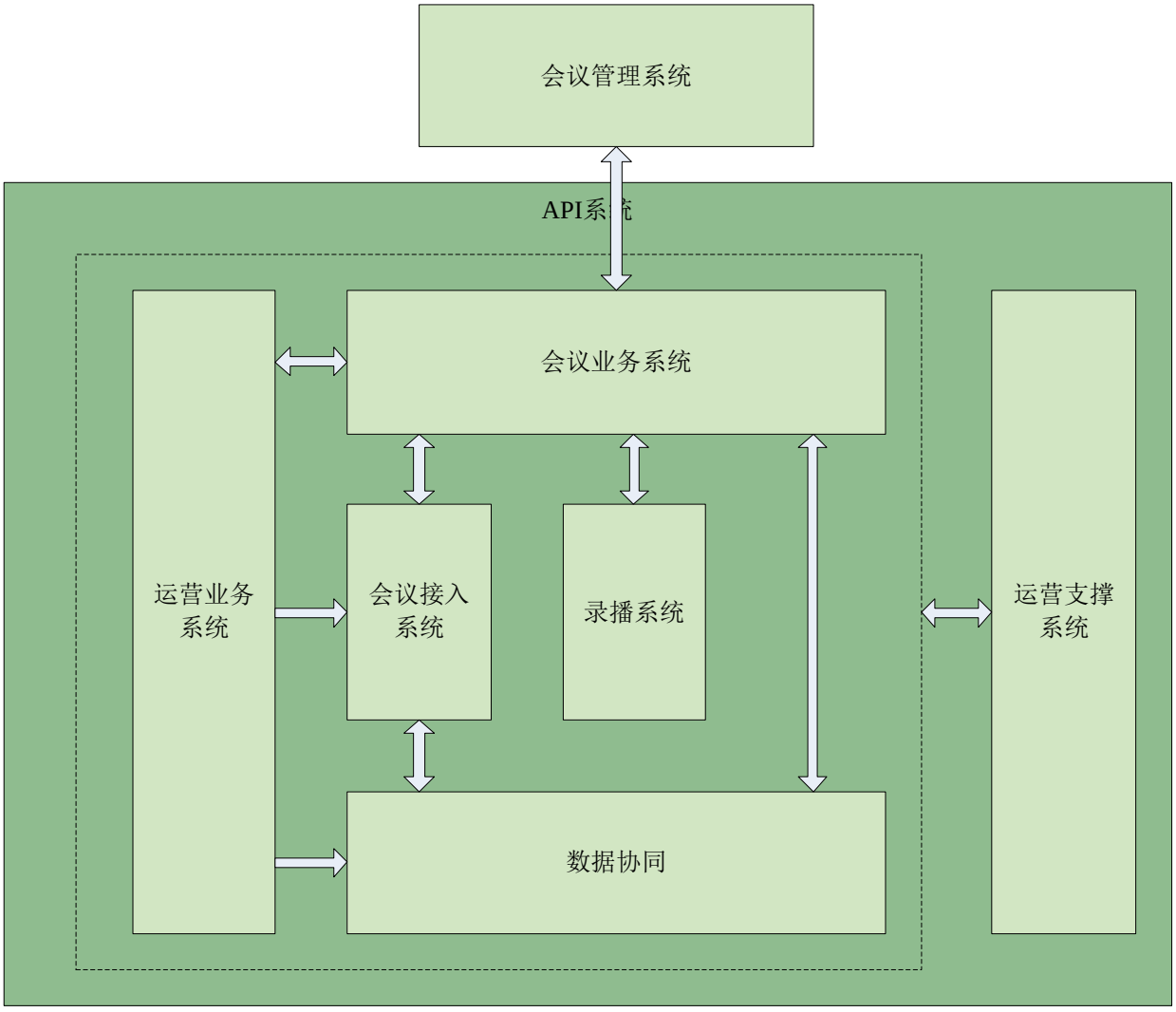
(五) 会议管理系统：提供实体会议室预定、虚拟会议室预定，实体会议室管理、实体会议预约、视频会议预约、web 会场管理等功能。

(六) 运营运营子系统：负责门户、计费、入网认证等一系列运营功能。

(七) 运营支撑子系统：提供网管，自动安装部署，服务器监控，终端监控，进程/线程监控及业务监控调试，配置管理及推送等功能。

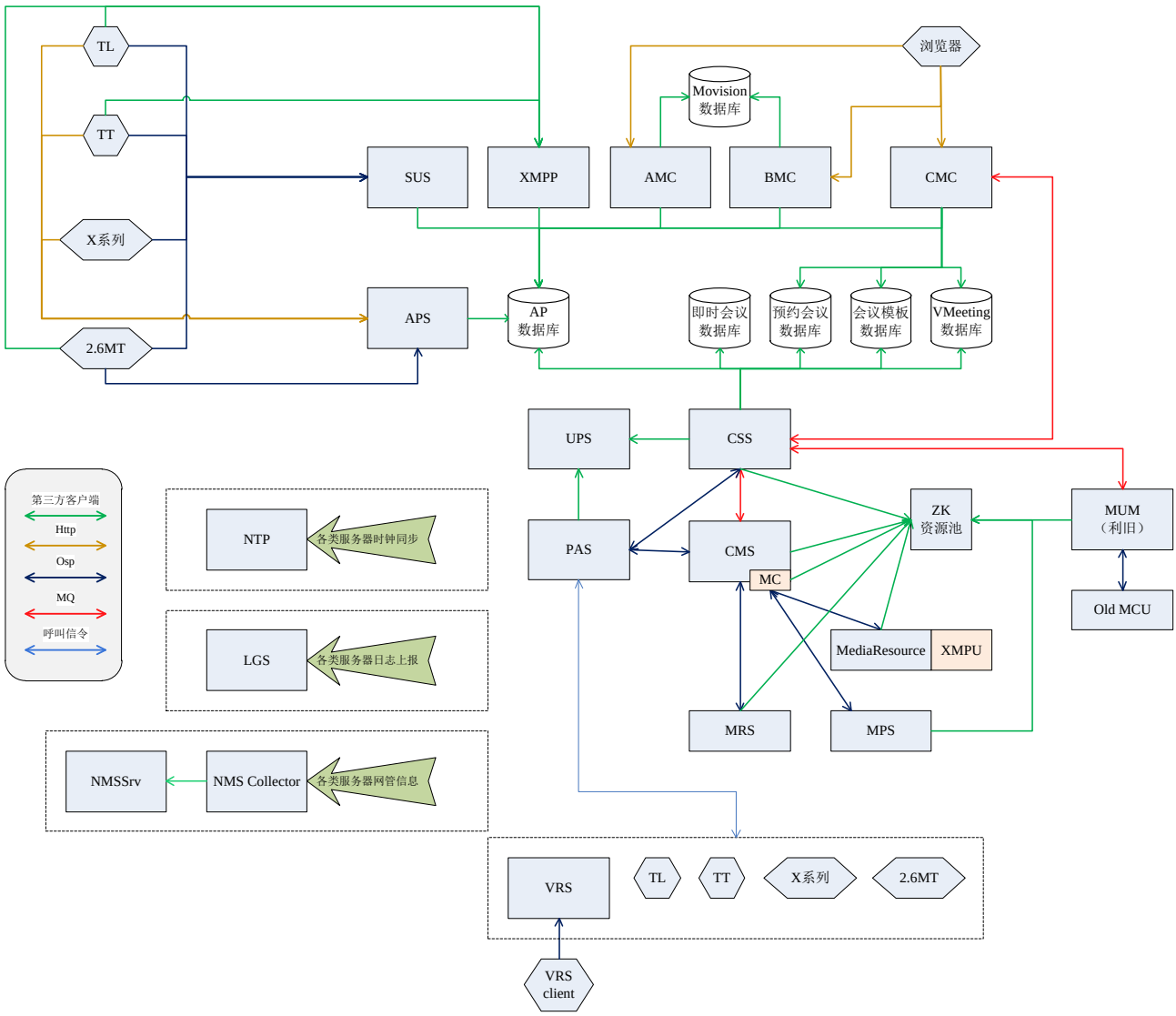
(八) API 子系统：提供平台对外的所有应用接口，及平台内部模块间的 API 调用接口。该子系统存在于所有业务系统中，在系统组成图示中不直接体现。(修改：统一入口)

4.3 子系统关系



4.4 系统模块组成关系

4.4.1 关系图



4.4.2 模块和子系统关系

(一) 会议业务子系统：CSS\CMS\UPS\MRS\MPS\MediaResource；
(二) 录播子系统：VRS；
(三) 接入子系统：SUS\APS\PAS；
(四) 数据协同子系统：XMPP;
(五) 会议管理系统：CMC;
(六) 运营运营子系统：AMC\BMC
(七) 运营支撑子系统：NTP\LGS\NMS。

5 会议平台可靠性设计

5.1 备份类型定义

(1) **热备：**主备系统部署同一套业务，同时处于上电运行状态，主系统对外提供服务。在主系统发生故障时，可以把实时业务状态和实时业务数据等“热数据”切换到备份系统上，备份系统接替主系统提供服务。由于切换时间极短，业务无中断，所以用户对故障无感知，对系统切换无感知。

(2) **冷备：**在主系统发生故障时，备份系统自动启动接替主系统工作。由于备份系统存在自启动的过程，因此用户会发现服务有明显的中断，进而推断出系统可能发生故障，但一般不会感知到系统切换。冷备系统不进行实时数据和状态的切换，通常是从数据库等永久化存储系统里进行初始化数据导入。

(3) **集群：**多个相同的系统同时上电运行，并同时为不同用户提供同类的服务，当某个系统出现故障时，可以自动或者用户手动选择另外一个空闲系统，为使用之前系统的用户继续提供相同的服务。集群系统在恢复服务的时候，视该系统提供的服务类型差异，可以不对之前服务的业务状态和数据进行恢复，也可以“准实时”恢复——恢复一个采样时间间隔前的状态和数据。因为存在不同系统之间的切换，和冷备一样，用户也会发现服务有明显的中断，但一般不会感知到系统切换。

(4) **跨机房同步：**同类系统分布在不同地域的机房中，通过专网或互联网联通，独立为本机房提供服务，同时将本机房的业务信息实时或非实时地同步到其它机房中。一个机房出现故障时，可以自动或者通过运维人员手动切换服务路径，由其它机房暂时接替故障机房的业务。跨机房同步是为了应对最极端的情况，避免出现某一地区的业务长时间（数天乃至数周）中断，在数小时或者一天之内回复对当地客户的服务。

5.2 主要应用场景概述

(1) 热备：设计和开发的复杂度高、难度大、成本高，一般应用在可靠性、实时性、用户体验要求非常高的系统上，主要是最核心的服务应用部分。简单的说，就是不能接受服务中断的部分。

(2) 冷备：对于实时性要求不高的单点设备，适合使用冷备方案。

(3) 集群：对于有动态扩展需求的设备，适合使用集群方案。

(4) 跨机房同步：对于有跨地区服务，或者单个机房无法承担业务容量的情况，需要使用跨机房同步。

以上四个方案并不是互相隔离的，结合实际情况可以两两组合乃至三个方案联合使用。例如冷备+集群、热备+冷备、热备+集群，这些都可以进一步提高系统的可靠性；而无论机房内部采用什么备份方案，在有合适方案的情况下，也可以配置跨机房同步。

5.3 具体分类

5.3.1 热备

(1) MediaResource(JD10000-CEU) -主备检测/协商，业务数据定时倒换，驱动/媒体数据定时倒换。

5.3.2 冷备

(1) CSS（JD10000-UMU， X86 服务器）- Keepalive 地址漂移，业务数据永久化存储，****可以考虑修改为集群*****

(2) UPU（JD10000-UMU，X86， 单 UPU 节点）- Keepalive 地址漂移，业务数据永久化存储

(3) CMU（JD10000，单 CMU 节点）——***可以考虑 CMU 做集群*****。

(4) PAS（JD10000）

5.3.3 集群

(1) CMU（JD10000，X86，多 CMU 节点）- zookeeper 节点登记/状态刷新，即时会议信息统一上报数据库，CSS、NU 访问

(2) MRS（JD10000-CEU，X86）- zookeeper 节点登记/状态刷新，CMU 访问

(3) XMPU（JD10000-XMPU）- 硬件架构实现，mediaresource 访问，mediaclient 间接访问

(4) MPU（X86）- zookeeper 节点登记/状态刷新，CMU 通过 mediaclient 间接访问

(5) PAS（X86）- UPU 登记/状态刷新，CMU 访问

5.3.4 跨机房同步

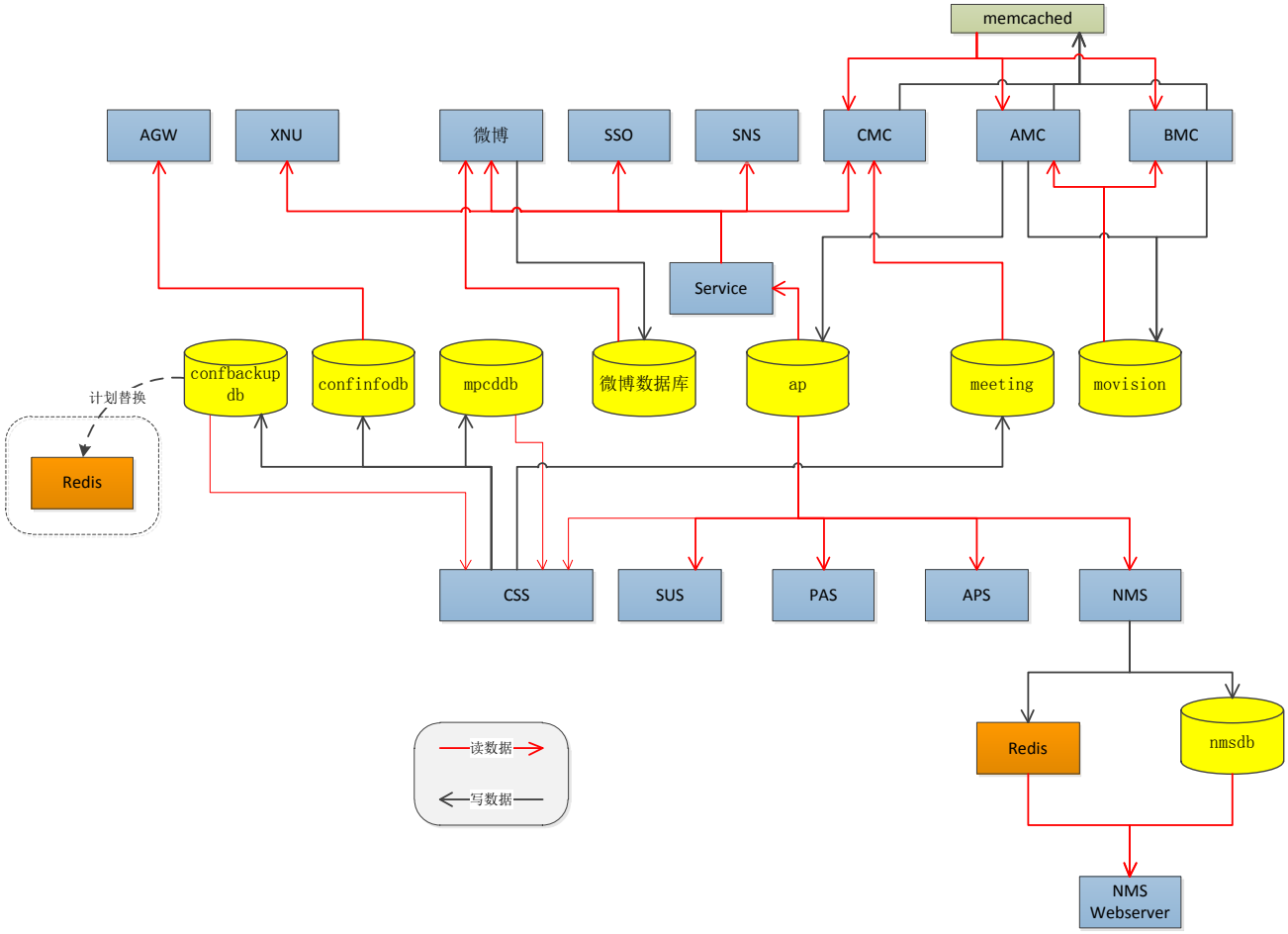
(1) CSS（JD10000-UMU， X86 服务器）- 尚无明确方案

(2) UPU（JD10000-UMU，X86，同一服务域下）-引入 Riak（dynamo 模型的开源实现）解决多写一致性问题，UPU 变为 Riak 的代理，负责往 Riak 写入数据，处理查询 Riak 时数据的多版本冲突问题，

同时负责往上级 UPU 进行数据同步与查找。

6 数据库和缓存数据库

6.1 数据库及缓存数据库-模块关系图



SSO 也读 memcached
Service 作为门户的 AP 查询/修改入口

6.2 数据库及缓存数据库功能

(1) confbackup：存放 CSS 恢复会议需要的会议详细信息。
(2) confinfodb：存放 AGW 及电话网关使用的会议索引信息（会议名，会议 E164，会议短号等）。
(3) mpcddb：存放公共模板信息，预约会议信息。
(4) ap：存放本服务域及下级服务域的用户账号信息，供各平台域登陆终端及服务器调用。
(5) meeting：存放 CSS 写入的预约会议及即时会议基本信息，以及根据 CMU 的会场状态通知由会管写入的会场状态信息，供会管使用。

(6) movision：记录核心域运营及账户信息，AMC/BMC 调用。
(7) nmsdb：统计，告警等供历史查询的数据，nsm webservice 读取。
(8) nms redis：动态变化的数据，比如会议信息，终端登录信息，nsm webservice 读取。

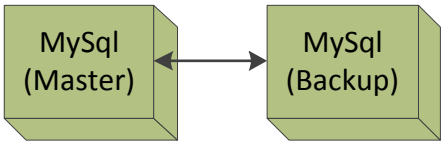
6.3 数据库

6.3.1 类型

数据库选用 MySql 作为服务组件。

6.3.2 部署

使用 mysql 自带的 mysqld_multi 实现主从复制。



6.4 缓存数据库

6.4.1 类型

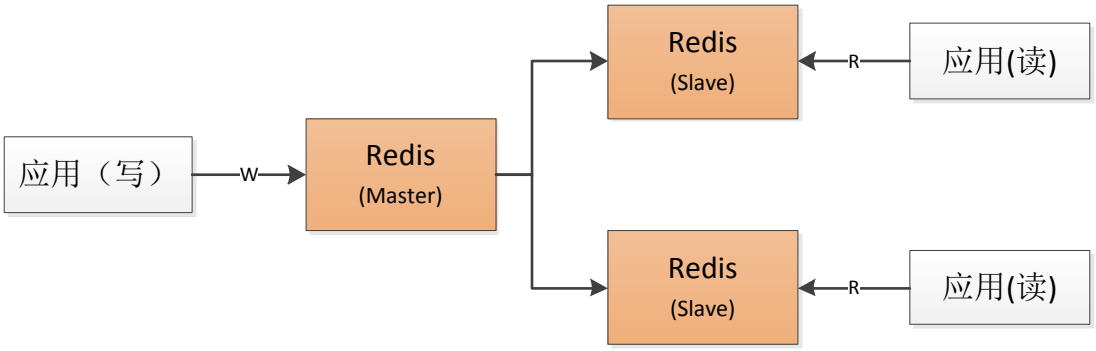
缓存选用 Redis 作为服务组件。

6.4.2 使用场景

(1) 动态数据库
(2) 多服务共享数据
(3) 动态数据缓存，实现业务数据分离。

6.4.3 部署

(1) 一主多从，硬件限制条件下可以单主。
(2) 读写分离。



6.4.4 高可靠性方案

6.4.4.1可靠性需求

业务对 Redis 的可靠性需求有以下几点：

(1) 读写分离，master 写，slave 读

(2) Master 宕机后备机自动接替，写客户端不需要重新配置 IP。

(3) Slave 宕机后，其它 slave 自动接替工作，读客户端不需要重新配置 IP。

6.4.4.2方案类型

目前，Redis 的高可靠性实现主要有 3 种：

(1) keepalived：通过 keepalived 的虚拟 IP，提供主从的统一访问，在主出现问题时，通过 keepalived 运行脚本将从提升为主，待主恢复后先同步后自动变为主，该方案的好处是主从切换后，应用程序不需要知道(因为访问的虚拟 IP 不变)，坏处是引入 keepalived 增加部署复杂性；

(2) sentinel(Redis2.4+)：通过 Sentinel 监控主从实例，自动进行故障恢复，该方案有个缺陷：因为主从实例地址(IP;PORT)是不同的，当故障发生进行主从切换后，应用程序无法知道新地址。

(3) Cluster(Redis3.0+)：Cluster 采用无中心结构，每个节点都保存数据和整个集群的状态；预分好 16384 个桶，根据 CRC16(key) mod 16384 的值，决定将一个 key 放到哪个桶中，每个 Redis 物理结点负责一部分桶的管理，当发生 Redis 节点的增减时，调整桶的分布即可；每个 Redis Node 可以有一个或者多个 Slave。当 Master 挂掉时，选举一个 Slave 形成新的 Master；支持在线增/减节点；不支持多 key 操作。

6.4.4.3方案分析

需求（1）可以通过 Redis 的基本配置简单实现，但是要求在客户端上分别配置好写 Master 和读 Slave 的地址（代替在 Redis.conf 上写死只读配置）。

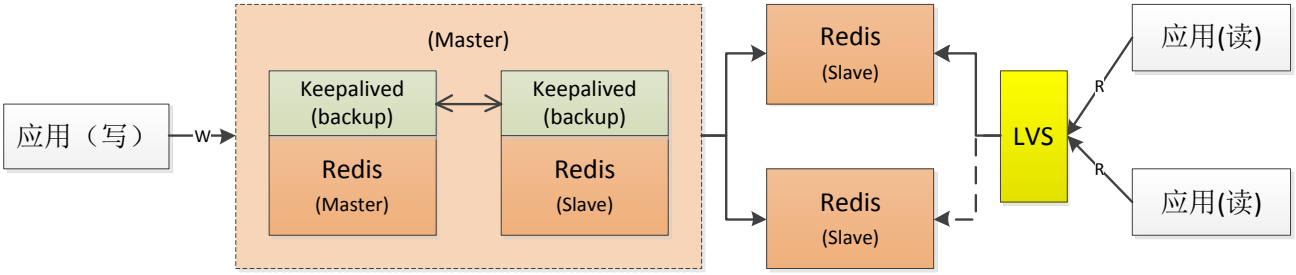
需求（2）中不需要写客户端手工修改配置的部分，可以通过 keepalived 漂移虚 IP 的功能实现；而备机自动接替 Master 的工作，可以通过 keepalived 调用脚本实现，也可以使用 sentinel 和 Cluster 实现。

需求（3）中不需要读客户端手工修改配置的部分，可以通过 keepalived 漂移虚 IP 的功能实现，也可以通过在 Slave 集群前增加负载均衡 LVS 实现。

通过以上分析，基于目前业务的规模，keepalived 的方案更适合当前的业务需求。而今后如果业务规模扩大，需要引入 sentinel 或是 Cluster，也还是需要 keepalived 的虚 IP 漂移。

6.4.4.4实施

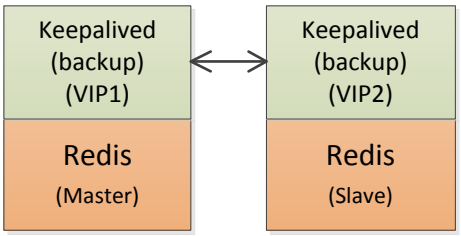
方案（1）：keepalived 主备 RedisMaster + LVS 负载均衡 RedisSlave，见下图：



Redis Master 使用 keepalived 进行主备，对外提供唯一的虚 IP，在 Active 服务器宕机的情况下也能保证负责写的应用能够继续工作；对于 Redis Slave，则使用 LVS 实现集群功能，避免出现 Slave 服务器宕机后读应用无法读取的情况。

该方案配置较为简单，对于 slave 的动态扩展也很方便，缺点则是服务器消耗较多，仅 Redis 服务器就至少需要 4 台，考虑 LVS 本身的主备则还要增加两台服务器。

方案（2）：keepalived 双主（Redis Master + Redis Slave），见下图：



Keepalived 配置双 VIP 实现双主；

VIP1 绑定 Redis master，VIP2 绑定 Redis slave，写应用链接 VIP1，读应用链接 VIP2；

Redis master/ Redis slave 都配置成读写模式；

在单个服务器宕机的情况下，VIP1/VIP2 合一，单机上的 Redis 同时承担读写工作；宕机服务器恢复后，VIP2 漂移到恢复后的宕机服务器上，重新恢复读写分离服务。

该方案结构紧凑，适合小规模部署；但在单机宕机时，读写服务合二为一，会暂时性的影响性能；并且不能动态扩充 slave 服务；同时配置比较复杂。

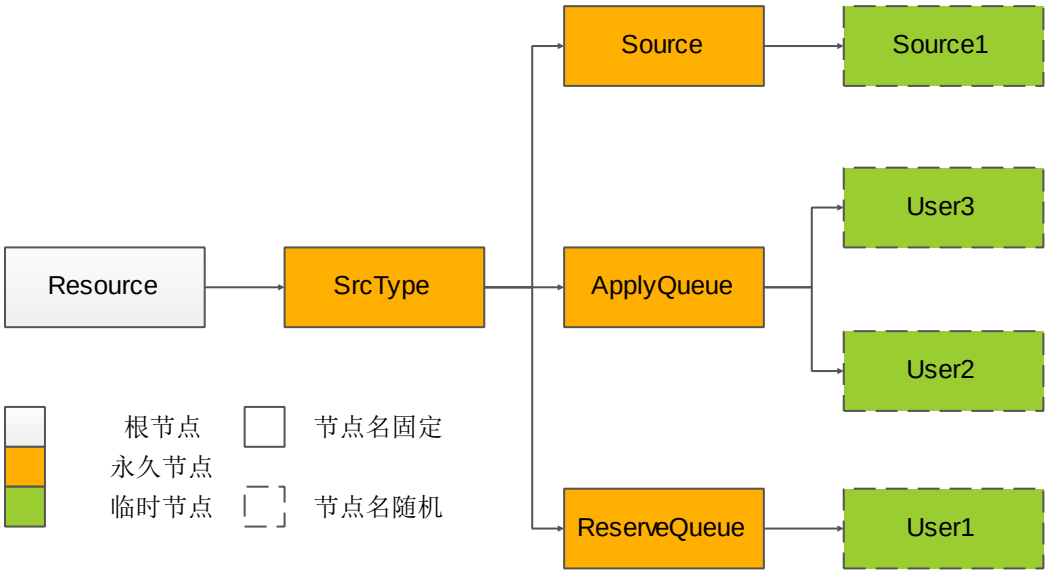
7 资源池

5.0 资源池的基础服务载体是 Zookeeper 分布式应用程序。

5.0 资源池利用 Zookeeper 提供的队列、订阅、自动同步功能，实现了资源的注册、查询、占用、释放等功能。

具体详见：《资源池设计》

7.1 资源节点模型



1） SrcType 为资源类型（CMU/Mpu/Xmpu）。

2） ApplyQueue 为申请资源的先进先出队列。

3） Source 为资源注册的父节点。

4） ReserveQueue 为预占队列，资源占用申请到实际占用之间为异步，所以在当获得占用权后，需要到该队列上创建临时节点，待实际占用到资源后，再从该队列将临时节点删除。

7.2 资源池应用原则

(1) 哪些服务适合上资源池？

A：

a) 提供的服务内聚在单一应用中。

b) 和服务对象无强绑定关系。

c) 有动态伸缩服务规模的需求。

d) 服务的性能可以被计算。

(2) 谁有权创建和修改资源临时节点？

A：

仅资源应用自身有权创建和修改自身的资源临时节点，从而避免出现节点内容和实际能力不符的

情况。

(3) 资源节点如何分类？

A：

节点以资源功能作为分类，若一个模块有两个功能，也分成两个节点来处理。

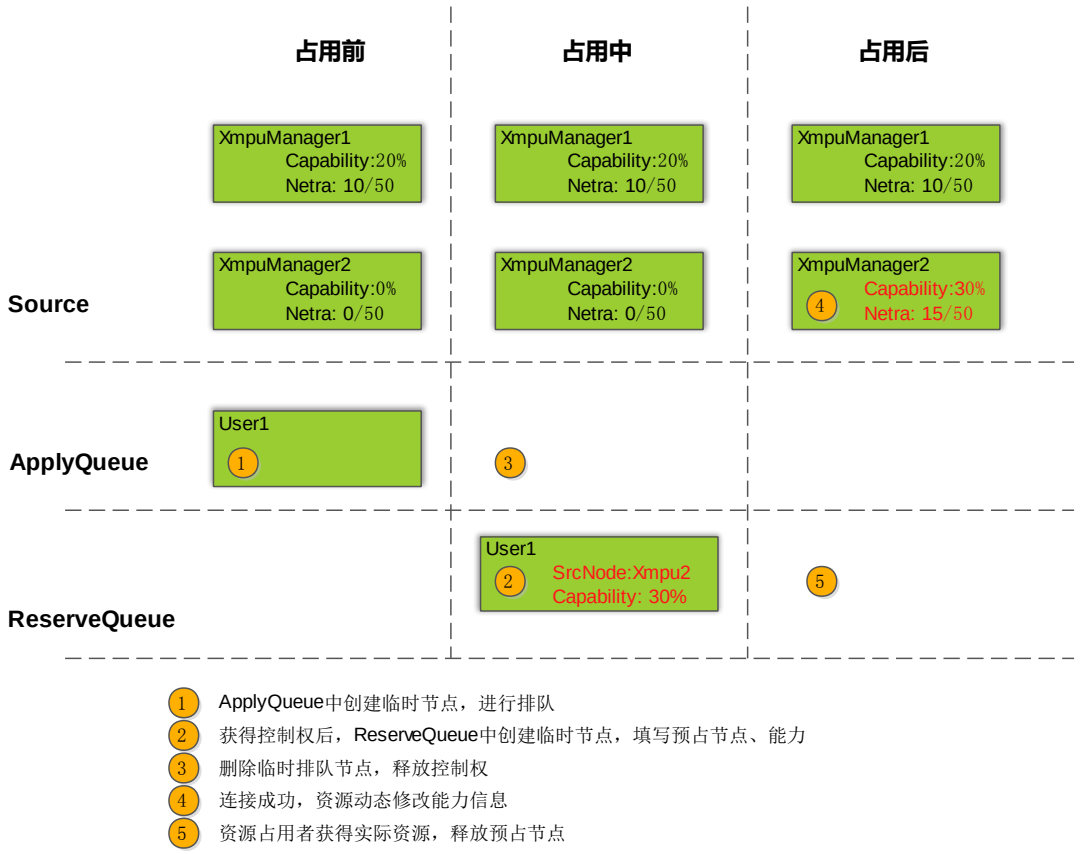
(4) 多个同类资源节点如何命名？

A：

临时节点名称前缀固定为资源类型，以 Zk 顺序创建生成的数字来解决名称问题。（比如 Mpu1，Mpu2，Mpu3）

7.3 占用/释放流程

7.3.1 占用



7.3.2 释放

1） 由资源占用者主动释放资源。

2） 资源根据当前能力状态刷新节点信息。

7.4 节点内容格式

Zookeeper 节点内容以文本文件方式存储，对于资源的提供方和使用方，理论上只需要双方使用同一套解析规则即可实现资源信息的互通。

建议：从通用性及规范性的角度，建议节点内容统一使用 json 格式。

8 开放平台 API 设计

8.1 统一接入

采用 OAuth2.0 流程进行授权。

基于用户信任应用或平台内部系统进行的二次开发，使用密码模式进行认证；租赁平台下第三方应用进行的二次开发，使用授权码模式进行认证。

详见：《开放平台统一接入》，《视讯 API 开发规范》

8.2 接口协议

8.2.1 协议

http 或 https

8.2.2 API 风格

API 主要采用 REST api 风格。

REST 的核心是可编辑的资源及其集合，每个资源或者集合有一个惟一的 URI。系统以资源为中心，构建并提供一系列的 Web 服务。 REST 的基本概念和原则包括：系统上的所有事物都被抽象为资源；每个资源对应唯一的资源标识； 对资源的操作不会改变资源标识本身；所有的操作都是无状态的。

API 主要采用 restful 风格（ restful 风格即 url 所指向的是一个资源如会议、终端等），但并非完全按照 restful 风格（如平台登录、退出等）。客户端通过 http 来完成交互。

客户端通过 http 来完成交互，http 获取方法共有四种：post/get/put/delete，分别对应于 CRUD。API 主要分为获取类(get)与操作类(post)，获取类直接通过 http 将内容以 Json 的形式返回；操作类先回复 ack/nack,具体的内容通过消息推送到 web 端来实现更新。

API 不支持 delete/put 操作，操作类的统一用 post，但是为了更方便的理解对应的操作，在下发的 Json 结构体中的 content 中添加“ _method”来标志对应的操作含义，该字段目前仅能是 post/delete/put 中的一种。

8.2.3 URL 构成规范

http[s]://{domain|ip}/api/{version}/{app}/{resources}/{rc_id}/{subres}…

8.2.4 公用参数

字段	含义
account_token	软件认证 token，必须（oauth2.0）
session_id	登陆成功后返回的 sessionid
access_token	用户认证 token（oauth2.0）

注： session_id 和 access_token 都是用来做用户验证，只需要保留一个就可以。

8.2.5 返回类型

支持两种返回类型:XML 和 JSON。

方式一（ 使用 http request header: Accept）：
GET /user/123 HTTP/1.1
Accept: application/xml //将返回 xml 格式数据
GET /user/123 HTTP/1.1
Accept: application/json //将返回 json 格式数据

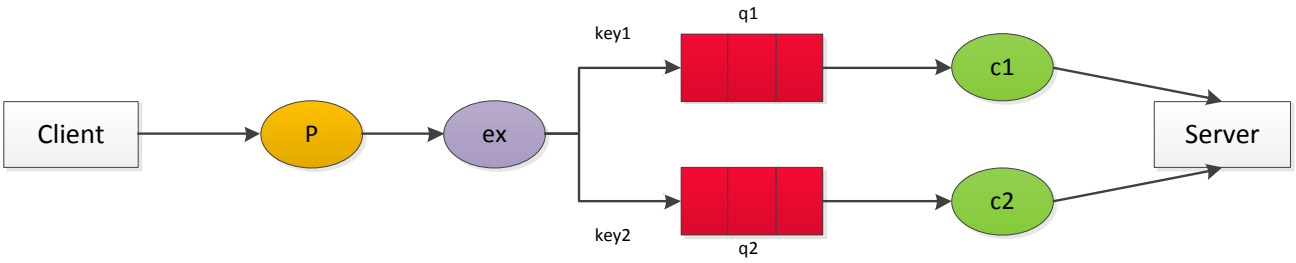
方式二（使用扩展名）：
/user/123.xml 将返回 xml 格式数据
/user/123.json 将返回 json 格式数据

注： 当两种方式冲突时，url 方式优先，当 url 没有后缀时，默认 XML。

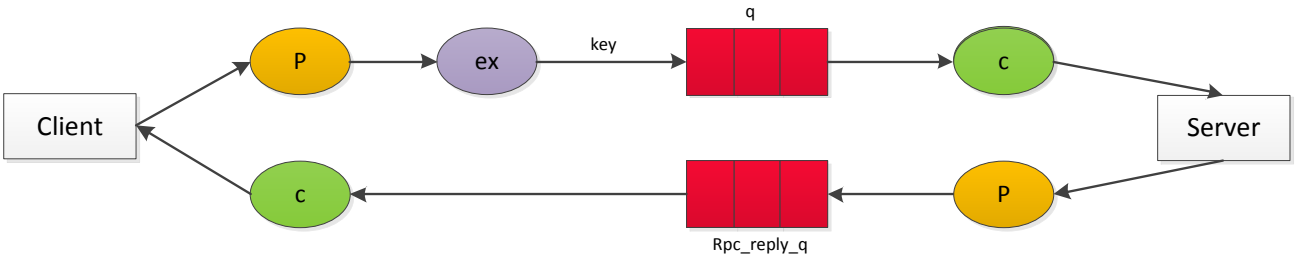
9 RabbitMQ 通信组件

9.1 通信模型

9.1.1 异步通信模型



9.1.2 同步通信模型（RPC）



9.2 命名规则

(1) Queue name -- 模块.功能.q[:自定义字段，一般为 IP 地址]，例如【cmu.conf.q:172.16.179.4】
(2) Key name --模块.功能.k[:自定义字段，一般为 IP 地址]，例如【cmu.conf.k:172.16.179.4】
(3) Exchange name --模块.功能.ex，例如【cmu.conf.ex】

10 遗留问题

(1) PAS 是否应该进一步强化，实现服务对外的弱耦合？
(2) 权限校验是否应统一入口，各模块通过权限校验代理模块进行权限校验，便于权限统一管理，以及今后接入第三方系统？
(3) 需要明确 MODB 多平台域自动选主的实现问题。
(4) 媒体选择是否可以进一步屏蔽底层实现？
(5) 引入 Redis 缓存。

(6) SSO 计划从 memcached 迁移到 redis，时间未定。