



手机百度云端技术架构设计与实践

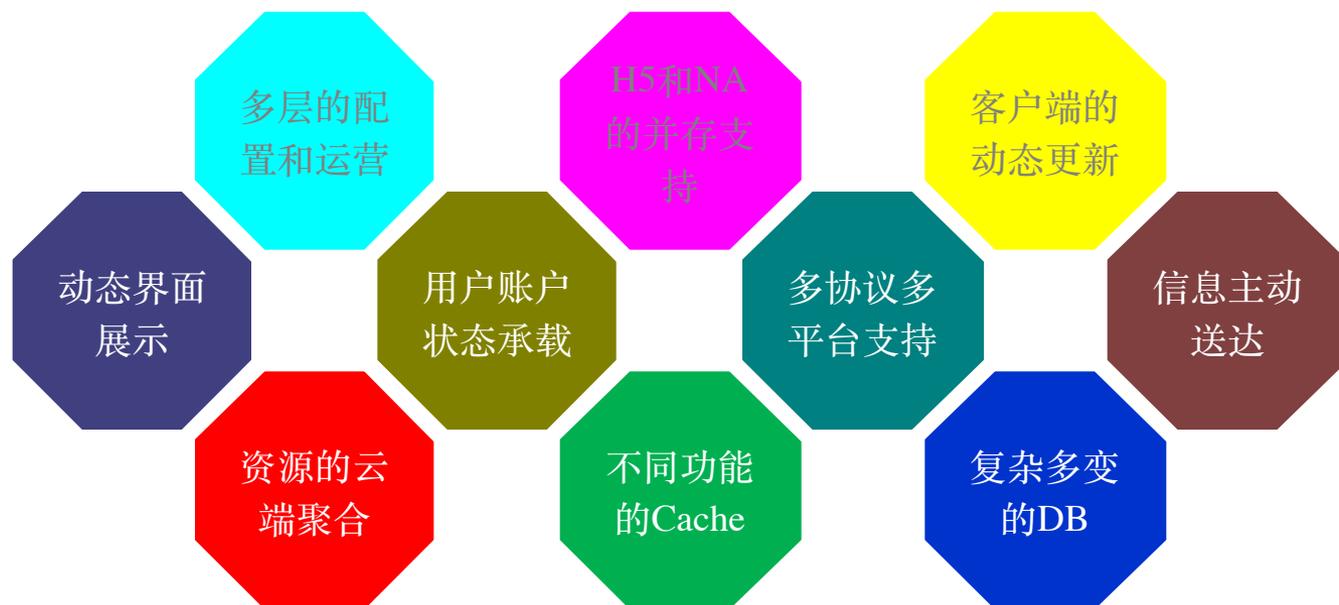
韩超 黎博

目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- 云端开发调试

从Web服务器到超级APP的服务器

- 高并发、大容量
- 高可用性、高可靠性
- 软件灵活性度高、兼容性问题突出
- 需要有面向未来的扩展性
- 软件快速的迭代



从Web服务器到超级APP的服务器

传统的Web网站的构建中，LAMP架构中的PHP+MySQL+缓存已经有了成熟理论和实践。

作为超级APP的服务器工程师，需要继承Web网站的技术底蕴，并针对自身特性做出改进。



目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- 云端开发调试

性能优化

性能是任何服务器设计必须考虑的问题，面对超级APP带来的高QPS、峰值的高并发、不确定的写请求，系统不可能无限制地扩容，软件设计要先行。Web网站性能优化的N种武器，要换一种方式用起来。

- 服务器的容量

服务器端有限的CPU、IO和内存会面对无限的客户端，压力会传递给数据存储、缓存、后端服务，云端的每一个环节都有可能被压垮。

- 用户体验

从前做Web网站，现在做Native客户端。Native并非天然就比Web网站快，因为浏览器和服务器的优化手段已经很成熟。

性能优化：Page缓存

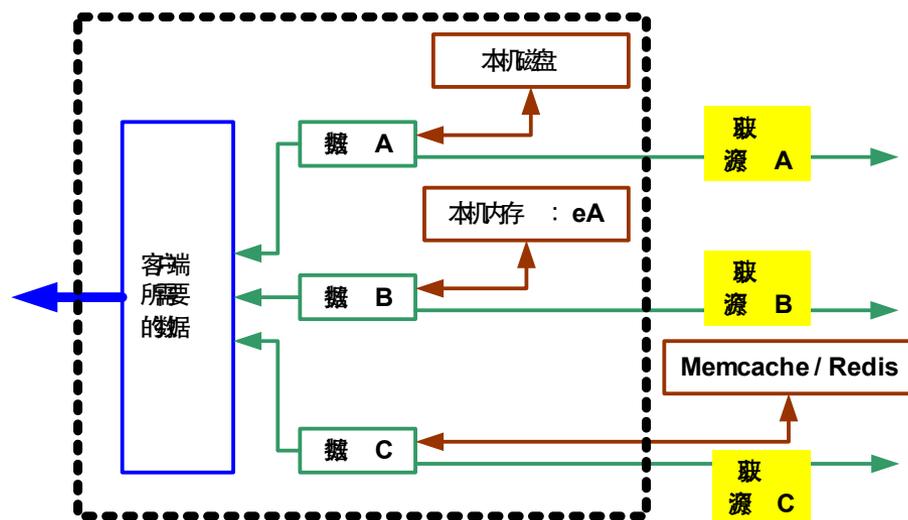
网站设计中存在纯服务端的Page，作为动态内容的缓存。用特定的更新机制进行控制，可减少服务端重复的开销。本机磁盘、本机缓存、分布式缓存均为可采用方式

例如：Smarty是利用本机资源构造页面读缓冲。

```
if (Smarty CacheID 具有) {  
    display CacheID ;  
} else {  
    生成页面 ;  
    display CacheID ;  
}
```

作为具有**资源服务聚合体**性质的服务器，不同的后端资源采用不同的缓存方式：访问频率、复用程度、更新方法。

对于客户端需要的JSON，通常是多种原始数据的组合，下发之前进行json_encode和array_merge。

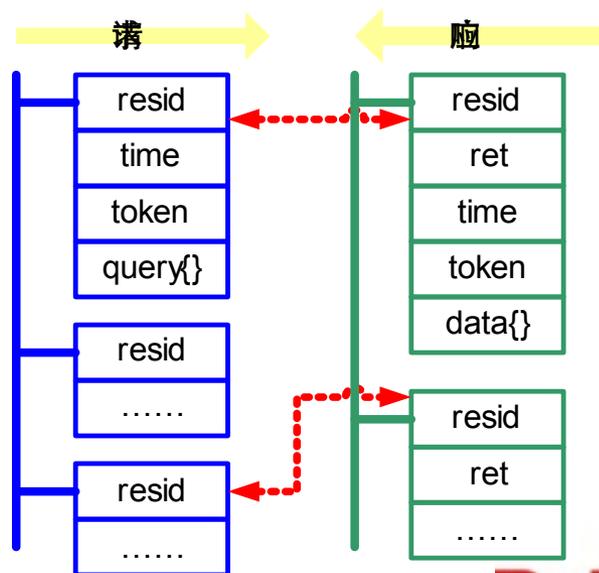


性能优化：缓存协商

Web应用常常通过HTTP的缓存协商避免无变化的数据重复请求和加载。其流程为：respond数据下头部，客户端缓存，下次request带上头，无变化304.

- 时间戳：Last-Modified
=> If-Modify-Since
- 摘要：Etag
=> If-None-Match
- 控制客户端（避免请求）：
Cache-Control / max age
Expires

对客户端需要的JSON等结构化数据，通过增加缓存协商的协议字段，可以获得更稳定、更灵活的效果。例如：单请求多数数据的不同缓存、时间签名双重控制、丰富返回值的语义。



性能优化：请求合并

Web网站中通过请求合并可以显著减少小请求的开销：

- **DataURI**：图片可通过Base64编码包含于页面的标签中，还可以在本地存储后通过<canvas>加载。
- **CSS Sprite**：将页面中诸多的小图片合并成一张大图，通过CSS的background-image和background-position属性定位到具体图片。
- **JS和CSS代码合并**：各自合并，统一合并。

对客户端的请求合并简单而灵活方式处理，在协议中承载二进制架构。静态的资源 and 动态拼接的内容均可合并。

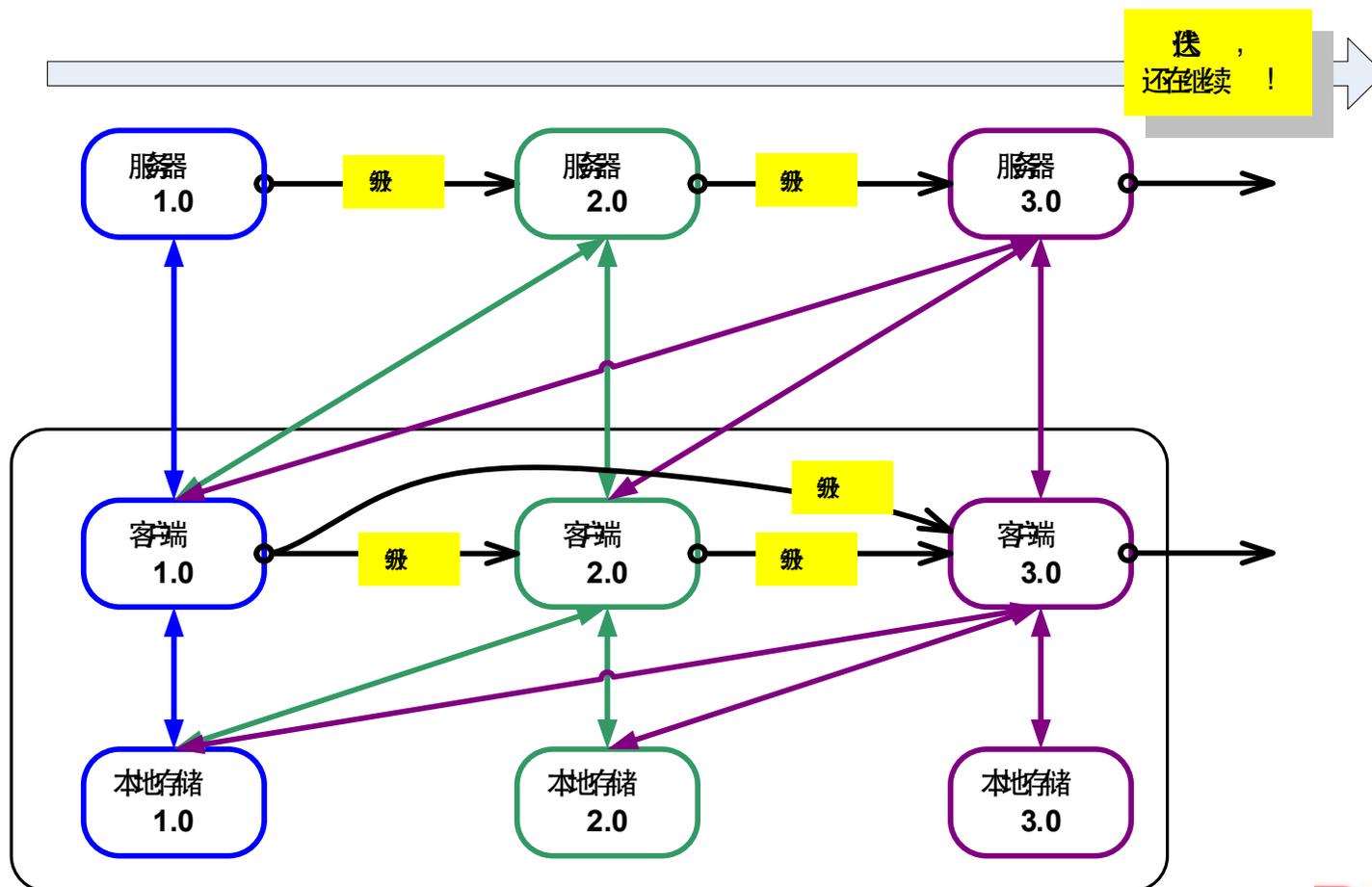
- **简单二进制blob**：通过启示地址、内容长度来存储资源
- **ProtocolBuffer**：用.proto的byte类型表示存储，经过protoc生成各种语言描述pb结构的文件

目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- 云端开发调试

兼容性扩展性

超级APP的服务器面临的开发模型.....

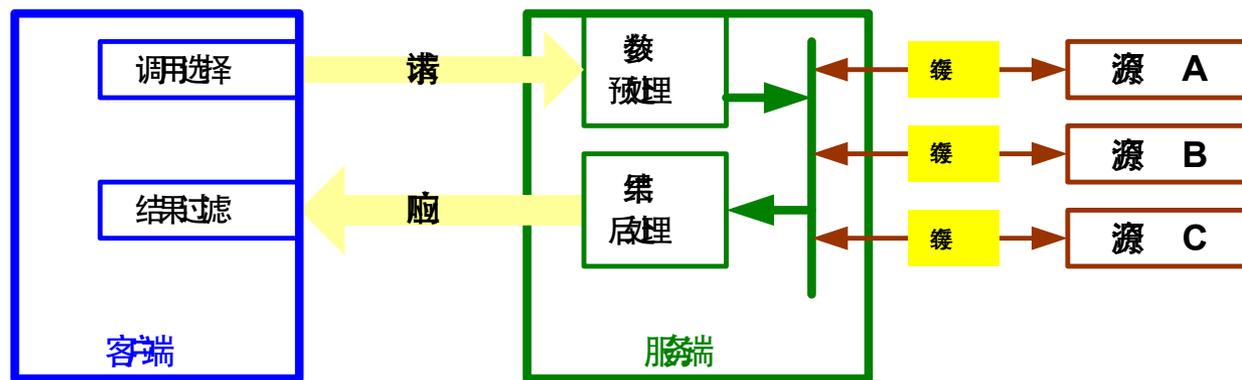


兼容性扩展性

- 移动应用的固有特点是用户手中的客户端的状态**不可控、不确定**；服务器需要支持所有**历史版本的客户端**，且需要有展望未来的能力；
- 有一个**抽象的、稳定的、可扩展的CS协议**；协议当中的数据表示可以具有面向对象的特性（封装、继承、多态）；
- 基于CS协议，服务端向下（向后）兼容，客户端向上（向前）兼容；
- 同时兼容支持多版本多平台的客户端版本；
- 保证客户端本地存储和服务端持久化存储兼容可用；
- 保证动态下发代码有完备的兼容和保护性策略；
- **双刃剑：客户端比浏览器增加很大复杂性，能帮服务器做的事也多。**

兼容性扩展性：多版本多平台

- 服务端被动兼容的条件是客户端的特性可知，通过客户端自身的信息进行兼容，例如：操作系统、版本号、渠道等；
- 代码分支方式：把稳定基础性功能分成不同的模块，找到逻辑上需要区分的地方，由上层代码对基础功能组合调用；
- 资源获取：调整后端请求、持久化存储、缓存的获取顺序，便于排列组合和再处理；
- 特制化逻辑：了解客户端的现实分布和业务情况，利用参数预处理筛选功能，对结果通过用个性化参数进行调整、过滤和合并；
- 缓存策略：提高Cache的命中率和Cache使用效率；



兼容性扩展性：存储逻辑

- 服务端的持久化存储：当请求中带有写入逻辑时，MySQL和Redis持久化存储将会更新。面对多种客户端、协议升级、逻辑错误，很可能出现特殊和异常数据，且写入造成的并发IO压力很大。
- 客户端的本地存储：客户端天然具有丰富的本地存储能力，有些时候，服务端的逻辑可以影响客户端本地存储，一旦发生错误，可能引起不可挽救的后果。

Web网站的复杂性较低：

- 用户对刷新、清除缓存等操作，有较强的认知
- LocalStorage是典型KV缓存，基本结构比较简单；
- AppCache是指定目标的存储；
- 浏览器DB的用途不是很广泛；

超级APP的服务器要能做什么：

- 充分利用分布式缓存系统，优化读、写的操作逻辑；
- 健全协议，确保客户端的存储可以被忽略和清除；
- 转换思维方式，利用客户端能力减少服务端持久化存储；

兼容性扩展性：动态代码

动态代码是从服务器下发的在客户端执行的代码，运行于客户端，却由服务器动态维护。软件架构中一旦具有由服务端向客户端下发动态代码功能，服务端将面临多维的复杂度：首先，动态代码自身的更新要可维护；其次，将面对多维客户端代码CS协议。

Web网站动态代码特点：

- JavaScript大部分时候按照文件维度全量更新；
- 可以通过附加于url的query选择是否利用浏览器端的缓存；
- 通过LocalStorage可存储JavaScript代码，并用版本号控制，自行更新代码

如何处理客户端动态代码：

- 全量更新要靠拉取CS拉取协议在后台进行，不可在运行时进行；
- 下载和安装流程要标准化，不能自更新等失去控制权的方式；
- 在DB中建表，维护客户端的查询条件，控制动态代码的更新；
- 隔离客户端和动态代码的CS协议，简化流程；

目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- 云端开发调试

云端可控性

由于客户端APP的固有特点，不可强制要求用户手中的客户端版本，从超级APP的功能上，要突破此瓶颈，在客户端不升级的情况下，实现云端可控的特性。

- 动态展示： 根据需要将最新的数据、样式、界面送达给用户；
- 动态事件和行为： 服务端产生的事件可送达客户端（客户端拉取或服务端推送），可控制客户端的各种行为；
- 动态的配置和策略： 服务器上具有动态配置和策略，客户端通过获得之后，根据自身的情况进行实施；

云端可控性



灵活性增加、复杂性也增加

服务器如果处理超级APP的动态化

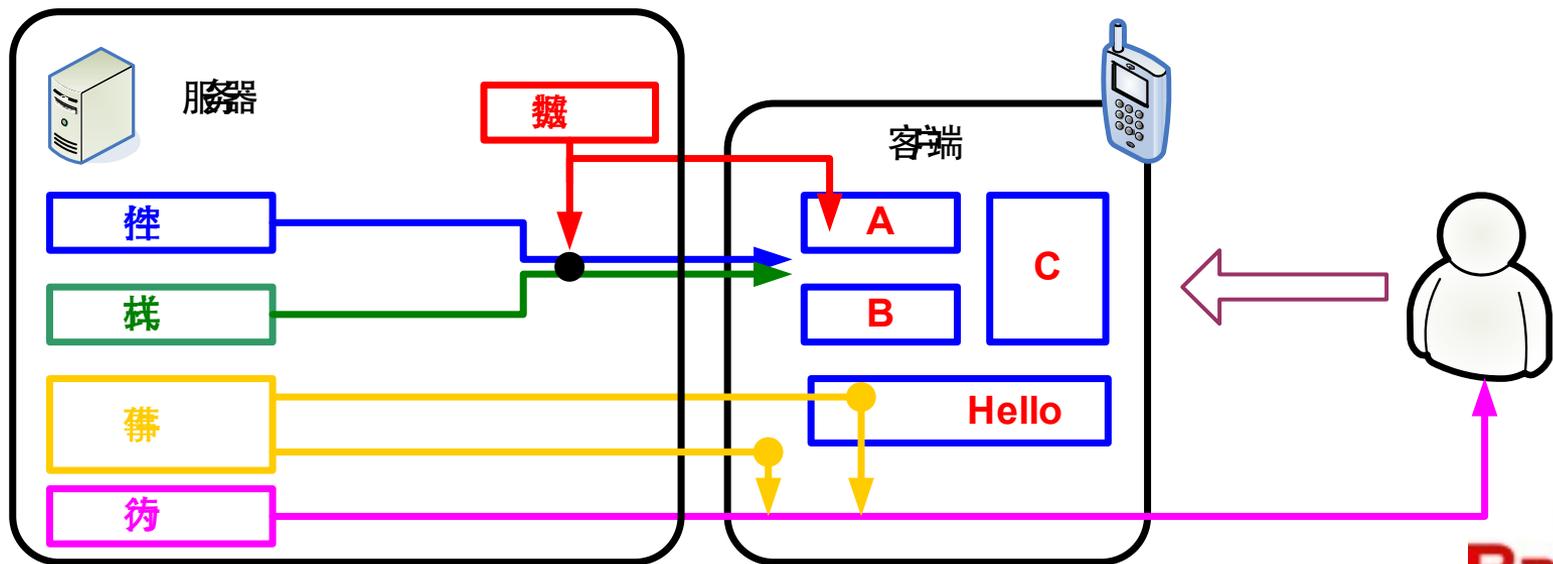
- 无模板：处理的仅有动态内容数据；
- 静态模板：处理模板的选择、继承，匹配动态内容数据；
- 动态模板：处理模板的下发，匹配动态数据；
- 动态代码：处理代码的下发、更新和管理；

或繁或简，一个自定义的浏览器！

云端可控性

自定义浏览框架的几大要素：

- **语言**：用来能够描述界面及其附属内容，需要支持层次结构和属性，客户端跨平台可以各自进行进行解析；
- **控件**：界面的承载体，由客户端实现，服务端下发内容；
- **样式**：动画、特效附属内容的抽象化，服务端配置指定；
- **事件**：输入的抽象化，客户端和服务端均有；
- **行为**：互动能力的体现，要由客户端定义，服务端下发控制；



目录

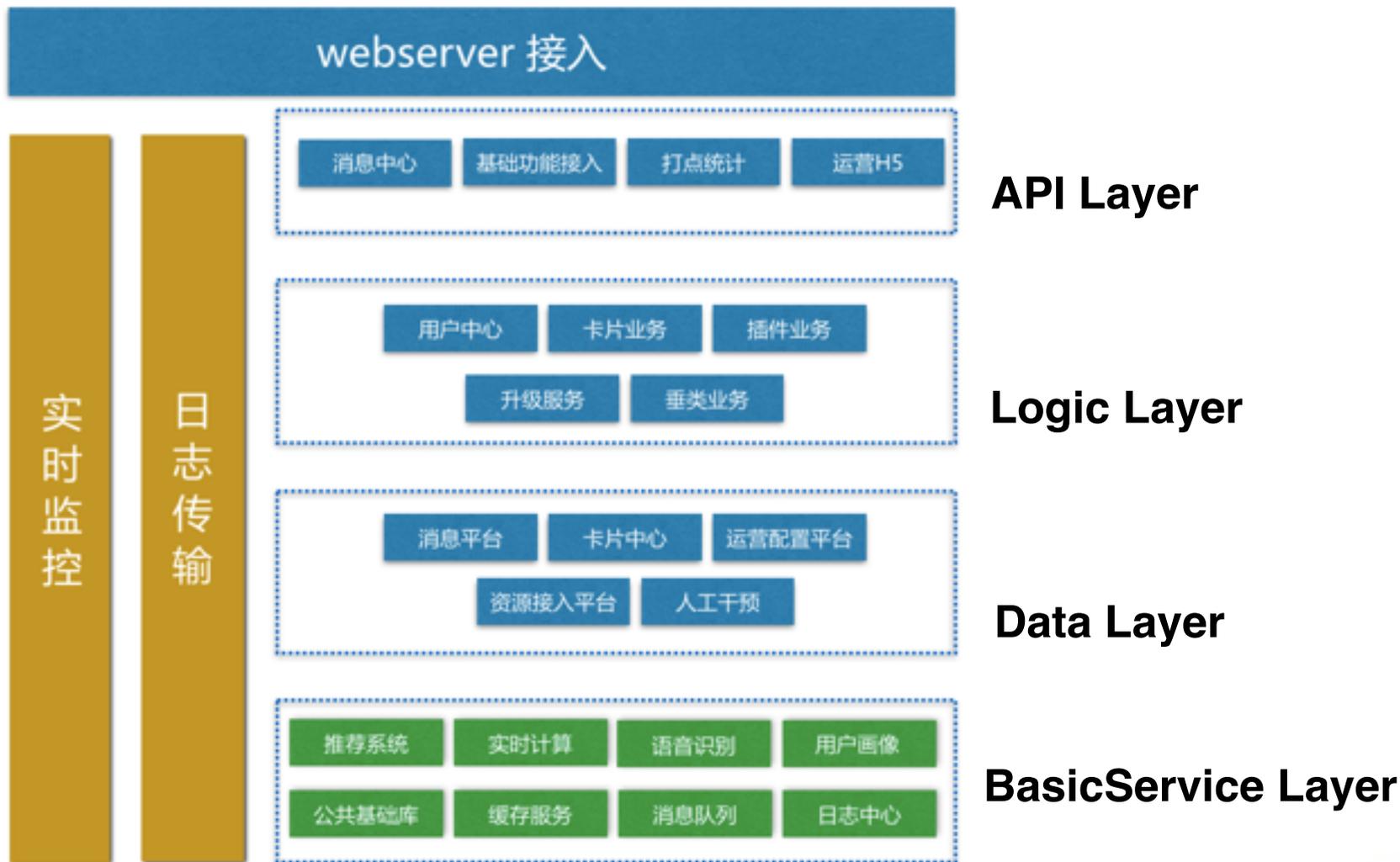
- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- 云端开发调试

手机百度业务特性

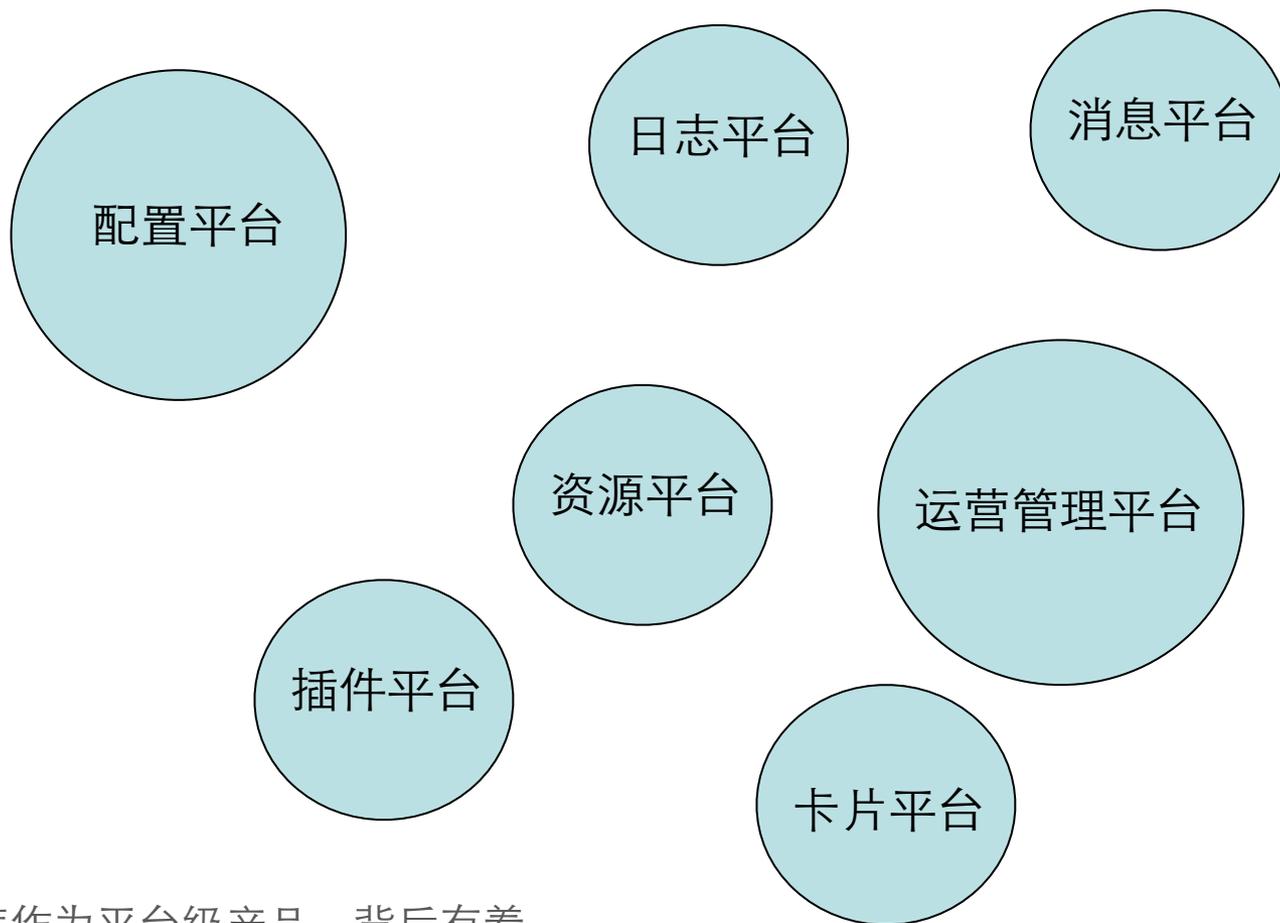


- 多系统/多版本/生命周期长
- 高并发/请求量不均
- 连接资源众多/稳定性不可控
- 需求复杂/迭代频繁

手机百度云端技术架构



平台化解决方案

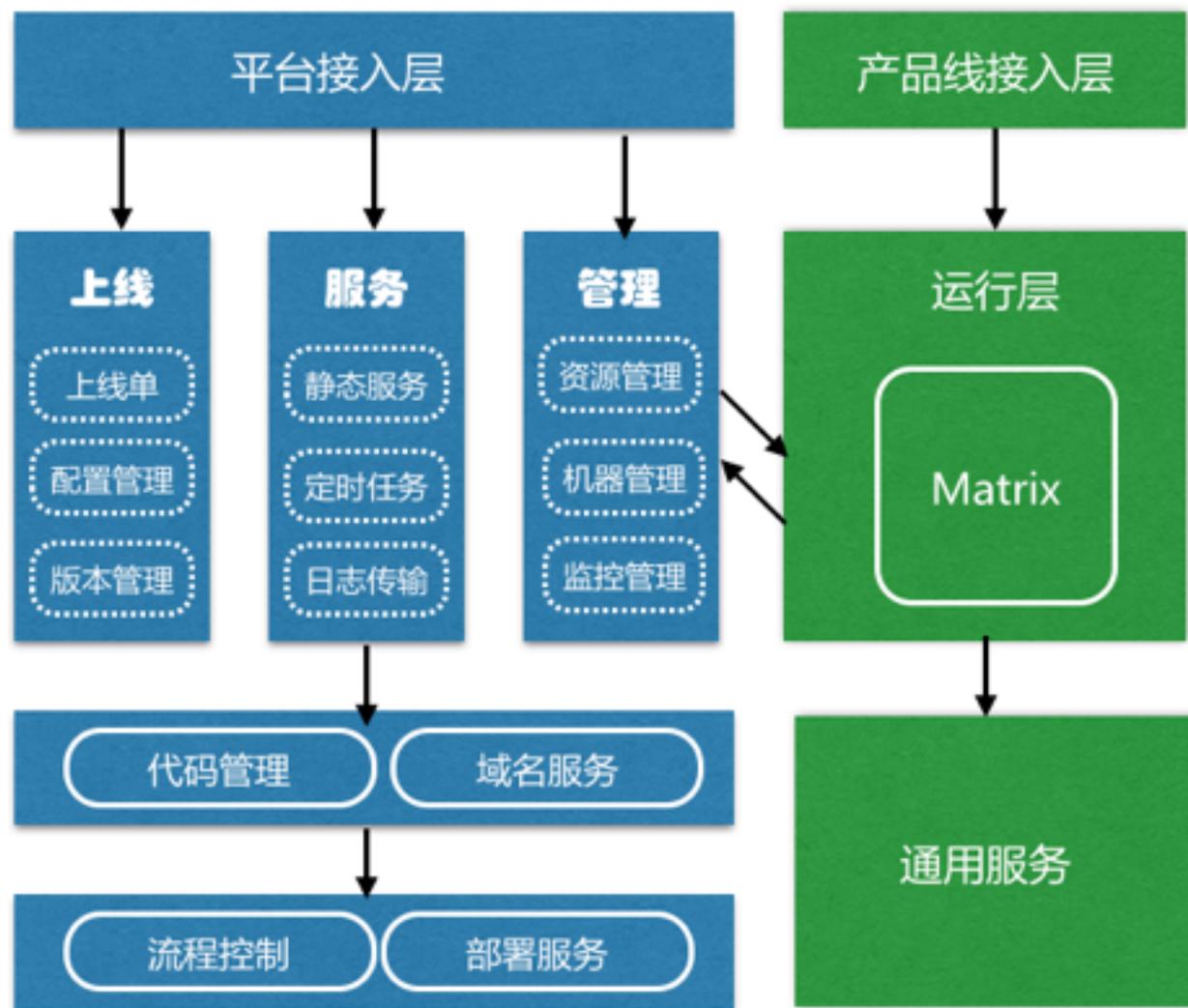


手机百度作为平台级产品，背后有着众多强大的平台化服务的支持

目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- **运行环境runtime**
- 云端服务开发调试

运行环境runtime



运行环境runtime

特性:

- 混部: 有效解决资源利用率问题
- 解耦: 方便故障响应处理
- 统一: 统一协调管理机器资源, 分钟级别扩容

目录

- 从Web服务器到超级APP的服务器
- 性能优化
- 兼容性扩展性
- 云端可控性
- 手机百度云端架构
- 运行环境runtime
- **云服务开发调试**

云服务开发调试

开发：

- 联调过程：工具集合百宝箱
- 日志追踪：NOTICE/DEBUG/TRACE 详细日志

上线：

- 灰度上线：预览机/首台/单机房/全流量
- 监控系统：QPS/SAL/错误率、统计报表、错误预警

云服务开发调试

工具化开发、联调、测试



- 云/端 API 联调测试：上传参数配置，下发数据解析
- 常用工具：加密解密、Urlencode、格式化数据
- 沙盒工具：提供桩数据联调

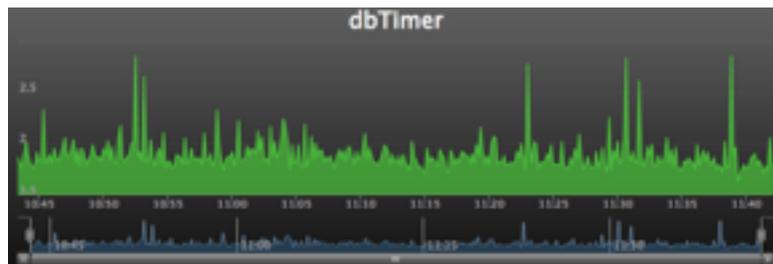
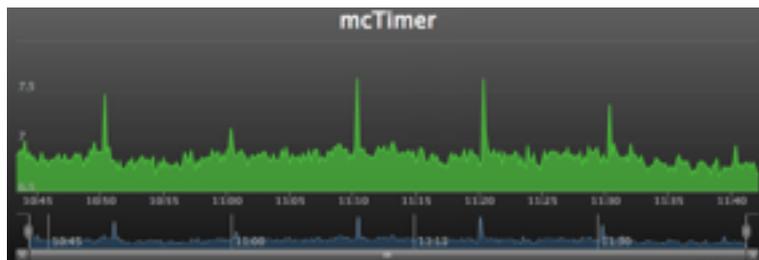
云服务开发调试

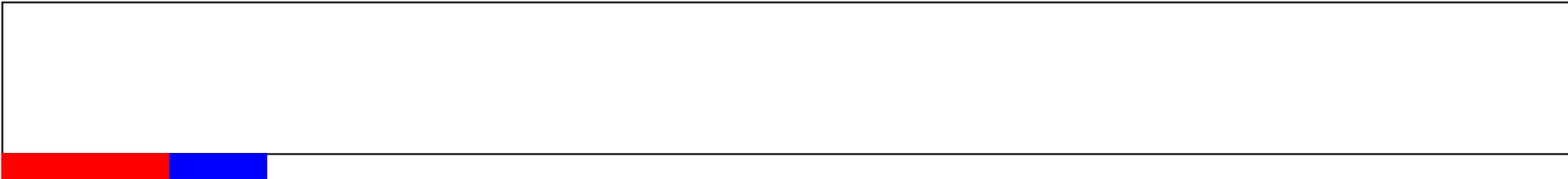
灰度上线：预览机/首台/单机房/全流量

| 上线步骤 | 操作 | 状态 |
|------|---|------|
| 准备阶段 | <input type="button" value="上线"/> | 部署成功 |
| 预览机 | <input type="button" value="上线"/> 执行时间：24秒 | 部署成功 |
| 单台 | <input type="button" value="上线"/> 执行时间：67秒 | 部署成功 |
| 单边 | <input type="button" value="上线"/> 执行时间：221秒 | 部署成功 |
| 全部 | <input type="button" value="上线"/> 执行时间：351秒 | 部署成功 |

云服务开发调试

监控系统：QPS/SAL/错误率、统计报表、错误预警





Q&A?