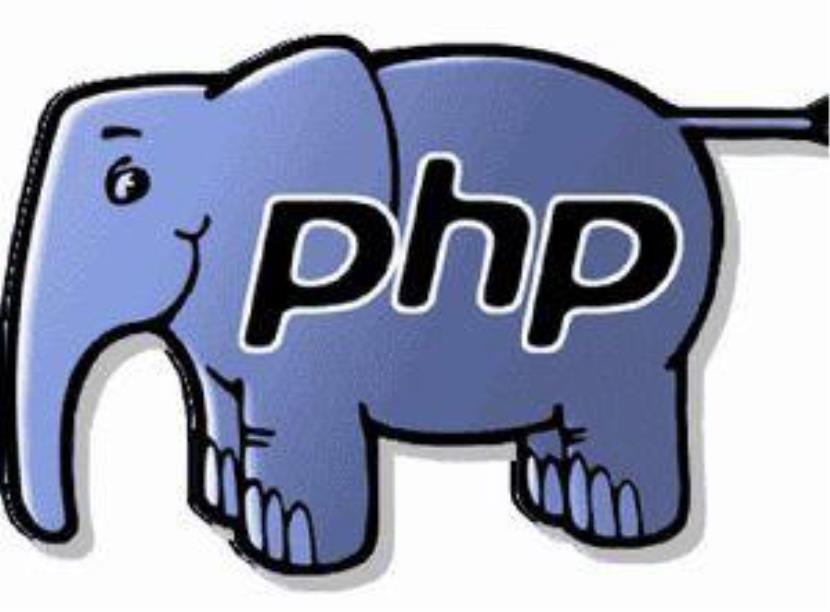
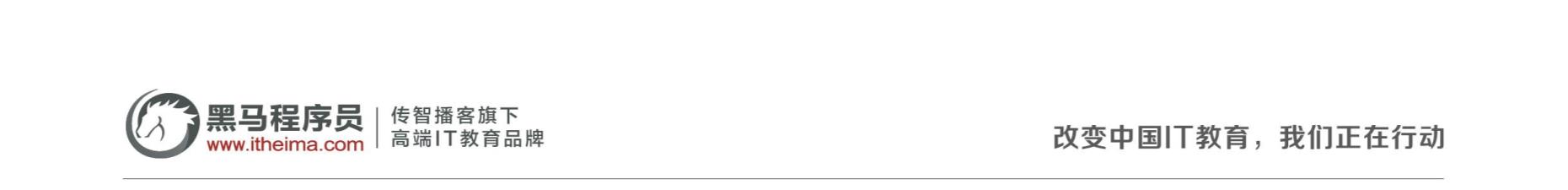
PHP 专题拓展

上海校区就业部 PHP 组——薛杰

2017 年 5 月

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



Websocket

一、Websocket 使用场景

1.社交聊天

最著名的就是微信，QQ，这一类社交聊天的 app。这一类聊天 app 的特点是低延迟，

高即时。即时是这里面要求最高的，如果有一个紧急的事情，通过 IM 软件通知你，假设网

络环境良好的情况下，这条 message 还无法立即送达到你的客户端上，紧急的事情都结束了，

你才收到消息，那么这个软件肯定是失败的。

2.弹幕

典型的例子就是 A 站和 B 站了。确实，他们的弹幕一直是一种特色。而且弹幕对于一

个视频来说，很可能弹幕才是精华。发弹幕需要实时显示，也需要和聊天一样，需要即时。

3.多玩家游戏

4.协同编辑

现在很多开源项目都是分散在世界各地的开发者一起协同开发，此时就会用到版本控制

系统，比如 Git，SVN 去合并冲突。但是如果有一份文档，支持多人实时在线协同编辑，那

么此时就会用到比如 WebSocket 了，它可以保证各个编辑者都在编辑同一个文档，此时不

需要用到 Git，SVN 这些版本控制，因为在协同编辑界面就会实时看到对方编辑了什么，谁

在修改哪些段落和文字。

5.股票基金实时报价

金融界瞬息万变——几乎是每毫秒都在变化。如果采用的网络架构无法满足实时性，那

么就会给客户带来巨大的损失。几毫秒钱股票开始大跌，几秒以后才刷新数据，一秒钟的时

间内，很可能用户就已经损失巨大财产了。

6.体育实况更新

全世界的球迷，体育爱好者特别多，当然大家在关心自己喜欢的体育活动的时候，比赛

实时的赛况是他们最最关心的事情。这类新闻中最好的体验就是利用 Websocket 达到实时的

更新！

7.视频会议/聊天

视频会议并不能代替和真人相见，但是他能让分布在全球天涯海角的人聚在电脑前一起

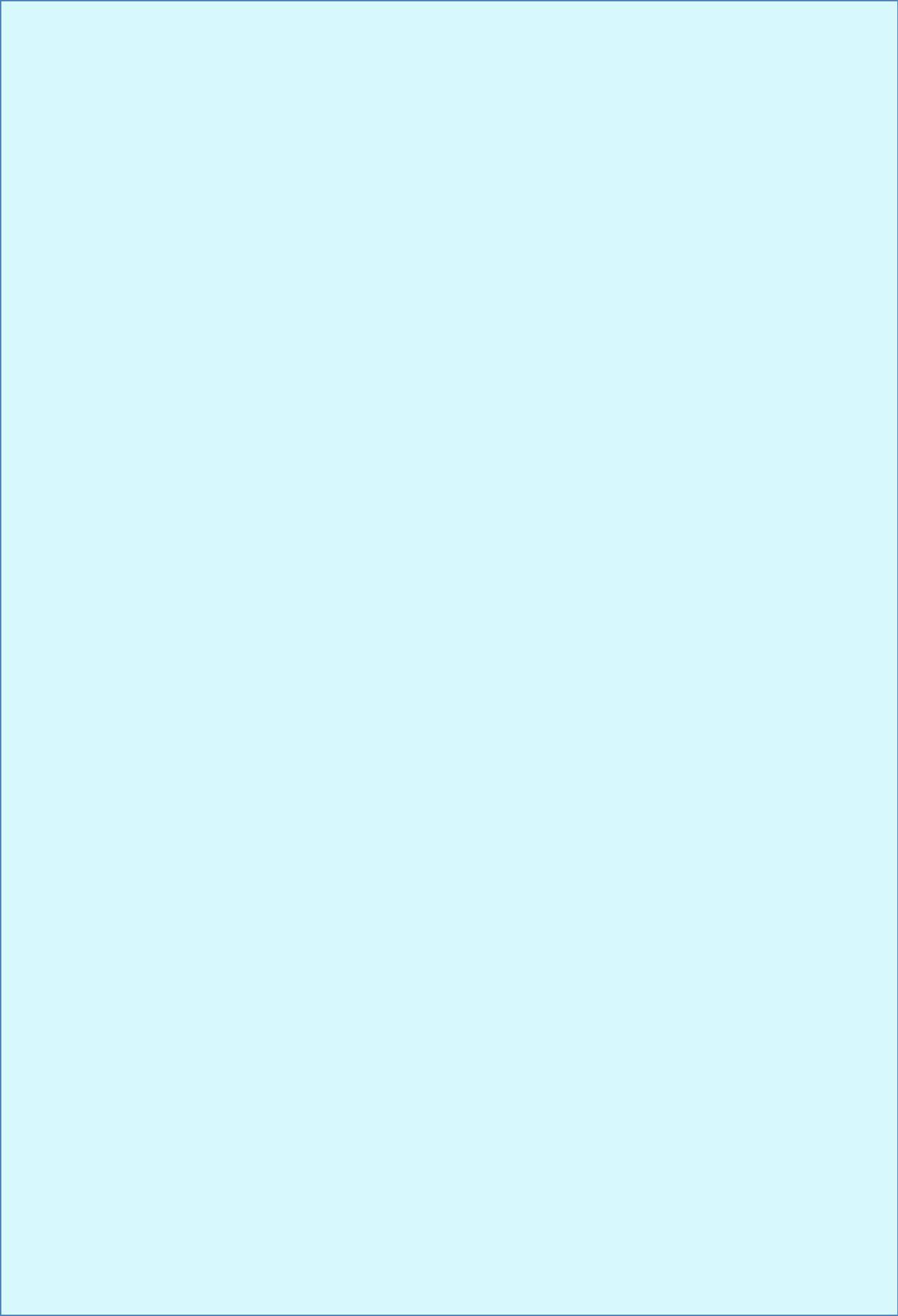
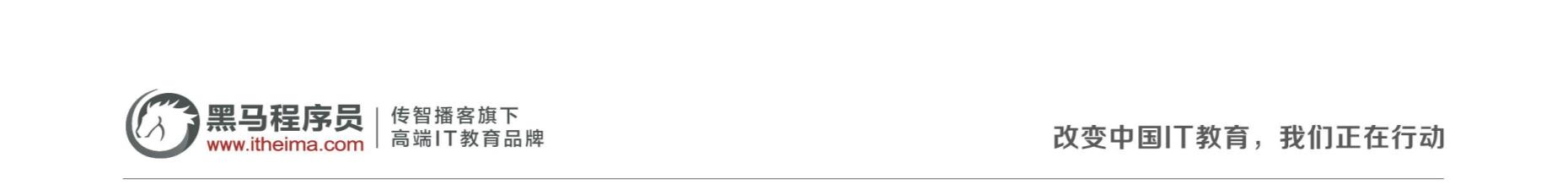
开会。既能节省大家聚在一起路上花费的时间，讨论聚会地点的纠结，还能随时随地，只要

有网络就可以开会。

8.基于位置的应用

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



越来越多的开发者借用移动设备的 GPS 功能来实现他们基于位置的网络应用。如果你

一直记录用户的位置(比如运行应用来记录运动轨迹)，你可以收集到更加细致化的数据。

9.在线教育

在线教育近几年也发展迅速。优点很多，免去了场地的限制，能让名师的资源合理的分

配给全国各地想要学习知识的同学手上，Websocket 是个不错的选择，可以视频聊天、即时

聊天以及其与别人合作一起在网上讨论问题...

10.智能家居

智能设备的状态必须需要实时的展现在手机 app 客户端上，毫无疑问选择了 Websocket。

二、websocket 协议原理

Websocket 是应用层第七层上的一个应用层协议，它必须依赖 HTTP 协议进行一次握

手 ，握手成功后，数据就直接从 TCP 通道传输，与 HTTP 无关了。

Websocket 的数据传输是 frame 形式传输的，比如会将一条消息分为几个 frame，按照

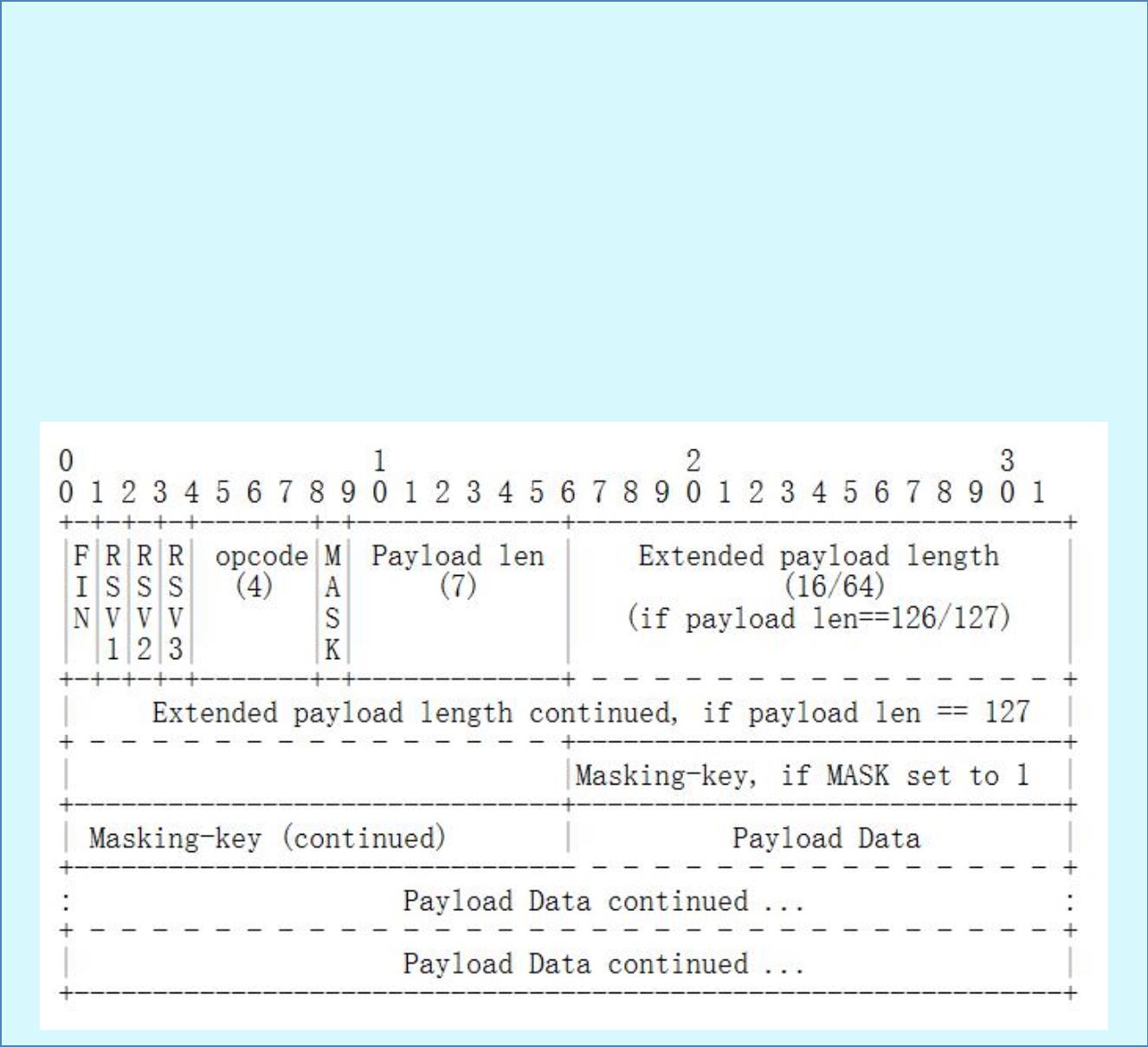
先后顺序传输出去。这样做会有几个好处：

1 大数据的传输可以分片传输，不用考虑到数据大小导致的长度标志位不足够的情况。

2 和 http 的 chunk 一样，可以边生成数据边传递消息，即提高传输效率。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



三、websocket 和 socket 的区别和联系

首先，Socket 其实并不是一个协议。它工作在 OSI 模型会话层（第 5 层），是为了方

便大家直接使用更底层协议（一般是 TCP 或 UDP ）而存在的一个抽象层。Socket 是对 TCP/IP

协议的封装，Socket 本身并不是协议，而是一个调用接口(API)

Socket 通常也称作”套接字”，用于描述 IP 地址和端口，是一个通信链的句柄。网络

上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个

Socket，一个 Socket 由一个 IP 地址和一个端口号唯一确定。应用程序通常通过”套接字”

向网络发出请求或者应答网络请求。

Socket 在通讯过程中，服务端监听某个端口是否有连接请求，客户端向服务端发送连接

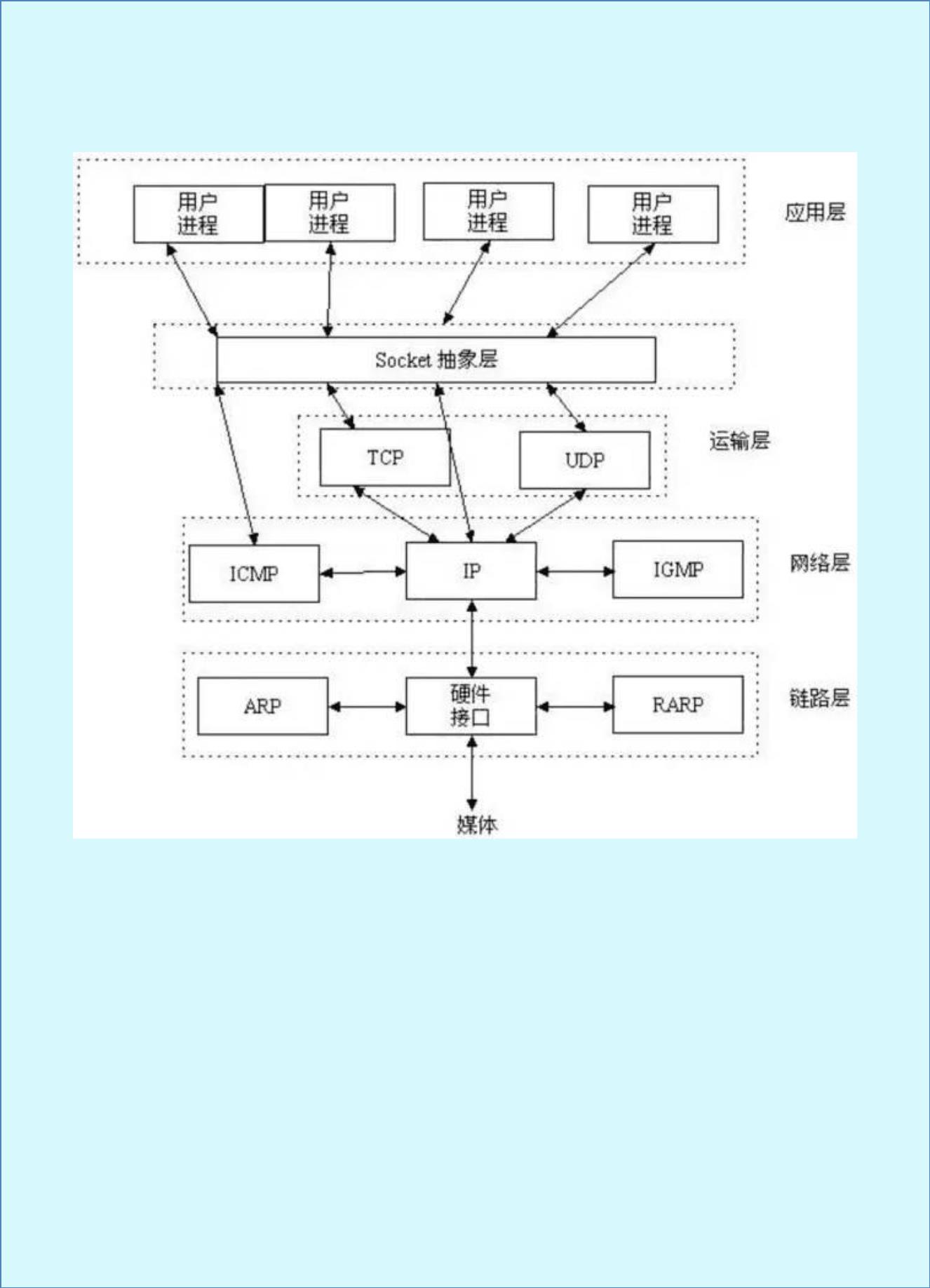
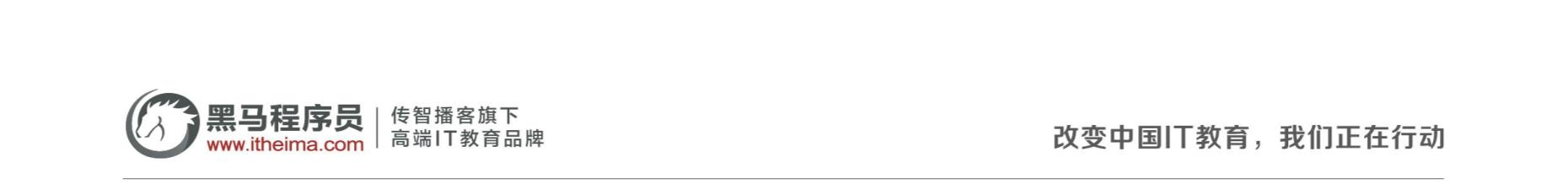
请求，服务端收到连接请求向客户端发出接收消息，这样一个连接就建立起来了。客户端和

服务端也都可以相互发送消息与对方进行通讯，直到双方连接断开。

所以基于 WebSocket 和基于 Socket 都可以开发出 IM 社交聊天类的 app

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



四、有哪些 WebSocket 和 Socket 的开源框架

Socket 开源框架有：CocoaAsyncSocket，socketio/socket.io-client-swift

WebSocket 开源框架有:facebook/SocketRocket，tidwall/SwiftWebSocket

PHP 中九大缓存技术

一、全页面静态化缓存

就是将页面全部生成 html 静态页面，用户访问时直接访问的静态页面，而不会去走 php

服务器解析的流程。此种方式，在 CMS 系统中比较常见，比如 dedecms；

一种比较常用的实现方式是用输出缓存：

Ob\_start()

\*\*\*\*\*\*要运行的代码\*\*\*\*\*\*\*

$content = Ob\_get\_contents();

\*\*\*\*将缓存内容写入 html 文件\*\*\*\*\*

Ob\_end\_clean();

二、页面部分缓存

该种方式，是将一个页面中不经常变的部分进行静态缓存，而经常变化的块不缓存，最

后组装在一起显示；可以使用类似于 ob\_get\_contents 的方式实现，也可以利用类似 ESI 之

类的页面片段缓存策略，使其用来做动态页面中相对静态的片段部分的缓存(ESI 技术：ESI

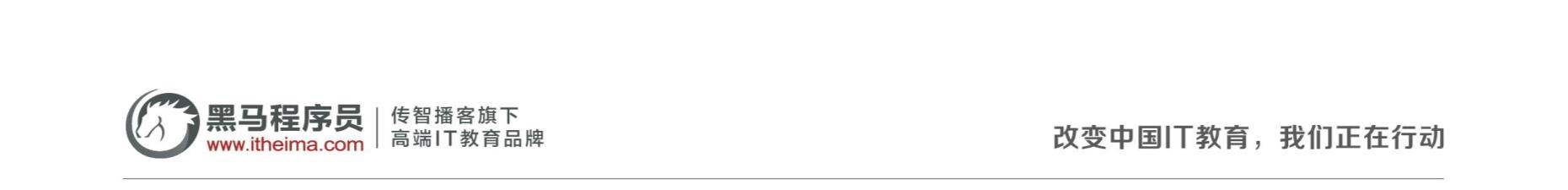
是一种数据缓冲/缓存服务器，它提供将 Web 网页的部分（这里指页面的片段）进行缓冲/

缓存的技术及服务)。

该种方式可以用于如商城中的商品页；

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



三、数据缓存

顾名思义，就是缓存数据的一种方式；比如，商城中的某个商品信息，当用商品 id 去

请求时，就会得出包括店铺信息、商品信息等数据，此时就可以将这些数据缓存到一个 php

文件中，文件名包含商品 id 来建一个唯一标示；下一次有人想查看这个商品时，首先就直

接调这个文件里面的信息，而不用再去数据库查询；其实缓存文件中缓存的就是一个 php

数组之类；

Ecmall 商城系统里面就用了这种方式；

四、查询缓存

其实这跟数据缓存是一个思路，就是根据查询语句来缓存；将查询得到的数据缓存在一

个文件中，下次遇到相同的查询时，就直接先从这个文件里面调数据，不会再去查数据库；

但此处的缓存文件名可能就需要以查询语句为基点来建立唯一标示；

按时间变更进行缓存

其实，这一条不是真正的缓存方式；上面的 2、3、4 的缓存技术一般都用到了时间变更

判断；就是对于缓存文件需要设一个有效时间，在这个有效时间内，相同的访问才会先取缓

存文件的内容，但是超过设定的缓存时间，就需要重新从数据库中获取数据，并生产最新的

缓存文件；比如，我将我们商城的首页就是设置 2 个小时更新一次；

五、按内容变更进行缓存

这个也并非独立的缓存技术，需结合着用；就是当数据库内容被修改时，即刻更新缓存

文件；

比如，一个人流量很大的商城，商品很多，商品表必然比较大，这表的压力也比较重；

就可以对商品显示页进行页面缓存；

当商家在后台修改这个商品的信息时，点击保存，我们同时就更新缓存文件；那么，买

家访问这个商品信息时，实际上访问的是一个静态页面，而不需要再去访问数据库；

试想，如果对商品页不缓存，那么每次访问一个商品就要去数据库查一次，如果有 10

万人在线浏览商品，那服务器压力就大了；

六、内存式缓存

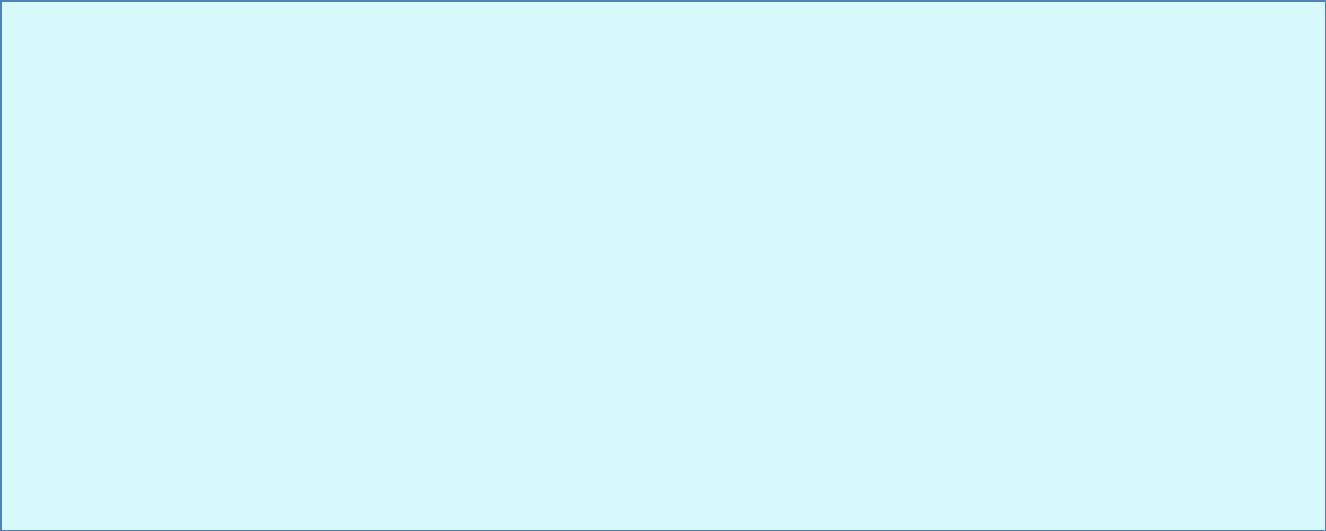
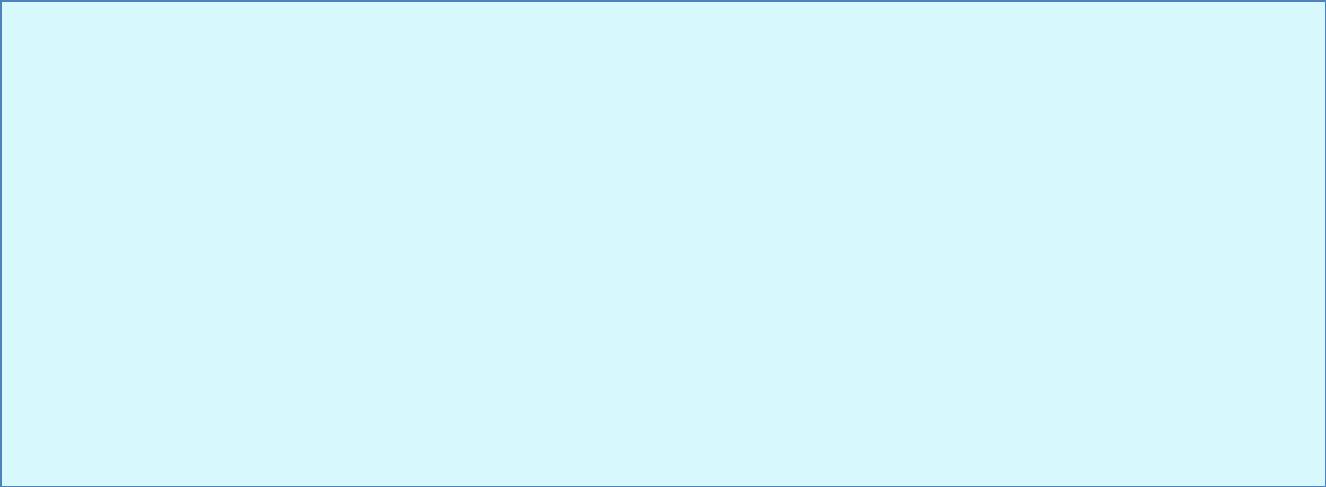
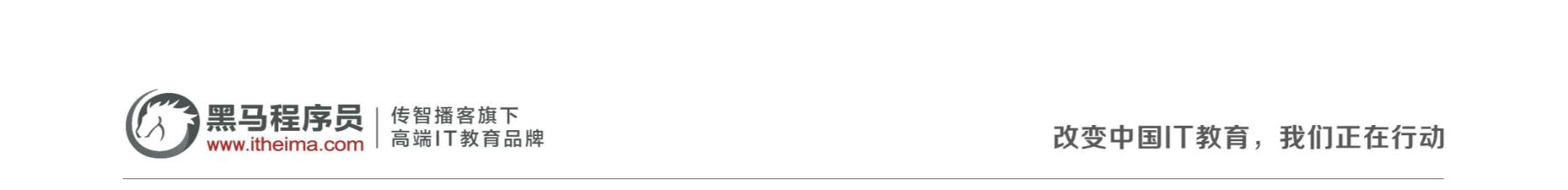
提到这个，可能大家想到的首先就是 Memcached；memcached 是高性能的分布式内存

缓存服务器。 一般的使用目的是，通过缓存数据库查询结果，减少数据库访问次数，以提

高动态 Web 应用的速度、 提高可扩展性。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



它就是将需要缓存的信息，缓存到系统内存中，需要获取信息时，直接到内存中取；比

较常用的方式就是 key–>value 方式；

<?php

$memcachehost = '192.168.6.191';

$memcacheport = 11211;

$memcachelife = 60;

$memcache = new Memcache;

$memcache->connect($memcachehost,$memcacheport) or die ("Could not

connect");

$memcache->set('key','缓存的内容');

$get = $memcache->get($key);

//获取信息

?>

七、apache 缓存模块

apache 安装完以后，是不允许被 cache 的。如果外接了 cache 或 squid 服务器要求进行

web 加速的话，就需要在 htttpd.conf 里进行设置，当然前提是在安装 apache 的时候要激活

mod\_cache 的模块。

安装 apache 时：./configure –enable-cache –enable-disk-cache –enable-mem-cache

八、php APC 缓存扩展

php 有一个 APC 缓存扩展，windows 下面为 php\_apc.dll，需要先加载这个模块，然后是在

php.ini 里面进行配置：

[apc]

extension=php\_apc.dll

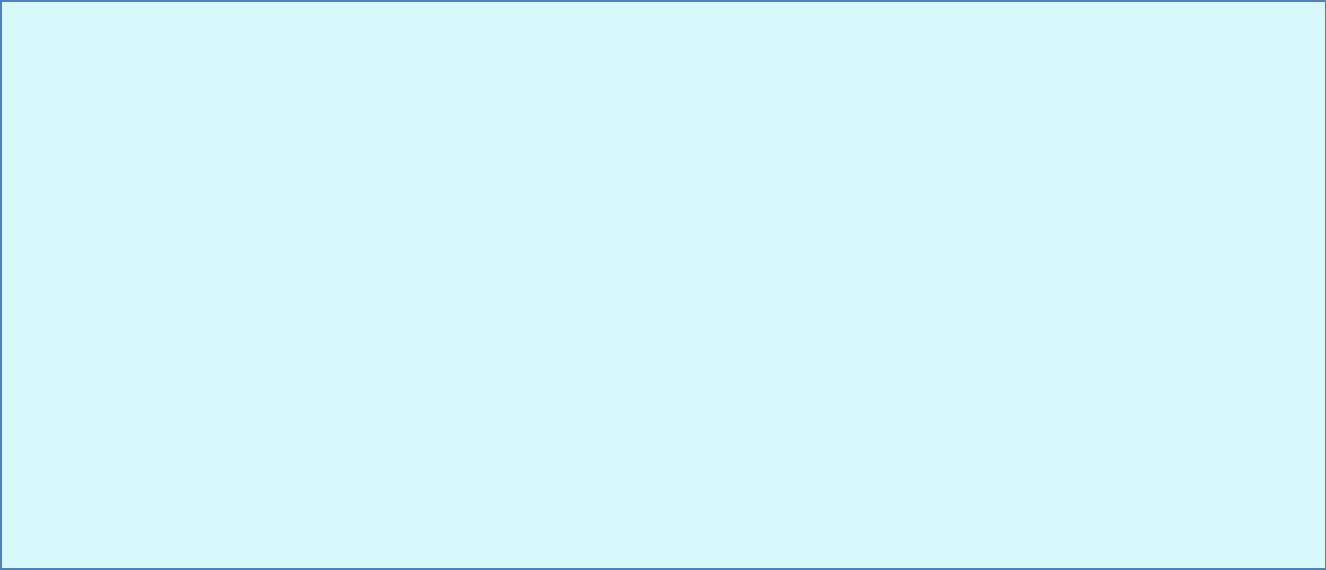
apc.rfc1867 = on

upload\_max\_filesize = 100M

post\_max\_size = 100M

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



apc.max\_file\_size = 200M

upload\_max\_filesize = 1000M

post\_max\_size = 1000M

max\_execution\_time = 600 ; 每个 PHP 页面运行的最大时间值(秒)，默认 30 秒

max\_input\_time = 600 ;

memory\_limit = 128M ;

每个 PHP 页面接收数据所需的最大时间，默认 60

每个 PHP 页面所吃掉的最大内存，默认 8M

九、Opcode 缓存

首先 php 代码被解析为 Tokens，然后再编译为 Opcode 码，最后执行 Opcode 码，返回

结果；所以，对于相同的 php 文件，第一次运行时可以缓存其 Opcode 码，下次再执行这个

页面时，直接会去找到缓存下的 opcode 码，直接执行最后一步，而不再需要中间的步骤了。

比较知名的是 XCache、Turck MM Cache、PHP Accelerator 等。

HTTP 缓存原理介绍（偏后端）

通过 Internet 获取资源既缓慢，成本又高。为此，Http 协议里包含了控制缓存的部分，

以使 Http 客户端可以缓存和重用以前获取的资源，从而优化性能，提升体验。虽然 Http 中

关于缓存控制的部分，随着协议演进，有一些变化。作为后端程序员，在开发 Web 服务时，

只需要关注请求头 If-None-Match、响应头 ETag、响应头 Cache-Control 就足够了。因为这三

个 Http 头就可以满足你的需求，并且，当今绝大多数的浏览器，都支持这三个 Http 头。所

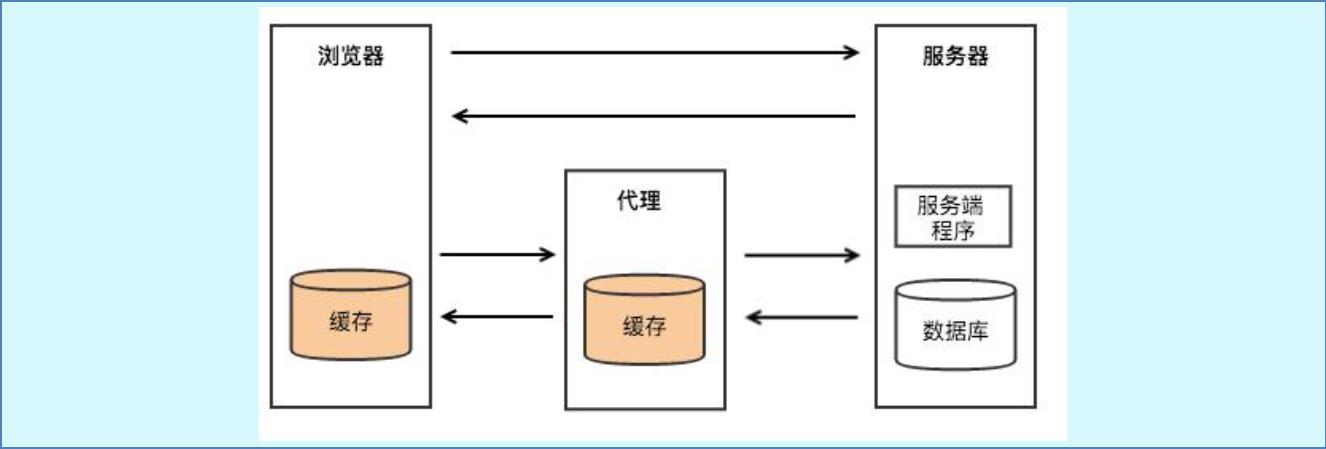
要做的就是，确保每个服务器响应都提供正确的 HTTP 头指令，以指导浏览器何时可以缓

存响应以及可以缓存多久

一、缓存在哪儿？

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



上图中有三个角色，浏览器、Web 代理和服务器，如图所示 Http 缓存存在于浏览器和

Web 代理中。当然在服务器内部，也存在着各种缓存，所谓的 Http 缓存控制，就是一种约

定，通过设置不同的响应头 Cache-Control 来控制浏览器和 Web 代理对缓存的使用策略，通

过设置请求头 If-None-Match 和响应头 ETag，来对缓存的有效性进行验证。

二、响应头 ETag

ETag 全称 Entity Tag，用来标识一个资源。在具体的实现中，ETag 可以是资源的 hash

值，也可以是一个内部维护的版本号。但不管怎样，ETag 应该能反映出资源内容的变化，

这是 Http 缓存可以正常工作的基础。

如上例中所展示的，服务器在返回响应时，通常会在 Http 头中包含一些关于响应的元

数据信息，其中，ETag 就是其中一个，本例中返回了值为 x1323ddx 的 ETag。当资源/file 的

内容发生变化时，服务器应当返回不同的 ETag。

三、请求头 If-None-Match

对于同一个资源，比如上一例中的/file，在进行了一次请求之后，浏览器就已经有了/file

的一个版本的内容，和这个版本的 ETag，当下次用户再需要这个资源，浏览器再次向服务

器请求的时候，可以利用请求头 If-None-Match 来告诉服务器自己已经有个 ETag 为 x1323ddx

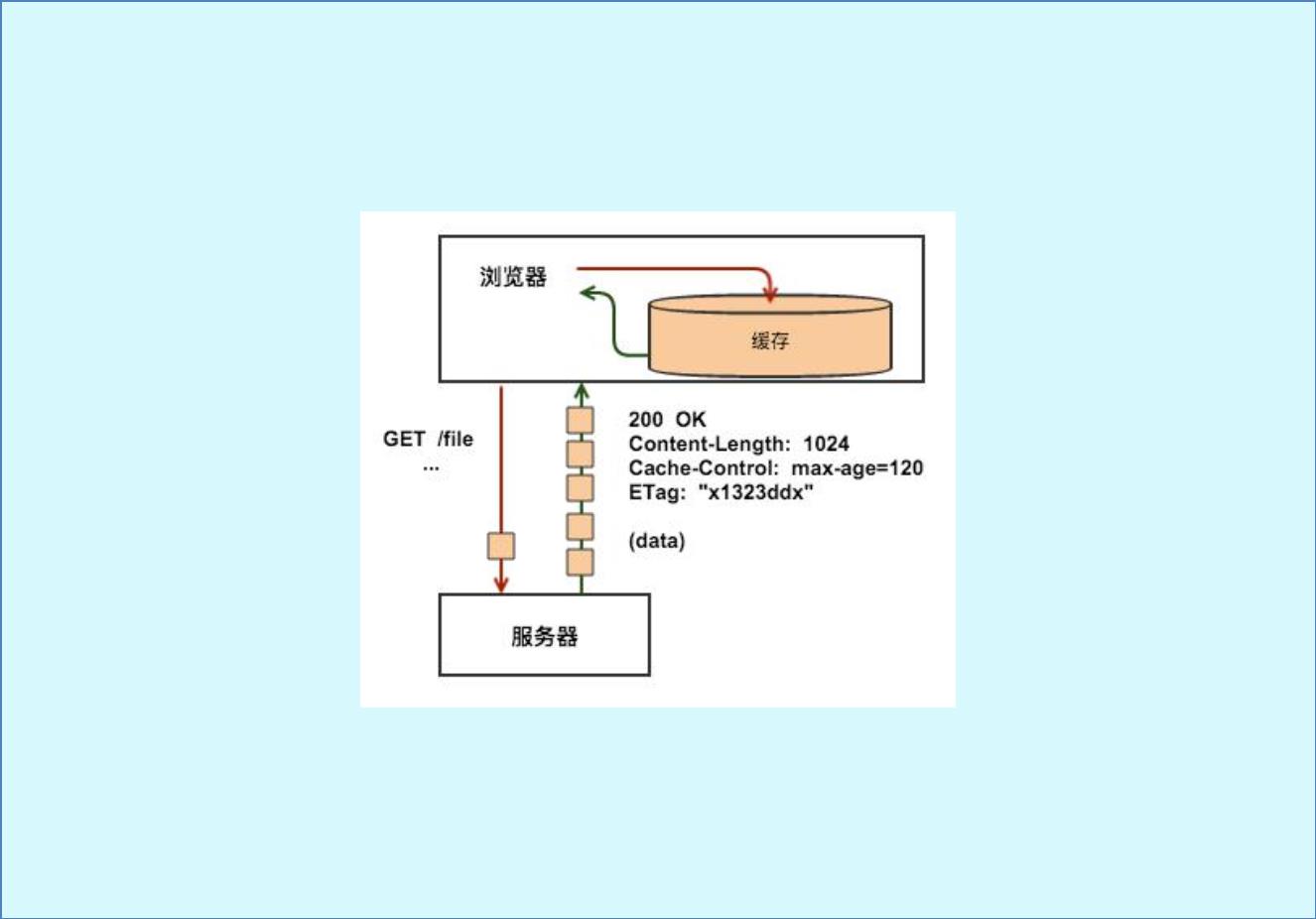
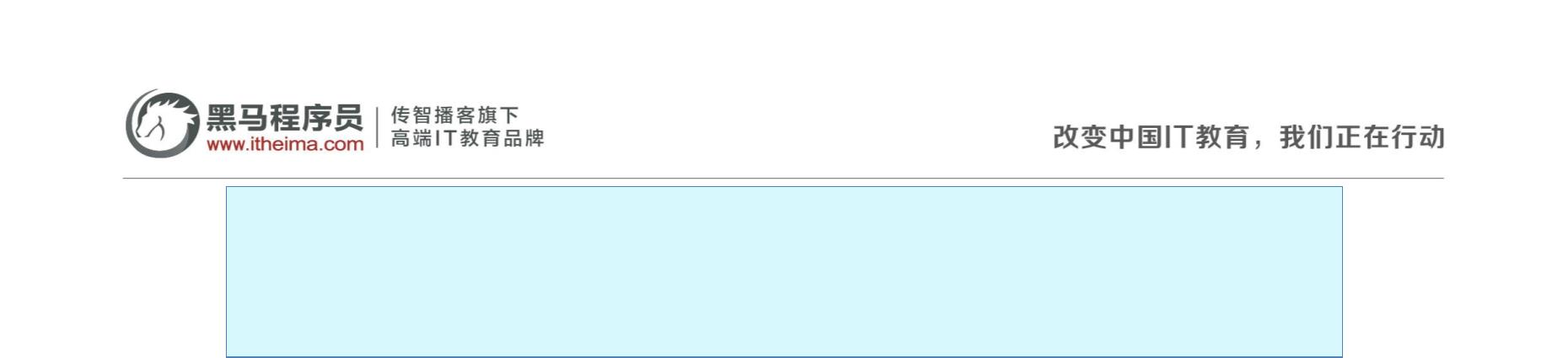
的/file，这样，如果服务器上的/file 没有变化，也就是说服务器上的/file 的 ETag 也是 x1323ddx

的话，服务器就不会再返回/file 的内容，而是返回一个 304 的响应，告诉浏览器该资源没有

变化，缓存有效。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



如上例中所示，在使用了 If-None-Match 之后，服务器只需要很小的响应就可以达到相

同的结果，从而优化了性能。

四、响应头 Cache-Control

每个资源都可以通过 Http 头 Cache-Control 来定义自己的缓存策略，Cache-Control 控制谁在

什么条件下可以缓存响应以及可以缓存多久。最快的请求是不必与服务器进行通信的请求：

通过响应的本地副本，我们可以避免所有的网络延迟以及数据传输的数据成本。为此，HTTP

规范允许服务器返回一系列不同的 Cache-Control 指令，控制浏览器或者其他中继缓存如何

缓存某个响应以及缓存多长时间。

Cache-Control 头在 HTTP/1.1 规范中定义，取代了之前用来定义响应缓存策略的头（例如

Expires）。当前的所有浏览器都支持 Cache-Control，因此，使用它就够了。

以下介绍可以在 Cache-Control 中设置的常用指令。

max-age

该指令指定从当前请求开始，允许获取的响应被重用的最长时间（单位为秒。例如：

Cache-Control:max-age=60 表示响应可以再缓存和重用 60 秒。需要注意的是，在 max-age

指定的时间之内，浏览器不会向服务器发送任何请求，包括验证缓存是否有效的请求，也就

是说，如果在这段时间之内，服务器上的资源发生了变化，那么浏览器将不能得到通知，而

使用老版本的资源。所以在设置缓存时间的长度时，需要慎重。

public 和 private

如果设置了 public，表示该响应可以再浏览器或者任何中继的 Web 代理中缓存，public 是默

认值，即 Cache-Control:max-age=60 等同于 Cache-Control:public, max-age=60。

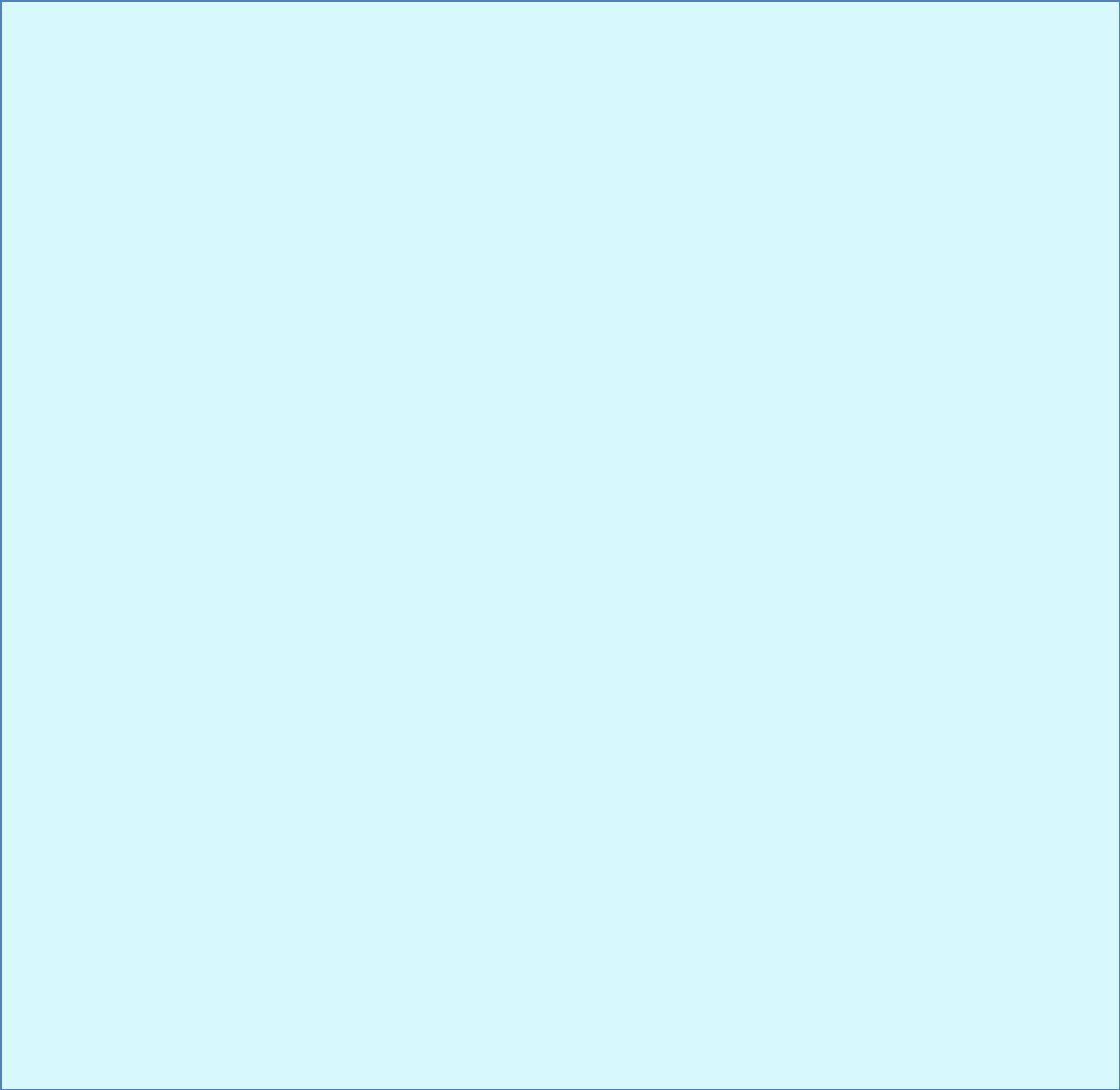
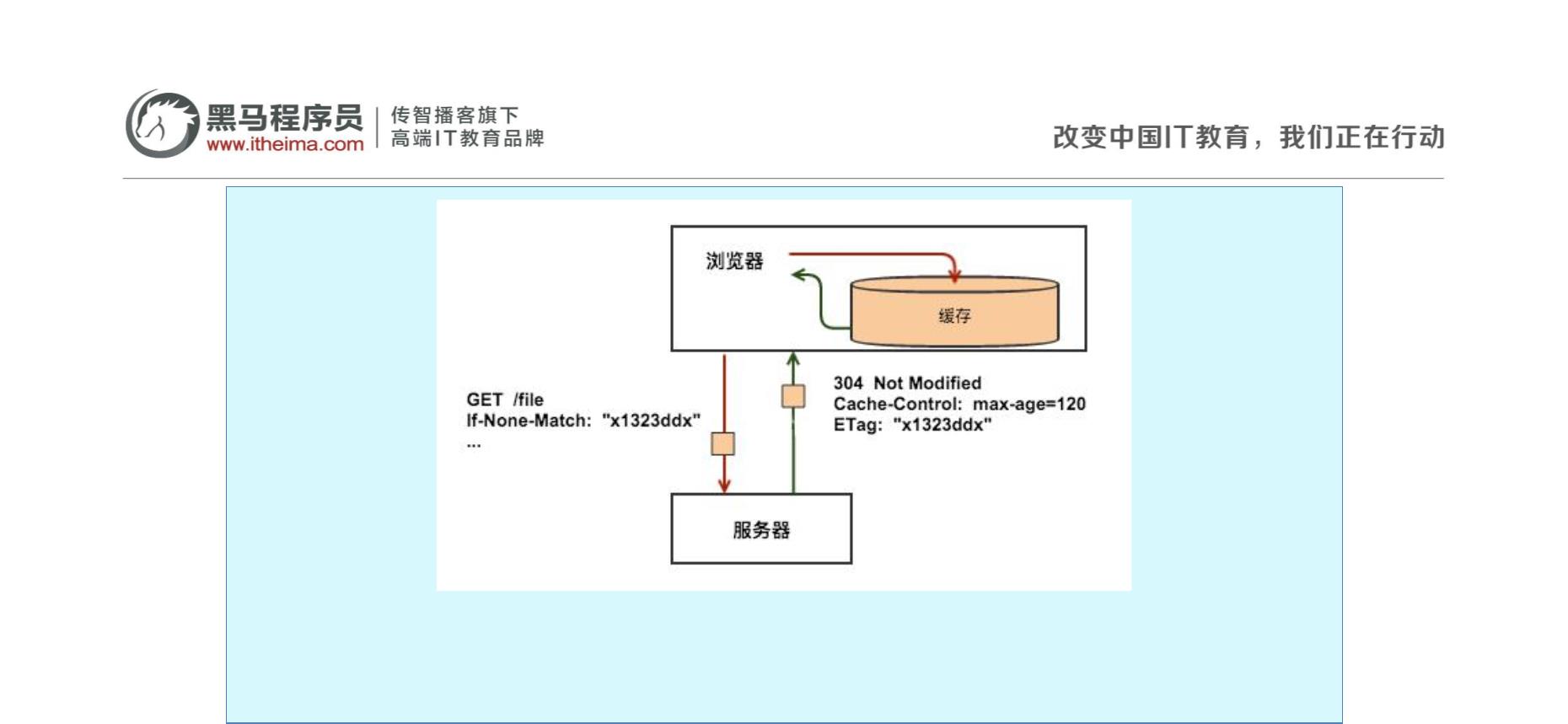
在服务器设置了 private 比如 Cache-Control:private, max-age=60 的情况下，表示只有用户的

浏览器可以缓存 private 响应，不允许任何中继 Web 代理对其进行缓存 - 例如，用户浏览

器可以缓存包含用户私人信息的 HTML 网页，但是 CDN 不能缓存。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



no-cache

如果服务器在响应中设置了 no-cache 即 Cache-Control:no-cache，那么浏览器在使用缓存的

资源之前，必须先与服务器确认返回的响应是否被更改，如果资源未被更改，可以避免下载。

这个验证之前的响应是否被修改，就是通过上面介绍的请求头 If-None-match 和响应头 ETag

来实现的。

需要注意的是，no-cache 这个名字有一点误导。设置了 no-cache 之后，并不是说浏览器就

不再缓存数据，只是浏览器在使用缓存数据时，需要先确认一下数据是否还跟服务器保持一

致。如果设置了 no-cache，而 ETag 的实现没有反应出资源的变化，那就会导致浏览器的缓

存数据一直得不到更新的情况。

no-store

如果服务器在响应中设置了 no-store 即 Cache-Control:no-store，那么浏览器和任何中继的

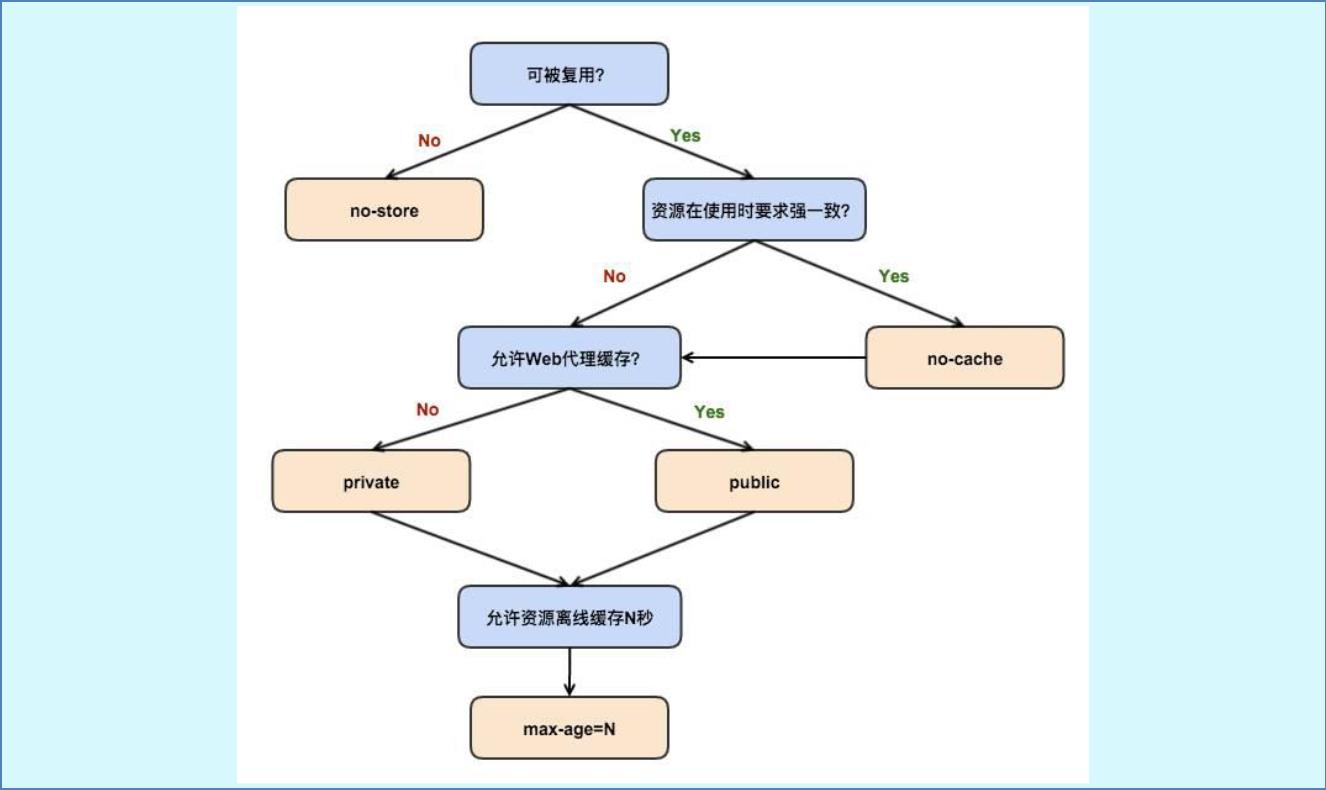
Web 代理，都不会存储这次相应的数据。当下次请求该资源时，浏览器只能重新请求服务

器，重新从服务器读取资源。

五、决定一个资源的 Cache-Control 策略的流程图：

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



Redis、Memcache 和 MongoDB 的对比（最全没有之一）

Memcached——

Memcached 的优点：

1、Memcached 可以利用多核优势，单实例吞吐量极高，可以达到几十万 QPS（取决于

key、value 的字节大小以及服务器硬件性能，日常环境中 QPS 高峰大约在 4-6w 左右）。适

用于最大程度扛量。

2、支持直接配置为 session handle。

Memcached 的局限性：

1、只支持简单的 key/value 数据结构，不像 Redis 可以支持丰富的数据类型。

2、无法进行持久化，数据不能备份，只能用于缓存使用，且重启后数据全部丢失。

3、无法进行数据同步，不能将 MC 中的数据迁移到其他 MC 实例中。

4、Memcached 内存分配采用 Slab Allocation 机制管理内存，value 大小分布差异较大时

会造成内存利用率降低，并引发低利用率时依然出现踢出等问题。需要用户注重 value 设计。

Redis——

Redis 的优点：

1、支持多种数据结构，如 string（字符串）、 list(双向链表)、dict(hash 表)、set(集合）、

zset(排序 set)、hyperloglog（基数估算）

2、支持持久化操作，可以进行 aof 及 rdb 数据持久化到磁盘，从而进行数据备份或数

据恢复等操作，较好的防止数据丢失的手段。

3、支持通过 Replication 进行数据复制，通过 master-slave 机制，可以实时进行数据的

同步复制，支持多级复制和增量复制，master-slave 机制是 Redis 进行 HA 的重要手段。

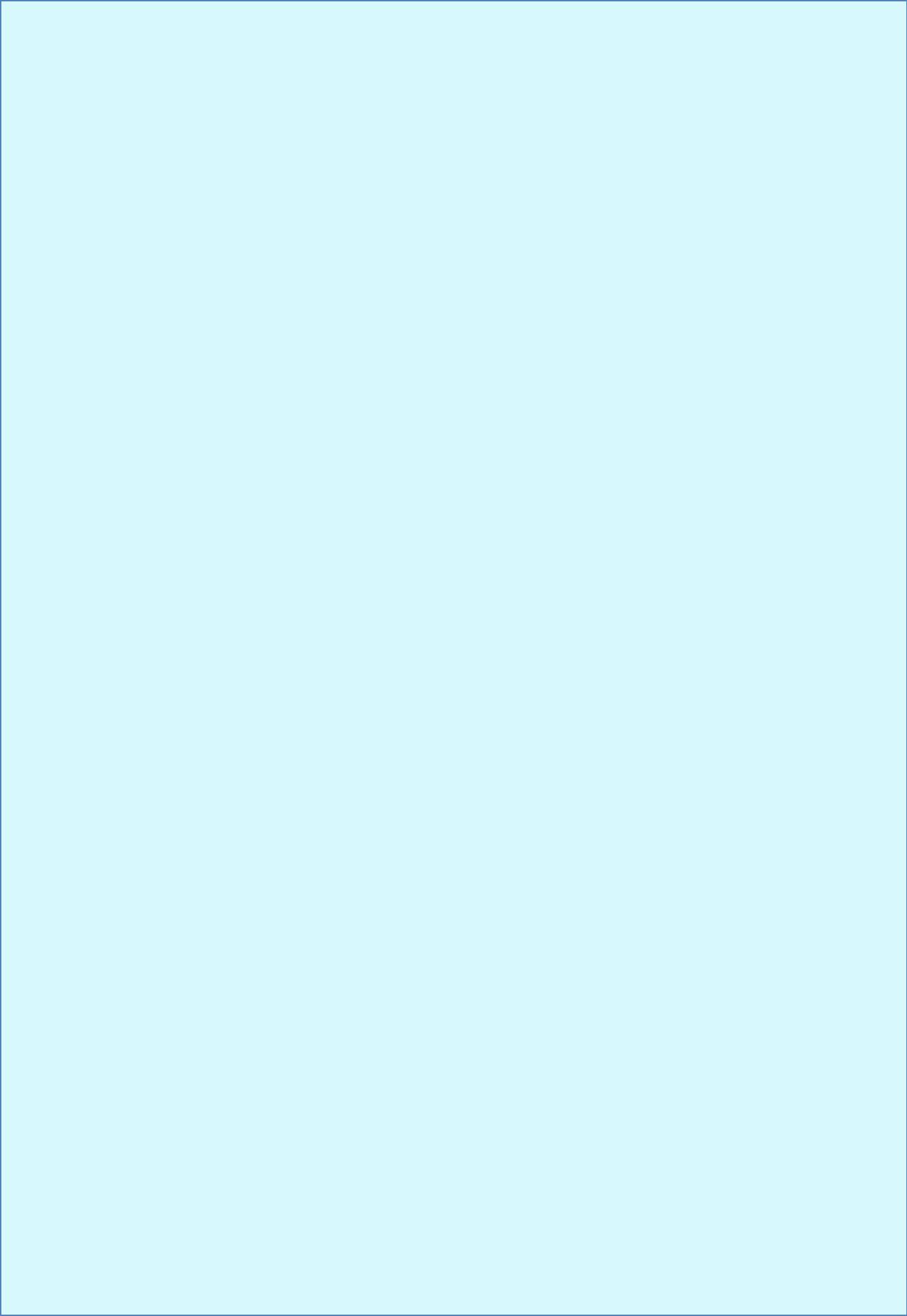
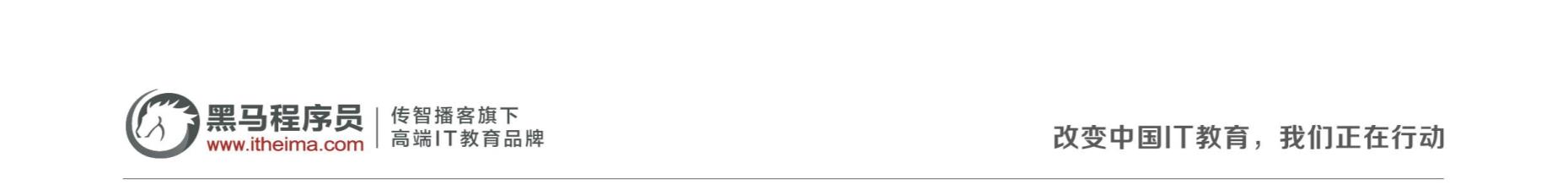
4、单线程请求，所有命令串行执行，并发情况下不需要考虑数据一致性问题。

5、支持 pub/sub 消息订阅机制，可以用来进行消息订阅与通知。

6、支持简单的事务需求，但业界使用场景很少，并不成熟。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



Redis 的局限性：

1、Redis 只能使用单线程，性能受限于 CPU 性能，故单实例 CPU 最高才可能达到 5-6wQPS

每秒（取决于数据结构，数据大小以及服务器硬件性能，日常环境中 QPS 高峰大约在 1-2w

左右）。

2、支持简单的事务需求，但业界使用场景很少，并不成熟，既是优点也是缺点。

3、Redis 在 string 类型上会消耗较多内存，可以使用 dict（hash 表）压缩存储以降低内

存耗用。

Memcached 和 Redis 都是 Key-Value 类型，不适合在不同数据集之间建立关系，也不适

合进行查询搜索。比如 redis 的 keys pattern 这种匹配操作，对 redis 的性能是灾难。

mongoDB——

mongoDB 是一种文档性的数据库。文档的数据库，即可以存放 xml、json、bson 类型

系列的数据。

这些数据具备自述性（self-describing），呈现分层的树状数据结构。redis 可以用 hash

存放简单关系型数据。

mongoDB 存放 json 格式数据。

适合场景：事件记录、内容管理或者博客平台，比如评论系统。

1.mongodb 持久化原理

mongodb 与 mysql 不同，mysql 的每一次更新操作都会直接写入硬盘，但是 mongo 不会，

做为内存型数据库，数据操作会先写入内存，然后再会持久化到硬盘中去，那么 mongo 是

如何持久化的呢

mongodb 在启动时，专门初始化一个线程不断循环（除非应用 crash 掉），用于在一定

时间周期内来从 defer 队列中获取要持久化的数据并写入到磁盘的 journal(日志)和

mongofile(数据)处，当然因为它不是在用户添加记录时就写到磁盘上，所以按 mongodb 开

发者说，它不会造成性能上的损耗，因为看过代码发现，当进行 CUD 操作时，记录(Record

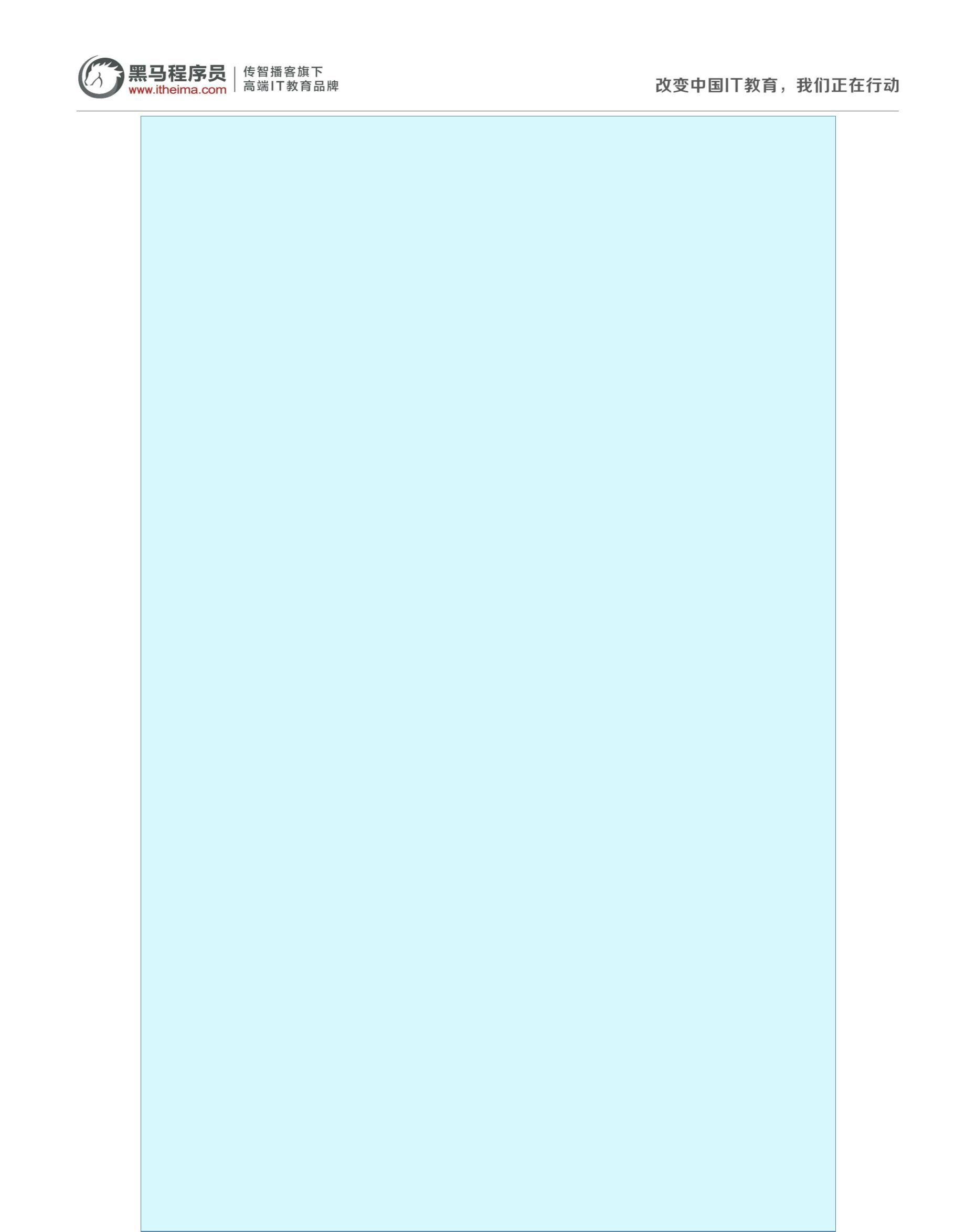
类型)都被放入到 defer 队列中以供延时批量（groupcommit）提交写入，但相信其中时间周

期参数是个要认真考量的参数，系统为 90 毫秒，如果该值更低的话，可能会造成频繁磁盘

操作，过高又会造成系统宕机时数据丢失过。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



2.什么是 NoSQL 数据库？NoSQL 和 RDBMS 有什么区别？在哪些情况下使用和不使用

NoSQL 数据库？

NoSQL 是非关系型数据库，NoSQL = Not Only SQL。

关系型数据库采用的结构化的数据，NoSQL 采用的是键值对的方式存储数据。

在处理非结构化/半结构化的大数据时；在水平方向上进行扩展时；随时应对动态增加

的数据项时可以优先考虑使用 NoSQL 数据库。

在考虑数据库的成熟度；支持；分析和商业智能；管理及专业性等问题时，应优先考虑

关系型数据库。

3.MySQL 和 MongoDB 之间最基本的区别是什么？

关系型数据库与非关系型数据库的区别，即数据存储结构的不同。

4.MongoDB 的特点是什么？

（1）面向文档（2）高性能（3）高可用（4）易扩展（5）丰富的查询语言

5.MongoDB 支持存储过程吗？如果支持的话，怎么用？

MongoDB 支持存储过程，它是 javascript 写的，保存在 db.system.js 表中。

6.如何理解 MongoDB 中的 GridFS 机制，MongoDB 为何使用 GridFS 来存储文件？

GridFS 是一种将大型文件存储在 MongoDB 中的文件规范。使用 GridFS 可以将大文件分

隔成多个小文档存放，这样我们能够有效的保存大文档，而且解决了 BSON 对象有限制的问

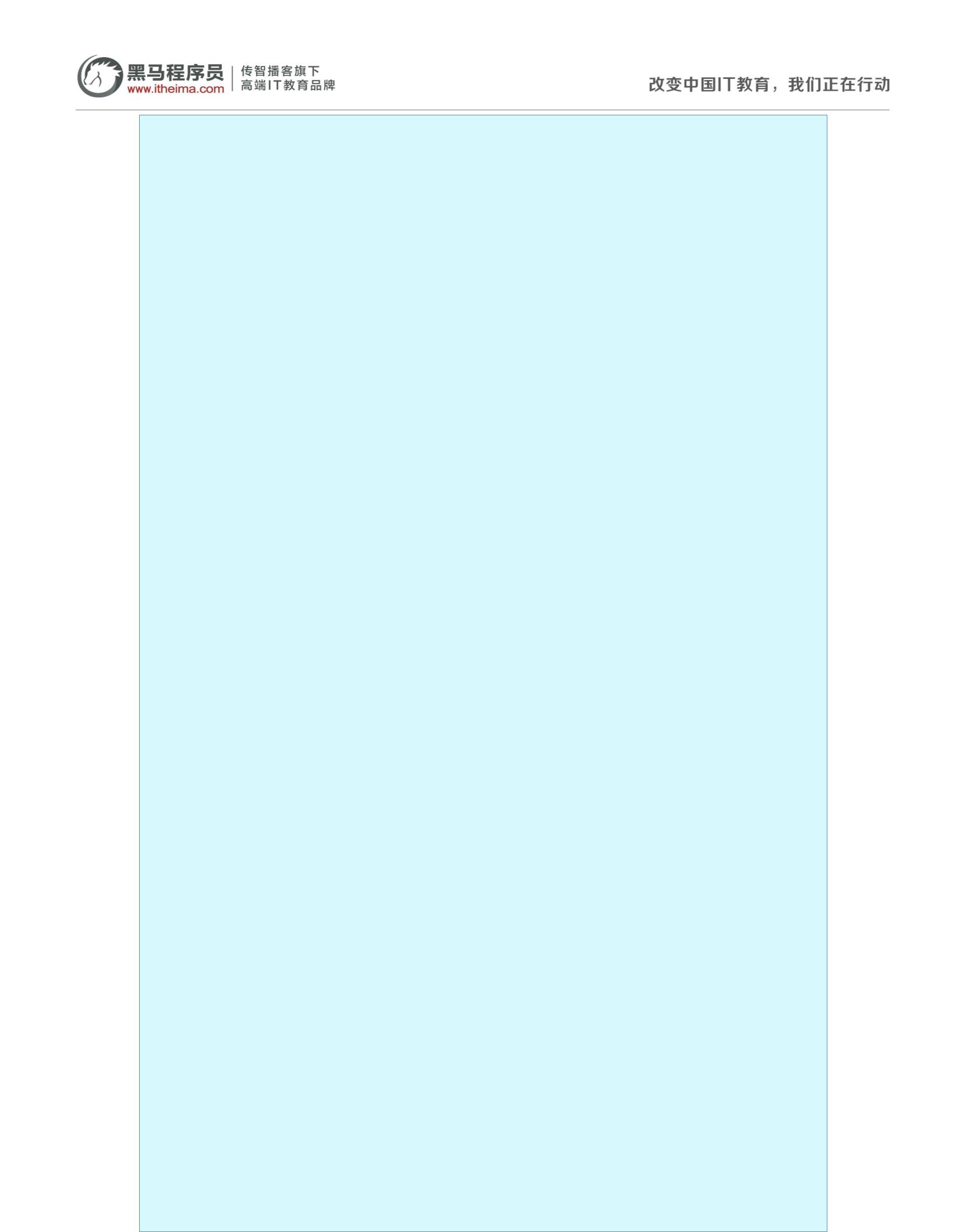
题。

7.为什么 MongoDB 的数据文件很大？

MongoDB 采用的预分配空间的方式来防止文件碎片。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



8.当更新一个正在被迁移的块（Chunk）上的文档时会发生什么？

更新操作会立即发生在旧的块（Chunk）上，然后更改才会在所有权转移前复制到新的

分片上。

9.MongoDB 在 A:{B,C}上建立索引，查询 A:{B,C}和 A:{C,B}都会使用索引吗？

不会，只会在 A:{B,C}上使用索引。

10.如果一个分片（Shard）停止或很慢的时候，发起一个查询会怎样？

如果一个分片停止了，除非查询设置了“Partial”选项，否则查询会返回一个错误。如

果一个分片响应很慢，MongoDB 会等待它的响应。

总结：

从以下几个维度，对 redis、memcache、mongoDB 做了对比

1、性能

都比较高，性能对我们来说应该都不是瓶颈

总体来讲，TPS（每秒事务处理量）方面 redis 和 memcache 差不多，要大于 mongodb

2、操作的便利性

memcache 数据结构单一

redis 丰富一些，数据操作方面，redis 更好一些，较少的网络 IO 次数

mongodb 支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰

富

3、内存空间的大小和数据量的大小

redis 在 2.0 版本后增加了自己的 VM 特性，突破物理内存的限制；可以对 key value 设

置过期时间（类似 memcache）

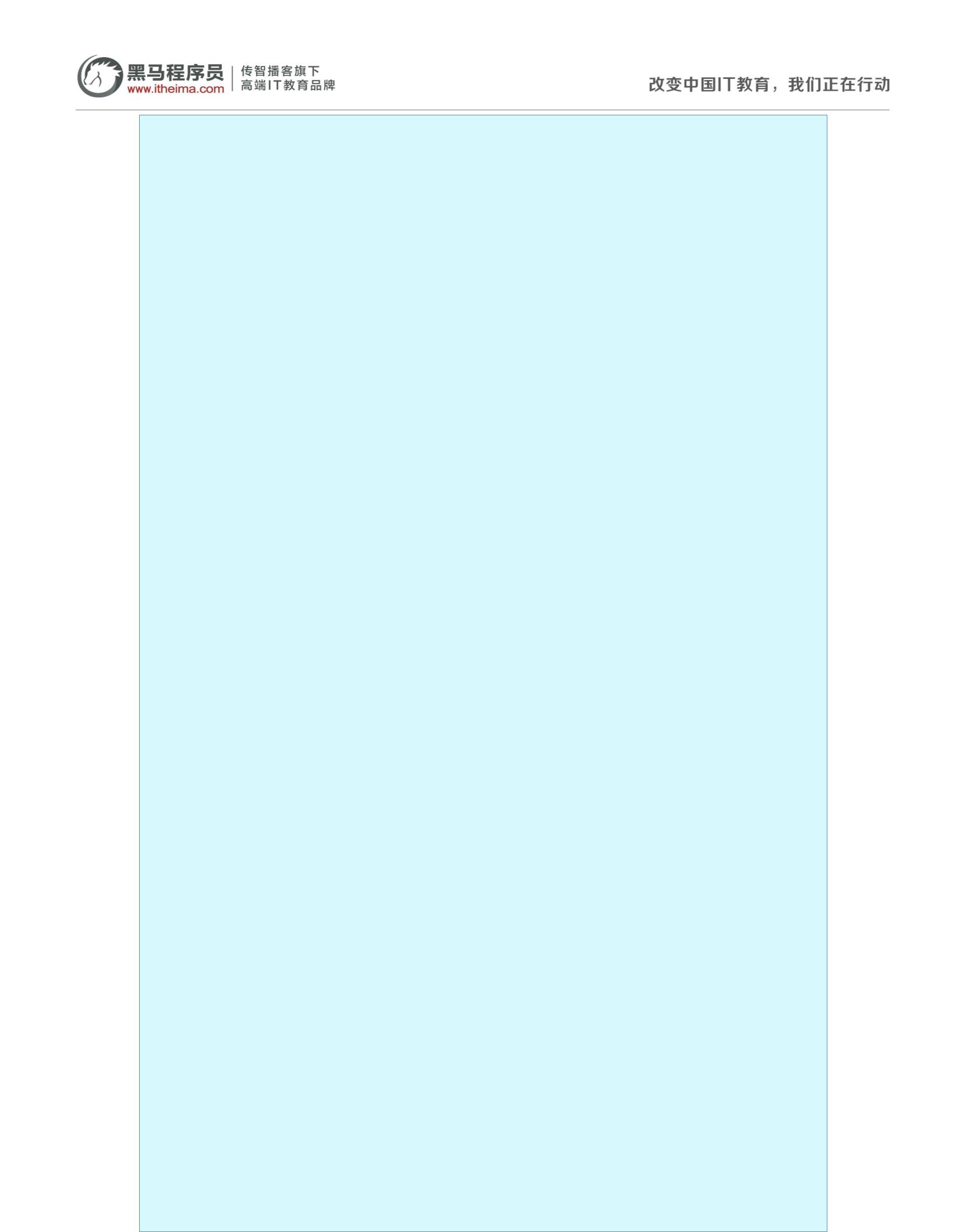
memcache 可以修改最大可用内存,采用 LRU 算法

mongoDB 适合大数据量的存储，依赖操作系统 VM 做内存管理，吃内存也比较厉害，

服务不要和别的服务在一起

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



4、可用性（单点问题）

对于单点问题，

redis，依赖客户端来实现分布式读写；主从复制时，每次从节点重新连接主节点都要依

赖整个快照,无增量复制，因性能和效率问题，

所以单点问题比较复杂；不支持自动 sharding,需要依赖程序设定一致 hash 机制。

一种替代方案是，不用 redis 本身的复制机制，采用自己做主动复制（多份存储），或

者改成增量复制的方式（需要自己实现），一致性问题和性能的权衡

Memcache 本身没有数据冗余机制，也没必要；对于故障预防，采用依赖成熟的 hash

或者环状的算法，解决单点故障引起的抖动问题。

mongoDB 支持 master-slave,replicaset（内部采用 paxos 选举算法，自动故障恢复）,auto

sharding 机制，对客户端屏蔽了故障转移和切分机制。

5、可靠性（持久化）

对于数据持久化和数据恢复，

redis 支持（快照、AOF）：依赖快照进行持久化，aof 增强了可靠性的同时，对性能有

所影响

memcache 不支持，通常用在做缓存,提升性能；

MongoDB 从 1.8 版本开始采用 binlog 方式支持持久化的可靠性

6、数据一致性（事务支持）

Memcache 在并发场景下，用 cas 保证一致性

redis 事务支持比较弱，只能保证事务中的每个操作连续执行

mongoDB 不支持事务

7、数据分析

mongoDB 内置了数据分析的功能(mapreduce),其他不支持

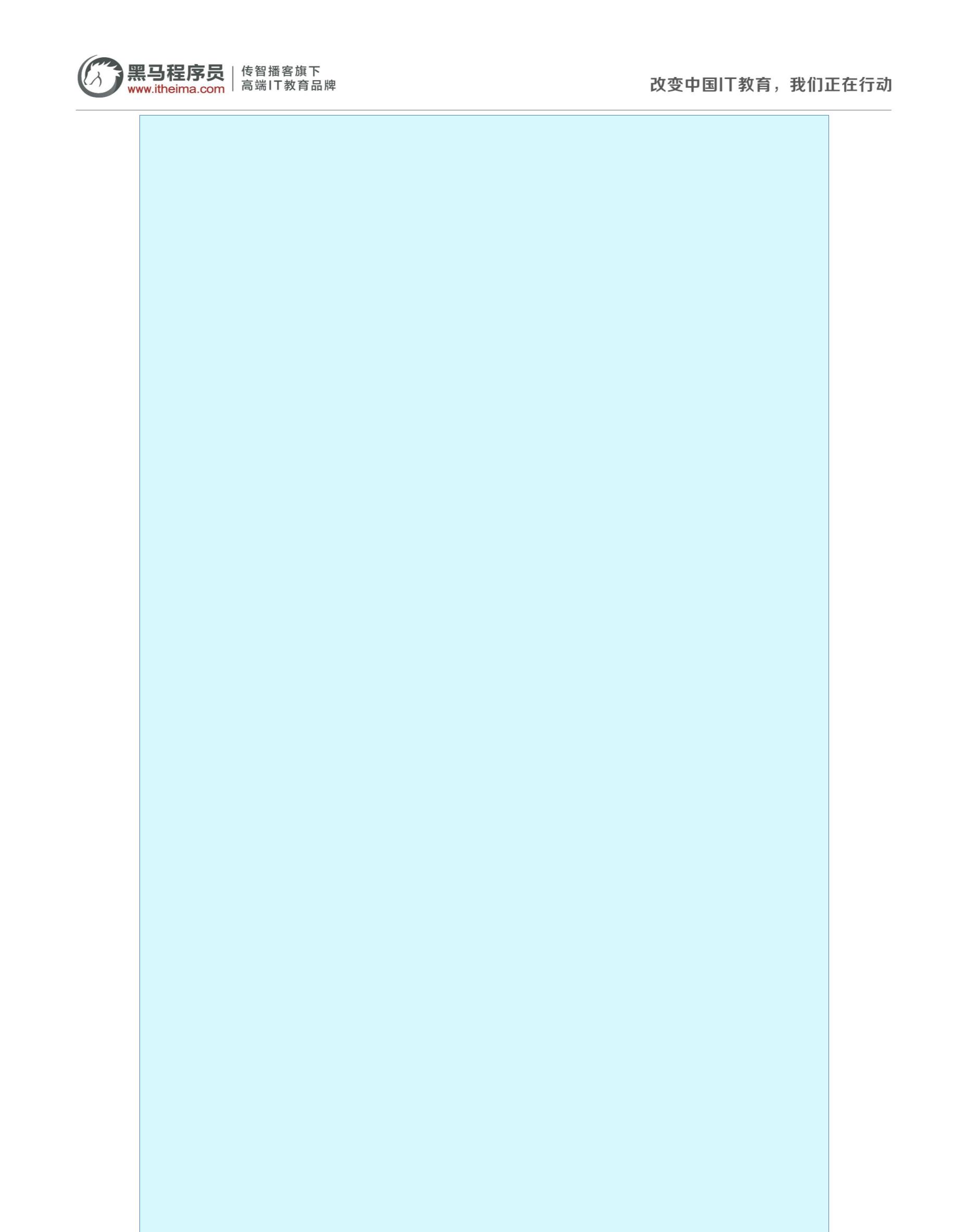
8、应用场景

redis：数据量较小的更性能操作和运算上

memcache：用于在动态系统中减少数据库负载，提升性能;做缓存，提高性能（适合读

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



多写少，对于数据量比较大，可以采用 sharding）

MongoDB:主要解决海量数据的访问效率问题

负载均衡原理

一、Http 重定向

当 http 代理（比如浏览器）向 web 服务器请求某个 URL 后，web 服务器可以通过 http

响应头信息中的 Location 标记来返回一个新的 URL。这意味着 HTTP 代理需要继续请求这个

新的 URL，完成自动跳转。

性能缺陷：

1、吞吐率限制

主站点服务器的吞吐率平均分配到了被转移的服务器。现假设使用 RR（Round Robin）

调度策略，子服务器的最大吞吐率为 1000reqs/s，那么主服务器的吞吐率要达到 3000reqs/s

才能完全发挥三台子服务器的作用，那么如果有 100 台子服务器，那么主服务器的吞吐率可

想而知得有大？相反，如果主服务的最大吞吐率为 6000reqs/s，那么平均分配到子服务器的

吞吐率为 2000reqs/s，而现子服务器的最大吞吐率为 1000reqs/s，因此就得增加子服务器的

数量，增加到 6 个才能满足。

2、重定向访问深度不同

有的重定向一个静态页面，有的重定向相比复杂的动态页面，那么实际服务器的负载差

异是不可预料的，而主站服务器却一无所知。因此整站使用重定向方法做负载均衡不太好。

需要权衡转移请求的开销和处理实际请求的开销，前者相对于后者越小，那么重定向的

意义就越大，例如下载。你可以去很多镜像下载网站试下，会发现基本下载都使用了 Location

做了重定向。

二、DNS 负载均衡

DNS 负责提供域名解析服务，当访问某个站点时，实际上首先需要通过该站点域名的

DNS 服务器来获取域名指向的 IP 地址，在这一过程中，DNS 服务器完成了域名到 IP 地址的

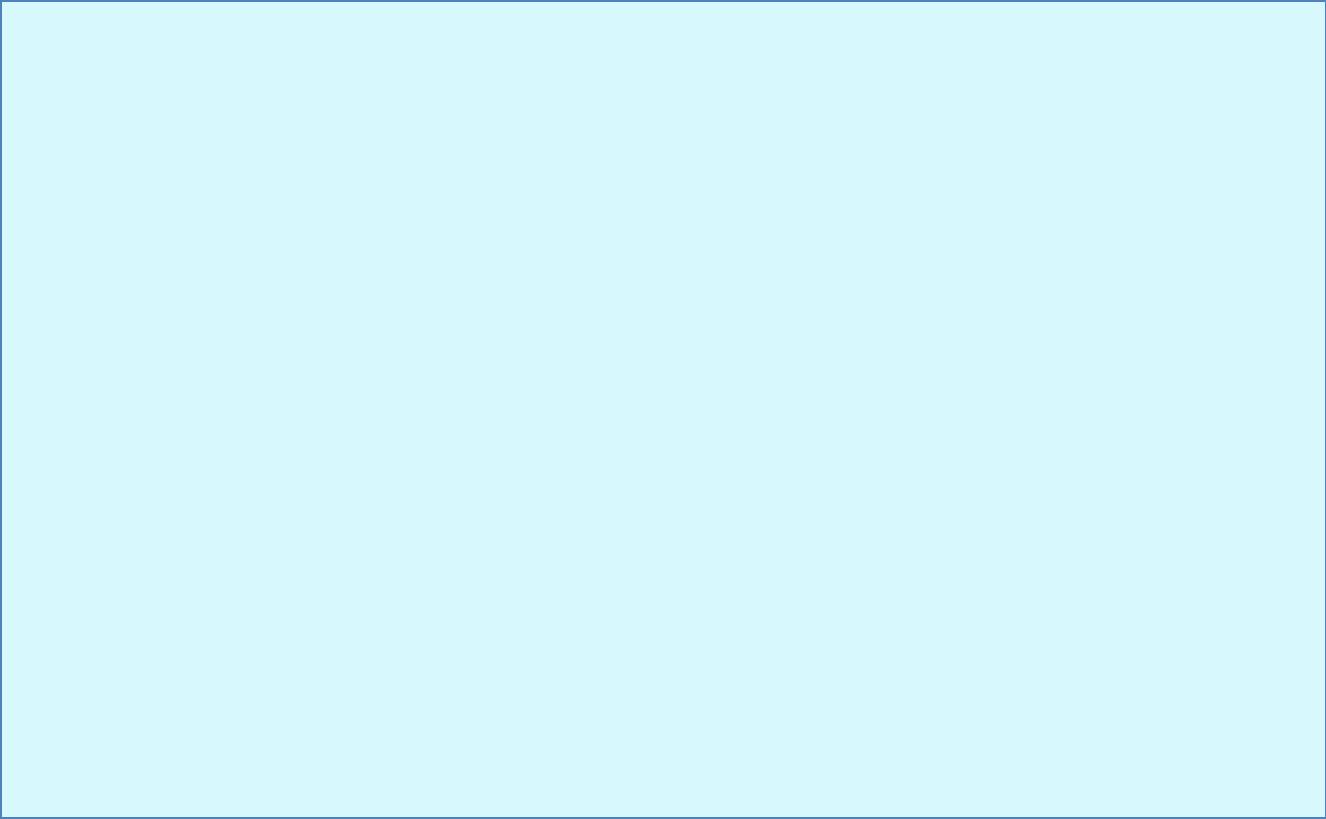
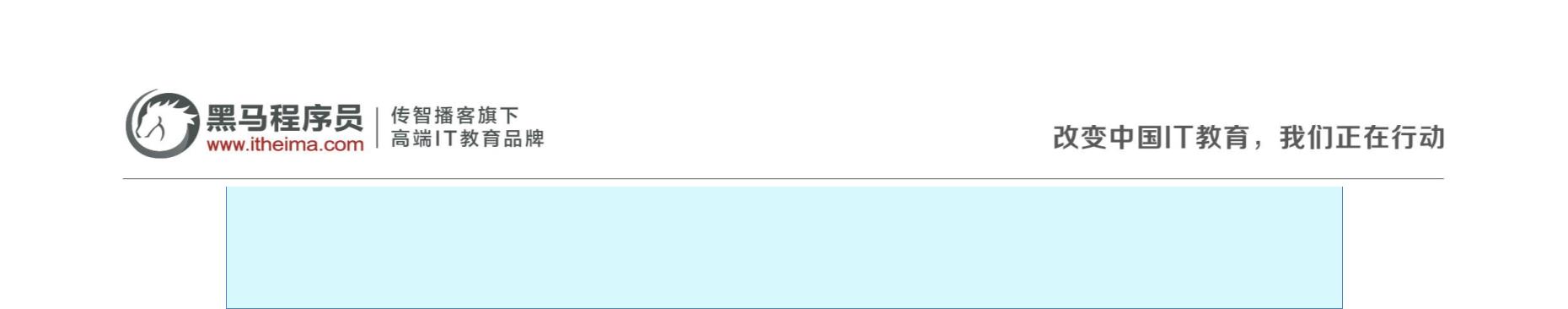
映射，同样，这样映射也可以是一对多的，这时候，DNS 服务器便充当了负载均衡调度器，

它就像 http 重定向转换策略一样，将用户的请求分散到多台服务器上，但是它的实现机制

完全不同。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



使用 dig 命令来看下”baidu”的 DNS 设置

可见 baidu 拥有三个 A 记录

相比 http 重定向，基于 DNS 的负载均衡完全节省了所谓的主站点，或者说 DNS 服务器

已经充当了主站点的职能。但不同的是，作为调度器，DNS 服务器本身的性能几乎不用担心。

因为 DNS 记录可以被用户浏览器或者互联网接入服务商的各级 DNS 服务器缓存，只有当缓

存过期后才会重新向域名的 DNS 服务器请求解析。也说是 DNS 不存在 http 的吞吐率限制，

理论上可以无限增加实际服务器的数量。

特性:

1、可以根据用户 IP 来进行智能解析。DNS 服务器可以在所有可用的 A 记录中寻找离用

记最近的一台服务器。

2、动态 DNS：在每次 IP 地址变更时，及时更新 DNS 服务器。当然，因为缓存，一定的

延迟不可避免。

不足：

1、没有用户能直接看到 DNS 解析到了哪一台实际服务器，加服务器运维人员的调试带

来了不便。

2、策略的局限性。例如你无法将 HTTP 请求的上下文引入到调度策略中，而在前面介

绍的基于 HTTP 重定向的负载均衡系统中，调度器工作在 HTTP 层面，它可以充分理解 HTTP

请求后根据站点的应用逻辑来设计调度策略，比如根据请求不同的 URL 来进行合理的过滤

和转移。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



3、如果要根据实际服务器的实时负载差异来调整调度策略，这需要 DNS 服务器在每次

解析操作时分析各服务器的健康状态，对于 DNS 服务器来说，这种自定义开发存在较高的

门槛，更何况大多数站点只是使用第三方 DNS 服务。

4、DNS 记录缓存，各级节点的 DNS 服务器不同程序的缓存会让你晕头转向。

5、基于以上几点，DNS 服务器并不能很好地完成工作量均衡分配，最后，是否选择基

于 DNS 的负载均衡方式完全取决于你的需要。

三、反向代理负载均衡

几乎所有主流的 Web 服务器都热衷于支持基于反向代理的负载均衡。它的核心工作就

是转发 HTTP 请求。

相比前面的 HTTP 重定向和 DNS 解析，反向代理的调度器扮演的是用户和实际服务器中

间人的角色：

1、任何对于实际服务器的 HTTP 请求都必须经过调度器

2、调度器必须等待实际服务器的 HTTP 响应，并将它反馈给用户（前两种方式不需要

经过调度反馈，是实际服务器直接发送给用户）

特性：

1、调度策略丰富。例如可以为不同的实际服务器设置不同的权重，以达到能者多劳的

效果。

2、对反向代理服务器的并发处理能力要求高，因为它工作在 HTTP 层面。

3、反向代理服务器进行转发操作本身是需要一定开销的，比如创建线程、与后端服务

器建立 TCP 连接、接收后端服务器返回的处理结果、分析 HTTP 头部信息、用户空间和内核

空间的频繁切换等，虽然这部分时间并不长，但是当后端服务器处理请求的时间非常短时，

转发的开销就显得尤为突出。例如请求静态文件，更适合使用前面介绍的基于 DNS 的负载

均衡方式。

4、反向代理服务器可以监控后端服务器，比如系统负载、响应时间、是否可用、TCP

连接数、流量等，从而根据这些数据调整负载均衡的策略。

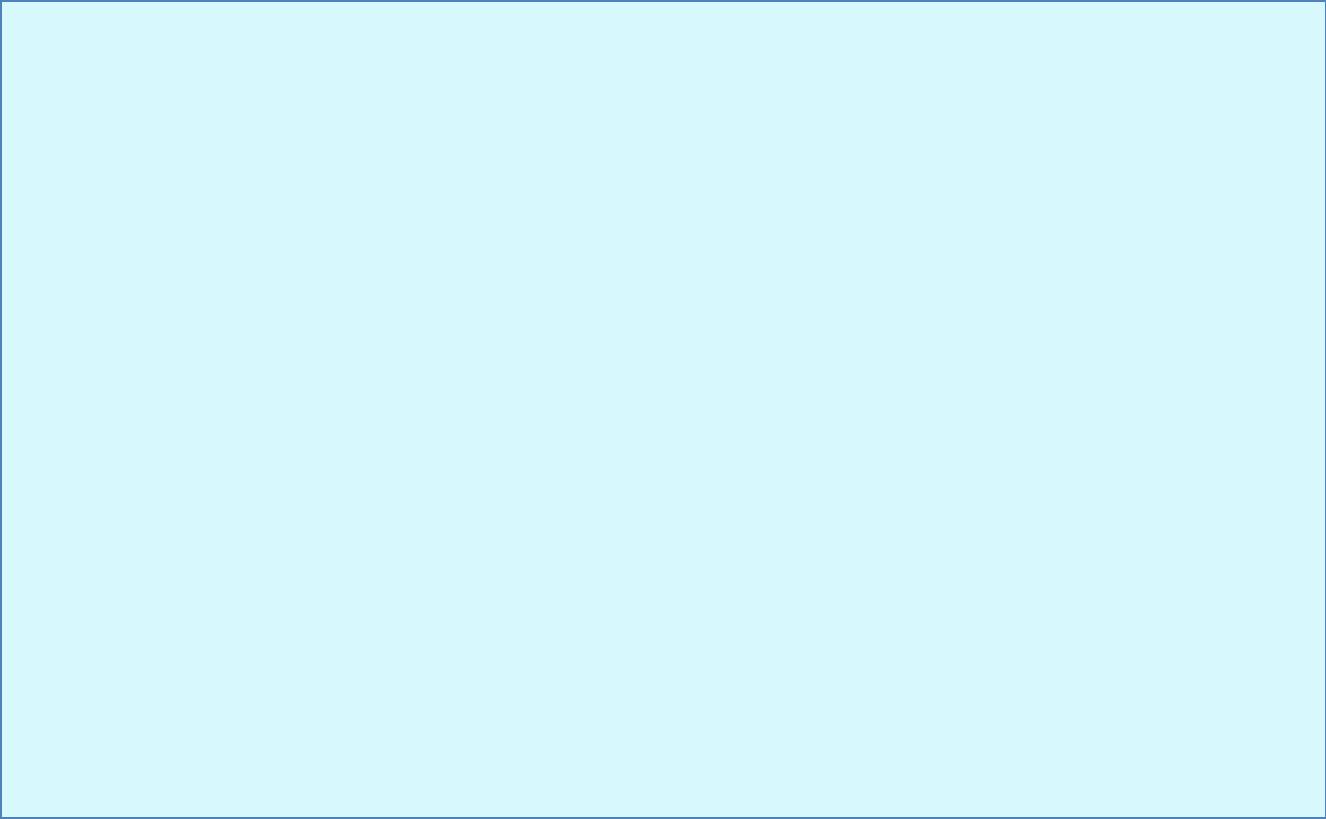
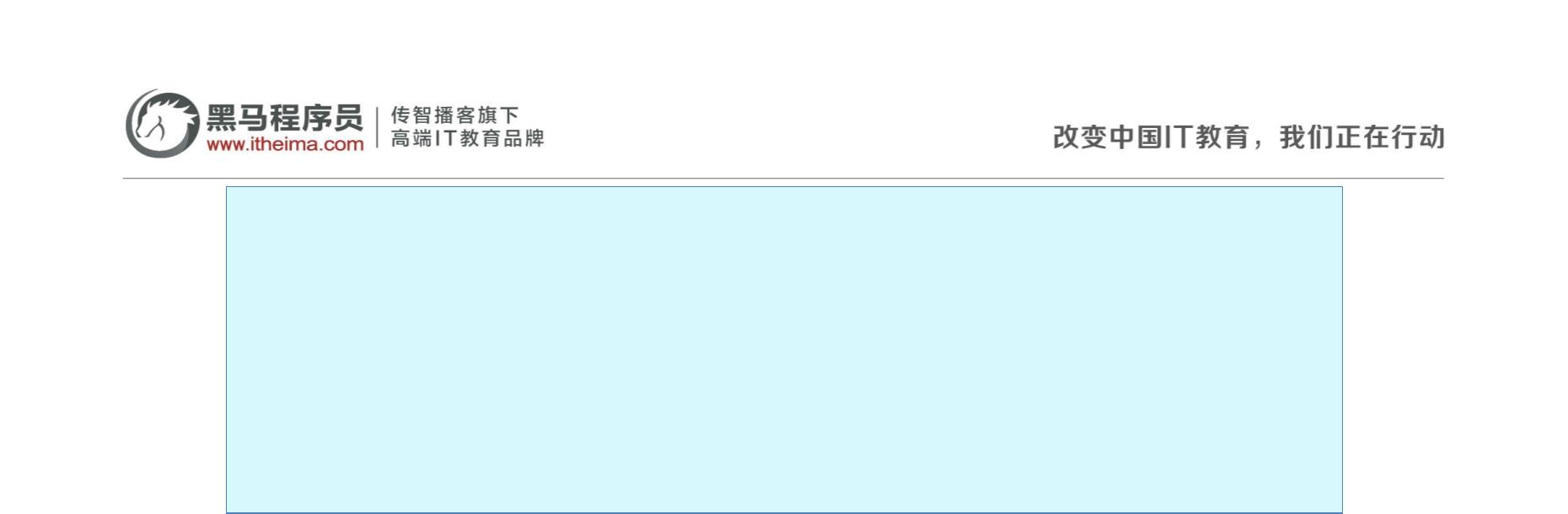
5、反射代理服务器可以让用户在一次会话周期内的所有请求始终转发到一台特定的后

端服务器上（粘滞会话），这样的好处一是保持 session 的本地访问，二是防止后端服务器

的动态内存缓存的资源浪费。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



四、IP 负载均衡（LVS-NAT）

因为反向代理服务器工作在 HTTP 层，其本身的开销就已经严重制约了可扩展性，从而

也限制了它的性能极限。那能否在 HTTP 层面以下实现负载均衡呢？

NAT 服务器:它工作在传输层，它可以修改发送来的 IP 数据包，将数据包的目标地址修

改为实际服务器地址。

从 Linux2.4 内核开始，其内置的 Neftilter 模块在内核中维护着一些数据包过滤表，这些

表包含了用于控制数据包过滤的规则。可喜的是，Linux 提供了 iptables 来对过滤表进行插

入、修改和删除等操作。更加令人振奋的是，Linux2.6.x 内核中内置了 IPVS 模块，它的工作

性质类型于 Netfilter 模块，不过它更专注于实现 IP 负载均衡。

五、直接路由（LVS-DR）

NAT 是工作在网络分层模型的传输层（第四层），而直接路由是工作在数据链路层（第

二层），貌似更屌些。它通过修改数据包的目标 MAC 地址（没有修改目标 IP），将数据包

转发到实际服务器上，不同的是，实际服务器的响应数据包将直接发送给客户羰，而不经过

调度器。

1、网络设置

这里假设一台负载均衡调度器，两台实际服务器，购买三个外网 ip，一台机一个，三台

机的默认网关需要相同，最后再设置同样的 ip 别名，这里假设别名为 10.10.120.193。这样

一来，将通过 10.10.120.193 这个 IP 别名来访问调度器，你可以将站点的域名指向这个 IP 别

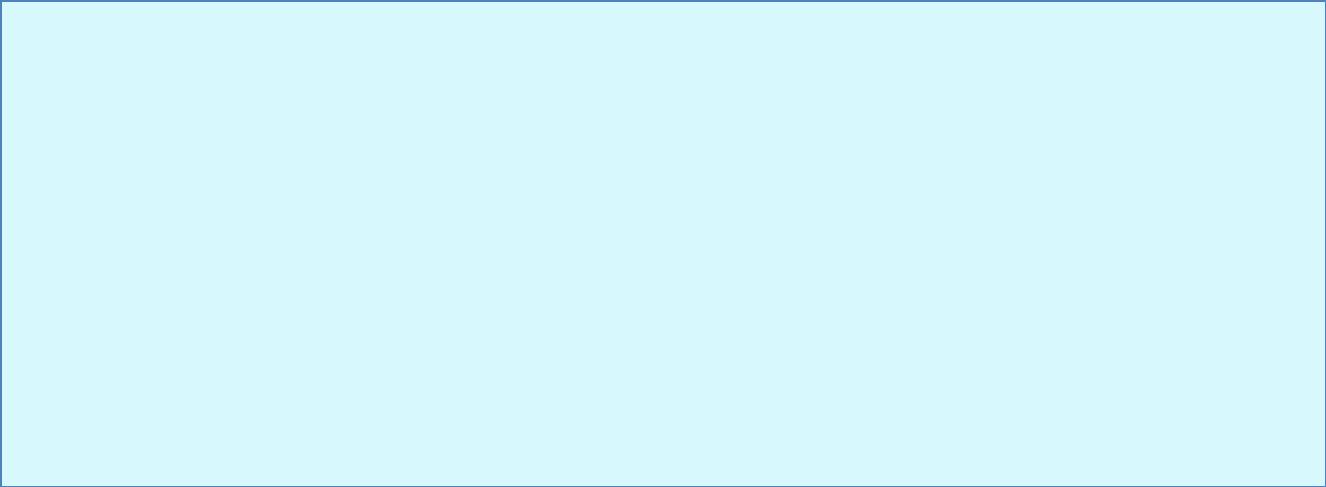
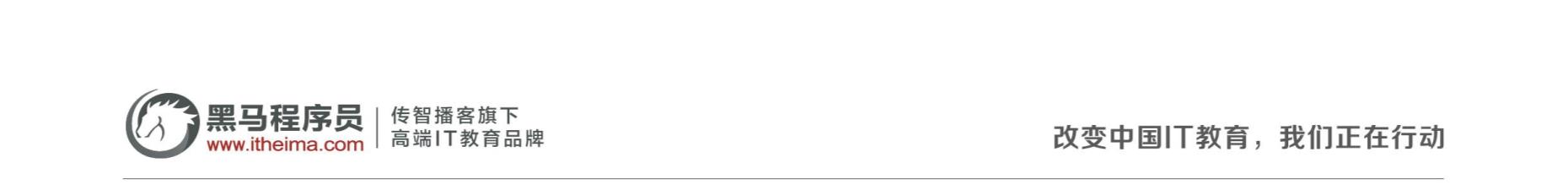
名。

2、将 ip 别名添加到回环接口 lo 上

这是为了让实际服务器不要去寻找其他拥有这个 IP 别名的服务器，在实际服务器中运行：

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



另外还要防止实际服务器响应来自网络中针对 IP 别名的 ARP 广播，为此还要执行：

echo "1" > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "2" > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo "1" > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo "1" > /proc/sys/net/ipv4/conf/all/arp\_announce

配置完了就可以使用 ipvsadm 配置 LVS-DR 集群了

ipvsadm -A -t 10.10.120.193:80 -s rr

ipvsadm -a -t 10.10.120.193:80 -r 10.10.120.210:8000 -g

ipvsadm -a -t 10.10.120.193:80 -r 10.10.120.211:8000 -g

-g 就意味着使用直接路由的方式转发数据包

LVS-DR 相较于 LVS-NAT 的最大优势在于 LVS-DR 不受调度器宽带的限制，例如假设三台

服务器在 WAN 交换机出口宽带都限制为 10Mbps，只要对于连接调度器和两台实际服务器

的 LAN 交换机没有限速，那么，使用 LVS-DR 理论上可以达到 20Mbps 的最大出口宽带，因

为它的实际服务器的响应数据包可以不经过调度器而直接发往用户端啊，所以它与调度器的

出口宽带没有关系，只能自身的有关系。而如果使用 LVS-NAT，集群只能最大使用 10Mbps

的宽带。所以，越是响应数据包远远超过请求数据包的服务，就越应该降低调度器转移请求

的开销，也就越能提高整体的扩展能力，最终也就越依赖于 WAN 出口宽带。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



总的来说，LVS-DR 适合搭建可扩展的负载均衡系统，不论是 Web 服务器还是文件服务

器，以及视频服务器，它都拥有出色的性能。前提是你必须为实际器购买一系列的合法 IP

地址。

六、IP 隧道（LVS-TUN）

基于 IP 隧道的请求转发机制：将调度器收到的 IP 数据包封装在一个新的 IP 数据包中，

转交给实际服务器，然后实际服务器的响应数据包可以直接到达用户端。目前 Linux 大多支

持，可以用 LVS 来实现，称为 LVS-TUN，与 LVS-DR 不同的是，实际服务器可以和调度器不

在同一个 WANt 网段，调度器通过 IP 隧道技术来转发请求到实际服务器，所以实际服务器

也必须拥有合法的 IP 地址。

总体来说，LVS-DR 和 LVS-TUN 都适合响应和请求不对称的 Web 服务器，如何从它们中

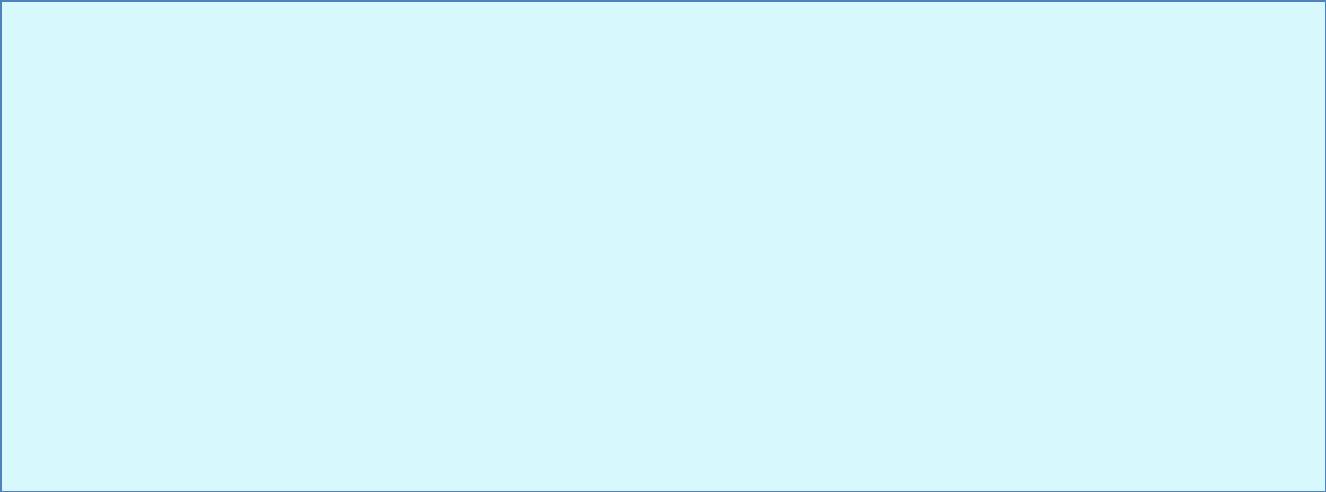
做出选择，取决于你的网络部署需要，因为 LVS-TUN 可以将实际服务器根据需要部署在不同

的地域，并且根据就近访问的原则来转移请求，所以有类似这种需求的，就应该选择

LVS-TUN。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



PHP 高并发秒杀解决方案/技术

在秒杀、抢火车票等地方，通常用遇到这样高并发的问题，下面提供了四种解决方案：

1、使用文件锁

$fp = fopen("order.lock", "r");

if(flock($fp,LOCK\_EX)){

//..处理订单的代码

flock($fp,LOCK\_UN);

}

fclose($fp);

2、使用消息队列（常用 Memcached 和 Redis）

比如：有 100 张票可供用户抢，那么就可以把这 100 张票放到缓存中，读写时不要加锁。

当并发量大的时候，可能有 500 人左右抢票成功，这样对于 500 后面的请求可以直接转到活

动结束的静态页面。进去的 500 个人中有 400 个人是不可能获得商品的。所以可以根据进入

队列的先后顺序只能前 100 个人购买成功。后面 400 个人就直接转到活动结束页面。当然进

去 500 个人只是举个例子，至于多少可以自己调整。而活动结束页面一定要用静态页面，不

要用数据库。这样就减轻了数据库的压力。

3、如果是分布式集群服务器，就需要一个或多个队列服务器

小米和淘宝的抢购还是有稍许不同的，小米重在抢的那瞬间，抢到了名额，就是你的，

你就可以下单结算。而淘宝则重在付款的时候的过滤，做了多层过滤，比如要卖 10 件商品，

他会让大于 10 的用户抢到，在付款的时候再进行并发过滤，一层层的减少一瞬间的并发量。

4、使用 Memcache 锁

product\_lock\_key 为票锁 key

当 product\_key 存在于 memcached 中时，所有用户都可以进入下单流程。

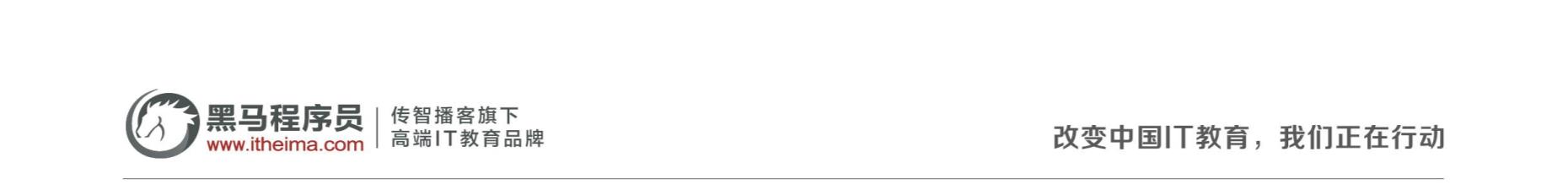
当进入支付流程时，首先往 memcached 存放 add(product\_lock\_key, “1″),如果返回成功，

进入支付流程。如果不成，则说明已经有人进入支付流程，则线程等待 N 秒，递归执行 add

操作。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



常见支付系统的对外接口

1.支付宝流程：

上图的流程中其实还隐藏了很多安全校验的细节，例如与支付宝接口之间的数据加密规

则和验签规则，异步回调接口的调用者 IP 白名单，支付宝订单信息反查及与 A 站点订单信

息比对校验（金额、用户、状态等）。另外，还有一些流程是可选的，例如，同步回调这一

步，如果将订单确认的流程加在里面，则有可能影响用户体验，所以在这里订单的确认流程

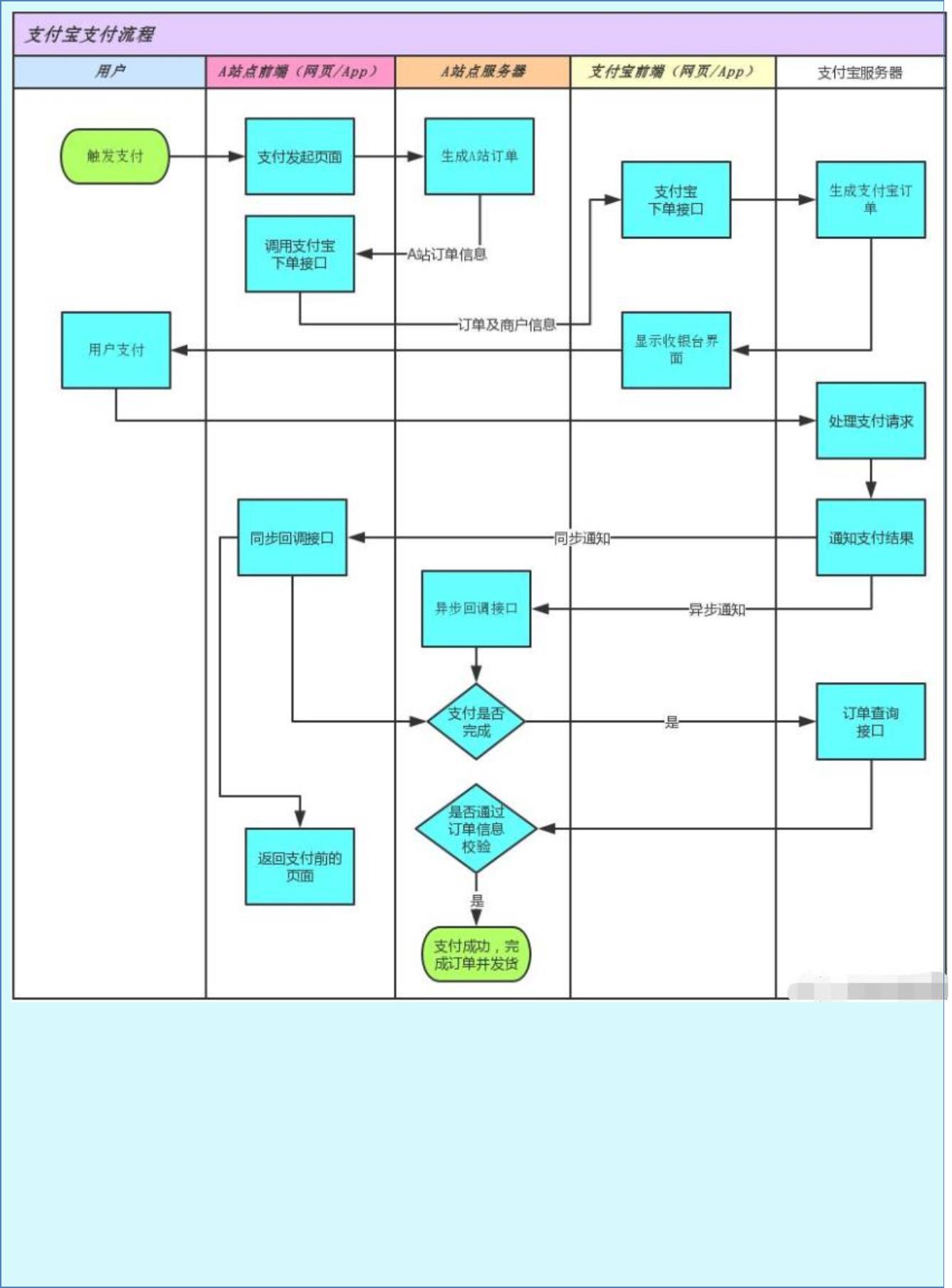
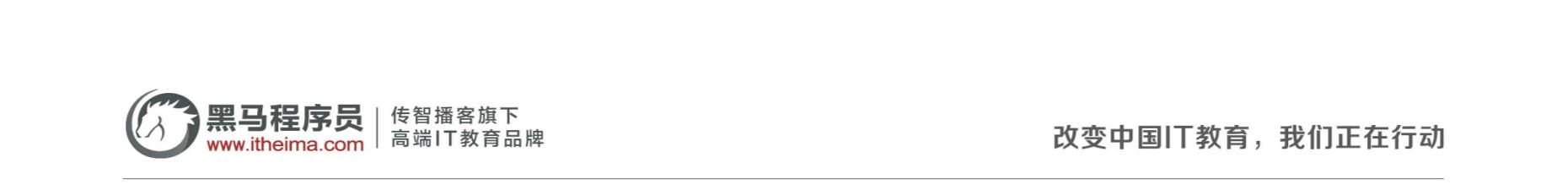
可以自己触发另开线程去跑，或者去除同步回调的确认订单功能，完全依赖异步回调来完成

订单。而异步回调里的订单反查也不是必须的，如果你认为白名单和验签规则足够可信的话，

不反查也可以

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



2.微信支付流程：

相比支付宝，其在初始的下单阶段有一些差异，主要表现在，下单动作是由 A 站点的

服务端调用的，而支付宝则是由前端发起调用的。相比起来，支付宝的下单动作由于是在前

端调用的，因此，A 站点需要将自己的订单信息返回到客户端，然后又客户端发起调用支付

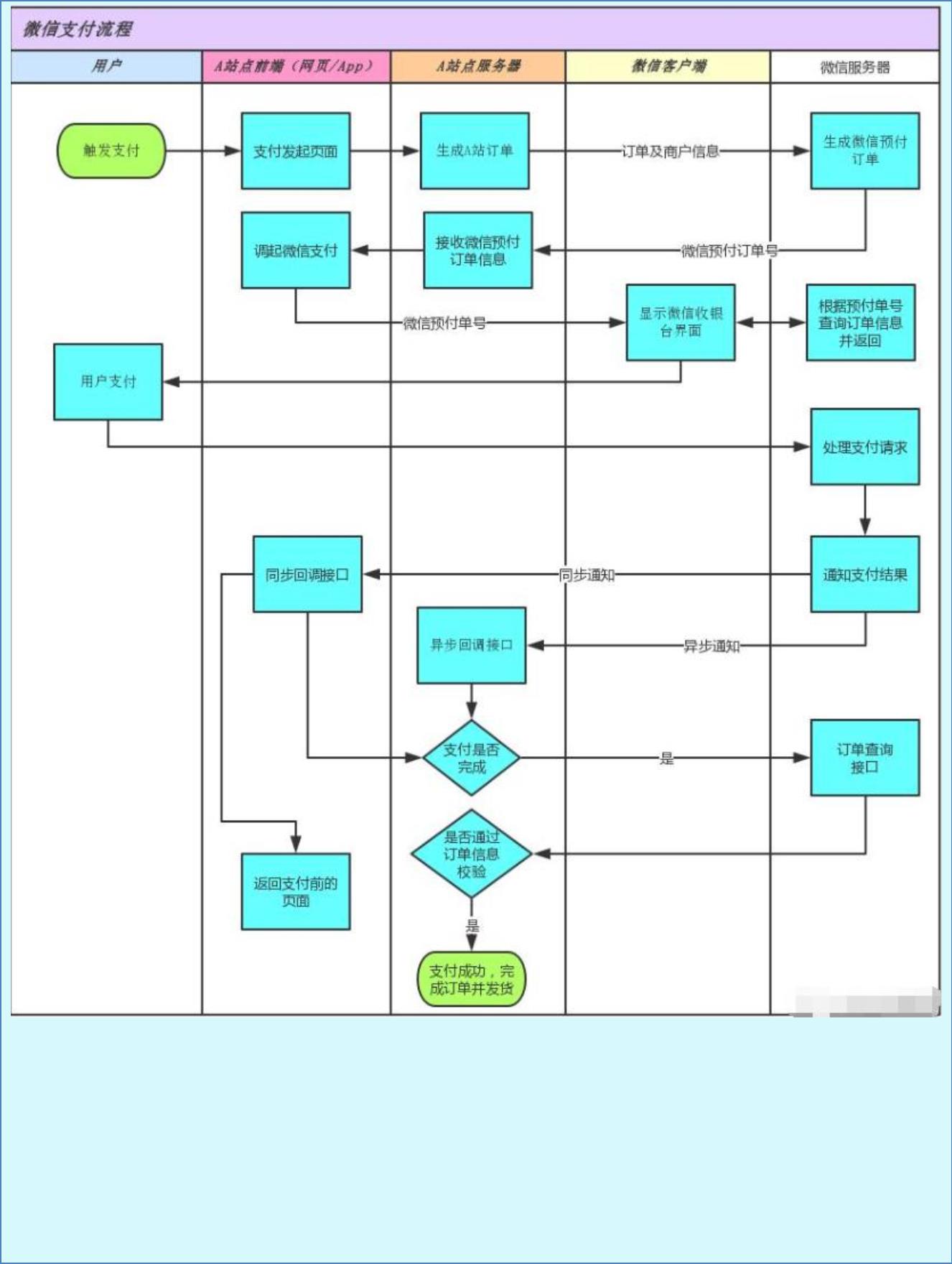
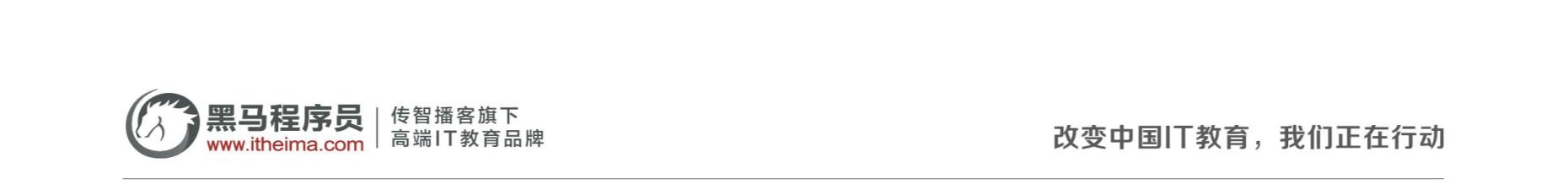
宝的下单接口，这样一来，如果安全、加密等做的不到位，很容易被恶意用户篡改信息。而

微信的下单接口是服务端调用的，A 站服务器只将获得的微信预付订单号返回给客户端，用

来发起调用微信客户端支付接口，这样一来，订单信息详情没有暴露在外，相对更安全一些

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



3.深度合作的积分抵扣模式

有时，一些站点或游戏会与某些第三方深度合作，将第三方的积分接入其中，以一定的

比例来抵扣和兑换其中的商品或道具。所谓深度合作，则是第三方会直接给出授权互信接口，

允许 A 站点直接调用扣减某用户的积分，用户只需将其第三方站点的账号与 A 站点的账号

绑定即可。这种模式，大多数情况下只有接口级的调用，没有前端收银台界面，流程图如下：

从上图可以看出，这种方式在流程上相对比较简单，只要 A 站点与第三方做好接口加

密、签名等安全措施，商定好结算规则，对接起来相对比较容易。另外，这种对接形式需要

第三方支持按照 A 站点的订单号来反查订单信息，方便对账结算和订单异常的自动化处理。

4.总结：支付系统对外接口的必备要素

a.信息加密：https、AES、RSA 非对称加密（公钥私钥）等；

b.签名规则：对传输的信息整体应用签名规则（sha256 等），生成签名字符串，用于防

止传输信息被篡改；

c.IP 白名单：对于服务器间调用的接口，可以将对方服务器 IP 加入白名单，防止非法 IP

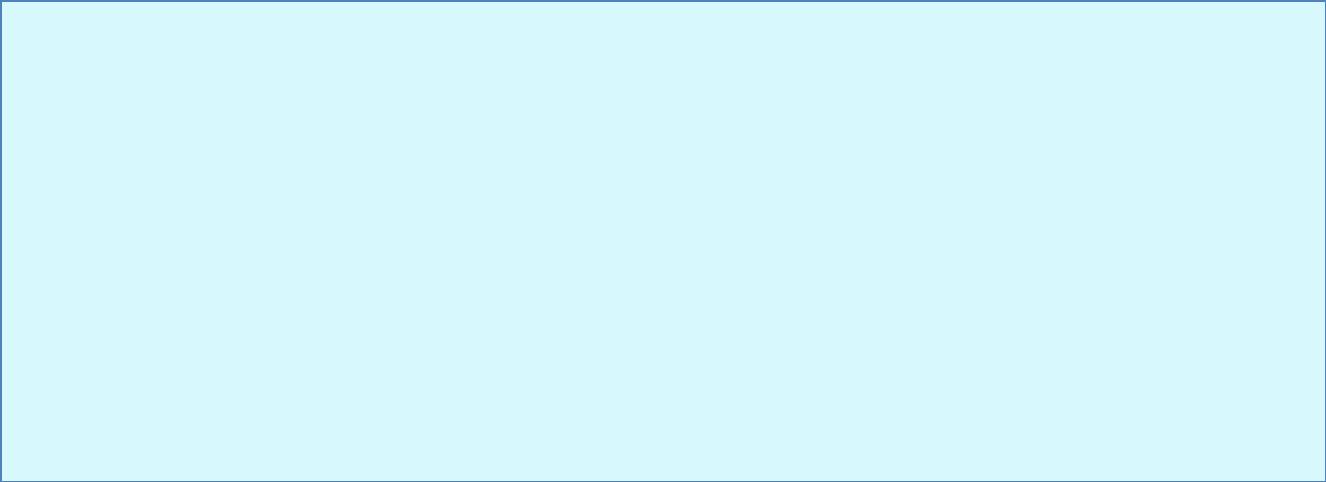
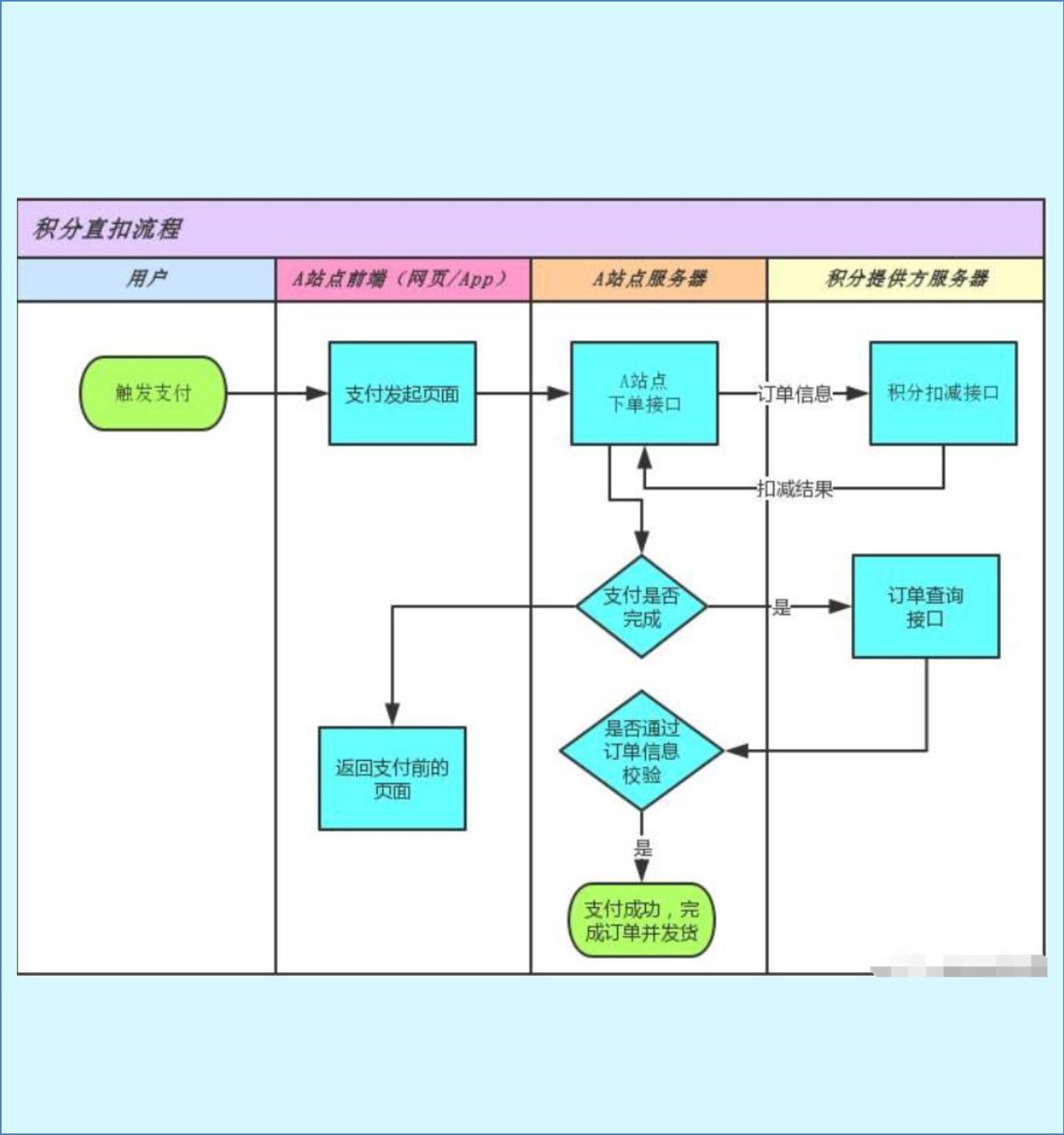
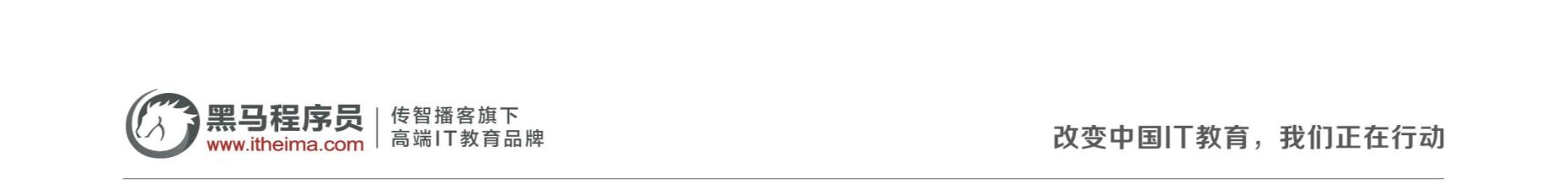
调用；

d.订单信息反查：A 站点主动向支付渠道查询订单信息，保证订单信息的有效性和准确

性。

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



PHP 加密处理方式

1.MD5 加密

$password = md5($\_POST["password"]);

MD5 的加密算法实在是显得有点简单了，而且很多破解密码的站点都存放了很多经过

MD5 加密的密码字符串，所以非常不提倡还在单单使用 MD5 来加密用户的密码的。

2.SHA1

$password=sha1($\_POST["password"]);

Md5()和 Sha1()加密算法是单向的，不存在逆向函数得到原来的明文数据

MD5()以 32 字符十六进制数字形式返回散列值。

Sha1()以 40 字符十六进制数字形式返回散列值。

3.SHA256 和 SHA512

SHA256 和 SHA512 都是来自于 SHA2 家族的加密函数，看名字就猜的出来了，这两个加

密方式分别生成 256 和 512 比特长度的 hash 字串。

他们的使用方法如下：

$password = hash("sha256", $password);

PHP 内置了 hash()函数，只需要将加密方式传给 hash()函数就好了。可以直接指明

sha256, sha512, md5, sha1 等加密方式。

4.URL 加密

urlencode，urldecode 这两个函数相对立——urlencode 对数据进行编码加密，urldecode

对数据进行编码解密

urlencode 的主要作用：

一.通常用于明文数据的隐藏

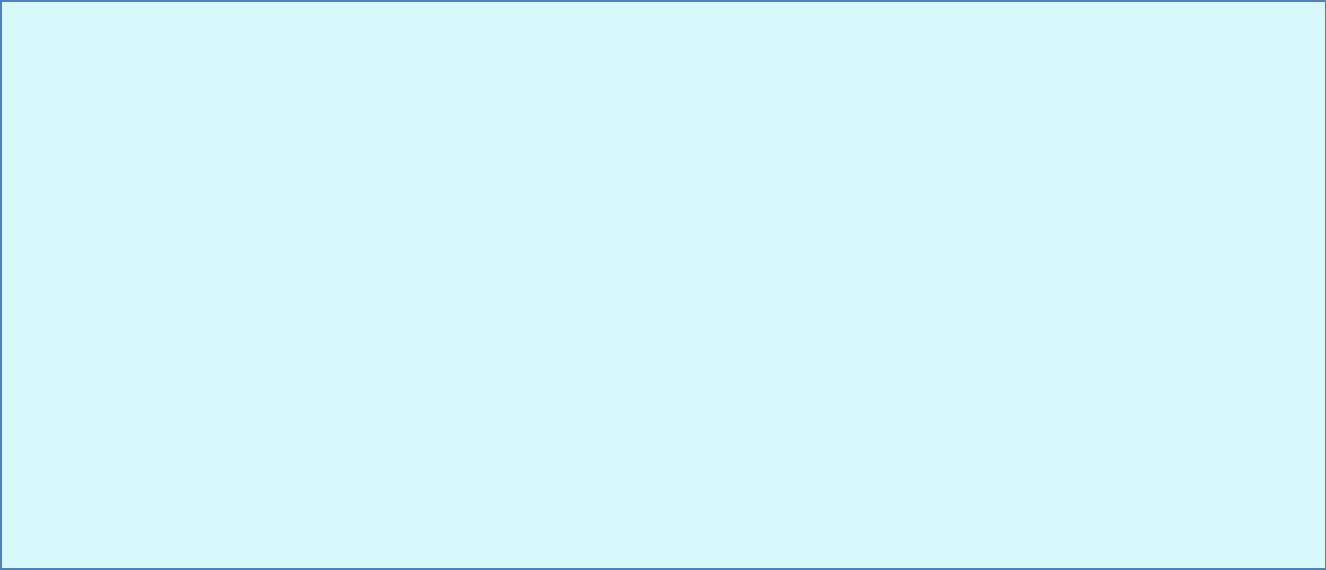
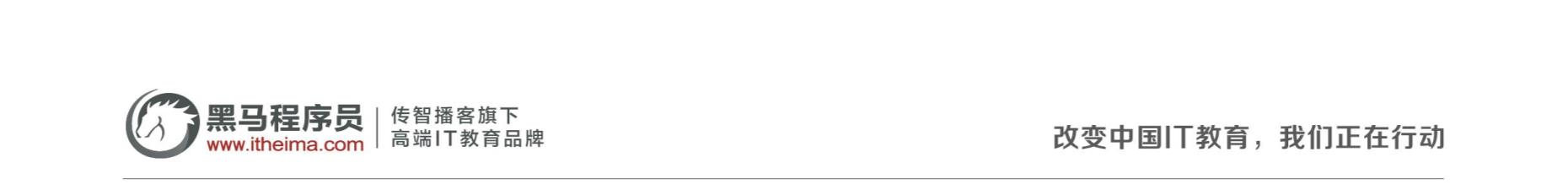
<?PHP

$url = 'admin.php?act=zhongguo&tx=123';

echo urlencode($url);

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



?>

二，针对 GET 提交数据进行编码

也许没有遇到过如下这种情况:

admin.php?act=del&add&data=abcd

这个东西什么玩意 act 是 del 还是 del&add。怎么办，这里就可以用 URL 编码去处理这

个问题，对 del&add 进行编码就可以使 act 可以等于 del&add,

<?php

$str = 'del&add';

$url = 'admin.php?act='.urlencode($str).'&data=abcd';

echo $url;

?>

urldecode 的主要作用：是对 urlencode 编码的数据进行解密

5.盐值

在加密的过程，还有一个非常常见的：盐值。对，在加密的时候其实会给加密的字符串添加

一个额外的字符串，以达到提高一定安全的目的：

function generateHashWithSalt($password) {

$intermediateSalt = md5(uniqid(rand(), true));

$salt = substr($intermediateSalt, 0, 6);

return hash("sha256", $password . $salt);

}

6.Bcrypt

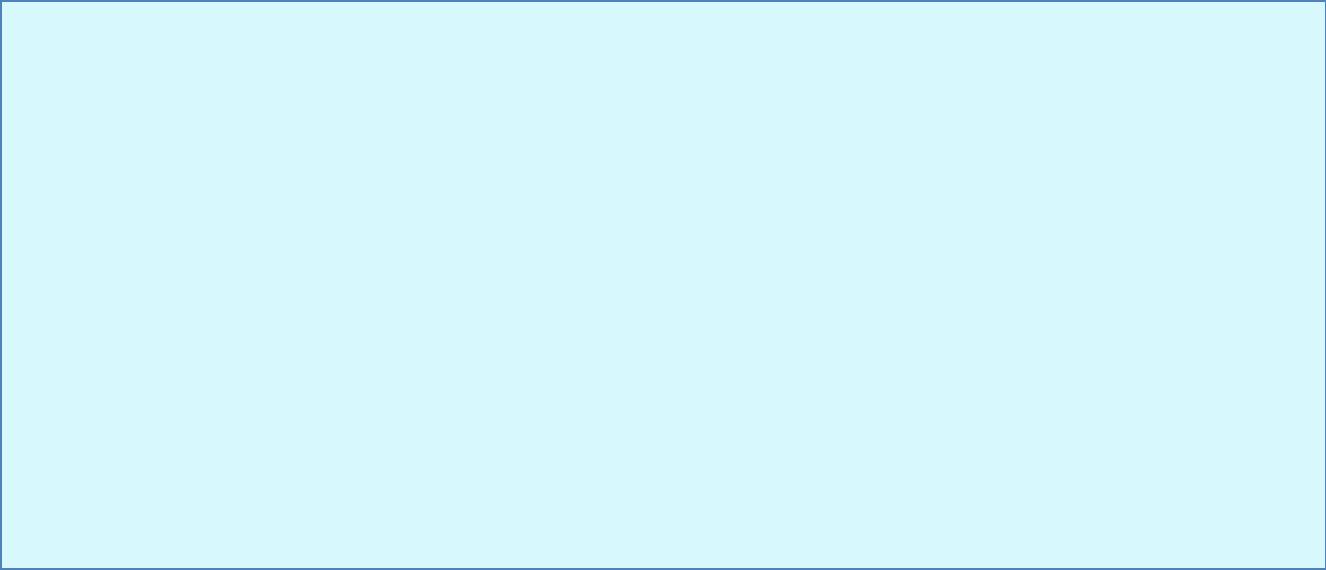
如果建议一种加密方式的话，Bcrypt 可能是推荐的最低要求了，因为会说到的 Hashing

API，不过 Bcrypt 也不失为一种比较不错的加密方式了。

function generateHash($password) {

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



if (defined("CRYPT\_BLOWFISH") && CRYPT\_BLOWFISH) {

$salt = '$2y$11$' . substr(md5(uniqid(rand(), true)), 0, 22);

return crypt($password, $salt);

}

}

Bcrypt 其实就是 Blowfish 和 crypt()函数的结合，这里通过 CRYPT\_BLOWFISH 判断 Blowfish

是否可用，然后像上面一样生成一个盐值，不过这里需要注意的是，crypt()的盐值必须以

$2a$或者$2y$开头

7.Password Hashing API（重要的加密方式）

Password Hashing API 是 PHP 5.5 之后才有的新特性，它主要是提供下面几个函数供我们

使用：

password\_hash() – 对密码加密.

password\_verify() – 验证已经加密的密码，检验其 hash 字串是否一致.

password\_needs\_rehash() – 给密码重新加密.

password\_get\_info() – 返回加密算法的名称和一些相关信息.

虽然说 crypt()函数在使用上已足够，但是 password\_hash()不仅可以使代码更加简短，

而且还在安全方面给了更好的保障，所以，现在 PHP 的官方都是推荐这种方式来加密用户

的密码，很多流行的框架比如 Laravel 就是用的这种加密方式。

$hash = password\_hash($passwod, PASSWORD\_DEFAULT);

对，就是这么简单，一行代码，All done。

PASSWORD\_DEFAULT 目前使用的就是 Bcrypt，所以在上面说推荐这个，不过因为

Password Hashing API 做得更好了。这里需要注意的是，如果代码使用的都是

PASSWORD\_DEFAULT 加密方式，那么在数据库的表中，password 字段就得设置超过 60 个字

符长度，也可以使用 PASSWORD\_BCRYPT，这个时候，加密后字串总是 60 个字符长度。

这里使用 password\_hash()完全可以不提供盐值(salt)和消耗值 (cost)，可以将后者理解为

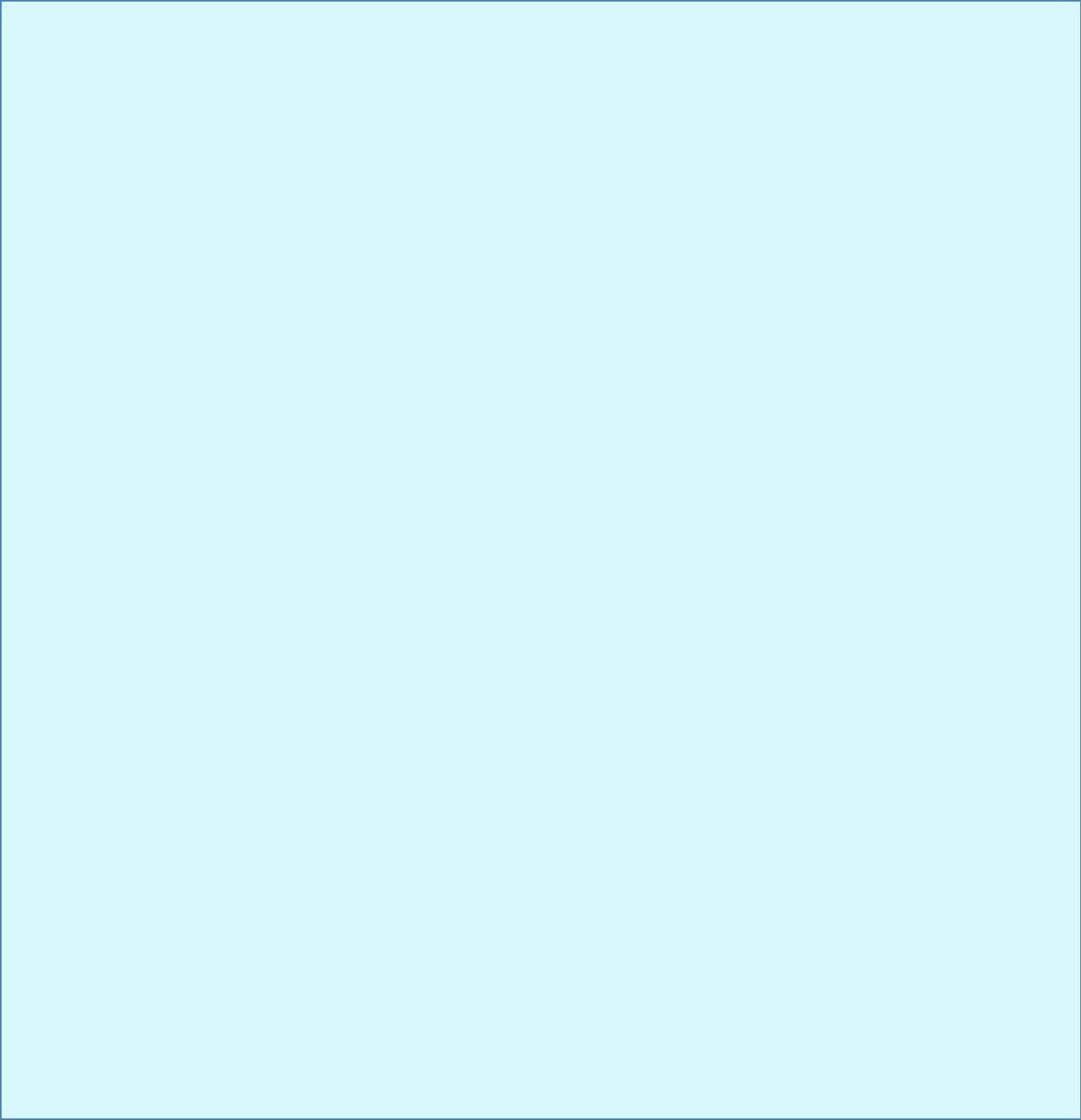
一种性能的消耗值，cost 越大，加密算法越复杂，消耗的内存也就越大。当然，如果需要指

定对应的盐值和消耗值，你可以这样写：

$options = [

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090



'salt' => custom\_function\_for\_salt(), //编写代码生成合适的 salt

'cost' => 12 // the default cost is 10];$hash = password\_hash($password,

PASSWORD\_DEFAULT, $options);

密码加密过后，需要对密码进行验证，以此来判断用户输入的密码是否正确：

if (password\_verify($password, $hash)) {

// Pass}else {

// Invalid

}

直接使用 password\_verify 就可以对之前加密过的字符串（存在数据库中）进行验证了。

然而，如果有时候需要更改加密方式，突然想更换一下盐值或者提高一下消耗值，这时

候就要使用到 password\_needs\_rehash()函数了：

if (password\_needs\_rehash($hash, PASSWORD\_DEFAULT, ['cost' => 12])) {

// cost 变成 12

$hash = password\_hash($password, PASSWORD\_DEFAULT, ['cost' => 12]);

// 不要忘了存储新的 hash

}

只有这样，PHP 的 Password Hashing API 才会知道重现更换了加密方式，这样的主要目

的就是为了后面的密码验证。

简单地说一下 password\_get\_info()，这个函数一般可以看到下面三个信息：

algo – 算法实例

algoName – 算法名字

options – 加密时候的可选参数

北京市昌平区建材城西路金燕龙办公楼一层

电话：400-618-9090

