大纲

- **前言**
  - 为什么需要“大规模计算” [HPC, DL, Business platform system, Cloud已经合流]
    - 导入 – 科学计算(天气预报)，DL, 互联网平台(Google, Amazon, Alibaba, MeiTuan, …)
- **基础篇**
  - 并发程序的样子 – Divide & Conquer, Model & Challenges, PCAM, Data/Task, …
    - 天气预报的计算
  - 运行环境
    - 硬件 – 自己梳理的3个方案 – Shared/Unshared Memory, Hybrid
    - 系统软件 – 协议栈, Modern OS, Distributed Job Scheduler, GTM等
- **算法级篇**
  - OpenMP, MPI, CUDA (DL的实现), Big Data 中的MR/Spark等 (只涉及在Big Data SDK之上的编程；大数据本身的介绍放到后一部分)
- **系统级篇 – 互联网平台的实现**
  - “秒杀”的技术架构
  - 计算广告
  - 系统架构 (HTAP等)
    - Flink, ClickHouse, MaxCompute, ELK …

Chapter 6: HUGE Concurrency architecture for “秒杀”

- **Architecture for “秒杀”**
  - Just “Divide and Conquer” – Data level, Module level
  - Big Data + Cloud now!
- **So-called Software architect [软件架构师]**

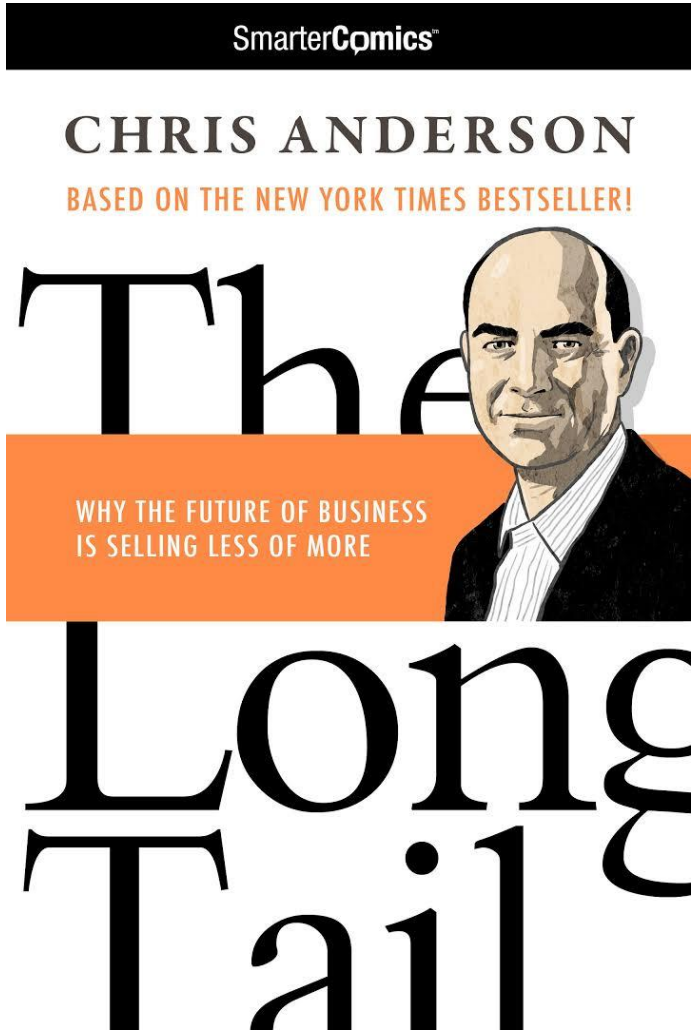
- **觉得还是要讲讲现在电子商务平台的背景 –**
  - 盈利 – 抽成，广告， 云计算，金融 (电子支付、小额贷款等)
    - 电子支付 – 延期转款等，但是，只是收集了钱还得生钱才行 (就像银行存款，还需要贷款出去，得到高额差价)
    - 所以，才有了支付宝那样的“花呗”、“理财产品” 等 – 可以说是野蛮生长了 (侵占了银行的功能)
  - 首先要做的就是 – 如何吸引庞大的用户?
    - 电商 – Super Market 的其实：货品丰富、质优价廉呗 – 长尾效应 (Long Tail)
    - 其他 – 自己尝试梳理下：Google, FB, Twitter, Baidu, MeiTuan, Youtube, WeChat, 抖音，西瓜，…

Background

We are now in IT age/ML age/AI age …

- **Platform is a popular business pattern for e-commerce**
  - Many great companies

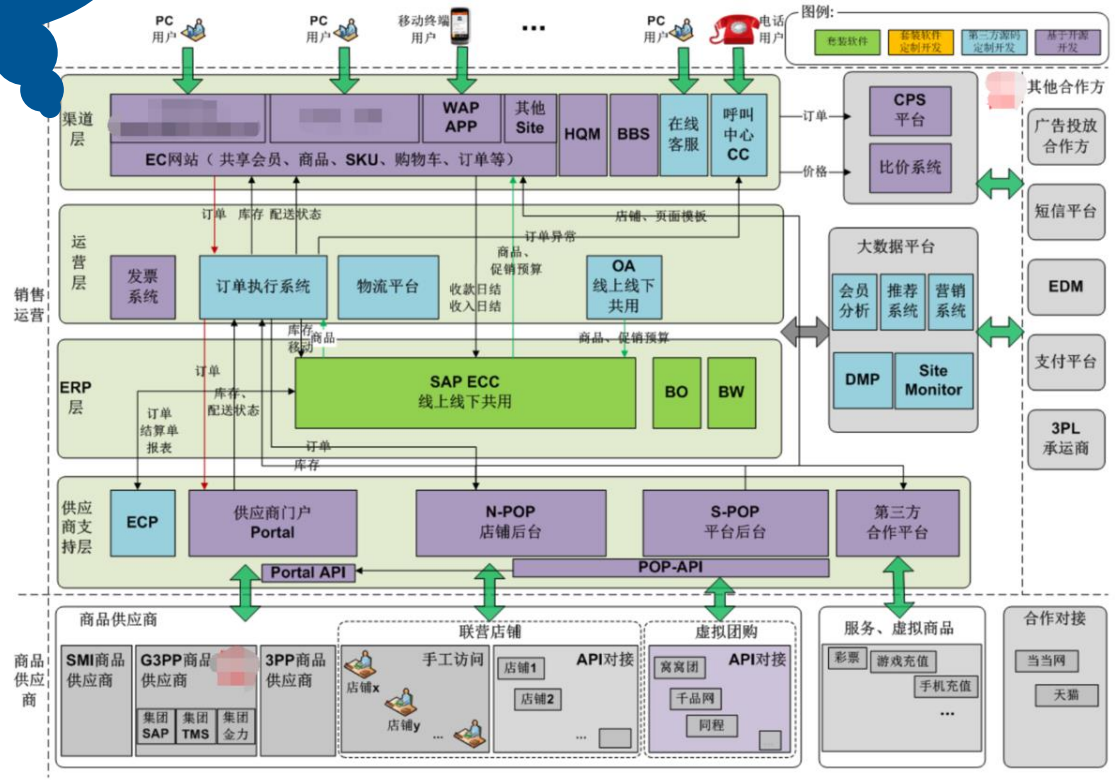
Long tail explains the Business @ IT age/E-commerce era



背后的技术支撑

Computers and data storage are **Distributed**!

- **Business pattern i**
  - How to earn money!
- **Then, to carry out the "long tail", we need the support of HUGE scale data (products, customers, …) and computing power**
  - You have to consider to do business with the whole world! – of course in many stages
- **So many –**
  - Customers, Computers, Data,
  - with real-time processing request

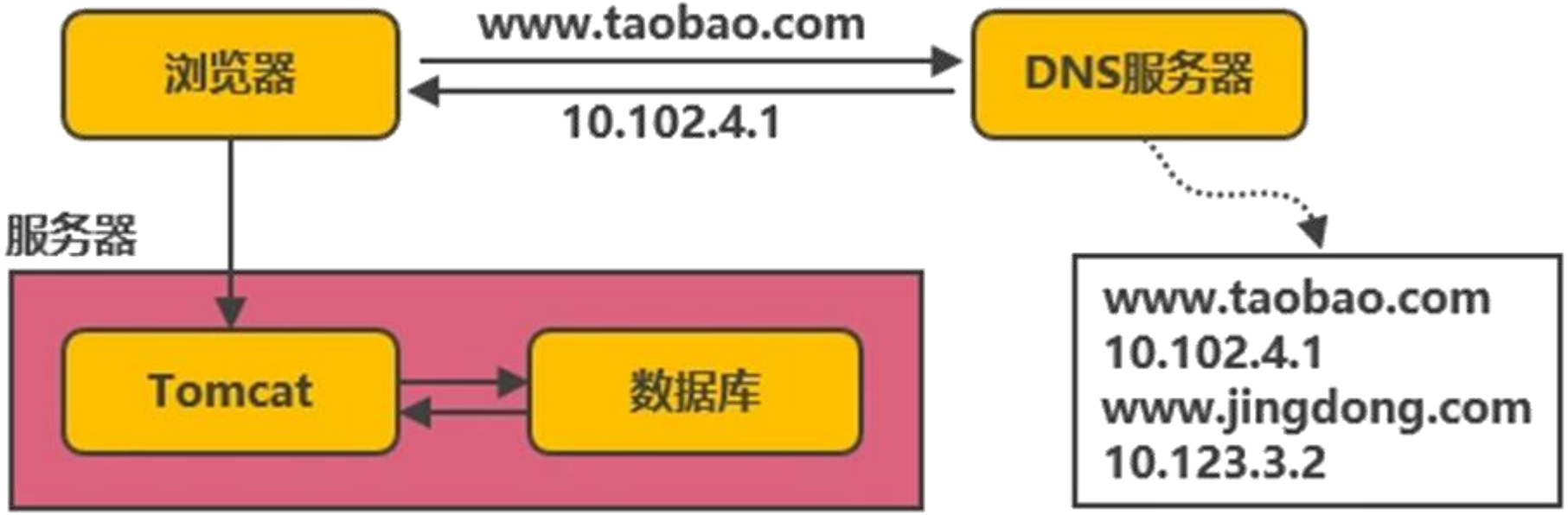


We focus on the tech – how to support the HUGE Concurrency?!

- **Then, to carry out the "long tail", we need the support of HUGE scale data (products, customers, …) and computing power**
  - You have to consider to do business with the whole world! – of course in many stages
- **So many –**
  - Customers, Computers, Data, with real-time processing request

- **分布式**
  - 系统中的多个模块在不同服务器上部署，即可称为分布式系统，
    - 如Tomcat和数据库分别部署在不同的服务器上，或两个相同功能的Tomcat分别部署在不同服务器上
- **高可用**
  - 系统中部分节点失效时，其他节点能够接替它继续提供服务，则可认为系统具有高可用性
- **集群**
  - 一个特定领域的软件部署在多台服务器上并作为一个整体提供一类服务，这个整体称为集群。
    - 如Zookeeper中的Master和Slave分别部署在多台服务器上，共同组成一个整体提供集中配置服务。在常见的集群中，客户端往往能够连接任意一个节点获得服务，并且当集群中一个节点掉线时，其他节点往往能够自动的接替它继续提供服务，这时候说明集群具有高可用性
- **负载均衡**
  - 请求发送到系统时，通过某些方式把请求均匀分发到多个节点上，使系统中每个节点能够均匀的处理请求负载，则可认为系统是负载均衡的
- **正向代理和反向代理**
  - 系统内部要访问外部网络时，统一通过一个代理服务器把请求转发出去，在外部网络看来就是代理服务器发起的访问，此时代理服务器实现的是正向代理；当外部请求进入系统时，代理服务器把该请求转发到系统中的某台服务器上，对外部请求来说，与之交互的只有代理服务器，此时代理服务器实现的是反向代理。简单来说，正向代理是代理服务器代替系统内部来访问外部网络的过程，反向代理是外部请求访问系统时通过代理服务器转发到内部服务器的过程。

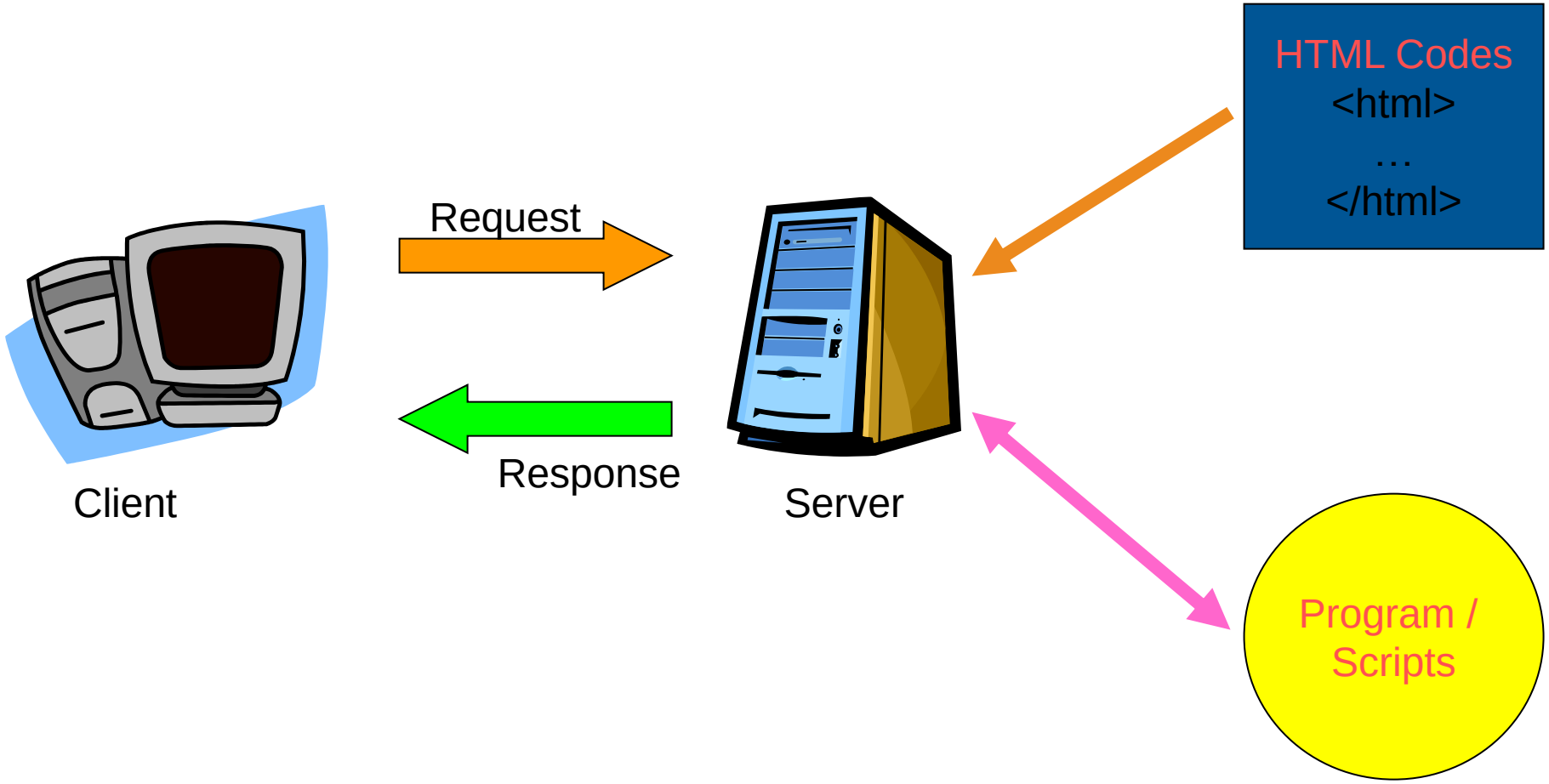
1 单机架构



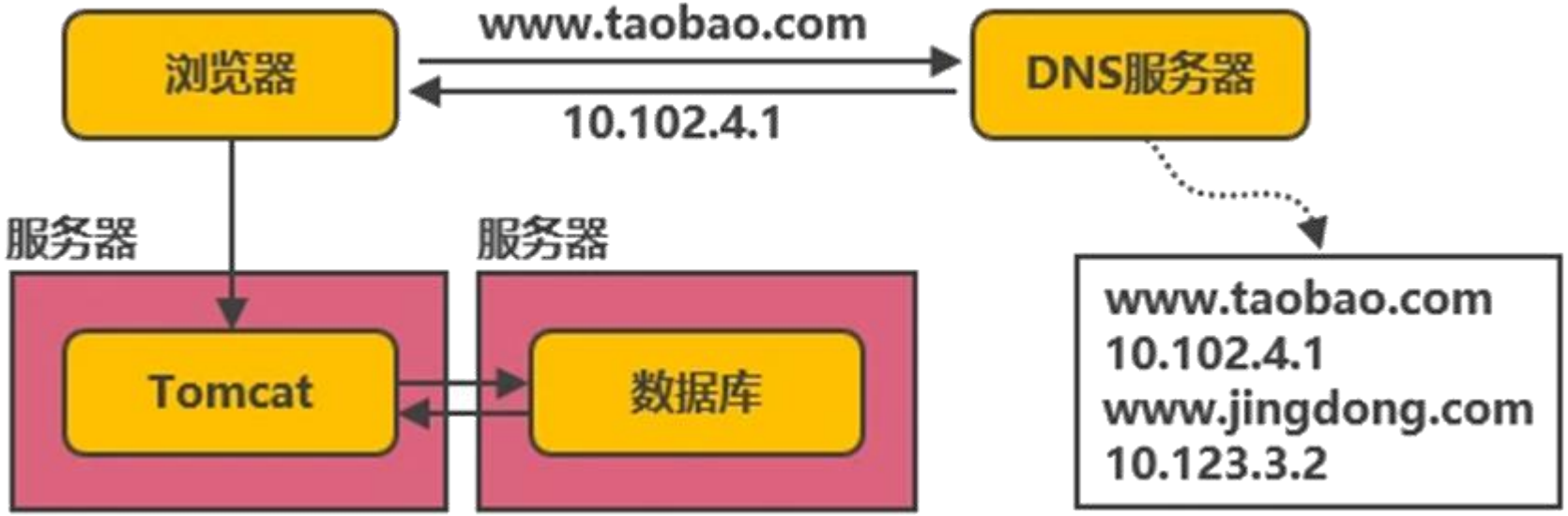
- 以淘宝作为例子。在网站最初时，应用数量与用户数都较少，可以把Tomcat和数据库部署在同一台服务器上。浏览器往www.taobao.com发起请求时，首先经过DNS服务器（域名系统）把域名转换为实际IP地址10.102.4.1，浏览器转而访问该IP对应的Tomcat。
- **随着用户数的增长，Tomcat和数据库之间竞争资源，单机性能不足以支撑业务**

By "Web-based Application"

- It's a quite popular application style every SSE student should know!



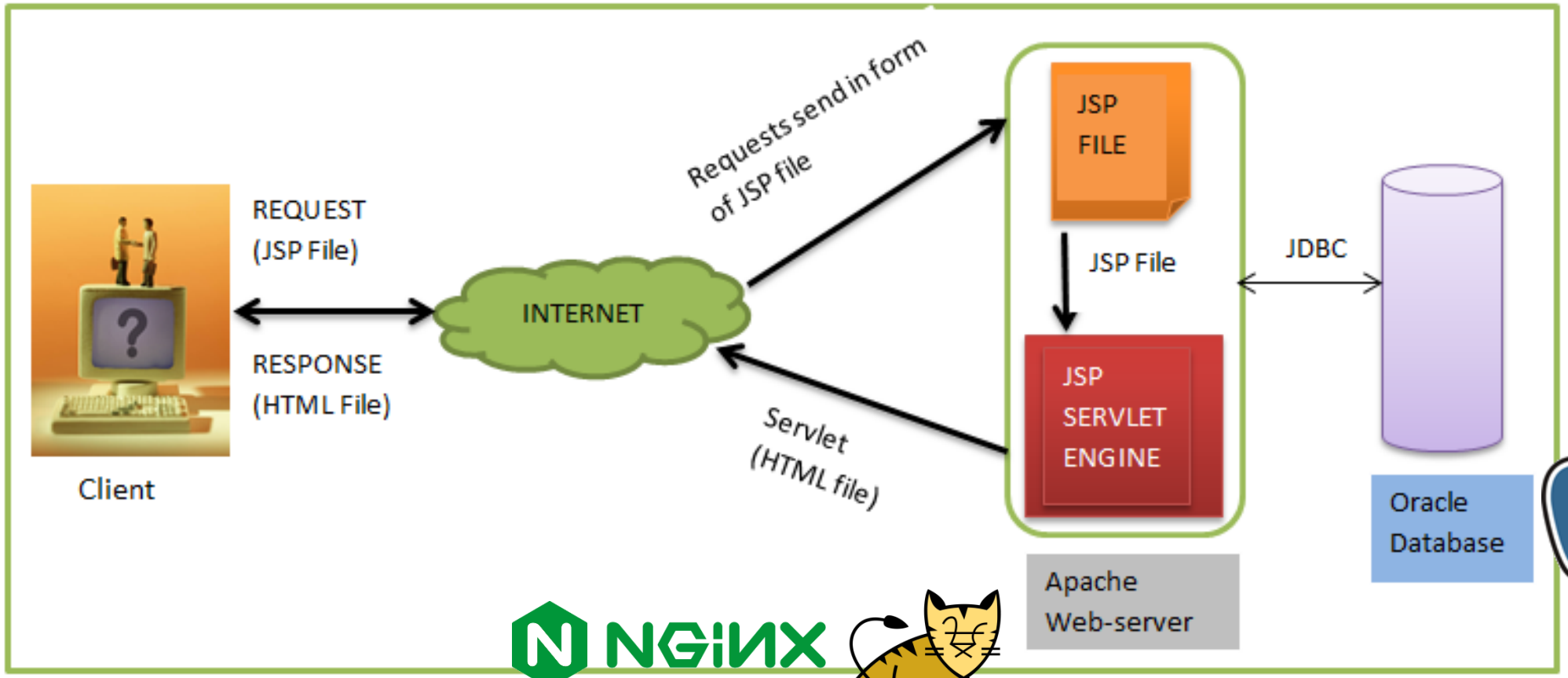
2 第一次演进：Tomcat与数据库分开部署



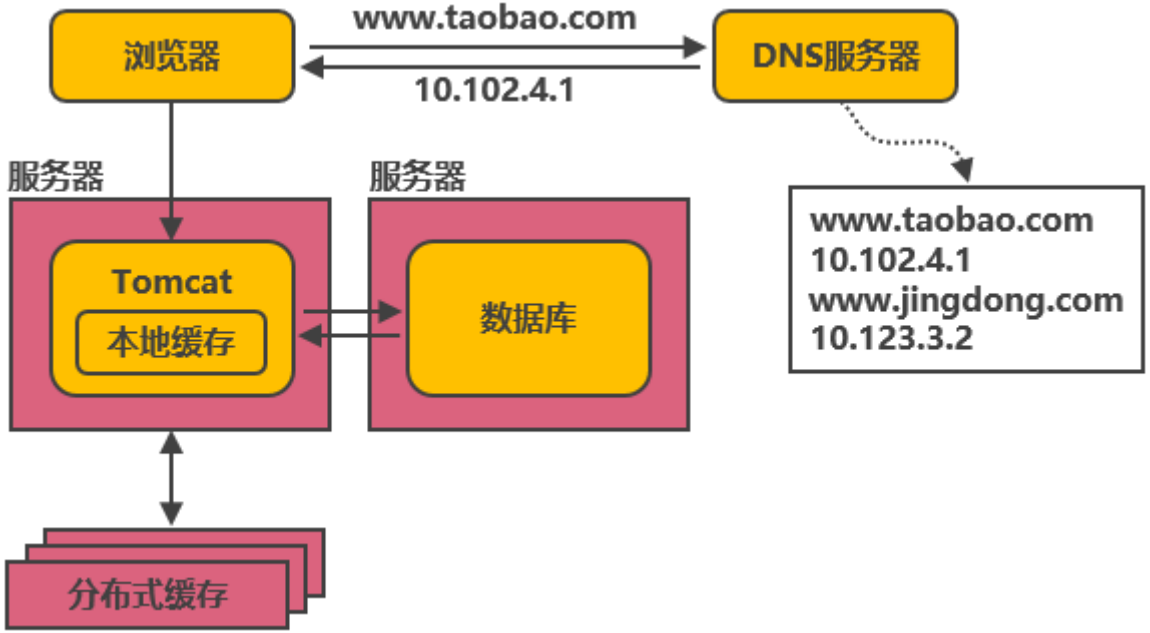
- Tomcat和数据库分别独占服务器资源，显著提高两者各自性能。

- **随着用户数的增长，并发读写数据库成为瓶颈**

3 tier architecture



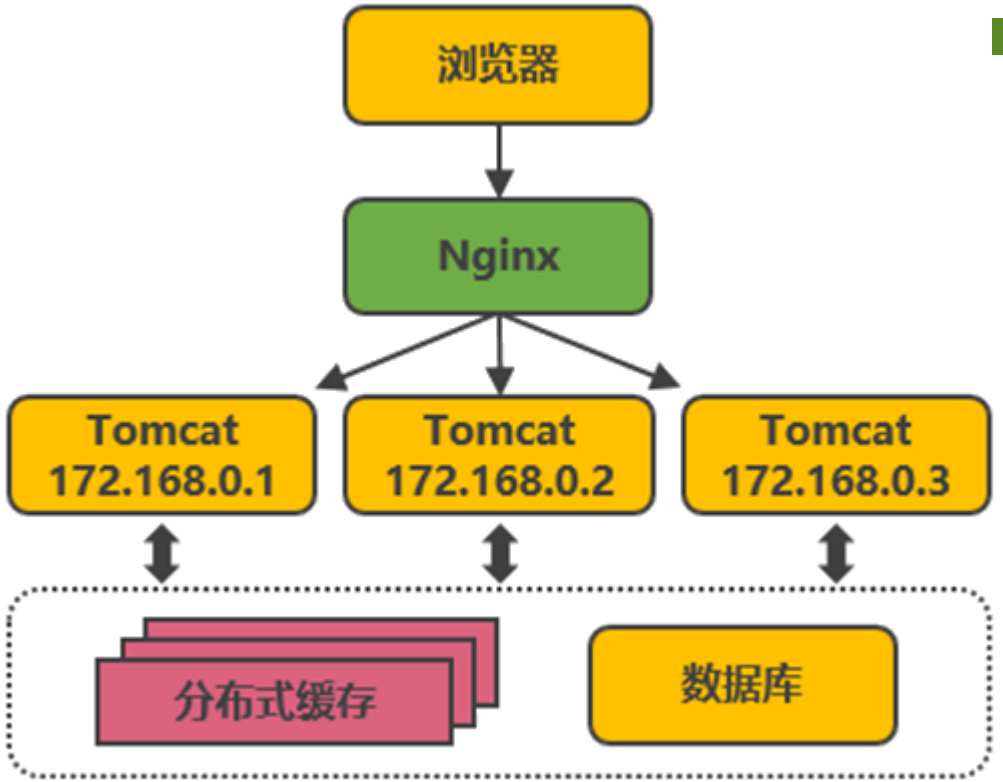
3 第二次演进：引入本地缓存和分布式缓存



- 在Tomcat同服务器上或同JVM中增加本地缓存，并在外部增加分布式缓存，缓存热门商品信息或热门商品的html页面等。通过缓存能把绝大多数请求在读写数据库前拦截掉，大大降低数据库压力。其中涉及的技术包括：使用memcached作为本地缓存，使用Redis作为分布式缓存，还会涉及缓存一致性、缓存穿透/击穿、缓存雪崩、热点数据集中失效等问题。

- **缓存抗住了大部分的访问请求，随着用户数的增长，并发压力主要落在单机的Tomcat上，响应逐渐变慢**

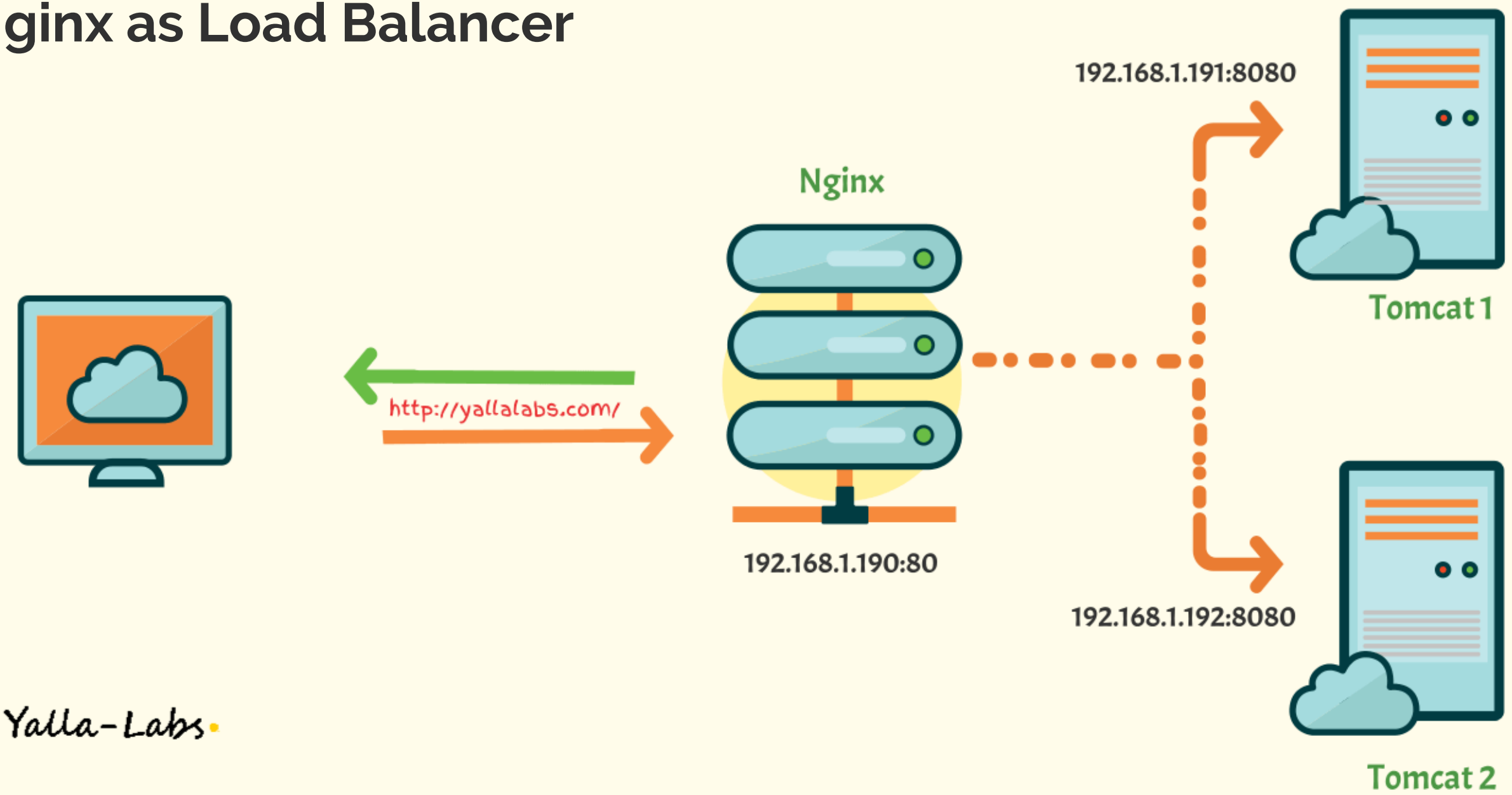
4 第三次演进：引入反向代理实现负载均衡



- 在多台服务器上分别部署Tomcat，使用反向代理软件（Nginx）把请求均匀分发到每个Tomcat中。此处假设Tomcat最多支持100个并发，Nginx最多支持50000个并发，那么理论上Nginx把请求分发到500个Tomcat上，就能抗住50000个并发。其中涉及的技术包括：Nginx、HAProxy，两者都是工作在网络第七层的反向代理软件，主要支持http协议，还会涉及session共享、文件上传下载的问题。

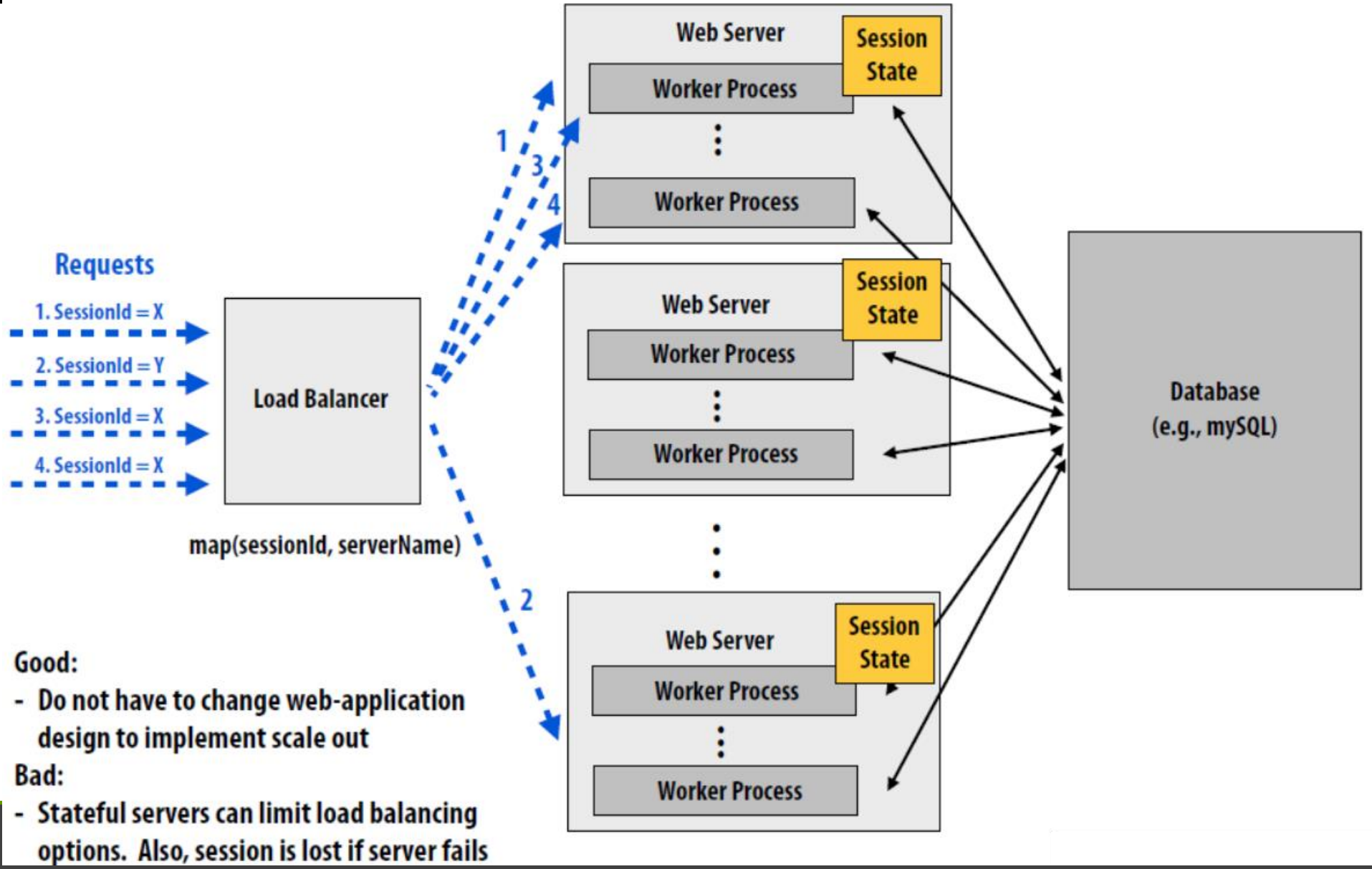
- **反向代理使应用服务器可支持的并发量大大增加，但并发量的增长也意味着更多请求穿透到数据库，单机的数据库最终成为瓶颈**

Nginx as Load Balancer



Load balancing with persistence

All requests associated with a session are directed to the same server (aka. session affinity, "sticky sessions")



Good:

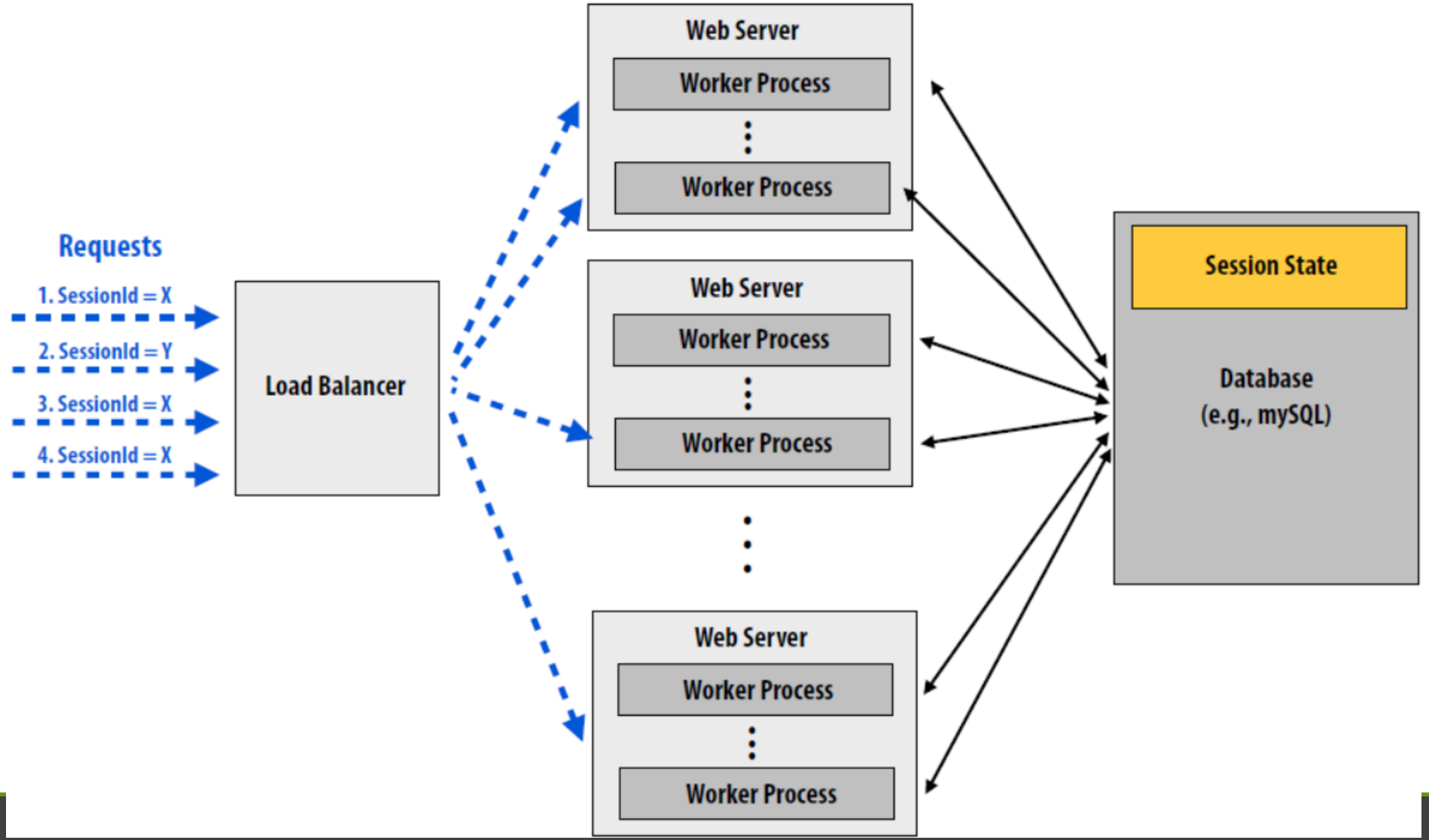
- Do not have to change web-application design to implement scale out

Bad:

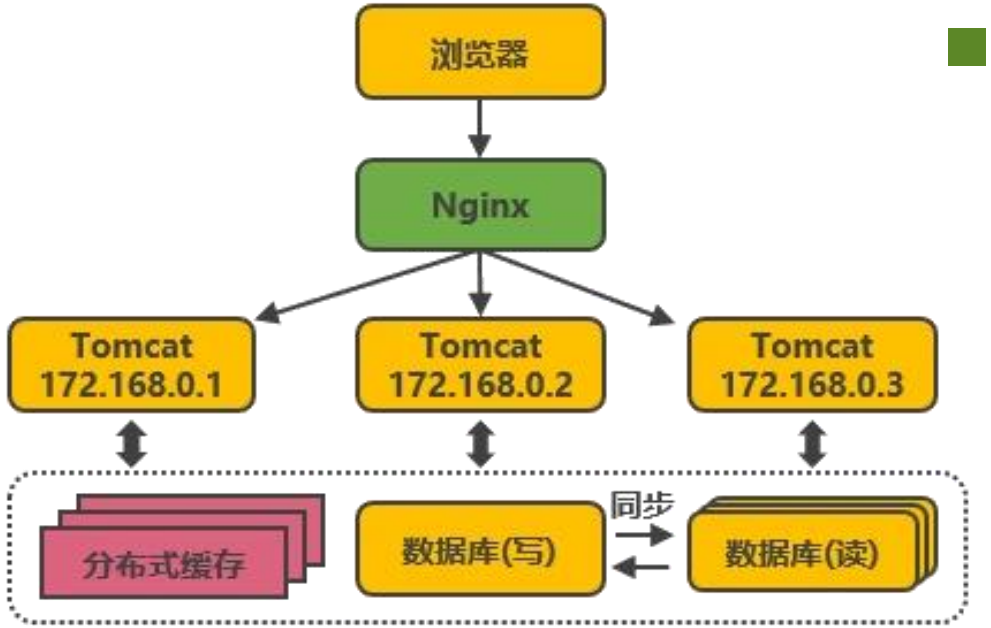
- Stateful servers can limit load balancing options. Also, session is lost if server fails

Desirable: avoid persistent state in web server

Maintain stateless servers, treat sessions as persistent data to be stored in the DB.



5 第四次演进：数据库读写分离

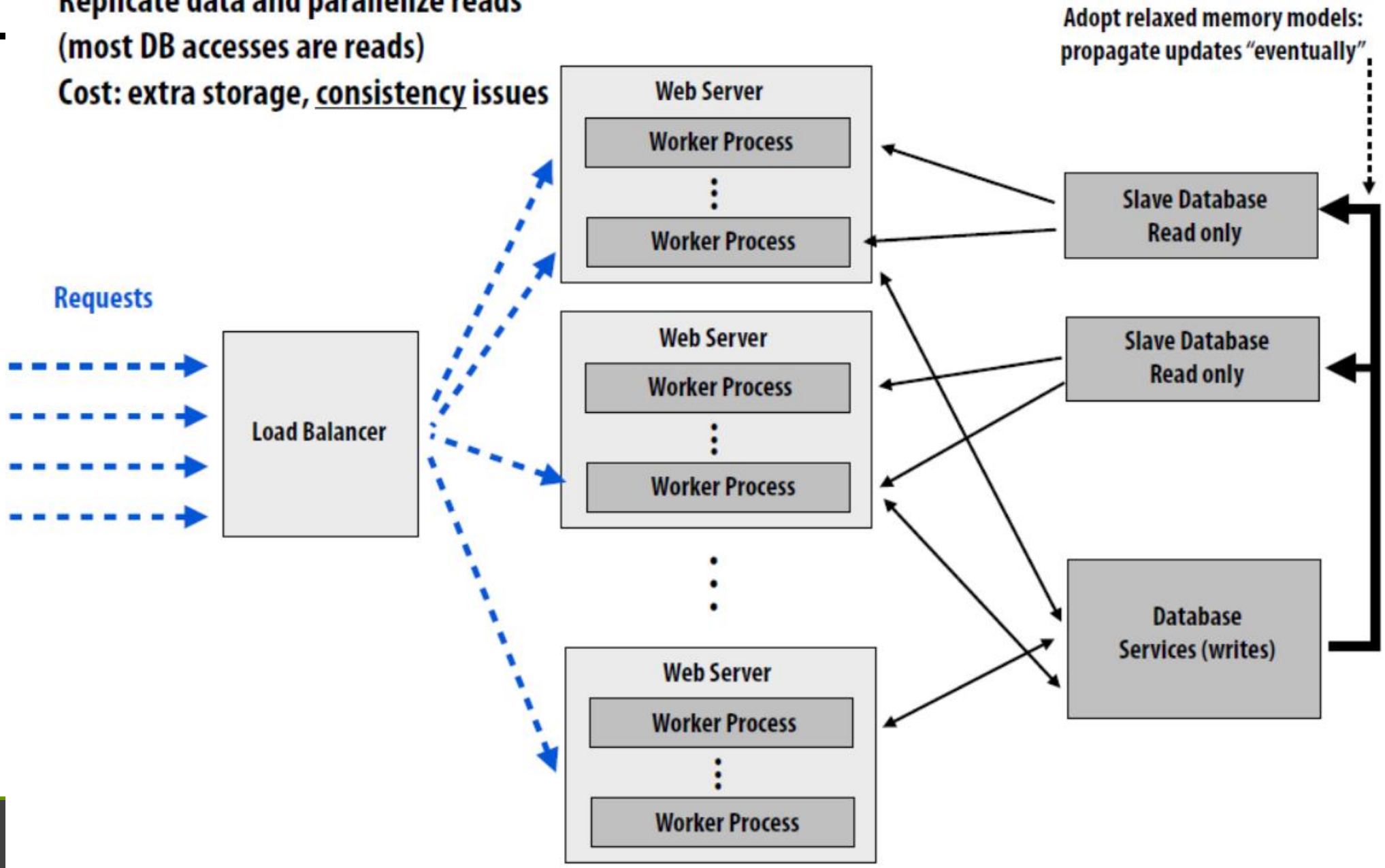


- 把数据库划分为读库和写库，读库可以有多个，通过同步机制把写库的数据同步到读库，对于需要查询最新写入数据场景，可通过在缓存中多写一份，通过缓存获得最新数据。其中涉及的技术包括：Mycat，它是数据库中间件，可通过它来组织数据库的分离读写和分库分表，客户端通过它来访问下层数据库，还会涉及数据同步，数据一致性的问题。

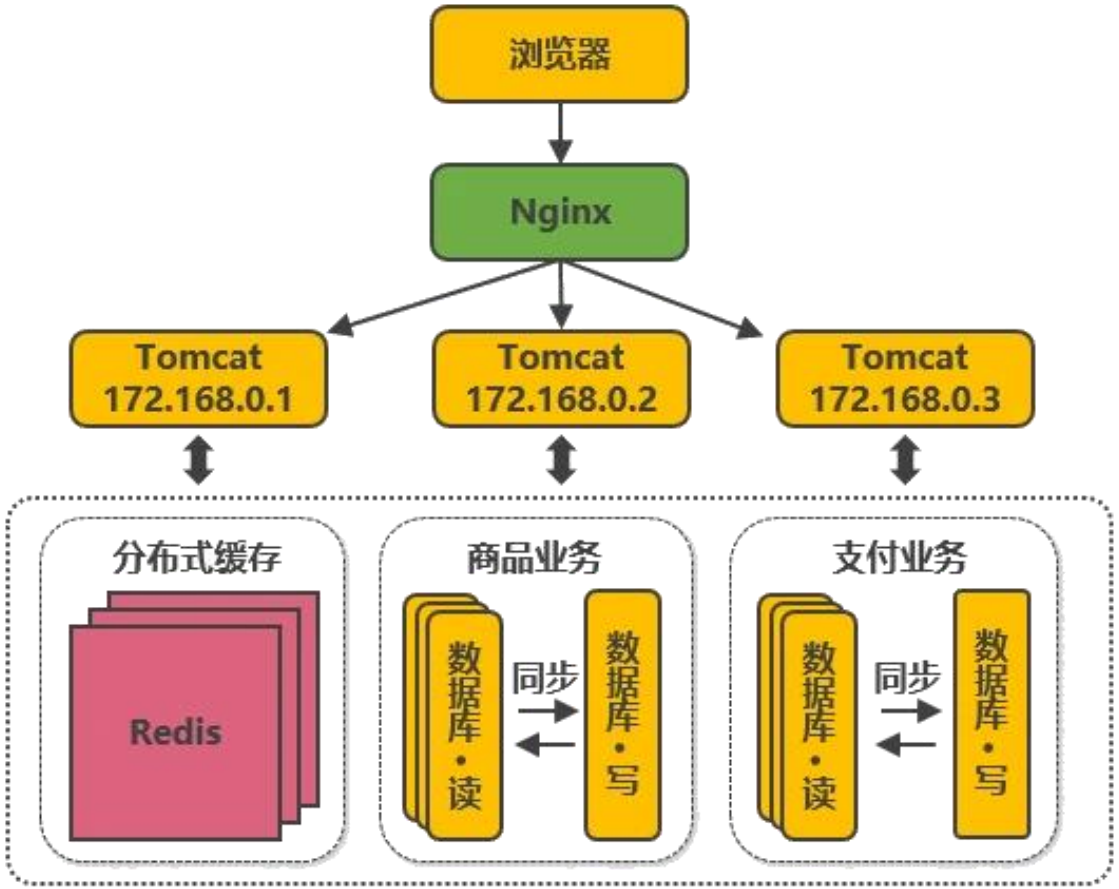
- **业务逐渐变多，不同业务之间的访问量差距较大，不同业务直接竞争数据库，相互影响性能**

Scaling out a database: replicate

Replicate data and parallelize reads
(most DB accesses are reads)
Cost: extra storage, consistency issues



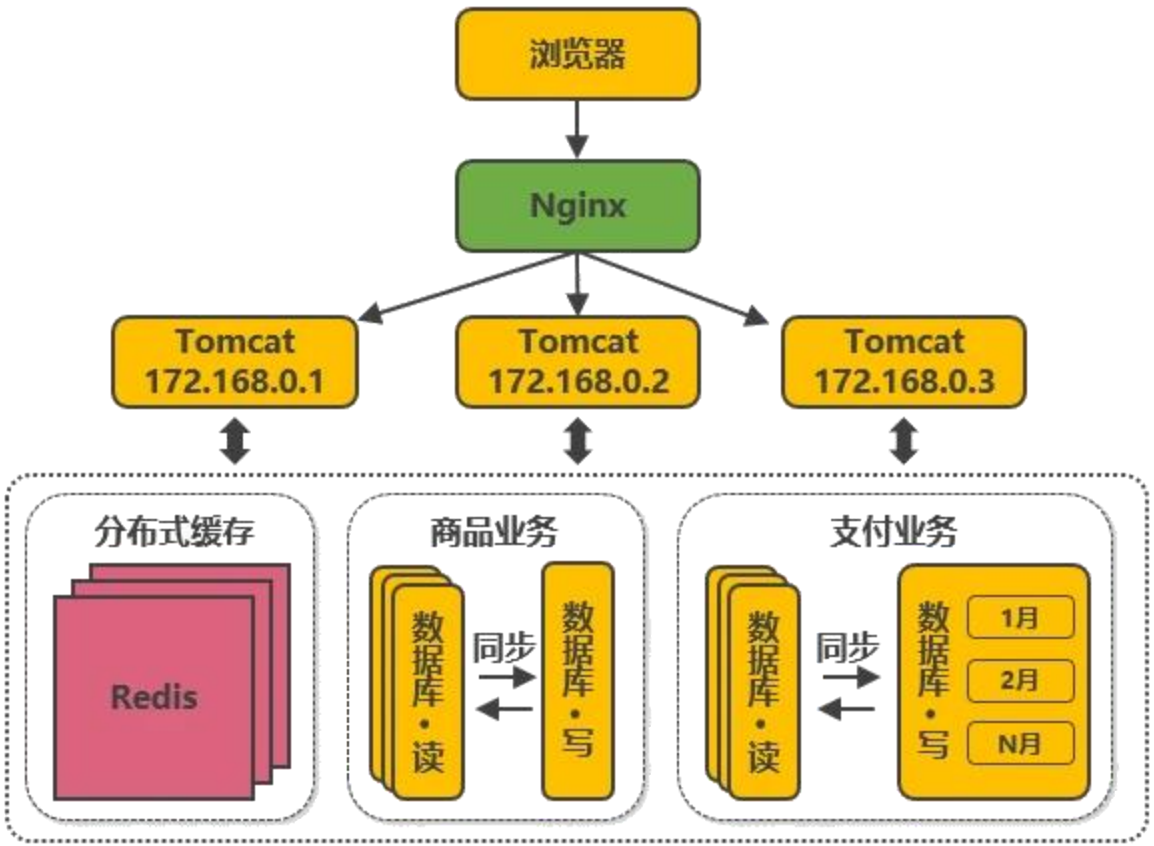
6 第五次演进：数据库按业务分库



- 把不同业务的数据保存到不同的数据库中，使业务之间的资源竞争降低，对于访问量大的业务，可以部署更多的服务器来支撑。这样同时导致跨业务的表无法直接做关联分析，需要通过其他途径来解决，但这不是本文讨论的重点，有兴趣的可以自行搜索解决方案。

- **随着用户数的增长，单机的写库会逐渐会达到性能瓶颈**

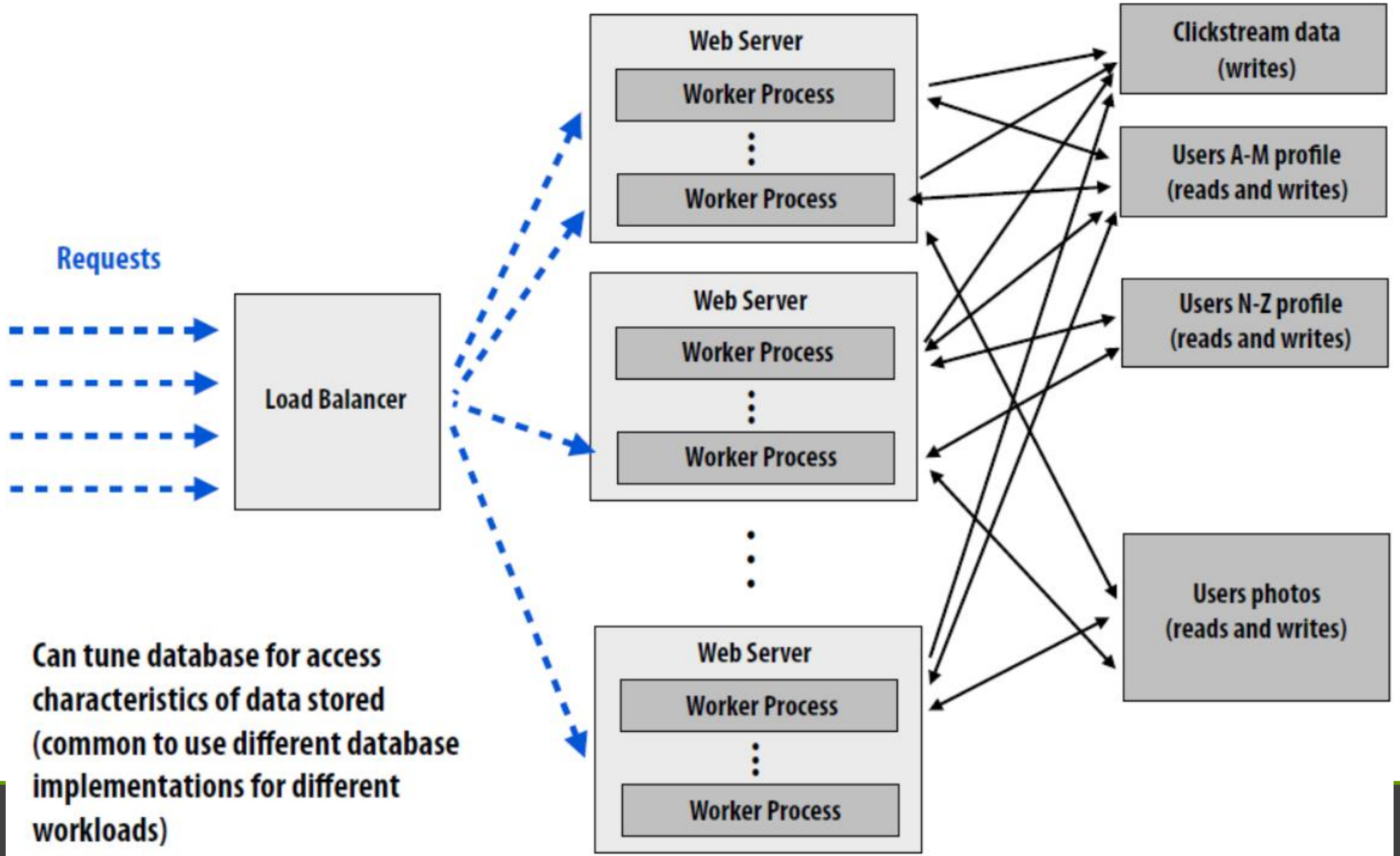
7 第六次演进：把大表拆分为小表



- 比如针对评论数据，可按照商品ID进行hash，路由到对应的表中存储；针对支付记录，可按照小时创建表，每个小时表继续拆分为小表，使用用户ID或记录编号来路由数据。只要实时操作的表数据量足够小，请求能够足够均匀的分发到多台服务器上的小表，那数据库就能通过水平扩展的方式来提高性能。其中前面提到的Mycat也支持在大表拆分为小表情况下的访问控制。
- 这种做法显著的增加了数据库运维的难度，对DBA的要求较高。数据库设计到这种结构时，已经可以称为分布式数据库，但是这只是一个逻辑的数据库整体，数据库里不同的组成部分是由不同的组件单独来实现的，如分库分表的管理和请求分发，由Mycat实现，SQL的解析由单机的数据库实现，读写分离可能由网关和消息队列来实现，查询结果的汇总可能由数据库接口层来实现等等，这种架构其实是MPP（大规模并行处理）架构的一类实现。
- 目前开源和商用都已经有不少MPP数据库，开源中比较流行的有Greenplum、TiDB、Postgresql XC、HAWQ等，商用的如南大通用的GBase、睿帆科技的雪球DB、华为的LibrA等等，不同的MPP数据库的侧重点也不一样，如TiDB更侧重于分布式OLTP场景，Greenplum更侧重于分布式OLAP场景，这些MPP数据库基本都提供了类似Postgresql、Oracle、MySQL那样的SQL标准支持能力，能把一个查询解析为分布式的执行计划分发到每台机器上并行执行，最终由数据库本身汇总数据进行返回，也提供了诸如权限管理、分库分表、事务、数据副本等能力，并且大多能够支持100个节点以上的集群，大大降低了数据库运维的成本，并且使数据库也能够实现水平扩展。

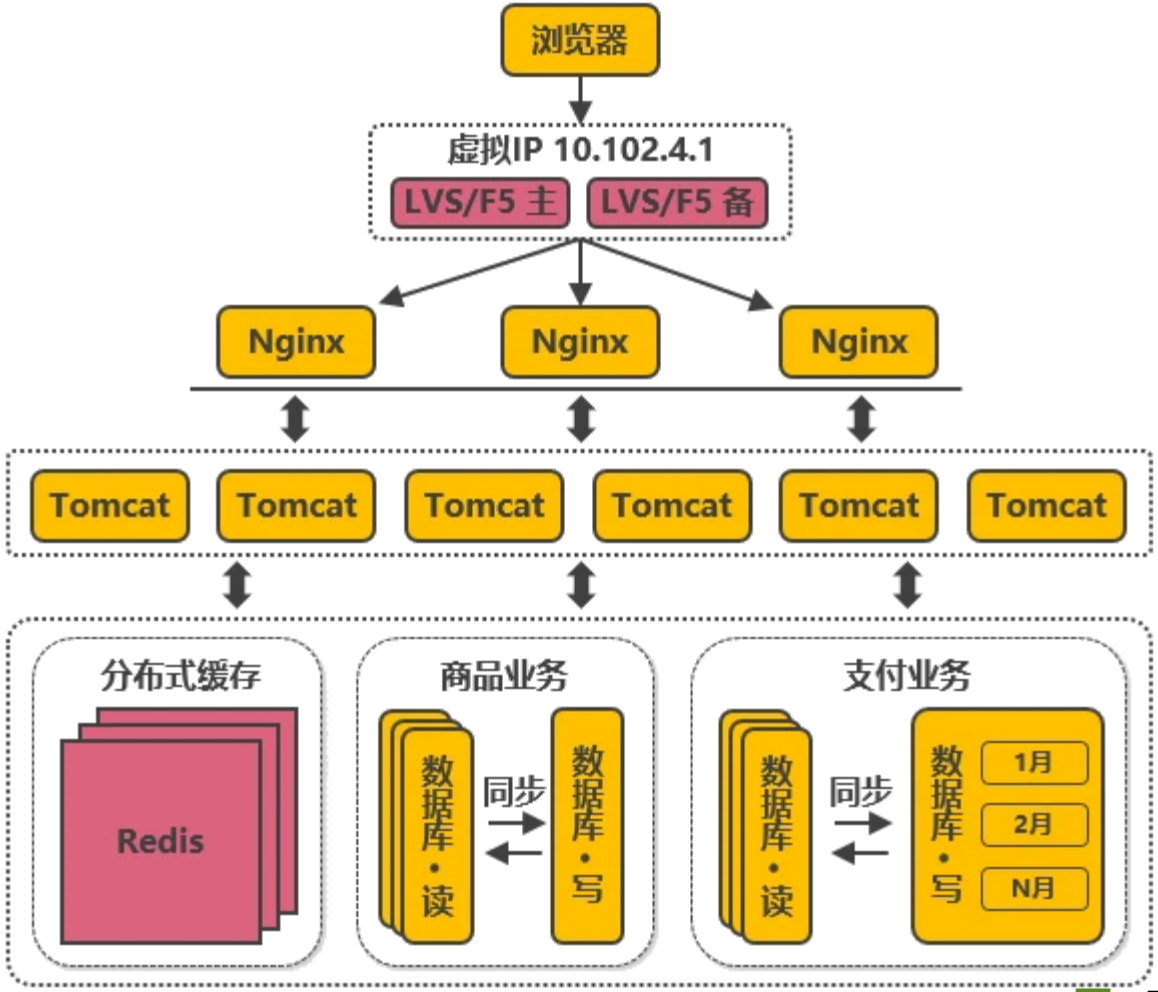
- **数据库和Tomcat都能够水平扩展，可支撑的并发大幅提高，随着用户数的增长，最终单机的Nginx会成为瓶颈**

Scaling out a database: partition



Can tune database for access characteristics of data stored (common to use different database implementations for different workloads)

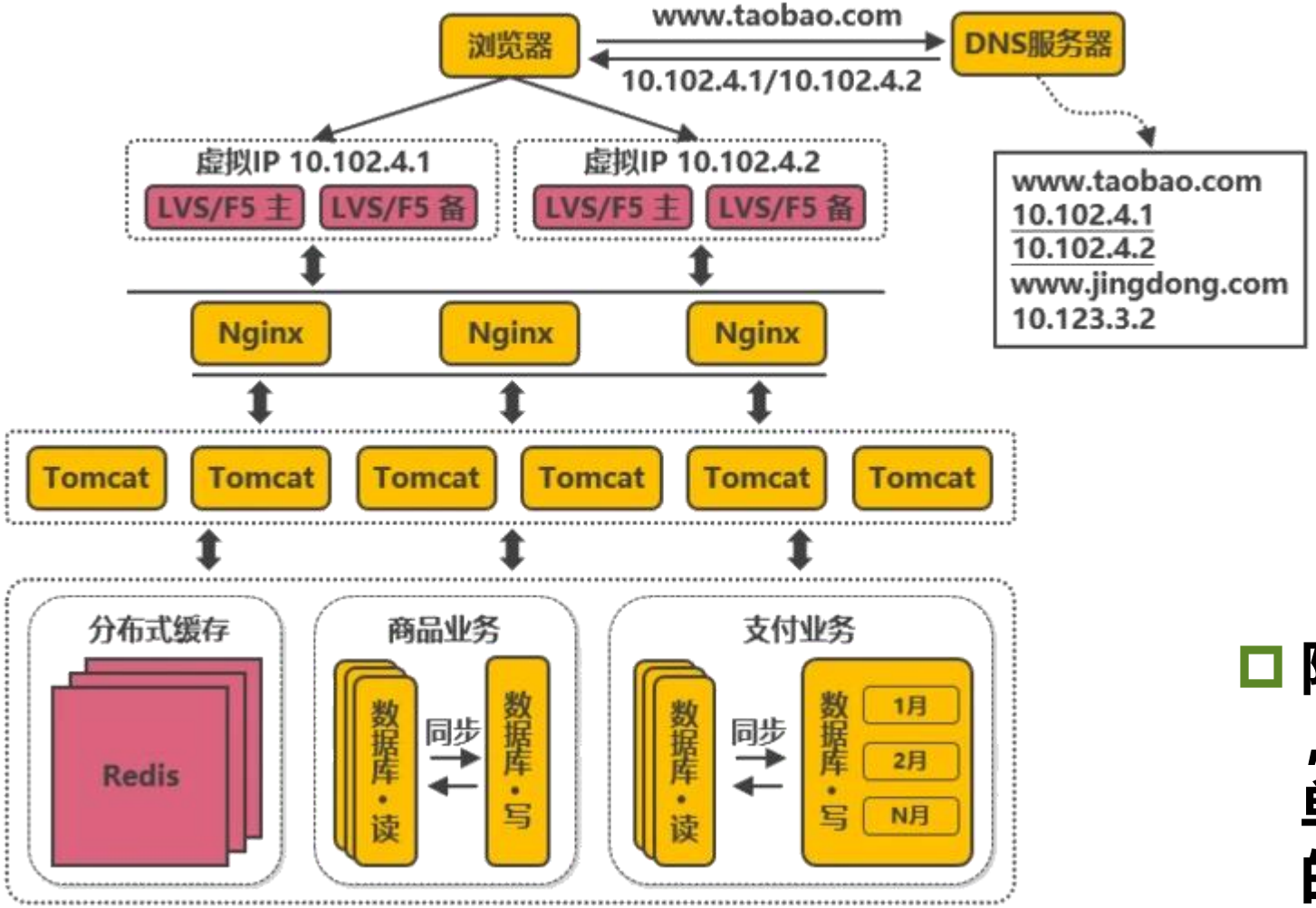
8 第七次演进：使用LVS或F5来使多个Nginx负载均衡



- 由于瓶颈在Nginx，因此无法通过两层的Nginx来实现多个Nginx的负载均衡。图中的LVS和F5是工作在网络第四层的负载均衡解决方案，其中LVS是软件，运行在操作系统内核态，可对TCP请求或更高层级的网络协议进行转发，因此支持的协议更丰富，并且性能也远高于Nginx，可假设单机的LVS可支持几十万个并发的请求转发；F5是一种负载均衡硬件，与LVS提供的能力类似，性能比LVS更高，但价格昂贵。由于LVS是单机版的软件，若LVS所在服务器宕机则会导致整个后端系统都无法访问，因此需要有备用节点。可使用keepalived软件模拟出虚拟IP，然后把虚拟IP绑定到多台LVS服务器上，浏览器访问虚拟IP时，会被路由器重定向到真实的LVS服务器，当主LVS服务器宕机时，keepalived软件会自动更新路由器中的路由表，把虚拟IP重定向到另外一台正常的LVS服务器，从而达到LVS服务器高可用的效果。
- 此处需要注意的是，上图中从Nginx层到Tomcat层这样画并不代表全部Nginx都转发请求到全部的Tomcat，在实际使用时，可能会是几个Nginx下面接一部分的Tomcat，这些Nginx之间通过keepalived实现高可用，其他的Nginx接另外的Tomcat，这样可接入的Tomcat数量就能成倍的增加

- **由于LVS也是单机的，随着并发数增长到几十万时，LVS服务器最终会达到瓶颈，此时用户数达到千万甚至上亿级别，用户分布在不同的地区，与服务器机房距离不同，导致了访问的延迟会明显不同**

9 第八次演进：通过DNS轮询实现机房间的负载均衡

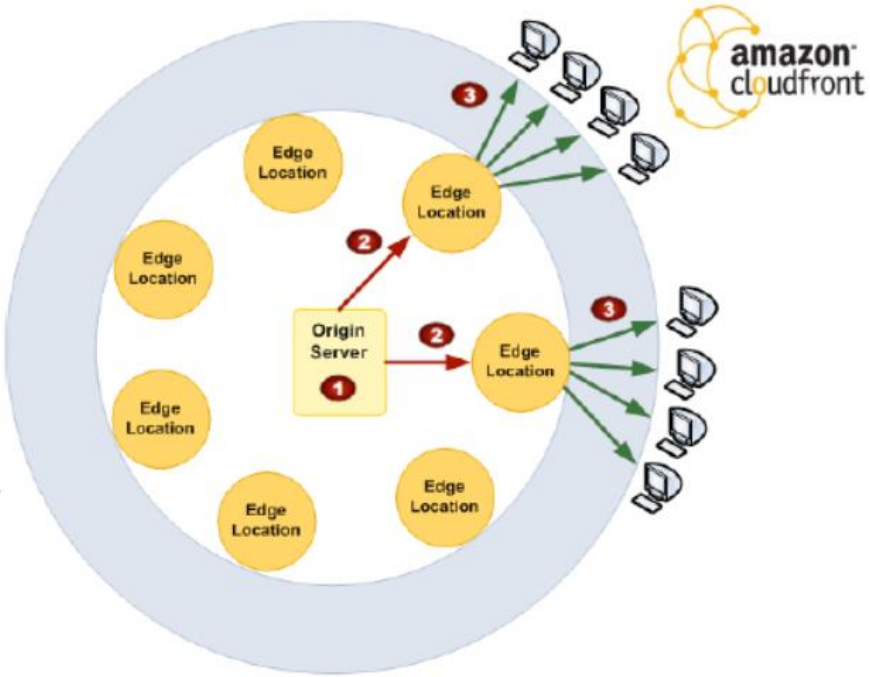


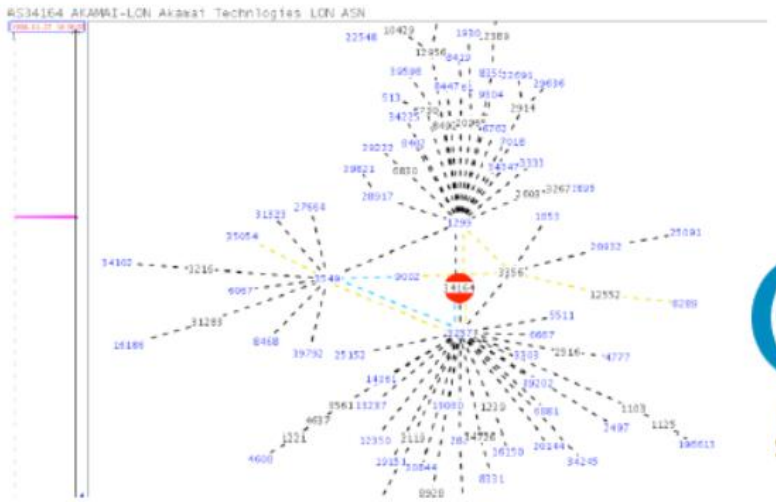
- 在DNS服务器中可配置一个域名对应多个IP地址，每个IP地址对应到不同的机房里的虚拟IP。当用户访问www.taobao.com时，DNS服务器会使用轮询策略或其他策略，来选择某个IP供用户访问。此方式能实现机房间的负载均衡，至此，系统可做到机房级别的水平扩展，千万级到亿级的并发量都可通过增加机房来解决，系统入口处的请求并发量不再是问题。
- **随着数据的丰富程度和业务的发展，检索、分析等需求越来越丰富，单单依靠数据库无法解决如此丰富的需求**

应该有一层 CDN – 业务的大范围(全国或全球)布局

Caching using content distribution networks (CDNs)

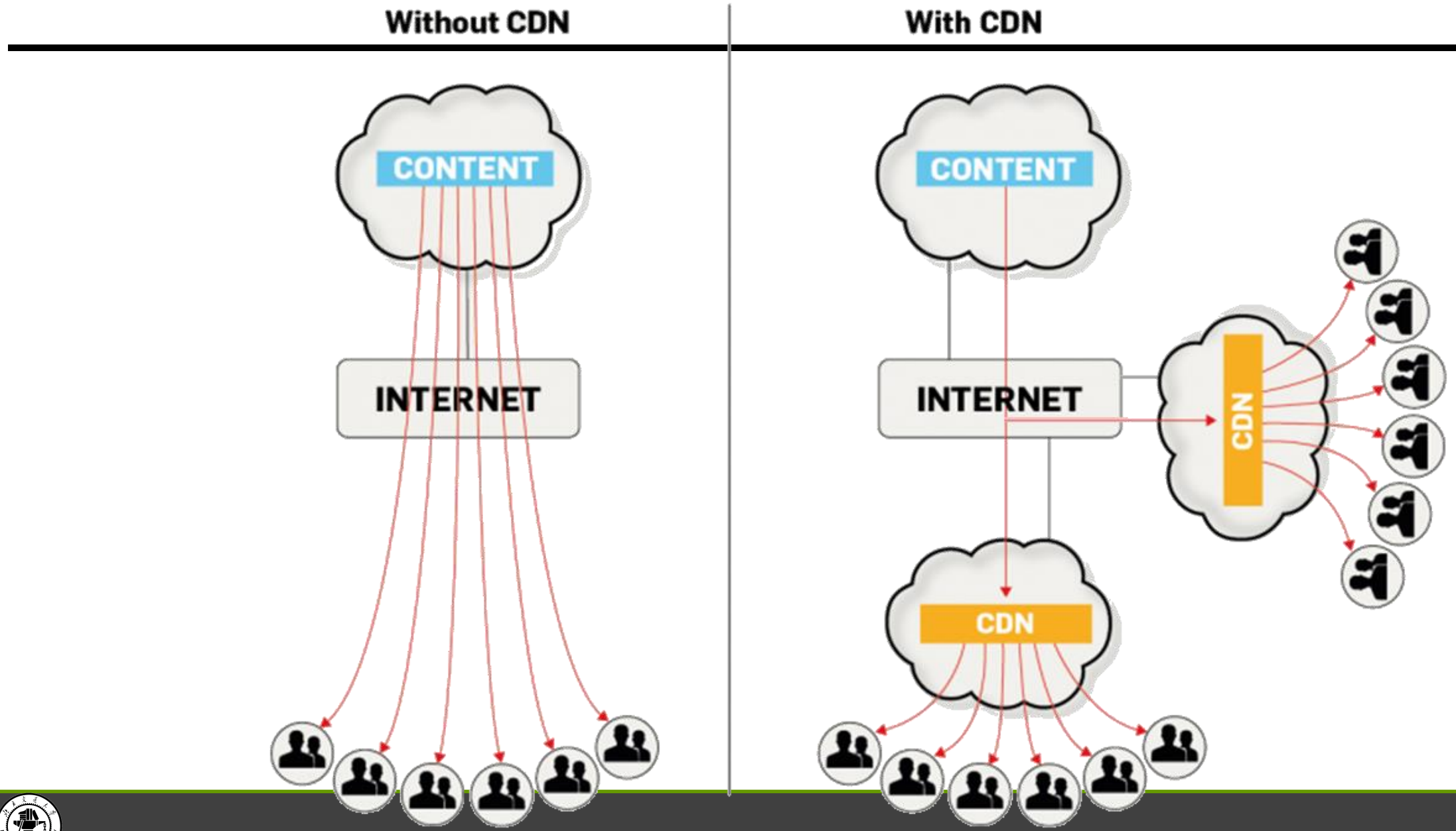
- Serving large media assets can be expensive to serve (high bandwidth costs, tie up web servers)
  - E.g., images, streaming video
- Physical locality is important
  - Higher bandwidth
  - Lower latency

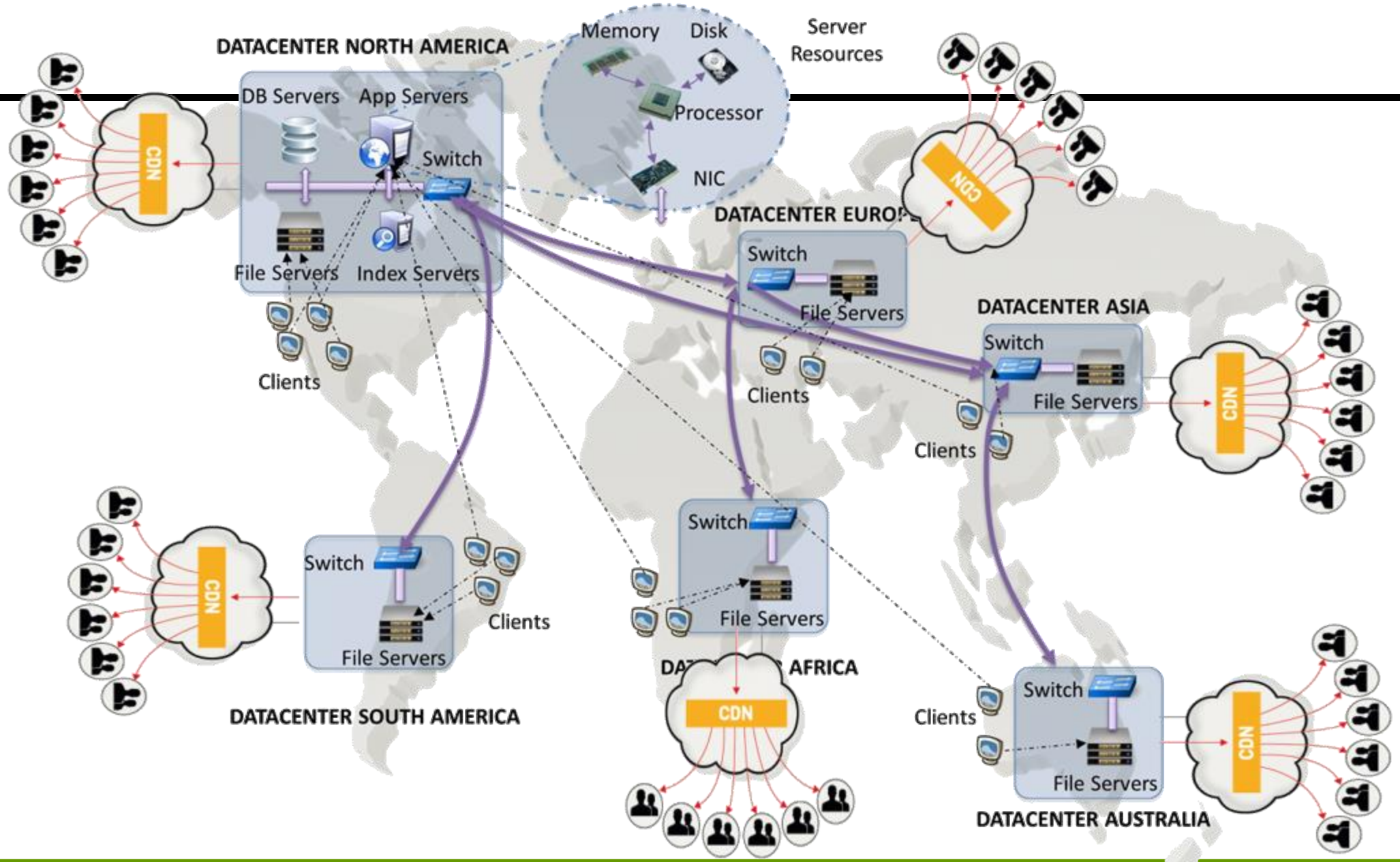




Akamai

London Content Distribution Network
Source: http://www.telco2.net/blog/2008/11/amazon_cloudfront_yet_more_tra.html


Without CDN
CONTENT
INTERNET
With CDN
CONTENT
INTERNET
CDN
CDN


DATACENTER NORTH AMERICA
Memory
Disk
Server Resources
Processor
NIC
DB Servers
App Servers
Switch
CDN
File Servers
Index Servers
Clients
DATACENTER EUROPE
Switch
File Servers
Clients
DATACENTER ASIA
Switch
File Servers
CDN
Clients
Switch
File Servers
Clients
DATACENTER SOUTH AMERICA
Switch
File Servers
CDN
Clients
DATACENTER AFRICA
CDN
Clients
Switch
File Servers
CDN
DATACENTER AUSTRALIA

Chapter 5: Content Delivery Network Companies

The industry has grown significantly in the recent years, as the importance of a CDN has also grown. The Internet is always evolving and so is the need for faster websites and better user experience. Nowadays, there are more and more CDN providers, but here is a quick list of the top 12:

https://www.globaldots.com/content-delivery-network-explained/

1. Akamai

Akamai is known globally to speed up networks and connections. They have a strong infrastructure to speed up sites and make things better for online users. Of course, this company works with big brands and large businesses in general. It's not commonly used by bloggers, but by customers with a huge page traffic and very well established brands.

2. MaxCDN

MaxCDN was founded in 2009 and is based in Los Angeles, California. MaxCDN has fast servers and a big community of users, and also, webmasters from around the World. They have one of the fastest response time and flexible pricing models available. Their CDN is almost always used for WordPress, Joomla, Drupal, OpenCart, PrestaShop, and all the other applications and it utilizes Anycast stateless routing for one-to-nearest content delivery over multiple 10 Gbit/s connections. It's mainly used for websites, blogs and gaming platforms.

3. Incapsula

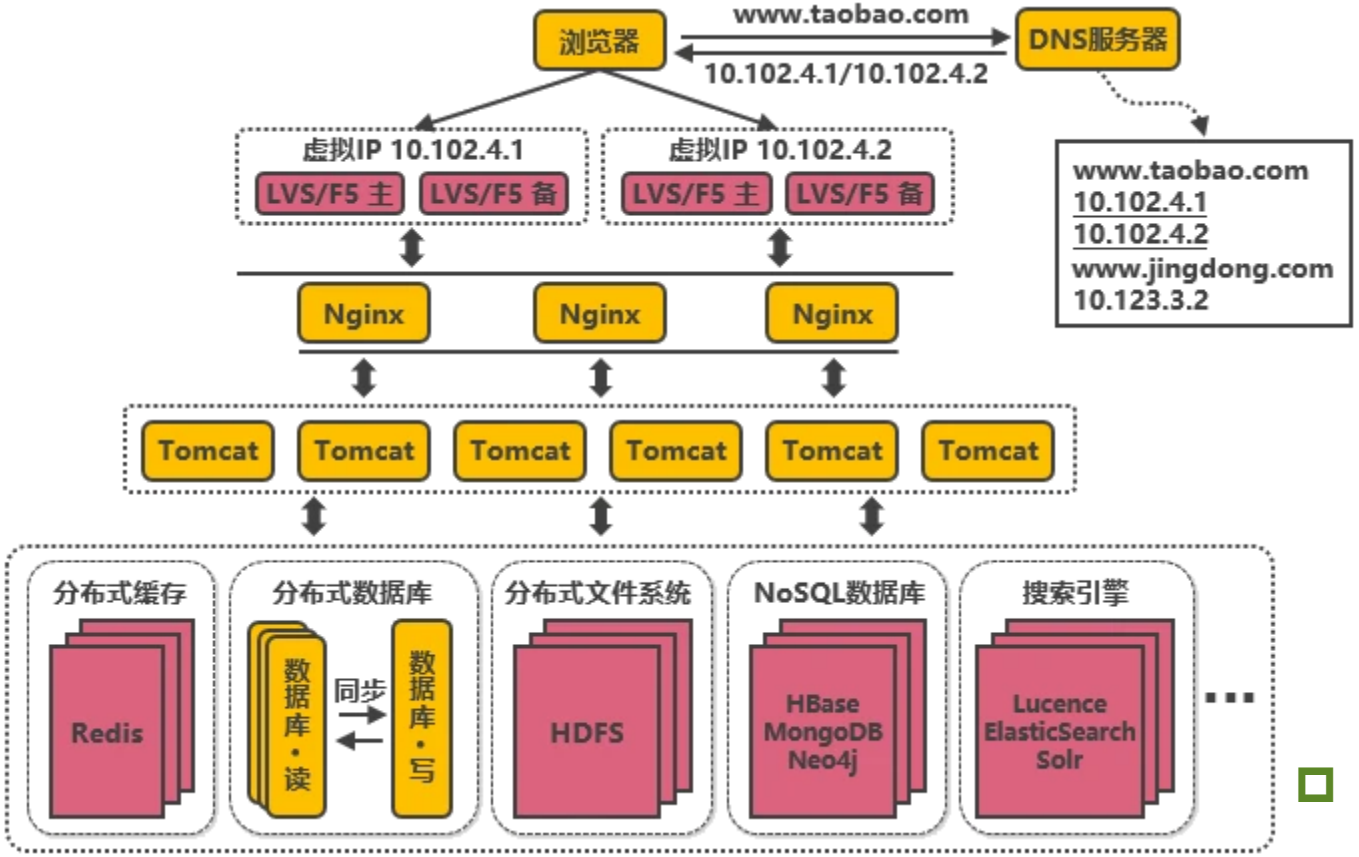
Incapsula Inc. is a Cloud-based application delivery platform. It uses a global content delivery network to provide website security, DDoS protection, load balancing and failover services to clients. It's stationed in Redwood Shores, CA. They offer CDNs paired with Web Security combined with smart balancer technology that handle the traffic between servers. The company works with popular sites like Moz, Wix, SIEMENS and others.

4. Rackspace

Founded back in 1998, Rackspace Inc. is a cloud computing company based in Windcrest, Texas. Rackspace is unique for it's cloud file storage and their "pay as you go" business model. It's a secure environment to host a large file and static websites. It's a well protected infrastructure, with almost two decades of experience, customer trust and brand awareness.

5. Cloudflare

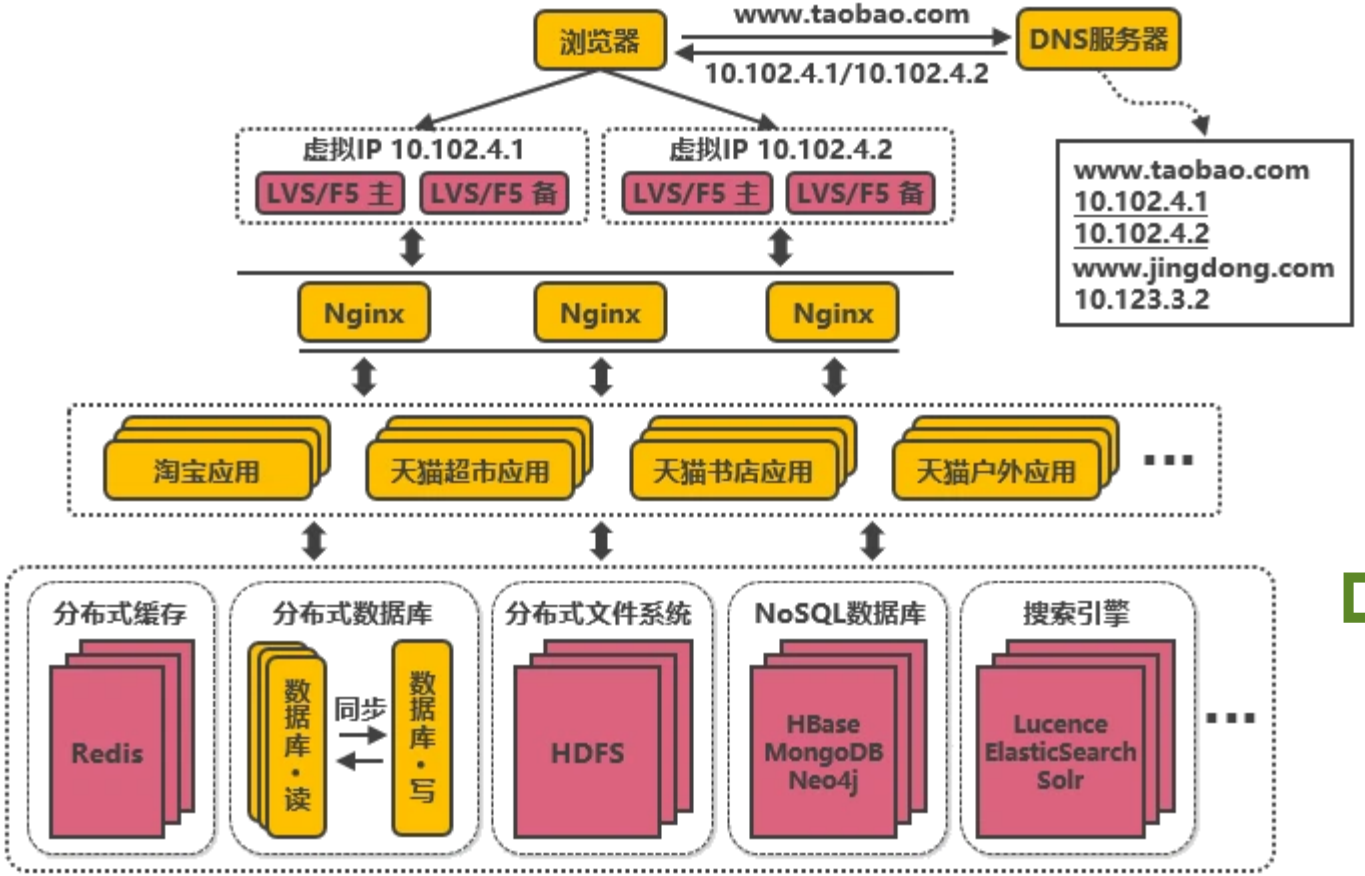
10 第九次演进：引入NoSQL数据库和搜索引擎等技术



- 当数据库中的数据多到一定规模时，数据库就不适用于复杂的查询了，往往只能满足普通查询的场景。对于统计报表场景，在数据量大时不一定能跑出结果，而且在跑复杂查询时会导致其他查询变慢，对于全文检索、可变数据结构等场景，数据库天生不适用。因此需要针对特定的场景，引入合适的解决方案。如对于海量文件存储，可通过分布式文件系统HDFS解决，对于key value类型的数据，可通过HBase和Redis等方案解决，对于全文检索场景，可通过搜索引擎如ElasticSearch解决，对于多维分析场景，可通过Kylin或Druid等方案解决。
- 当然，引入更多组件同时会提高系统的复杂度，不同的组件保存的数据需要同步，需要考虑一致性的问题，需要有更多的运维手段来管理这些组件等。

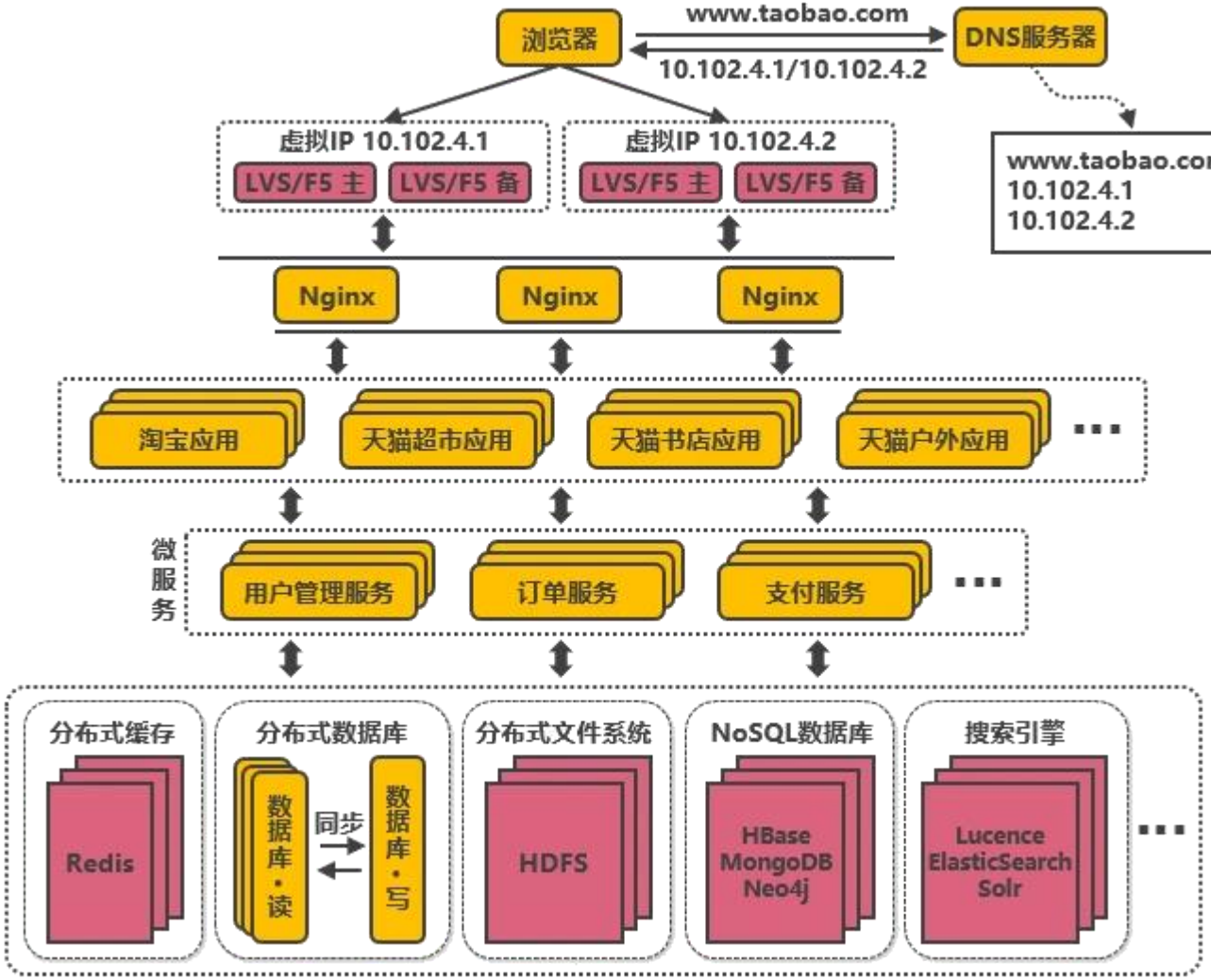
- **引入更多组件解决了丰富的需求，业务维度能够极大扩充，随之而来的是一个应用中包含了太多的业务代码，业务的升级迭代变得困难**

11 第十次演进：大应用拆分为小应用



- 按照业务板块来划分应用代码，使单个应用的职责更清晰，相互之间可以做到独立升级迭代。这时候应用之间可能会涉及到一些公共配置，可以通过分布式配置中心Zookeeper来解决。
- **不同应用之间存在共用的模块，由应用单独管理会导致相同代码存在多份，导致公共功能升级时全部应用代码都要跟着升级**

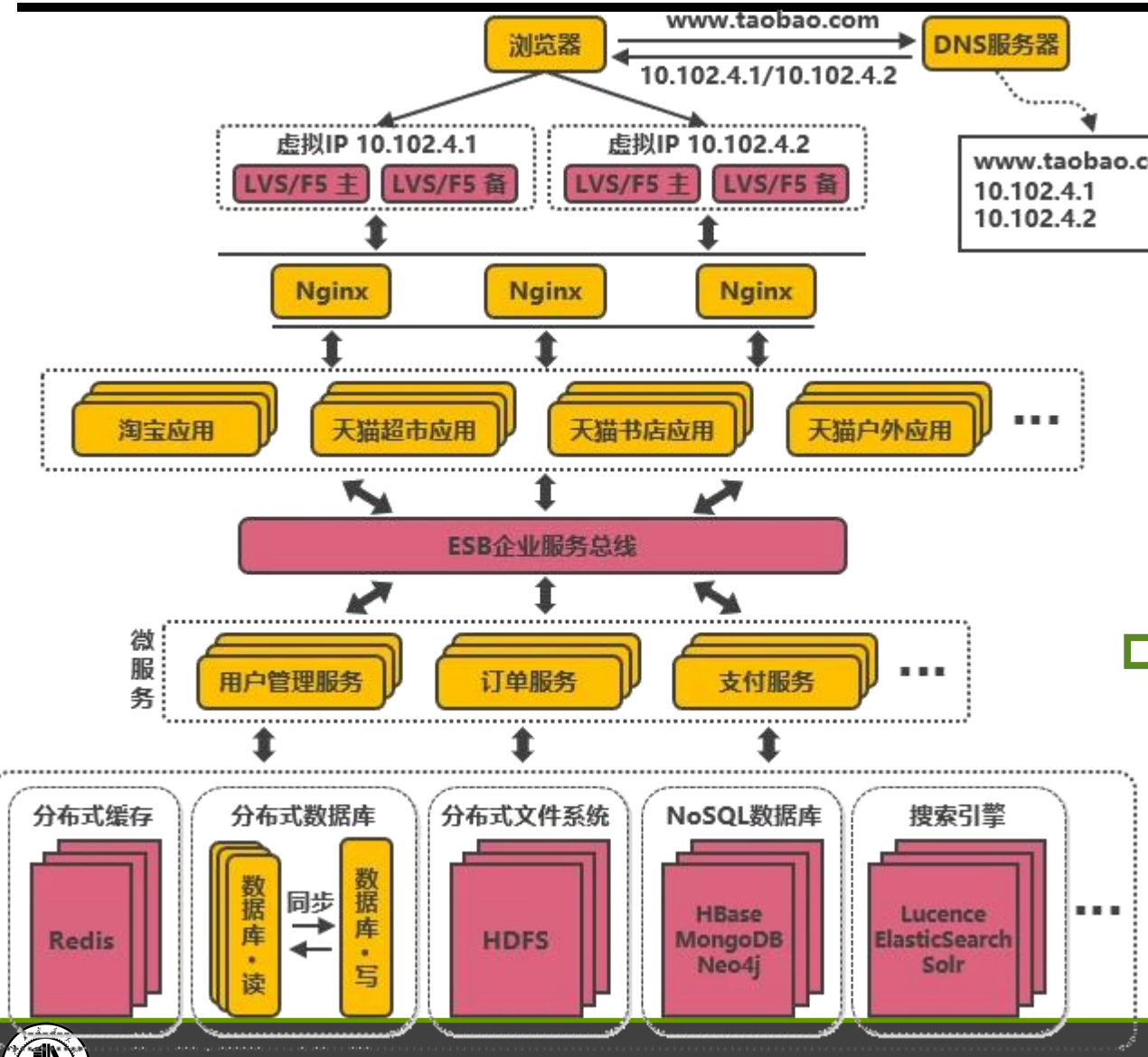
12 第十一次演进：复用的功能抽离成微服务



- 如用户管理、订单、支付、鉴权等功能在多个应用中都存在，那么可以把这些功能的代码单独抽取出来形成一个单独的服务来管理，这样的服务就是所谓的微服务，应用和服务之间通过HTTP、TCP或RPC请求等多种方式来访问公共服务，每个单独的服务都可以由单独的团队来管理。此外，可以通过Dubbo、SpringCloud等框架实现服务治理、限流、熔断、降级等功能，提高服务的稳定性和可用性。

- **不同服务的接口访问方式不同，应用代码需要适配多种访问方式才能使用服务，此外，应用访问服务，服务之间也可能相互访问，调用链将会变得非常复杂，逻辑变得混乱**

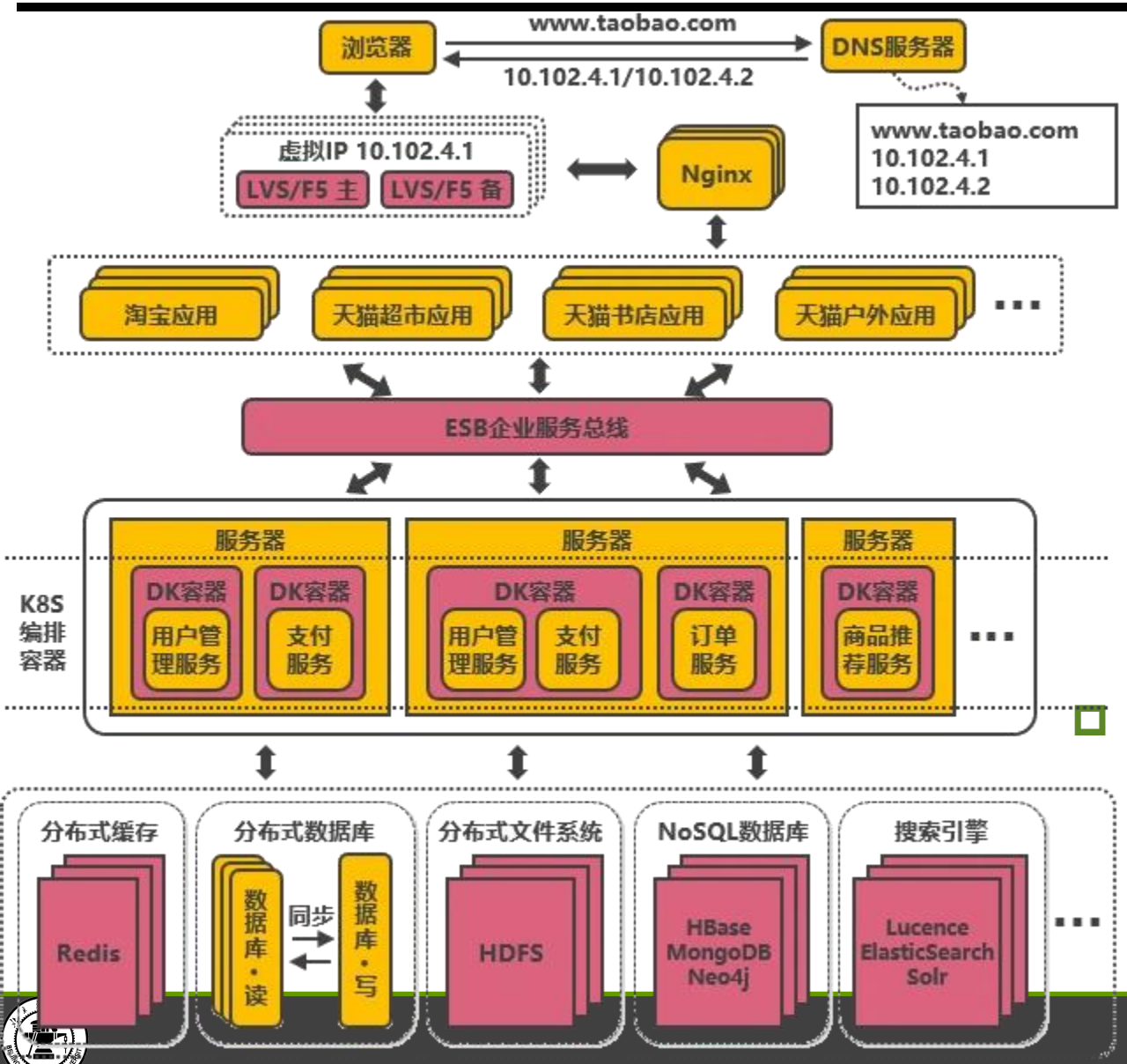
13 第十二次演进：引入企业服务总线ESB屏蔽服务接口的访问差异



- 通过ESB统一进行访问协议转换，应用统一通过ESB来访问后端服务，服务与服务之间也通过ESB来相互调用，以此降低系统的耦合程度。这种单个应用拆分为多个应用，公共服务单独抽取出来来管理，并使用企业消息总线来解除服务之间耦合问题的架构，就是所谓的SOA（面向服务）架构，这种架构与微服务架构容易混淆，因为表现形式十分相似。个人理解，微服务架构更多是指把系统里的公共服务抽取出来单独运维管理的思想，而SOA架构则是指一种拆分服务并使服务接口访问变得统一的架构思想，SOA架构中包含了微服务的思想。

- **业务不断发展，应用和服务都会不断变多，应用和服务的部署变得复杂，同一台服务器上部署多个服务还要解决运行环境冲突的问题，此外，对于如大促这类需要动态扩缩容的场景，需要水平扩展服务的性能，就需要在新增的服务上准备运行环境，部署服务等，运维将变得十分困难**

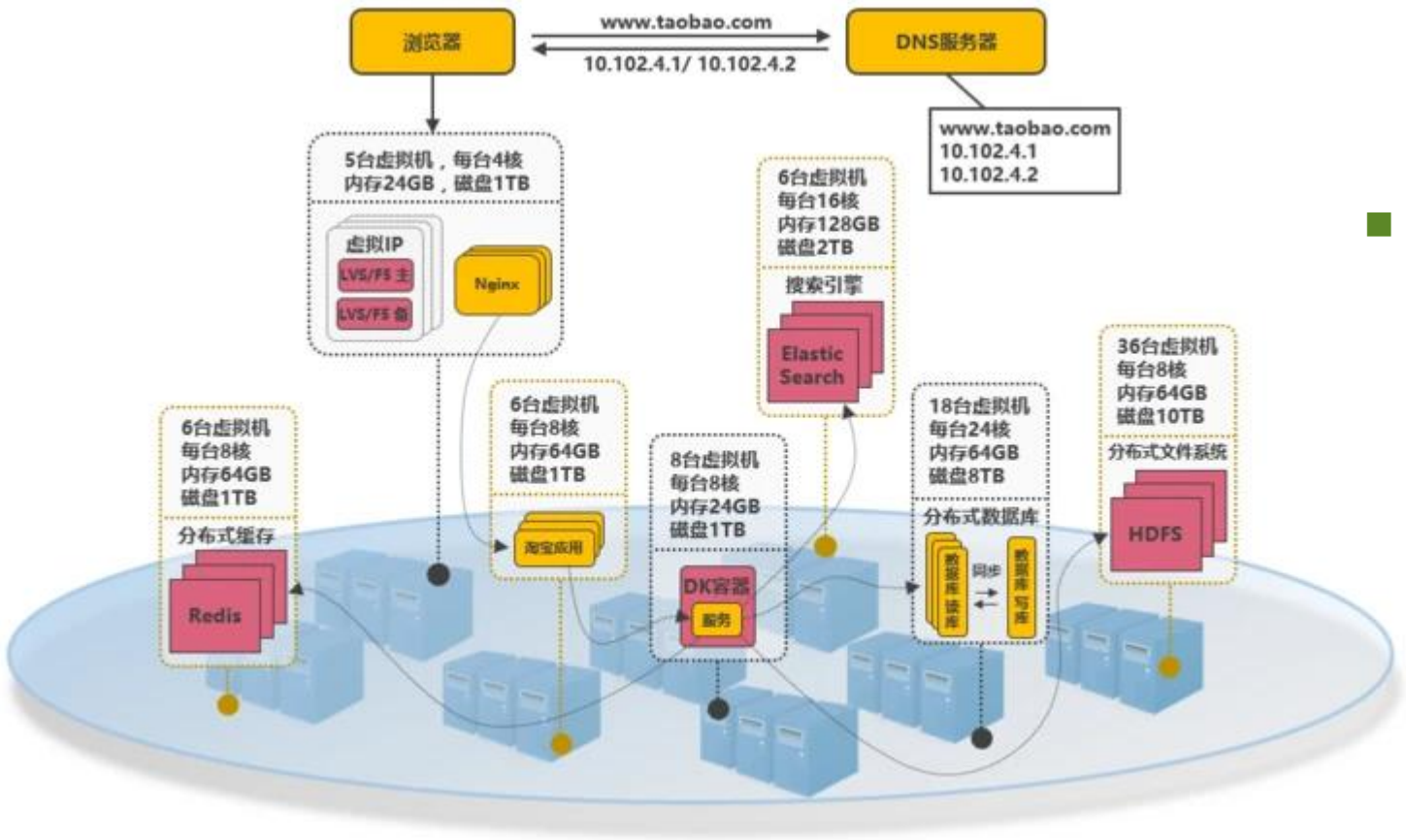
14 第十三次演进：引入容器化技术实现运行环境隔离与动态服务管理



- 目前最流行的容器化技术是Docker，最流行的容器管理服务是Kubernetes(K8S)，应用/服务可以打包为Docker镜像，通过K8S来动态分发和部署镜像。Docker镜像可理解为一个能运行你的应用/服务的最小的操作系统，里面放着应用/服务的运行代码，运行环境根据实际的需要设置好。把整个“操作系统”打包为一个镜像后，就可以分发到需要部署相关服务的机器上，直接启动Docker镜像就可以把服务起起来，使服务的部署和运维变得简单。
- 在大促的之前，可以在现有的机器集群上划分出服务器来启动Docker镜像，增强服务的性能，大促过后就可以关闭镜像，对机器上的其他服务不造成影响（在3.14节之前，服务运行在新增机器上需要修改系统配置来适配服务，这会导致机器上其他服务需要的运行环境被破坏）。

- **使用容器化技术后服务动态扩缩容问题得以解决，但是机器还是需要公司自身来管理，在非大促的时候，还是需要闲置着大量的机器资源来应对大促，机器自身成本和运维成本都极高，资源利用率低**

15 第十四次演进：以云平台承载系统

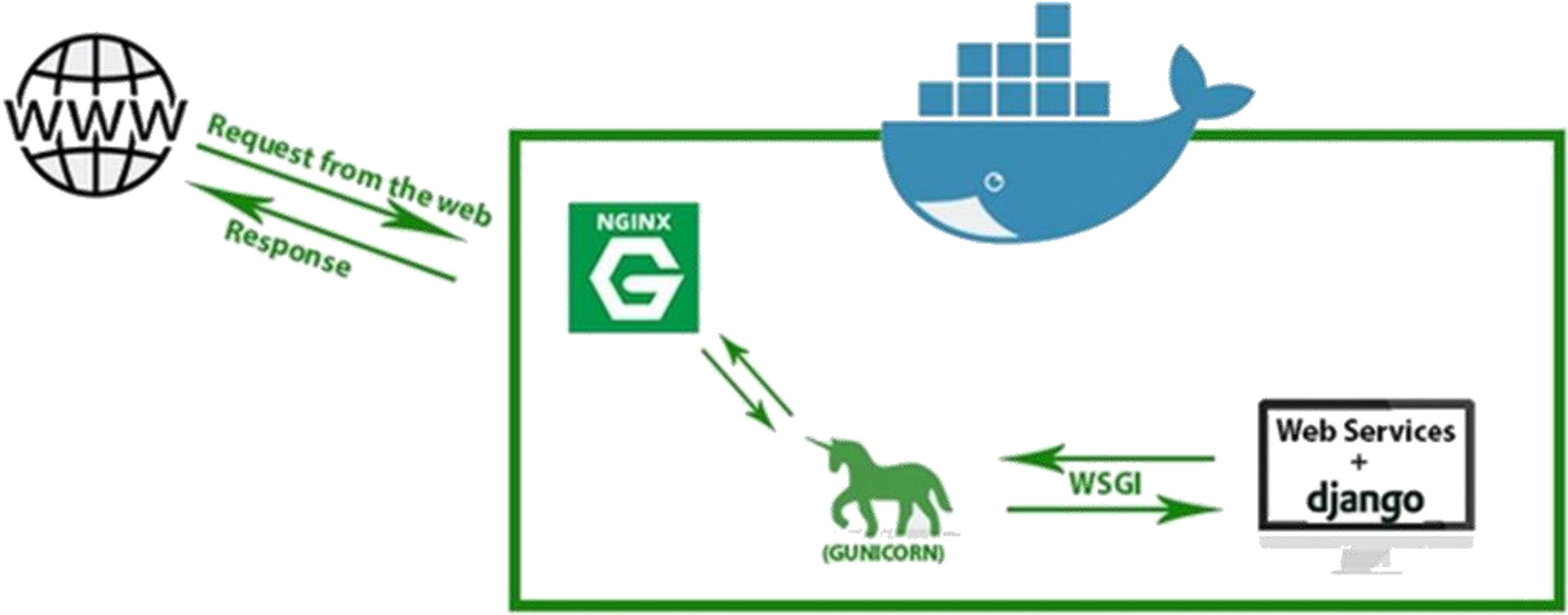


- 系统可部署到公有云上，利用公有云的海量机器资源，解决动态硬件资源的问题，在大促的时间段里，在云平台中临时申请更多的资源，结合Docker和K8S来快速部署服务，在大促结束后释放资源，真正做到按需付费，资源利用率大大提高，同时大大降低了运维成本。
- 所谓的云平台，就是把海量机器资源，通过统一的资源管理，抽象为一个资源整体，在之上可按需动态申请硬件资源（如CPU、内存、网络等），并且之上提供通用的操作系统，提供常用的技术组件（如Hadoop技术栈，MPP数据库等）供用户使用，甚至提供开发好的应用，用户不需要关系应用内部使用了什么技术，就能够解决需求（如音视频转码服务、邮件服务、个人博客等）。在云平台中会涉及如下几个概念：
  - IaaS：基础设施即服务。对应于上面所说的机器资源统一为资源整体，可动态申请硬件资源的层面；
  - PaaS：平台即服务。对应于上面所说的提供常用的技术组件方便系统的开发和维护；
  - SaaS：软件即服务。对应于上面所说的提供开发好的应用或服务，按功能或性能要求付费。

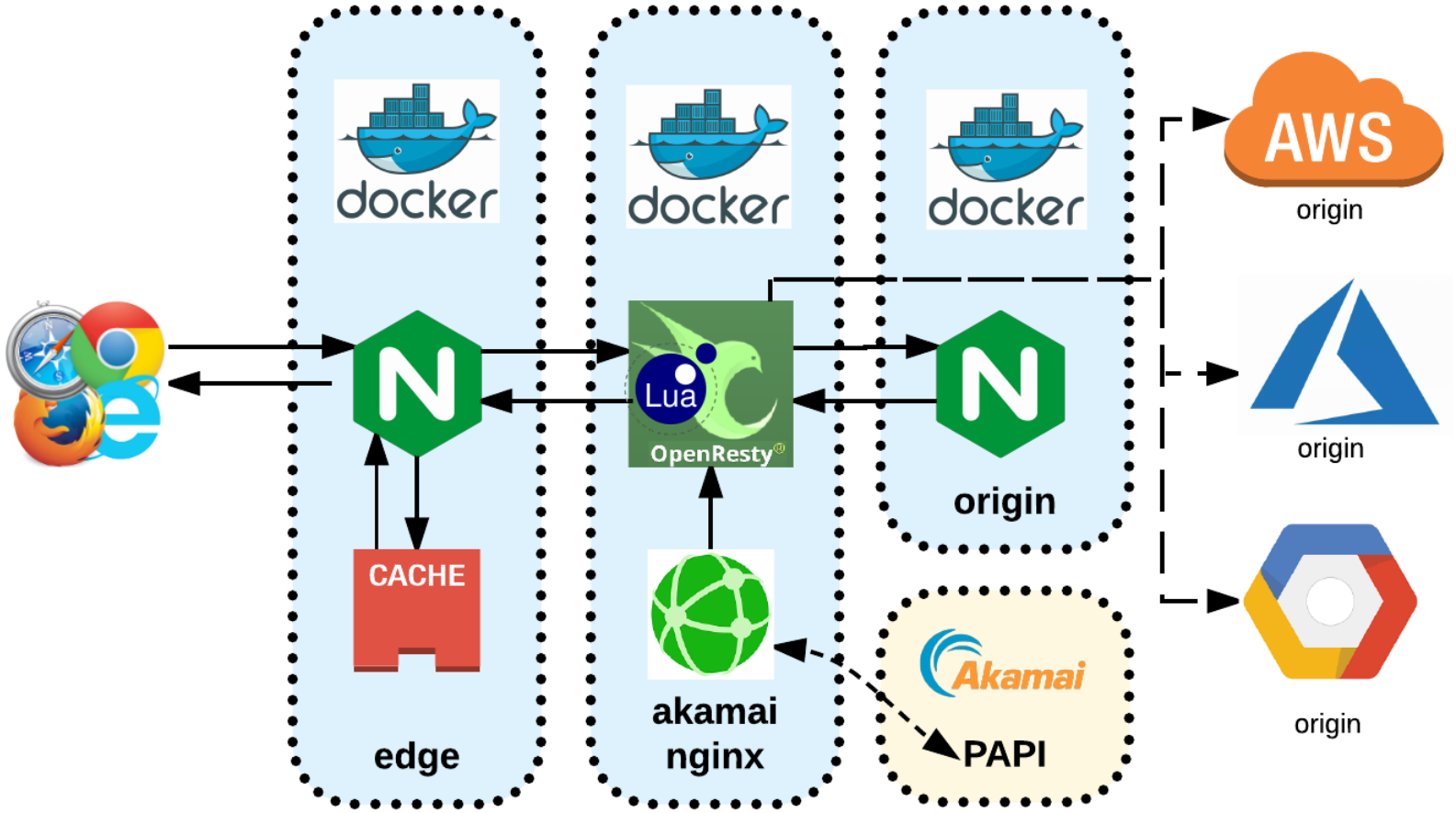
很佩服：阿里云已成为全球知名的云服务提供商！

- Many Cloud platforms

现在很流行Cloud 上部署 – 但，架构还是类似

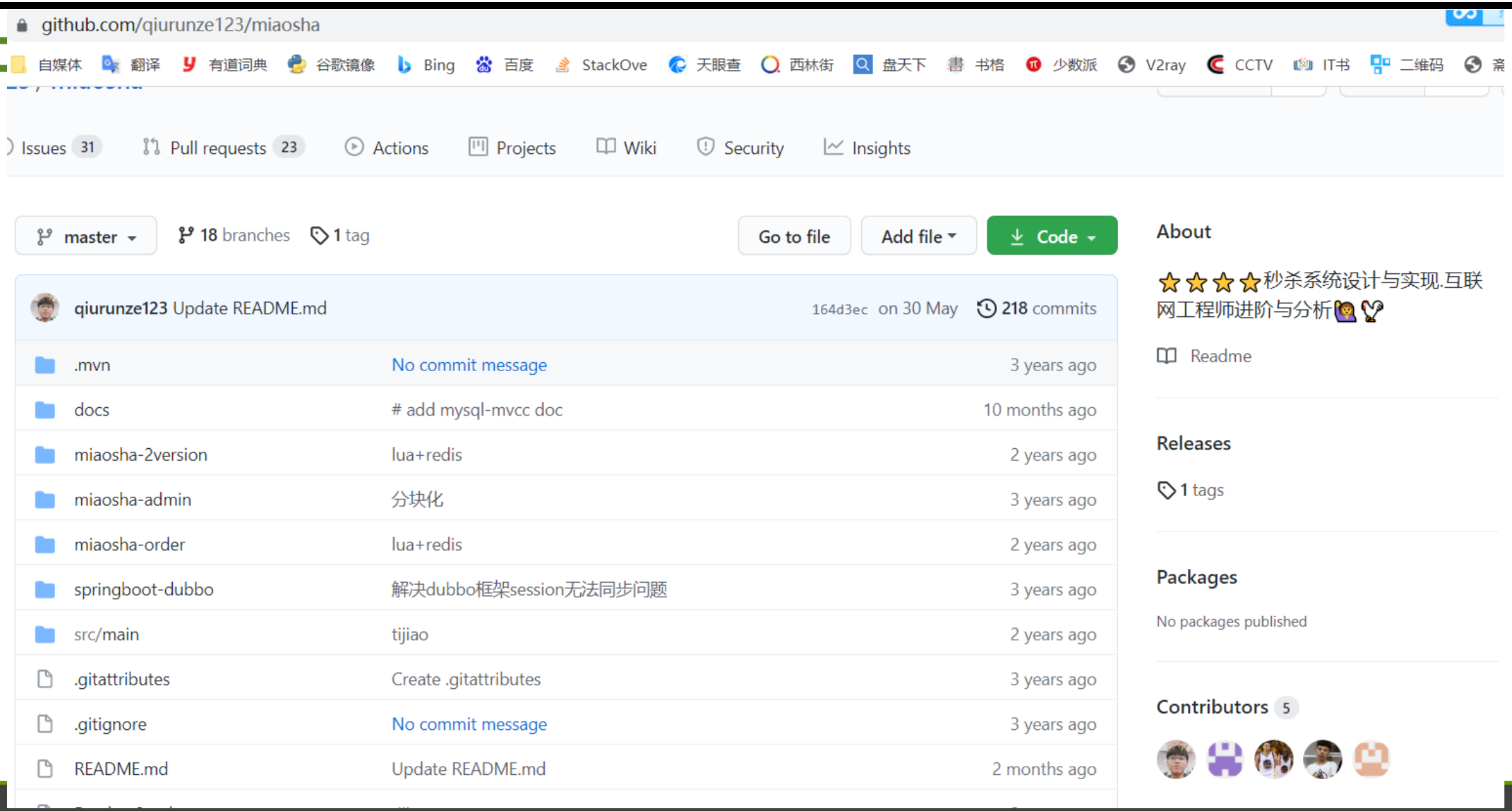


akamai-nginx request flow



Akamai PAPI is called once, to generate equivalent LUA config

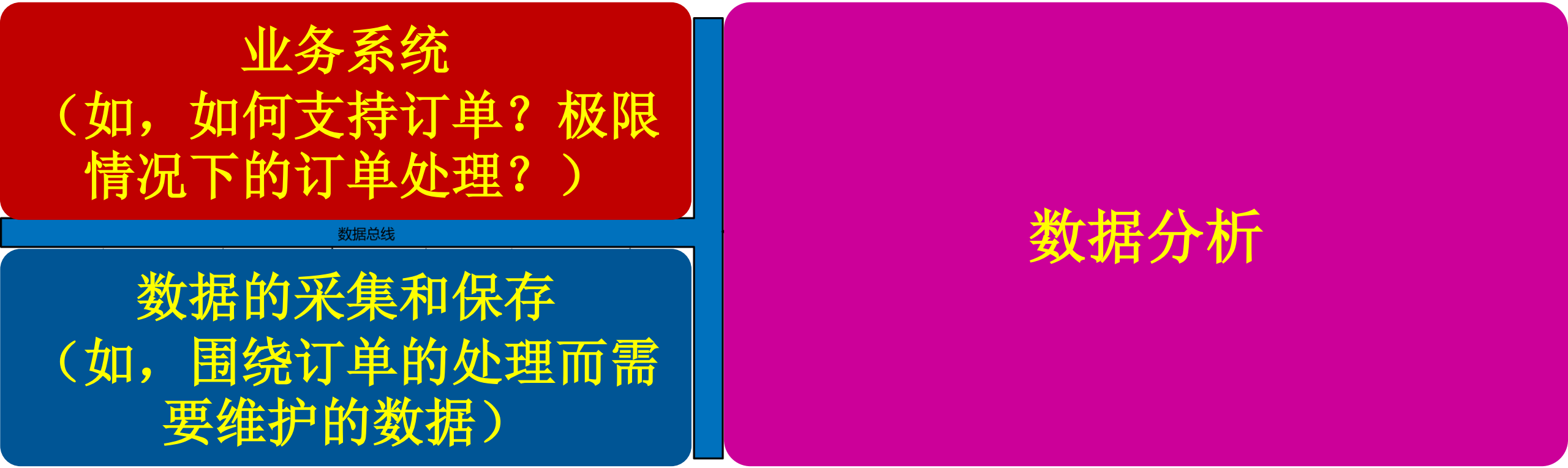
https://github.com/qiurunze123/miaosha



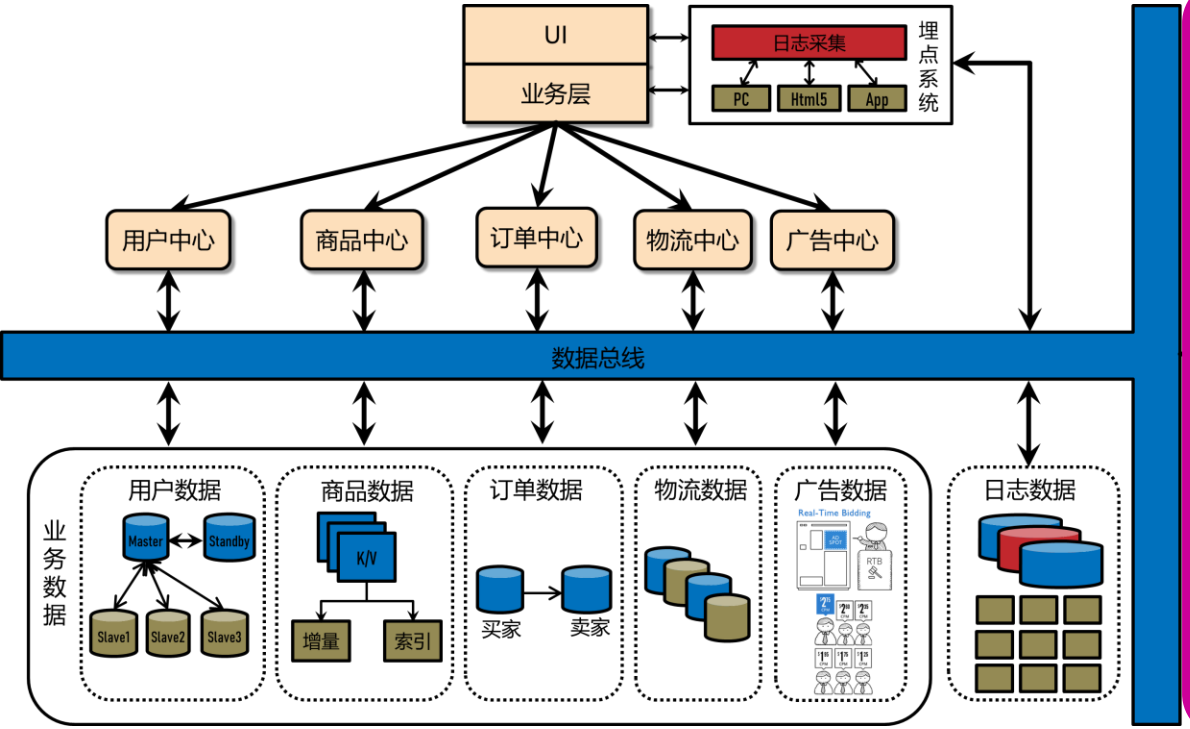
Summary

Now, a quite complex system

- But still the sketch is followed

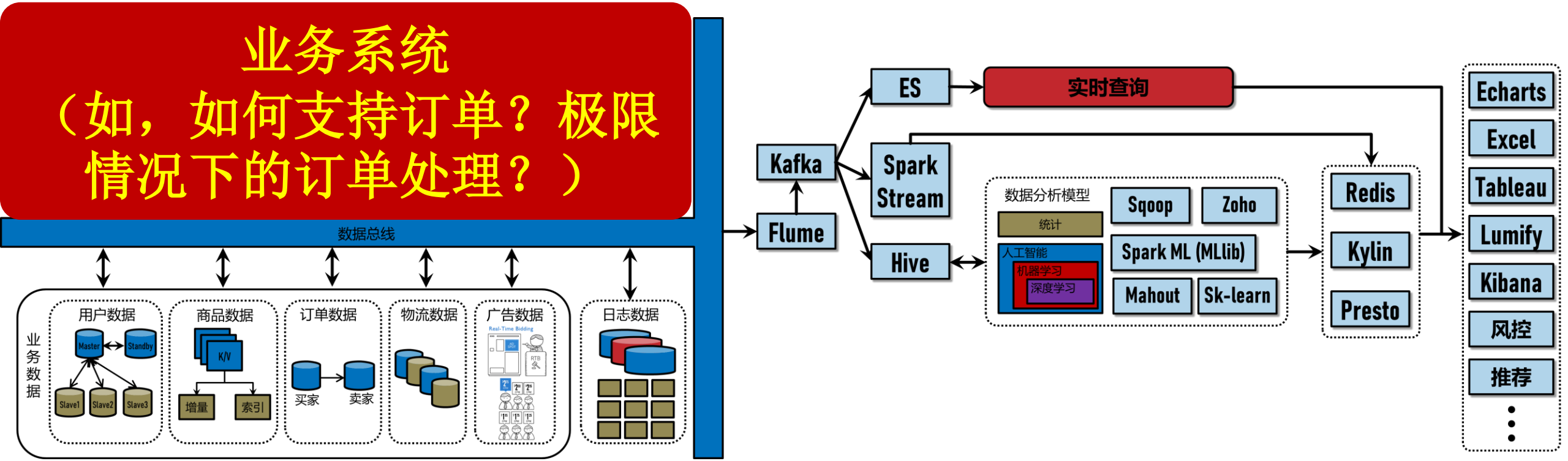


业务系统(Business System) 是立身之本~



数据分析

数据分析是提升效率和满意度的保障 – (例如精准营销)

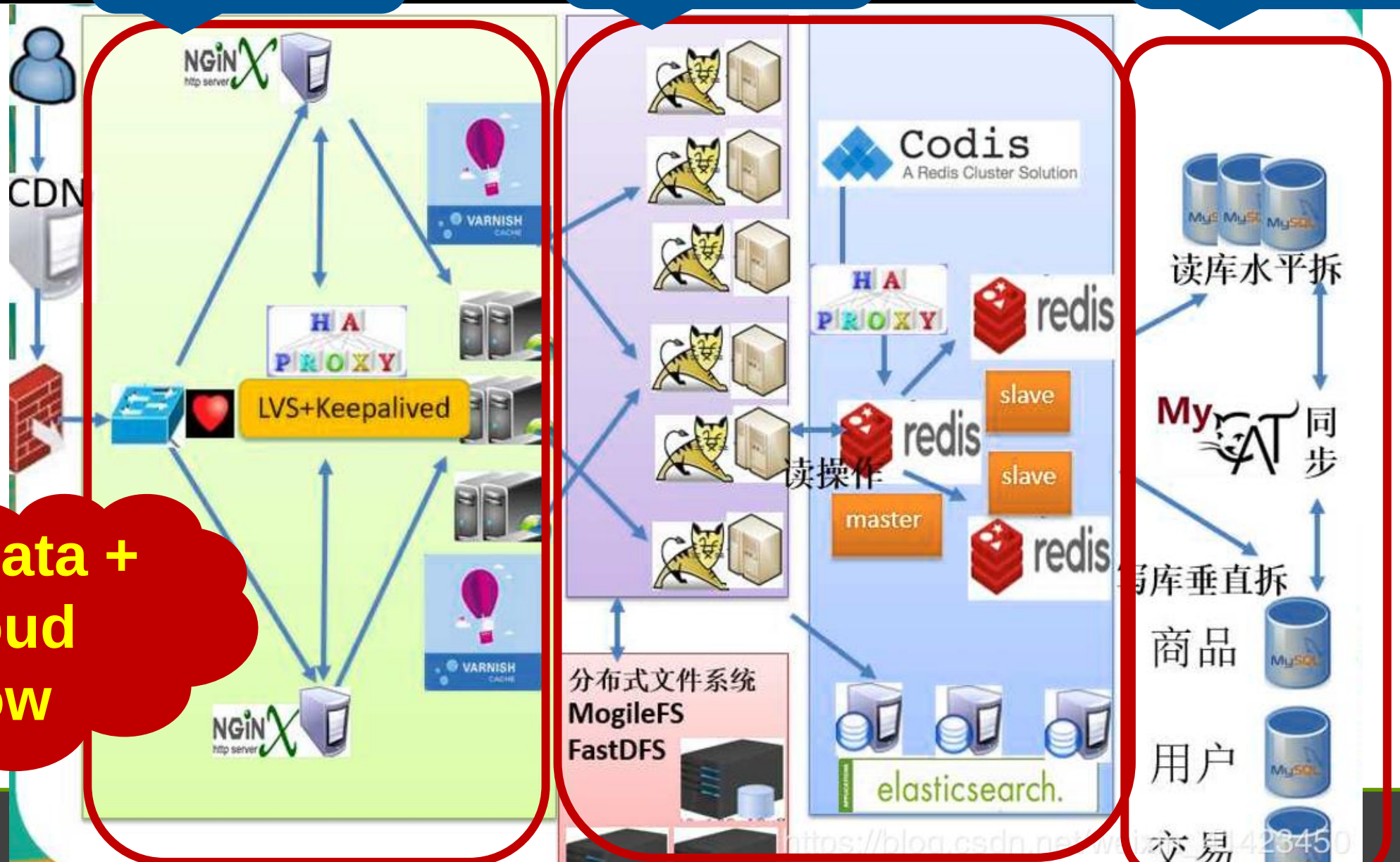


架构分解就是为了满足高并发和大数据，具体原则：

Load Balancer

Distributed Caches + Files

Distributed DBs

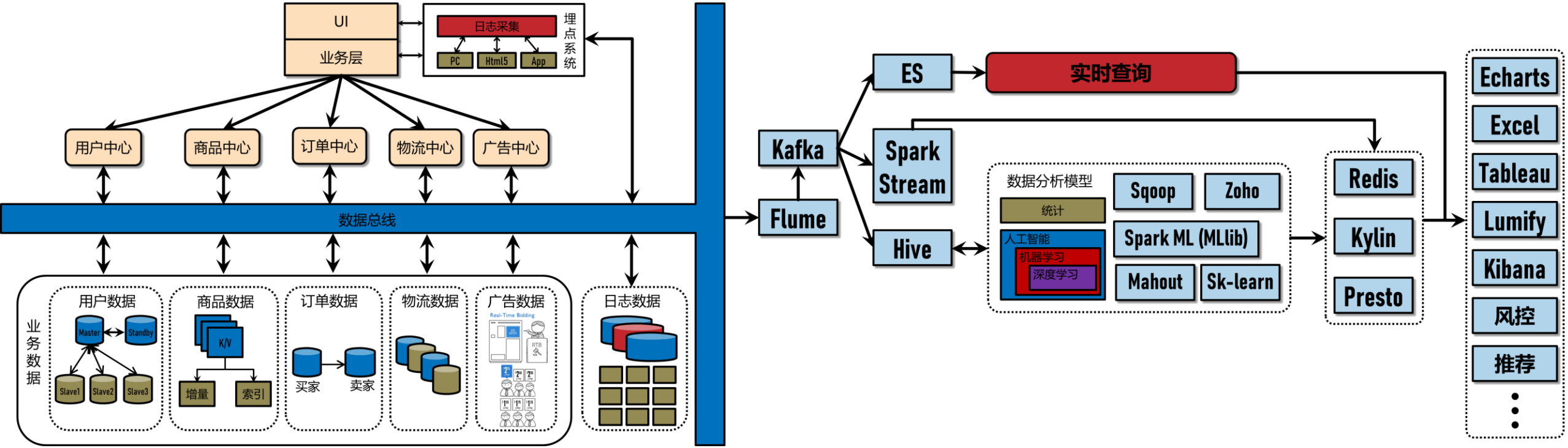


Big Data + Cloud now

Chapter 6: HUGE Concurrency architecture for “秒杀”

- **Architecture for “秒杀”**
  - Just “Divide and Conquer” – Data level, Module level
  - Big Data + Cloud now!
- **So-called Software architect [软件架构师]**

Big Data + Cloud now



https://zhuanlan.zhihu.com/p/495147699

- 2021年“双11”，被鲁肃称为阿里集团100%上云，也就是阿里巴巴的大大小小的业务，已经完全跑在了阿里云上

发文
评论
微博
空间
微信

全球首家！阿里所有业务100%上云

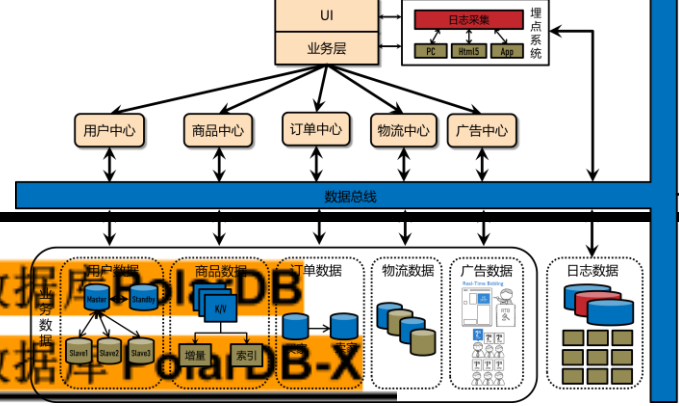
OFweek物联网 2021-11-11 18:06 发文

11月11日，阿里巴巴首席技术官程立表示，阿里巴巴业务已全部跑在阿里云上，体验如丝般顺滑。经过历年双11的“大考”，阿里技术实现了多级跳跃。

在此之前，全世界还没有一家大型科技公司把所有业务都放在自家公共云上，这意味着阿里云有能力应对高难度复杂环境下的技术挑战。

https://mp.ofweek.com/iot/a056714283287

阿里系的数据管理



云原生关系型数据库PolarDB

云原生分布式数据库PolarDB-X

查看全部产品 >

热门产品
弹性计算
存储
数据库
安全
大数据
人工智能
网络与CDN
视频服务
容器与中间件
开发与运维
物联网IoT
混合云
企业服务与云通信

搜索云产品

关系型数据库

云原生关系型数据库 PolarDB HOT
云原生分布式数据库 PolarDB-X
云数据库 RDS MySQL 版 HOT
云数据库 RDS SQL Server 版
云数据库 RDS PostgreSQL 版
云数据库 MariaDB 版
云数据库 OceanBase

数据库专属集群

云数据库专属集群 MyBase

NoSQL数据库

云原生多模数据库 Lindorm
云数据库 Redis 版
云数据库 MongoDB 版
云数据库 HBase 版
时序数据库 TSDB 版
图数据库 GDB
表格存储 Tablestore

数据仓库

云原生数据仓库 AnalyticDB MySQL 版
云原生数据仓库 AnalyticDB PostgreS...
云数据库 ClickHouse
云原生数据湖分析 DLA

数据库生态工具

数据传输服务 DTS
数据管理 DMS
数据库备份 DBS
数据库自治服务 DAS
数据库专家服务

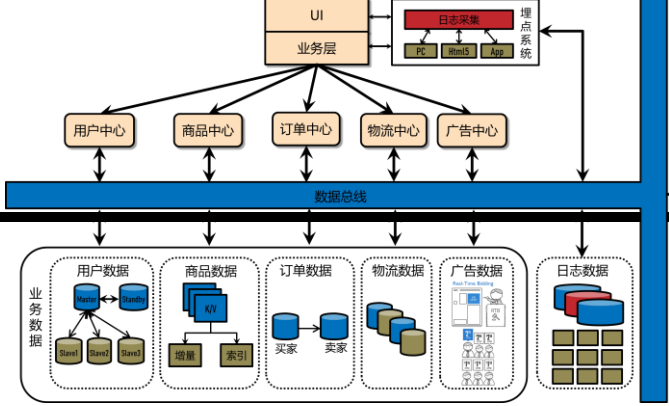
相关解决方案

数据传输
数据库安全
SaaS 上云数据库
企业级分布式数据库
Oracle 数据库一键上云
MySQL 数据库上云选型
云上 OLTP+OLAP 数据库服务
数据湖
数据库灾备
游戏行业多场景云数据库

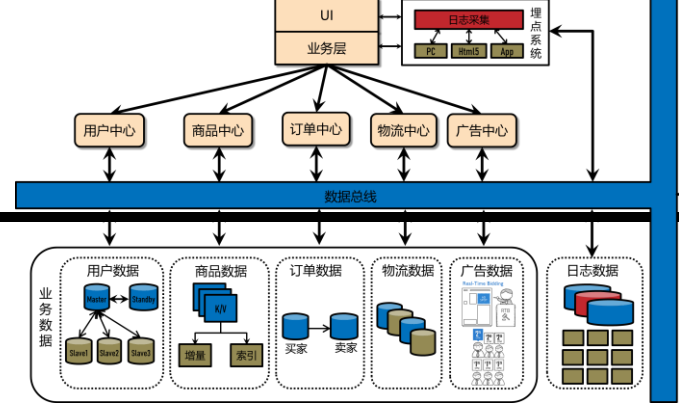
https://developer.aliyun.com/article/790008?spm=a2c6h.13813017.content3.1.297b18b7XA7QK8

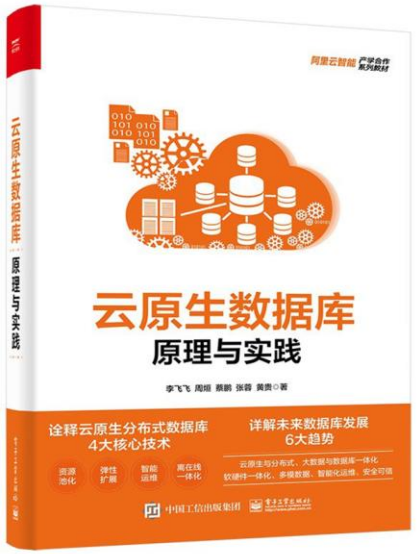
201

OceanBase 大事记

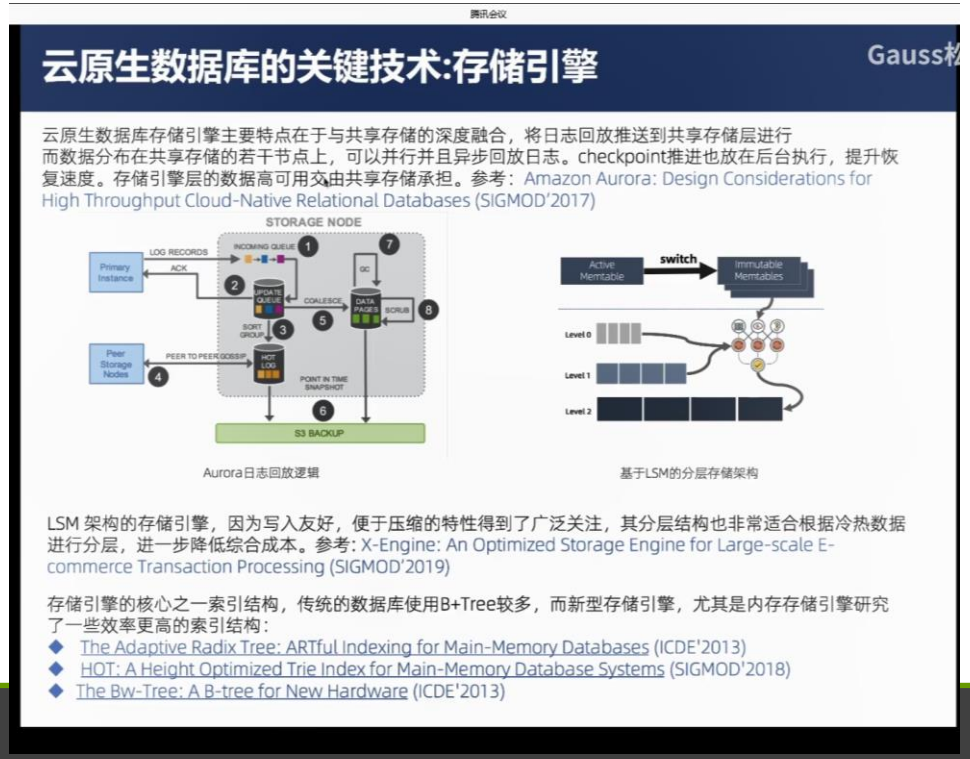


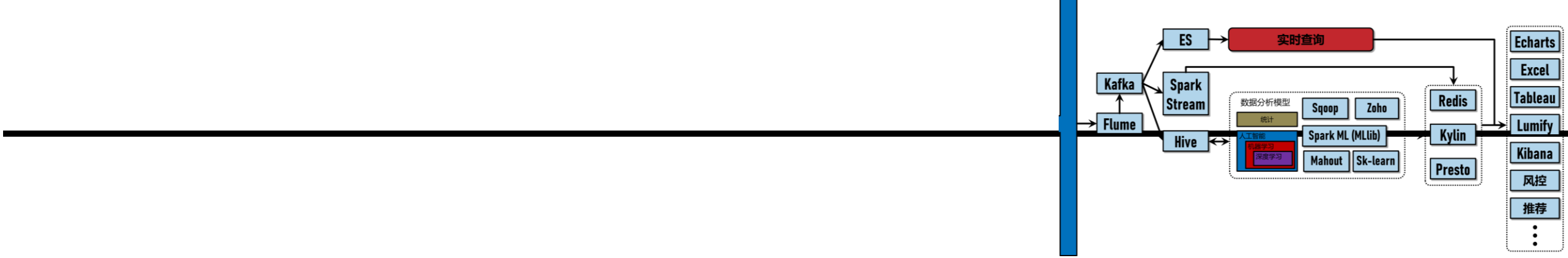
- 2010 创始人阳振坤加入阿里巴巴，OceanBase 正式立项；
- 2011 OceanBase 0.1 版本发布，应用于淘宝收藏夹；
- 2014 OceanBase 0.5 版本发布，替代 Oracle 在支付宝交易系统上线，负担“双十一”10% 流量；
- 2015 网商银行成立，OceanBase 成为全球首个应用在金融核心业务系统的分布式关系数据库；
- 2016 OceanBase 1.0 版本在支付宝账务系统上线，支撑 12 万笔/秒支付峰值；
- 2017 支付宝首次把账务库在内的所有核心数据链路搬到 OceanBase 上，创造 4200 万次 / 秒数据库处理峰值纪录。同年，OceanBase 1.x 版本在多家商业银行上线；
- 2018 OceanBase 2.0 版本正式发布，降低金融业务向分布式架构转型的技术风险；
- 2019 OceanBase 获得 TPC-C 基准测试排名榜首。

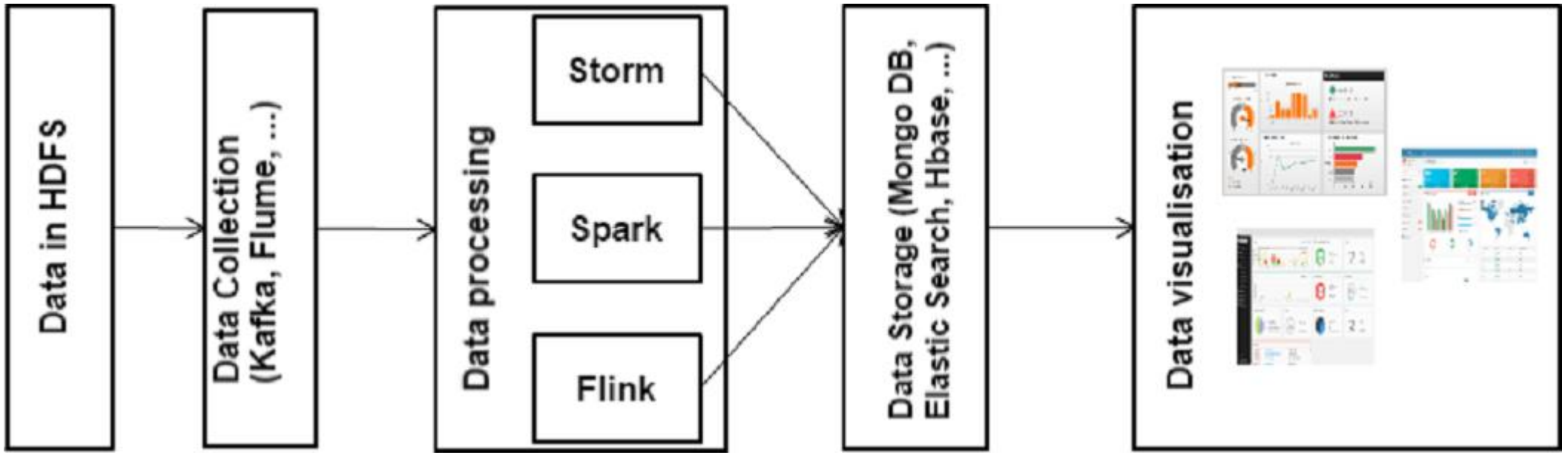




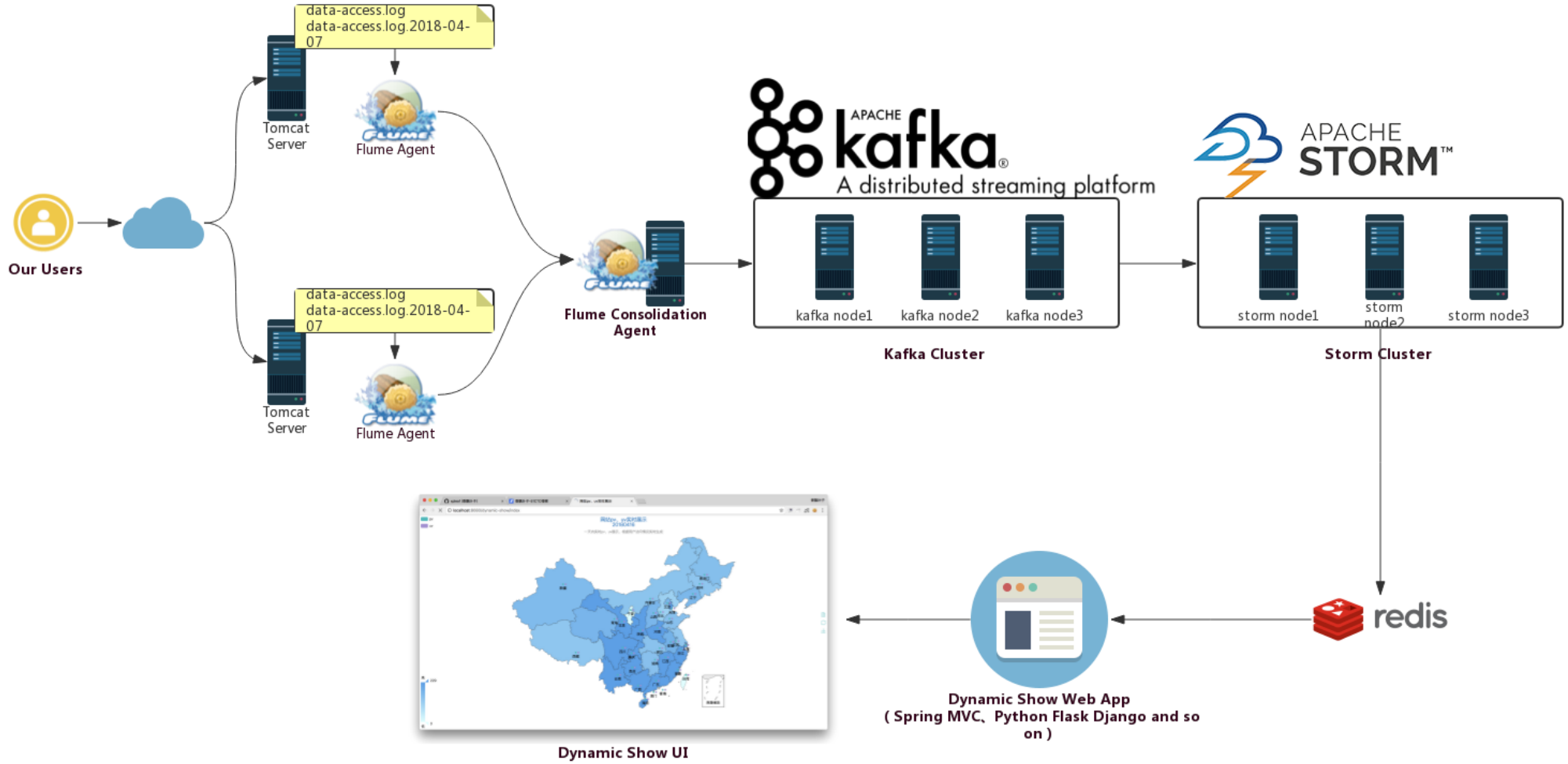
- **云原生数据库：原理与实践（全彩）（博文视点出品）**
- 李飞飞，周烜，蔡鹏，张蓉，黄贵 … 著
- **2022**
- 电子工业出版社

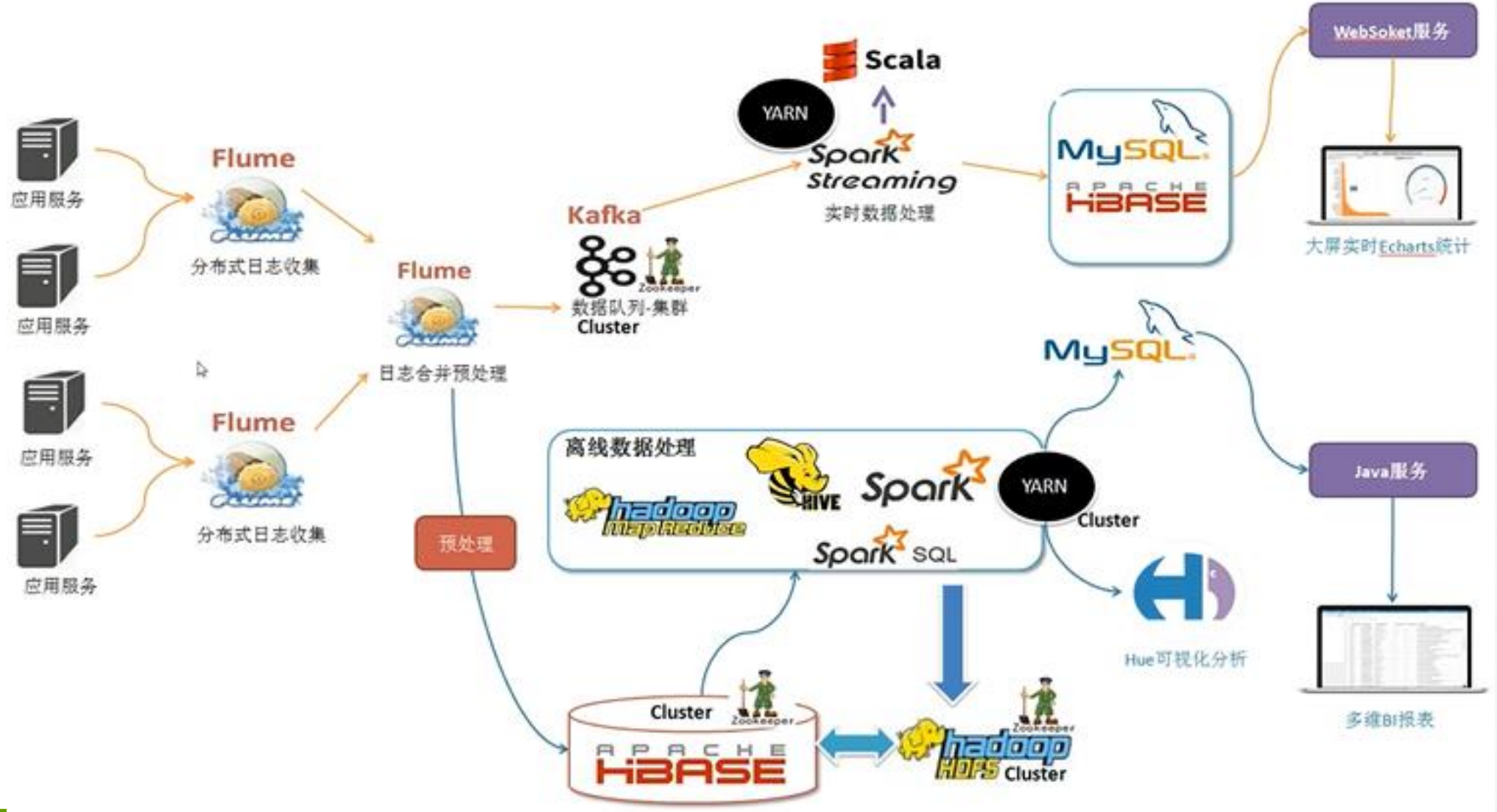



ES
实时查询
Kafka
Spark Stream
Flume
Hive
数据分析模型
统计
人工智能
机器学习
深度学习
Sqoop
Zoho
Spark ML (MLlib)
Mahout
Sk-learn
Redis
Kylin
Presto
Echarts
Excel
Tableau
Lumify
Kibana
风控
推荐


Data in HDFS
Data Collection (Kafka, Flume, ...)
Data processing
Storm
Spark
Flink
Data Storage (Mongo DB, Elastic Search, Hbase, ...)
Data visualisation

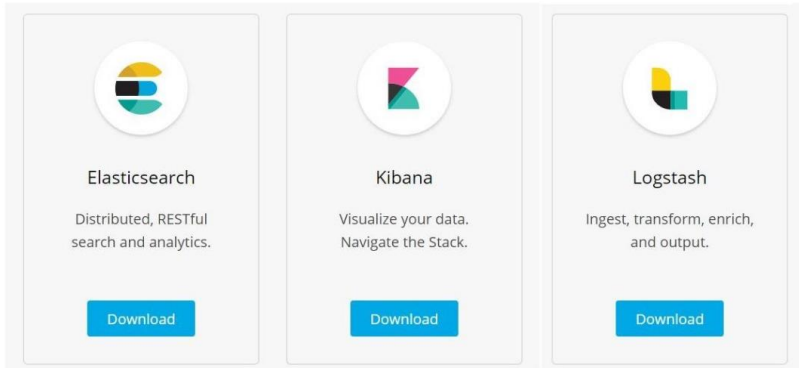
Another example with Storm, Redis, Dynamic UI etc.

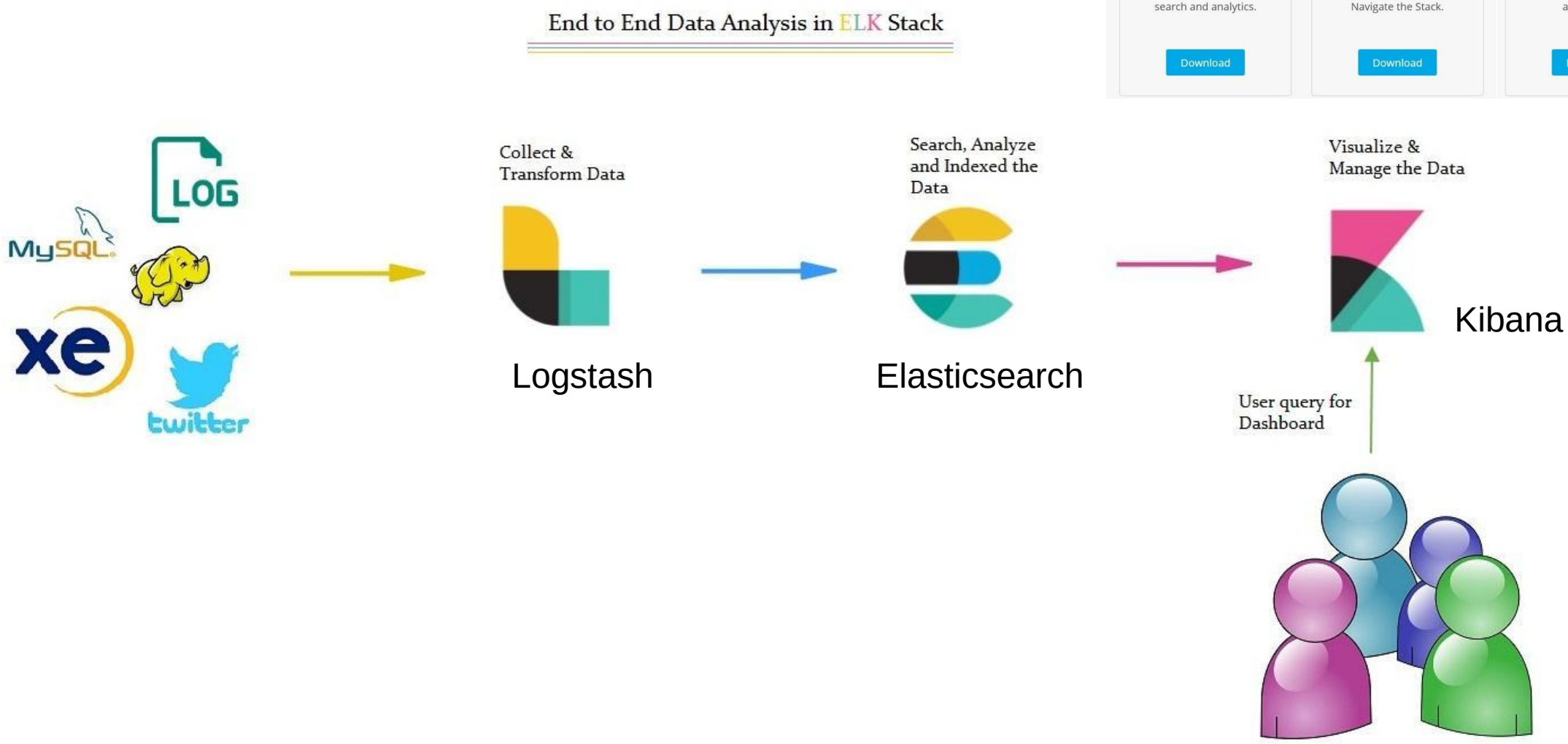



应用服务
应用服务
应用服务
应用服务
Flume
分布式日志收集
Flume
分布式日志收集
Flume
日志合并预处理
Kafka
Zookeeper
数据队列-集群
Cluster
Scala
YARN
Spark
Streaming
实时数据处理
MySQL
APACHE
HBASE
WebSoket服务
大屏实时Echarts统计
MySQL
离线数据处理
HIVE
Spark
hadoop
MapReduce
YARN
Cluster
Spark SQL
预处理
Java服务
Hue可视化分析
多维BI报表
Cluster
Zookeeper
APACHE
HBASE
hadoop
HDFS
Cluster

Data Analytics with ELK (Elasticsearch, Logstash and Kibana)

1. Go to its official website https://www.elastic.co/downloads and download below products in a separate directory





https://www.tatvasoft.com/blog/data-analytics-elasticsearch-logstash-kibana-elk/

Chapter 6: HUGE Concurrency architecture for “秒杀”

- **Architecture for “秒杀”**
  - Just “Divide and Conquer” – Data level, Module level
  - Big Data + Cloud now!
- **So-called Software architect [软件架构师]**

https://baike.baidu.com/item/软件架构师

- 软件架构师是软件行业中一种新兴职业，工作职责是在一个软件项目开发过程中，将客户的需求转换为规范的开发计划及文本，并制定这个项目的总体架构，指导整个开发团队完成这个计划。主导系统全局分析设计和实施、负责软件构架和关键技术决策的人员

- 软件架构师一般都是具备计算机科学或软件工程的知识，**由程序员做起，然后再慢慢发展为架构师的**。
  - 在国内，很多大学目前还没有设立软件架构的学位课程，虽然IT业界对设计和架构的兴趣日渐高涨，但各学校还是无法在课程中增加相应的内容来体现这一趋势。从这个方面来说，学校教育已经远远落后于产业发展。

能力要求

https://baike.baidu.com/item/软件架构师

- 在技术全面、成熟练达、洞察力强、经验丰富，具备在缺乏完整信息、众多问题交织一团、模糊和矛盾的情况下，软件架构师能迅速抓住问题要害，并做出合理的关键决定的能力、具备战略性和前瞻性思维能力，善于把握全局，能够在更高抽象级别上进行思考。主要包括如下：
  - 1 对项目开发涉及的所有问题领域都有经验，包括彻底地理解项目需求，开展分析设计之类软件工程活动等
  - 2 具备领导素质，以在各小组之间推进技术工作，并在项目压力下做出牢靠的关键决策；
  - 3 拥有优秀的沟通能力，用以进行说服、鼓励和指导等活动，并赢得项目成员的信任；
  - 4 以目标导向和主动的方式来不带任何感情色彩地关注项目结果，构架师应当是项目背后的技术推动力，而非构想者或梦想家（追求完美）；
  - 5 精通构架设计的理论、实践和工具，并掌握多种参考构架、主要的可重用构架机制和模式（例如J2EE架构等）；
  - 6 具备系统设计员的所有技能，但涉及面更广、抽象级别更高； 活动确定用例或需求的优先级、进行构架分析、创建构架的概念验证原型、评估构架的概念验证原型的可行性、组织系统实施模型、描述系统分布结构、描述运行时刻构架、确定设计机制、确定设计元素、合并已有设计元素、构架文档、参考构架、分析模型、设计模型、实施模型、部署模型、构架概念验证原型、接口、事件、信号与协议等。

主要任务

https://baike.baidu.com/item/软件架构师

- 架构师的主要任务不是从事具体的软件程序的编写，而是从事更高层次的开发构架工作。他必须对开发技术非常了解，并且需要有良好的组织管理能力。可以这样说，一个架构师工作的好坏决定了整个软件开发项目的成败。
  - 1 领导与协调整个项目中的技术活动（分析、设计和实施等）
  - 2 推动主要的技术决策，并最终表达为软件构架
  - 3 确定和文档化系统的相对构架而言意义重大的方面，包括系统的需求、设计、实施和部署等“视图”
  - 4 确定设计元素的分组以及这些主要分组之间的接口
  - 5 为技术决策提供规则，平衡各类涉众的不同关注点，化解技术风险，并保证相关决定被有效的传达和贯彻
  - 6 理解、评价并接收系统需求
  - 7 评价和确认软件架构的实现 专业技能

https://baike.baidu.com/item/软件架构师

- 虽然大学要加强软件架构学课程的建设，但是，软件架构师的成长应该有一个实践的教育过程，并不是简单的学校的理论学习或者通过大型软件公司的认证就能成为合格的软件架构师。除了信息系统综合知识在学校学习外，软件架构师的大部分知识和经验将来自实际开发工作。根据软件架构师的任职条件，一名合格的软件架构师的成长应该经历8年以上的软件项目开发实际工作经验。一般需要经历程序员、软件设计师等阶段，然后再发展成为软件架构师。
- 当然，并不是每一位程序员经过8年后都可以成长为软件架构师的。一个软件工程师在充分掌握了软件架构师工作所必需的基本理论和技能后，如何得到和利用机会、如何利用所掌握的技能进行应用系统的合理架构、如何不断的抽象和总结自己的架构模式、如何深入行业成为能够胜任分析、架构为一体的精英人才，这就在于机遇、个人的努力和天赋了。

https://baike.baidu.com/item/软件架构师

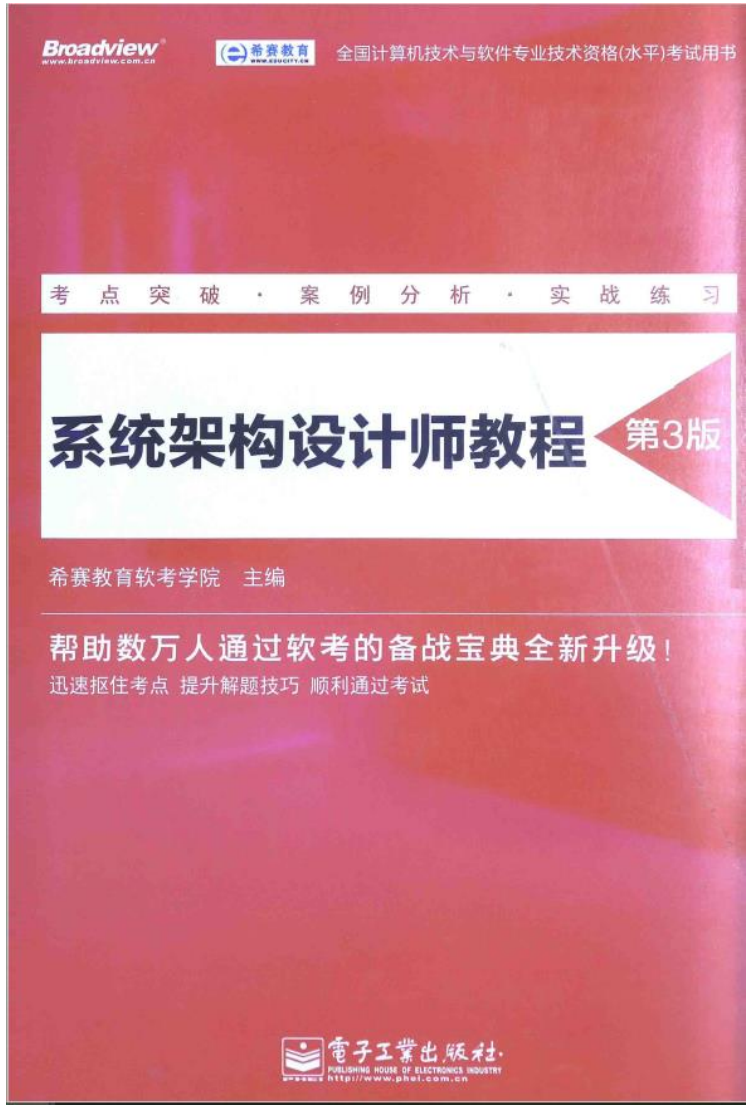
- 就目前来看，国内软件架构师的培养途径主要有两种方式，一种是大学（软件学院）教育方式，另一种是个人自我培养然后再进行相应的培训和认证。但是，不管哪种方式都有其不足之处。
  - 软件学院的培养方式能够系统的学习软件架构师必需的知识体系，但是，软件架构师不是简单的通过理论学习就能够培养出来的，软件学院的学生可能缺乏必要的设计、开发经验和相关的领域知识。尽管软件学院也强调给予学生实践的机会，但毕竟这种机会是有限的。有关“三分之一的师资来自企业”的规定，在部分软件学院中也没有得到真正落实，导致传授给学生的还是一些纯理论知识。
  - 自我培养方式的主要对象是具有一定年限的软件开发和设计人员，如Microsoft、IBM、Sun等公司的软件架构师认证对学员的基础并没有具体的要求，只要交纳规定的费用，然后进行几天的集中培训，通过考试就发给学员证书，甚至不需要考试就直接发放证书。这些开发人员在自我培养的过程中不一定能够系统的学习软件架构师的理论知识，他们只具有一定的开发和设计经验，仅仅经过几天的培训，是不太可能培养出合格的软件架构师的。而且，作为某个厂商的培训和认证，其最终目的是培育自己的市场，培养一批忠诚的用户，而不是为中国培养软件架构师。因此，也存在很大的问题和缺陷。

https://baike.baidu.com/item/软件架构师

- 针对软件架构师在软件组织中的作用和其在国内的培养现状，有分析家认为有必要将软件架构师的教育、培训和认证作为发展民族软件产业的[illegible]培养方案。因此，提出以下一些关于软件架[illegible]
  - (1) 确定软件架构师在软件组[illegible]软件架构师的职业及其相关制度[illegible]
  - (2) 坚持以大学教育为[illegible]大学可以聘请现有的软件[illegible]识体系；通过项目实践使其[illegible]
  - (3) 作位第2条的补充，[illegible]
  - (4) 对国外一些大公司的软[illegible]持符合中国实际情况的原则。例如，在认证考试之[illegible]通过认证后的适当时间内进行重新认证和继续教育。
  - (5) 建立完善的软件架构师教育和认证制度，使得通过认证的人员能够在实际的软件开发中成为称职的和优秀的软件架构师。并通过此制度能够为国家培养出更多、更优秀的软件架构师，解决当前软件架构师急缺问题。

个人观点：架构师，是能力的层次，不建议作为职位来理解。

意思就是，还是要打好扎扎实实的编程技能+商务思维，然后在工作中积极承担挑战、好好表现，坚持学习和积累，也就水到渠成了 ☺



- 由希赛教育软考学院组织编写，用以作为计算机技术与软件专业技术资格（水平）考试中的系统架构设计师级别的考试辅导指定教材。内容涵盖了的系统架构设计师考试大纲的所有知识点，对系统架构设计师所必须掌握的基础理论知识做了详细的介绍，重在培养系统架构设计师所必须具备的专业技能和方法

目录