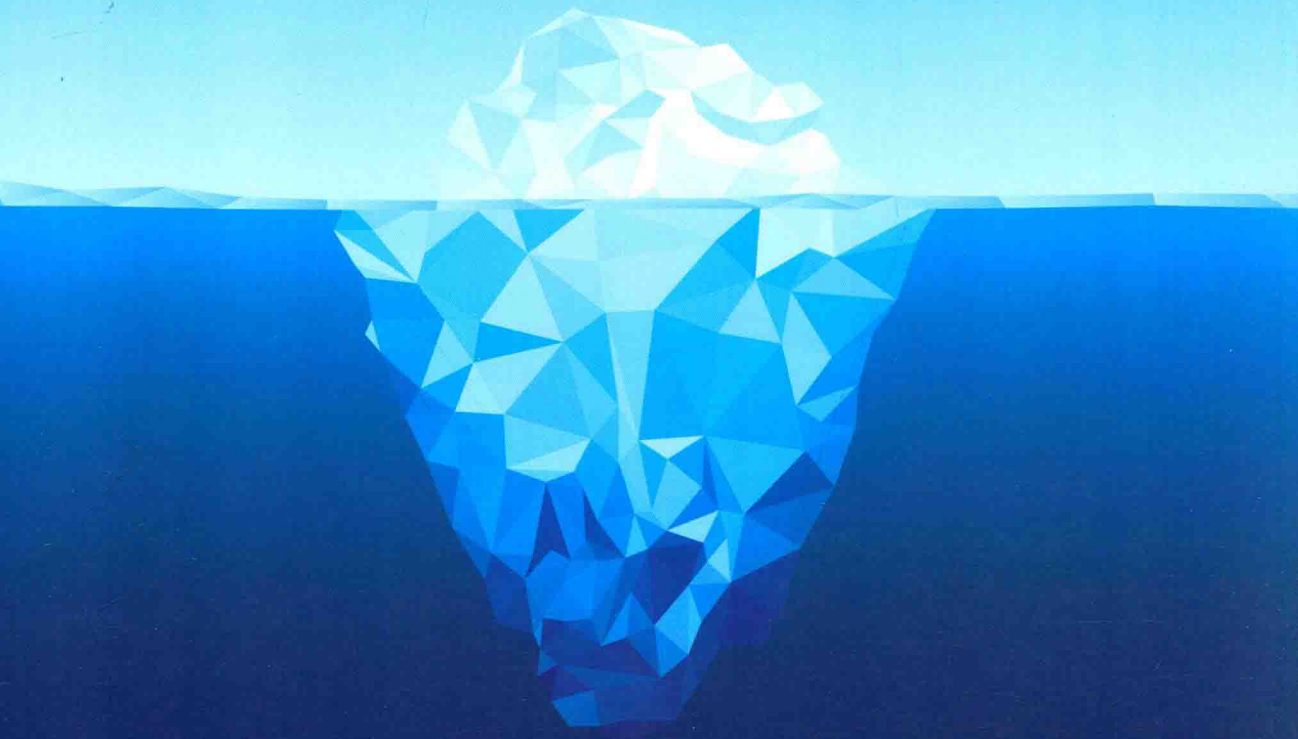
从本书中你能收获到的不仅仅只是架构理论，
更多的是来自互联网场景下大型网站架构演变过程中
核心技术难题的解决方案。

Broadview
www.broadview.com.cn



# 人人都是架构师

## 分布式系统架构落地与瓶颈突破

高翔龙 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

# 人人都是架构师

## 分布式系统架构落地与瓶颈突破

高翔龙 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

## 内 容 简 介

本书并没有过多渲染系统架构的理论知识，而是切切实实站在开发一线角度，为各位读者诠释了大型网站在架构演变过程中出现一系列技术难题时的解决方案。本书首先从分布式服务案例开始介绍，重点为大家讲解了大规模服务化场景下企业应该如何实施服务治理；然后在大流量限流/消峰案例中，笔者为大家讲解了应该如何有效地对流量实施管制，避免大流量对系统产生较大冲击，确保核心业务的稳定运行；接着笔者为大家讲解了分布式配置管理服务；之后的几章，笔者不仅为大家讲解了秒杀、限时抢购场景下热点数据的读/写优化案例，还为大家讲解了数据库实施分库分表改造后所带来的一系列影响的解决方案。

本书适用于任何对分布式系统架构感兴趣的架构师、开发人员以及运维人员。相信阅读本书你将会有知其然和知其所以然的畅快感。



**图书在版编目（CIP）数据**

人人都是架构师：分布式系统架构落地与瓶颈突破 / 高翔龙著. —北京：电子工业出版社，2017.5
ISBN 978-7-121-31238-0

Ⅰ. ①人… Ⅱ. ①高… Ⅲ. ①分布式计算机系统－系统设计 Ⅳ. ①TP338.8

中国版本图书馆 CIP 数据核字（2017）第 066402 号

策划编辑：孙学瑛
责任编辑：徐津平
特约编辑：赵树刚
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：12.5　字数：220 千字
版　　次：2017 年 5 月第 1 版
印　　次：2017 年 5 月第 1 次印刷
定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

# 前言

## 本书的创作初衷

任何一本书，都是一个用于承载知识的载体，读者可以从中探寻自己想要知道的答案。对于我而言，书本就是带我领略奇妙计算机世界最快的一条途径。之所以想创作一本与大型分布式系统架构相关的书籍，是因为我在最近几年的实际工作中经历了太多的技术难题。每当我和我的团队尝试解决这些问题之前，时常想着能否从市面上现有的架构书籍中寻求到解决方案；但事与愿违，目前市面上高歌架构理论的读物居多，而真正讲解大型网站在架构演变过程中出现技术难题时应该如何解决的书籍却寥寥无几。对于这块领域的空白，我想尝试着去创作，尽量把我自己脑海中的内容写出来，让更多人受益，毕竟架构是需要落地的，否则便是一纸空谈。

## 本书内容重点

本书每一章的内容几乎都是独立的，大家完全可以挑选自己感兴趣或者有需要的部分进行阅读。本书一共包含 5 章，笔者首先从分布式服务案例开始讲起，将大家带进分布式系统的殿堂。在第 1 章中，笔者讲解了大型网站的架构演变过程，让大家对分布式系统建立一个基本的认识。当然，本章的重点是讲解企业在大规模服务化后应该如何实施服务治理，以及应该如何构建一个分布式调用跟踪系统，以一

种可视化的方式来展现跟踪到的每一个请求的完整调用链，并收集调用链上每个服务的执行耗时，整合孤立日志等。

为了避免大促场景下峰值流量过大，对系统造成较大负载导致产生雪崩现象，笔者在本书的第 2 章为大家讲解了大流量限流/消峰案例，让系统的负载压力始终处于一个比较均衡的水位，从而保护系统的稳定运行。笔者首先从限流算法开始讲起，然后分享了业务层面和技术层面等两个维度的流量管制方案。当然，本章的重点是为大家演示如何通过 MQ 来实现大流量场景下的流量消峰。

本书的第 3 章为大家讲解了分布式配置管理服务案例（配置中心）。尽管目前一些中小型互联网企业仍然将本地配置作为首选，但是当网站发展到一定规模后，继续采用本地配置所暴露的问题将会越来越多。大型网站使用分布式配置管理平台不仅能够实现配置信息的集中式管理、降低维护成本和配置出错率，还能够动态获取/更新配置信息。本章的重点是为大家演示如何基于 ZooKeeper 构建一个分布式配置管理平台，以及使用淘宝 Diamond 和百度 Disconf 系统来实现分布式配置管理服务。

热点数据的读/写操作其实是秒杀、限时抢购场景下最核心的技术难题。在大促场景下，由于峰值流量较大，大量针对同一热卖商品的并发读/写操作一定会导致后端的存储系统产生性能瓶颈，因此第 4 章为大家讲解了大促场景下热点数据的读/写优化案例。尽管商品信息可以缓存在分布式缓存中，通过集群技术，可以在理论上认为其容量是无限的，但是对于大促场景下的热卖商品来说，由于单价比平时更给力、更具吸引力，因而自然会比平时吸引更大的流量进来；这时同一个 Key 必然会落到同一个缓存节点上，而分布式缓存在这种情况下一定会出现单点瓶颈，因此笔者为大家演示了如何实施多级 Cache 方案来防止分布式缓存系统出现单点瓶颈。由于写操作无法直接在缓存中完成，因此大量的并发更新热点数据（库存扣减）都是针对数据库中同一行的——本书以 MySQL 为例，而这必然会引起大量的线程来相互竞争 InnoDB 的行锁；并发越大时，等待的线程就越多，这会严重影响数据库的 TPS，导致 RT 线性上升，最终可能引发系统出现雪崩。为了避免数据库沦为瓶颈，笔者为

大家演示了如何通过分布式锁、乐观锁在分布式缓存系统中扣减库存、通过抢购限流控制单机并发写流量，以及如何使用阿里开源的 AliSQL 数据库提升“秒杀”场景性能。

在本书的最后一章，笔者为大家讲解了数据库分库分表案例。本章演示了如何通过分库分表中间件 Shark 来帮助企业实施分库分表改造，以及分库分表后所带来一系列影响的解决方案，并重点分享了笔者在实际工作中订单业务实施分库分表改造后，应该如何同时满足 Buyer 和 Seller 的多维度查询需求。

## 本书面向的读者

本书适用于任何对分布式系统架构感兴趣的架构师、开发人员以及运维人员。笔者尽量用通俗易懂的文字描绘本书的各个知识点，并引用了大量在实际工作中笔者遇到的那些真实案例，相信阅读本书时你将会有知其然并知其所以然的畅快感。

## 读者讨论

由于笔者能力有限，书中难免会出现一些错误或者不准确的地方，你可以通过邮箱 gao_xianglong@sina.com 将问题反馈给我，我会尽量对所有问题都给予答复。

## 致谢

首先我要感谢我们家莹宝宝，是你的支持和鼓励才让我有了继续创作下去的勇气。还记得在本书的创作过程中，每当我写完一节时，我都会“强迫”你高声朗读帮我梳理下笔的准确度；以及每当我头痛欲裂思绪全无时，你的陪伴点燃了我在每个凌晨的斗志；甚至在我烦躁时，你总是毫无怨言地忍受着我的“坏脾气”。谢谢你的包容和体贴，我爱你。

其次我要感谢我的团队：我的两位 BOSS——冰冰和校长，最牛的 MySQL DBA 平哥，架构师大飞、青龙、小狼、僧哥、布爸，感谢你们平时在工作上的支持。

当然，本书能够顺利出版，离不开本书的两位编辑：孙学瑛老师和 Anna 老师的共同努力；感谢你们辛苦的文字校对工作，同时也祝愿孙学瑛老师家的猴宝宝健康茁壮地成长。

最后感谢那些曾经帮助过我的所有人，我爱你们！

高翔龙

2016 年 12 月 31 日深夜

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务。

- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与我们交流**：在页面下方【读者评论】处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31238

二维码：

# 目录

# 1 第1章 分布式服务案例

写一本书，远远不是大家想象得那么简单，更不是思绪如泉涌般的信手拈来，其中的艰辛和酸甜苦辣想必只有作者自己才能够深刻体会到。笔者在创作本书时，可谓是下笔艰难、字斟句酌，当然这一切都是为了能够将笔者想说的和想展现的内容毫无保留地诠释给各位读者。本章作为本书的开篇，也是笔者深思熟虑后的结果。目前，在互联网场景下应对大流量、高并发，服务化架构改造是必不可少的。在本章中，笔者为大家讲解了互联网场景下大型网站架构的演变过程、服务化的一些必备基础知识，以及如何使用开源社区流行的 RPC 和服务治理框架 Dubbo 帮助企业落地服务化架构。当然，本章的重点是为大家演示如何实现一个拥有较低侵入性的分布式调用跟踪系统来帮助企业实施服务治理，因为在一些大规模的服务调用场景下，我们必然需要一种有效且可视化的手段来帮助开发人员、架构师更好地梳理清楚服务或服务之间的依赖关系、调用顺序，以及服务的执行耗时等。

OK，接下来就请大家跟随笔者一起来探询分布式场景下服务化的真谛吧！Let's GO!

## 1.1 分布式系统的架构演变过程

在移动互联网的浪潮中，你我正生逢其时地享受着当下，如果你愿意做一只站在风口上等待起飞的猪，那么请认真地问问自己，是否已经准备好了？互联网究竟是什么？简而言之，互联网诠释的是一种精神，融入了高度开放、分享，以及自由的精神。如果你想融入这个圈子，那么请务必先舍弃掉与互联网精神背道而驰的陈旧观念和思维，否则你自始至终都只会被孤立在局外。

互联网悄然改变了世界，改变了人们对事务的认知，缩短了人与人之间的距离。无论你是否愿意承认，互联网已经完全影响并融入我们的生活中。我们的长辈们，也从早期对新鲜事物的排斥，变成现在的欣然接受，这就是互联网与生俱来的魅力和魔力。笔者的母亲从来就不是一个喜欢追赶潮流的人，但是她早已智能设备从不离身，每天早上起床的第一件事情就是拿起智能手机，刷刷朋友圈、看看时事政治、做回“吃瓜群众”，八卦下娱乐新闻，甚至衣食住行也几乎是通过互联网这个载体一键搞定的，如图 1-1 所示。既然互联网能使我们的生活质量更好，那就请张开双臂紧紧拥抱它。

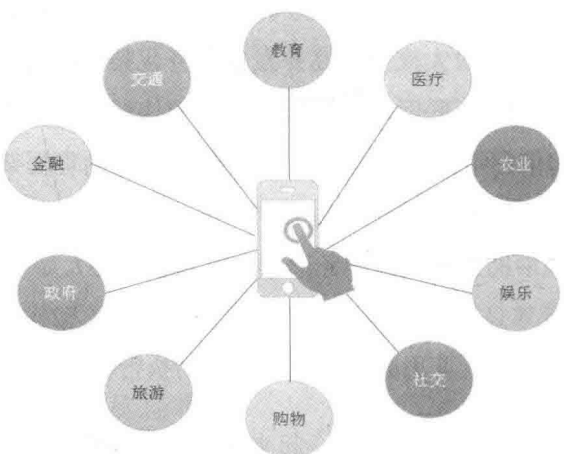



图 1-1 拥有互联网的生活

拥有互联网的夜晚是明亮的，当然为这寂静夜空点燃光明的正是聚集在各个互联网企业内部的技术团队，正是这些家伙昼夜颠倒的辛勤付出（无数的通宵、无数的会议、无数的版本迭代、无数的交互体验优化、无数的系统架构升级），才换来今日互联网的光彩夺目。不过任何事物都如同硬币一般具备两面性，我们不能“光见贼吃肉，不见贼挨打”，这句话放在互联网领域似乎相当应景，很多电商网站为了吸引用户流量，往往都会以极低的价格作为诱饵，但是当促销活动正式开始后，峰值流量却远远超出了系统所能够承受的处理能力，这必然只会产生一种结果——宕机。如何帮助企业顺利走出困境，打造真正具备高性能、高可用、易扩展、可伸缩，以及相对安全的网站架构才是架构师、技术大牛们应该重点思考的问题和责任，哪怕是系统宕机，也要尽最大努力做到“死而不僵”，用技术支撑业务的野蛮生长，活下去才会有更美好的明天。

一般来说，网站由小变大的过程，几乎都需要经历单机架构、集群架构、分布式架构。伴随着业务系统架构一同演变的还有各种外围系统和存储系统，比如关系型数据库的分库分表改造、从本地缓存过渡到分布式缓存等。当系统架构演变到一定阶段且逐渐趋向于稳定和成熟后，架构师们需要对技术细节追本溯源，如果现有的技术或者框架不能有效满足业务需要，就需要从“拿来主义”的消费者角色转变为自行研发的生产者角色。当然，在这条技术之路离你和你所在的企业还很遥远的时候，尽管未雨绸缪利大于弊，但这却并不是你现阶段的工作重点。在此大家需要注意，对技术由衷的热爱和痴迷本身并没有什么不对，但是千万不能够过于沉醉而选择刻意忽略掉业务，任何技术的初衷都是为了更好地服务业务，一旦脱离业务，技术必然会失去原有的价值和意义，切勿舍本逐末。

### 1.1.1 单机系统

任何一个网站在发布初期几乎都不可能立马就拥有庞大的用户流量和海量数据，都是在不停的试错过程中一步一步演变其自身架构，满足其自身业务。所以我

们常说，互联网领域几乎没有哪一个网站天生就是大型网站，因为往往系统做大与业务做大是呈正比的，大型网站都是从小型网站逐渐演变过来的，而不是被刻意设计出来的。试想一下，如果业务不见起色，一味地追求大型网站架构又有何意义呢？

对于一个刚上线的项目，我们往往会将 Web 服务器、文件服务器和数据库全都部署在同一台物理服务器上，这样做的好处只有一个——省钱，如图 1-2 所示。资金紧张且用户流量相对较小的网站，采用这样的架构进行部署确实非常实惠，毕竟用户流量上不去，自然就没有必要考虑更多的问题，哪怕是系统宕机，影响范围也不大。当然，一旦业务开始加速发展，用户逐渐增多，系统瓶颈便会开始暴露，这时架构师就可以考虑对现有网站架构做出以下四点调整：

- 独立部署，避免不同的系统之间相互争夺共享资源（比如 CPU、内存、磁盘等）；
- Web 服务器集群，实现可伸缩性；
- 部署分布式缓存系统，使查询操作尽可能在缓存命中；
- 数据库实施读/写分离，实现 HA（High Availability，高可用性）架构。

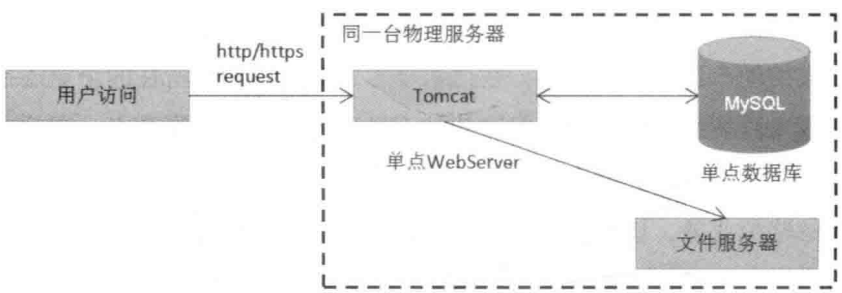



图 1-2 在同一台机器上部署单机系统

## 1.1.2 集群架构

一旦用户开始增多，并发流量上来后，为了让用户拥有更好的操作体验，我们

不得不对单机系统架构做出调整和优化，因此在这个阶段主要需要解决的问题就是提升业务系统的并行处理能力，降低单机系统负载，以便支撑更多的用户访问操作。

集群（Cluster）技术可以将多台独立的服务器通过网络相互连接组合起来，形成一个有效的整体对外提供服务，使用集群的意义就在于其目标收益远高于所付出的实际成本和代价。一般，互联网领域都有一个共同的认知，那就是当一台服务器的处理能力接近或已超出其容量上限时，不要企图更换一台性能更强劲的服务器，通常的做法是采用集群技术，通过增加新的服务器来分散并发访问流量，1 台不够就扩到 2 台，2 台不够就扩到 4 台，只要业务系统能够随意支持服务器的横向扩容，那么从理论上来说就应该无惧任何挑战，从而实现可伸缩性和高可用性架构。

如图 1-3 所示，对于无状态的 Web 节点来说，通过 Nginx 来实现负载均衡调度似乎是一个不错的选择，但是在生产环境中，Nginx 也应该具备高可用性，这可以依靠 DNS 轮询来实现。在集群环境中，Web 节点的数量越多，并行处理能力就越强，哪怕其中某些节点因为种种原因宕机，也不会使系统的整体服务不可用。

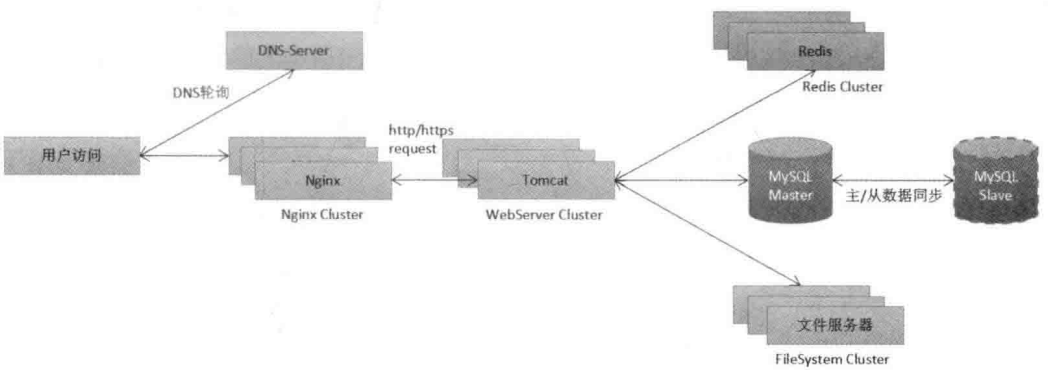



图 1-3 在独立的机器上部署集群系统

伴随着 Web 集群改造的还有分布式缓存和数据库，对于查询操作我们应该尽可能在缓存命中，从而降低数据库的负载压力。尽管缓存技术可以解决数据库的大部