大规模分布式系统架构与设计实战 (大数据技术丛书)

作者：彭渊

大数据技术丛书

大规模分布式系统架构与设计实战

彭渊　著

ISBN：978-7-111-45503-5

本书纸版由机械工业出版社于2014年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目录

前言

在大数据、云计算如火如荼的今天，各类技术产品顺应潮流层出不穷。大家是不是有这种感觉：Hadoop还没学完，Storm就来了；Storm刚学会安装配置，Spark、Hama、Yarn等又一起出现了；同时国内外各大云平台厂商，如Google、亚马逊、阿里云等，还在推各自应用开发平台……要学习的东西太多了，就是这样疲于奔命地学，刚学会了某个产品的安装配置与开发步骤，没多久它又过时了。

这么多千姿百态的分布式技术和产品背后有没有某些共性的东西呢？能让我们换了马甲还能认出它，让我们超越学习每个产品的“安装配置开发”而掌握背后的精髓呢？有没有可能学一反三，学一招应万招，牢牢掌握好技术的船舵，穿越一次次颠覆性的技术浪潮？本书的目的就是为你揭示分布式技术的核心内幕，透彻理解其精髓，站在浪潮之巅。

因此，这不是一本讲如何使用Hadoop的书，而是一本讲实现Hadoop功能的书，是一本讲如何简化实现分布式技术核心功能的书。这不是一本空谈概念、四处摘抄的书，而是来源于作者十多年来在私企、港企、外包、创业、淘宝、华为等企业打拼，从底层程序员一路走到首席架构师的实战经验总结。绝技源于江湖，将军发于卒伍，这本书讲的是你在课本上学不到核心技术，无论你是在中国什么样的IT企业做什么样的分布式应用，这本书对你都具备参考性。

本书面向千千万万战斗在一线攻城拔寨的程序员、工程师们，你可以有很多基础，也可以从头开始，本书尽量做到深入浅出和通俗易懂，希望帮助你降低分布式技术的学习成本，帮助你更容易完成工作任务，更轻松地挣钱。

本书根据分布式技术的主要应用，分别介绍分布式并行计算的基本概念、分布式协调、分布式缓存、消息队列、分布式文件系统、分布式作业调度平台等，详细阐述分布式各技术的架构原理和实现方式，并附带大量示例，便于读者实际操作运行。基于本书原理，作者用Java实现并开源了Fourinone框架，这是一个高效的分布式系统，归纳在150KB源码里，代码不到1万行，让你能够轻松掌握。学习开发核心技术的诀窍是多动手，建议读者运行本书附带的大量DEMO，在运行后细细体会分布式的理论，进行反思和总结。本书归纳的设计思想和算法不局限于某个框架，读者领会后可以用任何语言来实现自己的分布式系统。

本书各章有一定的独立性，阅读本书的方式比较自由，可以从头开始，也可以随性翻阅。从第2章开始，每章都有理论部分与示例，读者可以先运行DEMO，不清楚的地方再回看原理；也可以先看原理，再运行DEMO加深理解。由于时间的限制，且本书写作的时期是在作者最为忙碌和事业的转折时期，匆忙中，难免出错，请朋友们海涵，并提出意见以便于今后纠正。最后感谢机械出版社华章公司所有幕后编辑的大量工作，感谢所给予我帮助与支持的领导和朋友。

本书所有源码附带在光盘里。你也可以登录开源地址下载，开源地址：http：//code.google.com/p/fourinone

作者联系方式：邮箱：Fourinone@yeah.net

QQ群1：1313859

QQ群2：241116021

QQ群3：23321760

第1章　概述

在概述分布式核心技术之前，我们有必要先概括阐述一下分布式计算、并行计算、云计算等相关概念，以及市场上流行的相关技术产品，如Hadoop生态体系，然后再结合背景引出我们为什么要归纳出一个轻量级的分布式框架。本章为后续章节的背景。本章意在使读者对分布式技术话题的前因后果先有所了解。

由于只是概述，我们对涉及的分布式计算概念和Hadoop生态体系只是蜻蜓点水地带过，目的仅是让读者了解到这些内容大致是什么，详细的使用方法和开发方法可以参考其他书籍。

1.1　分布式计算、并行计算、云计算概述

1.什么是分布式计算

图1-1　科学中的分布式计算

经科学研究发现：目前存在很多万亿次计算实例，其中涉及的问题都需要非常巨大的计算能力才能解决。这类问题很多还是跨学科的、极富挑战性的、人类亟待解决的科研课题，如图1-1所示。

除此之外还有很多研究项目需要巨大的计算能力，例如：

1）解决较为复杂的数学问题，例如：GIMPS（寻找最大的梅森素数）。

梅森素数(Mersenne Prime）是指形如2^p－1的正整数，其中指数p是素数，常记为Mp。若Mp是素数，则称为梅森素数。p=2，3，5，7时，Mp都是素数，但M11=2047=23×89不是素数，是否有无穷多个梅森素数是数论中未解决的难题之一。截至2012年7月已累计发现47个梅森素数，最大的是p=43，112，609，此时Mp是一个12，978，189位数。

值得一提的是，中国的一位数学家算出了梅森素数的分布规律图，并用简练的数学公式描述了出来。如果借助计算机的并行计算，也许会对寻找该数字分布规律有所帮助。

2）研究寻找最为安全的密码系统，例如：RC-72（密码破解）。

3）生物病理研究，例如：Folding@home（研究蛋白质折叠、误解、聚合，以及由此引起的相关疾病）。

4）各种各样疾病的药物研究，例如：United Devices（对抗癌症的有效药物）。

5）信号处理，例如：SETI@Home（寻找地球外文明）。

由上不难看出，这些项目都很庞大，都需要惊人的计算量，仅由单个电脑或个人在一个能让人接受的时间内计算完成是决不可能的。在以前，这些问题都应该由超级计算机来解决。但是，超级计算机的造价和维护非常昂贵，这不是一个普通的科研组织能承受的。随着科学的发展，一种廉价的、高效的、维护方便的计算方法应运而生——分布式计算！

所谓分布式计算其实就是一门计算机科学，它研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给许多计算机进行处理，最后把这些计算结果综合起来得到最终的结果。

最近的一个分布式计算项目已经使用世界各地成千上万位志愿者的计算机来进行操作，利用这些闲置计算能力，通过因特网，你可以分析来自外太空的电信号，并探索可能存在的外星智慧生命；你可以寻找超过1000万位数字的梅森素数；你也可以寻找并发现对抗艾滋病病毒的更为有效的药物。

讨论

我们能否利用访问淘宝网的几千万个用户的电脑做一次分布式计算？

这里仅仅点拨一下，回答该问题其实涉及侵犯用户隐私。用户用电脑上网，安装了很多客户端软件，有的软件是拥有本地所有权限的，它们是否在偷偷用用户的电脑干其他私活，用户也不知道。这曾经引发过国内两大客户端巨头的官司。但是由此可以看出，千千万万用户的电脑是可以利用起来做计算的，当然计算必须围绕一个消息中枢模式进行，因为用户电脑间的网络结构千差万别，无法直接连接。

2.什么是并行计算

并行计算其实早就有了，所有大型编程语言都支持多线程，多线程就是一种简单的并行计算方式，多个程序线程并行地争抢CPU时间。

并行计算（Parallel Computing）是指同时使用多种计算资源解决计算问题的过程。并行计算的主要目的是快速解决大型且复杂的计算问题。此外还包括：利用非本地资源节约成本，即使用多个“廉价”计算资源取代大型计算机，同时克服单个计算机上存在的存储器限制问题。

传统上，串行计算是指在单个计算机（具有单个中央处理单元）上执行软件写操作。CPU逐个使用一系列指令解决问题，但在每一个时刻只能执行一种指令。并行计算是在串行计算的基础上演变而来的，它努力仿真自然世界中的事务状态：一个序列中众多同时发生的、复杂且相关的事件。

为利用并行计算，通常计算问题表现为以下特征：

- 将工作分解成离散部分，有助于同时解决；
- 随时并及时地执行多个程序指令；
- 多计算资源下解决问题的耗时要少于单个计算资源下的耗时。

并行计算是相对于串行计算来说的，所谓并行计算分为时间上的并行和空间上的并行。时间上的并行就是指流水线技术，而空间上的并行则是指用多个处理器并发地执行计算。

3.并行计算与串行计算的关系

并行计算与串行计算的关系如图1-2所示。

图1-2　串行（上图），并行（下图）

结合图1-2，对串行计算和并行计算分析如下：

- 传统的串行计算，分为“指令”和“数据”两个部分，并在程序执行时“独立地申请和占有”内存空间，且所有计算均局限于该内存空间。
- 并行计算将进程相对独立的分配于不同的节点上，由各自独立的操作系统调度，享有独立的CPU和内存资源（内存可以共享）；进程间相互信息交换是通过消息传递进行的。

4.什么是云计算

云计算是一种理念，是旧瓶子装新酒，它实际上是分布式技术+服务化技术+资源隔离和管理技术（虚拟化），如图1-3所示。商业公司对云计算都有自己的定义，例如：

- 一种计算模式：把IT资源、数据、应用作为服务通过网络提供给用户（如IBM公司）。
- 一种基础架构管理方法论：把大量的高度虚拟化的资源管理起来，组成一个大的资源池，用来统一提供服务（如IBM公司）。
- 以公开的标准和服务为基础，以互联网为中心，提供安全、快速、便捷的数据存储和网络计算服务（如Google公司）。

图1-3　云计算示意图

通俗意义上的云计算往往是上面这个架构图包含的内容，开发者利用云API开发应用，然后上传到云上托管，并提供给用户使用，而不关心云背后的运维和管理，以及机器资源分配等问题。

虚拟化和服务化是云计算的表现形式（参见图1-4）：

图1-4　云计算表现形式

❑　虚拟化技术包括：资源虚拟化、统一分配监测资源、向资源池中添加资源。虚拟化的技术非常多，有的是完全模拟硬件的方式去运行整个操作系统，比如我们熟悉的VMWare，可以看做重量级虚拟化产品。也有通过软件实现的，共享一个操作系统的轻量级虚拟化，比如Solaris的Container、Linux的lxc(关于cgroup的方式我们在7.2节也会谈到）。虚拟化的管理、运维多数是通过工具完成的，比如Linux的VirtManager、VMWare的vSphere、VMWare的vCloud等等。

❑　服务思想包括：

○　软件即服务(Software-as-a-Service，SAAS）。是目前最为成熟的云计算服务模式。在这种模式下，应用软件安装在厂商或者服务供应商那里，用户可以通过某个网络来使用这些软件。这种模式具有高度的灵活性、可靠性和可扩展性，因此能够降低客户的维护成本和投入，而且运营成本也得以降低。最著名的例子就是Salesforce.com。

○　平台即服务(Platform-as-a-Service，PAAS）。提供了开发平台和相关组件，软件开发者可以在这个开发平台之上开发新的应用，或者使用已有的各种组件，因此可以不必购买开发、质量控制或生产服务器。Salesforce.com的Force.com、Google的App Engine和微软的Azure(微软云计算平台）都采用了PAAS的模式。

○　基础设施作为服务(Infrastructure as a Service，IAAS）。通过互联网提供了数据中心、硬件和软件基础设施资源。IAAS可以提供服务器、操作系统、磁盘存储、数据库和/或信息资源。用户可以像购买水电煤气一样购买这些基础设施资源使用。

1.2 分布式产品Hadoop、ZooKeeper、HBase概述

1.Hadoop

说到云计算技术和产品，不能不提到Google这家企业。曾经，微软是IT行业的象征，号称只招最聪明的人。十年后，微软逐渐疲软了下来，Google这家企业取而代之号称只招最聪明的人。

从搜索引擎到卫星地图，到云计算，到风靡世界的Android，到现在的无人汽车、Google眼镜，以及传说中的机器人之父和在他家里满地爬的机器人……这个行业总有这么一家企业成为最高科技的领袖，控制着大部分的核心技术，聚拢着大部分的一流人才，垄断着最大份额的市场，几乎让其他公司望而却步。

因此，我们不难理解这个行业的结果，第一的企业吃肉，第二喝点汤，其他都亏损。

也许在学校读书时，考试成绩前十名都是好学生，但是IT行业的社会竞争，必须要做到第一，成为所在领域的老大才有利可图。

再简单回顾一下云计算相关技术产品的发展史：

- 2002～2004：Apache Nutch。
- 2004～2006：Google发表GFS和MapReduce相关论文。Apache在Nutch中实现HDFS和MapReduce。
- 2006～2008：Hadoop项目从Nutch中分离。
- 2008年7月，Hadoop赢得Terabyte Sort Benchmark。

从上面我们可以看到早在2004～2006年，Google就发表了两篇与GFS和MapReduce相关的论文，分别是分布式文件系统和基于它的Map/Reduce并行计算。Apache的开源项目Hadoop便是根据这两篇论文的思想实现的Java版本，Hadoop引起关注是在它赢得了一次TB排序比赛：Hadoop Wins Terabyte Sort Benchmark：One of Yahoo's Hadoop clusters sorted 1 terabyte of data in 209 seconds，which beat the previous record of 297 seconds in the annual general purpose(Daytona）terabyte sort benchmark.This is the first time that either a Java or an open source program has won.

用了大量的机器在209秒完成1TB排序，提升了之前297秒的记录。

值的一提的是，Hadoop的作者Doug Cutting同时也是Lucene和Nutch的作者，早年供职Yahoo，后来担任Apache软件基金会主席。

Hadoop的名字来源于Doug Cutting儿子的一个宠物名。Hadoop从最初的HDFS分布式文件系统发展到后来的Hadoop+ZooKeeper+Hive+Pig+HBase生态体系。

HDFS提供了一个可扩缩的、容错的、可以在廉价机器上运行的分布式文件系统，按行进行存储，按64MB块进行文件拆分。

图1-5 HDFS体系结构

我们可以看到，HDFS的架构是一个NameNode和多个DataNode的结构（参见图1-5）：

- NameNode

存储HDFS的元数据(metadata）。

管理文件系统的命名空间（namespace）。

创建、删除、移动、重命名文件和文件夹。

接收从DataNode来的Heartbeat和Blockreport。

- DataNode

存储数据块。

执行从NameNode来的文件操作命令。

定时向NameNode发送Heartbeat和Blockreport。

除了提供分布式文件存储外，Hadoop还提供基于Map/Reduce的框架，进行按行的并行分析，可以用来查询和计算。

图1-6是以Word Count为例子演示Map/Reduce机制的图，学习过Hadoop的人一般都看到过，有趣的是，我们在2.1.4节也会基于Fourinone的方式实现一个Word Count。

图1-6 Word Count

图1-7中Hadoop的Map/Reduce实际上是1.0版，也许Hadoop项目组也意识到该框架的局限性，在目前Map/Reduce2.0（Yarn）版中实际上已经完全进行了重构，整个设计思想是做成一个资源和任务的调度框架，再也不是Google的Map/Reduce相关论文阐述的内容了。关于Yarn我们在7.4.2节也会谈到。

关于Hadoop的详细资料可以参考以下信息：

http：//hadoop.apache.org/hdfs/docs/current/hdfs_design.html

http：//labs.google.com/papers/gfs.html

2.ZooKeeper

ZooKeeper在Hadoop生态体系中是作为协同系统出现的，为什么会独立出一个协同系统呢？我们看看跟分布式协同相关的一些重要概念。

- 分布式协同系统：大型分布式应用通常需要调度器、控制器、协同器等管理任务进程的资源分配和任务调度，为避免大多数应用将协同器嵌入调度控制等实现中，造成系统扩充困难、开发维护成本高，通常将协同器独立出来设计成为通用、可伸缩的协同系统。
- ZooKeeper：Hadoop生态系统的协同实现，提供协调服务，包括分布式锁、统一命名等。
- Chubby：Google分布式系统中的协同实现和ZooKeeper类似。
- Paxos算法：1989年由莱斯利·兰伯特提出，此算法被认为是处理分布式系统消息传递一致性的最好算法。
- 领导者选举：计算机集群中通常需要维持一个领导者的服务器，它负责进行集群管理和调度等，因此集群需要在启动和运行等各个阶段保证有一个领导者提供服务，并且在发生故障和故障恢复后能重新选择领导者。

当前业界分布式协同系统的主要实现有ZooKeeper和Chubby，ZooKeeper实际上是Google的Chubby的一个开源实现。ZooKeeper的配置中心实现更像一个文件系统，文件系统中的所有文件形成一个树形结构，ZooKeeper维护着这样的树形层次结构，树中的结点称为znode，每个znode存储的数据有小于1MB的大小限制。ZooKeeper提供了几种znode类型：临时znode、持久znode、顺序znode等，用于不同的一致性需求。在znode发生变化时，通过“观察”（watch）机制可以让客户端得到通知。可以针对ZooKeeper服务的“操作”来设置观察，该服务的其他操作可以触发观察。ZooKeeper服务的“操作”包括一些对znode添加修改获取操作。ZooKeeper采用一种类似Paxos的算法实现领导者选举，以解决集群宕机的一致性和协同保障。总体上说，ZooKeeper提供了一个分布式协同系统，包括配置维护、名字服务、分布式同步、组服务等功能，并将相关操作接口提供给用户。

ZooKeeper的结构如图1-7所示。

图1-7　ZooKeeper的结构示意

ZooKeeper大致工作过程如下：

1）启动ZooKeeper服务器集群环境后，多个ZooKeeper服务器在工作前会选举出一个Leader，若在接下来的工作中这个被选举出来的Leader死了，而剩下的ZooKeeper服务器会知道这个Leader死掉了，系统会在活着的ZooKeeper集群中会继续选出一个Leader，选举出Leader的目的是在分布式的环境中保证数据的一致性。

2）另外，ZooKeeper支持watch(观察）的概念。客户端可以在每个znode结点上设置一个watch。如果被观察服务端的znode结点有变更，那么watch就会被触发，这个watch所属的客户端将接收到一个通知包被告知结点已经发生变化。若客户端和所连接的ZooKeeper服务器断开连接时，其他客户端也会收到一个通知，也就是说一个ZooKeeper服务器端可以服务于多个客户端，当然也可以是多个ZooKeeper服务器端服务于多个客户端。

我们这里只是讲述了ZooKeeper协同的基本概念，第3章我们还会详细讲如何实现这样的协同系统，并与ZooKeeper进行一些对比。

3.HBase

HBase是NoSQL技术的产物，NoSQL风行后，很多互联网应用需要一个面向键/值的列存储数据库，并可以支持水平扩充，当然Google在这个领域又走在了前面。

HBase是Google Bigtable的开源实现。Google Bigtable利用GFS作为其文件存储系统，HBase利用Hadoop HDFS作为其文件存储系统；Google运行MapReduce来处理Bigtable中的海量数据，HBase同样利用Hadoop MapReduce来处理海量数据；Google Bigtable利用Chubby作为协同服务，HBase利用ZooKeeper作为对应的功能。

图1-8是HBase的架构，里面的HDFS也就是Hadoop中的分布式文件系统。对里面主要的核心组件简单介绍如下：

图1-8　HBase

（1）Client

Client包含访问HBase的接口，维护着一些cache来加快对HBase的访问，比如Regione的位置信息。

（2）ZooKeeper

保证任何时候，集群中都只有一个Master。存储所有Region的寻址入口。实时监控Region server的状态，将Region server的上线和下线信息实时通知给Master。存储HBase的schema包括有哪些表，每个表有哪些列。

（3）Master

为Region server分配Region。负责Region server的负载均衡。发现失效的Region server并重新分配其上的Region。GFS上的垃圾文件回收。处理schema更新请求。

（4）Region server

Region server维护Master分配给它的Region，处理对这些Region的IO请求。Region server负责切分在运行过程中变得过大的Region。

我们走马观花般地过了一遍Hadoop主要的技术和产品，当然Hadoop体系包括的还有很多，我们只是描述了它最主要的部分，图1-9是Hadoop产品的生态图，有十多个，估计需要工程师们挑灯苦学了。

图1-9　Hadoop产品生态环境

1.3 Fourinone的产生背景

1.使用Hadoop时碰到的问题

笔者最开始尝试大数据并行计算分析是为了解决淘宝网的秒杀作弊问题。秒杀曾经是最成功的电商促销活动，往往在短时间内造成平时很多倍的销量，当然这也给系统造成很大的压力，但是这些都不是问题，关键是出现了像蟑螂一样的东西，怎么都灭不掉，那就是秒杀器。

当市场上不断出现秒杀器后，想用手工秒杀商品越来越难了，秒杀活动几乎全部被秒杀器软件垄断。于是买家开始大量抱怨不想参加秒杀了，卖家也不想发货给使用秒杀器作弊的人，甚至社会上流言淘宝网搞虚假秒杀，严重扰乱了市场秩序。

虽然技术人员也尝试屏蔽秒杀器作弊，但是一直收效甚微，比如尝试把验证码设计得很复杂，比如多加一些验证参数。写过秒杀器软件的人都明白，这样的防范把自己折磨得很辛苦，但是对秒杀器是不起作用的，因为秒杀器实际上是一个复杂的模拟HTTP交互的网络通信软件，它反复跟网站服务器交互HTTP报文，如果发现报文参数不够，会不断变更报文直到通过，因此只要秒杀器写得足够好，理论上就无法屏蔽它。

因此，唯一的办法是要深入分析HTTP报文本身的差异性。秒杀器和浏览器制造的报文是有区别的，这个区别或者大或者小，都是可以通过技术手段区分出来的，这是一个识别模型。由于每日大型、小型秒杀的订单数量太大，为了在短时间内识别出每个订单报文是否作弊，我们考虑用并行计算多机完成，为了完成这个技术任务，我们首先想到了用Hadoop。

按照Hadoop Map/Reduce框架的开发步骤，我们要实现Map和Reduce接口，取出每行，执行一段逻辑，打包好，启动jobtracker……我们学会了按照这个步骤开发后，尝试套用Map/Reduce框架会发现一些的问题，比如：

1）HTTP协议报文一个请求占多行，各行之前有一定逻辑关系，不能简单以行拆分和合并。

2）复杂的中间过程的计算套用Map/Reduce不容易构思，如大数据的组合或者迭代，多机计算并不只有拆分合并的需求（请参考2.1.11节）。

3）Hadoop实现得太复杂，API枯燥难懂，不利于程序员迅速上手并驾驭。

4）Map/Reduce容易将逻辑思维框住，业务逻辑不连贯，容易让程序员在使用过程中总是花大量时间去搞懂框架本身的实现。

5）在一台机器上未能很直接看出并行计算优势。

6）比较难直观设计每台计算机干哪些事情，每台计算机部署哪些程序，只能按照开发步骤学习，按照提供方式运行。

7）没有一个简单易用的Windows版，需要模仿Linux环境，安装配置复杂。

2.抽取一个简化的并行计算框架

我们从秒杀作弊分析延伸出了想法，提取和归纳了分布式核心技术，建立了一个简化的并行计算框架用于业务场景需要。我们的思路有以下几点：

1）对并行计算Map/Reduce和forkjoin等思想进行对比和研究。

2）侧重于小型灵活的、对业务逻辑进行并行编程的框架，基于该框架可以简单迅速建立并行的处理方式，较大程度提高计算效率。可以在本地使用，也可以在多机使用，使用公共内存和脚本语言配置，能嵌入式使用。

3）可以并行高效进行大数据和逻辑复杂的运算，并对外提供调用服务，所有的业务逻辑由业务开发方提供，未来并行计算平台上可以高效支持秒杀器作弊分析、炒信软件分析、评价数据分析、账务结算系统等业务逻辑运算。

4）可以试图做得更通用化，满足大部分并行计算需求。

下一章我们顺着这个思路深入解析并行计算模式的设计过程。