O'REILLY®

Broadview®
www.broadview.com.cn

用Mesos框架构建分布式应用

Building Applications on Mesos

利用弹性的、可扩展的分布式系统

[美] David Greenberg 著
崔婧雯 译

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

O'REILLY®

用Mesos框架构建分布式应用

Building Applications on Mesos

【美】David Greenberg 著

崔婧雯 译

電子工業出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

Apache Mesos是先进的集群管理器，既可以作为灵活的部署系统，也可以作为强大的执行平台。它不仅为分布式应用程序提供了良好的资源隔离，而且突破性地实现了资源的灵活共享，极大地提高了资源的整体利用率。

本书深入浅出，首先介绍了Mesos的基础知识，随后重点介绍Mesos的两种开源框架（Marathon和Chronos）。以实际程序样例为线索，一步步讲解如何配置，如何交互，以及如何构建深度集成。接着详细介绍如何为Mesos构建自定义的框架，如何构建核心Mesos API。最后深入研究Mesos的一些高级特性，比如和Docker的集成、其内部架构，以及一些最先进的API，包括数据库的持久化磁盘管理和框架预约系统。

版权贸易合同登记号 图字：01-2016-1971

图书在版编目（CIP）数据

用Mesos框架构建分布式应用 /（美）大卫 · 格林伯格（David Greenberg）著；崔婧雯译. —北京：电子工业出版社，2017.1
书名原文：Building Applications on Mesos
ISBN 978-7-121-30677-8

Ⅰ. ①用… Ⅱ. ①大… ②崔… Ⅲ. ①数据处理软件 Ⅳ. ①TP274

中国版本图书馆CIP数据核字(2016)第311330号

责任编辑：徐津平
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：9.25　字数：175千字
版　　次：2017年1月第1版
印　　次：2017年1月第1次印刷
定　　价：55.00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。

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始,O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来,而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者,O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来了革命性的“动物书”;创建了第一个商业网站(GNN);组织了影响深远的开放源代码峰会,以至于开源软件运动以此命名;创立了Make杂志,从而成为DIY革命的主要先锋;公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议集聚了众多超级极客和高瞻远瞩的商业领袖,他们共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择,O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务还是面授课程,每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列(真希望当初我也想到了)非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人,他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了:‘如果你在路上遇到岔路口,走小路(岔路)。’回顾过去Tim似乎每一次都选择了小路,而且有几次都是一闪即逝的机会,尽管大路也不错。”

——Linux Journal

前言

本书使用的排版约定

本书使用如下排版约定：

斜体（*Italic*）

表示新概念、URL、邮箱地址、文件名和文件扩展名。

等宽字体（`Constant width`）

用于程序列表，并且在正文内指代程序组件，比如变量或者功能名称、数据库、数据类型、环境变量、声明和关键字。

等宽斜体（*`Constant width bold`*）

表示应该由用户提供的值或者由上下文确定的值所替代的部分。

书中切口处的“▭”表示原书页码。

该图标表示技巧或建议。

该图标表示一般注释。

该图标表示警告或者注意事项。

Safari® 在线书籍

Safari 在线书籍是按需数字图书馆，以书籍和视频的格式提供来自技术和商业领域世界领先行业专家的专业内容。

技术专家、软件开发人员、web 设计人员以及商业和创作专业人士使用 Safari 在线书籍作为其搜索、解决问题、学习和认证培训的首要资源。

Safari 在线书籍为企业、政府、教育部门和个人提供一系列购买计划和定价。

会员可以在全搜索数据库里访问到上千本书籍、培训视频和正式发表前的草稿，这些资源来自 O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 等众多出版商。想要了解 Safari 在线书籍的更多信息，请访问网站。

如何联系我们

请将对本书的评价和发现的问题通过如下地址告知出版者。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

我们提供了本书网页，上面列着勘误表、示例和其他信息。请通过 http://bit.ly/building-applications-on-mesos 访问该页。

要给出本书意见或者询问技术问题，请发送邮件到 bookquestions@oreilly.com。

更多有关书籍、课程、会议和新闻的信息，请见网站 http://www.oreilly.com。

在 Facebook 找到我们：http://facebook.com/oreilly

在 Twitter 上关注我们：http://twitter.com/oreillymedia

在 YouTube 上观看：http://www.youtube.com/oreillymedia

鸣谢

本书的写作和出版花费了大量工作，没有大家的帮忙和支持，就没有本书最终的出版面世。

首先，感谢 Brain Foster 以及 O'Reilly 团队，他们为这本书的出版做出了很大贡献。

还要感谢我所任职的公司 Two Sigma，它给了我撰写此书的时间和支持。

感谢 Matt Adereth、Adam Bordelon、Niklas Nielsen 和 David Palaitis 的反馈和审查，这帮助大幅提高了本书的质量。

最后感谢我的妻子 Aysylu Greenberg，感谢她在撰写本书过程中给予我的爱和支持。

目录

第1章

Mesos介绍

让我们坐上时光机，回溯到1957年。那时，使用晶体管的计算机刚刚开始出现在大学和实验室里。不过问题是，每台计算机一次只能给一个人使用。因此，大家使用纸质签到表来预约上机的时间段。因为计算机比笔和纸强大很多，因此使用计算机的需求激增。同时，计算机又非常昂贵，如果大家不能充分利用自己的预约时间，就会浪费掉上千美元！幸运的是，在那时，操作系统的概念或多或少已经开始孕育了。一个名为John McCarthy的天才，他发明了LISP，想到一个伟大的主意——能否让所有用户都能将他们的工作提交给计算机，然后计算机自动在很多不同的工作间共享CPU资源呢？

作业成为程序

现在称为应用或者程序的东西以前称为作业（job）。在shell里仍然能够看到作业这个术语，如果将某个进程放到后台，就可以使用命令jobs来查看shell启动的所有程序。

一旦可以在作业间共享单台机器，就无须人工使用签到表来预约上机时间了——现在，每个人都可以很轻松地使用机器并且共享，因为机器能够按照所配置的配额，优先级，或者完全均等（如果确实需要）地执行。

飞速前进到2010年：随着网络化数据传输和存储费用的降低，这时已经可以将收集到的所有信息全都储存起来了。要处理所有这些数据，很可能需要使用Storm（一种分布式实时数据处理系统）和Hadoop。因此，你可能拥有很多机器：几台运行Hadoop JobTrack和Storm Nimbus（每台都有自己独特且精细的配置），几台运行HDFS NameNode和Secondary NameNode，还有15台机器上安装Hadoop TaskTrackers和HDFS DataNodes，10台机器用来运行Storm Supervisor。这时就已经需要管理并且购买30台机器。但是，如果想将其中5台Hadoop机器改成Storm worker，会非常困难，因

为需要将 Hadoop 机器彻底重新预配为 Storm 机器——大量实践证明，这可不像所期望的那么容易。

继续前进到 2011 年：Berkeley AMP 实验室正在起步。这正是现在给业界带来 Spark、Tachyon 和很多其他分布式可扩展系统的组织。该实验室的一个项目称为 Mesos：它意图成为能够在很多独立应用程序，比如 Hadoop 和 MPI 之间，共享计算集群的平台。在 Mesos 的论文[1]里，作者展示了所达到的卓越成果：可以在很多不同应用程序之间，包括 MPI 和 Hadoop，共享由普通计算机构成的单一集群的硬件资源。Mesos 高效地解决了集群里重新配置的问题。当然，像 Google 和 Yahoo！这样的公司对 AMP 实验室很感兴趣，因为他们也在致力于解决类似问题。Twitter 公司的团队在此领域更进一步：他们意识到 Mesos 解决了一直令他们挣扎痛苦的关键问题——如何高效使用数据中心里的大量机器——因此，他们成功雇佣了该论文的第一作者，也是 Mesos 的架构师，博士候选人 Ben Hindman，在 Twitter 公司里帮助解决这一问题。接下来的几年里，Mesos 从一个纯粹的研究项目发展成为在 Twitter 和很多其他公司里支撑成千上万台服务器的核心基础架构。

至此，大家应该对 Mesos 有所了解了（从历史角度和上下文里），一定还想知道 Mesos 到底是什么（从技术角度而言）。Mesos 是一种系统，帮助用户治理计算集群或者数据中心所有不同的机器，将每台机器当作单独的逻辑实体。使用了 Mesos 后，某个应用程序独占一组固定机器的问题就迎刃而解了。用户可以轻松调度哪个软件运行在哪些机器上，甚至可以将哪些软件运行在哪些机器上的选择权委托给 Mesos，让 Mesos 来替用户做这个决定，使得用户可以有更多时间和精力解决别的（更有意思并且更加重要的）问题。比如，在我的公司里，Mesos 帮助我们快速实践新的分布式系统，比如 Spark 和 Storm。

3 如何使用 Mesos

从最简单的角度看，Mesos 是跨集群管理 CPU、内存以及其他资源的编排平台。Mesos 使用容器化技术，比如 Docker 和 Linux Container（LXC），来达到上述目的。但是，Mesos 在此之上提供了更多的功能——它提供了实时 API，可以用来和集群交互并且辅助开发。

很多人想知道 Mesos 在其技术堆栈里所处的正确位置。Mesos 是部署基础架构的一部分吗？应该由基础架构团队管理吗？或者 Mesos 是一种应用程序开发平台，像 Heroku 那样？可能 Mesos 的确应该属于应用程序团队……不过，这些问题的答案并不直接，因

1 虽然大多数人认为第一篇论文是 Ben Hindman、Andy Konwinski、Matei Azharia 等人撰写的 NSDI 2011 论文 *Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center*，但是其实 Mesos 最初发表在 2009 年的 *Nexus: A Common Substrate for Cluster Computing* 一文里，由和上文相同的作者撰写，当时称为 Nexus。

为 Mesos 所提供的功能跨越了 Infrastructure as a Service（IaaS，基础架构即服务）和 Platform as a Service（PaaS，平台即服务），以便达到系统级别上更高的效率。一方面，Mesos 是基础架构：它是用户部署 Hadoop 和 Storm 以及其他业务所需 PaaS 软件的平台。另一方面，假定用户尝试开发一种能够处理大规模计算的应用程序，类似 Hadoop 或 Spark。这时，Mesos 则又成为用户用来构建这些程序的平台。因此，与其将 Mesos 当作黑盒子，不如尝试深入研究其 API 并且学习如何为其做开发。

实际上，最好将 Mesos 当作服务于不同目标的综合体。它让用户在开发和部署应用程序时都无须再关注单台机器。它允许用户将集群看成一个单一的大型资源组。Mesos 帮助用户更为敏捷地测试并且部署新的分布式系统，当用户需要在这些流程里引入开发即运维（DevOps）团队时，能够更加快速且高效地提供部署内部应用程序的平台。对于小型项目而言，Mesos 能够容纳很多不同的系统——比如像 Ansible、Chef、Puppet 这样的部署工具现在就能够降级到基本角色，从其中引导 Mesos。Mesos 是 DevOps 的终极工具，彻底模糊了应用程序运行 API 和应用程序部署之间的界线：Twitter 只有三个运维人员，管理着成千上万个节点的 Mesos 集群。

本书探讨 Mesos 是如何完美扮演这两种不同角色的：部署系统和执行平台。

Mesos 作为部署系统

可以把 Mesos 看成一个小型部署系统。首先一起看看如今典型的部署系统，比如 Ansible 和 Chef。要使用这些工具，用户需要编写一系列任务，这些任务需要在集群里的不同机器上执行。任务可以修改文件，安装程序，并且和监管系统（比如 Supervisor 和 SystemD）交互。任务组可以组合成“角色”和“recipe”，代表更高级别的概念，比如搭建一个数据库或者 web 服务器。最终，这些任务组组成“inventory”，或者主机集，可以根据规范在其上完成配置。

这些工具比 Bash、Perl 和 SSH 有用得多，但是，它们都带有一些根本的限制。一方面，它们允许用结构化，可理解的格式编写配置信息。它们借鉴了软件开发领域的最佳实践——源码控制，封装，以及通过库函数和插件重用代码——并且将这些最佳实践扩展到系统管理领域。但是，它们最根本的设计是让一组机器遵循某个固定的配置。这样设计是刻意的：一旦用户运行了 Ansible 或者 Puppet 的配置，就希望集群处在相同的，已知的一致状态。但是，如果希望集群能够动态重新分配资源，或者根据多种外部因素，比如当前负载等，动态地更改配置，那么这些工具就爱莫能助了。

从这个角度可以认为 Mesos 是一种部署系统。分布式应用程序能够融入 Mesos 框架，从而运行在 Mesos 集群里。Mesos 不会强制用户使用某个固定的配置描述，每种框架本质

上是一种角色或者 recipe：框架包含安装、启动、监控，以及使用某种应用程序的所有逻辑。很多框架本身也是平台，当很多应用程序遵守相同的通用部署模式（比如 web 服务器）时，那么单个框架就可以管理所有这些应用程序（比如 Marathon，一种针对无状态服务的 Mesos 框架）。Mesos 的魔力在于，既然框架是一个运行着的程序，那么它就能够动态做出决策，应对工作负载以及其他集群条件的变化。比如，一个框架能够观测到有多少可用资源，然后可以改变其执行策略，以便能够基于集群条件更好地执行。因为部署系统持续运行着，它能够实时监测并且适应发生的故障，在服务器故障时自动启动新的服务器。Mesos 框架比很多传统部署系统所使用的静态描述要强大得多。

因此，可以认为 Mesos 能够代替 Ansible 或者 Chef 所提供的大多数功能。用户仍然需要一个传统的配置管理器来引导 Mesos 集群，但是，Mesos 为托管的应用程序提供了更加强大的平台：框架能够自动并且动态地适应故障机器，改变工作负载，让运维人员能够安心处理别的工作。

5 Mesos 作为执行平台

还可以把 Mesos 当作托管应用程序的平台。你可能正在使用 Heroku 运行 web 服务，或者可能管理 Hadoop 集群来执行 MapReduce 作业。在这些场景里，Heroku 和 Hadoop 集群是 web 服务和 MapReduce 作业的平台，Mesos 集群也可以作为高级别应用程序的平台。但是如果你的系统已经在工作了，为什么还要向其中加入 Mesos 呢？因为 Mesos 提供了灵活性，并且降低了被新技术赶超的风险。使用了 Mesos 之后，启动 Spark 集群或者切换到更新的技术就变得很简单，只需要启动框架，看着它引导自身就可以了。

试想一下 Heroku/PaaS 场景：Marathon 能够可靠地启动应用程序，当有机器崩溃或者关机维护时，能够自动启动新实例。Mesos-DNS 和 HAProxy 则帮助管理负载均衡（见第 3 章）。但是一旦运行了 Mesos，为什么还要操心 Hadoop 集群的维护呢？可以通过 Myriad 启动 Mesos 上的 Hadoop，甚至也不用再操心 HDFS 的管理，因为还有针对 HDFS 的框架（Mesos HDFS）。

综上，可以将 Mesos 当作一种执行平台：与其购买第三方的 PaaS 解决方案，并且为每种大数据分析技术都创建出定制集群，还不如使用单个 Mesos 集群。使用 Mesos 集群作为基础之后，用户可以在现在，或者当新需求新技术出现时，轻松启动所需的任何框架，来提供所需的任何功能。

本书是如何组织的

这里介绍本书的其他部分是如何组织的。首先，我们会一起学习 Mesos 本身，会介绍

Mesos 架构及其框架，学习如何跨集群分配资源。还会深入研究所需配置，并且介绍实用的命令行设置，来调优集群的整体性能和工作负载。

之后，我们会一起学习已有的两种开源 Mesos 框架——Marathon 和 Chronos，它们提供了应用程序托管和作业调度的功能。本书会一步步讲解如何配置它们，如何交互，以及如何构建深度集成。学习完这些框架之后，读者就能够掌握一些 Mesos 上的可靠性高且可用性强的通用应用程序架构，比如 web 服务和数据流水线。

本书后面的部分会详细介绍如何为 Mesos 构建一个自定义的框架。一起学习用 Java 实现样例作业调度框架的流程中所使用到的 API。在开发该框架的过程中，会深入探讨如何构建核心 Mesos API，有哪些常见的陷阱以及应该如何规避，如何决定是构建新框架还是采用已有系统。

本书的最后，会深入研究 Mesos 的一些高级特性，比如和 Docker 的集成，其内部架构，以及一些最先进的 API，包括数据库的持久化磁盘管理，以及框架预约系统。

本章小结

Mesos 本质上是操作系统：它管理计算机，将其统一治理成单一的逻辑单元。企业使用 Mesos 是因为它的可靠性、灵活性和高效性。Mesos 集群只要很少的管理员，就能保证大规模机器群运作良好。

框架是运行在 Mesos 上的分布式应用程序，对于它们而言，Mesos 提供了基础架构和执行服务。框架可以是应用程序，比如 Hadoop 或 Storm，也可以是部署系统，比如 Ansible 或 Chef。Mesos 和框架携手并进，将 Mesos 集群里的机器片分配给不同的角色。这样的能力——将某个框架所独占的机器群壁垒打破——正是 Mesos 带来更高运维效率的秘密所在。

本章介绍了 Mesos 的高层级视野，下一章我们开始深入 Mesos 概念，并且在实践中开始 Mesos 的学习！