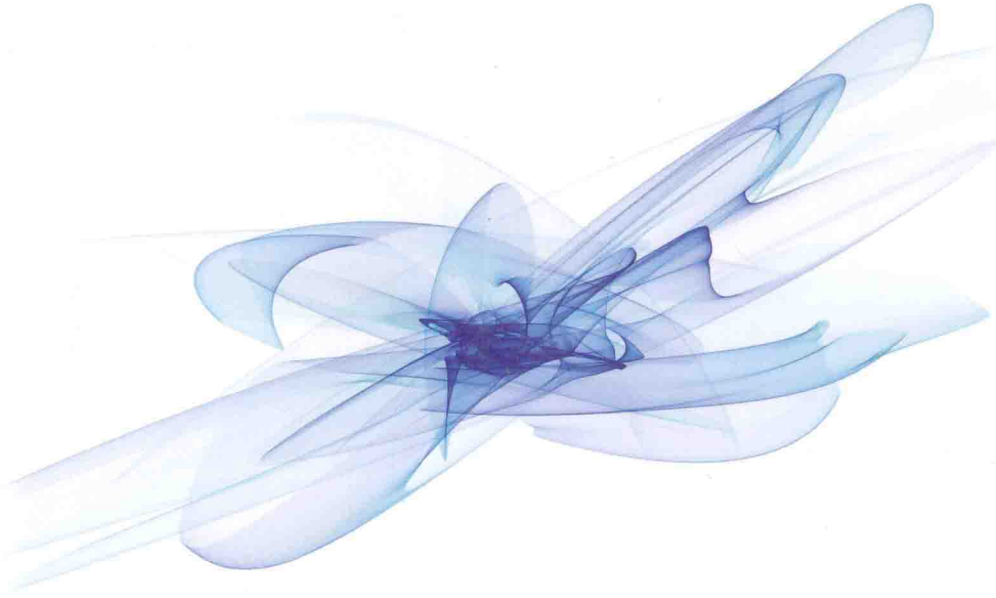
华章科技

[PACKT] PUBLISHING

从多个角度全面讲解Storm实时数据处理技术和最佳实践，为快速掌握并灵活应用Storm提供实用指南

从实际问题出发，系统介绍Storm的基本应用、多语言特性、完整业务系统实现和产品交付的最佳实践方法；从产品持续交付角度，分析并实践集成、测试和交付的所有步骤

大数据技术丛书



Storm Real-Time Processing Cookbook

Storm实时数据处理

（澳） Quinton Anderson 著

卢誉声◎译

机械工业出版社
China Machine Press

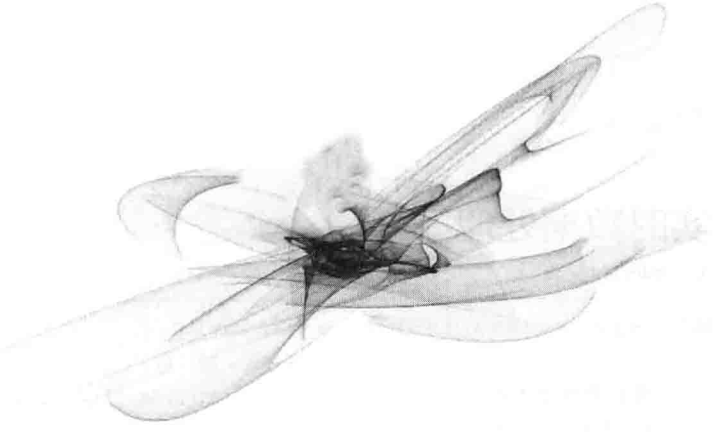
大数据技术丛书

Storm Real-Time Processing Cookbook

Storm实时数据处理

（澳） Quinton Anderson 著

卢誉声◎译



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Storm 实时数据处理 /（澳）安德森（Anderson，Q.）著；卢誉声译．—北京：机械工业出版社，2014.6
（大数据技术丛书）
书名原文：Storm Real-Time Processing Cookbook

ISBN 978-7-111-46663-5

I. S… II. ① 安… ② 卢… III. 数据处理软件 IV. TP274

中国版本图书馆 CIP 数据核字（2014）第 103057 号

本书版权登记号：图字：01-2013-7570

Storm 实时数据处理

[澳] Quinton Anderson 著

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：高婧雅 责任校对：殷 虹
印 刷：三河市宏图印务有限公司 版 次：2014 年 6 月第 1 版第 1 次印刷
开 本：186mm×240mm 1/16 印 张：12.75
书 号：ISBN 978-7-111-46663-5 定 价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

华章计算机
HZBOOKS | Computer Science and Technology

译 者 序

随着互联网业务数据规模的急剧增加，人们处理和使用数据的模式已然发生了天翻地覆的变化，传统的技术架构越来越无法适应当今海量数据处理的需求。MapReduce、Hadoop 以及一些相关技术的出现使得我们能处理的数据量比以前要多得多，这类技术解决了我们面对海量数据时的措手不及，也在一定程度上缓解了传统技术架构过时的问题。

但是，随着业务数据规模的爆炸式增长和对数据实时处理能力的需求越来越高，原本承载着海量数据处理任务的 Hadoop 在实时计算处理方面越发显得乏力。原因很简单，像 Hadoop 使用的 MapReduce 这样的数据处理技术，其设计初衷并不是为了满足实时计算的需求。那么就目前来说，有没有什么行之有效的办法能简单地将 Hadoop 转换成实时计算系统呢？

这个问题的答案可能令人略感失望：没有。Hadoop 作为批处理系统，与实时处理系统在需求上存在着本质的区别。要做到实时性，不仅需要及时地推送数据以便处理，还要将数据划分成尽可能小的单位，而 HDFS 存储推送数据的能力已经远不能满足实时性的需求。另外，Hadoop 还存在配置、一致性和可伸缩性方面的问题。

那么问题来了，怎么才能构建出一个可靠的实时处理系统呢？

答案是 Storm。

从整体架构上看，Storm 和 Hadoop 非常类似。Storm 从架构基础本身就实现了实时计算和数据处理保序的功能，而且从概念上看，Storm 秉承了许多 Hadoop 的概念、术语和操作方法，因此如果你对 Hadoop 非常熟悉，那么将 Storm 与 Hadoop 集成也不是什么难事。通过 Storm Trident 提供的高级抽象元语，你可以像 Hadoop Cascading 简化并行批处理那样简化并行实时数据处理。

我本人在实时计算服务器开发方面具有一定的经验，对大数据处理解决方案十分感兴趣，也对相关技术有一些了解，并且有幸承担了本书的翻译工作。本书从多个角度解析了有关 Storm 的最佳实践，无论是从最基本的应用、多语言特性、完整业务系统的实现，还是最终交付至产品环境，本书都给出了翔实的最佳实践方法。不仅如此，本书还从产品持续交付的角度，分析并实践了集成、测试和交付的所有步骤，让人受益匪浅。相信你会和我一样，通过阅读本书，能对 Storm 本身及构建成熟实时计算系统的方法有更进一步的了解。

翻译本书对我来说其实也是一个渐进学习的过程，书中提及了大量方法和工具的应用，

让我从零散的概念分支逐渐有了清晰的知识主干，形成了系统的知识点。通过对本书的翻译，我对这个领域的内容有了更加深刻的理解。这也是我翻译此书最大的收获之一。在翻译过程中，我不仅查阅了大量国内外的相关资料，还与原书作者进行了深入的沟通，力求做到专业词汇准确权威，书中内容正确。

在翻译过程中我得到了很多人的帮助，特在此一一感谢。首先是我的家人，你们是我学习和前进的动力。感谢鲁昌华教授，在我的成长道路上给予了很大的支持和鼓励。感谢我在思科系统（中国）研发的同事们，在我的学习工作中给予了很大帮助。感谢我的好友金柳颀，感谢你在翻译过程中的通力合作以及在技术问题上的共同探讨。还要感谢机械工业出版社华章公司对我的信任与支持。

现在我怀着期盼和忐忑的心情将这本译著呈献给大家，我渴望得到您的认可，更渴望和您成为朋友。如果您有任何问题和建议，请与我联系（samblg@me.com）。让我们一起探讨，共同进步。

卢誉声

前　言

开源已经在许多方面从根本上改变了软件的原有面貌。在很多应用环境中，人们都会争论使用开源带来的好处和坏处，主要体现在支持、风险以及总体拥有成本等方面。开源在某些领域比其他领域流行，比如在研究机构中就比在大型金融服务提供商中应用得多。在某些新兴领域，比如 Web 服务供应商、内容供应商以及社交网络等，开源软件占据主导地位。其原因是多方面的，其中成本是一个非常大的因素。怎么说呢？如果方案要上升到网络规模，那么一般会应用“大数据”解决方案，以期获得更好的效果。凭借极佳的可用性，这些解决方案每秒处理数百万条请求，同时通过各式各样的服务为客户提供定制体验。

这种规模的系统设计需要我们用不同的方式思考问题，采用不同的架构解决方案，而且要了解什么时候应该接受系统的复杂性以及什么时候应该避免这种复杂性。作为行业中人，我们应该掌握这种大规模批处理系统的设计方法。由 MapReduce、Bulk Synchronous Parallel 以及其他计算范式衍生而来的大规模计算集群已经广泛实施，并且得到了很好的理解。开源掀起了创新浪潮，并推动其发展，而即使是顶级软件开发商也只能努力将 Hadoop 集成到自家的技术架构中，更别提试图与开源竞争了。

然而，客户是永不满足的，他们想要更多的数据、更多的服务、更多的价值、更多的便利，而且现在就要，并希望成本更低。随着数据量的增加，对实时响应时间的需求也在提高。专注于实时性、规模化的计算平台新时代已经来临。这带来了许多新的挑战，不管是理论上还是实践上都具有挑战性。

本书将帮助你掌握一个平台：Storm 处理系统。Storm 处理系统是一个开源的、实时的计算平台，最初由社交分析公司 Backtype 的 Nathan Marz 编写，后来被 Twitter 收购，并作为开源软件发布。从那时起，该平台茁壮成长，目前已经成长为一个日益扩大的开源社区：拥有众多用户、贡献者以及产品页面上的成功案例。写这篇前言的时候，该项目在 GitHub 上拥有超过 6000 个星、3000 多个 Twitter 粉丝，每个节点每秒可以完成超过 100 万次交易，而且有近 80 名相关用户在使用 Storm 产品实例。这些数字极其可观。此外，在本书的结尾你会发现，无论你用哪种语言（与你的思维方式以及解决方案交付方式相一致的语言）研发系统，以 Storm 平台为基础都会非常愉快。

本书通过一系列实例指导你学习 Storm。书中的例子源于实战用例，并随着内容的展开

介绍各种概念。此外，本书旨在围绕 Storm 技术促进 DevOps [⊖]实践，使读者能够开发 Storm 解决方案，同时可靠地交付具有价值的产品。

Storm 处理系统简介

开源项目普遍存在缺乏文档的问题，但 Storm 并不存在这个问题，其项目有着高质量且编写良好的文档，由活跃的用户社区提供支持。本书并不想照本宣科，而是通过丰富的例子对现有资料进行补充，在必要的时候辅以概念和理论的探讨。强烈建议读者在阅读第 1 章之前，花些时间阅读 Storm 的入门文档，具体有以下几个网页：

- https://github.com/nathanmarz/storm/wiki/Rationale；
- https://github.com/nathanmarz/storm/wiki/Concepts；
- https://github.com/nathanmarz/storm/wiki/Understanding-the-parallelism-of-a-Storm-topology。

章节介绍

第 1 章 介绍搭建 Storm 本地开发环境的方法，包括所需工具和推荐的开发流程。

第 2 章 教你创建一个日志流处理解决方案、基本的统计面板和日志搜索功能。

第 3 章 介绍 Trident，Trident 是基于 Storm 元语进行实时计算的一类更高级的处理数据流的抽象元语，支持高效的企业数据管道。

第 4 章 介绍实现分布式远程过程调用的方法。

第 5 章 介绍在不同的编程语言中实现 Topology 的方法，并在现有技术基础上增加一些新的技术。

第 6 章 介绍 Storm 与 Hadoop 的集成方法，创建一个完整的 Lambda 架构。

第 7 章 简要介绍机器学习，讨论多种基于 Storm 实时处理项目的实现方法。

第 8 章 介绍搭建持续交付流水线和可靠部署 Storm 集群到产品环境的方法。

第 9 章 介绍多种在 Amazon 云计算平台中自动化配置 Storm 集群的方法。

阅读前提

本书采用 Ubuntu 或 Debian 操作系统作为基本的开发环境。第 1 章会介绍如何配置其余

⊖ DevOps是一组过程、方法和系统的统称，用于促进开发、技术运营和质量保障部门之间的沟通、协作与整合。DevOps的目的是实现业务敏捷，它将敏捷开发扩展到了运维上，通过快速、反应灵敏且稳定的业务运维，使其能够与开发过程的创新保持同步，实现真正的IT融合。——译者注

必备的工具。如果你不能使用 Ubuntu 作为开发用的操作系统，任何基于 *Nix 的操作系统都是可以的，因为书中所有示例都以 bash 命令行界面为基础构建。

读者对象

如果你想学习实时处理技术，或想了解通过 Storm 实现实时处理的方法，那么本书非常适合你。本书假定你是一名 Java 开发者，要是你还具备 Clojure、C++ 和 Ruby 的开发经验，那将是锦上添花的事情，不过那并不是必需的。如果你还了解一些 Hadoop 或类似的技术就更好了。

本书版式约定

在本书中，读者会发现针对不同信息类型的文本样式。下面是这些样式的示例和解释。

代码段版式如下所示：

```
<repositories>

    <repository>
      <id>github-releases</id>
      <url>http://oss.sonatype.org/content/repositories
        /github-releases/</url>
    </repository>

    <repository>
      <id>clojars.org</id>
      <url>http://clojars.org/repo</url>
    </repository>

    <repository>
      <id>twitter4j</id>
      <url>http://twitter4j.org/maven2</url>
    </repository>
</repositories>
```

所有命令行输入和输出如下所示：

```
mkdir FirstGitProject
cd FirstGitProject
git init
```

下载本书的示例代码

访问 http://www.packtpub.com/，可以下载本书及你所购买的所有 Packt 图书的示例代码。如果你是从其他地方购买的本书英文版，可以访问 http://www.packtpub.com/support 并

注册，以便通过电子邮件取得示例文件。

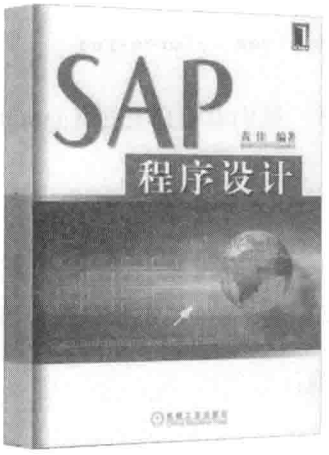
作者通过 Bitbucket 账户来维护开源版本的代码：https://bitbucket.org/qanderson。

致谢

感谢 Storm 社区，正是他们的努力构建出了开源社区中如此出色的平台，当然，还要特别感谢 Storm 的核心作者——Nathan Marz。

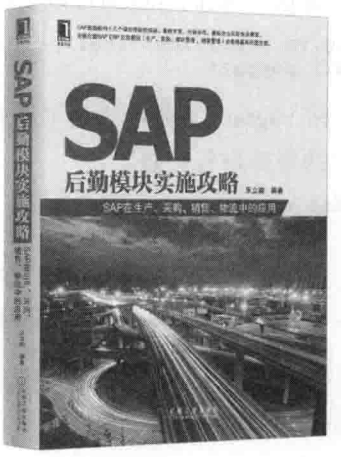
我要感谢我的妻子和孩子对我长时间写作本书和其他相关项目的宽容，感谢这段时间他们给予我的帮助，我爱他们。此外，我还要感谢所有参与本书审校工作的朋友。

推荐阅读



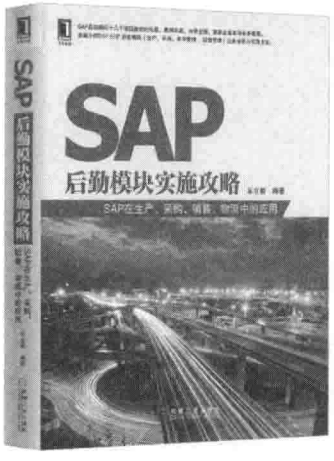
SAP程序设计领域畅销书，多年来畅销不衰。

详细讲解SAP系统开发实施过程中的各个环节及其设计方法，侧重于系统技术实现细节，深入浅出地介绍了SAP系统、ABAP语言以及ABAP工作台工具，同时涉及SAP系统结构知识。



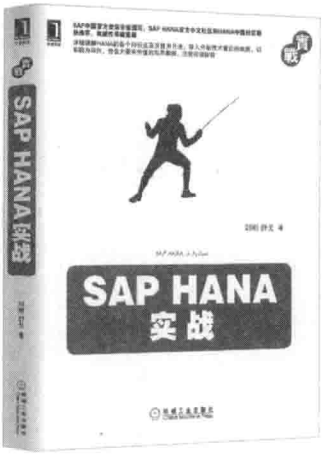
SAP高级顾问数十个项目的经验总结，不仅从宏观上系统介绍了各个SAP后勤模块的功能和它们之间的关系，而且从微观上详细讲解了各个模块的使用方法。

包含大量来自实企业的实际案例，涉及生产计划与执行管理、采购与库存管理、销售与分销管理、运输管理和变更管理等环节，实用性极强。



资深SAP HCM高级顾问十余年工作经验总结，对初学者如何成长为金领顾问给出了详细的指导。

从解决方案、项目管理和功能模块三个维度对SAP HCM进行了系统而详细的讲解，案例丰富，紧贴企业实际业务需求。



SAP中国官方资深专家撰写，SAP HANA官方中文社区和HANA中国社区联袂推荐，权威性毋庸置疑。

详细讲解HANA的各个知识点及其使用方法，深入分析技术背后的本质，以实践为导向，包含大量有价值的实用案例，注重阅读体验。

目　录

译者序
前言

第 1 章　搭建开发环境 / 1

1.1　简介 / 1
1.2　搭建开发环境 / 1
1.3　分布式版本控制 / 3
1.4　创建“Hello World”Topology / 6
1.5　创建 Storm 集群——配置机器 / 12
1.6　创建 Storm 集群——配置 Storm / 18
1.7　获取基本的点击率统计信息 / 23
1.8　对 Bolt 进行单元测试 / 31
1.9　实现集成测试 / 34
1.10　将产品部署到集群 / 37

第 2 章　日志流处理 / 38

2.1　简介 / 38
2.2　创建日志代理 / 38
2.3　创建日志 Spout / 40
2.4　基于规则的日志流分析 / 45
2.5　索引与持久化日志数据 / 49
2.6　统计与持久化日志统计信息 / 53
2.7　为日志流集群创建集成测试 / 55
2.8　创建日志分析面板 / 59

第 3 章　使用 Trident 计算单词重要度 / 71

3.1　简介 / 71
3.2　使用 Twitter 过滤器创建 URL 流 / 71
3.3　从文件中获取整洁的词流 / 76
3.4　计算每个单词的相对重要度 / 81

第 4 章　分布式远程过程调用 / 85

4.1　简介 / 85
4.2　通过 DPRC 实现所需处理流程 / 85
4.3　对 Trident Topology 进行集成测试 / 90
4.4　实现滚动窗口 Topology / 95
4.5　在集成测试中模拟时间 / 98

第 5 章　在不同语言中实现 Topology / 100

5.1　简介 / 100
5.2　在 Qt 中实现多语言协议 / 100
5.3　在 Qt 中实现 SplitSentence Bolt / 105
5.4　在 Ruby 中实现计数 Bolt / 108
5.5　在 Clojure 中实现单词计数 Topology / 109

第 6 章　Storm 与 Hadoop 集成 / 113

6.1　简介 / 113
6.2　在 Hadoop 中实现 TF-IDF 算法 / 115
6.3　持久化来自 Storm 的文件 / 121
6.4　集成批处理与实时视图 / 122

第 7 章　实时机器学习 / 127

7.1　简介 / 127
7.2　实现事务性 Topology / 129
7.3　在 R 中创建随机森林分类模型 / 134
7.4　基于随机森林的事务流业务分类 / 143

7.5 在 R 中创建关联规则模型 / 149
7.6 创建推荐引擎 / 152
7.7 实时在线机器学习 / 157

第 8 章 持续交付 / 162

8.1 简介 / 162
8.2 搭建 CI 服务器 / 162
8.3 搭建系统环境 / 164
8.4 定义交付流水线 / 166
8.5 实现自动化验收测试 / 170

第 9 章 在 AWS 上部署 Storm / 177

9.1 简介 / 177
9.2 使用 Pallet 在 AWS 上部署 Storm / 177
9.3 搭建虚拟私有云 / 181
9.4 使用 Vagrant 在虚拟私有云上部署 Storm / 189

第 1 章　搭建开发环境

1.1　简介

本章将简要介绍 Storm 处理系统。这将涵盖所有你想知道的内容，从搭建你的开发环境到部署 Topology 时需要注意的操作关注事项，再到基本的质量实践，比如对 Storm Topology 进行单元测试和集成测试。在阅读完本章后，你将能够构建、测试和交付基本的 Storm Topology。

本书并不准备对 Storm 处理系统及其元语和架构进行理论介绍。我们假定你在阅读本书之前已经通过浏览如 Storm wiki 这样的在线资源了解了 Storm 的基本概念。

当系统在产品环境中能持续可靠地产生商业价值时，才能交付系统。为了实现这个目的，在开发 Storm Topology 时，必须始终考虑质量问题和操作注意事项。

1.2　搭建开发环境

开发环境涵盖了构建 Storm Topology 所需的各种工具和系统。虽说本书重点关注的是每个有技术侧重点的 Storm 交付，但需要指出的是，对于一个软件开发团队来说，无论使用集中式开发环境还是分布式开发环境，都需要更多的工具和流程来保证高效工作，而且不能仅仅局限于本书所讨论的内容。

无论是为了将来的开发工作，还是为了实现书中的例子，以下几类工具和流程都是快速搭建开发环境必不可少的：

- SDK
- 版本控制
- 构建环境
- 系统配置工具
- 集群配置工具

书中描述的配置和安装方法都基于 Ubuntu。不管怎么说，适用于 Ubuntu 的这些方法还是比较容易移植到其他 Linux 发行版中的。如果将这些方法应用于其他发行版时出现任何问题，你可以通过 Storm 邮件列表寻求支持：https://groups.google.com/forum/#!forum/storm-user。

环境变量是导致系统可操作性和可用性下降的“罪魁祸首”，如果部署环境与开发环境不同，那么环境变量很可能在这时引起大问题。所以尽可能让开发环境和部署环境保持一致。

1.2.1 实战

Step01 从 Oracle 网站（http://www.oracle.com/technetwork/java/javase/downloads/index.html）下载最新版本的 J2SE 6 SDK，并通过以下命令进行安装：

```
chmod 775 jdk-6u35-linux-x64.bin
yes | jdk-6u35-linux-x64.bin
mv jdk1.6.0_35 /opt
ln -s /opt/jdk1.6.0_35/bin/java /usr/bin
ln -s /opt/jdk1.6.0_35/bin/javac /usr/bin
JAVA_HOME=/opt/jdk1.6.0_35
export JAVA_HOME
PATH=$PATH:$JAVA_HOME/bin
export PATH
```

Step02 然后安装版本控制系统 Git：

```
sudo apt-get install git
```

Step03 接下来，安装构建系统 Maven：

```
sudo apt-get install mvn
```

Step04 安装 Puppet、Vagrant 和 VirtualBox，以便于对应用程序和环境进行配置：

```
sudo apt-get install virtualbox puppet vagrant
```

Step05 最后，你还需要安装一个集成开发环境（IDE）：

```
sudo apt-get install eclipse
```

自从 Sun 公司被 Oracle 收购以后，人们就一直在争论该使用哪种 Java SDK。虽说作者理解 OpenJDK 的必要性，但书中的例子都是在 Oracle JDK 下测试通过的。总的来说，OpenJDK 和 Oracle JDK 没有什么区别，只是 Oracle JDK 为了追求稳定而在功能方面相对滞后。

1.2.2 解析

对于任何 Java 开发工作来说 JDK 都是必不可少的。GIT 是近几年被广泛采用的开源分布式版本控制系统。稍后我们将简要介绍 GIT。

Maven 是一种广泛使用的“约定优于配置”的构建系统。Maven 包含了许多有用的功能，

比如项目对象模型（POM），POM 能够让我们以有效的方式管理库、依赖和版本。Maven 在互联网上拥有许多二进制仓库可供使用，所以我们可以直接用恰当的方式来维护二进制依赖项，然后打包部署 Topology。

由于 DevOps 和持续交付的不断发展，Puppet 系统已经被广泛用于为 Linux、其他一些操作系统和应用程序提供声明式服务器配置。Puppet 能够对服务器的状态和部署环境进行编程，这一点非常重要，因为这样就可以通过 GIT 版本控制系统管理服务器状态，而且对于服务器的修改都可以安全删除。这么做好处很多，包括可确定的服务器平均恢复时间（MTTR）和审计跟踪，说白了就是让系统更加稳定。这对实现持续交付来说至关重要。

Vagrant 是一个在开发过程中非常有用的工具，它能自动配置 VirtualBox 虚拟机，这对 Storm 处理系统这样的以集群为基础的技术来说尤为重要。为了测试一个集群，你必须构建一个真正的机群或配置多台虚拟机。Vagrant 能准确无误地在本地配置虚拟机。

虚拟机在 IT 运维和系统开发过程中能够大显身手，但需要指出的是，虽然我们都知道在本地部署的虚拟机性能不太理想，但其可用性却完全依赖于可用的内存容量。处理能力不是需要考虑的主要问题，因为现代处理器的利用率普遍严重偏低，尽管处理 Topology 时利用率会高一些。建议你的计算机至少保留 8GB 的内存容量。

1.3 分布式版本控制

传统版本控制系统都是集中式的。每个客户端都包含一份从当前版本签出的文件，而当前版本则取决于客户端使用的分支。所有历史版本都会存储在服务器上。这样的做法效果不错，不仅能让团队紧密协作，还能知道其他成员在做什么工作。

但集中式服务器存在一些较为明显的缺点，这让分布式版本控制系统的应用范围越来越广。首先，集中式服务器存在单点故障问题，如果服务器由于某种原因死机或无法访问，开发人员会因此难以继续工作。其次，如果服务器上的数据由于某种原因损坏或丢失，代码库的历史记录也就随之丢失。

基于以上两个原因，开源项目极大地推动了分布式版本控制的发展，但究其主要原因，还是分布式系统能帮助我们实现协作模型。开发人员可以在本地环境下用自己的方式进行工作，然后在方便的时候，一次性或多次将这些改动分布到一个或多个远程代码仓库。

除此之外，还有一个很明显的优势就是，由于每一个客户端都拥有代码仓库的完整镜像，因此会自然存在多个仓库备份。所以，如果任何一个客户端或服务器出现问题，都可以轻松恢复，实现数据还原。