Spring Batch 权威指南

[美] 迈克尔·T. 米内拉(Michael T. Minella) 著
张 坤 张 渊 译

清华大学出版社
北 京

北京市版权局著作权合同登记号　图字：01-2020-2330

Michael T. Minella
The Definitive Guide to Spring Batch：Modern Finite Batch Processing in the Cloud, Second Edition
EISBN: 978-1-4842-3723-6

图书在版编目(CIP)数据

Spring Batch 权威指南 / (美)迈克尔・T. 米内拉(Michael T. Minella)著；张坤，张渊译. —北京：清华大学出版社，2021.1

书名原文：The Definitive Guide to Spring Batch：Modern Finite Batch Processing in the Cloud, Second Edition

ISBN 978-7-302-56772-1

Ⅰ. ①S…　Ⅱ. ①迈…②张…③张…　Ⅲ. ①数据处理　Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字(2020)第 218041 号

责任编辑：王　军
封面设计：孔祥峰
版式设计：思创景点
责任校对：成凤进
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：26.25　　字　　数：702 千字
版　　次：2021 年 1 月第 1 版　　印　　次：2021 年 1 月第 1 次印刷
定　　价：98.00 元

产品编号：086011-01

译 者 序

提到数据处理，很多人可能会想到大数据平台上的批处理和流式处理。为了使用 Hadoop 生态，应用开发者可能会在层出不穷的“大数据概念”中举步不前，在眼花缭乱的组件面前望而却步。假如要处理的数据集不是那么“大”，处理也不是很复杂，那么对于 Java 开发者而言，有没有简单易用的框架和工具呢？Spring Batch 就是可能的答案。对于数据工程师而言，Spring Batch 未免有些鸡肋；但是对于 Java 应用开发者，尤其是有一定经验的 Java 应用开发者而言，Spring Batch 未尝不是易于上手且能够解决实际问题的实用工具。

本书将理论与实际相结合，介绍了使用 Spring Batch 读取输入、加工处理以及输出时的各种考虑因素和不同实现选项，贯穿本书的是一个对账单示例：根据客户数据和客户交易记录文件，生成汇总的对账单文件。在开发过程中，本书使用了轻量级的敏捷开发方法：使用用户故事描述需求，进行多次迭代，不断地完善功能。如果跳过其中的理论讲解，你就会发现，其实并不需要编写太多代码，就可以完成端到端的数据处理任务。仔细回顾，你甚至会发现，如果转换逻辑不多，那么大部分代码仅仅用于作业、步骤等的配置。Spring Batch 提供了常见的输入输出工具，这样开发人员就可以更多地关注于编写数据转换逻辑。在实际场景中，批处理程序可能无法满足性能要求，所以本书还介绍了单机环境中的性能调优方法，以及分布式环境中的批处理。本书最后简单介绍了不同层次的自动化测试。

希望读者在深入阅读各个章节的过程中，也能思考本书的组织逻辑以及各个技术决策背后的原因。这样在实现功能的同时，就能建立体系化的思考方式。

清华大学出版社的编辑老师，一直关注本书的翻译工作，期间给予我们极大的耐心和许多建设性的反馈意见，在此特别感谢。本书的作者 Michael T. Minella 也是一位热心人士，回答了我们在翻译过程中的不解之处。在翻译本书期间，我们经历了新冠肺炎疫情在国内的爆发和有效控制，人们的生活和工作都受到了巨大影响。在很长一段时间里，人们都居家办公。本书能够成功翻译，离不开家人在这段时间内给予的有力支持，正因为他们，我们才能在安心的环境中投入时间进行翻译和校对。

本书涉及大量的专业术语和技术实践，我们也竭尽所能让这本译作简单易懂。但毕竟水平有限，错误和失误在所难免。如有任何意见和建议，请不吝指正，不胜感激！

译 者

2020 年 6 月

作者简介

Michael T. Minella 是一位软件工程师、作家和演说家，拥有超过 18 年的专业经验。Michael 是 Pivotal 的软件工程主管，领导了 Spring Batch 和 Spring Cloud Task 项目，他也是 JSR-352(Java Batch)的专家组成员。另外，他还是一名 Java Champion 和 JavaOne Rockstar，曾在许多不同的 Java 国际会议上发表演讲。

在日常工作外，Michael 还在播客 OffHeap 上扮演“脾气暴躁的人”。他个人对信息安全话题(InfoSec)也很感兴趣。Michael 的爱好包括摄影和木工，他和爱人 Erica、孩子 Addison 生活在一起。

技术审稿人简介

Wayne Lund 在为埃森哲(Accenture)的全球架构群组工作时，担任首席技术架构师，是 Spring Batch 最初的创造者之一，他在 JavaOne 2007 大会上将 Spring Batch 交付给 Java 社区。埃森哲的全球架构群组专注于 OSS 项目，帮助客户采用 Spring 作为抽象 JEE 的首选平台，并偏爱轻量级框架。在 Spring 被 VMWare 收购之后，Wayne 加入了 vFabric 群组(现在是 Pivotal 服务的一部分)，该群组在销售渠道中支持 Spring Source、RabbitMQ、Gemfire 和其他开源的轻量级框架。Wayne 目前是 Pivotal Data Services 的咨询平台架构师，负责帮助提供启用了 Spring 的数据产品解决方案，包括 Spring Cloud Data Flow(Spring Cloud Stream、Spring Cloud Task 和 Spring Batch)以及使用 RabbitMQ 和 Kafka 的消息处理。

Felipe Gutierrez 是解决方案软件架构师，拥有墨西哥蒙特雷大学蒙特雷分校计算机科学专业的学士和硕士学位。Felipe 有超过 20 年的 IT 经验，他为多个垂直行业的公司开发项目，比如零售、医疗保健、教育和银行等。Felipe 目前是 Pivotal 的平台和解决方案架构师，擅长 Cloud Foundry PAS 和 PKS、Spring 框架、Spring 云原生应用、Groovy 和 RabbitMQ 等技术。他曾在诺基亚、苹果、Redbox、高通等大公司担任解决方案架构师。Felipe 还是畅销书 *Spring Boot Messaging* 和 *Introducing Spring Framework* 的作者。

致　　谢

从我写第一本书到现在的这段时间里，我的生活发生了很大的变化，其中的很多变化都对我的职业生涯产生了实质性影响，这让我有机会撰写有关 Spring Batch 的第二本书。在此，我想利用本书的“致谢”部分，对一路帮助过我的各位人士表达我的感激之情。

首先，我要感谢 Dave Syer，不仅因为他是 Spring Batch 框架的创造者，让我在过去 6 年多的时间里有幸撰写了两本书，而且因为他是令我敬仰的开源实践者。当初，在我写完第一本书之后，我遇到了他并送了他一本。他向我推荐了一份使用 Java Community Process (JCP)围绕批处理(JSR-352)创建 Java Specification Request (JSR)的工作。这让我遇到了下一个我要感谢的人。

Wayne Lund 是 JSR-352 专家组的最初成员之一，也是本书的技术编辑之一，他对我取得今天的成就有巨大的影响。正是在 JSR-352 专家组里，我们会面并一起工作，根据我们在 Spring Batch 方面的经验来改进 Java 批处理规范的设计。在开发 JSR 期间，Wayne 告诉我 Spring 团队正在为 Spring Batch 项目寻找新的领军人，并问我是否有兴趣。我依然相信，我的大多数非技术背景的家人和朋友并不明白被邀请加入 Spring 团队意味着什么。我在 Spring 工程团队工作时，享受到前所未有的快乐!谢谢 Wayne，感谢你最初对我的信任和一直以来提供的支持。

我还要感谢我在 Pivotal 工作时的经理 Brian Dussault。在我的职业生涯中，我有幸为许多出色的经理工作过，对于其中的许多人，我愿意为之工作并且十分乐意再次与之共事，但是没有人能像 Brian 那样给予我支持和信任。

我还要感谢另外两组人。首先是 Apress 团队。在漫长的写作过程中，Steve Anglin 和 Mark Powers 给予我充分的理解。我敢肯定，我不是最容易共事的作者，但我很幸运他们成为本书的编辑。没有他们对本书提供的持续支持，本书不可能顺利完成。我还要感谢技术编辑 Felipe Guiterrez，他的评论和鼓励对最终的结果产生了巨大的影响。

最后且最重要的是，我要感谢我的家人。任何写过书的人都知道，写作的过程需要付出很大的代价。作者们需要将大量的时间、精力和情感投入到写作过程中。感谢我的女儿 Addison，她每天都用她的热情、无尽的好奇心激励着我。我还要感谢我的妻子 Erica，是她支持我完成了这本书。如果没有她一直以来的鼓励和支持，我早就半途而废了。对我来说，你们就是我的整个世界，谢谢你们!

前　言

Spring Batch 是我深度参与的第一个开源项目。在我的记忆中，有两件事拖延了 Spring Batch 1.0 的发布：一是确保服务质量(Quality of Service，QoS)特性在实际工作中发挥作用；二是在 API 的设计上需要投入大量的精力。无论如何，错误是不可避免的，但我认为，至少可以说我们尽了最大努力，让生活有了一个良好的开端。

如果回顾一下 Spring Batch 的发展历程，你就会发现它起源于批处理领域，并且诞生于世界各地的许多企业长时间不断重复发明的过程之中。我第一次看到这些代码是在 2006 年，当时是 Rob Harrop 在伦敦一家银行做咨询工作时写的一个很小的原型。在我们将 Spring Batch 的一些有用特性分离出来，以便在其他项目中共享之后，这些部分最终在 Spring Retry 中完成。Spring Batch 剩下的大部分，以及 Spring Batch 面向状态机的世界观，都来自与埃森哲的合作。从那时起，有太多的贡献者加入，在此无法一一列出，但值得特别提及的是 Lucas Ward，他是 Spring Batch 在早期的另一位家长和看护者。我还记得，Robert Kasanicky 和 Dan Garrette 为 Spring Batch 1.0 在 2008 年的成功发布做出过巨大贡献。

以上贡献者也为 Spring Batch 2.0 在 2010 年的成功发布发挥了重要作用，我们在 Spring Batch 2.0 中引入了 chunk 的概念以及一些特性，以支持分布式处理、并行处理和 Java 5 的新语言特性。chunk(“块”)是可以一起处理的一组数据项，这为提高效率和可伸缩性提供了可能。Spring Batch 2.0 在很长一段时间里代表最高的技术水平，并且在 JSR-352 规范启动时，成为其中的一部分。来自埃森哲的 Wayne Lund 在 Spring Batch 项目的早期就已加入其中，他也是 JSR-352 专家组的一员，现在是 Pivotal 的平台架构师。

那时候，Michael Minella 也是 JSR-352 专家组的成员，他在现实生活中大量地使用 Spring Batch 并撰写了一本书。当他在 2012 年加入 Spring 团队时，Spring 正好开始 Spring Batch 3.0 的准备和发布工作。在 Spring Batch 3.0 中，我们第一次看到了@EnableBatchProcessing 注解，并且将重心从 XML 配置转到使用 Java 进行配置。Michael 很快就以项目负责人的角色接管了这个项目，领导 Spring Batch 的 3.x 系列版本一直到 4.0 版本。在 4.0 版本中，Java 8 成为基线，并且添加了一些新的流畅风格(fluent-style)的配置构建器。与 Spring Cloud Data Flow 的连接以及分布式处理的工业化也在这一时期发生。在 2018 年年初，Mahmoud Ben Hassine 作为新的项目联合负责人加入进来，他一直在帮助 Michael 推动 Spring Batch，并仔细听取了许多用户的反馈。

所以，在本书写作期间，Spring Batch 已经发展了十年，期间也不断有优秀的贡献者加入进来。在未来几年里，Spring Batch 肯定还有很多事情要做，因为批处理似乎永远不会消失。这确实很有意思。

Dave Syer，Spring Batch 项目创始人

2019 年于伦敦

目　录

第 1 章

批处理和 Spring

在最近的 IT 技术会议上，几乎没怎么出现有关批处理的话题。快速浏览一下最大型的 Java 会议，你会发现几乎没有专门针对这个主题的演讲。会议室里坐满了学习流式处理的听众，有关数据科学的讲座吸引了大批听众，专注于基于 Web 系统的云原生应用的博客斩获众多的访问流量。尽管如此，批处理依然存在。

在美国，你的个人银行报表和 401K 报表(401K 指的是美国在 1981 年创立的一种养老金计划，由于相关规定都在美国的《国内税收法》的条款 401 中，因此简称 401K)都是通过批处理生成的。你从喜爱的商城那里收到的折扣邮件也可能是通过批处理来发送的，甚至维修人员上门修理洗衣机的顺序也是由批处理决定的。在诸如 Amazon 的网站上，推荐相关产品的数据科学模型是通过批处理生成的。编排大数据的任务也是通过批处理实现的。在人们从 Twitter 获取新闻的时代里，Google 认为等待页面刷新会花费太长的时间以至于无法提供搜索结果，YouTube 可以让一个人一夜成名，那么为什么我们还需要批处理呢？

以下列举了几个合理的解释理由：

- 你并不总是能够立即获得所有的必要信息。对于给定的处理过程，批处理允许你在进行必要的处理前收集所需的信息。以每月的银行对账单为例，在每笔交易之后都打印采用一定文件格式的对账单是否有意义？更合理的做法是在月底重新查看经过审查的交易列表，然后从中构建对账单。
- 有时候批处理具有良好的业务意义。尽管大多数人都希望在网上购物时，当他们进行购买的那一刻，商品就被放到一辆送货卡车上，但对于零售商来说，这可能并不是最好的做法。假设顾客临时改变了主意，想要取消订单，这时如果商品还没有进行配送，那么取消订单的损失(否则就配送出去了)就会少一些。给顾客留出一些时间，并且批量地进行配送，能够为零售商节省很多成本。
- 批处理可以更好地利用资源。数据科学方面的示例在这里也是合理的。典型情况下，数据模型的处理被分为两个阶段。首先是模型的生成，此时需要对大量的数据进行密集的数学处理，这会花费很多时间；其次是对生成的模型进行评估，或者使用生成的模型对新数据进行评分，这一阶段非常快速。第一个阶段的意义在于，流式系统实时地使用了在流式用例之外通过批处理得到的结果(即数据模型)。

本书主要介绍如何使用 Spring Batch 框架进行批处理。本章将介绍批处理的历史，指出开发批量作业时面临的挑战，为使用 Java 和 Spring Batch 开发批处理提供案例，最后在较高层次上概述 Spring Batch 框架及其特性。

1.1 批处理的历史

对批处理历史的介绍实际上也就是对计算自身历史的介绍。

1951 年，UNIVAC 成为第一台商业化生产的计算机。在这之前，计算机都是为特定功能而设计的独特的定制机器。例如，1946 年美国军方委托制造了一台计算机来计算炮弹的弹道，名为 ENIAC，成本约为 500 万美元(按 2017 年美元价值计算)。UNIVAC 由 5200 个真空管组成，质量超过 14t，拥有 2.25 MHz 的惊人速度，并且能够运行从磁带驱动器里加载的程序。很快，UNIVAC 被认为是第一个商用的批处理器。

在深入研究历史之前，我们应该明确地定义什么是批处理。你开发的大部分应用都包含交互元素，无论是用户单击 Web 应用中的链接，在胖客户端将信息输入表单，还是通过某种中间件接收消息，抑或在手机或平板电脑上单击应用。批处理与这些应用的类型刚好相反。批处理被定义为对有限数据进行处理，并且没有交互或中断。一旦启动了批处理过程，批处理就将以某种形式完成，而不需要任何干预。

在计算机和数据处理演进了四年之后，计算领域出现了巨大的变化——高级语言，它们最先由 IBM 704 上的 Lisp 和 Fortran 引入。但自那以后，面向业务的通用语言(Common Business Oriented Language，COBOL)成为批处理领域的重量级语言。COBOL 在 1959 年首次发布，并在 1968 年、1974 年、1985 年、2002 年和 2014 年进行了修订，现代商业中仍然运行着批处理。2012 年，《计算机世界》杂志所做的调查显示，超过 53%的受访企业使用 COBOL 进行新业务的开发。有趣的是，该调查还指出，他们的 COBOL 开发人员的平均年龄在 45～55 岁。

在过去的 20 多年里，COBOL 还没有出现过被广泛采用的重大修订。教授 COBOL 及相关技术的学校数量大幅减少，取而代之的是 Java 和.NET 等新技术。COBOL 的硬件十分昂贵，而资源变得稀缺。

大型机并不是进行批处理的唯一场所。前面提到的那些电子邮件可能是由不在大型机上运行的批处理发送的。我们从你最喜欢的快餐连锁店的销售终端下载的数据也是批量的。但是，大型机上的批处理过程与我们通常为其他环境(例如，C++和 UNIX)编写的批处理过程之间存在着显著差异。每个批处理过程都是定制开发的，它们几乎没有共用的地方。自从 COBOL 接管批处理以来，很少有新的工具或技术出现。虽然定时作业(Cron Job)已经能够在 UNIX 服务器上启动定制开发的进程，以及在 Microsoft Windows 服务器上启动调度任务，但是还没有出现被业界接受的用于执行批处理的新工具。

直到 Spring 出现后，在 2017 年，由 Accenture 的富大型机和批处理实践驱动，并由 Accenture 与 Interface21(Spring 框架的最初创造者，现在是 Pivotal 的一部分)合作，才为企业级批处理创建出一种全新的开源框架。受多年来被认为是 Accenture 架构的支柱这一理念的启发，这种协作产生了在 JVM 上进行批处理的实际标准。

Accenture 由于是首次正式进军开源领域，因此该公司选择将其在批处理方面的专长与 Spring 的流行度和特性集结合起来，以希望创建出健壮的、易于使用的框架。在 2008 年 3 月底，Spring Batch

1.0.0 正式向公众发布，它代表了在 Java 世界中首次基于标准进行批处理的方法。一年多以后，在 2009 年 4 月，Spring Batch 发布了 2.0.0 版本，其中添加了很多特性，例如，用 JDK 1.5+取代对 JDK 1.4 的支持、分块(Chunk-Based)处理、优化配置选项以及在框架内部大幅增加对伸缩性的支持选项。3.0.0 版本在 2014 年的春天出现，其中引入了新的 Java 批处理标准 JSR-352 的实现。4.0.0 版本包含了 Spring Boot 世界中的基于 Java 的配置。

1.2　批处理面临的挑战

毫无疑问，你非常熟悉基于 GUI 编程(胖客户端和 Web 应用等)的挑战：安全问题、数据校验以及实现对用户友好的错误处理。不可预测的使用模式会导致资源利用率大幅提升。所有这些都与同一件事情有关：用户与软件交互的能力。

然而，批处理与此不同。之前讲过，批处理不需要进行额外的交互就可以完成。正因为如此，许多 GUI 应用的问题不复存在。是的，这里需要考虑安全性，数据校验也是必要的，但是使用峰值和友好的错误处理是可预测的，也可能不适用于批处理。你可以预测批处理中的负载并进行相应的设计。只能根据可靠的日志记录和通知，才能快速而明显地表明批处理失败，因为技术资源可以解决任何问题。

批处理世界里的一切都是小菜一碟，没有挑战吗？很抱歉，事实不是这样，因为在许多常见的软件开发挑战中，批处理有自身独有的特点。软件架构通常包含许多非功能性需求：可维护性(maintainability)、易用性(usability)、伸缩性(scalability)等。这些都与批处理相关，只是方式不同。

易用性、可维护性和伸缩性是相关的。在批处理中，无须考虑用户界面，所以易用性并不是指好看的 GUI 和炫酷的动画。在批处理中，易用性与代码相关：包括错误处理和可维护性。你可以轻松地扩展公共组件以添加新特性吗？它们有不错的单元测试覆盖率，以便修改原有组件时，让你知道对整个系统的影响吗？在作业失败时，你能够知道是在何时、何地发生，并且在不花费大量时间进行调试就知道原因吗？以上是易用性影响批处理的所有方面。

接下来探讨一下伸缩性。是时候核实一下现状了：你上一次为一个每天有一百万访客的网站工作是什么时候？每天有十万访客呢？不过，企业内部开发的大多数 Web 站点都不会被浏览很多次。但是，需要在一晚上处理一百万个或更多个事务的批处理并不罕见。让我们把加载 Web 页面所需的 8 秒时间视为可靠的平均值。如果需要花费这么长的时间通过批处理来处理一个事务，那么处理十万个事务需要超过 9 天的时间(处理一百万个事务需要三个月以上)。在现代企业里，这样的情况对于任何系统都不实用。最重要的底线是，批处理过程需要能够处理的范围通常比过去开发的 Web 应用或胖客户端应用大一个或多个数量级。

我们还必须考虑可用性(Availability)。再说一次，这与你可能习惯的 Web 应用或胖客户端应用有所不同。通常批处理不是 24×7 运行的。事实上，它们通常有约定的运行时间。大多数企业都将作业安排在给定的时间运行，此时所需资源(硬件、数据等)已经可用。比如，以建立退休账户对账单为例。尽管可以在一天的任何时候运行作业，但是很可能最佳的进行时间是在闭市之后，这样就可以使用基金的收盘价来计算余额。你可以按需运行吗？你能在规定的时间内完成作业，以免影响其他系统吗？这些问题和其他问题都会影响批处理系统的可用性。

最后，我们必须考虑安全性。通常，在批处理世界中，安全性与侵入系统和破坏东西无关。在安全性方面，批处理所要扮演的角色是保证数据的安全。数据中的敏感字段是否已加密？是否无意中记

录了个人信息？如何访问外部系统——是否需要登录凭证，是否能够以适当的方式确保这些登录凭证的安全性？数据验证也是安全性的一部分。一般情况下，虽然要处理的数据已被审查过，但是你仍然应该确保遵循一定的规则。

如你所见，在开发批处理时，会涉及大量的技术性挑战。从大多数系统的大规模伸缩到安全性，批处理都会遇到。这就是开发批处理的乐趣：你将更多地关注于解决技术问题，而不是调试最新的 JavaScript 前端框架。问题是，既然大型机上已经有了基础设施，而采用新平台也会有风险，那么为什么还要使用 Java 进行批处理呢?

1.3 为什么使用 Java 进行批处理

基于上面列出的种种挑战，为什么还要选择 Java 和开源工具(比如 Spring Batch)进行批处理呢？我能想到如下 6 个使用 Java 和开源工具进行批处理的原因：可维护性、灵活性、伸缩性、开发资源、社区支持和成本。

首先是可维护性。批处理必须考虑可维护性。批处理代码的生命周期通常比其他应用长得多。原因在于：没有人能看到批处理代码。与那些必须紧跟当前趋势和风格的 Web 应用和客户端应用不同，批处理程序用于处理数字并构建静态输出。只要还在运行，大部分人只需要享用批处理结果即可。正因为如此，我们需要以一种很容易但不会招致很大风险的方式修改代码。

下面进入 Spring 框架这一话题。Spring 框架是为下面这些可以利用的特性而设计的：可测试性和抽象。Spring 框架通过依赖注入进行对象的解耦，同时 Spring portfolio 提供了一些额外的测试工具，以允许你构建出健壮的测试套件，把维护风险降到最低。在不深入研究 Spring 和 Spring Batch 工作方式的情况下，Spring 提供了一些工具，可以使用声明的方式对文件和数据库执行 I/O 操作等。你无须编写 JDBC 代码和 Java 中梦魇般的文件 I/O API。Spring Batch 给应用带来了注入事务和提交次数的功能，所以不必管理当前处于处理的什么位置，也不必管理故障发生时应该做什么。这些仅仅是 Spring Batch 和 Java 为你提供的可维护性方面的优势。

使用 Java 和 Spring Batch 的原因还包括灵活性。在大型机世界中，只能在大型机上运行 COBOL 或 CICS，就是这样。这最终会成为一种完全定制化的解决方案，因为没有被业界接受的批处理框架。大型机和 C++/UNIX 都没有提供用于部署的 JVM 和 Spring Batch 的特性集。想要使用 UNIX/Linux 或 Windows 在服务器、桌面或大型机上运行批处理流程？没关系。想要部署到应用服务器、Docker 容器或云环境中？选择适合需求的即可。想要使用瘦 WAR(Thin WAR)、胖 JAR(Fat Jar)或者其他任何新的热门技术？Spring Batch 都能应对。

然而，Spring Batch 的灵活性不仅来自 Java 的“一次编写，随处运行”特性，灵活性的另一方面是在系统之间共享代码的能力。对于 Web 应用中经测试和调试过的相同服务，批处理过程依然可以使用。事实上，访问曾被锁定在其他平台上的业务逻辑的能力是迁移到这个平台的最大胜利。通过使用 POJO 来实现业务逻辑，就可以在 Web 应用和批处理过程中——毫不夸张地说，可以在使用 Java 进行开发的任何地方——使用它们。

Spring Batch 的灵活性还包括扩展用 Java 编写的批处理的能力。让我们来看看用来对批处理过程进行伸缩的一些方案。

- 大型机：大型机在伸缩性方面的附加能力有限。为了并行地完成任务，唯一真正可行的方

式是在单个硬件上并行地运行完整的程序。这种方式的局限性在于需要编写和维护代码来管理并行处理，比如错误处理和跨程序的状态管理。此外，处理能力受限于一台机器的资源。

- 定制化的处理：即使在 Java 中，从头开始也是让人望而却步的。在大量数据的基础上达到伸缩性和可靠性并非易事。你会再次面对编写负载均衡时的相同问题。当开始跨物理设备或虚拟机分发计算时，你还会面临巨大的基础设施方面的复杂性。你必须关注各个部分之间的通信是如何工作的。还有数据可靠性方面的问题。如果定制化的 Worker 宕机会发生什么？此外还有更多的问题。我并不是说这些无法实现。我的意思是，你最好把时间花在编写业务逻辑上，而不是重新发明轮子。
- Java 和 Spring Batch：尽管 Java 本身拥有处理上述大多数元素的工具，但是以可维护的方式将这些元素组合在一起则非常困难。Spring Batch 解决了这个问题。想在单独服务器上的单独 JVM 中运行批处理吗？没问题。随着业务持续增长，现在需要在五个不同的节点之间分配账单计算工作，以便在一夜之间完成所有计算。这也是可以的。每个月都有峰值，你能够在那一天使用云资源进行扩展吗？这当然可以。数据可靠性？只需要进行一些配置并记住一些关键原则，就可以完全处理事务回滚和提交计数。

你将看到，随着对 Spring Batch 框架及相关生态系统的深入，困扰前面的批处理方案的问题可以使用经过良好设计和测试的解决方案来缓解。到现在为止，本章已经讨论了选择 Java 和开源工具进行批处理的技术原因。然而，技术并不是做出类似决定的唯一原因。找到合格的开发资源来编写和维护系统是很重要的。前面提到过，批处理中的代码相比 Web 应用来说生命周期要长得多。正因为如此，找到理解相关技术的人与技术本身同样重要。Spring Batch 基于非常流行的 Spring 框架，遵循 Spring 中的惯例，并且使用 Spring 工具和其他基于 Spring 的应用。Spring Batch 是 Spring Boot 的一部分。所以，对于任何拥有 Spring 经验的开发人员来说，Spring Batch 的学习难度是最小的。但是，你能找到 Java，特别是 Spring 方面的资源吗？

在 Java 中，做很多事情的理由之一是社区支持。Spring 框架家族通过 GitHub、StackOverflow 和相关资源，在网上拥有一些非常活跃的大型社区。作为家族一员，Spring Batch 也有成熟的社区。再加上能够访问源代码的强大优势以及提供按需购买支持的能力，Spring Batch 已包含所有的支持基础。

最后是成本问题。任何软件项目都有很多成本：硬件、软件、许可费、薪酬、咨询费用、支持合同等。然而，Spring Batch 不仅仅最划算，而且也是最便宜的。在使用云资源和开源的框架后，唯一的经常性支出费用是开发薪酬——这相比与其他方案相关的定期许可费和硬件支持合同要少得多。

不言而喻。使用 Spring Batch 不仅仅是技术上最合理的方法，而且是最经济有效的方法。下面让我们开始了解 Spring Batch 到底是什么。

1.4　Spring Batch 的其他用途

到目前为止，你肯定想知道将大型机替换为 Spring Batch 是否有好处。当你持续地思考正在面对的项目时，你并不是每天都在提取 COBOL 代码。如果这就是 Spring Batch 框架所能带来的全部好处，

那么用处并不大。然而，Spring Batch 框架可以帮助你处理许多其他用例。

Spring Batch 最常见的用例可能就是 ETL 处理了，也就是抽取(Extract)、转换(Transform)和加载(Load)。将数据从一种格式转换到另一种格式是企业数据处理的重要组成部分。Spring Batch 基于块的处理和极强伸缩能力，因而非常适合处理 ETL 工作负载。

Spring Batch 的另一个用例是数据迁移。在重写系统时，通常会将数据从一种形式迁移到另一种形式。风险在于，与通常的开发工作相比，你可能会编写缺乏测试的一次性解决方案，并且缺少对数据一致性方面的控制。然而，在考虑 Spring Batch 的特性时，这些似乎是顺其自然的。你无须编写大量代码就能获得并运行一个简单的批量任务，Spring Batch 还提供了许多数据迁移工具，它们应该但很少提供诸如提交次数和回滚的功能。

Spring Batch 的第三个用例是任何需要执行并行处理的过程。随着芯片制造商的生产工艺不断接近摩尔定律的极限，开发人员意识到继续提高应用性能的唯一方式并不是让单个操作变得更快，而是以并行方式执行更多的操作。人们最近发布了很多用来协助并行处理的框架。大部分的大数据平台，比如 Apache Spark、YARN、GridGain、Hazlecast 等，都是最近几年才出现的，它们的目的就是试图利用多核处理器和众多的云端服务器。然而，诸如 Apache Spark 的框架需要修改代码和数据，以便适应它们的算法或数据结构。Spring Batch 能够在多个核心或服务器上对处理进行伸缩(如图 1-1 所示的 Master/Worker 步骤配置)，并且能够使用与 Web 应用相同的对象和数据源。

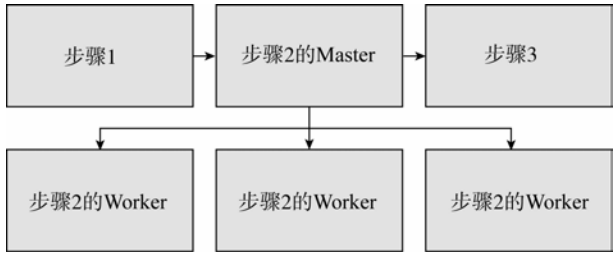

图 1-1　简化并行处理

编排工作负载是 Spring Batch 的另一个常见用例。通常企业级的批处理不止包含一个步骤，而是需要协调很多已解耦的步骤。你可能需要加载文件，对数据进行两种独立的处理，然后输出结果。Spring Batch 能够很好地编排这些任务。其中的一个示例是 Spring Cloud Data Flow 使用 Spring Batch 来处理“组合任务”(composed task)。这里，Spring Batch 调用 Spring Cloud Data Flow 来启动其他的功能，并且跟踪哪些已完成和哪些需要完成。图 1-2 展示了 Spring Cloud Data Flow 提供的用来构建“组合任务”的拖拉界面。

最后是持续处理，又称 24×7 处理。在许多用例中，系统接收到持续或接近持续的数据输入。尽管以数据到来的速率接收数据对于防止积压是必要的，但是，将数据按照块进行批量处理的话，可能会得到更好的性能(如图 1-3 所示)。Spring Batch 提供了一些工具，以便能够以可靠且可伸缩的方式进行这种类型的处理。使用 Spring Batch 框架的特性，可以从队列里读取消息，将每一批次作为一个块，然后在永不结束的循环中一起进行处理，从而能够在大数据量的情况下提高吞吐量，而不必了解从头开发此类解决方案的复杂细节。

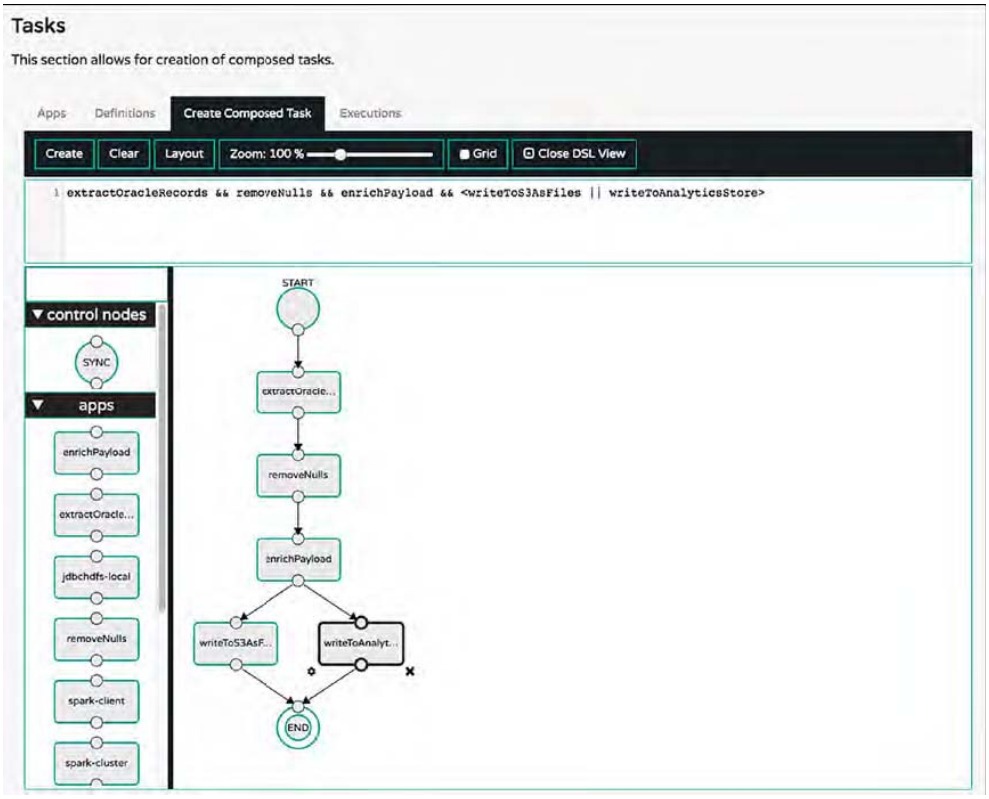

图 1-2　通过 Spring Cloud Data Flow 编排任务

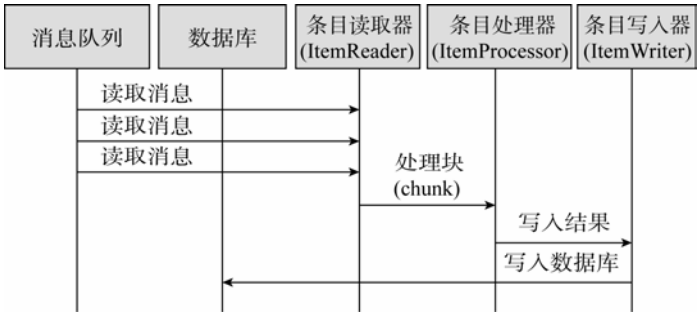

图 1-3　通过批量处理消息提高吞吐量

如你所见，尽管 Spring Batch 是为实现类似大型机的处理而设计的，但是作为框架，Spring Batch 可以用来简化各种各样的开发问题。在了解了有关什么是批处理以及为什么应该使用 Spring Batch 之后，让我们最终开始研究 Spring Batch 框架本身。

1.5　Spring Batch 框架

Spring Batch 是由埃森哲(Accenture)和 SpringSource 合作开发的框架，它以一种基于标准的方式来

实现常见的批处理模式和范式。

Spring Batch 实现的特性包括数据验证、格式化输出、以可重用的方式实现复杂逻辑以及处理巨大数据集的能力。当深入本书中的示例时，你就会发现如果熟悉 Spring，那么 Spring Batch 就很有意义。

Spring Batch 的架构如图 1-4 所示。

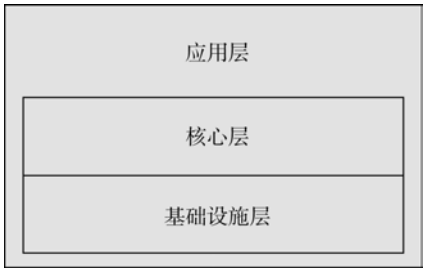

图 1-4　Spring Batch 的架构

Spring Batch 采用分层的配置并由三层组成。上面是应用层(application layer)，其中包括所有用来构建批处理的定制化代码和配置。业务逻辑、服务以及组织任务的配置等，都是应用层关心的内容。注意，应用层并不在其他两层之上，而是封装了其他两层。原因是，尽管你开发的大部分东西都由应用层(可与核心层一起协作)组成，但有时候你会编写定制化的基础设置部分，比如自定义的读取器和写入器。

应用层在大部分时间里都在与下一层——核心层——交互。核心层包含用于定义批处理域的所有部分。核心组件的元素包括作业(Job)和步骤(Step)接口，以及用来执行作业的如下两个接口：JobLauncher 和 JobParameters。

最下面是基础设施层。在处理任何东西前，都需要读取或写入文件、数据等。在任务失败后，必须能够进行重试。这些部分被认为是一些通用的基础设施，并且位于 Spring Batch 框架的基础设施组件中。

■ **注意**　人们对 Spring Batch 的常见误解是：认为 Spring Batch 是调度器，或者认为 Spring Batch 提供了调度器。事实并非如此。在 Spring Batch 框架中，没有办法进行作业调度，比如在给定时间或在给定的事件发生后执行。有很多方式可用来启动作业，从简单的定时作业脚本到 Quartz，甚至企业级的调度器(如 Control-M)都可以做到，但 Spring Batch 框架本身并不包括这些。第 4 章将介绍如何启动作业。

下面浏览一下 Spring Batch 的一些特性。

1.5.1　使用 Spring 定义作业

批处理中有不少领域特定的概念。作业(Job)表示处理过程，在从头到尾整个执行过程中，没有任何终端或交互。一个作业可能包括多个步骤(Step)。每个步骤可能有相关的输入输出。当步骤失败时，步骤可能是可重复的，也可能是不可重复的。作业流可能是有条件的(比如，仅仅在计算收入的步骤返回超过 1 000 000 美元时才执行计算奖金的步骤)。Spring Batch 提供了类、接口、XML 模式和 Java 配置实用程序，它们使用 Java 定义了这些概念以适当地划分关注点，并以 Spring 使用

者熟悉的方式将它们连接在一起。例如，代码清单 1-1 展示了一个基本的 Spring Batch 作业，可通过 Java 语言进行配置。所以，你只需要对 Spring 有一定的基本了解，就可以非常快速地掌握 Spring Batch 框架。

代码清单 1-1　Spring Batch 作业配置示例

```
@Bean
public AccountTasklet accountTasklet() {
   return new AccountTasklet();
}

@Bean
public Job accountJob() {
   Step accountStep =
      this.stepBuilderFactory
          .get("accountStep")
          .tasklet(accountTasklet())
          .build();
   return this.jobBuilderFactory
              .get("accountJob")
              .start(accountStep)
              .build();
}
```

代码清单 1-1 创建了两个 Bean。第一个 Bean 是 AccountTasklet。AccountTasklet 是一个自定义的组件，里面包含了步骤中的业务逻辑。Spring Batch 将一遍又一遍地调用 execute 方法，每次调用都发生在新的事务中，直到 AccountTasklet 表明调用已经结束。

第二个 Bean 是真正的 Spring Batch 作业。在这个 Bean 中，我们从刚刚定义的 AcountTasklet 里创建了一个单独的步骤，并且使用了工厂(factory)提供的构建器(builder)。然后，我们使用工厂提供的构建器在这个步骤中创建了一个作业。在应用启动时，Spring Boot 会找到这个作业并且自动执行。

1.5.2　管理作业

编写能够一次性地处理一些数据，然后永远不再运行的 Java 程序确实是可行的。但是，关键型任务的处理需要更健壮的方法。一些特性，比如保存作业的状态以便重新执行、在作业失败时通过事务管理维护数据的一致性、将以往作业的执行性能指标保存为趋势等，都是企业级批处理系统期望的。Spring Batch 包含了这些特性，并且大部分默认都是开启的。在处理任务时，它们对性能和需求只有非常微小的影响。

1.5.3　本地和远程的并行化

正如前面所讨论的，批处理作业的规模以及能够扩展它们的需求，对于任何企业级批处理解决方案来说都是至关重要的。Spring Batch 提供了通过许多不同方法来解决这个问题的能力。从简单的基于线程的实现(每个提交间隔都在线程池自己的线程中处理)，到并行地运行所有步骤，再到配置通过分区从远程主机获得的工作单元的工作网格，Spring Batch 及相关的生态系统提供了不同选项的集合，包括并行块/步骤的处理以及远程块的处理和分区。

1.5.4 标准化 I/O

从具有复杂格式的平面文件、XML 文件(XML 是流，从不作为整体加载)、数据库或 NoSQL 存储，到以简单的配置写入文件或 XML，使用 Spring Batch 编写作业的可维护性表现在，可以通过代码抽象文件和数据库进行输入输出。

1.5.5 Spring Batch 生态系统的其他部分

与 Spring 产品组合中的大部分项目类似，Spring Batch 并不是孤立存在的。Spring Batch 只是生态系统的一部分，在 Spring Batch 生态系统中，一些其他项目对 Spring Batch 进行了扩展和补充，以提供更健壮的解决方案。Spring 产品组合中与 Spring Batch 一起工作的其他一些项目如下。

1. Spring Boot

Spring Boot 于 2014 年引入，它采用一种有主见的方式开发 Spring 应用。现在，Spring Boot 几乎是开发 Spring 应用的标准方式，它提供了易于打包、部署和启动包含批处理的所有 Spring 工作负载的设施和工具。Spring Boot 也是 Spring Cloud 提供的云原生方案的支柱之一。因此，Spring Boot 也将是本书开发批量应用的主要方法。

2. Spring Cloud Task

Spring Cloud Task 是 Spring Cloud 项目下的子项目，提供了在云环境中执行有限任务的设施。作为针对有限工作负载的框架，批处理是一种能够与 Spring Cloud Task 良好集成的处理风格。Spring Cloud Task 为 Spring Batch 提供了许多扩展，包括发布提示性消息(作业的开始/结束、步骤的开始/结束等)以及动态伸缩批处理作业的能力(而不是使用 Spring Batch 直接提供的各种静态方法)。

3. Spring Cloud Data Flow

自行编写批处理框架不仅仅意味着必须重新开发 Spring Batch 提供的开箱即用的性能、可伸缩性和可靠性等特性，还需要某种管理和编排工具集以完成作业的启动和停止，以及查看既往作业的运行数据等工作。然而，如果使用 Spring Batch，那么除了能够完成以上工作以外，你还可以使用 Spring Batch 提供的如下附加功能：Spring Cloud Data Flow。Spring Cloud Data Flow 是用来在云平台(CloudFoundry、Kubernetes 等)上编排微服务的工具。如果以微服务的方式开发批处理应用，就可以将它们以动态方式部署。

1.5.6 Spring 的所有特性

虽然 Spring Batch 包含了一系列令人印象深刻的特性，但最棒的特性在于 Spring Batch 构建于 Spring 之上，从而拥有了 Spring 为任何 Java 应用提供的详尽特性，包括依赖注入(Dependency Injection)、面向接口编程(Aspect-Oriented Programming，AOP)、事务管理以及用于处理大部分通用任务(JDBC、JMS、Email 等)的模板和助手，这些特性为你构建企业级的批处理过程提供了所需的几乎一切。

如你所见，Spring Batch 为开发人员带来了很多东西。Spring 框架的成熟开发模型以及可伸缩性和可靠性等特性，让你能够使用 Spring Batch 快速运行批处理任务。

1.6　如何阅读本书

在介绍了批处理和 Spring Batch 之后，我相信你一定迫不及待地想要深入研究一些代码，并了解使用 Spring Batch 框架构建批处理过程的全部内容。第 2 章将介绍批处理的架构，定义我们已开始使用的一些术语(如作业、步骤等)，并介绍如何设置第一个 Spring Batch 项目。

我写作本书的主要目的之一是，不仅让你深入了解 Spring Batch 框架是如何工作的，而且展示如何在现实案例中应用这些工具。第 3 章将提供第 10 章中示例应用的项目需求和技术架构。

本书的代码示例可以在 GitHub 上找到，下载网址为 https://github.com/Apress/def-guide-spring-batch，也可通过扫描封底的二维码下载。

1.7　本章小结

本章介绍了批处理的历史，其中涵盖了开发人员在进行批处理过程中面临的一些挑战，并且证明了使用 Java 和开源技术来克服这些挑战是合理的。最后，本章通过展示 Spring Batch 框架的高级组件和特性，让你对 Spring Batch 框架有了一定的了解。到目前为止，你应该已经很好地理解了自己将要面临的挑战，并了解了 Spring Batch 中用来应对这些挑战的工具。现在，你所要做的就是学习如何去做。让我们开始吧！