vmware | SPRING & CLOUD FOUNDRY 开发者大会

用Spring和RabbitMQ技术应对消息传送挑战

Gary Russell，主管工程师，VMware (@gprussell)

应对消息传送挑战

- 现代应用程序的趋势
- 通过 Spring Integration实现模块化
- 通过 AMQP 进行分发
- 通过 Hadoop 实现统一

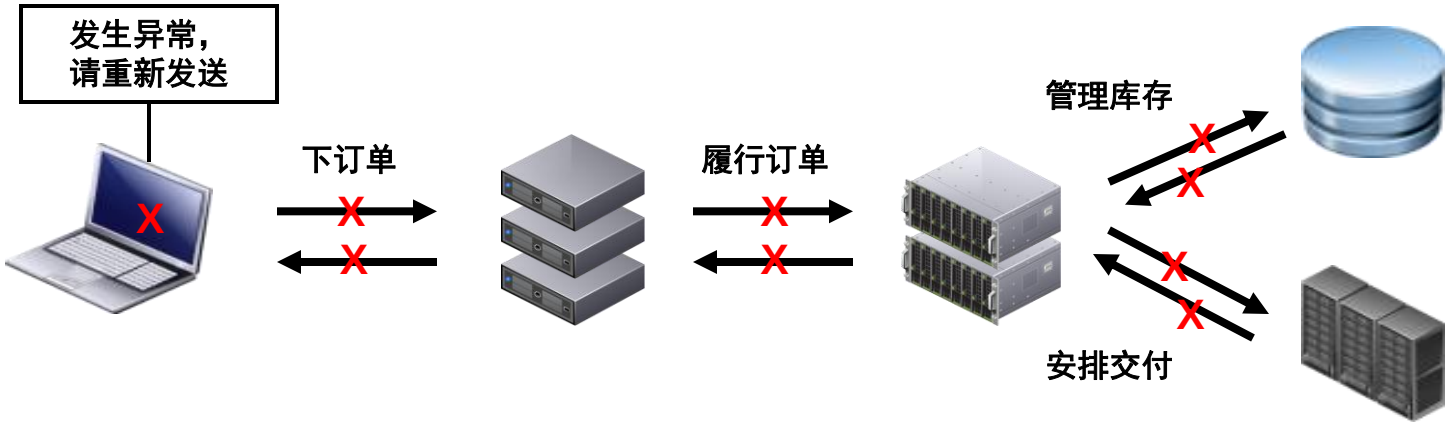
现代应用程序趋势

- 利用框架编写
  - 开发人员工作效率和创新

- 资源密集型
  - 多台设备意味着更多连接
  - Web 定位推动数据呈指数级增长

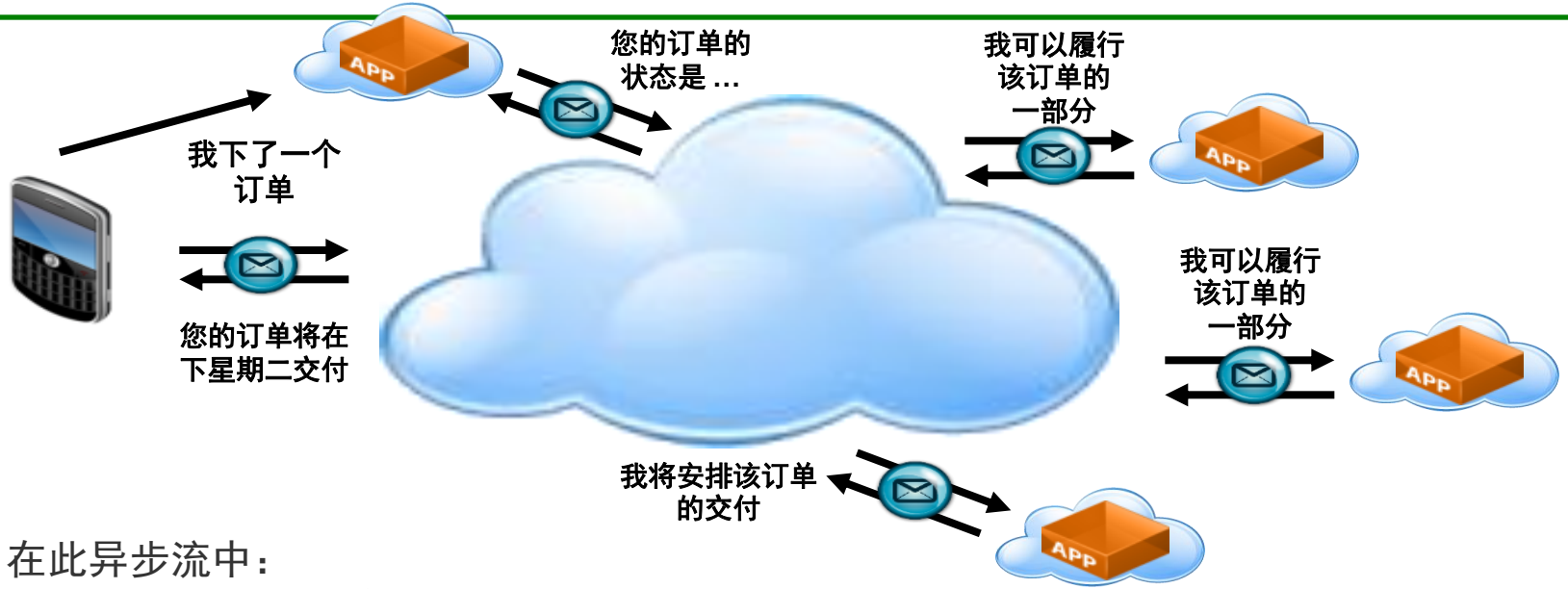
- 部署在虚拟化和云基础架构上
  - 呈现向混合（公有和私有）基础架构发展的趋势

同步的架构很脆弱



如果此同步流中的任何组件不可用，则所有传送中的消息都会受到影响，无论这些消息当前处于哪个组件中。而且由于要在各个组件间维持状态，因此重置系统可能很复杂

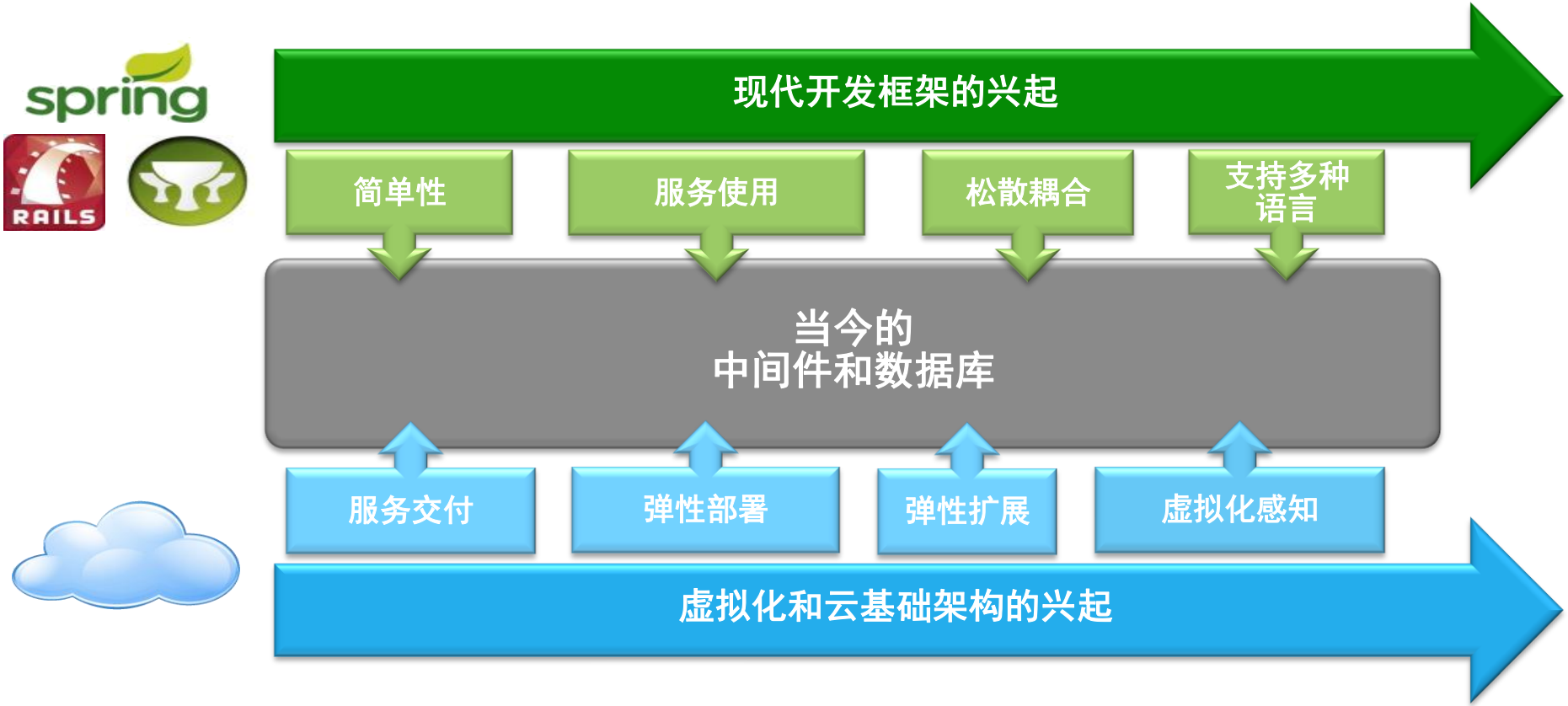
异步架构主导着整个 Web



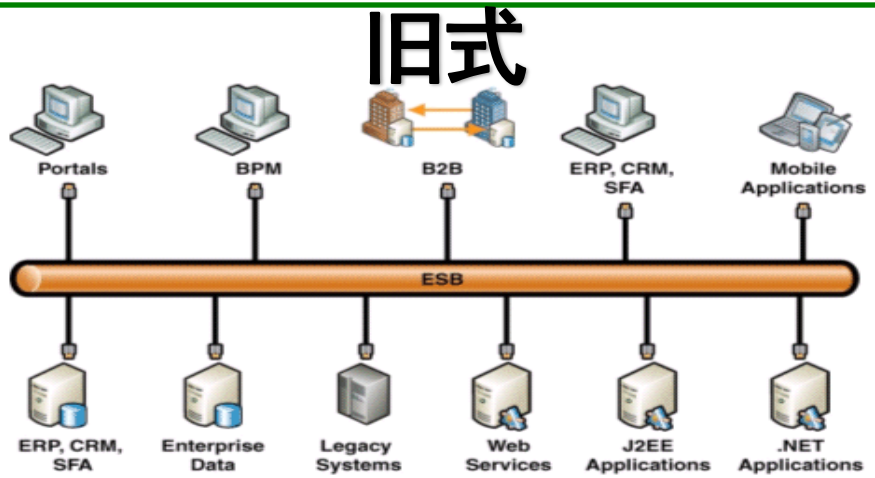
在此异步流中：

- 所有的状态都保存在这次被传送的简单消息中
- 每个无状态组件都仅与代理云进行交互
- 如果某个组件失去联系，则只有该组件中正在传输的消息才需要重新传送
- 可以按需对组件进行快速置备

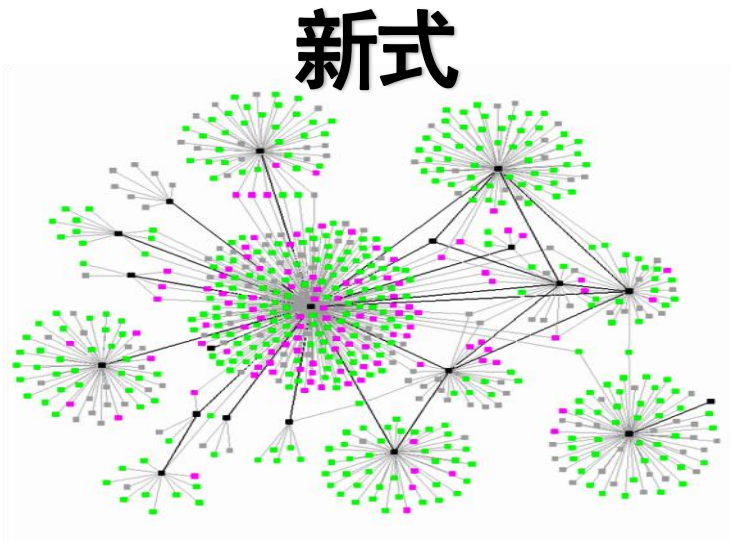
当今的中间件和数据库无法跟上使用需求



现代应用程序需要现代的消息传送

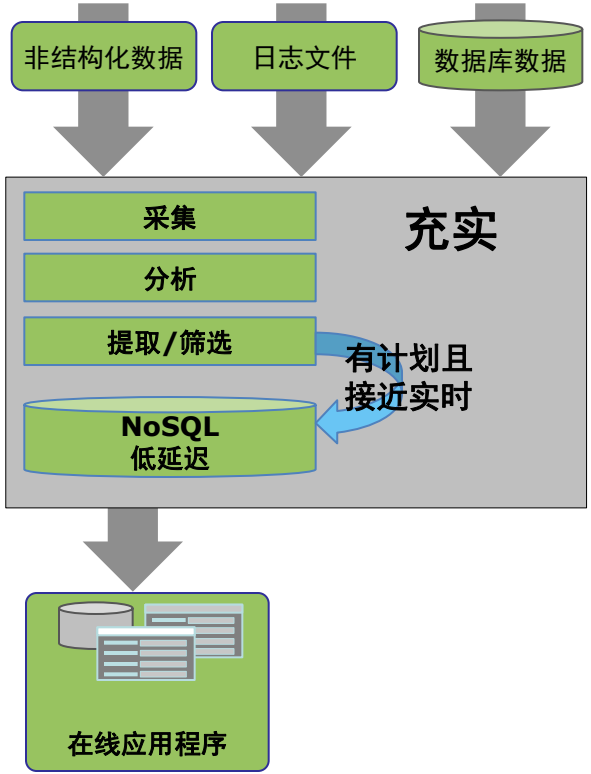


传统消息传送方法依赖已知节点之间可预测的，静态的交互



现代消息传送方法则接受 Web 的不可预测、动态和瞬态的特点

现代应用程序需要现代的数据管理



采集

- 采集数据经常是过于庞大而且无法管理

处理

- 揭示数据间的总体特征
- 通过分析快速发现模式
- 从巨量数据流中筛选出有用的数据
- 基于调度的微观或宏观批处理

交换

- 将结果推送到 NoSQL，以实现实时交付
- 利用模式在适当时间交付适当内容/提供给适当人员

通过 Spring Integration实现模块化

什么是 Spring 集成？

- 从核心上说，是一个嵌入式消息总线
  - 灵感来源于 Gregor Hohpe 和 Bobby Woolf 合著的《企业集成模式》(*Enterprise Integration Patterns*)（2004 年）
  - 可在任何 Spring ApplicationContext *内*运行
  - 所有组件都是 Spring 管理的对象

- 同时也是一个应用程序集成框架
  - 可通过适配器连接到其他系统
  - 单向通道适配器
  - 双向消息传送网关

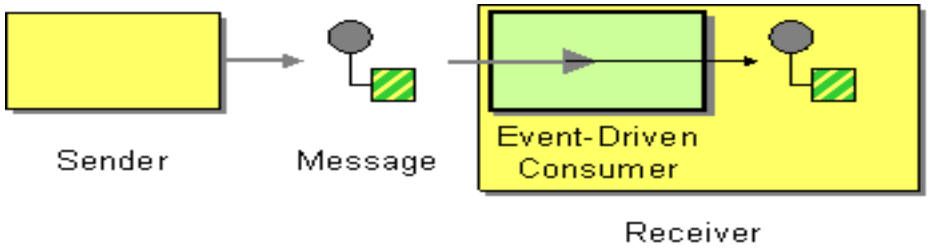
如何在现代应用程序中使用消息传送？

- **事件**
  我需要知道何时执行操作
- **片段**
  我只需向您提供此数据的一个片段
- **路由**
  控制哪些人获取哪些消息，而无需更改发送人或接收人
- **批处理**
  生成者和使用者可以在独立的时间运行
- **发布**
  告知想知道此消息的所有人
- **模块性**
  部署、扩展分布式系统并控制其版本

事件驱动的消费者

我需要知道何时执行操作 …

```
<service-activator
    input-channel="hotDrinks"
    ref="barista"
    method="prepareHotDrink"
    output-channel="preparedDrinks" />
```

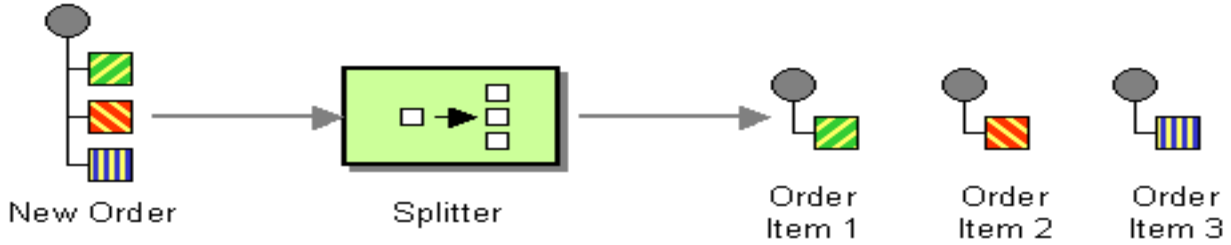


*事件驱动的消费者*是指消息在通道上传输时自动获得消息的使用者。接收人的操作方式就像是消息传输是一个触发接收人进行操作的事件。

片段

我只需向您提供这段数据 …

```
<int:splitter
        input-channel="orders"
        expression="payload.items"
        output-channel="drinks" />
```

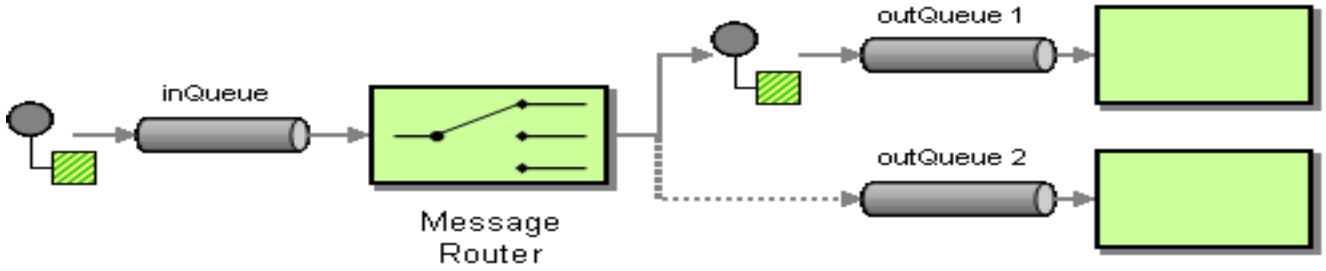


使用Splitter 将复合消息分割为一系列单独消息，每条消息都包含与一个具体项相关的数据。

路由

控制哪些人获得哪些消息

```
<int:router
        input-channel="drinks"
        expression="payload.iced ? 'coldDrinks' : 'hotDrinks'" />
```

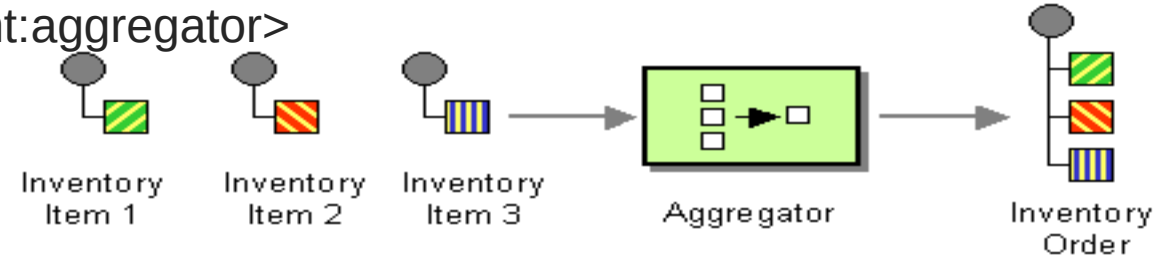


*路由器*使用来自一个*通道*的某条*消息*并根据一系列条件将其重新发布到另一个不同的*通道*中。

批处理

生产者和消费者可以在独立的时间运行

```
<int:aggregator
    input-channel="preparedDrinks"
    method="prepareDelivery"
    output-channel="deliveries" >
    <bean class="org.sf.integration.samples.cafe.xml.Waiter"/>
</int:aggregator>
```

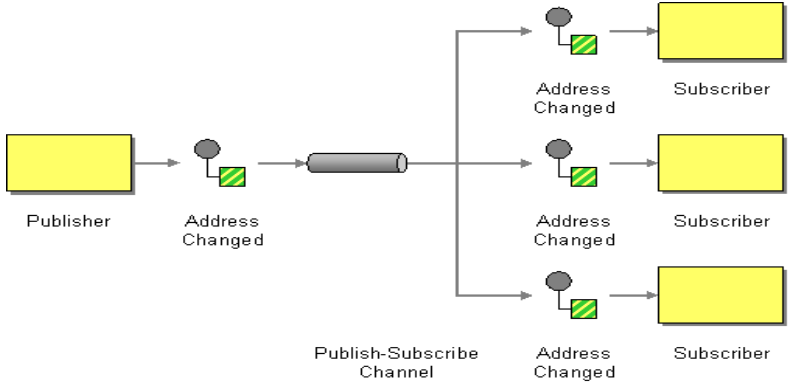


使用状态筛选器（一种*聚合器*）收集并存储经过拆分的消息，直到接收完一个完整系列的相关消息，然后发布一条从拆分消息提取得来的消息。

发布

告知想知道此消息的所有人

```
<int-event:outbound-channel-adapter
        channel="eventChannel"/>
```



发送到“eventChannel”通道的所有消息都将作为 ApplicationEvents 发布到在相同 Spring ApplicationContext 内注册的任何相关 ApplicationListener 实例。

模块性

部署、扩展分布式系统并控制其版本

本地

```
<service-activator
    input-channel="coldDrinks"
    ref="barista"
    method="prepareColdDrink"
    output-channel="preparedDrinks" />

<service-activator
    input-channel="hotDrinks"
    ref="barista"
    method="prepareHotDrink"
    output-channel="preparedDrinks" />
```

分布式

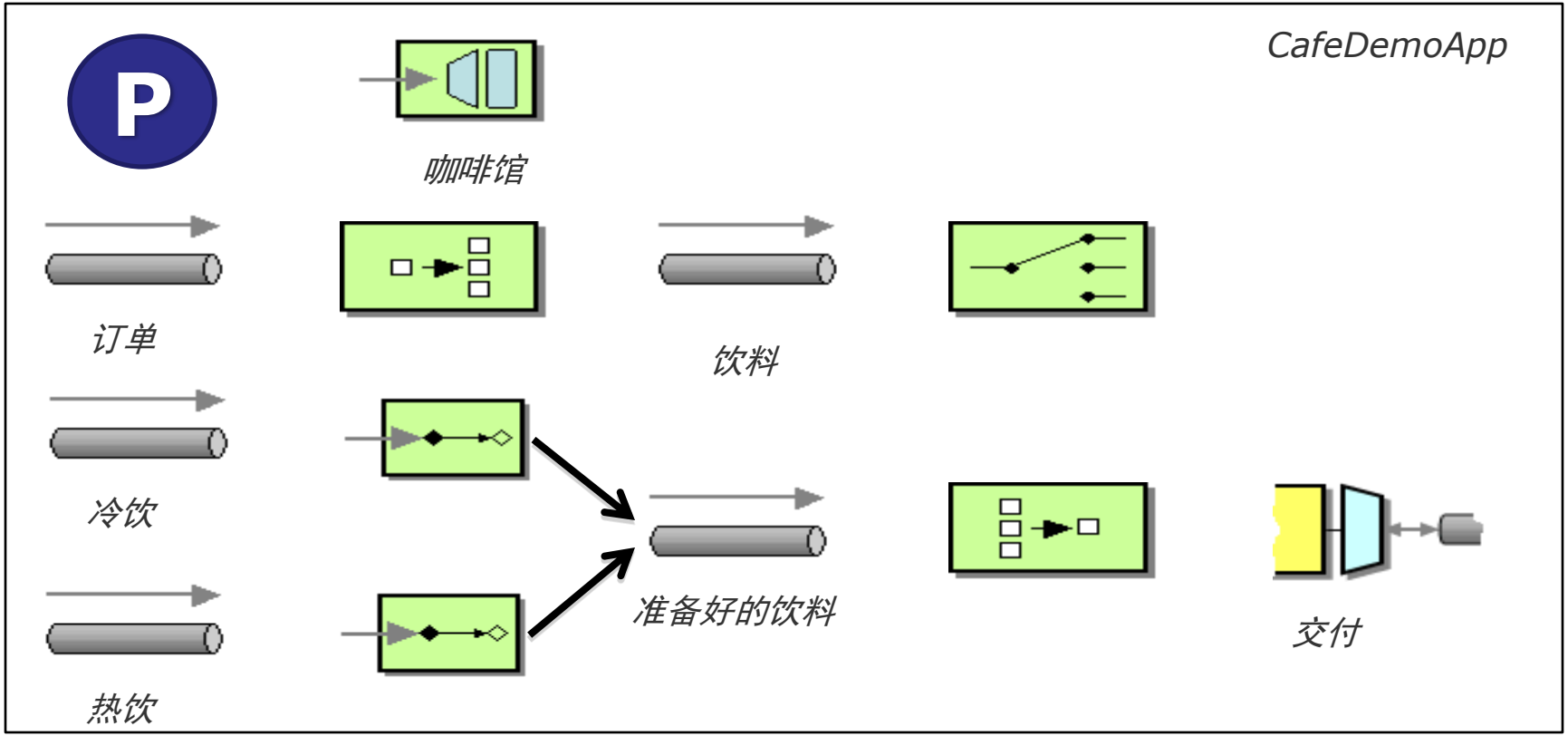
```
<int-amqp:outbound-gateway
    id="coldDrinksBarista"
    request-channel="coldDrinks"
    reply-channel="preparedDrinks"
    routing-key="ordered.drinks.cold" />

<int-amqp:outbound-gateway
    id="hotDrinksBarista"
    request-channel="hotDrinks"
    reply-channel="preparedDrinks"
    routing-key="ordered.drinks.hot" />
```

Spring 集成不会在测试开始前强制让您作出有关部署的最终决定。可随时通过配置方式独立地对各个模块进行版本控制、部署和扩展，无需编写代码。

演示 – Spring Integration咖啡馆

Spring Integration咖啡馆



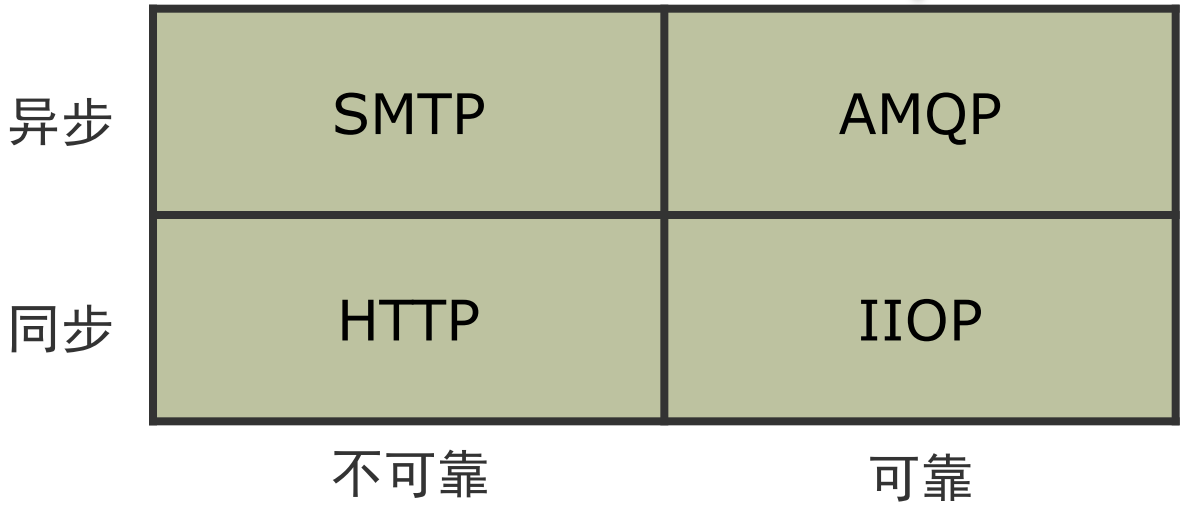
Spring Integration咖啡馆 – 简单 HTTP 部署

通过 AMQP 进行分发

高级消息队列协议

“与电子邮件类似，但您可以通过它汇款”

开源、应用广和适应范围广



为何选择 AMQP?

一种协议，而非 API

- **已定义的一系列消息传送功能**，称为 AMQ 模型
- **一种网络线路级协议**，AMQP

在商用硬件上

- 日常为每秒 1-2.5 万条消息 *
- 网卡通常是瓶颈

* 非持久性消息

为何选择 AMQP?

AMQP 安全性

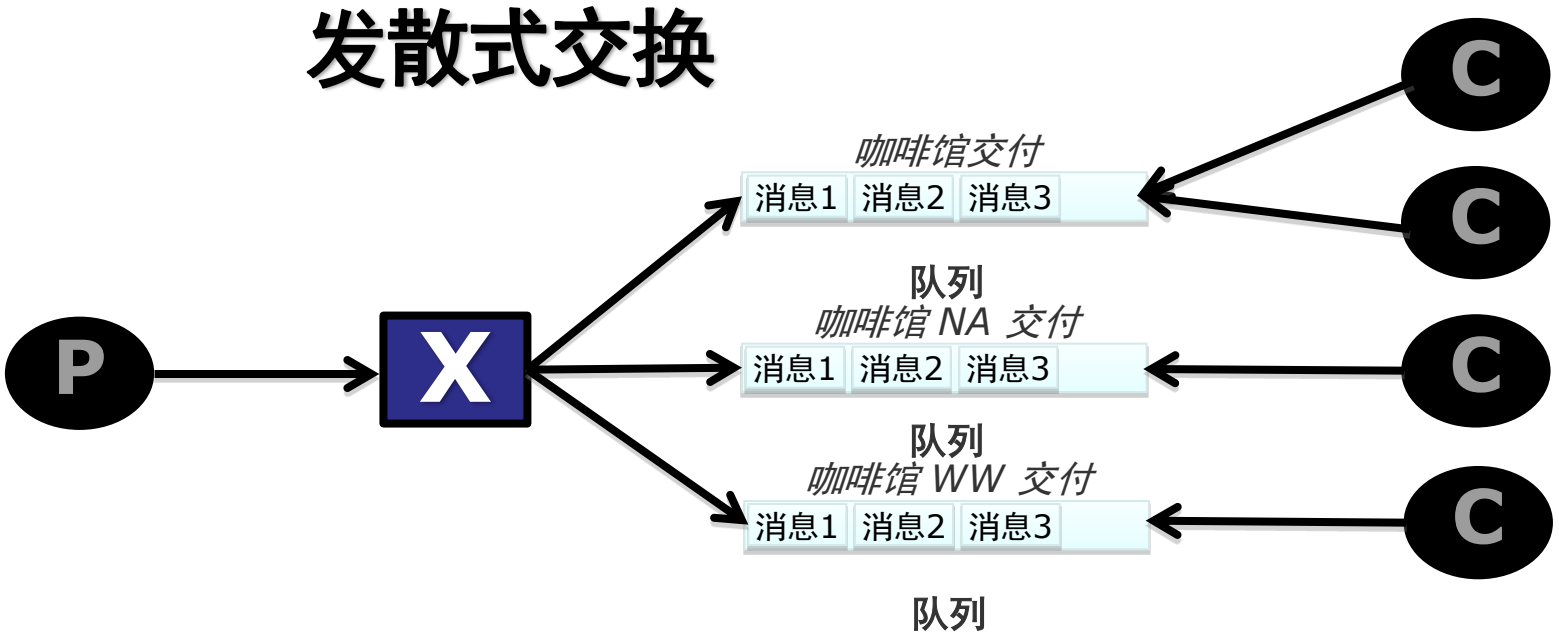
- 代理支持独立的虚拟主机
- 三个级别的权限
- 支持 AMQP over SSL

设计为可水平扩展

- 经常有几十个群集代理
- JPMorgan 每天发送 10 亿条 AMQP 消息

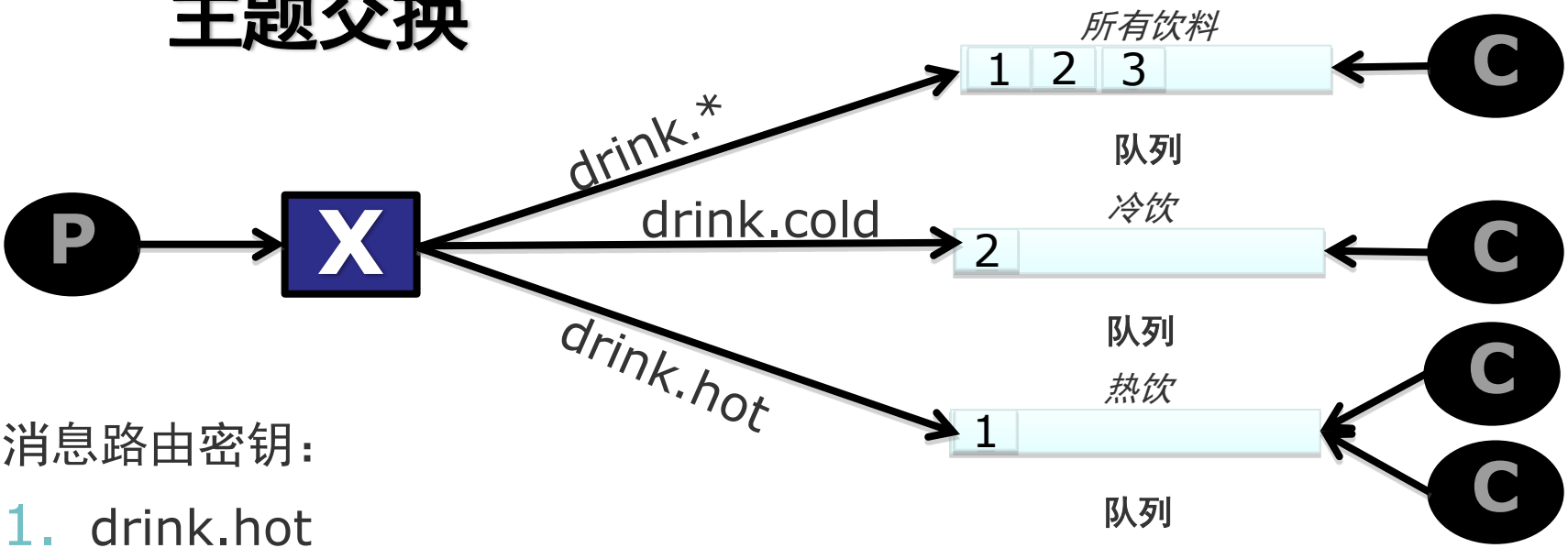
直接交换

AMQP 体系结构


发散式交换
P
X
咖啡馆交付
消息1 消息2 消息3
队列
咖啡馆 NA 交付
消息1 消息2 消息3
队列
咖啡馆 WW 交付
消息1 消息2 消息3
队列
C
C
C
C

AMQP 体系结构

主题交换

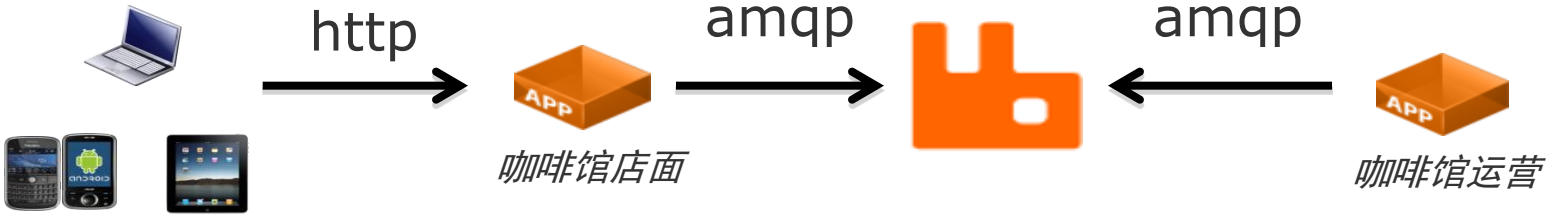


消息路由密钥：

1. drink.hot
2. drink.cold
3. drink.warm

演示 – 通过 AMQP 将 Spring Integration咖啡馆转成分布式演示

Spring 集成咖啡馆 – 通过 AMQP 进行分布式部署



通过 Apache Hadoop 实现统一

现代应用程序需要新的企业数据平台

- **方便数据采集**
  - 从所有来源收集数据 - 结构化数据和非结构化数据
  - 全速批处理、异步、流式传输、实时
- **综合数据处理**
  - 转换、优化、聚合、分析、报告
- **简化的数据交换**
  - 为便于查询和处理与企业数据系统实现互操作
  - 与分析应用程序共享数据
- **易于操作的平台**
  - 规模化置备、监控、诊断和管理
  - 可靠性、可用性、经济性、可扩展性、互操作性

企业数据平台要求

数据平台由一系列组件集成，供您以任何格式规模化地采集、处理和共享数据之用

- 实时并批量地存储和集成数据源
- 可处理和管理任何规模的数据，并且包含对数据进行排序、筛选、汇总和应用基本功能的工具
- 开启并促进与外部应用程序的数据交换
- 包含进行管理和操作以及了解性能并确保可用性的工具

Apache Hadoop

通过横向扩展存储和处理进行开源数据管理

处理

Map Reduce

- 跨“节点”分布
- 本机冗余
- 名称节点可跟踪位置

存储

HDFS

- 将任务拆分到“靠近”数据的各个处理器并汇集结果
- 自我修复的高带宽存储集群

Apache Hadoop 的特征

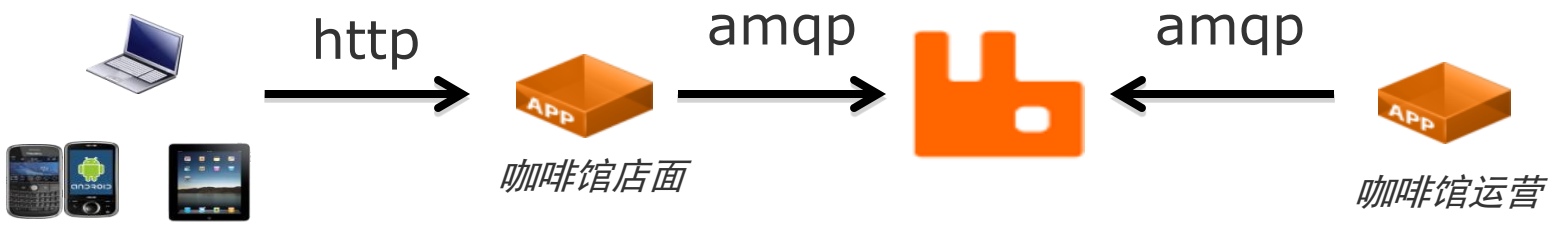
- **可扩展**
  - 可高效存储和处理 PB级别 的数据
  - 可通过添加处理和存储能力实现线性扩展
- **可靠**
  - 冗余存储
  - 可在节点和机架之间进行故障转移
- **灵活**
  - 可以任何格式存储所有类型的数据
  - 可在分析和共享数据时应用schema
- **经济**
  - 采用商用硬件
  - 开源软件可防供应商套牢

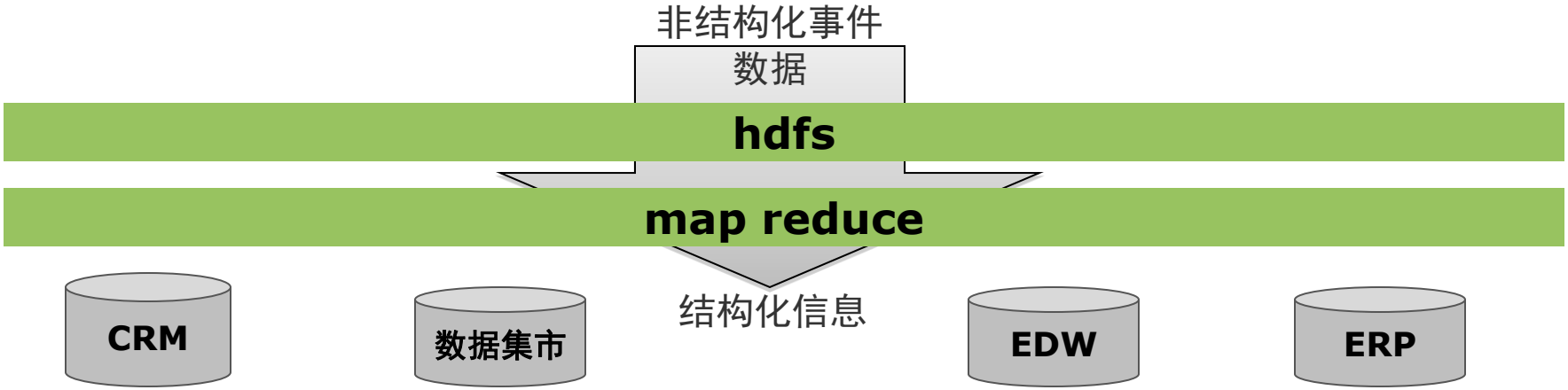
磁带故事

关系型	对比	Hadoop
写入时需要	schema	读取时需要
读取速度快	速度	写入速度快
标准化和结构化	监管	结构松散
受限制，无数据处理	处理	对数据实行耦合处理
结构化	数据类型	多种类型且非结构化
交互式 OLAP 分析 复杂 ACID 事务 操作型数据存储	最适合用途	数据发现 处理非结构化数据 海量存储/处理

演示 – 在混合云中通过 RabbitMQ 部署 Spring Integration咖啡馆演示

Spring 集成咖啡馆 – 通过 Hadoop 进行分布式部署





总结

- Spring 集成
  - 用于在您的 Spring 应用程序中引入消息传送抽象的 DSL
  - 从代码外部化消息传送和持久性概念
- AMQP
  - 一种 TCP/IP 协议，而非 API
  - 快速、可靠、开源、安全、可扩展
- Apache Hadoop
  - 面向大型企业的开源数据管理
  - 横向扩展存储和处理能力

vmware | SPRING & CLOUD FOUNDRY
开发者大会

谢谢您！