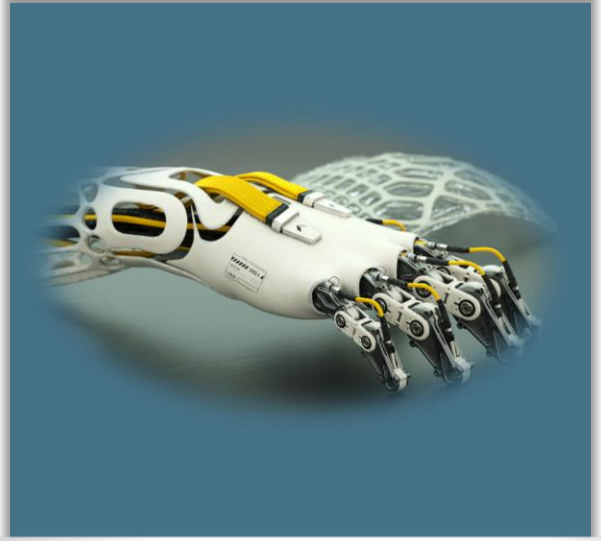
智能感知与物联网

授课人：王闻博

Email: wenbo_wang@kust.edu.cn

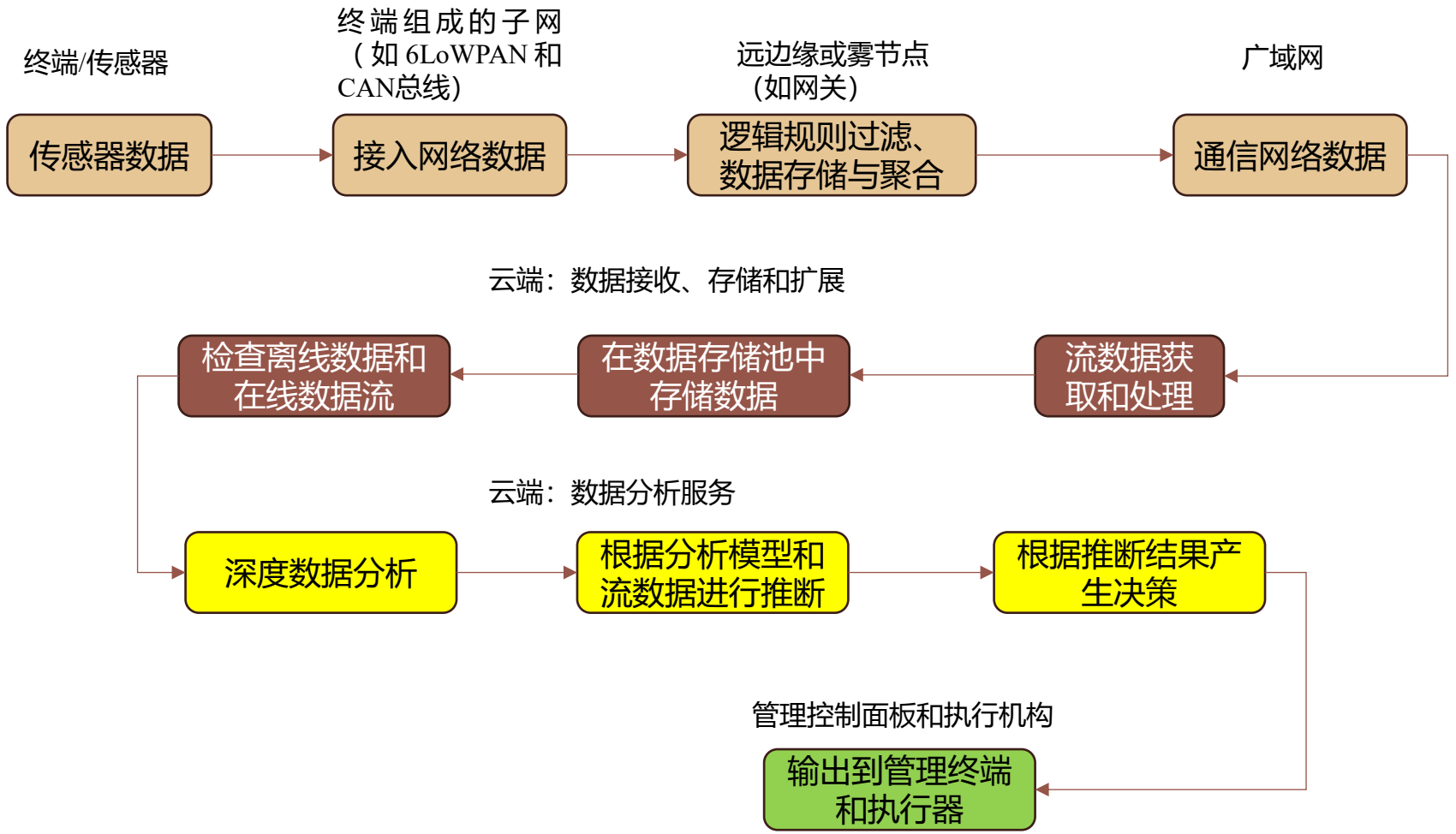
昆明理工大学 机电工程学院

2026年5月26日



第六章
物联网使能的智能制造

回顾：基于物联网大数据的分析-决策数据管道

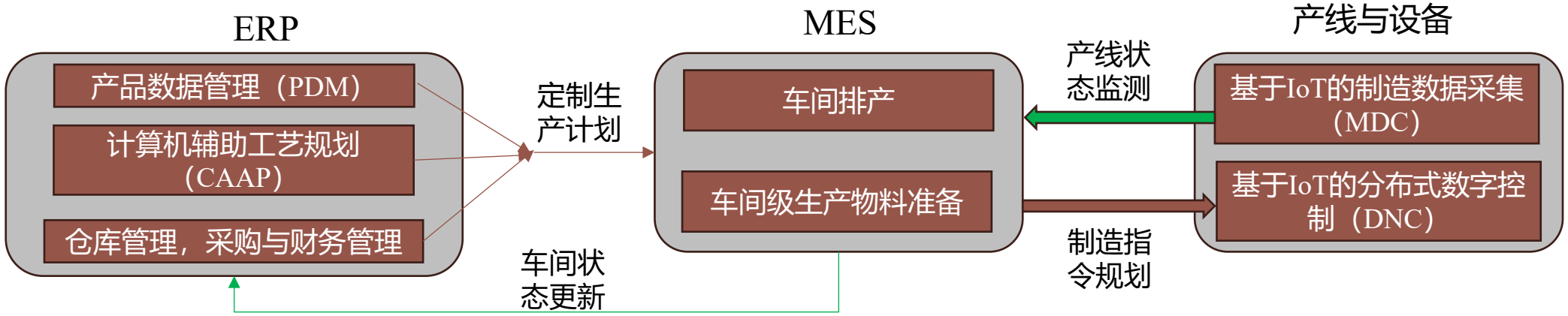


从边到云的数据分析-控制响应模块：

- **业务规则引擎：定义数据处理行为并产生执行结果。**
- **流数据处理引擎：基于流数据进行离线化注入（到数据库）和在线化处理（产线规划引擎）。**
- **数据分析引擎：进行模型的生成和训练，以及接入其他深度数据分析方法。**
- **信号与控制规划-回传：生成控制信号并回传到边缘端或执行机构。**

基于“人-信息-物理系统（Human-Cyber-Physical Systems，HCPS）的智能工厂”

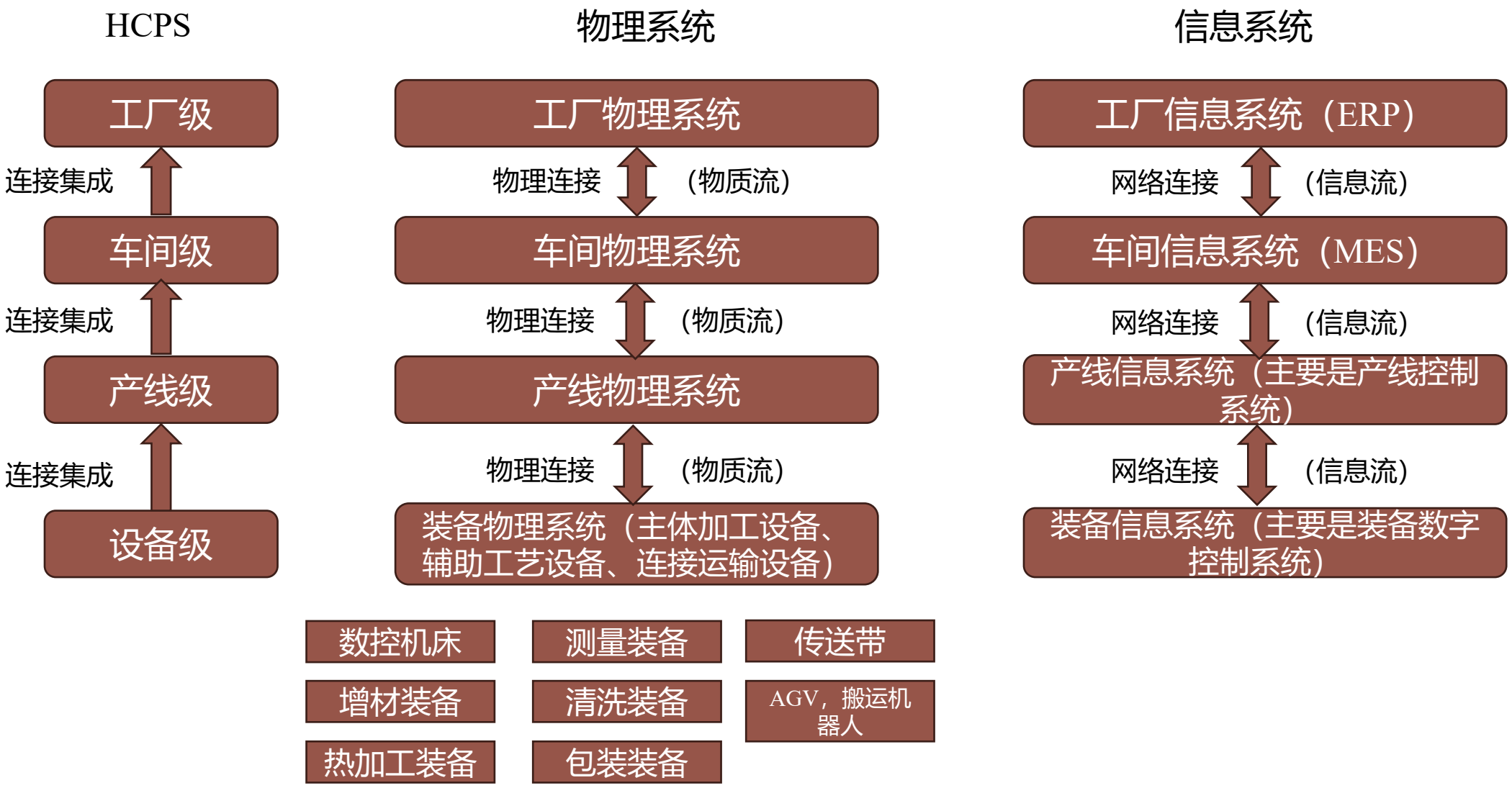
- **智能工厂（第二代，HCPS1.5）：**
  - **定义：将信息、网络、自动化、现代管理与制造技术结合，在工厂实现数字化网络化制造平台，从而改善工厂的管理和生产等各环节，实现敏捷制造。**
  - **智能工厂的子系统分层结构：**
    - **生产线层面：产线、设备管理与控制系统。**
    - **车间层面：制造执行系统（Manufacturing Execution System，MES）。**
    - **工厂层面：企业资源计划系统（Enterprise Resource Planning，ERP）。**



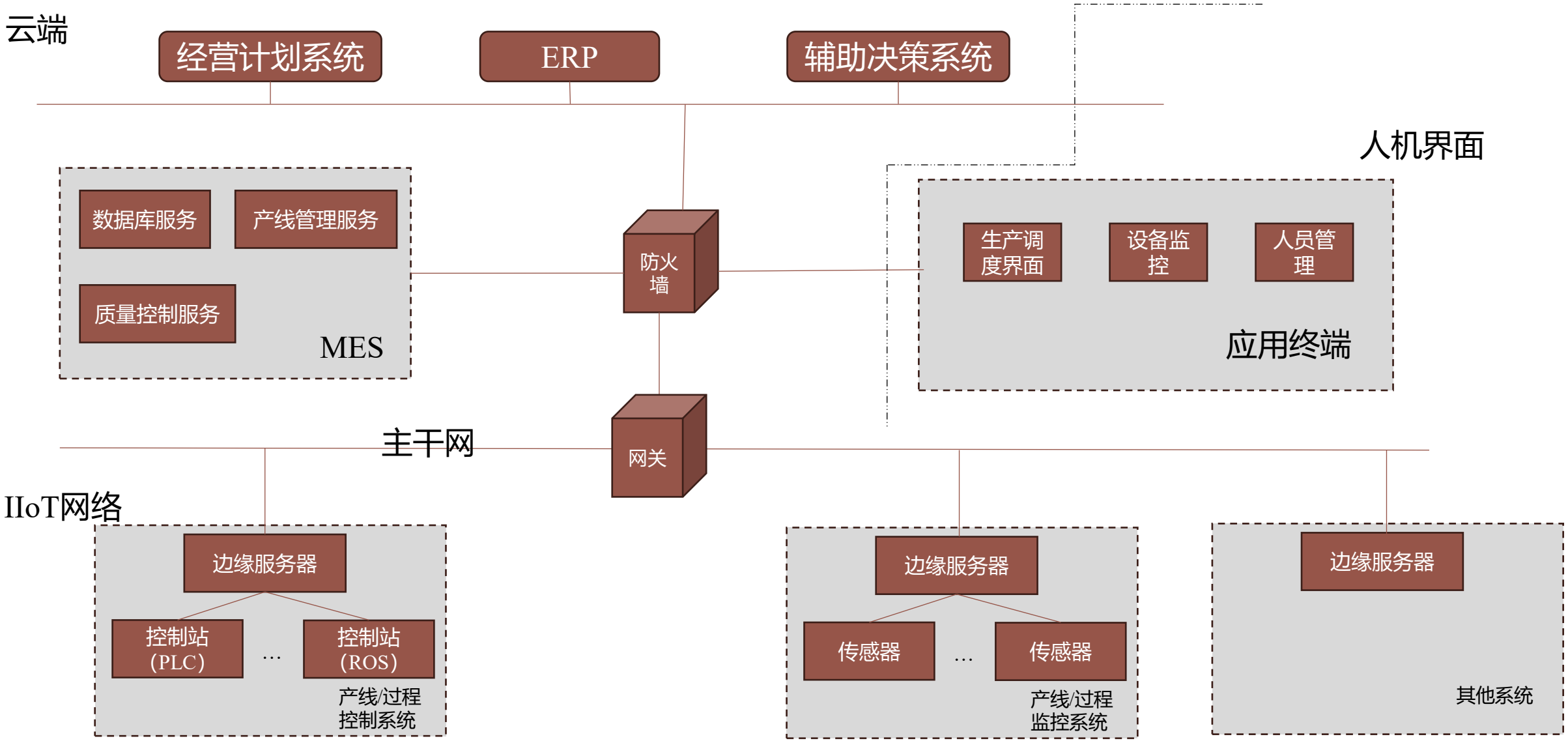
新一代智能工厂

- 智能工厂（第三代，HCPS2.0）：
  - 愿景：基于“人工智能+工业互联网+数字化制造”构建“数据+平台+应用”的模式。
  - 特点1：通过工业互联网打破企业在产品设计、加工、后处理、测试、仓储与物流等业务模块的“数据孤岛”。
  - 特点2：通过对生产过程中采集到的大量数据的分析与学习，自动优化生产过程、识别设备故障、保障设备健康。
  - 特点3：借助工业大数据，对企业生产经营、运行管理等数据进行综合分析，进行管理优化和流程再造。
  - 特点4：通过智能控制系统、工业IoT网络、信息安全技术等在智能仓储、先进制造的应用，使能企业在供应链管理、质量管理、商业模式上的创新。

数字化工厂的HCPS架构



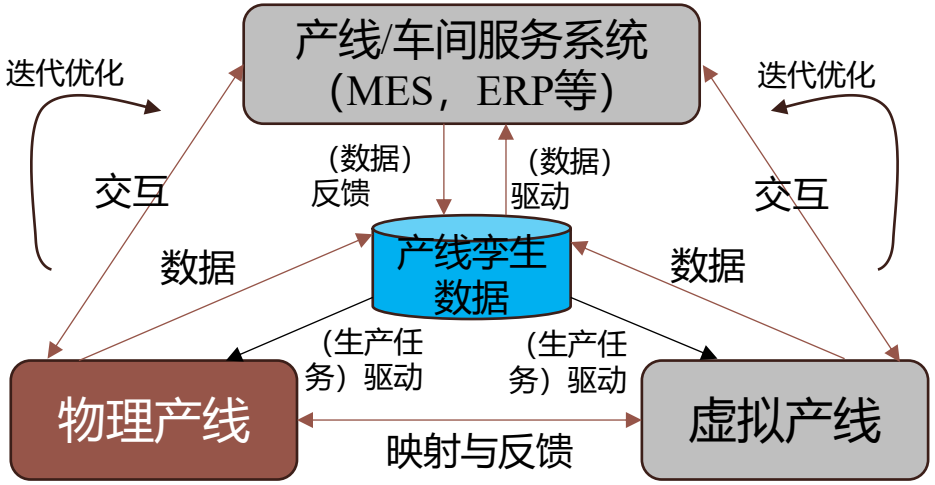
云-网结合框架下的工业HCPS系统



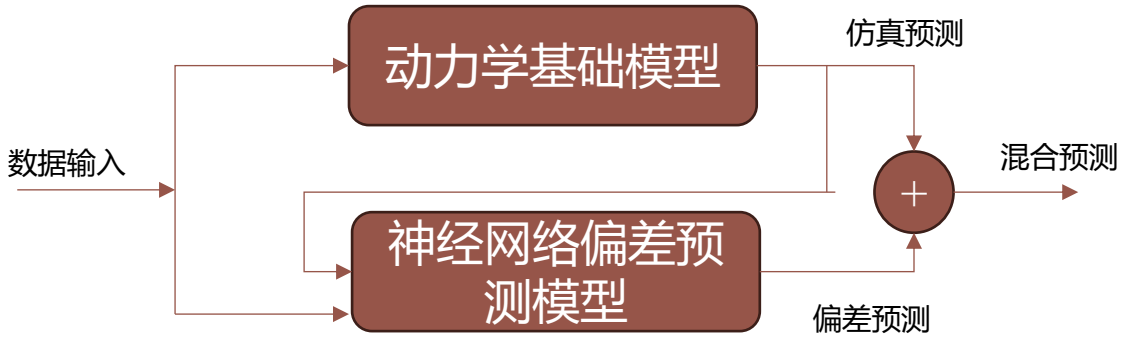
某工厂级信息系统部署案例

智能工厂的数字孪生

- **产线数字孪生的建立：构建三级数理仿真模型**
  - **几何仿真：根据装备尺寸和连接关系，按产品工艺流程和布局进行产线的建模、仿真和优化；**
  - **运动学仿真：模拟加工过程中机床、刀具、工装等设备的动作，评估运动行为和生产布局；**
  - **动力学仿真：考虑设备加工过程中的物理化学反应，对产品的成型进行动力学层面的建模和描述。**
- **数据驱动的产线建模**
  - **利用数据挖掘描述对象的运行规律和相关模式，通过深度学习、强化学习等模型进行系统的无机制模型式描述。**
- **数据机理混合建模**
  - **串联型建模：先建立机制模型，再通过机器学习等方法对机制模型中的未知参数进行估计；**
  - **并联型建模：将机理模型和基于数据的误差补偿方法结合构造参数预测模型。**



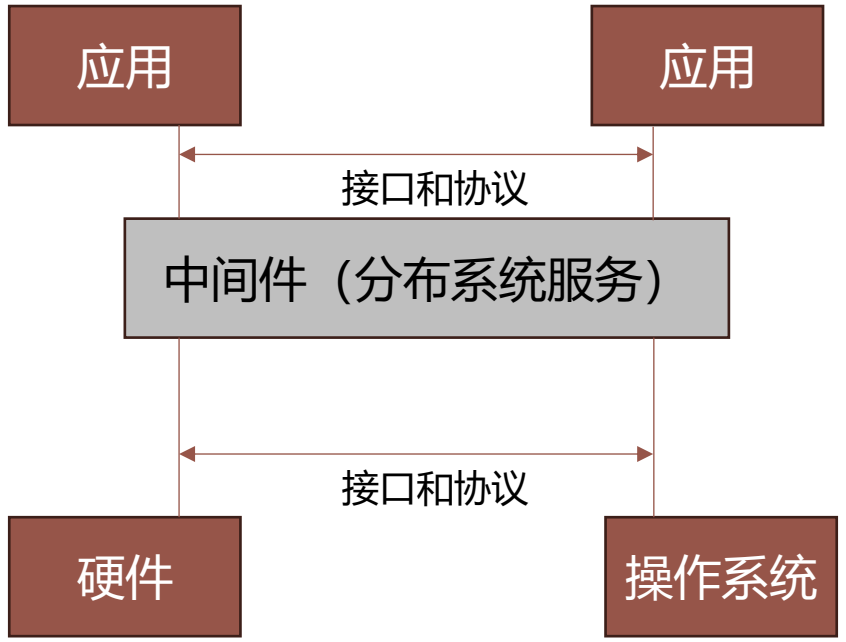
产线数字孪生框架



并联混合建模框架

物联网在智能工厂CPS系统中的位置：“中间件（Middleware）”

- **中间件的通用定义**
  - **中间件的引入目的：为调和系统软、硬件体系结构的多样性和系统网络节点的异构性（屏蔽底层硬件、操作系统及网络协议的异构性）；为保证数据的传输、导入/导出（到管理系统）、以及数据融合等需求；为分布式系统提供安全性。**
  - **常见定义：中间件是一类位于操作系统与应用软件之间的独立软件系统或服务程序，分布式应用系统借助这种软件，可实现在不同应用系统之间的资源共享。中间件是位于平台（硬件、操作系统）和应用之间的通用服务，为各应用之间的互联提供标准化程序接口和交互协议。**
  - **中间件的主要组成：平台+通信。**



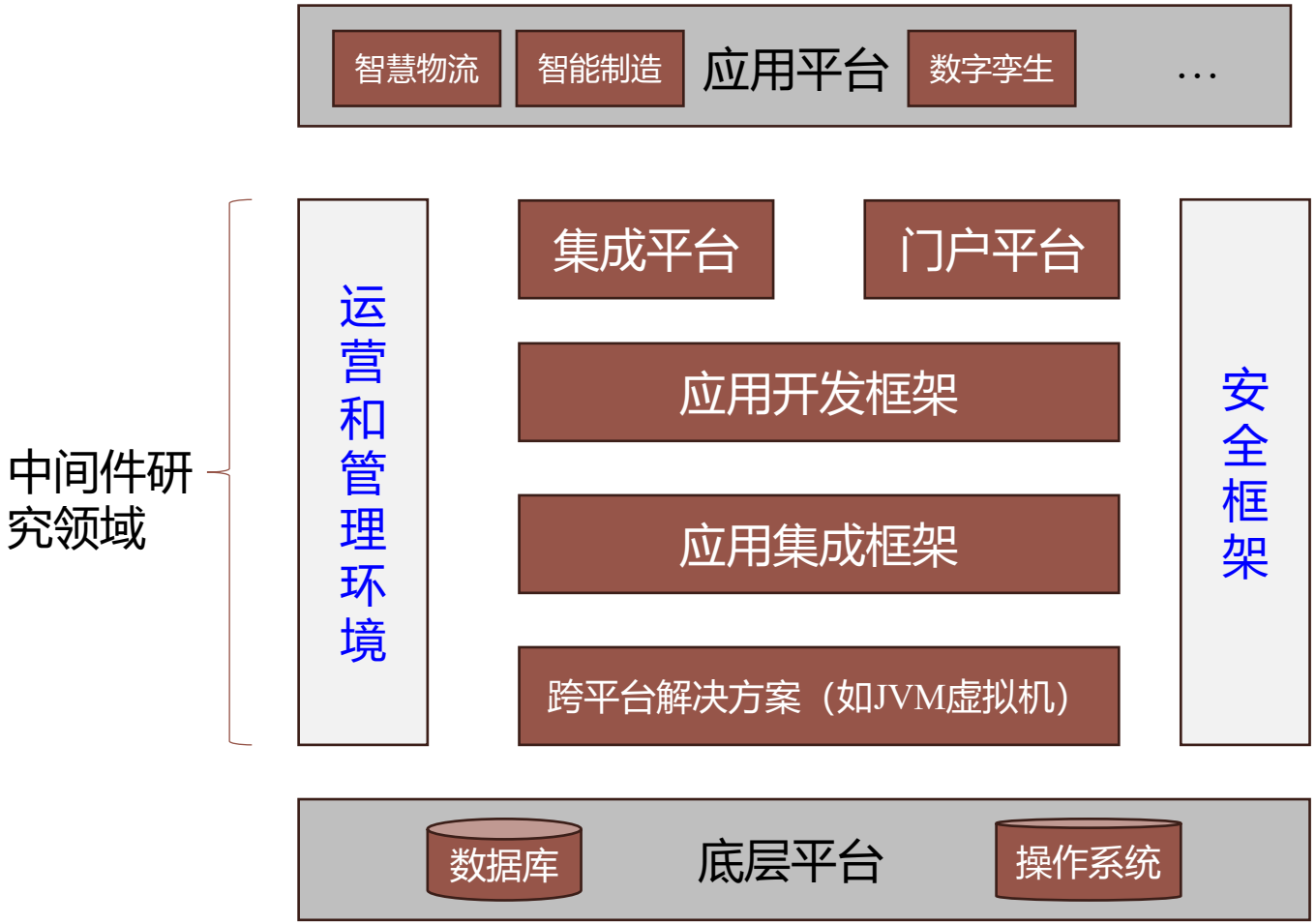
物联网作为中间件的技术特征

- **支持多种构架**
  - 中间件可以是独立的，也可以是非独立的，独立中间件介于数据和应用之间，完成多个数据源和后台（云端）应用程序的连接。
  - 举例：Kepware是一种常用的工业连接中间件（平台+通信），通过提供组合驱动程序以及支持多个协议（如OPC），帮助企业连接各种自动化设备硬件和应用系统。
- **数据流处理**
  - 数据处理是中间件的最重要特征之一，中间件应具有数据收集、过滤、整合传递等特性，以便最终将正确信息传递到企业后端业务应用，如ERP和MES。
- **支持过程流**
  - 中间件常采用程序逻辑和存储再传（store-and-forward）的功能来提供顺序的消息流。它应具有数据流设计和管理的能力。

物联网作为中间件的技术特征（续）

- **支持多种数据协议标准**
  - 为提供多种分布式软件模块之间的互操作性，提供系统集成，中间件应提供对多种数据协议（如编码标准）的支持，并具有数据整合和集成的能力。
- **提供安全特性**
  - 中间件应为业务实现提供必要的安全功能，如安全管理、安全监控、安全审计等。
- **提供应用开发环境和应用适配接口**
  - 中间件应提供基于应用框架接口的开发环境，为各种应用业务提供必要的编程接口、工具集和设计模式。
  - 中间件应提供必要的业务接口定义和封装，用来解决不同基础或应用软件层接口不一致的问题。

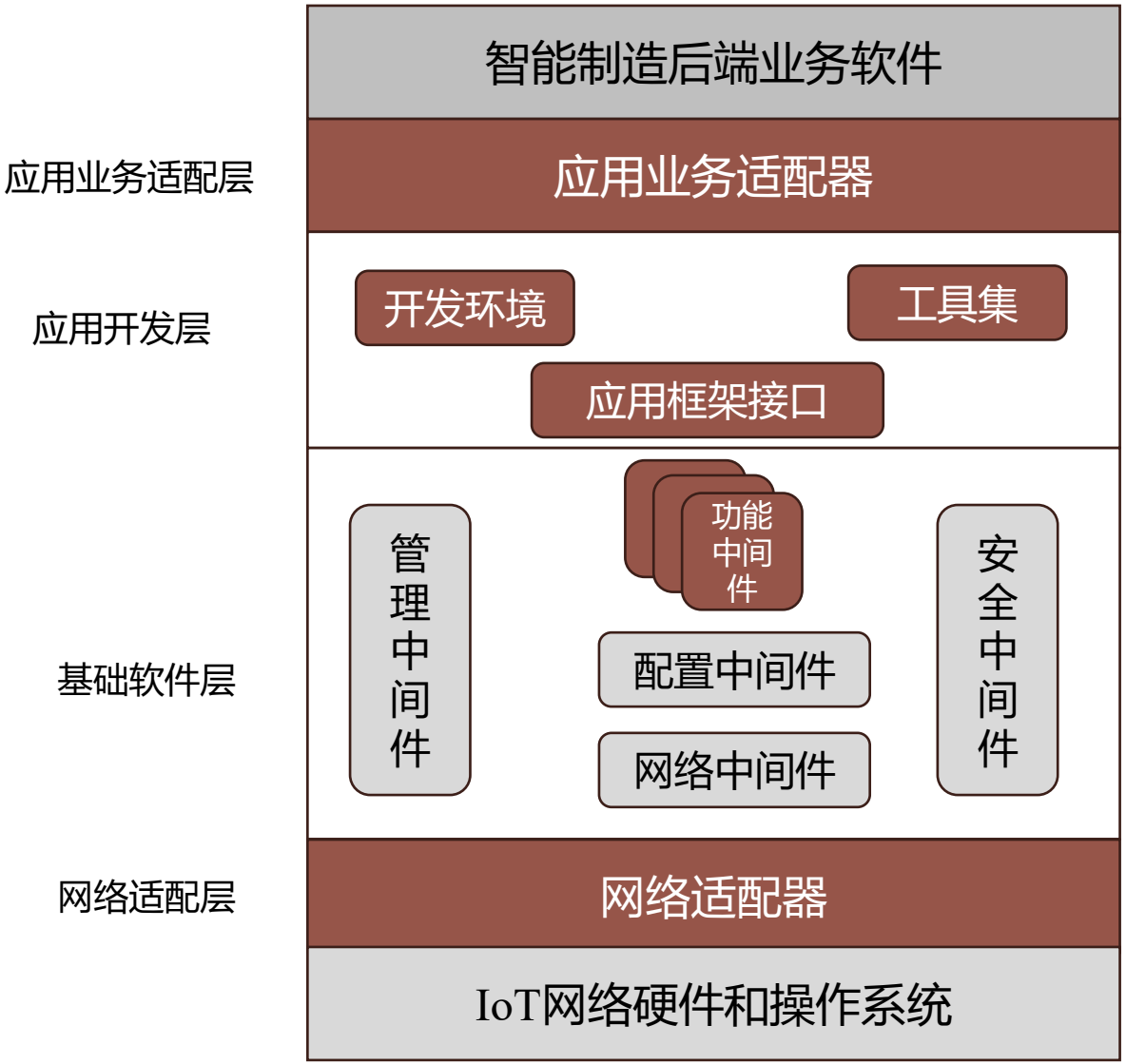
物联网中间件的应用领域和研究方向



- **历史上，通用中间件曾被作为国家863计划计算机软硬件领域“六个一”发展目标之一，直到当前依然是颇具挑战的技术方向之一**
  - **一芯**：掌握计算机芯片设计核心技术。
  - **一机**：与网络相连的各种智能电器。
  - **一器**：高性能网络服务器。
  - **一网**：下一代互联网，研制未来互联网的关键技术平台。
  - **一件**：即网络中间件（在分布式系统中起着至关重要的作用，管理计算资源和网络通信，实现资源共享和功能共享）。
  - **一台**：智能化人机交互平台，代表应用如语音识别、自然语言处理、脑机接口等。

物联网中间件的平台体系结构

- **（接上页）IoT中间件可分为四个层次（按作用分类）：**
  - **网络适配层；**
  - **基础软件层；**
  - **应用开发层；**
  - **应用业务适配层。**



中间件按技术和作用分类

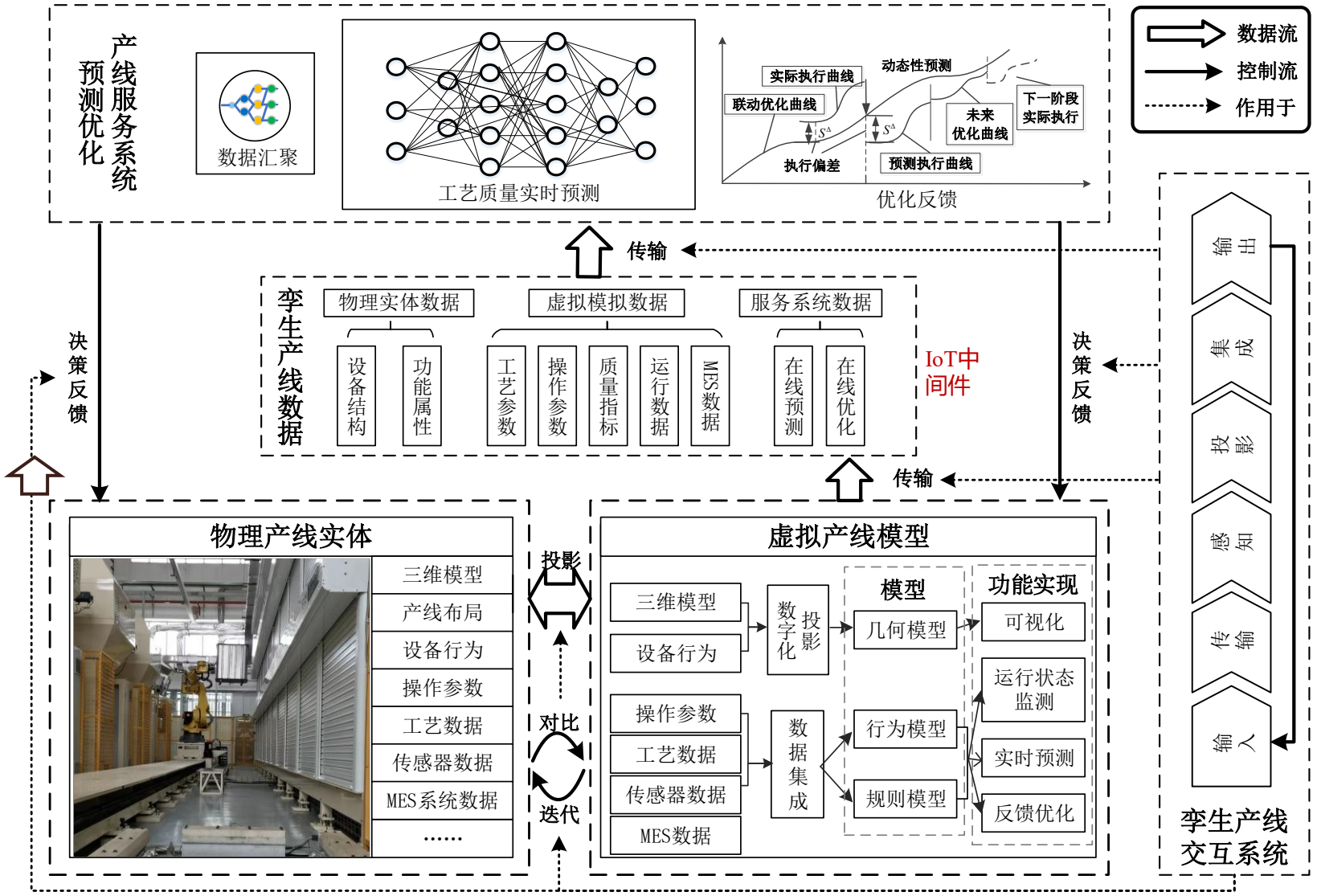
- **数据访问中间件（如ODBC、JDBC）**
  - **Data Access Middleware：**以**数据资源访问抽象**为中心；在系统中建立数据应用资源的互操作模式，实现异构环境下的数据库连接或文件系统连接。数据访问中间件是应用最广泛、技术最成熟的一种。在这种方式中，数据库是信息存储的核心单元，中间件主要提供**统一的 API 和驱动管理，并完成分布式事务支持和通信的功能。**
- **远程过程调用中间件**
  - **Remote Procedure Call (RPC)：**以**计算/服务过程调用抽象**为中心；应用程序使用RPC来**“远程”执行一个位于不同地址空间里的过程，隐藏网络通信的复杂性使得其与本地调用看起来无差别。**RPC中间件在客户机/服务器(C-S)应用方面，在技术上比数据访问中间件更进一步：它不仅关注数据的传输，还关注如何在分布式系统中执行特定的功能或过程。

中间件按技术和作用分类（续）

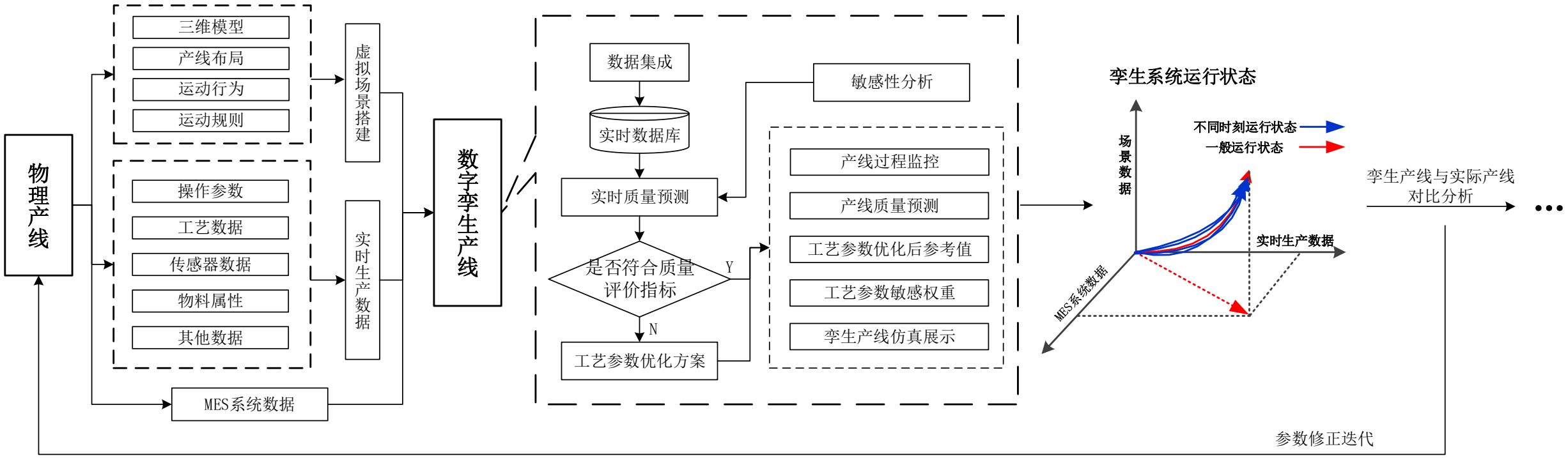
- **面向消息的中间件（Message-Oriented-Middleware，如IBM MQ、RabbitMQ 、Kafka）**
  - **Message Oriented Middleware (MOM)**：提供异步、松耦合的通信机制，通过消息代理（broker）实现消息的存储和转发，进行与平台无关的数据交流，并基于数据通信进行分布式系统的集成。**通过消息传递和消息排队模型，中间件可在分布式环境下扩展进程间的通信。它提供平台无关性、异步通信和与网络复杂性隔离。**
- **面向对象中间件（Object Oriented Middleware，如CORBA、DCOM）**
  - **Object Oriented Middleware (OOM)**：将面向对象编程模型扩展到分布式异构环境，允许客户端像调用本地对象一样调用远程对象的方法。它为在异构的分布式计算环境中进行对象请求的传递提供了一种通信机制。**这种中间件的主要功能是实现不同对象之间的互操作，从而完成系统的快速集成。**
- **事件处理中间件（事件总线，如Apache Kafka、AWS EventBridge、Google Cloud Pub/Sub）**
  - 事件处理中间件在分布、异构的环境下提供交易完整性和数据完整性的环境平台，并针对分布式应用的速度和可靠性要求进行了优化，确保在高并发、大数据量的环境下仍能保持高效的性能。**事件处理中间件提供了一套事件处理的API**，它以事件为驱动单元，构建松耦合、高度可扩展的分布式系统。**这些API通常包括事件的发布、订阅、处理等功能，使得开发者能够轻松地处理分布式环境中的各种事件。**

将物联网系统中间件嵌入到产线数字孪生系统:一个案例

- **烟丝切割产线的数字孪生实现:**
  - 基于IoT网络实现了网络适配（MQTT到OPC/ OPC UA）以及应用适配（产线状态预测服务+数字孪生）——**协议网关中间件/集成中间件**;
  - 基于IoT网络提供的数据接口开发出一套以深度学习为基础的产线质量预测模型;
  - 基于IoT系统实现了MES系统和产线服务系统的连接。



数字孪生案例（续）：数字孪生产线数据传输过程



作业（2026-06-02）

前章（云计算）作业：

- **题1：简述工业物联网从边缘到云端的软件结构框架，分别从数据流向和层次化服务模块两个角度入手。**
- **题2：结合课本第8章以及课件内容，简述联邦学习在云-边协同框架下的训练规则和数据流（画出数据流图）。**
- **题3：对比MQTT和CoAP协议，列出它们在参与者、数据请求模式、以及信息确认方法等方面的异同。**
- **题4（编程实现）：在个人电脑上尝试部署MQTT Mosquitto服务，将其配置在localhost（127.0.0.1）：1883。使用Python的paho库编写发布者和订阅者的服务端，以默认QoS服务等级发布消息。要求：至少发布一则消息“Hello MQTT！”到test/topic下，发布-订阅结构为{“id”: “随机整数”}的消息10条到同一自定义topic，并以JSON形式在订阅者当前目录存为“result.json”文件。（程序及输入-输出结果以打印文档的形式提交）**

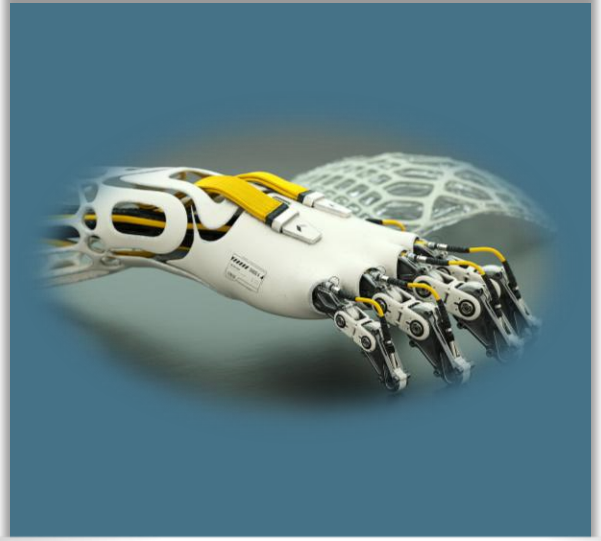
本章作业：

- **题1：简述物联网中间件为何产生，以及它的作用、分类分别是什么？（请通过搜索、查阅相关资料，在课件基础上做答。）**
- **题2：画出物联网系统数据-信息流处理的“端到云到端”模块图，并尝试解释普通“数据库”和云端“数据池/数据仓库”之间的差别。**
- **题3：简述数字孪生中机制建模和数据驱动建模的差别。**

结语（Q&A）

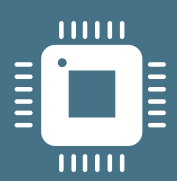
- **参考文献**

- [1] 周济等，智能制造导论，高等教育出版社，2021.
- [2] 黄玉兰，物联网概论，人民邮电出版社，2018.
- [3] 郭斌等，智能物联网导论，机械工业出版社，2022.
- [4] 王兴伟等，工业互联网，科学出版社，2024.
- [5] 褚华，霍秋艳主编，软件设计师教程，清华大学出版社，2018.



附录

面向物联网系统的流式数据处理技术




附录

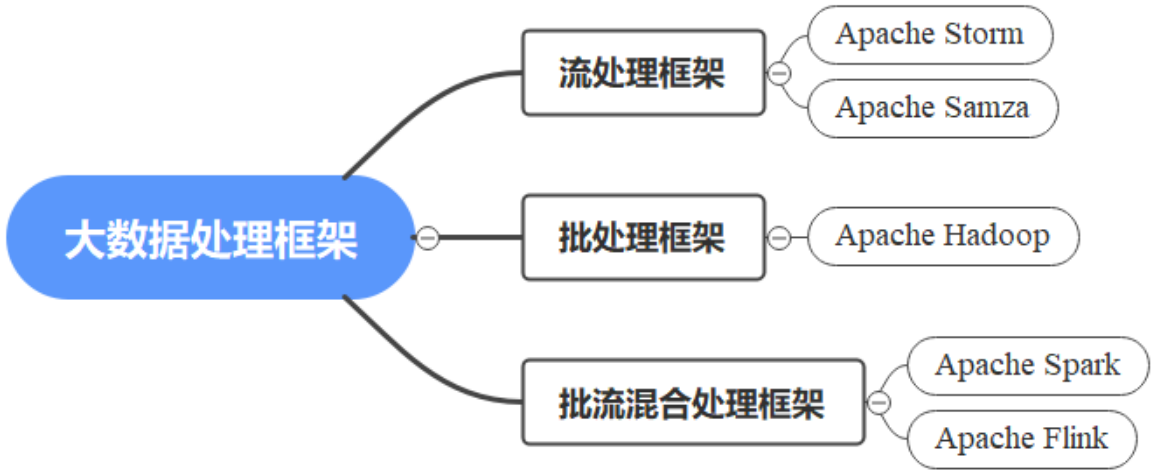
1 流处理概述

2 流处理模型

3 流处理引擎架构

流处理概述

- **大数据处理框架（Big Data Processing Framework）**
  - **定义**
    - 大数据处理框架是一种软件平台，它为跨计算机集群的分布式大规模数据集处理提供抽象层及工具集。
  - **分类：**如右图，大数据处理框架可分为
    - **批处理框架；**
    - **流处理框架；**
    - **批流混合处理框架。**

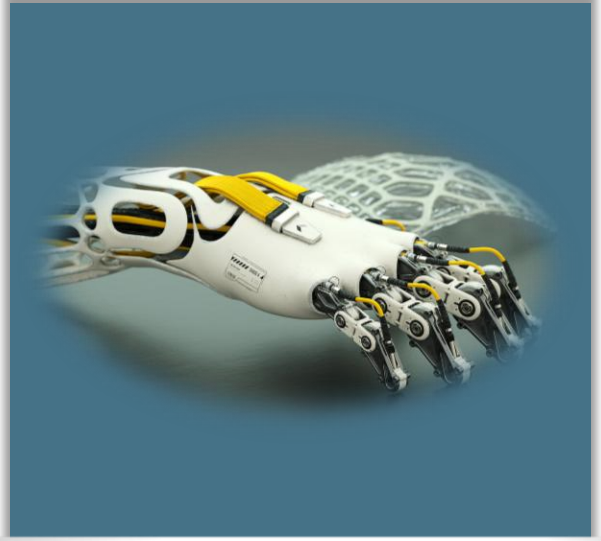


流处理概述（续）

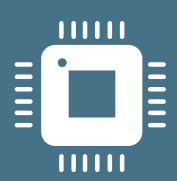
- **流处理与批处理**
  - **流处理（Stream Processing）**
    - 流处理是用于从“无限数据”中提取信息的规范和技术集合。
    - 我们的IT系统建立在有限资源（如内存和存储等）的硬件之上，因而不可能容纳无限的数据集。因此，以事件流的形式来观察处理系统接收到的数据，并称这种数据为**数据“流”**。
    - 流处理器理论上是永不停止的。
  - **批处理（Batch Processing ）**
    - 批处理是针对有限数据集的计算分析，其处理时间是有限的。

流处理概述（续）

- **流处理的应用场景**
  - **设备监控**
    - 一家小型创业公司推出了基于云的物联网设备监控器，用于收集、处理、存储多达1000万台设备的数据。其在程序中多处用到了流处理器，包括利用内存存储来实时更新仪表盘，以及持续的数据聚合，如去重统计、最大值、最小值等。
  - **故障检测**
    - 一家大型的硬件制造商通过构建复杂的流处理管道来接收设备的指标数据。并通过时序分析来检测潜在的故障，将纠错方法自动发送给设备。
  - **总结**
    - 诸多不同的应用场景的共同之处在于处理数据并在数据接收后短时间内产生有用的洞察结果。因此在任何情况下，数据都是越快消费越好。



附录
面向物联网系统的流式数据处理技术




附录

1 流处理概述

2 流处理模型

3 流处理引擎架构

流处理的一般模型

- **数据源与接收器**
  - 数据源（Data Source）:是流式数据的起点，负责提供实时的数据流，这些数据以无界序列的形式存在。
  - 接收器（Data Sink）： 是流式数据处理的终点，负责将经过计算或转换后的数据输出到外部系统，作为流处理流水线中数据落地的关键环节，直接影响结果的存储、展示或下游系统的消费。
  - 作用
    - 数据源和接收器定义了系统的边界。
    - 在实际情况中，可能将Apache Spark集群连接到其他的分布式系统（例如Kafka）。此时一个框架的接收器就是下游框架的数据源。这个链条通常称为**管道（Pipeline）**。

图：简化的流处理模型

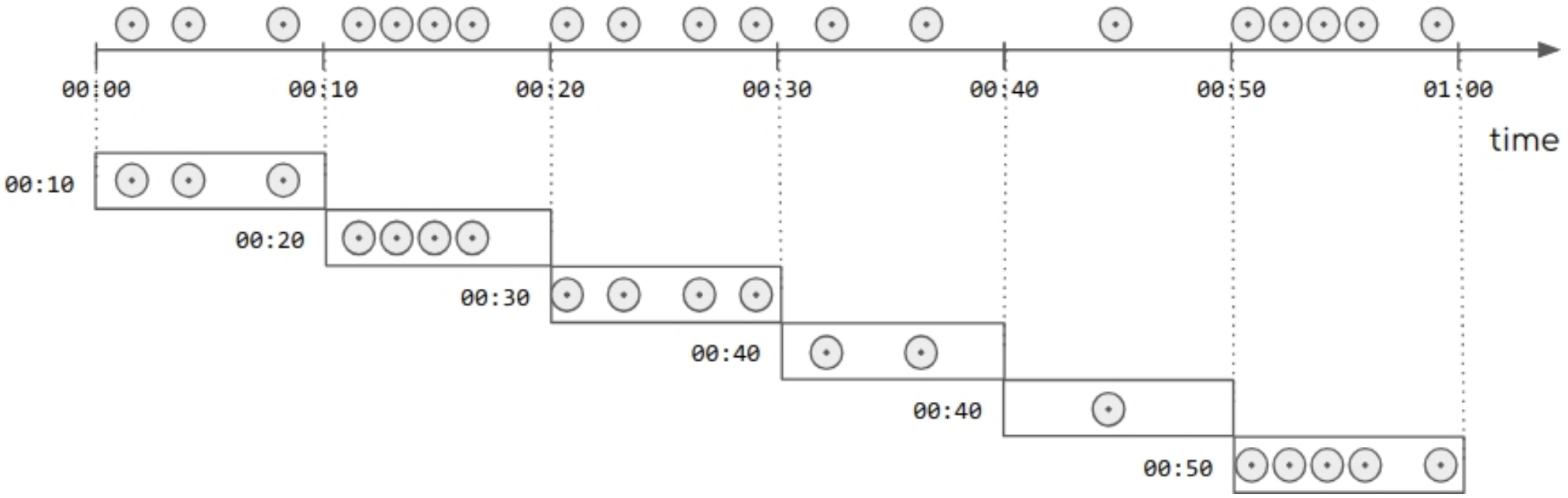
图表来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O'Reilly Media, 2019.

流处理的一般模型（续）

- **窗口聚合**
  - **动机**
    - 流处理系统通常用于处理实时发生的行为，其基本思想是系统长期运行并处理连续的数据流。随着新事件的到来，时间线上之前的事件会逐渐与当前的处理逻辑无关。
    - 单独处理最小时间粒度下的每个事件既不实用也缺乏意义，因为原始数据颗粒度过细，无法直接体现实际价值。
  - **定义**
    - 窗口（Windows）：将无限持续的流式数据按特定规则动态划分为有限的数据集合。
    - 窗口聚合：将窗口中的元素通过某种映射得到结果，该结果是窗口中的元素共同作用的结果。

窗口聚合（续）

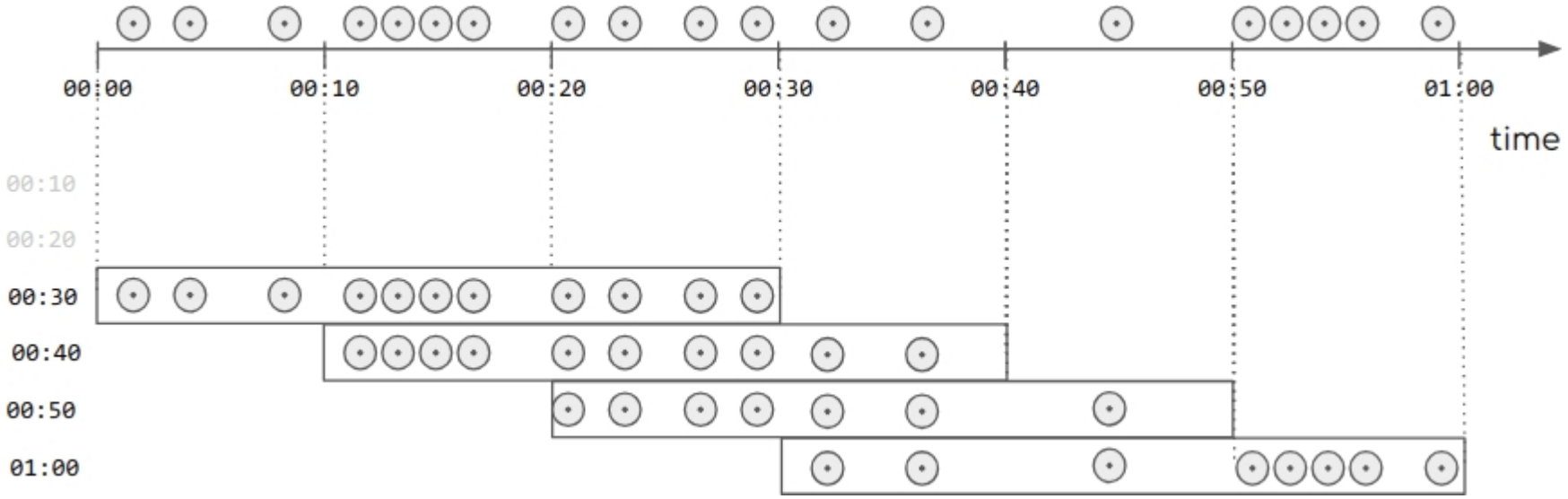
- **滚动窗口（Tumbling Windows ）**
  - 对于固定时间段的分组方式，每个分组紧接着上一个分组且无重叠，称为滚动窗口。
  - 下图展示了长度为10秒的滚动窗口，描绘了滚动窗口的滚动特征。



图来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O'Reilly Media, 2019.

窗口聚合（续）

- **滑动窗口（Sliding Windows ）**
  - 由两个时间定义：窗口长度（Window Size）和报告频率（Slide Size，每一次滑动的长度也称滑动步长）。
  - 滑动窗口与窗口内的数据平均函数组合起来就是最常见的滑动窗口形式：移动平均。
  - 下图展示了一个窗口大小为30秒，报告频率为10秒的滑动窗口。



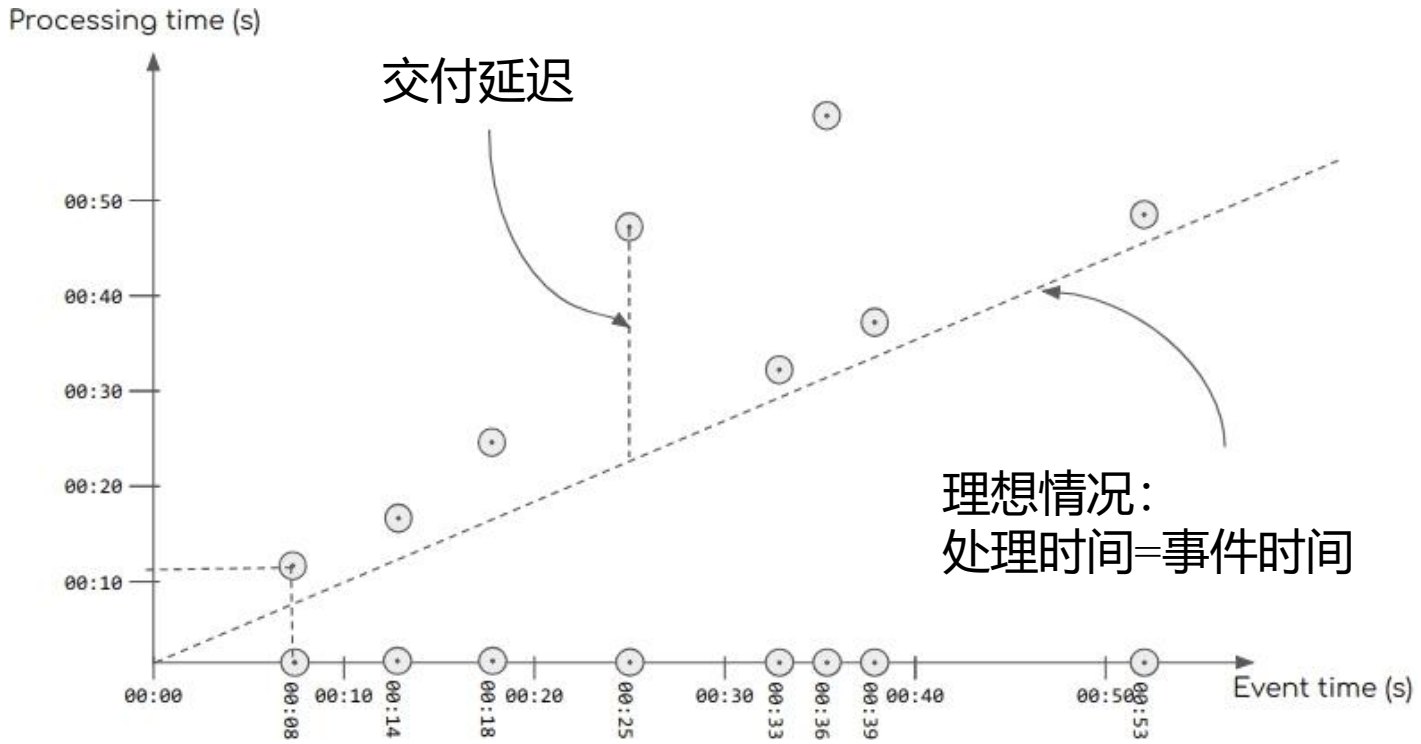
图来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O'Reilly Media, 2019.

流处理模型

- **有状态的流处理**
  - **定义**
    - 基于曾观察到的数据来理解新的数据，称为有状态的流处理。它作为一种规则，会根据输入数据流中观察到的数据计算新元素，并刷新相关内部数据来协助计算。
  - **例子：数控机床刀具磨损**
    - 传统方法刀具断裂后停机；
    - 有状态方案根据历史数据分析负载-电流波形相关性，提前2周检测轴承磨损，从而有效降低维护成本。
  - **局限**
    - 有状态流处理需要更多的计算资源来产生结果：
    - 需要遍历整个流，并在每一步保存中间值。

流处理中的时间概念

- **事件时间与处理时间**
  - 事件时间指的是最初事件生成的时间，处理时间指的是流处理系统处理事件的时间。



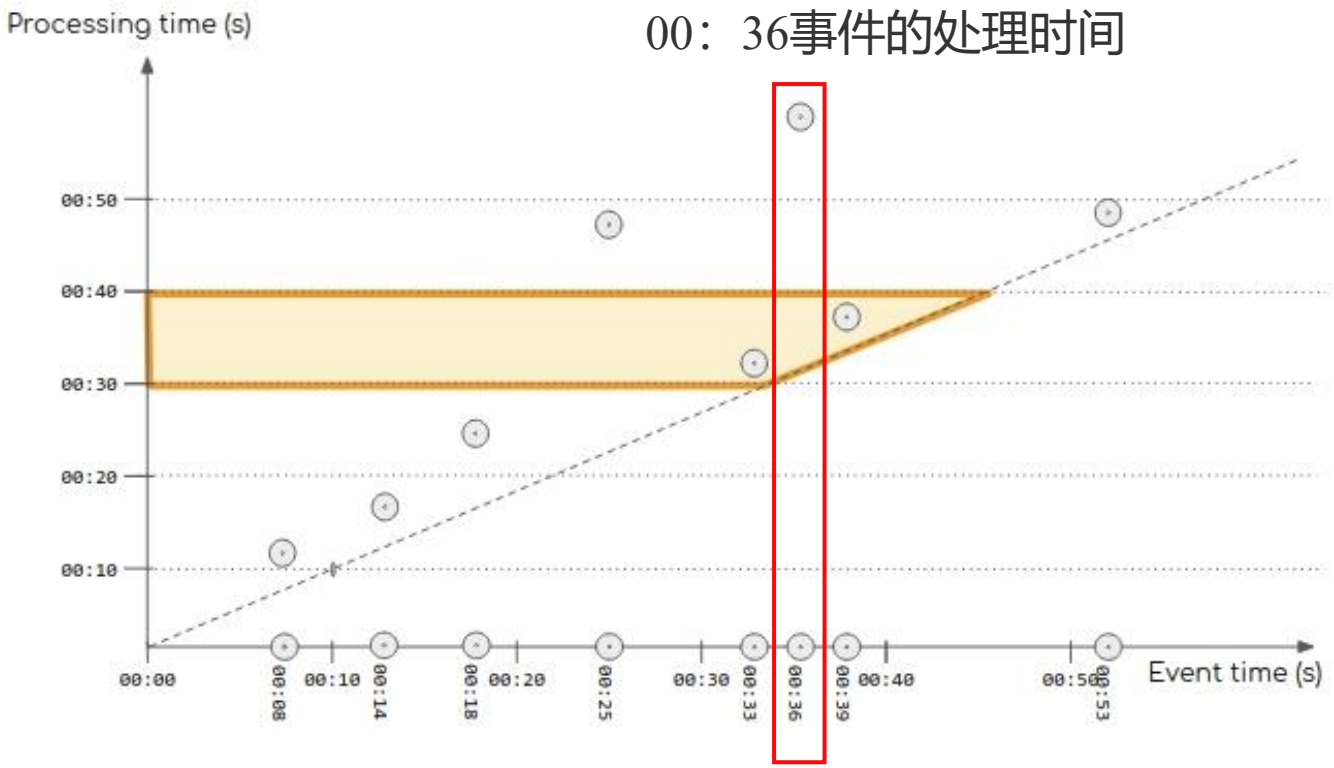
左图：

- X轴表示事件时间，轴上的点表示每个事件生成的时间；
- Y轴表示处理时间，图中的每个点表示对应X轴上的相应事件被处理；
- 对角线表示最佳处理时间；
- 对角线与处理事件的垂直距离表示“交付延迟”：事件产生与最终消费（处理）之间的时间差。

图来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O’Reilly Media, 2019.

流处理模型

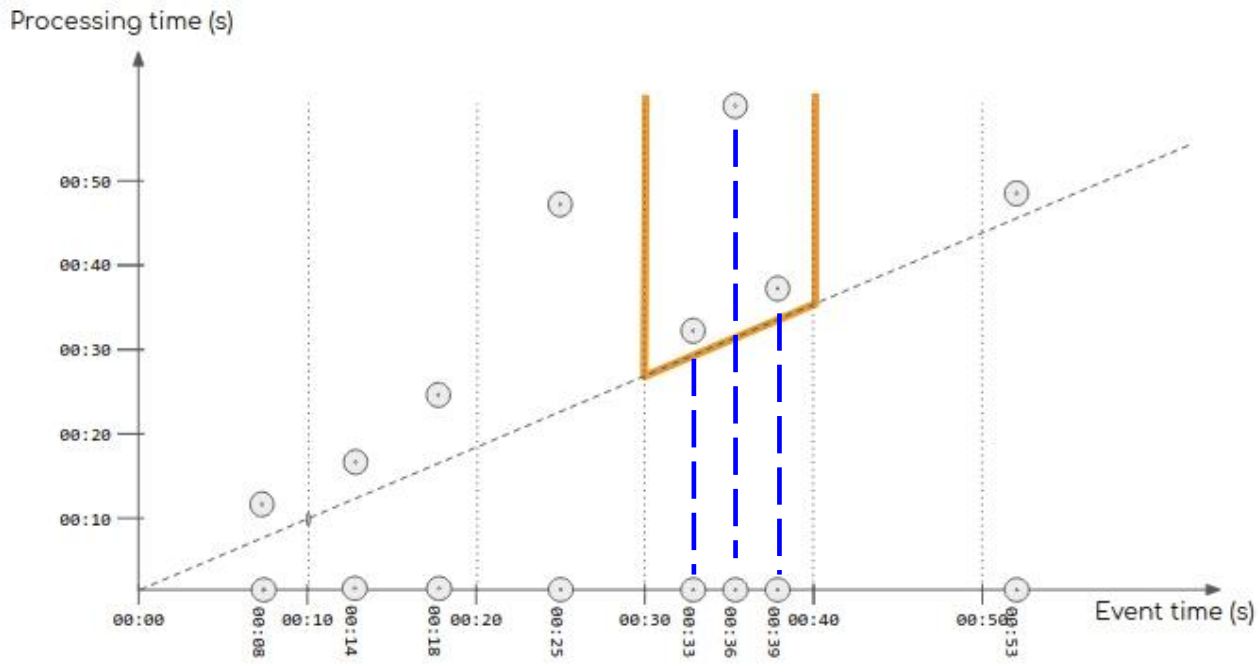
- **基于处理时间定义窗口**
  - **特征（基于高亮窗口，见右图）**
    - 窗口边界容易定义，可以根据时间来区分事件在窗口的内部或者外部；
    - 窗口中包含的事件与事件生成的时间无关。例如00：36事件由于延迟并未出现在高亮窗口中。



图来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O'Reilly Media, 2019.

流处理模型（续）

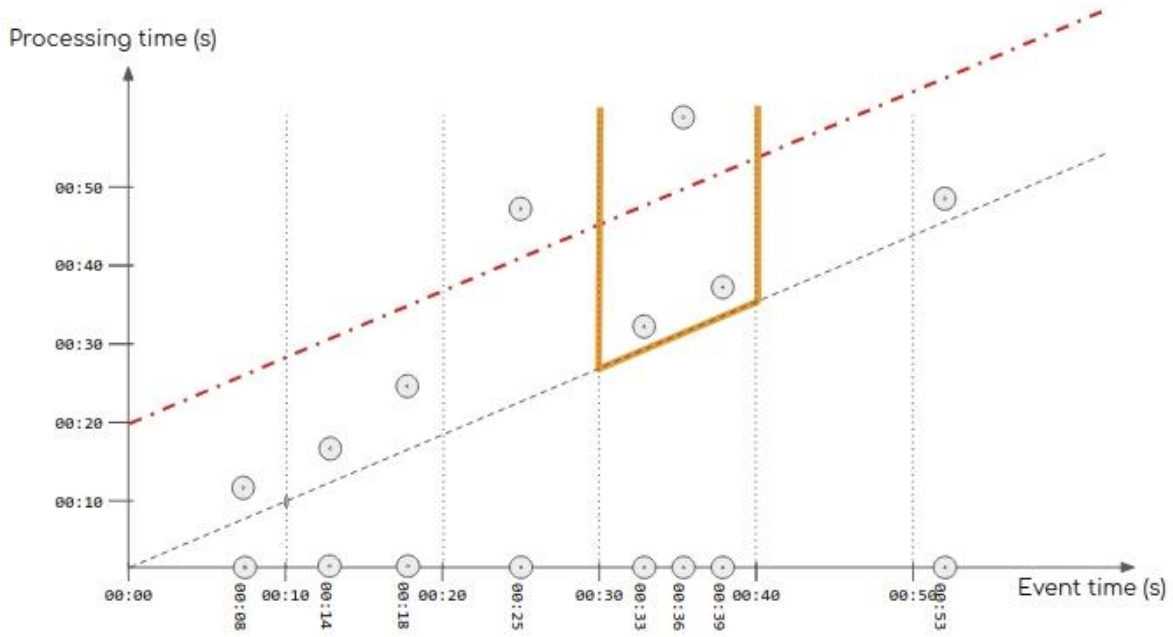
- **基于事件时间定义窗口**
  - **特征（基于高亮窗口，见右图）**
    - 高亮窗口包含了此段时间内“创建”的所有事件；
    - 该窗口没有定义窗口结束的自然上界。00：36事件迟到了超过20秒，因此高亮窗口进行统计至少要等到01：00。

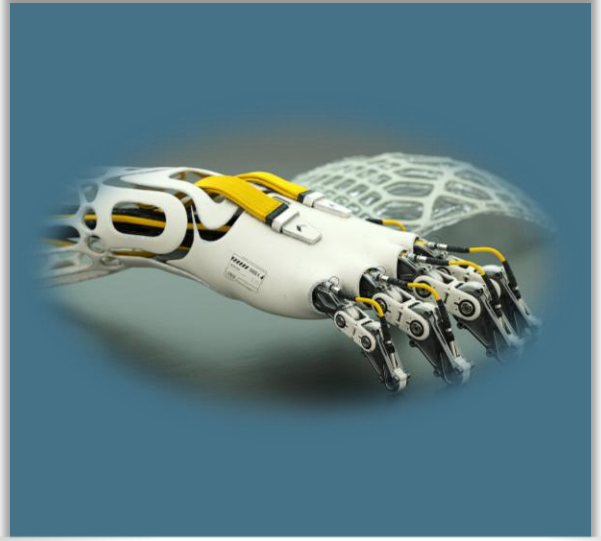


图来源：G. Maas and F. Garillot, Stream Processing with Apache Spark: Best Practices for Scaling and Optimizing Apache Spark. Sebastopol, CA: O’Reilly Media, 2019.

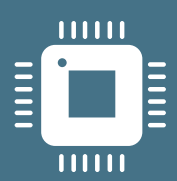
流处理模型（续）

- **使用水位线计算**
  - 水位线是数据流中给定任意时刻流处理器能够接受的最晚时间戳。
  - 任何比该时间戳更晚的事件都不会被流处理考虑。但流处理引擎会设置特殊的途径处理这些“迟到”事件。
  - 右图显示了利用水位线（红色虚线）关闭了事件窗口的开放边界，提供了窗口事件的归属标准。





附录
面向物联网系统的流式数据处理技术




附录

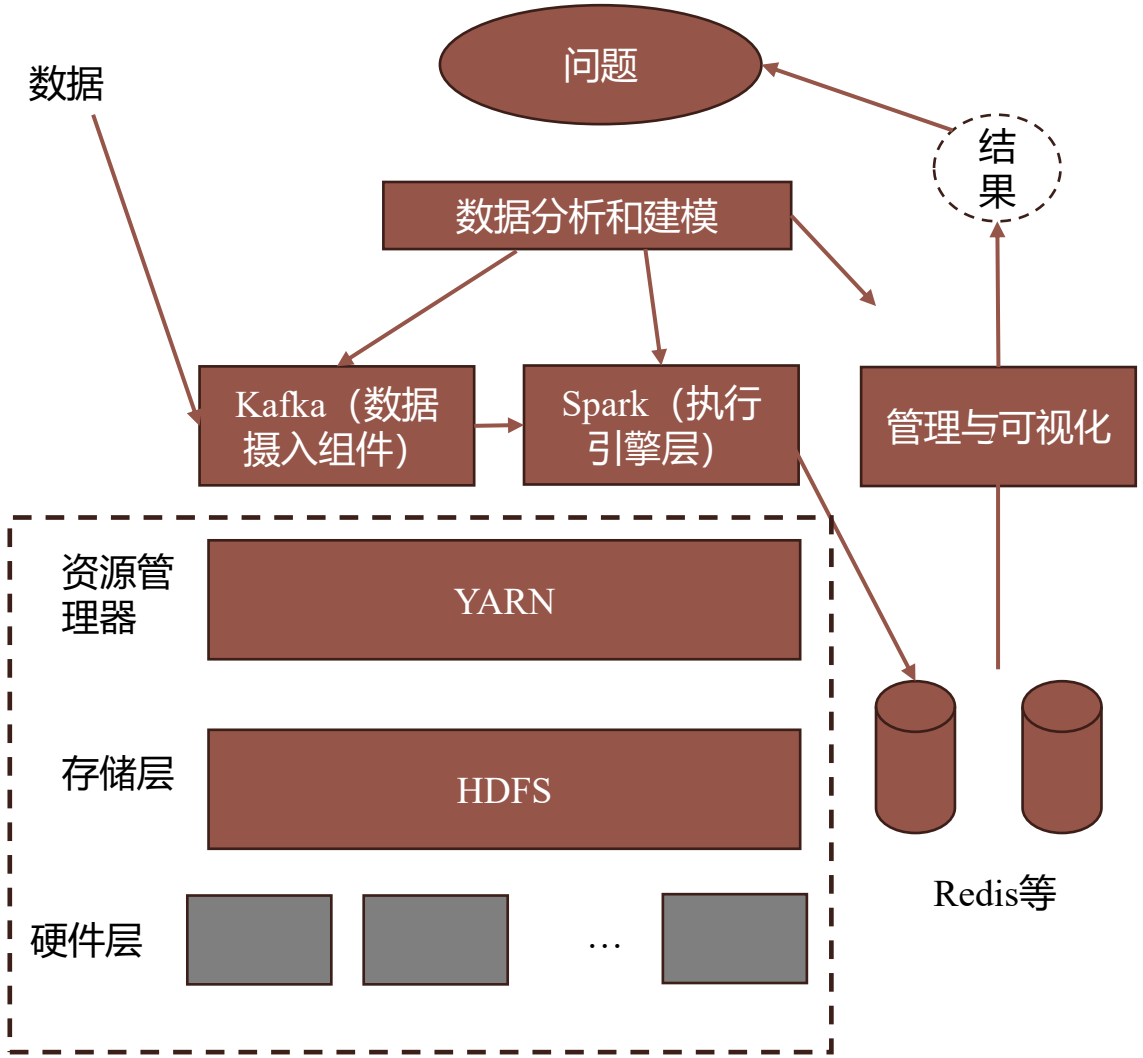
1 流处理概述

2 流处理模型

3 流处理引擎架构

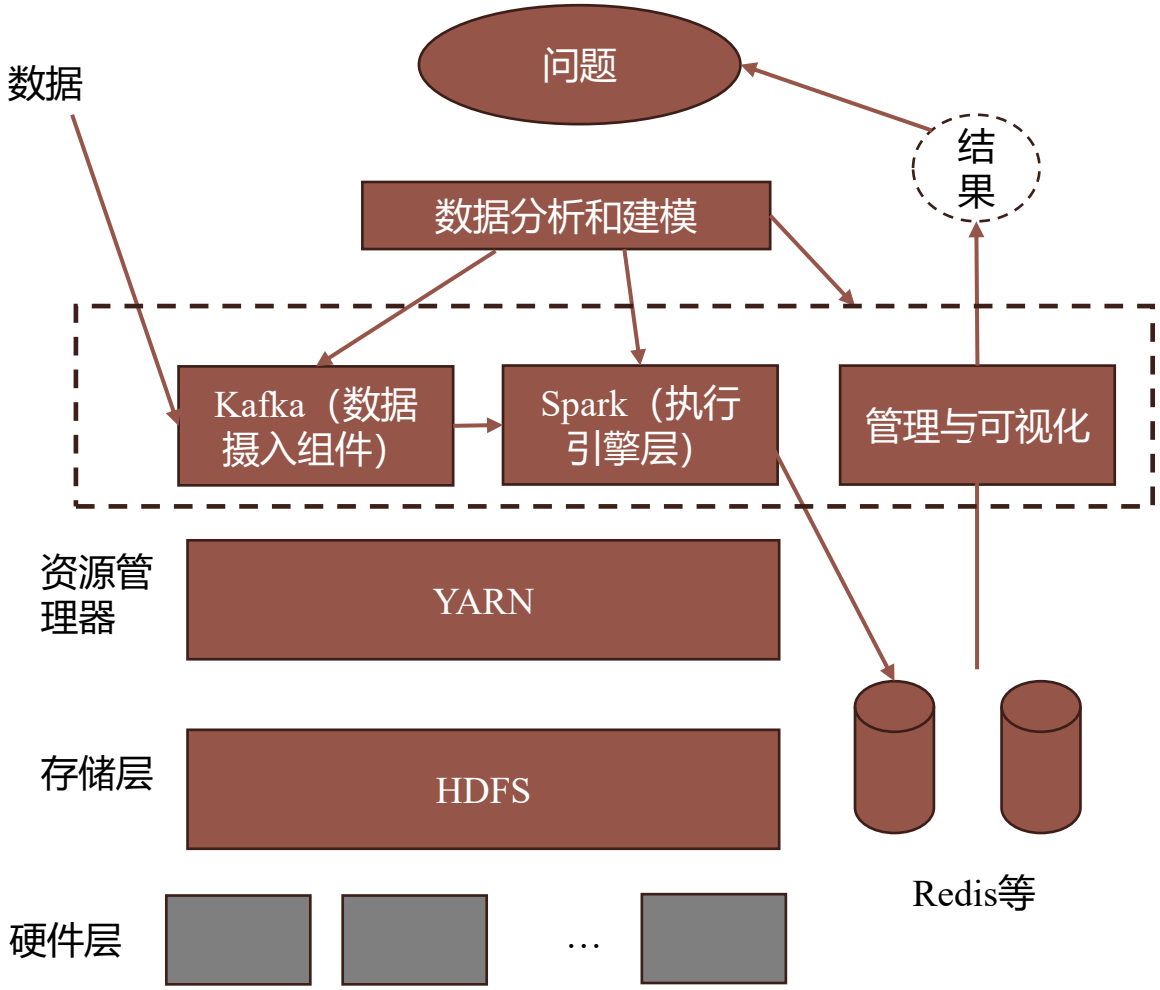
流处理引擎架构

- **流处理数据平台架构的组件——从底层硬件到上层数据处理的整个过程中，包括以下几个层面（见右图虚线框）：**
  - **硬件层**：本地的硬件设施；
  - **存储层**：通常是Hadoop分布式文件系统（HDFS）这一类分布式存储系统；
  - **资源管理层**：作为一个协调节点，用于提交作业到集群上执行，例如YARN、Standalone、Kubernetes等云上资源调度器；



流处理引擎架构（续）

- **流处理数据平台架构的组件（见右图虚线框）**
  - **执行引擎**：用于真正执行计算，进行数据处理操作。例如Apache Spark、 Apache Flink；
  - **数据摄入组件**：通过高吞吐量、分布式、持久化的特性，构建数据流动的“中枢管道”。例如Apache Kafka等消息系统；
  - **可视化层**：运用Web开发等相关技术，对结果数据进行实时可视化展示。



Lambda架构：流处理和批处理的混合架构

- **在流处理引擎基础上，拓展批处理层从而形成三层拓扑结构**
  - **批处理层**：通常基于Hadoop集群，处理速度慢于流处理层，但通过增加延迟提高了数据处理的吞吐量和准确性；
  - **速度层（流处理层）**：实时内存数据流，一般基于Aparche Spark/Fluent结构；
  - **服务层**：存储、分析和可视化批处理和流处理结果的复合层。