COMP3027J 课程
软件架构
软件架构风格（数据流风格）

邓永健

北京工业大学计算机学院

数据挖掘与安全实验室（DMS 实验室）

大纲

大纲

从传统建筑风格出发

传统建筑风格

定义

建筑风格是一种主要依据形态特征（如形式、技术、材料等）对建筑进行分类的方式。

定义

“风格”——经过长时间的实践，已被证明具有良好的工艺可行性、性能和实用性，可以直接拿来遵循和模仿（重复使用）。

软件架构风格

定义

描述了一类架构

- 与问题无关

在实践中屡屡发现

一组设计决策

具有已知特性，允许重复使用

一套之前已成功运用过的建筑设计方案。

一种架构可以采用多种架构风格。

软件架构风格

软件架构风格描述了特定领域中软件系统族的**组织惯用模式**，反映了该领域中许多系统共有的**结构和语义特征**，并指导如何有效地将各种模块和子系统组织成一个完整的系统。

建筑风格
=[组件/连接器词汇表、拓扑结构、语义约束]

福利

1. 促进可重复使用性

- 设计复用

同一领域内的系统通常具有相似的架构，这些架构反映了该领域的概念。
应用程序产品线围绕核心架构构建，并包含满足特定客户需求的变体。

- 代码复用

！ 风格不变方面的共享实现

- 文档复用

福利

2. 提高开发效率和生产力

3. 为更多的新设计思路提供一个起点

4. 促进设计师之间的交流

诸如“客户端 - 服务器”这样的短语蕴含了大量的信息。

5. 快速为软件系统找到适用的 SA 设计方案

6. 权衡利弊并预先评估系统的安全性。

7. 降低 SA 设计的风险

软件架构风格的内容

1. **姓名**

   每种架构风格都有一个独特的、简短的描述性名称。

2. **问题**

   每种架构风格都包含对要解决的问题的描述。问题陈述可能描述一类问题或一个具体问题。

3. **背景**

   这些假设是为使架构风格能够用于解决问题而必须满足的条件。它们包括对解决方案的限制以及可能使解决方案更易于使用的可选要求。

软件架构风格的内容

4. 模型

该模型旨在描述软件架构风格。

5. 后果

使用这种风格的优缺点

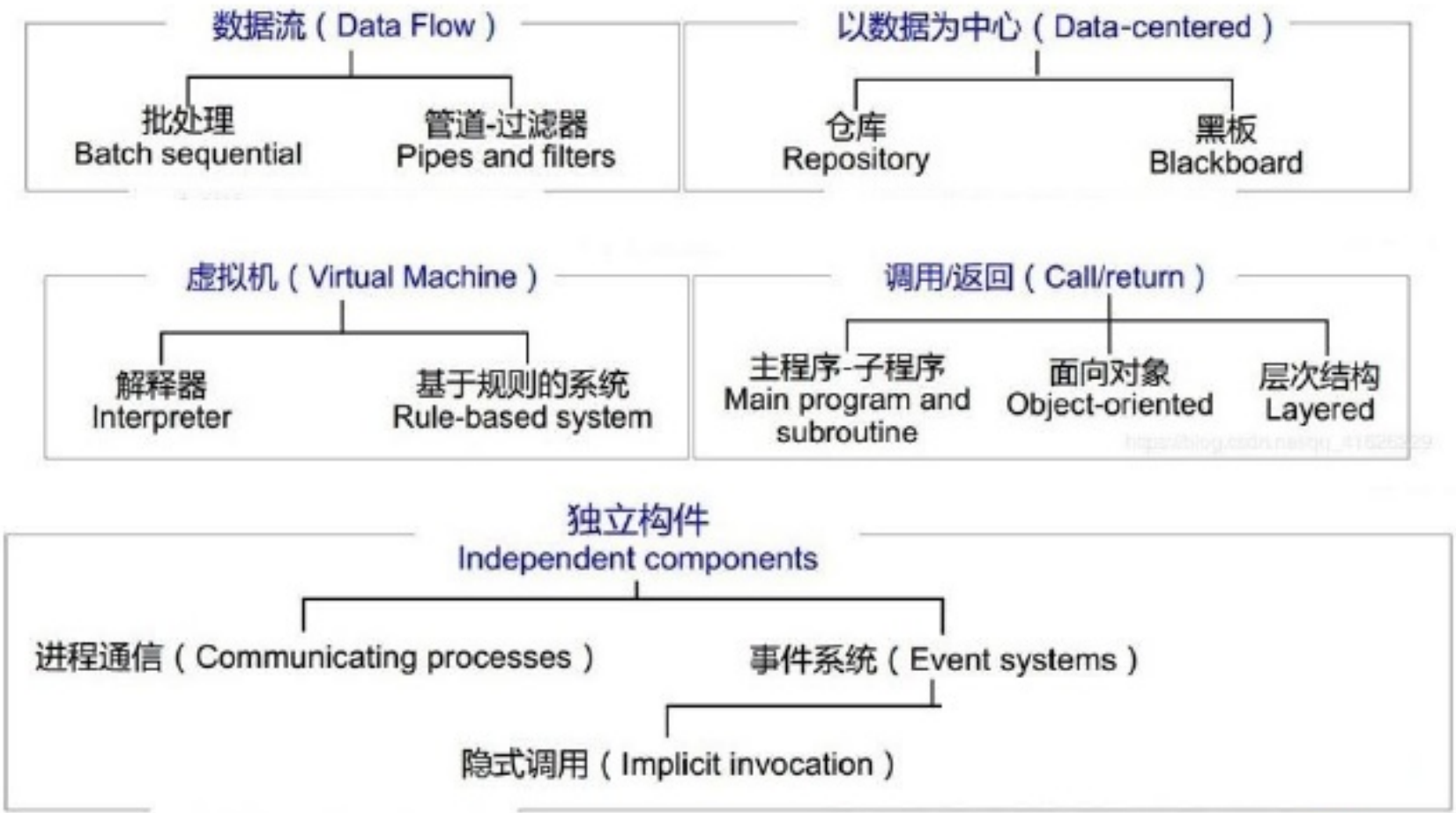
6. 实施

- 有助于设计师对本架构设计进行定制以获得最佳效果的其他实施建议

7. 已知用途

- 已知的该风格在现有系统中的应用实例

软件架构风格分类法


数据流（Data Flow）
批处理
Batch sequential
管道-过滤器
Pipes and filters
以数据为中心（Data-centered）
仓库
Repository
黑板
Blackboard
虚拟机（Virtual Machine）
解释器
Interpreter
基于规则的系统
Rule-based system
调用/返回（Call/return）
主程序-子程序
Main program and subroutine
面向对象
Object-oriented
层次结构
Layered
独立构件
Independent components
进程通信（Communicating processes）
事件系统（Event systems）
隐式调用（Implicit invocation）

大纲

数据流风格的特点

数据流系统是指这样一种系统：

数据的可用性决定着计算（处理）是否执行。

该设计的结构由数据在各处理过程之间有序流动所决定。

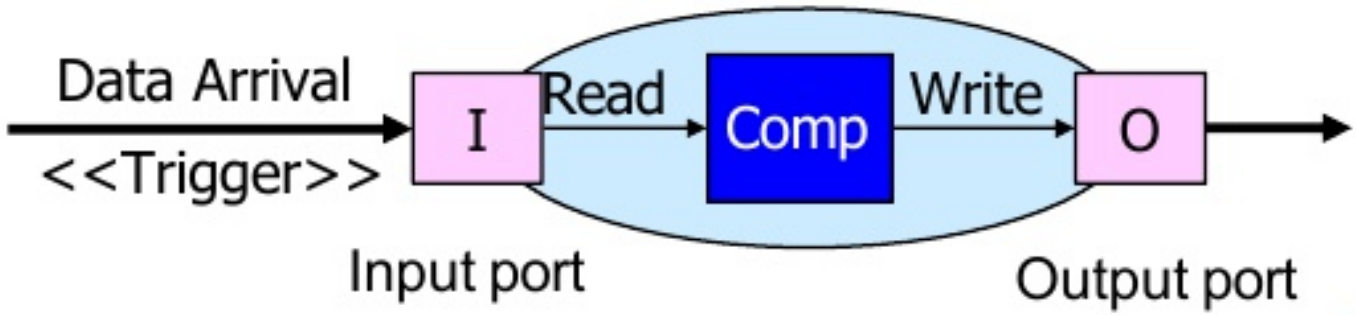
在纯数据流系统中，各进程之间除了数据交换外，不存在任何其他交互。

数据流风格的组成部分

组件：数据处理组件

- 构件接口包括输入端口和输出端口，输入端口用于读取数据，输出端口用于写入数据。

计算模型：从输入端口读取数据，进行计算，然后将数据写入输出端口。



数据流风格的连接器

连接件：数据流（流）

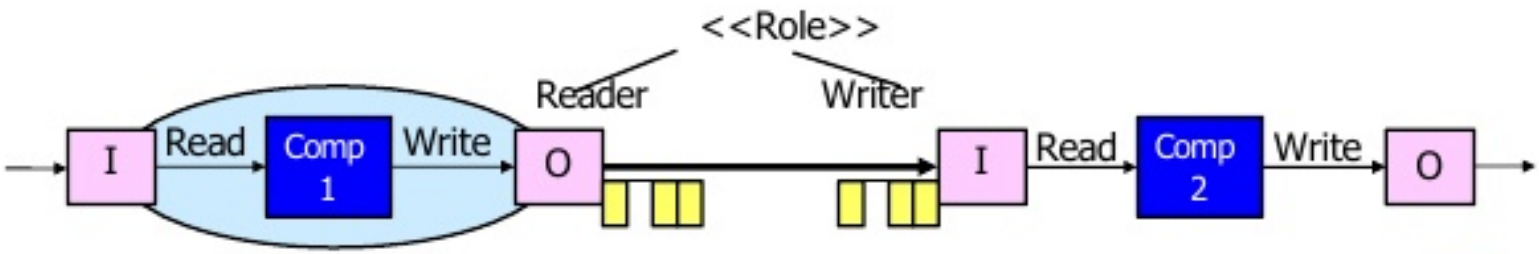
单向、通常是异步、有缓冲的

有缓冲)

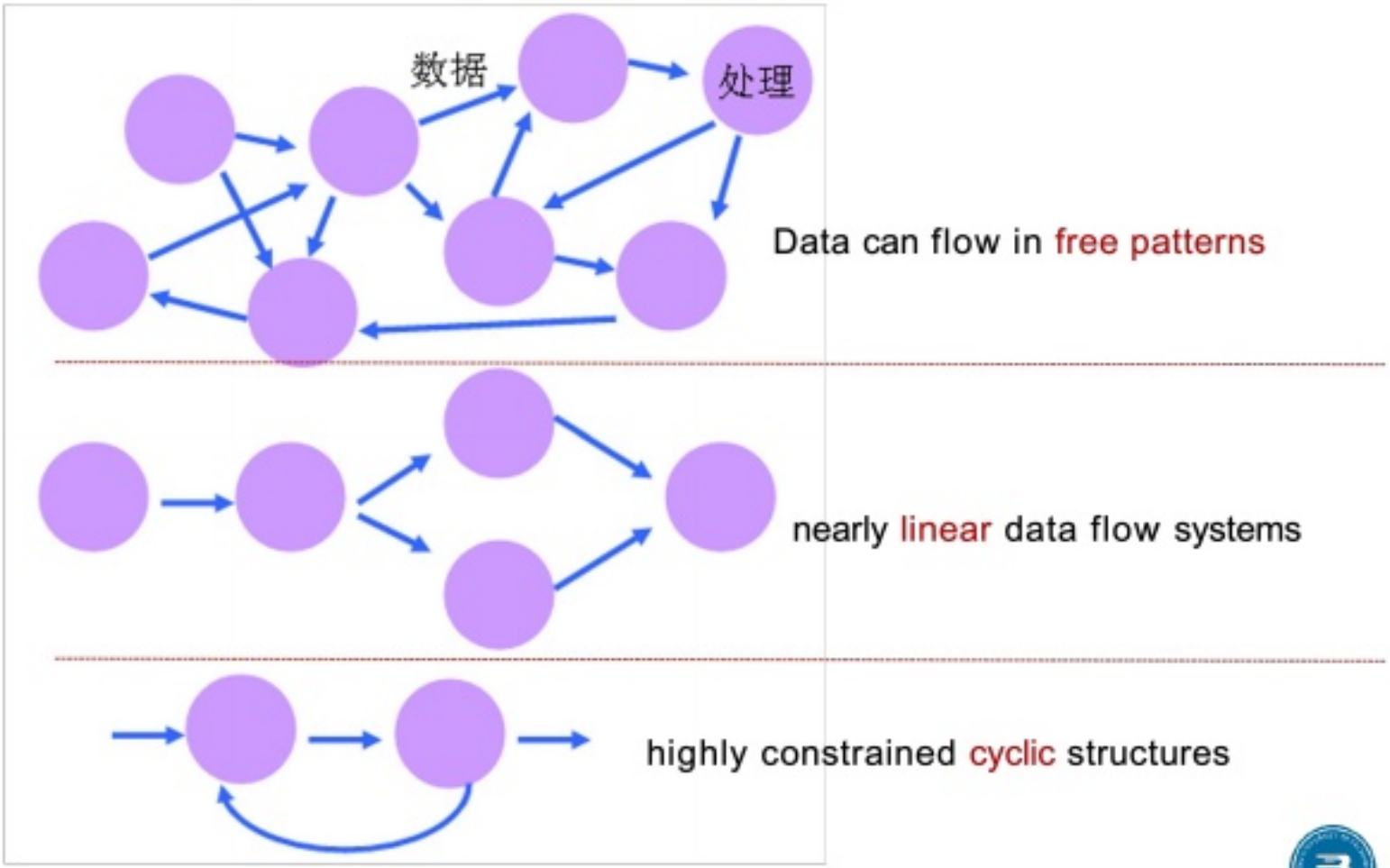
接口角色分为读取者和写入者。

传输到另一个处理的输入端口的模型

传送到另一个处理的输入端口)



系统中的数据流模式


数据
处理
Data can flow in free patterns
nearly linear data flow systems
highly constrained cyclic structures

大纲

定义

1.组件（处理步骤）是独立的程序（基本构件：独立的应用程序）

2.Connectors are some type of media - traditionally tape (连件： 某种类型的媒质**)**

3.Topology: Connectors define data flow graph (连接件定义了相应的数据流图，表达拓扑结构**)**

4.Processing steps are independent programs(每个处理步骤是一个独立的程序**)**

5.Each step runs to completion before the next step starts(每一步必须在前一步结束后才能开始**)**

6.Data transmitted as a whole between steps(数据必须是完的， 以整体的方式传递**)**

定义

1.组件（处理步骤）是独立的程序（基本构件：独立的应用程序）

2.Connectors are some type of media - traditionally tape (连件： 某种类型的媒质**)**

3.Topology: Connectors define data flow graph (连接件定义了相应的数据流图，表达拓扑结构**)**

4.Processing steps are independent programs(每个处理步骤是一个独立的程序**)**

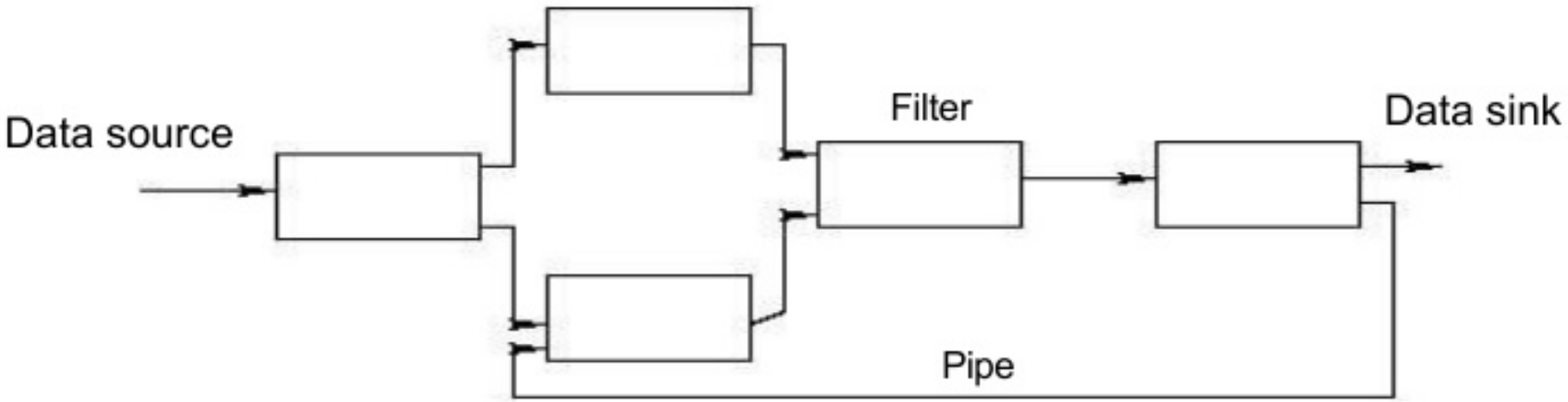
5.Each step runs to completion before the next step starts(每一步必须在前一步结束后才能开始**)**

6.Data transmitted as a whole between steps(数据必须是完的， 以整体的方式传递**)**

大纲

管道与过滤器

一条管道由一系列处理单元组成，每个单元的输出都是下一个单元的输入。通常，在相邻单元之间会提供一定量的缓冲区。



定义

1. 情景：

- 过滤器封装了数据，数据持续生成，系统需要对这些数据执行一些处理（分析、计算、转换等）。

2. 解决方案：

将系统分解为若干个按顺序排列的处理步骤，这些步骤通过数据流相互连接，前一步骤的输出即为后一步骤的输入；

每个处理步骤均由一个过滤组件（Filter）来实现；

管道负责在处理步骤之间传输数据。

3. 每个处理步骤（过滤器）都有一组输入和输出。过滤器从管道中读取输入数据流，在内部对其进行处理，然后生成输出数据流并写入。

把它纳入计划。

定义

4. 组件：过滤器——处理数据流

- 过滤器封装了一个处理步骤（算法或计算）

据终点/汇点是特定的过滤器。

据终止点可以看作是特殊的过滤器)

5. 连接件：管道将源与终端过滤器连接起来

管道，连接一个源和一个目的过滤器)

管道将一个过滤器的输出数据传输到另一个过滤器的输入端。

数据可能是一串 ASCII 字符流。

6. 拓扑结构：连接器定义数据流图

7. 样式不变量：过滤器相互独立

管道与过滤器

逐步将数据从源端转换到目标端（递增地读取和消费数据流），数据一到达便被处理，而不是先收集再处理，也就是说，在输入被完全消费之前，输出就已经产生了。

过滤器

1. 逐步将部分源数据转换为接收数据。
2. 流到流的转换

过滤器

过滤器是独立的实体。

- 处理流中无上下文信息
- 不保留状态（即每次实例化之间没有状态保存）
- 对上下游过滤器一无所知
  何了解)

管道

管道：将一个过滤器的输出数据传输到另一个过滤器的输入（或者传输到设备或文件）。

从一个数据源到一个数据接收端的单向流动（单向流）

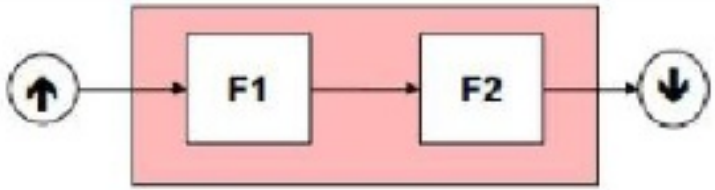
管道可能具有缓冲区。

管道构成数据传输图。

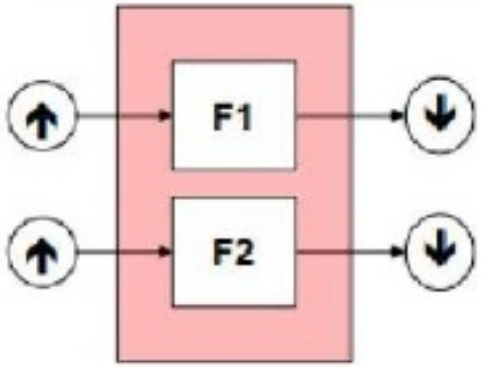
在不同管道中流动的数据流可能具有不同的数据格式。当数据流经每个过滤器时，都会被过滤器丰富、优化、转换、融合、分解等，从而发生变化。

Pipe

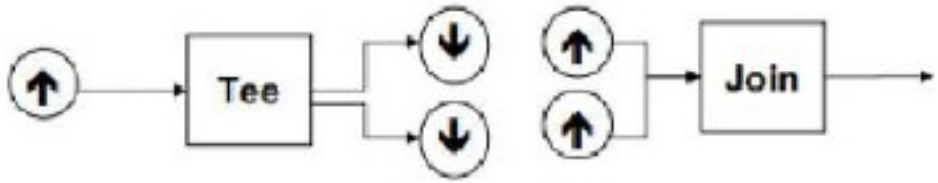
- Sequential Composition
  Unix: F1 | F2



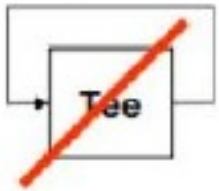
- Parallel Composition
  Unix: F1 & F2



- Tee & Join



- Restriction to **Linear Composition**



示例

编译器

图像处理

信号处理

- 声音与图像处理（语音和视频流）

管道与过滤器风格的优点

1. 高内聚：过滤器是自包含的处理服务，执行特定功能，因此内聚性相当高。
2. 低耦合：过滤器仅通过管道进行通信，因此在耦合方面受到“一定程度”的限制。
3. 可重用性：支持重用过滤器。
4. 无论是作为并发系统还是时序系统，实现起来都很简单。
5. 扩展性：易于添加新的过滤器。
6. 灵活性：（1）过滤器的功能可以轻松重新定义；（2）控制可以重新路由。

管道-过滤器风格的缺点

1. **不适用于处理交互的应用程序。**

   在早期阶段提出，当时对交互式应用程序的需求并不高

   当需要逐步显示更改时，这个问题尤为突出。

1. **系统性能不高，增加了写入过滤器的复杂性。**

   由于缺乏通用的数据传输标准，每个过滤器都增加了数据解析和合成的工作量。

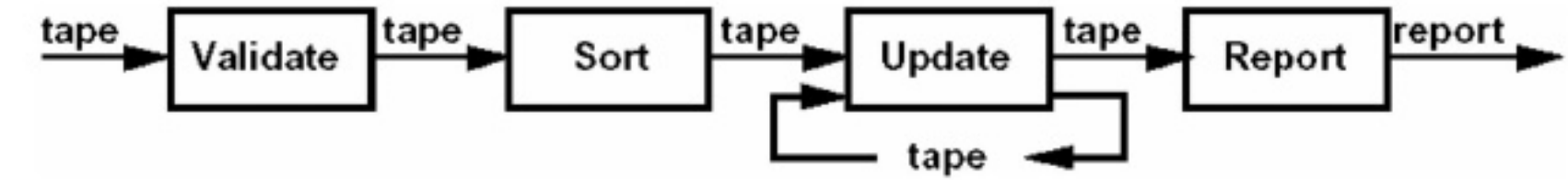
   大部分处理时间都花在了格式转换上。

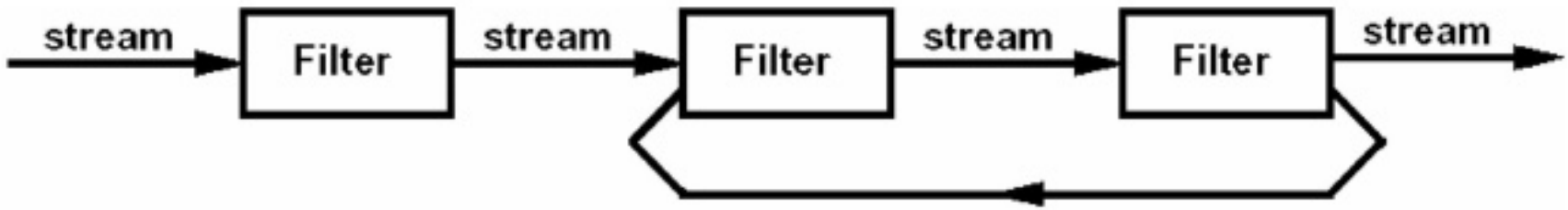
   - 不适用于需要大量共享数据的应用程序设置

大纲

批处理顺序模式与管道-过滤器模式

将任务分解为一系列按固定顺序排列的计算单元，并且这些单元之间仅通过数据传输进行交互。





批处理顺序式与管道-过滤器式

批量顺序	管道与过滤器
整体传输数据（总量）	增量的
粗粒度（构件粒度较大）	细粒度（构件粒度较小）
高延迟（实时性差）	结果开始处理（实时性好）
无并发	可并发

、