COMP3027J
Software Architecture
Software Architectural Styles (Data Flow Style)

DENG, YONGJIAN

Faculty of Computer Science, BJUT

Data Mining & Security Lab (DMS Lab)

北京工業大學
BEIJING UNIVERSITY OF TECHNOLOGY

Outline

Outline

北京工業大學 BEIJING UNIVERSITY OF TECHNOLOGY

Starting from Traditional Architectural Style

A

B

C

Traditional Architectural Style

Definition

Architectural style constitutes a mode of classifying architecture largely by morphological characteristics in terms of form, techniques, materials, etc.

Definition

”Style” - After a long period of practice, it has been proven to have good process feasibility, performance, and practicability, and can be directly used to follow and imitate (reuse).

Software Architectural Style

Definition

- describes a class of architectures
- independent of the problems
- found repeatedly in practice
- a package of design decisions
- has known properties that permit reuse
- A set of architectural design solutions that have been used successfully before.
- An architecture can use several architectural styles

Software Architectural Style

The software architectural style describes the *idiomatic paradigm of the organization* of software system families in a specific domain, re- flects the *structural and semantic characteristics* shared by many systems in the domain, and guides how to effectively organize various modules and subsystems into a complete system.

Architecture style
=[Component/Connector
vocabulary, Topology,
Semantic Constraints
]

Benefits

1. **Promote reusability**
   - Design reuse
     - ! Systems in the same domain often have similar architectures that reflect domain concepts
     - ! Application product lines are built around a core architecture with variants that satisfy particular customer requirements
   - Code reuse
     - ! Shared implementations of invariant aspects of style
   - Documentation reuse

Benefits

2. **Improves development efficiency and productivity**
3. **Provide a starting point for additional and new design ideas**
4. **Promotes communications among the designers**
   - Phrase such as “client-server”conveys a lot of information

5. **Quickly find an applicable SA design solution for a software system**
6. **Make trade-offs and pre-evaluate the SA of the system**
7. **Diminish the risks of SA design**

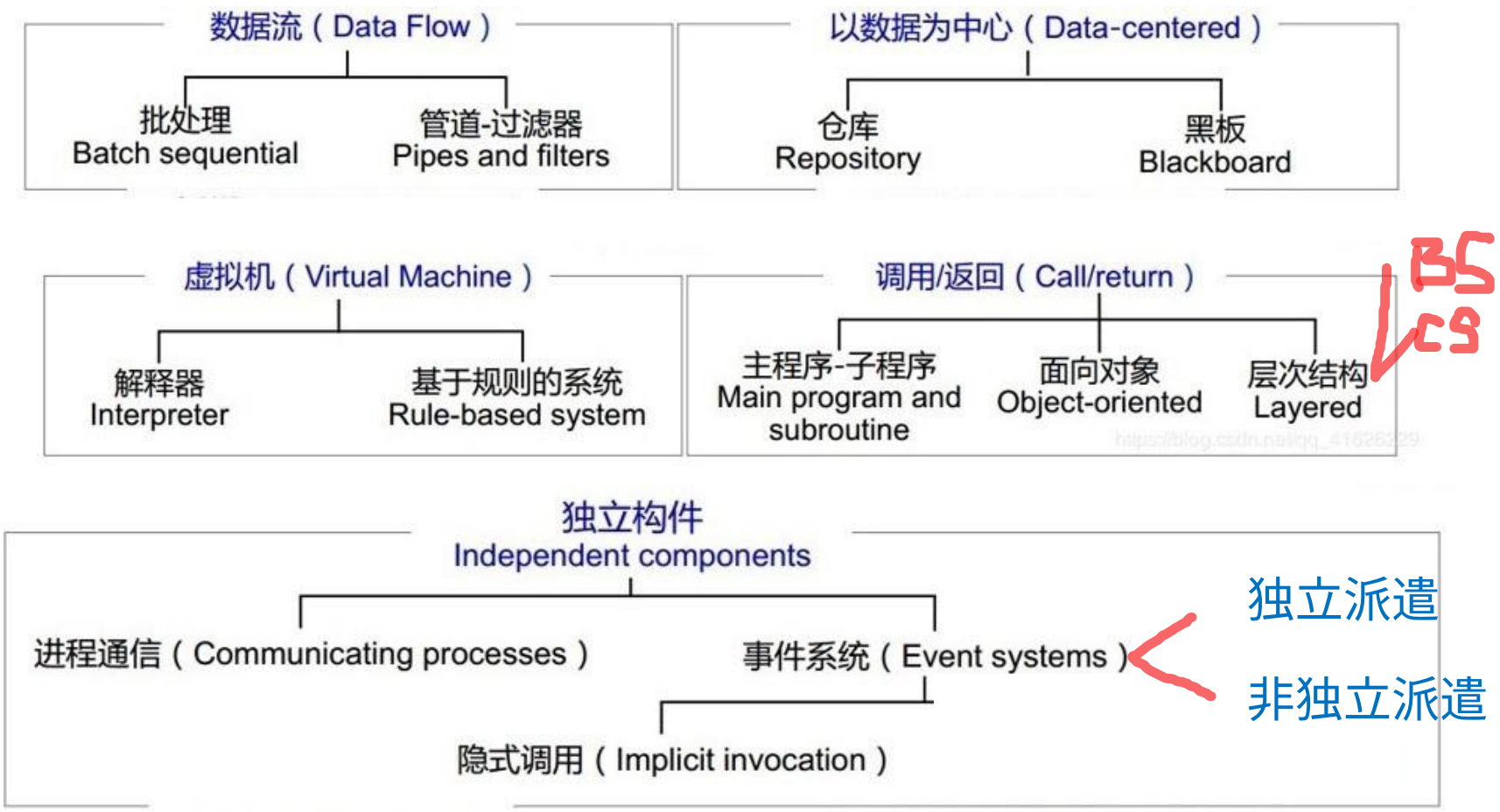
Content of Software Architectural Style

1. **Name**
   - Each architecture style has a unique, short descriptive name
2. **Problem**
   - Each architecture style contains a description of the problem to be solved. The problem statement may describe a class of problems or a specific problem
3. **Context**
   - The assumptions are conditions that must be satisfied in order for the architecture style to be usable in solving the problem. They include constraints on the solution and optional requirements that may make the solution easier to use

Content of Software Architectural Style

4. **Models**
   - The model is to describe the software architecture style
5. **Consequences**
   - The advantages and disadvantages of using this style
6. **Implementation**
   - Additional implementation advice that can assist designers in customizing this architectural design for the best results
7. **Known Uses**
   - Known applications of the style within existing systems

Taxonomy of Software Architectural Style



Outline

Charecteristics of Data Flow Style

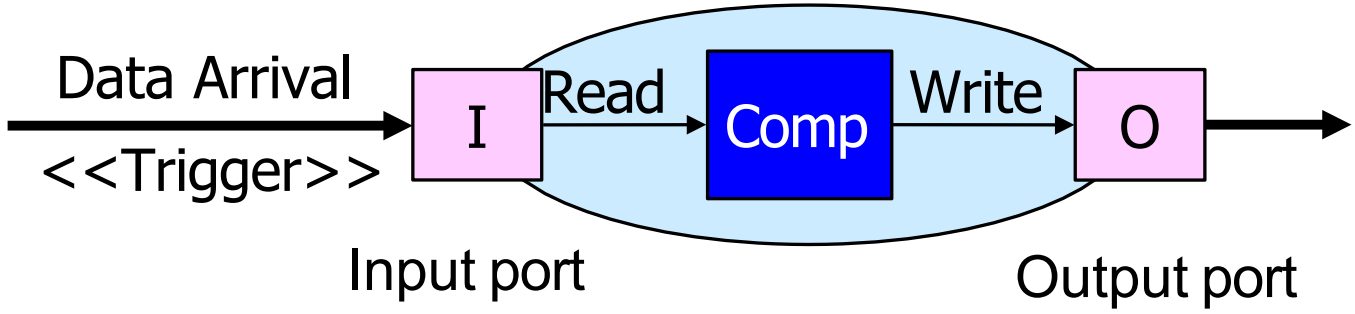
A data flow system is one in which:

- the availability of data controls the computation (数据的可用性决定着处理 < 计算单元 > 是否执行)
- the structure of the design is decided by orderly motion of data from process to process (系统结构由数据在各处理之间的有序移动决定)
- in a pure data flow system, there is no other interaction between processes (在纯数据流系统中，处理之间除了数据交换没有任何其他的交互)

Components of Data Flow Style

Components: data processing components(基本构件：数据处理)

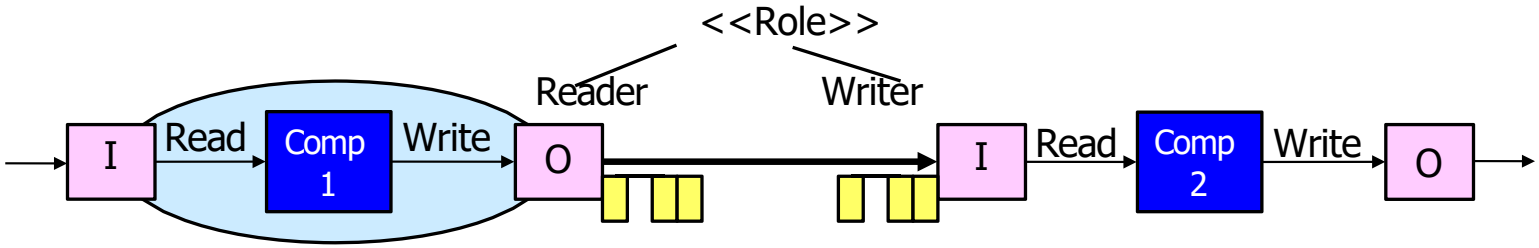
- Interfaces are input ports and output ports (构件接口：输入端口和输出端口)，Input ports read data; output ports write data
- Computational model: read data from input ports, compute, write data to output ports (计算模型：从输入端口读数，经过计算/ 处理，然后写到输出端口)



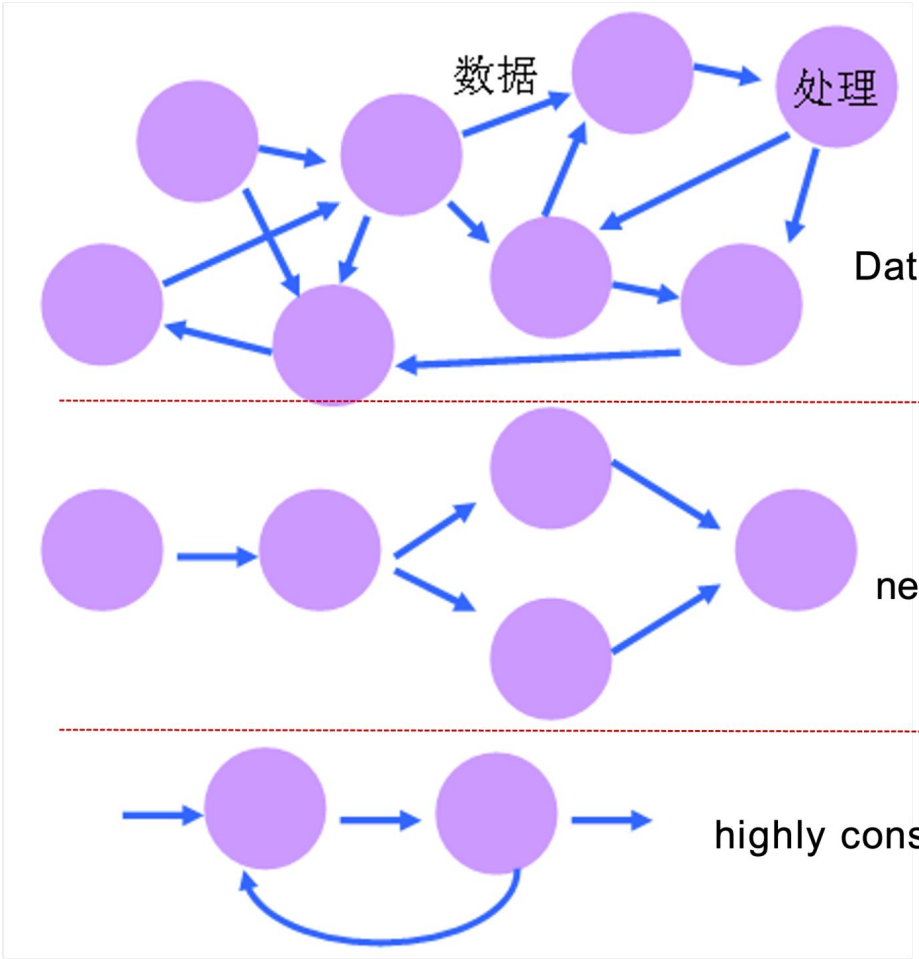
Connectors of Data Flow Style

Connectors: data flow (stream) (连接件：数据流)

- Unidirectional, usually asynchronous, buffered (单向、通常是异步、有缓冲)
- Interfaces are reader and writer roles (接口角色：reader 和 writer)
- Computational model (计算模型：把数据从一个处理的输出端口传送到另一个处理的输入端口)



Patterns of Data Flow in Systems



Data can flow in free patterns

nearly linear data flow systems

highly constrained cyclic structures

Outline

Definition

1.Components (processing steps) are independent programs (基本构件：独立的应用程序)

2.Connectors are some type of media - traditionally tape (连接件：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（数据必须是完整的，以整体的方式传递）

Definition

1.Components (processing steps) are independent programs (基本构件：独立的应用程序)

2.Connectors are some type of media - traditionally tape (连接件：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）

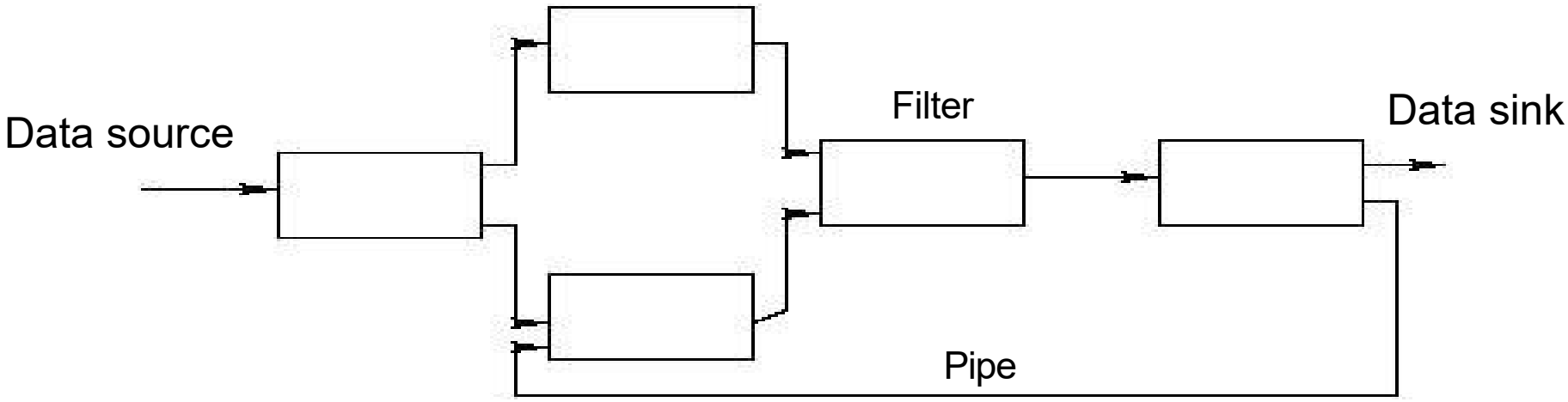
5.Each step runs to completion before the next step starts（每一步必须在前一步结束后才能开始）

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

Outline

Pipe-and-Filter

A pipeline consists of a chain of processing elements, and the output of each element is the input of the next. Usually, some amount of buffering is provided between consecutive elements.



Definition

1. Scenario:

- A filter encapsulates a Data is continuously generated, and the system needs to perform some processing (analysis, calculation, conversion, etc.) on these data.

2. Solution :

- Decompose the system into several sequential processing steps, which are connected by data flow, and the output of one step is the input of another step;
- Each processing step is implemented by a filter component (Filter);
- Pipelines are responsible for data transfer between processing steps.

3. Each processing step (filter) has a set of input and output. The filter reads the input data stream from the pipeline, processes it internally, and then generates the output data stream and writes it into the pipeline.

Definition

4. Components: Filters —process data streams (构件：过滤器，处理数据流)

- A filter encapsulates a processing step (algorithm or computation) (一个过滤器封装了一个处理步骤)
- Data source and data end/sink are particular filters (数据源点和数据终止点可以看作是特殊的过滤器)

5. Connectors: A pipe connects a source and an end filter (连接件：管道，连接一个源和一个目的过滤器)

- Pipes move data from a filter output to a filter input
- Data may be a stream of ASCII characters

6. Topology: Connectors define data flow graph

7. Style invariants：Filters are independent

Pipe-and-Filter

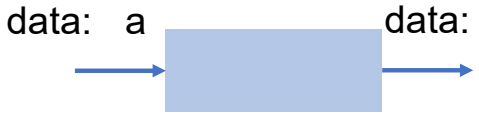
Incrementally transform data from the source to the sink (递增的读取和消费数据流)，数据到来时便被处理，不是收集然后处理，即在输入被完全消费之前，输出便产生了。

Filter

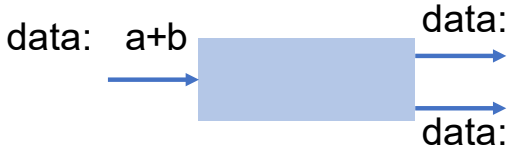
1. **Incrementally transform some of the source data into sink data.**
2. **Stream-to-stream transformation**

通过计算和增加信息来丰富数据

通过浓缩和删减来精炼数据



通过改变数据表现方式来转化数据



将一个数据流分解为多个数据

将多个数据流合并为一个数据流

Filter

Filters are independent entities.

- no context in processing streams (无上下文信息)
- no state preservation between instantiations (不保留状态)
- no knowledge of upstream/downstream filters (对其他过滤器无任何了解)

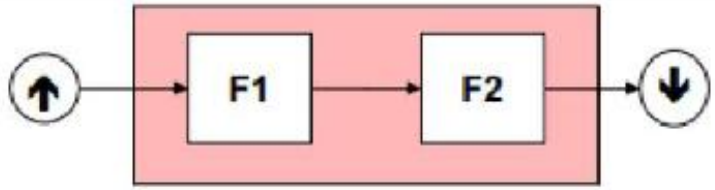
Pipe

Pipes: move data from a filter's output to a filter's input (or to a device or file).

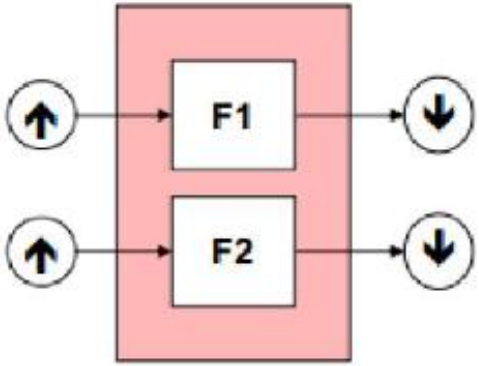
- One-way flow from one data source to one data sink (单向流)
- A pipe may implement a buffer (可能具有缓冲区)
- Pipes form data transmission graph (管道形成传输图)
- The data streams flowing in different pipelines may have different data formats. When the data flows through each filter, it is enriched, refined, converted, fused, decomposed, etc. by the filter, thus changing.

Pipe

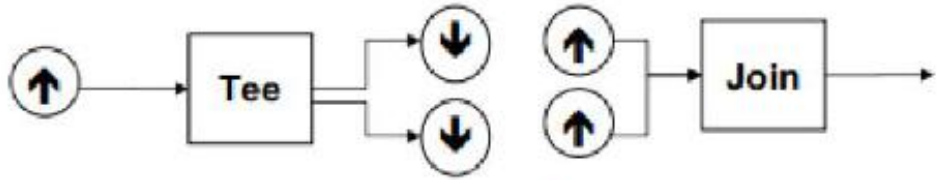
- Sequential Composition
  Unix: F1 | F2



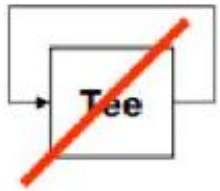
- Parallel Composition
  Unix: F1 & F2



- Tee & Join



- Restriction to **Linear Composition**



Examples

- Compiler (编译器)
- Image processing (图像处理)
- Signal processing (信号处理)
- Voice and video streaming (声音与图像处理)

Advantages of Pipe-and-Filter Style

1. High cohesive(高内聚): Filters are self-containing processing service that performs a specific function thus it is fairly cohesive.

2. Low coupling(低耦合): Filters communicate through pipes only, thus it is "somewhat"constrained in coupling.

3. Reusability: Supports reuse filters.

4. Simple to implement as either a concurrent(并发系统) or sequential system(时序系统).

5. Extendibility(扩展性): easy to add new filters.

6. Flexibility(灵活性): (1)functionality of filters can be easily redefined; (2)control can be re-routed.

Disadvantages of Pipe-and-Filter Style

1. Not suitable for applications that handle interactions.

- Proposed in the early stage when the demand for interactive applications was not high
- This problem is especially acute when changes need to be displayed incrementally

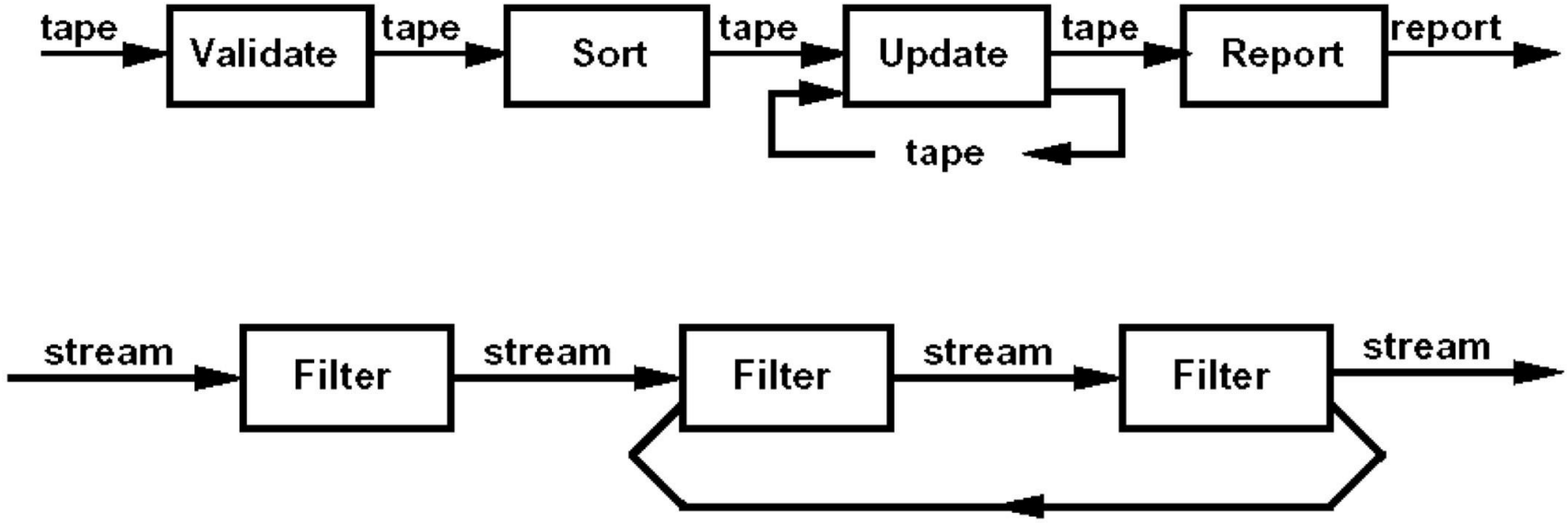
1. The system performance is not high, increasing the writing filters' complexity.

- Lack of common standards for data transfer, each filter adds to the work of parsing and synthesizing data
- Most of the processing time is spent on format conversion
- Not suitable for app setups that require large amounts of shared data

Outline

Batch Sequential Versus Pipe-and-Filter 优势，劣势

Decompose the task into a series of computing units in a fixed order & Only interact with each other through data transfer.



Batch Sequential Versus Pipe-and-Filter

Batch Sequential	Pipe-and-Filter
total(整体传递数据)	incremental(增量)
coarse grained(构件粒度较大)	fine grained (构件粒度较小)
high latency (延迟高，实时性差)	results starts processing (实时性好)
no concurrency (无并发)	concurrency possible (可并发)