Apache Flink 流批一体的资源管理与任务调度

Resource Management and Task Scheduling for Streaming and Batch Processing in Apache Flink

徐帅 Shuai Xu
阿里巴巴高级技术专家 Staff Engineer, Alibaba

宋辛童 Xintong Song
阿里巴巴高级开发工程师 Engineer II, Alibaba

FLINK FORWARD # ASIA

实时即未来 # Real-time Is The Future

FLINK FORWARD

Contents
目录

01 流批一体的资源管理

Resource Management for Streaming and Batch Processing

02 流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

流批一体的资源管理

Resource Management for Streaming and Batch Processing

01

流批一体的资源管理的核心技术问题

Key technique problems of resource management for streaming and batch processing

特点

Characteristics

挑战

Challenges

解决方案

Solutions

特点1：内存消费主体差异

Characteristic 1 – Different memory consumers

流处理作业

Streaming processing jobs

State backends

State backends

Memory/FsStateBackend 使用 Java 堆内内存

Memory/FsStateBackend uses Java on-heap memory

RocksDBStateBackend 使用非 Java 内存

RocksDBStateBackend uses non Java memory

批处理作业

Batch processing jobs

缓存哈希表、排序缓冲等

Cached hash tables, sorting buffers, etc.

使用 Flink 自行管理的 Managed Memory

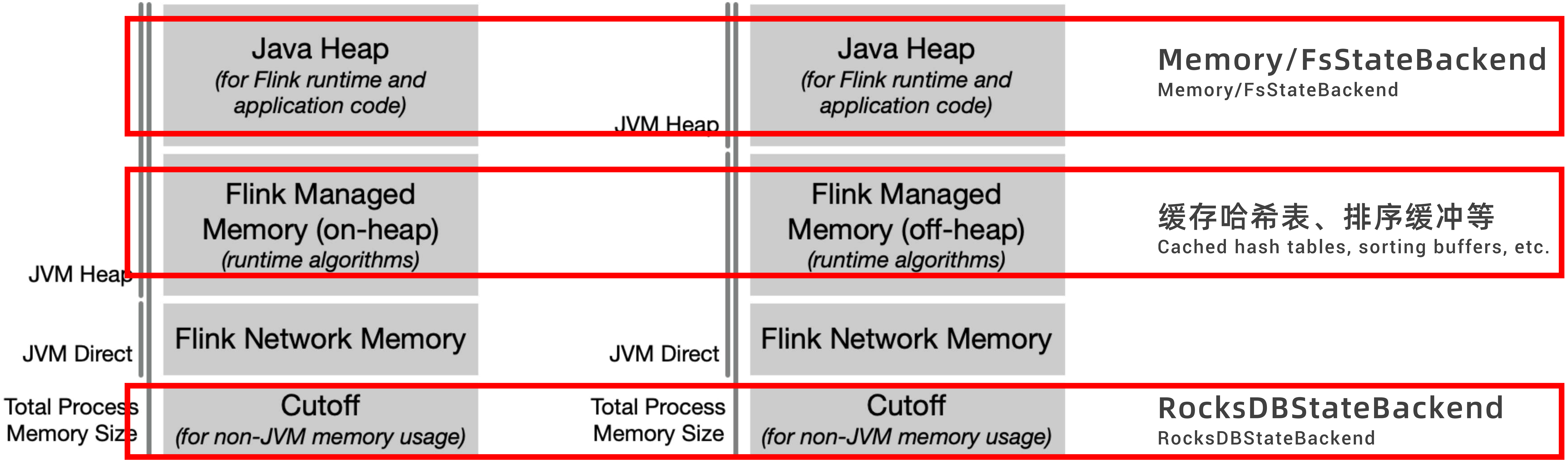
Use managed memory that managed by Flink itself

可使用 Java 堆内或堆外内存

Can use Java on-heap or off-heap memory

挑战1：能否用一套配置满足流和批作业的不同需求?

Challenge 1 – Can we use one unified set of configurations to satisfy requirements of streaming and batch processing jobs?



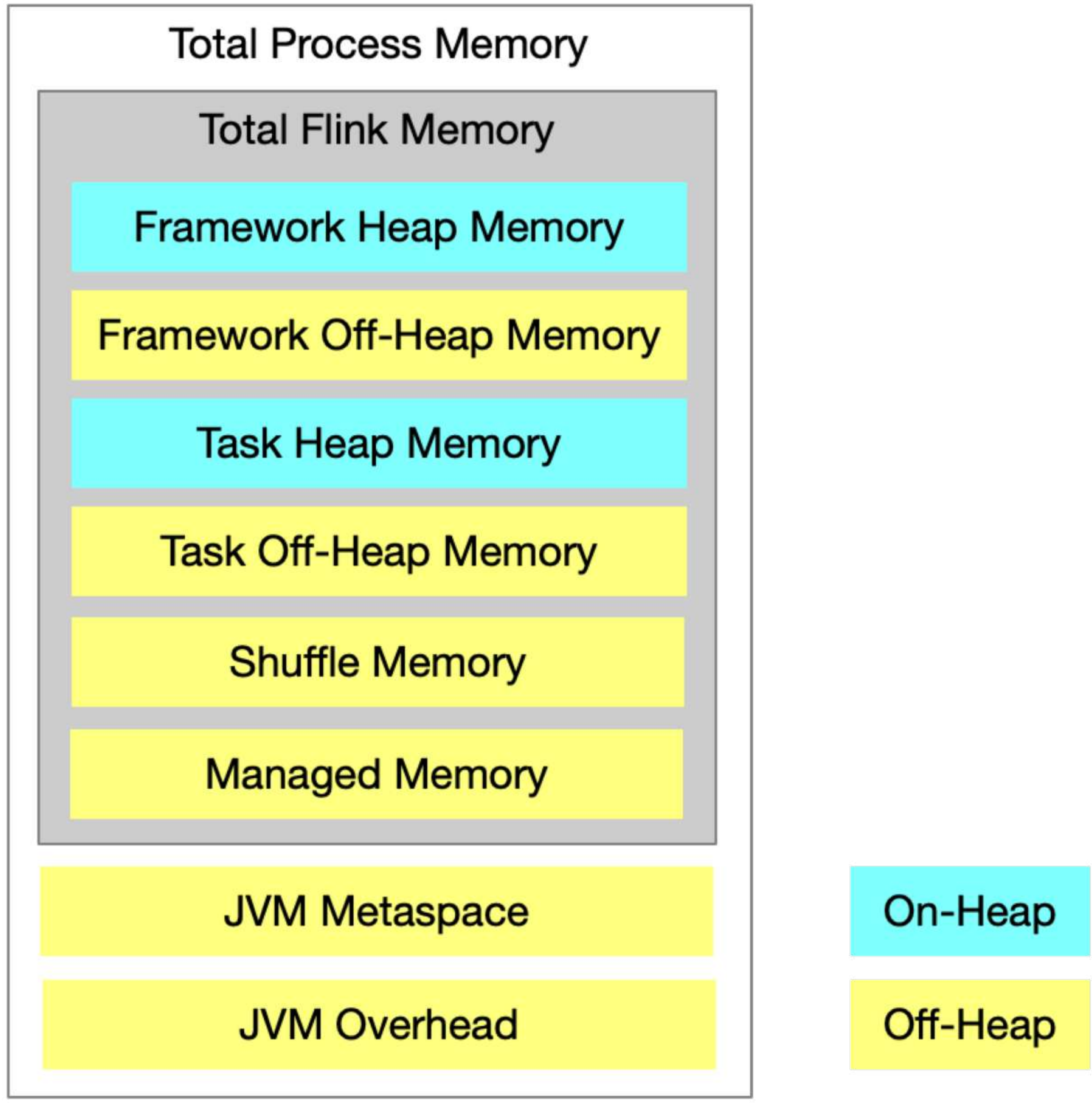
Java Heap
(for Flink runtime and application code)
Flink Managed Memory (on-heap)
(runtime algorithms)
JVM Heap
Flink Network Memory
JVM Direct
Total Process Memory Size
Cutoff
(for non-JVM memory usage)
JVM Heap
Java Heap
(for Flink runtime and application code)
Flink Managed Memory (off-heap)
(runtime algorithms)
Flink Network Memory
JVM Direct
Total Process Memory Size
Cutoff
(for non-JVM memory usage)
Memory/FsStateBackend
Memory/FsStateBackend
缓存哈希表、排序缓冲等
Cached hash tables, sorting buffers, etc.
RocksDBStateBackend
RocksDBStateBackend

TaskExecutor 内存模型

TaskExecutor memory model

解决方案1

Solution 1



RocksDBStateBackend 使用 managed memory

RocksDBStateBackend uses managed memory

Managed memory 不在使用 Java 堆内内存

Managed memory no longer uses Java on-heap memory

请求 managed memory 的两种方式

Two ways of requesting managed memory

- **请求由 Flink 申请、封装的内存段**

  Request memory segments that are allocated and wrapped by Flink

- **请求特定大小的内存预算，供消费主体使用**

  Request to reserve certain size of memory budget, to be used by the consumer

解决方案1

Solution 1

FLIP-49: 统一的 TaskExecutor 内存配置

FLIP-49: Unified Memory Configuration for TaskExecutors

- **统一流、批作业的内存配置**

  Unify memory configurations for streaming and batch processing

- **梳理 TaskExecutor 内存组成部分**

  Re-organize memory components of TaskExecutors

- **简化配置和计算逻辑**

  Simplify configuring and computing logics

已全部完成，将随 Flink 1.10 发布

Completed, will be available in Flink 1.10

特点2：同时运行 vs. 顺序运行

Characteristic 2 – Simultaneous execution vs. sequential execution

流处理作业

Streaming processing jobs

所有任务必须同时运行

All tasks must be running at the same time

使数据得以在节点之间顺畅流动

Allow data to flow through vertices

批处理作业

Batch processing jobs

Slot 复用：资源不足时，可以先运行部分任务，待其结束并释放资源后再运行其他任务

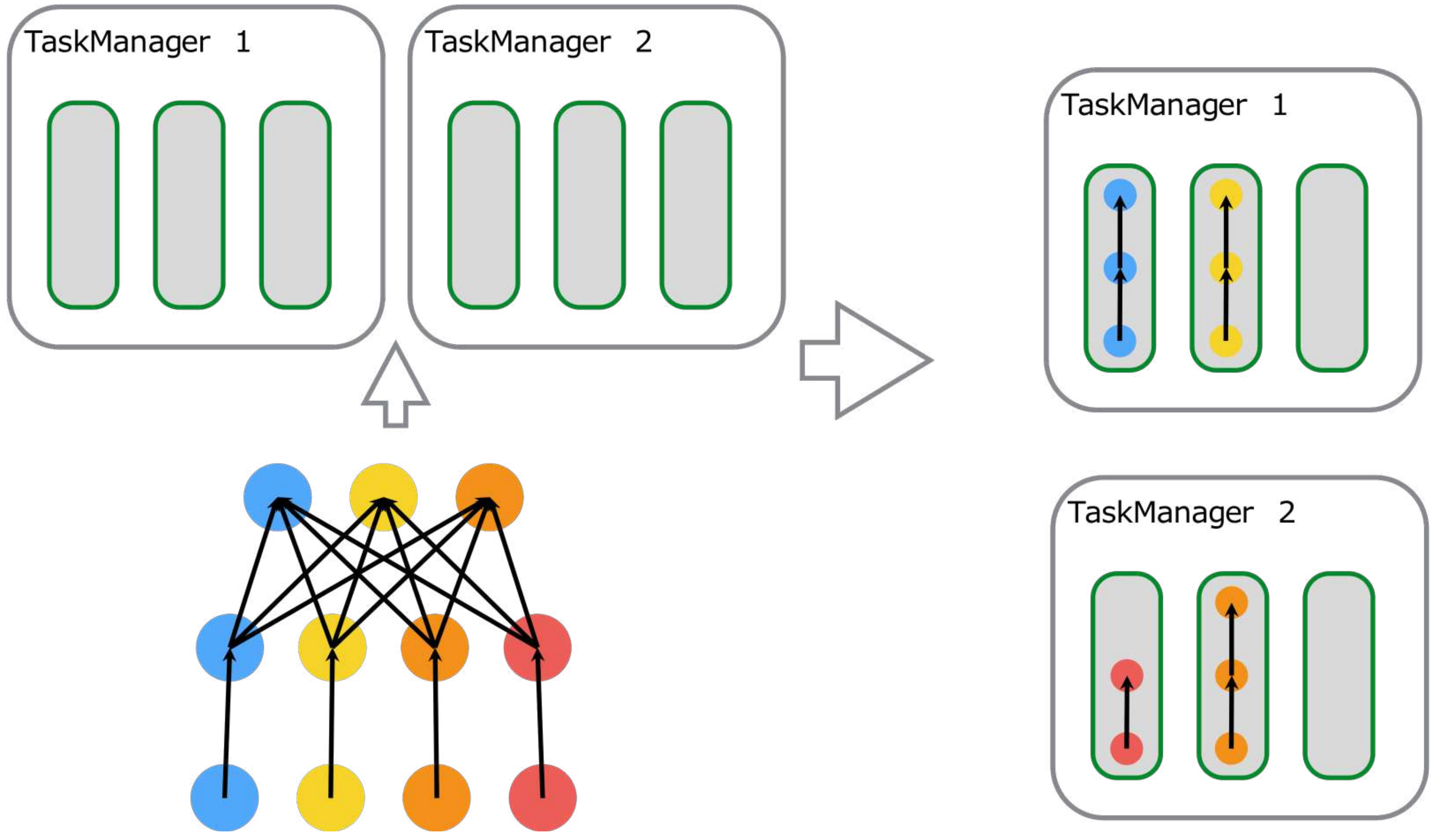
Slot Reusing - If resource is not enough, we can first execute some of the tasks, and wait for their finishing and releasing resources to execute the rest of the tasks.

任务依赖：任务间可能存在输入依赖，有时下游任务需要等上游任务结束才能开始运行

Task Dependency - There might be input dependencies between tasks, requiring downstream tasks to start only after upstream tasks finish

流作业的 Slot Sharing

Slot Sharing for streaming processing



Slot Sharing Group 中的任务可共用 slot

Tasks of vertices in the same slot sharing group can share slots

默认所有节点在一个 Slot Sharing Group 中

All vertices are in one slot sharing group by default

一个 slot 中相同任务只能有一个

There can be at most one task of the same vertex in each slot

优点

Advantages

运行一个作业所需的 slot 数量为最大并发数

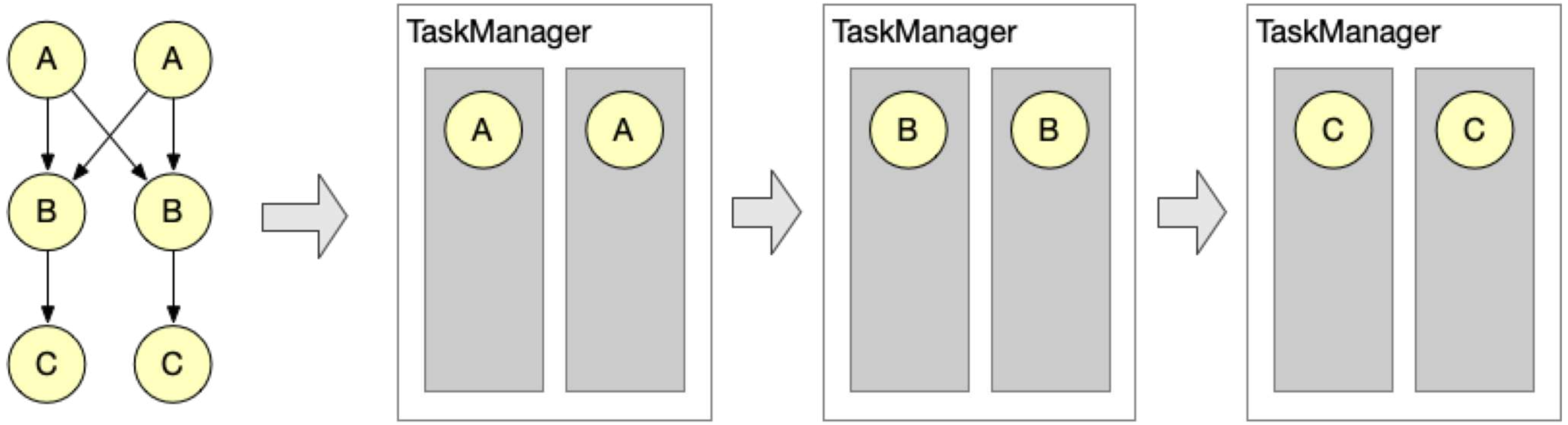
Number of slots needed for executing the job is its max vertex parallelism

相对负载均衡

Relative load balancing

挑战2-1：批作业 Slot Sharing 的问题

Challenge 2-1 – Problems of Slot Sharing for batch processing



Slot Sharing Group 中的任务并不同时运行

Tasks of the same slot sharing group do not run at the same time

资源限制，效率低

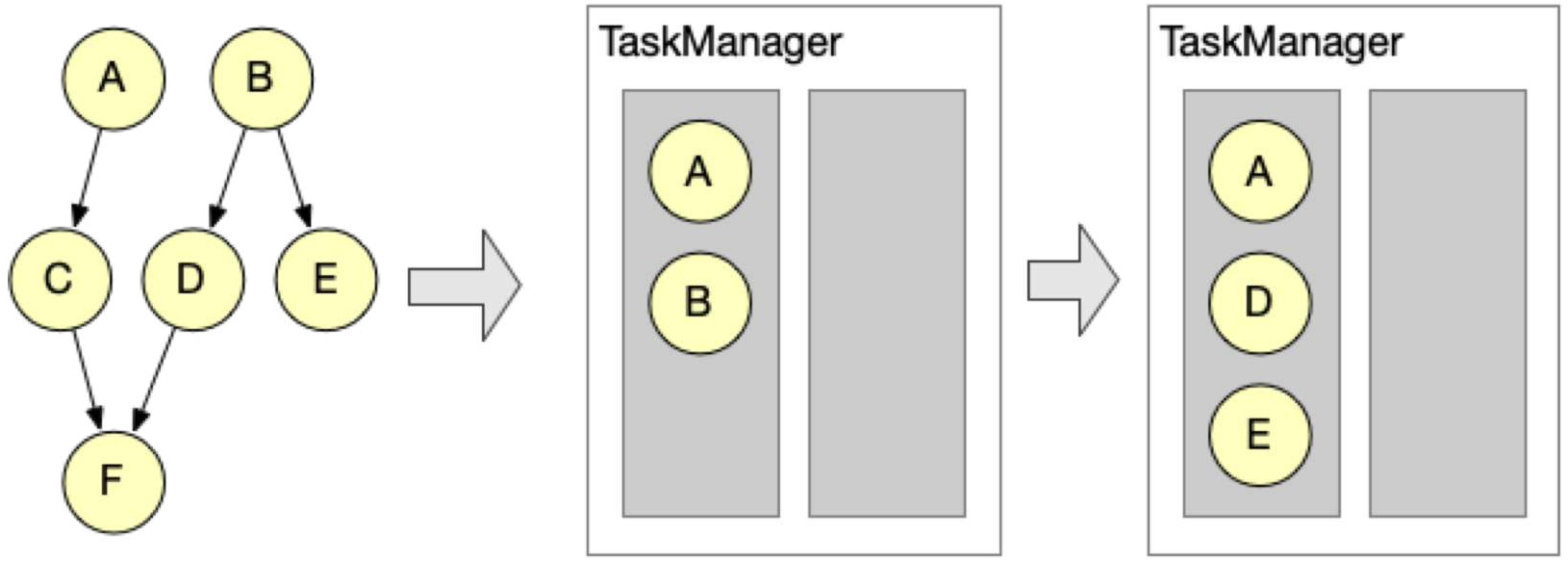
Low efficiency due to idle resources

Slot 中同时运行的任务数量是变化的

Amount of tasks running in the same slot changes

需要为后续任务预留出合适的资源

Need to reserve proper resources for upcoming tasks



解决方案2-1

Solution 2-1

哪些任务是有可能同时运行的?

Which tasks can possibly run at the same time?

Pipelined Region

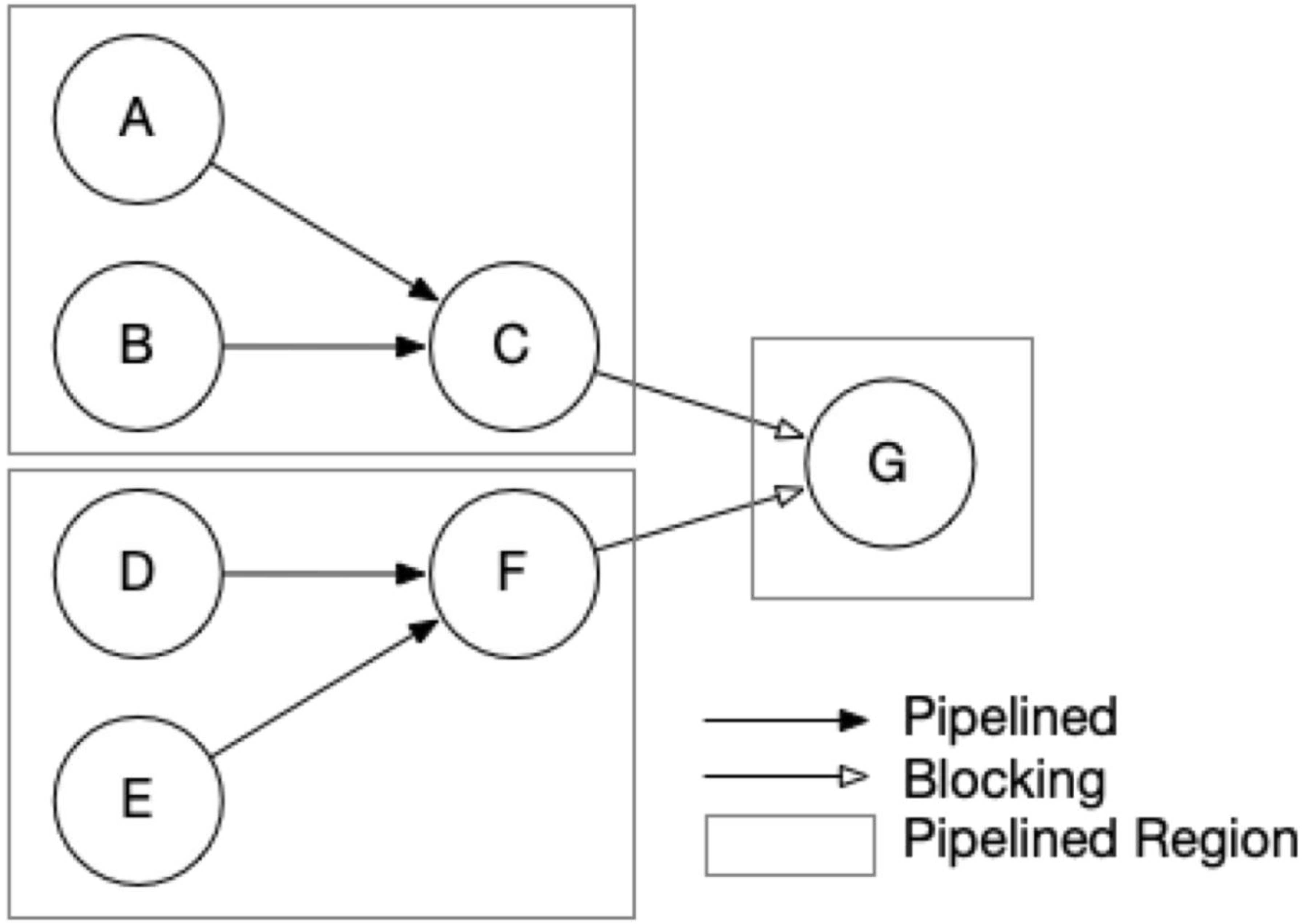
Pipelined Region

算子可以使用 slot 中的多少资源?

How many of the slot resources can a operator use?

基于 fraction 的相对资源预算

Fraction based relative resource quota



解决方案2-1

Solution 2-1

编译阶段：

At compiling:

设置 slot sharing group 为 pipelined region

Set slot sharing groups to pipelined regions

根据 slot sharing group 中需要使用某种资源的算子数量为算子设置 fraction

Set fraction for operators according to number of operators in the slot sharing group that use certain resource

部署阶段：

At deploying:

根据 slot 资源及算子的 fraction 决定算子资源预算

Decide operator's resource quota according to the slot's resource and the operator's fraction

解决方案2-1

Solution 2-1

FLIP-53: 细粒度的算子资源管理

FLIP-53: Fine Grained Operator Resource Management

- **细粒度的算子资源预算管理**

  Fine grained operator resource quota management

- **Slot Sharing 的处理**

  Slot sharing handling

大部分工作已完成，预计 Flink 1.10 能够全部完成

Mostly finished, should be completed in Flink 1.10

挑战2-2：如何满足算子确定的资源需求

Challenge 2-2: How to satisfy deterministic operator resource requirements

解决方案2-1 本质上解决了一个什么问题？

Essentially, what problem does solution 2-1 solve?

算子可以使用多少资源

How many resources an operator can use

这个解决方案隐含的假设是什么？

What is the implicit assumption behind this solution?

算子使用的资源可多可少

The amount of resource an operator uses does not really matter

不一定成立

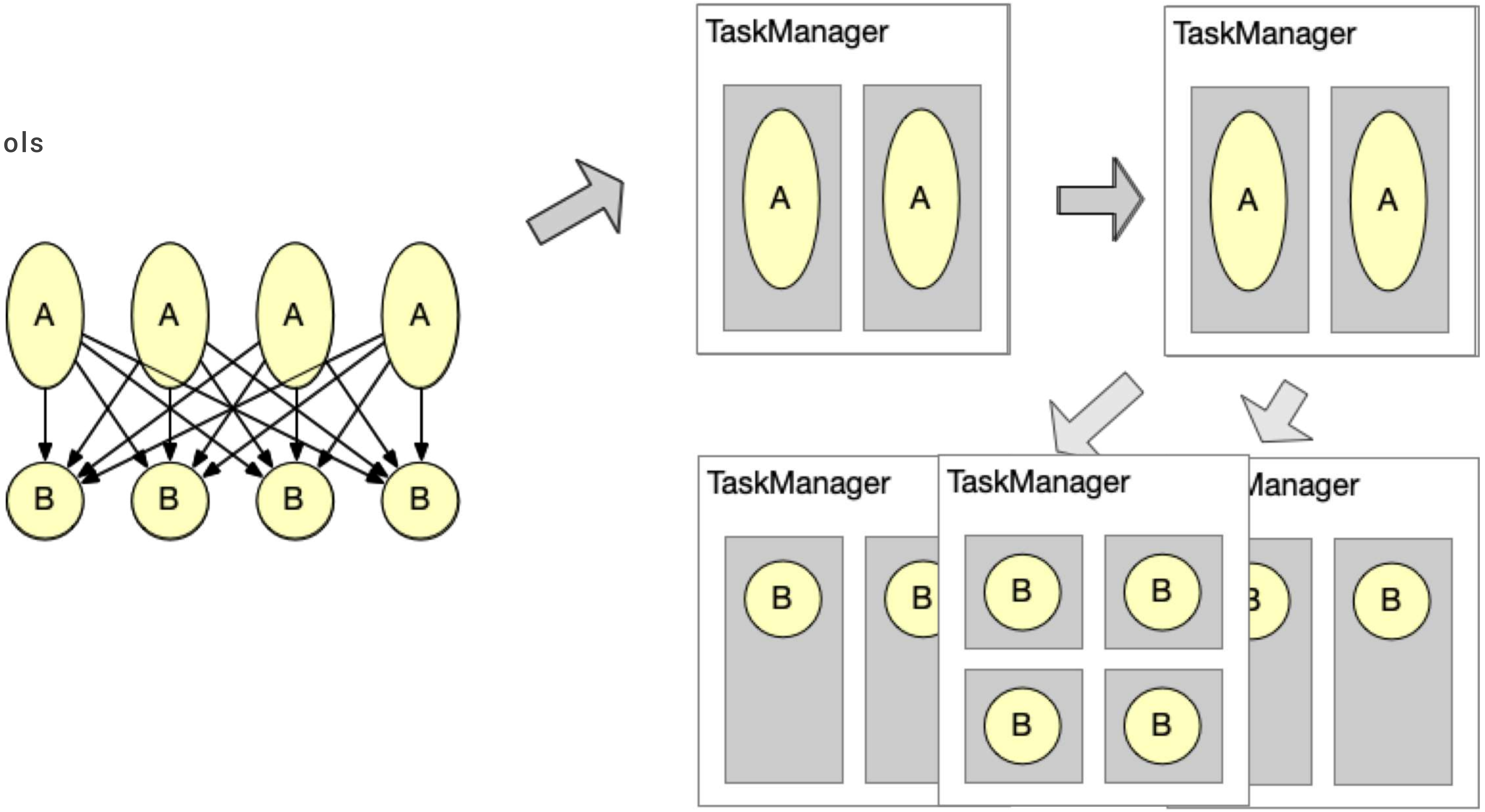
Not always true

挑战2-2：如何满足算子确定的资源需求

Challenge 2-2: How to satisfy deterministic operator resource requirements

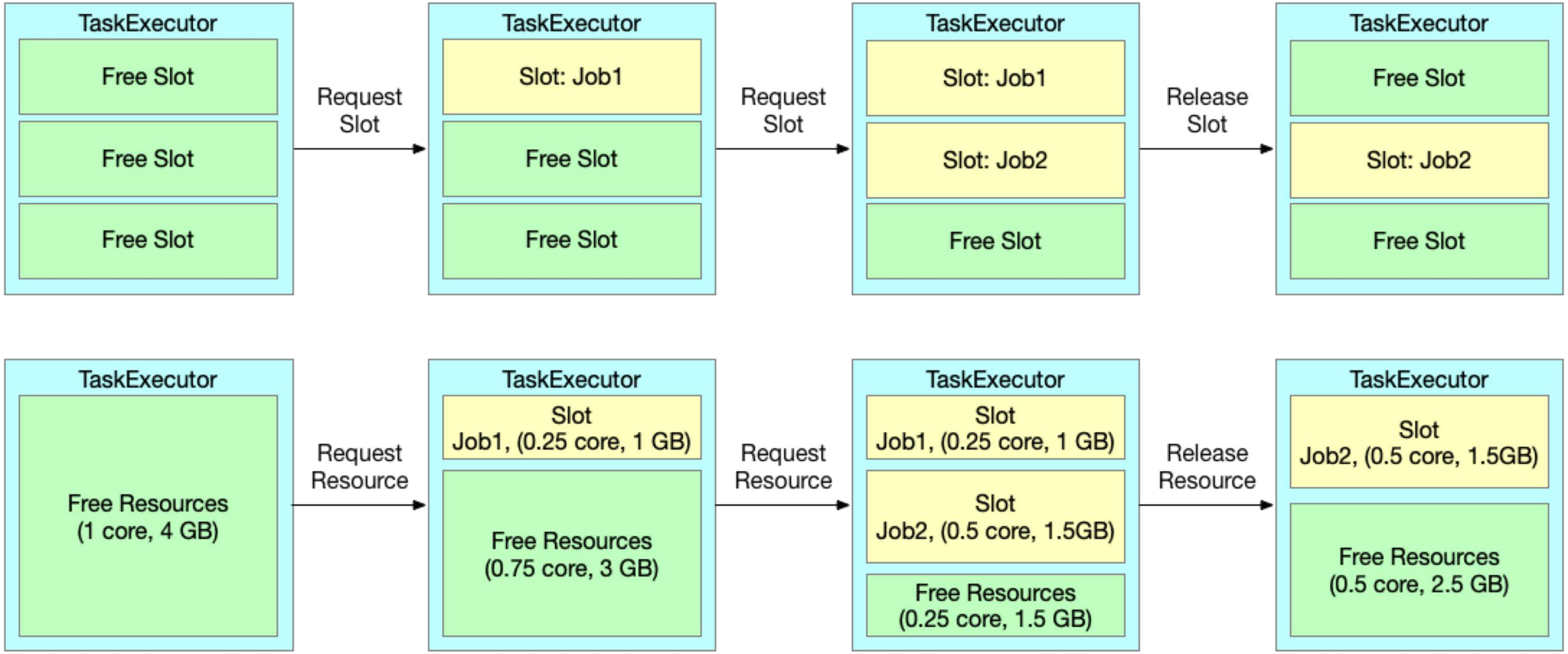
我们姑且先假设算子的资源需求是已知的
Let's assume operator resource requirements are known

- 经验性预估、半自动化/自动化工具
  Empirical estimation, semi-automatic/automatic tools



解决方案2-2

Solution 2-2



解决方案2-2

Solution 2-2

FLIP-56: 动态 slot 分配

FLIP-56: Dynamic Slot Allocation

动态创建、销毁 slot

Dynamically create and destroy slots

TM 资源使用记账

Bookkeeping of TM resource usages

非细粒度资源请求的处理

Handling of non fine grained resource requirements

开发中，按目前进度很难在 Flink 1.10 完成，预计下一个版本可全部完成

In developing, may not make Flink 1.10, should be completed in the next release

资源管理的两种理念

Two philosophies of resource management

自顶向下的资源管理

Top-Down Resource Management

Slot Sharing
Slot sharing

作业整体资源决定单个任务资源
First decide job resource, then derive task resource

配置难度低，效率一般
Easy to config, with normal efficiency

适合基础用户、小规模作业
Recommended for primary users and small jobs

自底向上的资源管理

Bottom-Up Resource Management

细粒度资源管理
Fine grained resource management

单个任务资源决定作业整体资源
First decide task resource, then derive job resource

配置难度高，效率高
Difficult to config, with high efficiency

适合深度用户、大规模作业
Recommended for expert users and big jobs

两种资源管理理念反映出了不同场景下的需求差异
The two philosophies reflects different demands in different scenarios
两种理念均有存在的价值
The two philosophies both should be preserved

流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

02

Contents
目录

01 流批一体的资源管理

Resource Management for Streaming and Batch Processing

02 流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

a 如何“完美”地调度流批作业？

How to schedule batch and streaming job perfectly?

b 基于Concurrent-Group的调度

Scheduling based on concurrent group

c 测试效果与未来发展

Test result and future work

Contents
目录

01 流批一体的资源管理

Resource Management for Streaming and Batch Processing

02 流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

a 如何“完美”地调度流批作业？

How to schedule batch and streaming job perfectly?

b 基于Concurrent-Group的调度

Scheduling based on concurrent group

c 测试效果与未来发展

Test result and future work

调度系统的目标

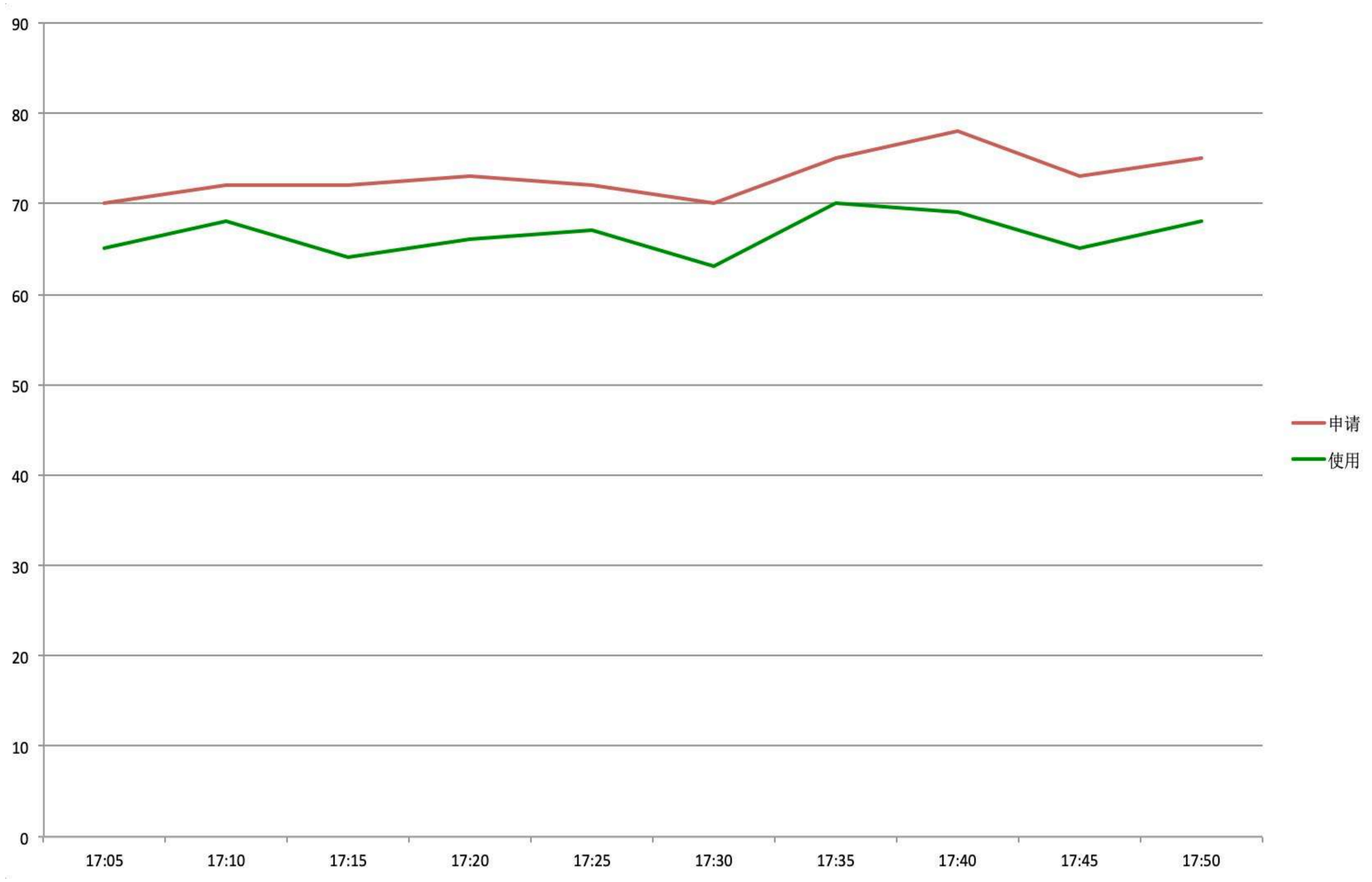
The goal of scheduling system

集群层面：

The whole cluster level:

提升集群实际利用率

Increase the actual utilization of cluster



调度系统的目标

The goal of scheduling system

作业层面：

For a specified job level:

- **资源不足时能够以最小资源运行**

  Can run with a minimal resource

- **资源充足时能够充分利用资源**

  Fully utilize the resource in the cluster

- **不占用当前用不上的资源**

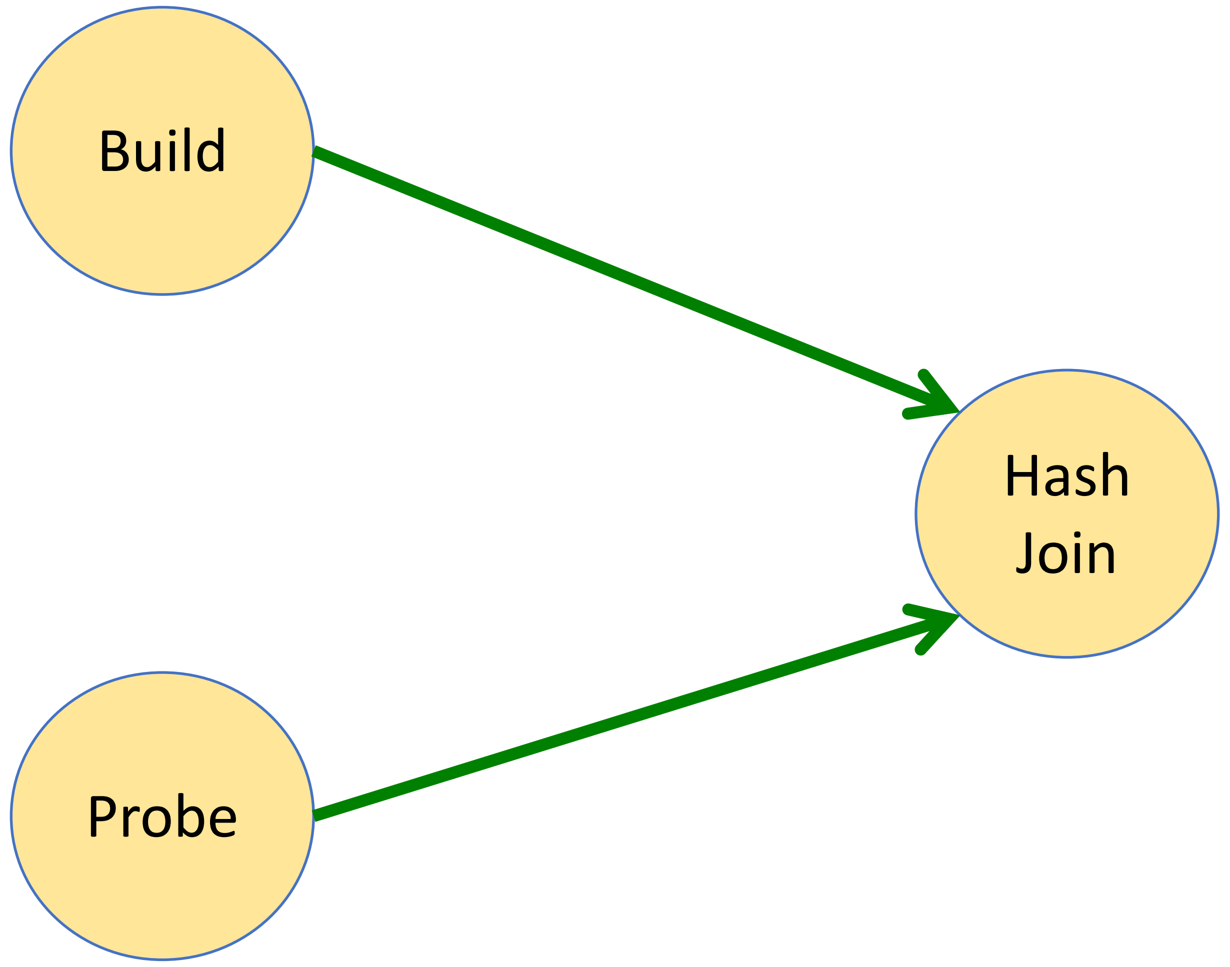
  Occupy resource only when really need

批作业的特性

The characteristics of batch jobs

算子可能按序读取上游数据：

The operator may consume its inputs in order:

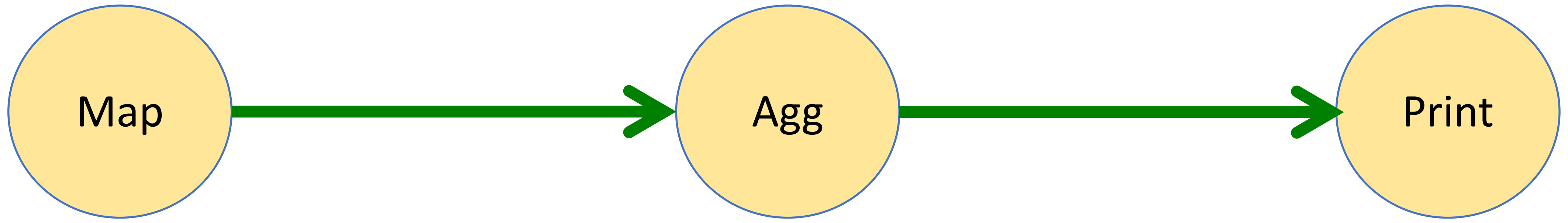


批作业的特性

The characteristics of batch jobs

算子需要计算一定时间后才会输出结果：

The operator may compute for a long time before outputing the result:



Contents
目录

01 流批一体的资源管理

Resource Management for Streaming and Batch Processing

02 流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

a 如何“完美”地调度流批作业？

How to schedule batch and streaming job perfectly?

b 基于Concurrent-Group的调度

Scheduling based on concurrent group

c 测试效果与未来发展

Test result and future work

如何划分Group?

How to split a job graph into concurrent groups?

划分规则:

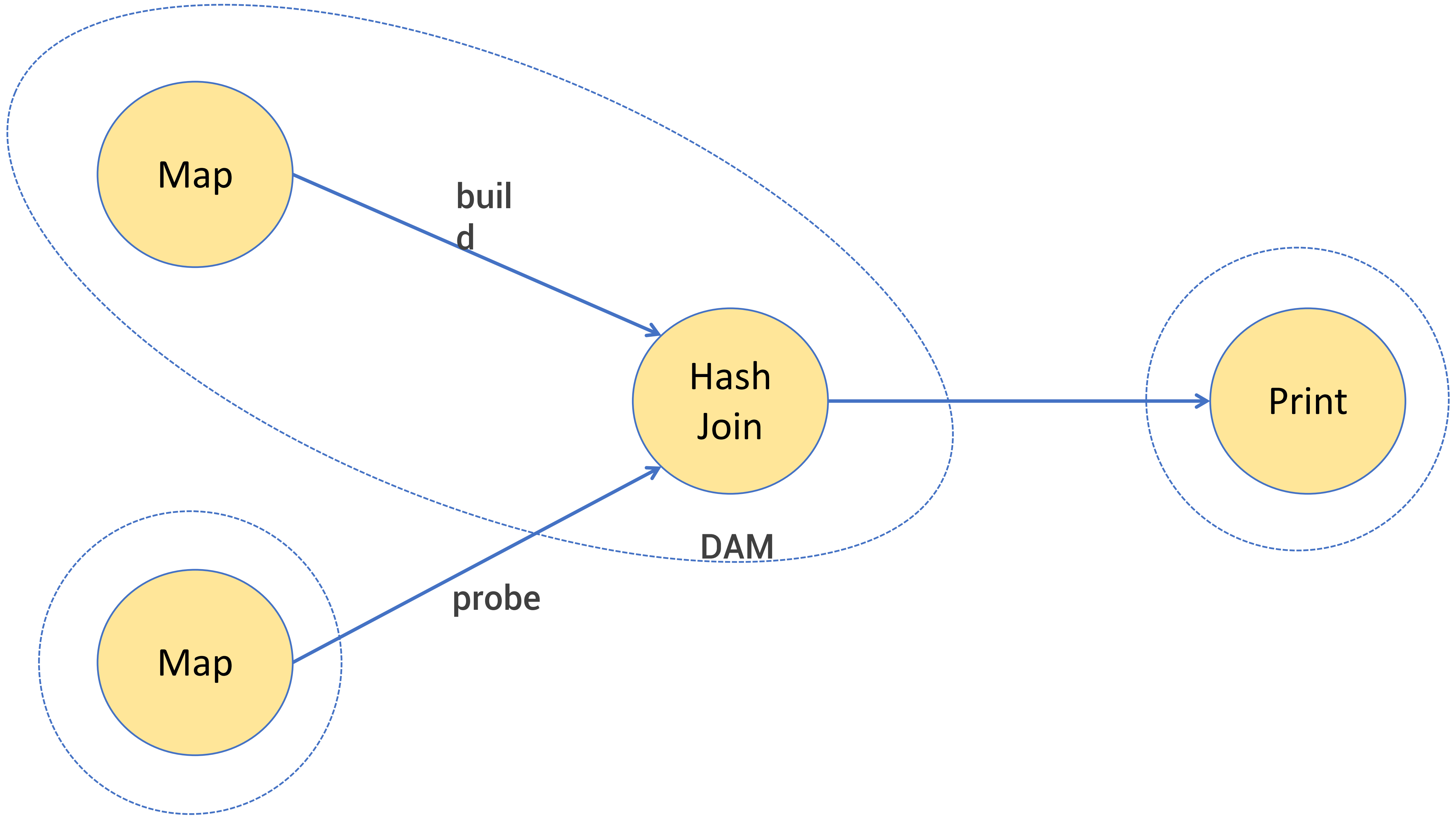
The rule for group split:

- **如果一个节点是DAM的，则与它的下游不放到同一个Group中。**

  Put the vertex and its downstream into different groups if it is dam.

- **如果一个节点的多个输入有读取顺序，则后读的输入跟它不放在同个Group中。**

  Put the upstream with later order into a different group.



Group排序

The groups should be sorted and in order

排序依据：

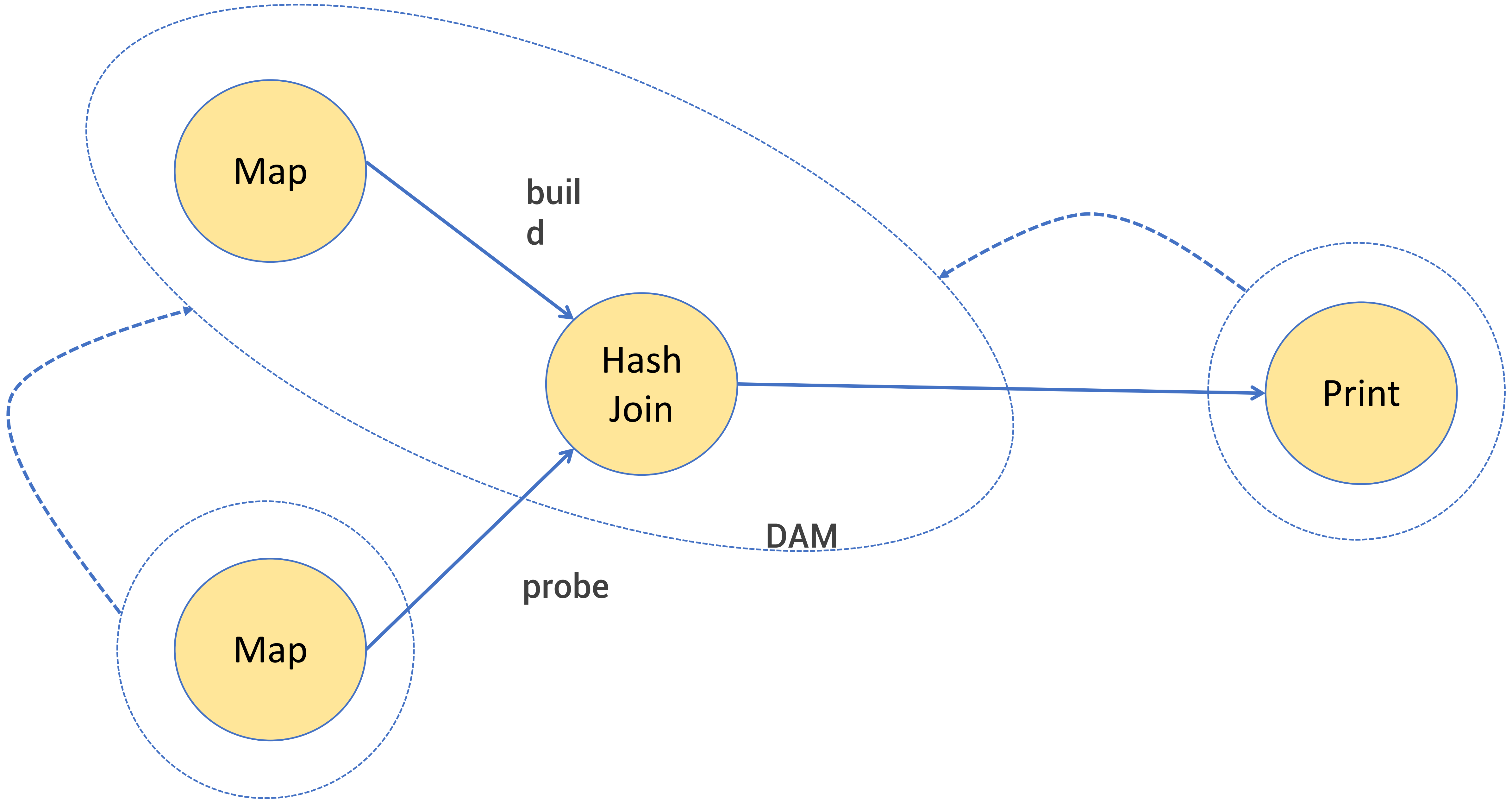
The rule for sorting groups：

- **根据节点间的输入输出顺序**

  The data flow between two vertices

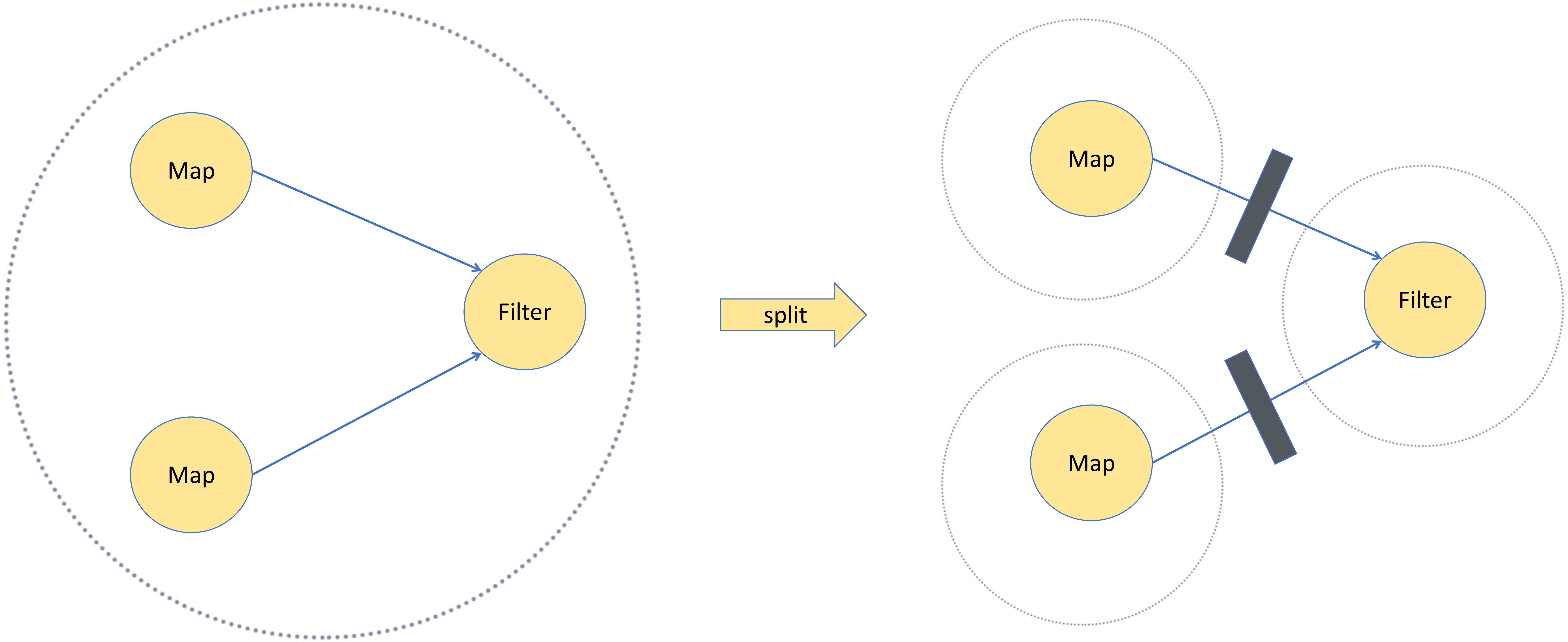
- **输入的读取顺序**

  The order of data consumed by the same consumer



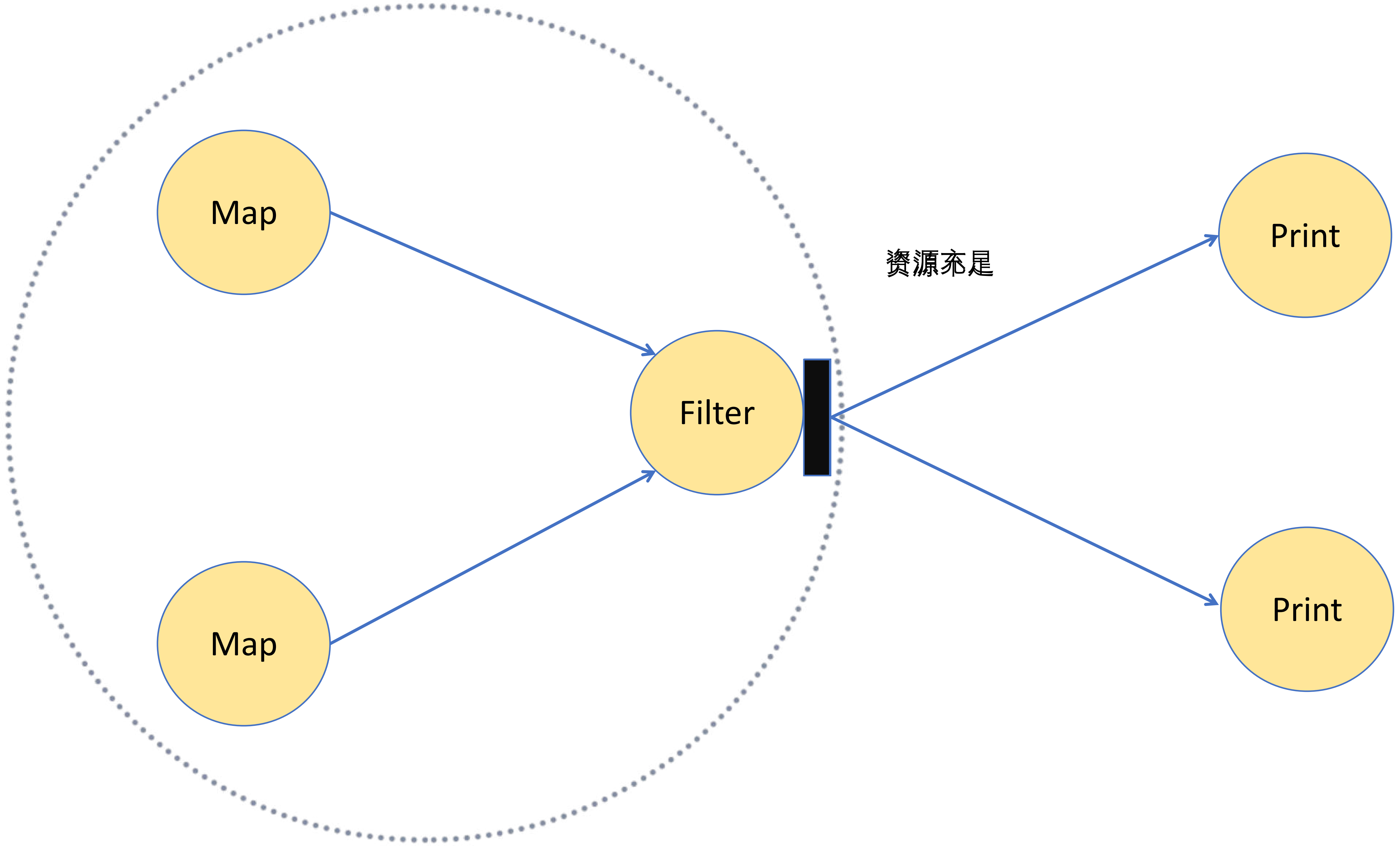
Group资源不足?

What to do if there is no enough resource for a group?



下游Group何时启动？

When should the downstream groups be scheduled?



Contents
目录

01 流批一体的资源管理

Resource Management for Streaming and Batch Processing

02 流批一体的任务调度

Task Scheduling for Streaming and Batch Processing

a 如何“完美”地调度流批作业?

How to schedule batch and streaming job perfectly?

b 基于Concurrent-Group的调度

Scheduling based on concurrent group

c 测试效果与未来发展

Test result and future work

测试效果

Test results

16台机器上跑通
10T的TPC-DS测试

Pass TPC-DS tests on 16 machines

性能提升13%

Improve the performance by 13%

未来工作

Future work

支持资源抢占

Support resource preemption

统一Pipeline和Blocking的资源

Unify resource usage of pipeline and blocking

THANKS