Dryad

Distributed Data Parallel Programs from Sequential Building Blocks

Yahn-Chung (Andrew) Chen

Overview

- Design Goals
- The Design of Dryad
- Discussion & QA

Problem

- How can we make it easier for developers to write efficient parallel and distributed application?

Problem

- How can we make it easier for developers to write efficient parallel and distributed application?

- Developers no need to have understanding in standard concurrency mechanism. (No OS course in the future)

Dryad goal

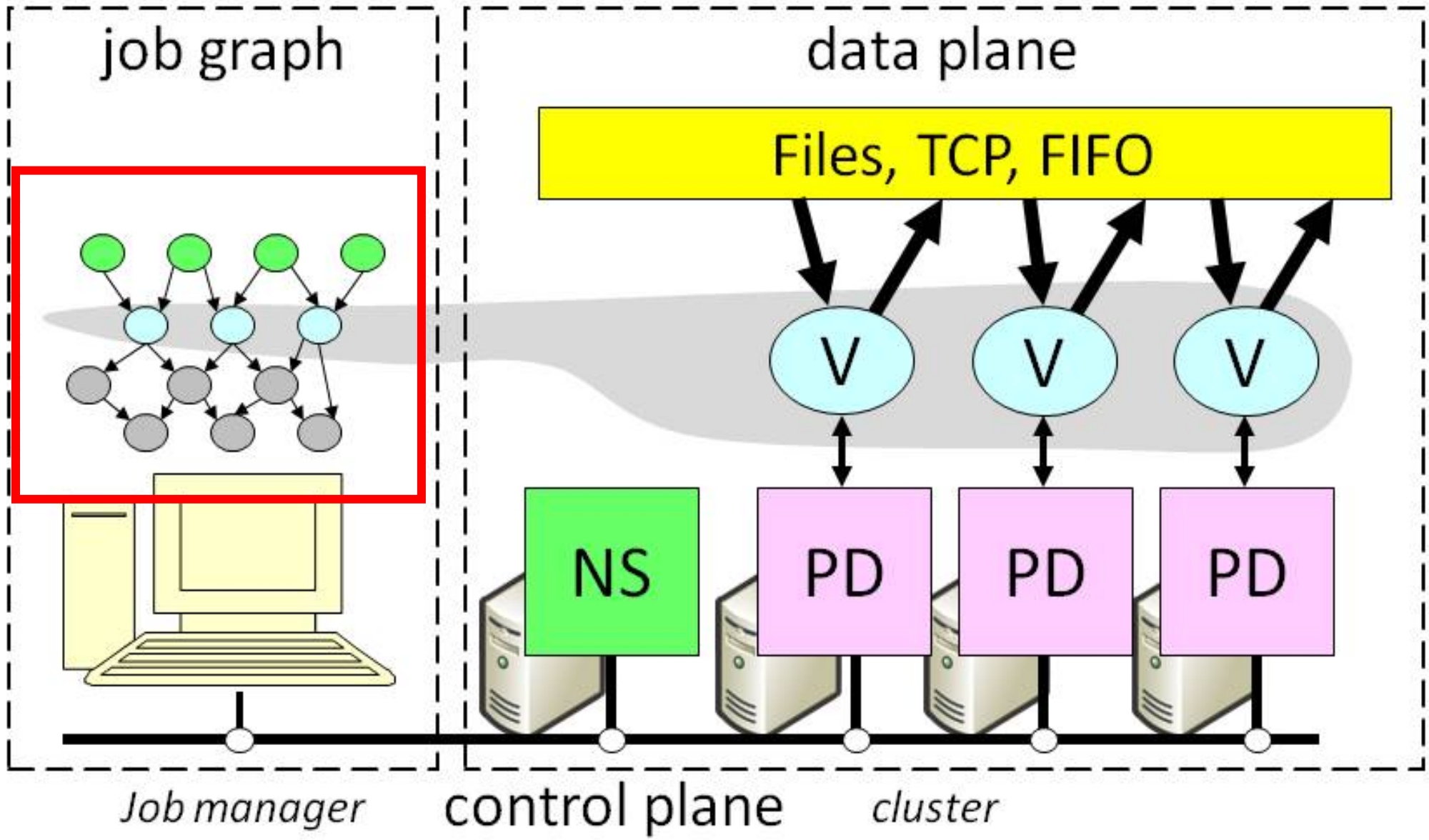
- General-purpose execution environment for distributed, data-parallel applications
  - Focus on throughput than latency

- Automatic management for scheduling, distribution, fault tolerance, etc.

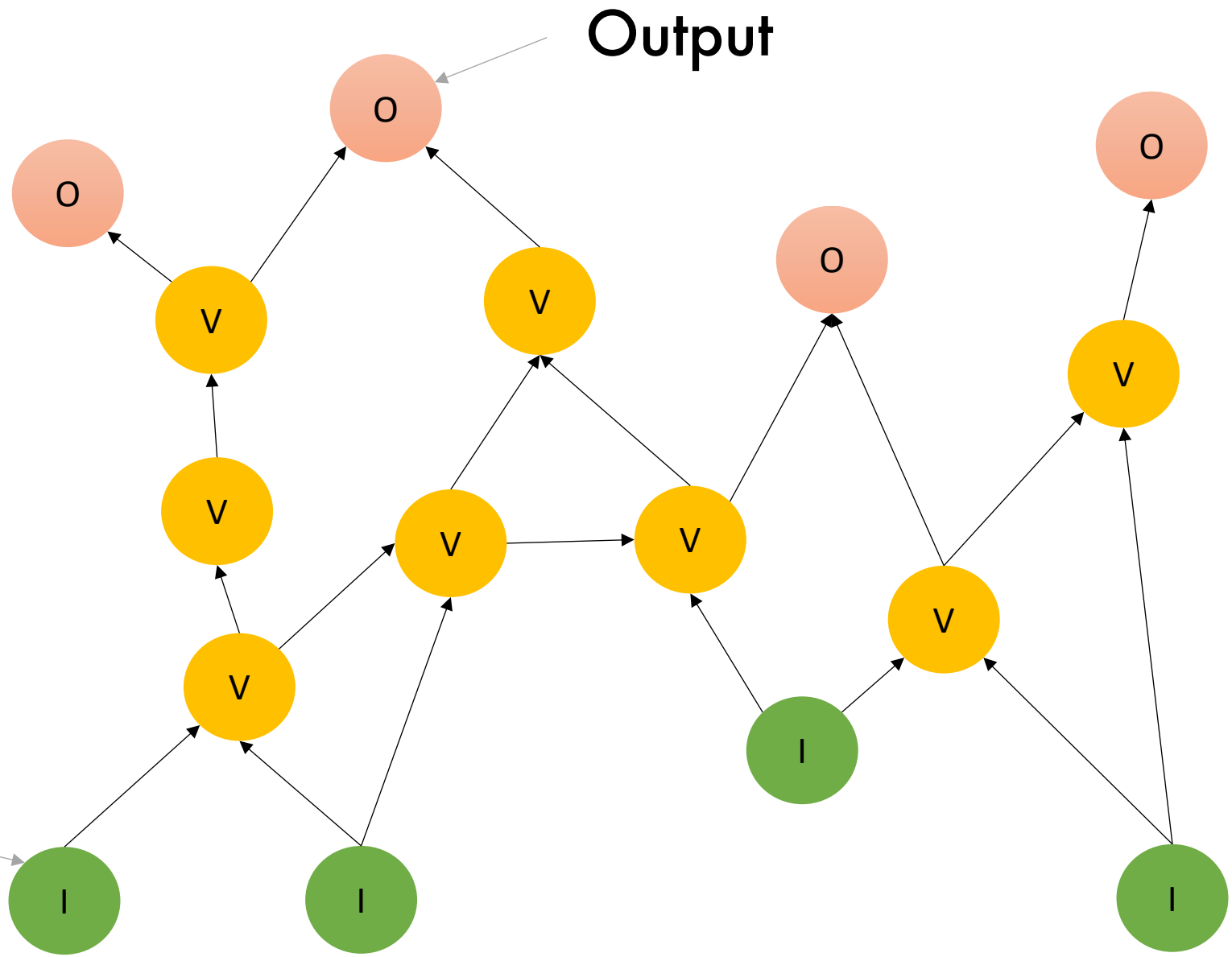
What's Dryad?

- Many programs can be represented a distributed execution graph

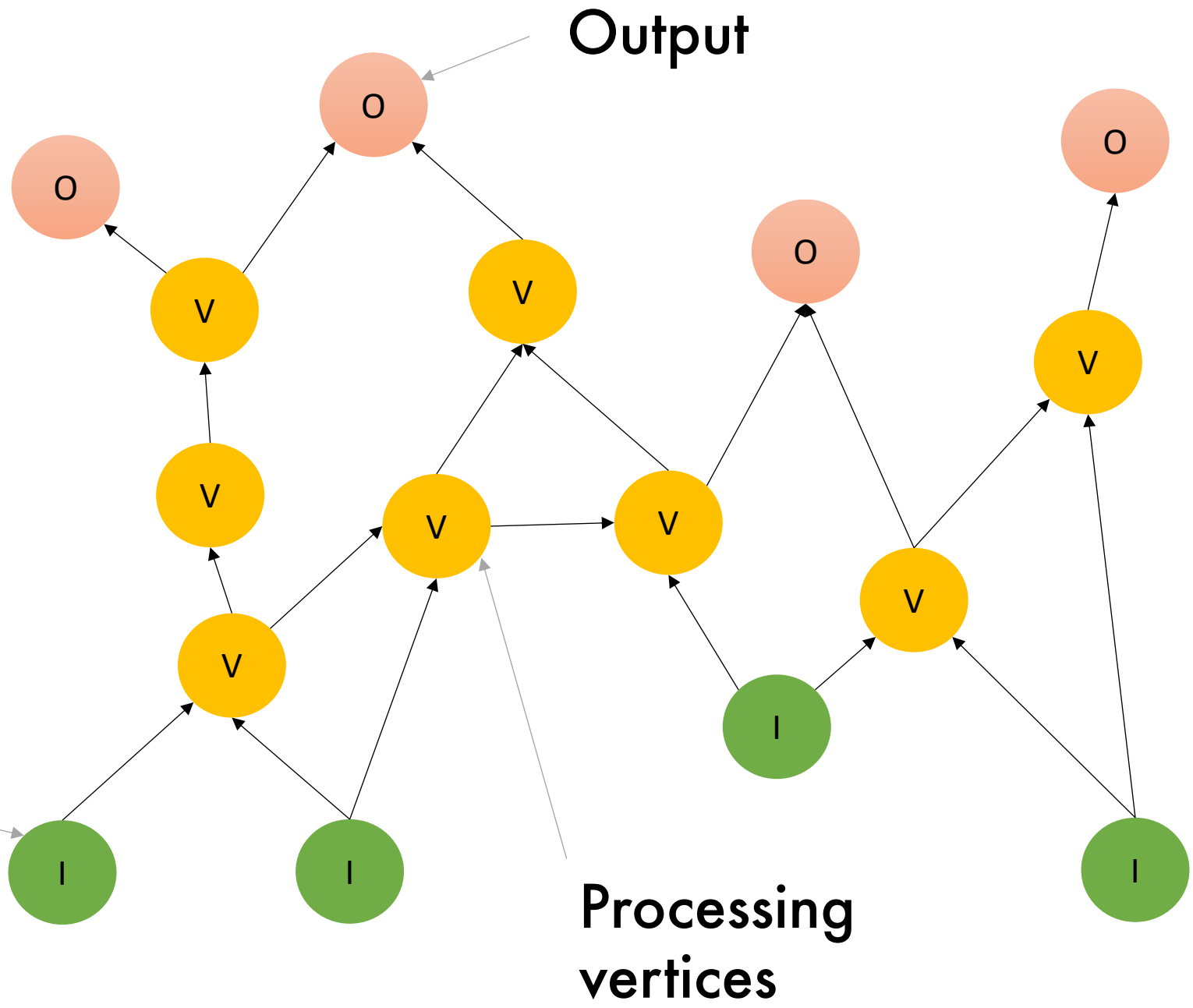
- Dryad is the middleware abstraction run for you
  - Dryad see arbitrary graphs
  - Above Dryad is just graph manipulation


job graph
data plane
Files, TCP, FIFO
V
V
V
NS
PD
PD
PD
Job manager
control plane
cluster

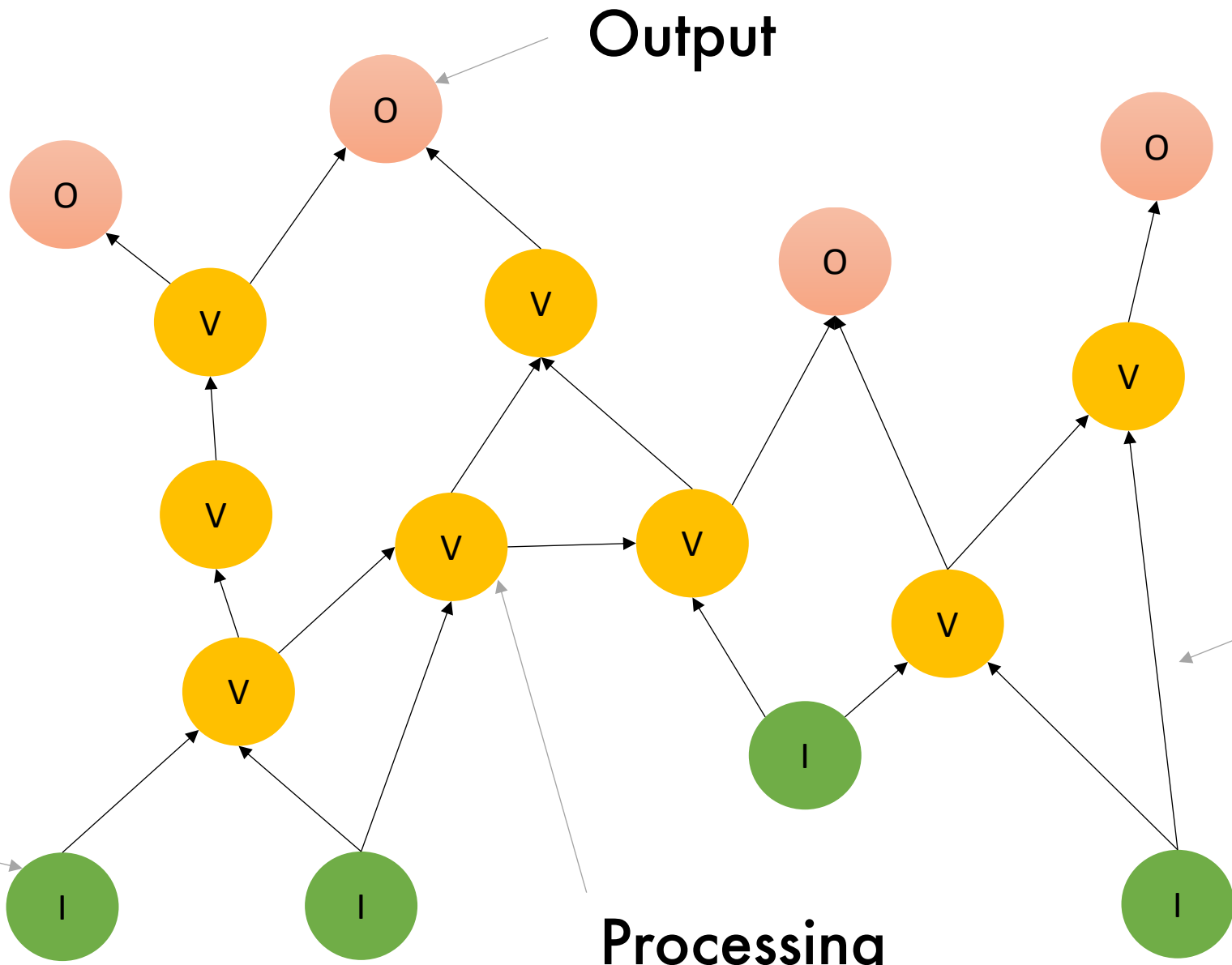
Directed Acyclic Graph

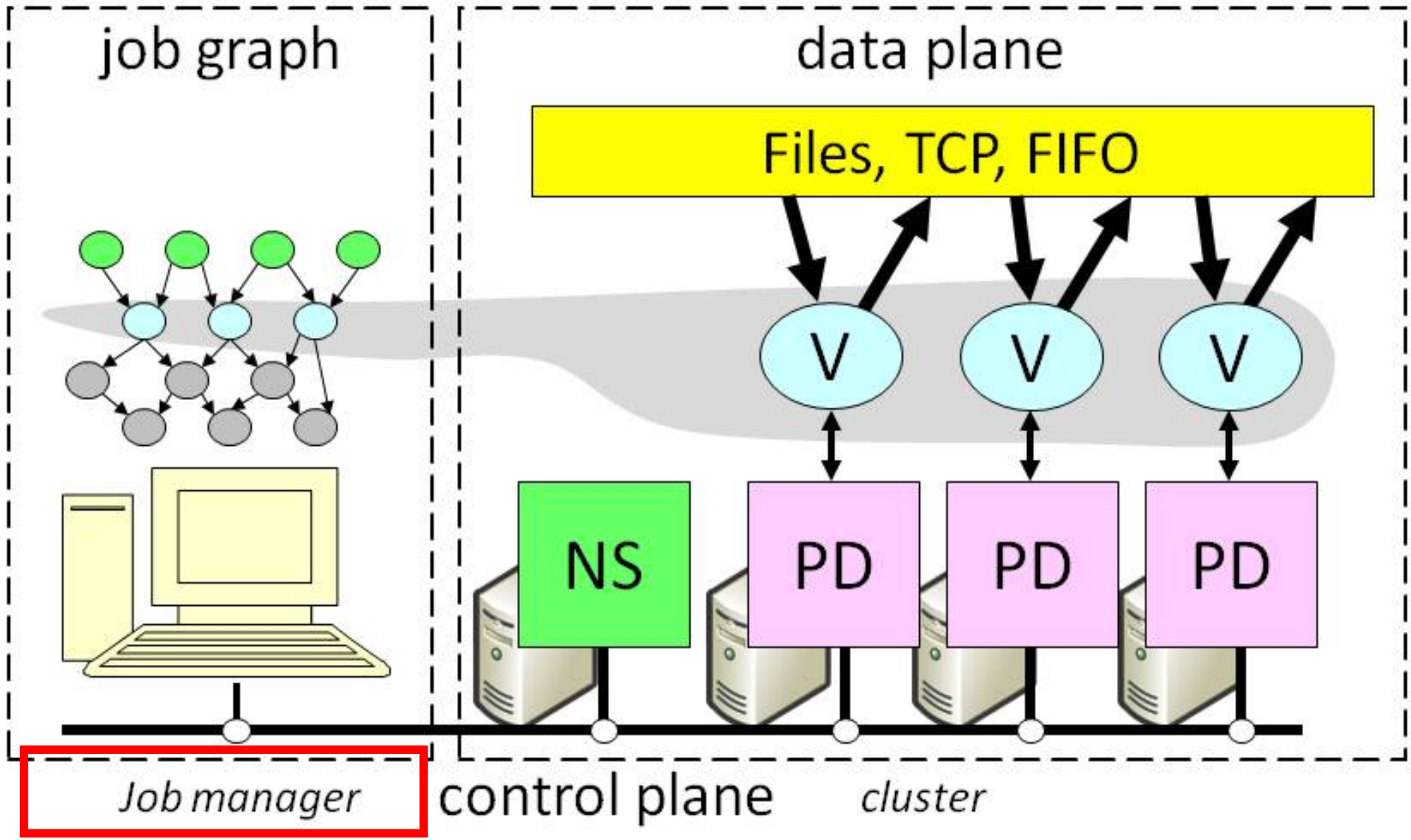


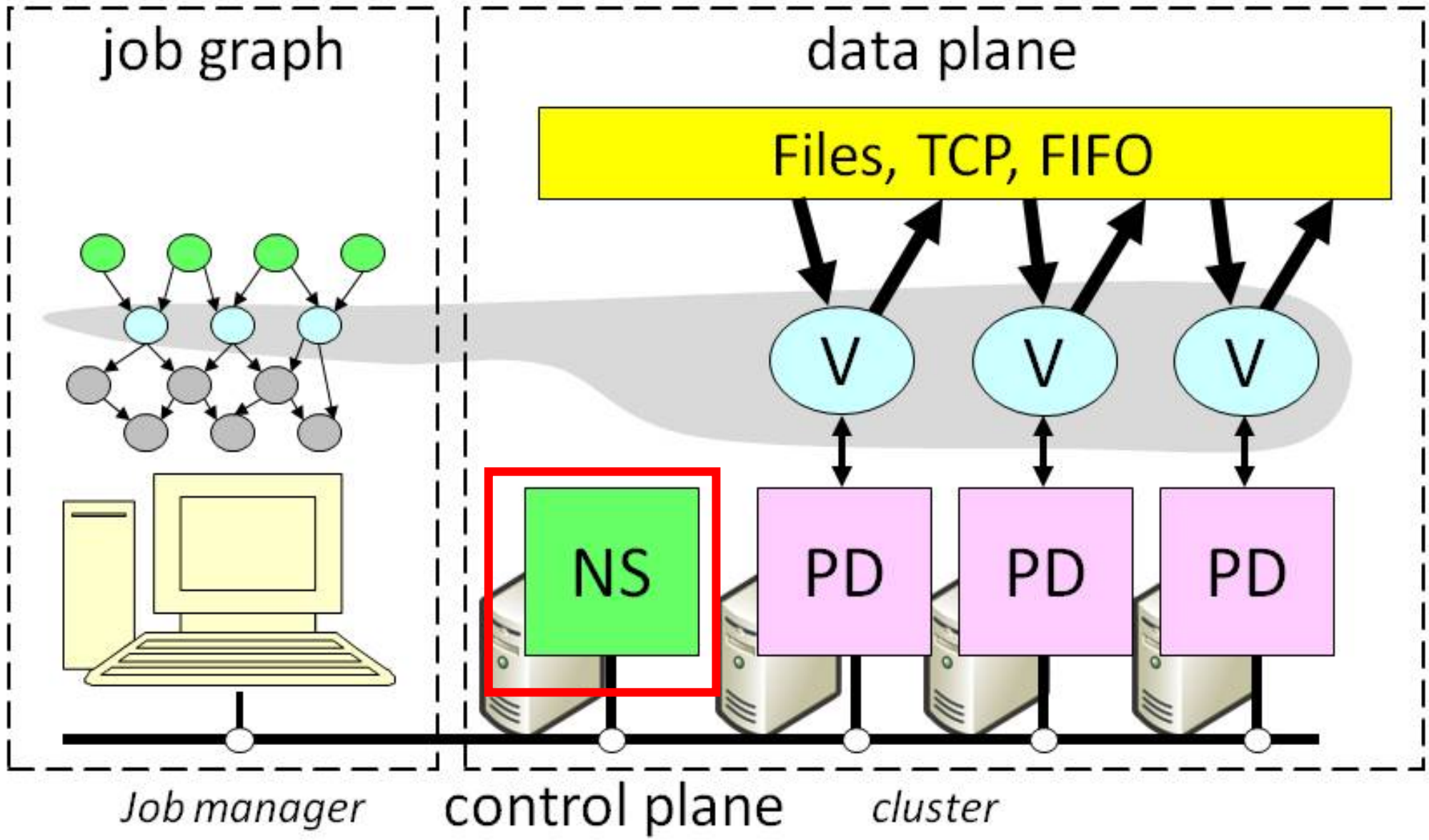
Directed Acyclic Graph

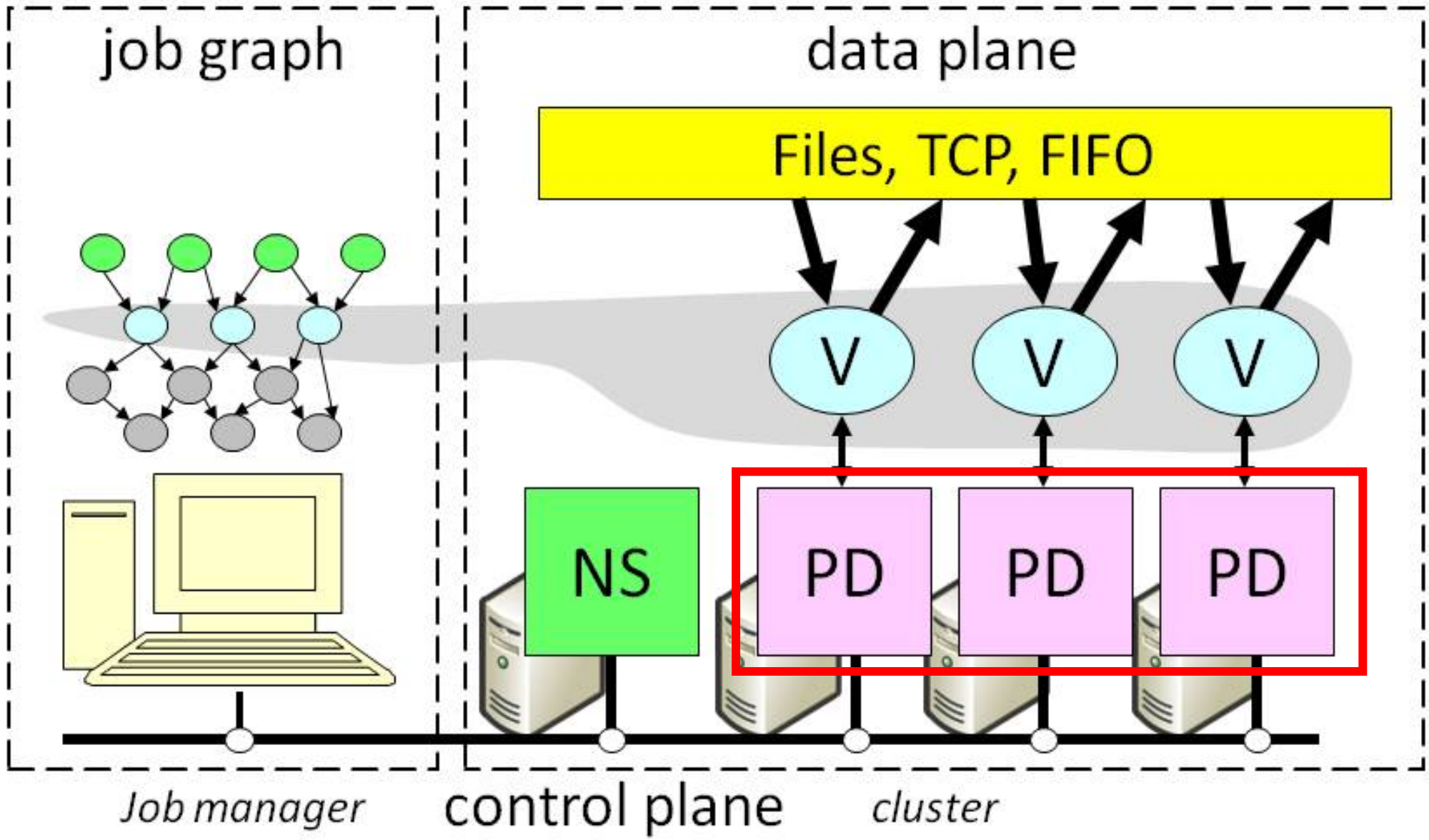


Directed Acyclic Graph

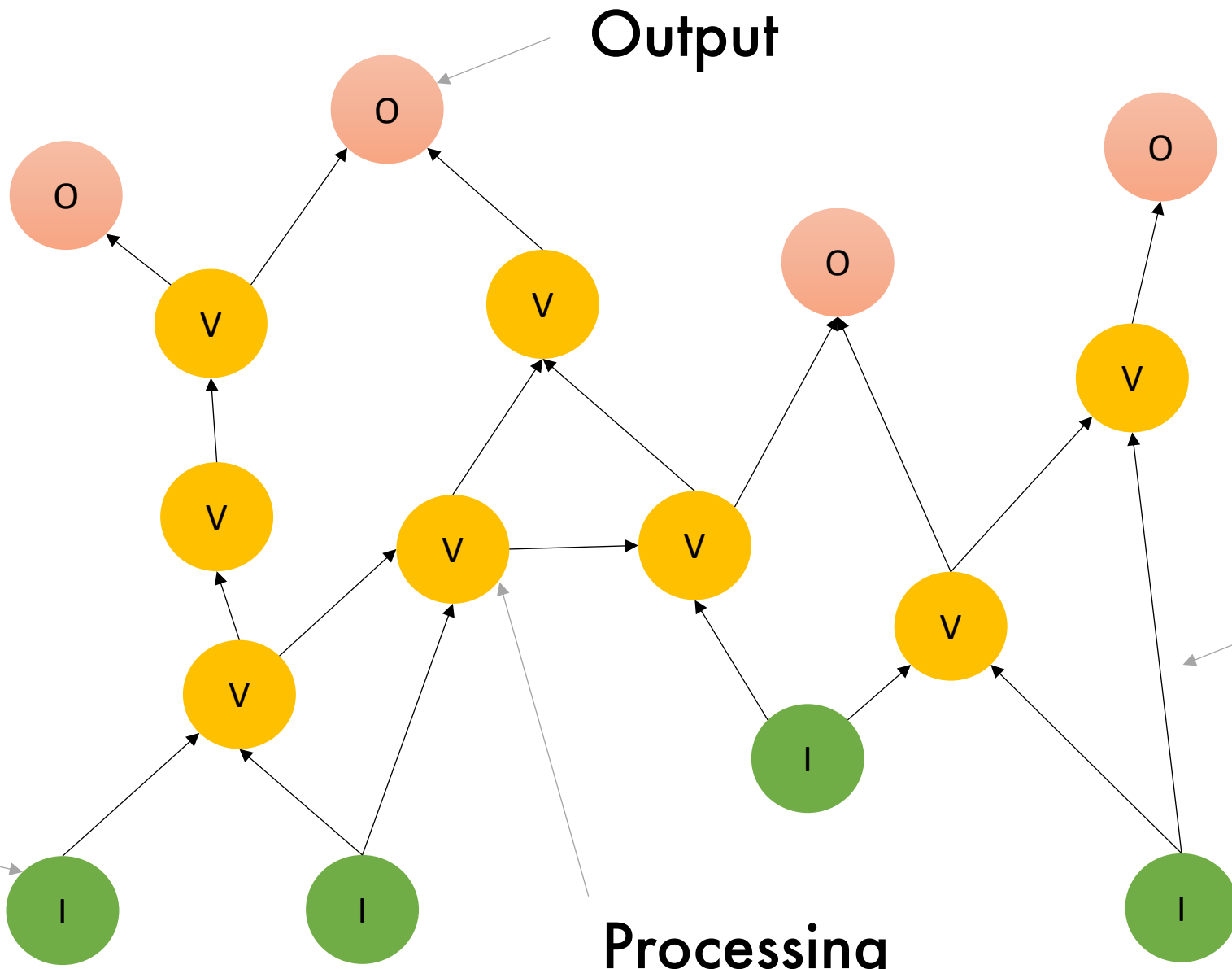



job graph
data plane
Files, TCP, FIFO
V
V
V
NS
PD
PD
PD
Job manager
control plane
cluster


job graph
data plane
Files, TCP, FIFO
V
V
V
NS
PD
PD
PD
Job manager
control plane
cluster


job graph
data plane
Files, TCP, FIFO
V
V
V
NS
PD
PD
PD
Job manager
control plane
cluster

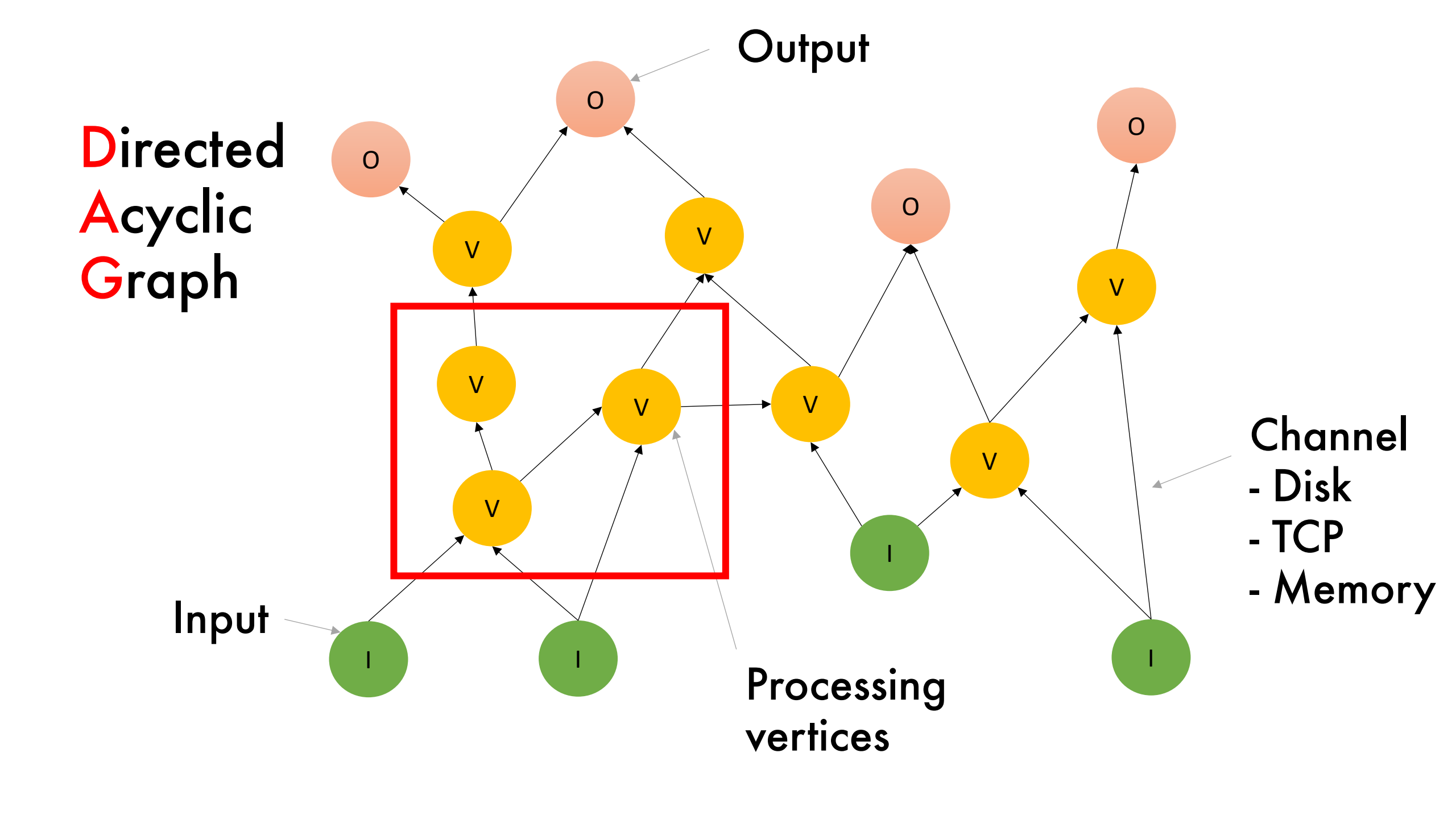
Directed Acyclic Graph



Channel
- Disk
- TCP
- Memory

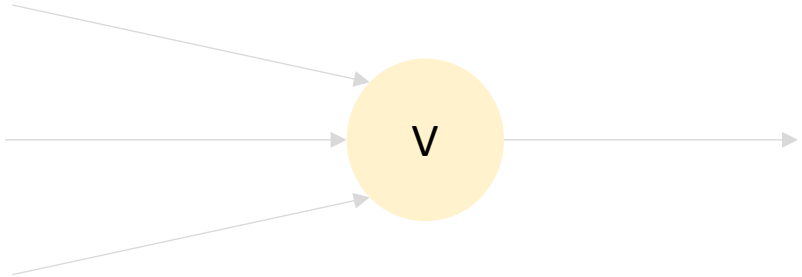
Channel Types

- Sequence of structed items
- Implementation
  - Temporary Disk file
    - Item are serialized in buffer
  - TCP pipe
    - Item are serialized in buffer
  - Shared-memory FIFO
    - Pass pointer to items directly


Directed
Acyclic
Graph
Output
O
O
O
O
V
V
V
V
V
V
V
V
V
Channel
- Disk
- TCP
- Memory
Input
I
I
I
Processing
vertices

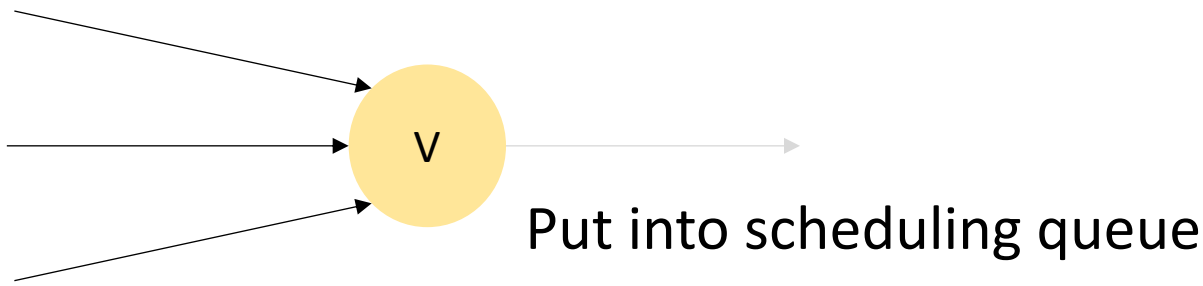
Job execution

- Scheduler in Job Manager
- Exec of vertex
  - Version number
  - Exec record
    - State
    - Predecessor's version



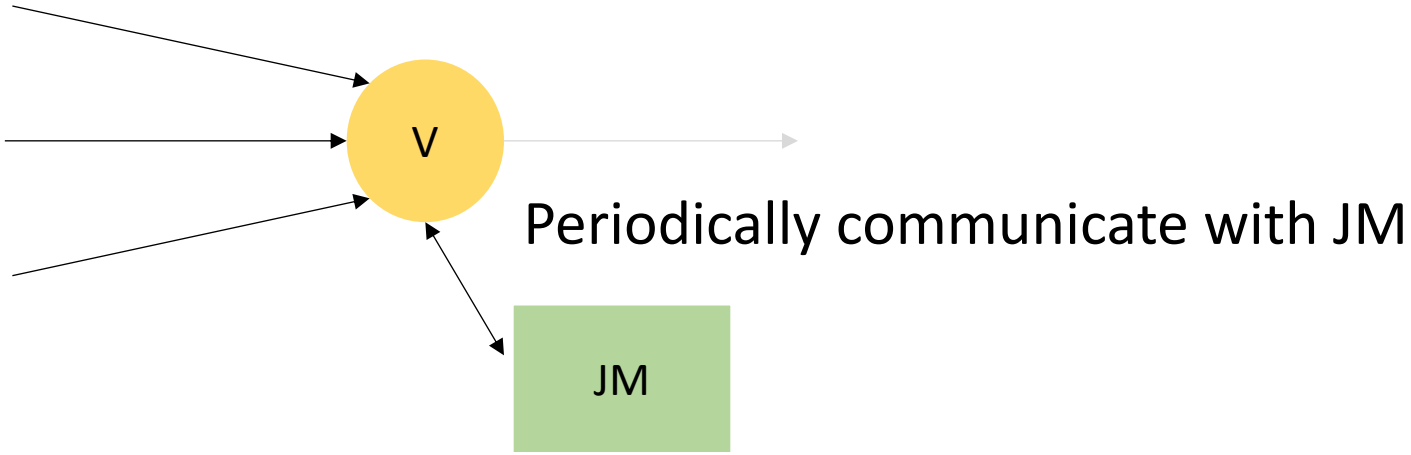
Job execution

- Scheduler in Job Manager
- Exec of vertex { Version number, Exec record } { State, Predecessor's version }



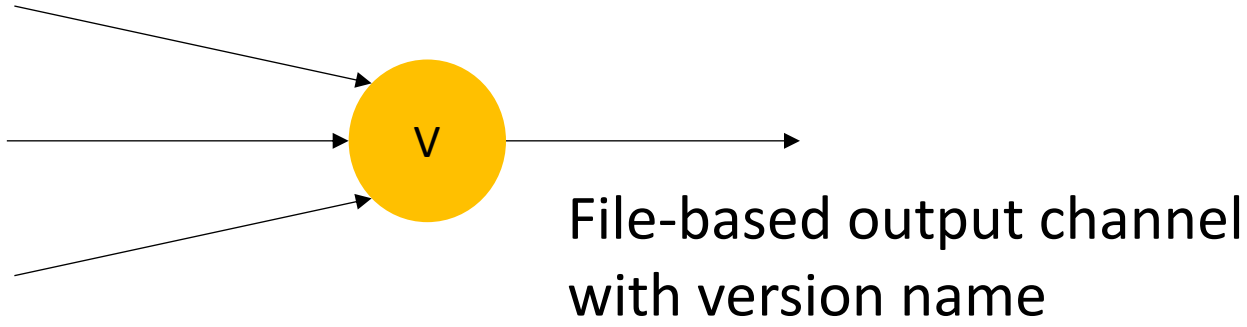
Job execution

- Scheduler in Job Manager
- Exec of vertex { Version number, Exec record } { State, Predecessor's version }



Job execution

- Scheduler in Job Manager
- Exec of vertex { Version number, Exec record } { State, Predecessor's version }



Job execution

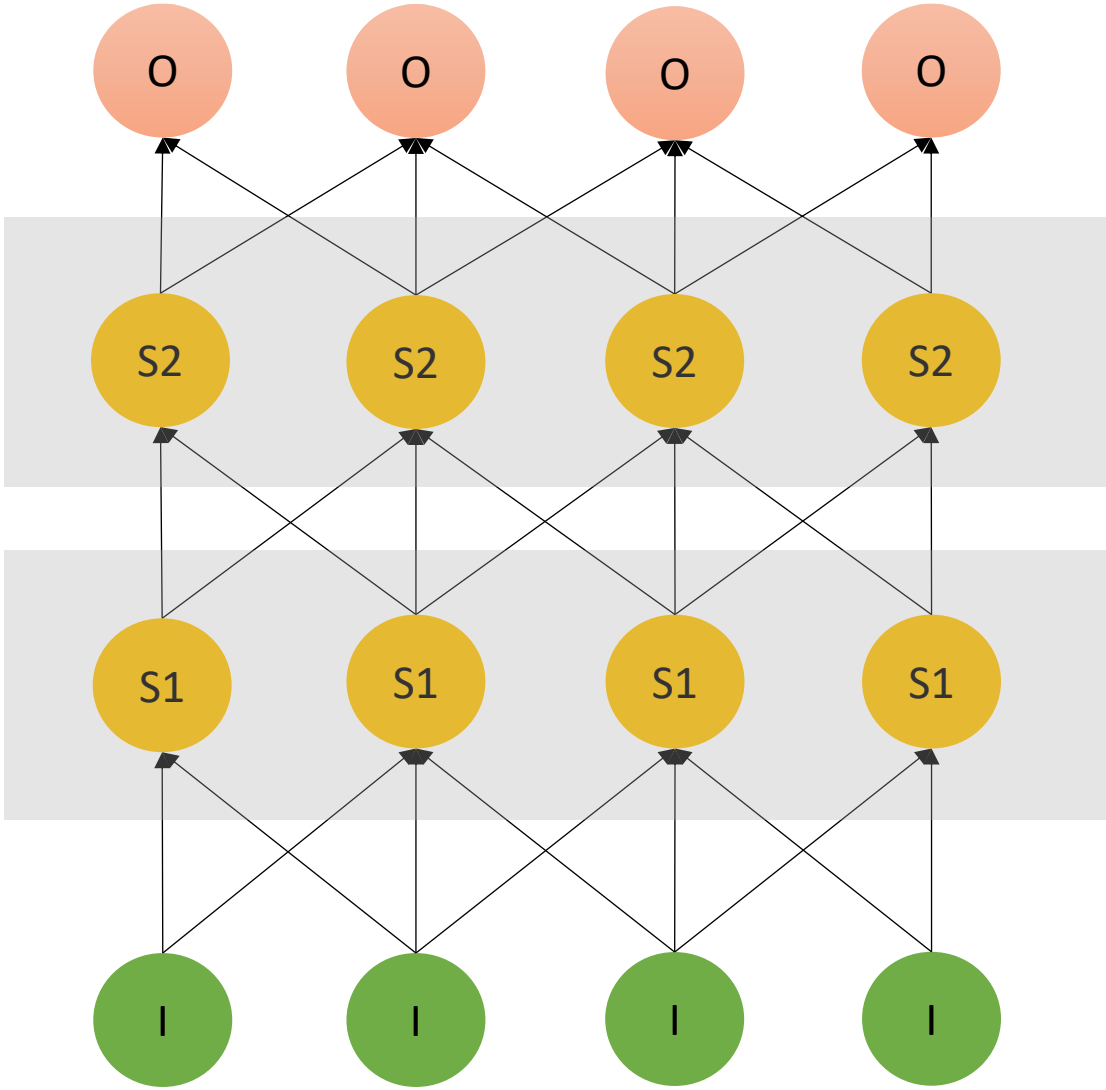
- Scheduler in Job Manager
- Exec of vertex
  - Version number
  - Exec record
    - State
    - Predecessor's version
    - Preference(optional)

Fault tolerance

- Vertex error → Job manger
- Process crashes → Job manager
- Daemon fail → Job manager receive timeout (heartbeat)
- Read Error on input channel → Mark the channel as failed

Stage manager

- Each vertex belongs to a stage
- Report statistics
- Get callback on interesting events
  - Gather execution statistics
  - Request duplicate executions



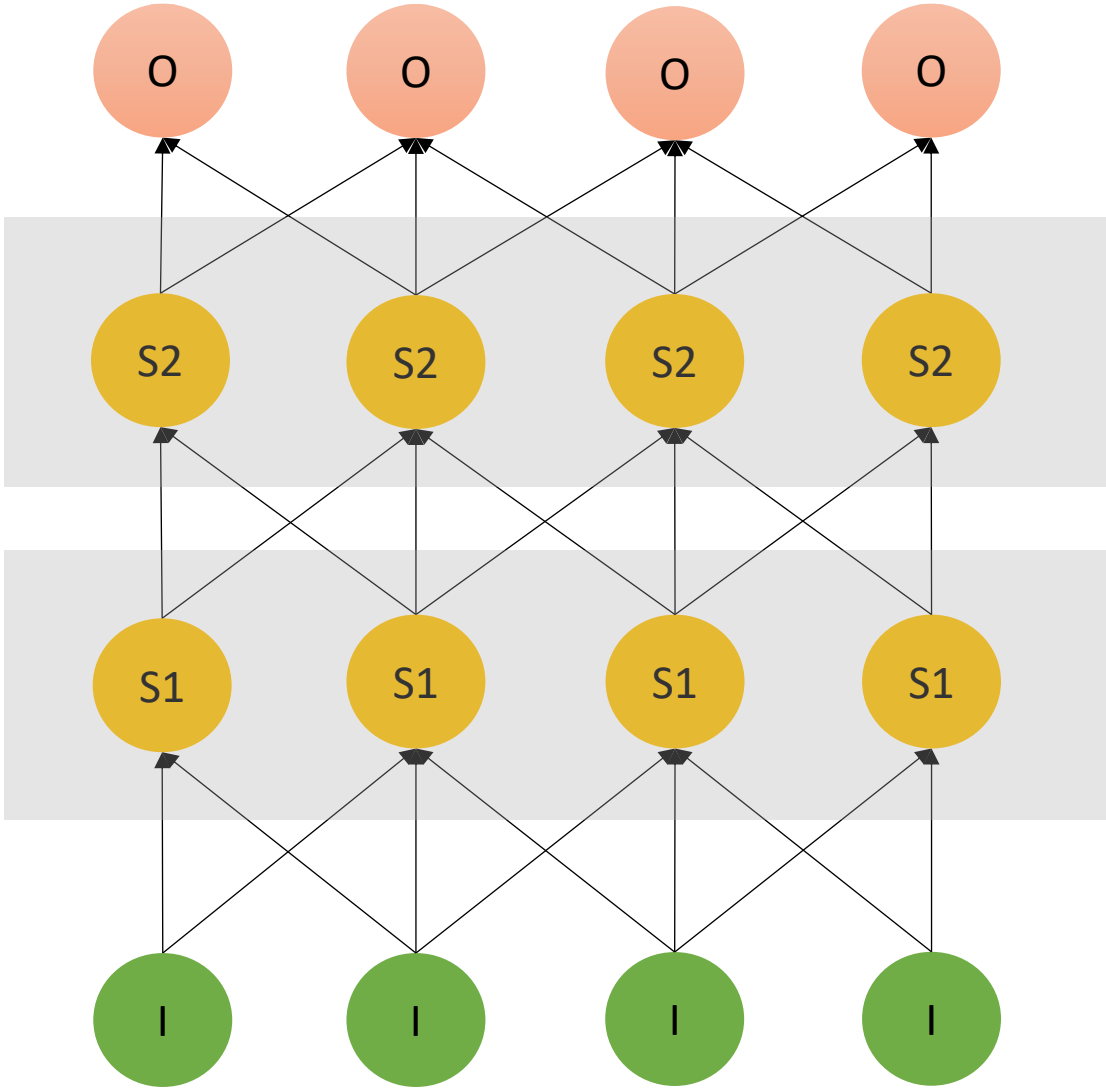
Connection Manager

- Any pair of stages can be linked

- Get callback on interesting events

  In upstream stage

  - For dynamic optimization



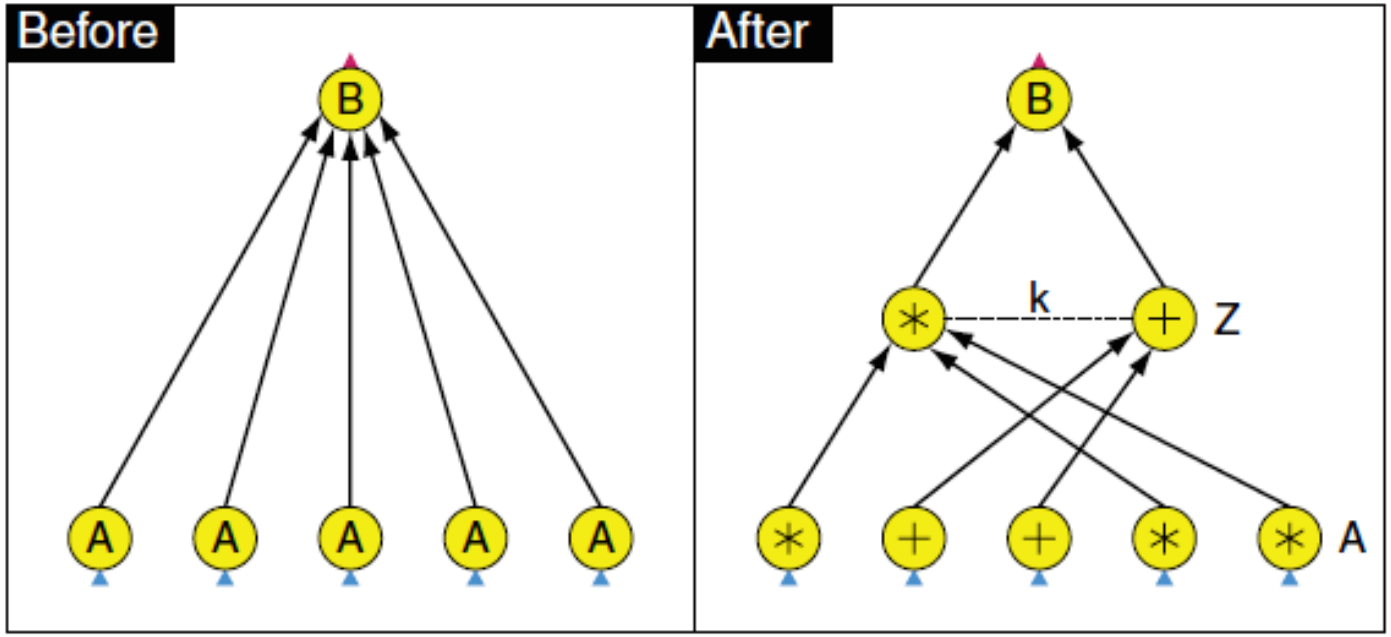
Aggregation Manager

- Graph Refinement

  Data reduction

  Insert new layer

  Reduce network traffic



Example

- SkyServer DB Query

  Table U (objId, color)

  Table N (objId,neightborId)

  Find neighbor star with similar colors

  Join U + N to find

  T = U.color, N.neighborID WHERE U.objId = N.objId

  Join U + T to find

  U.objID WHEREU.objId = T.objId AND U.color = T.color

Example

- SkyServer DB Query

  Table U (objId, color)

  Table N (objId,neightborId)

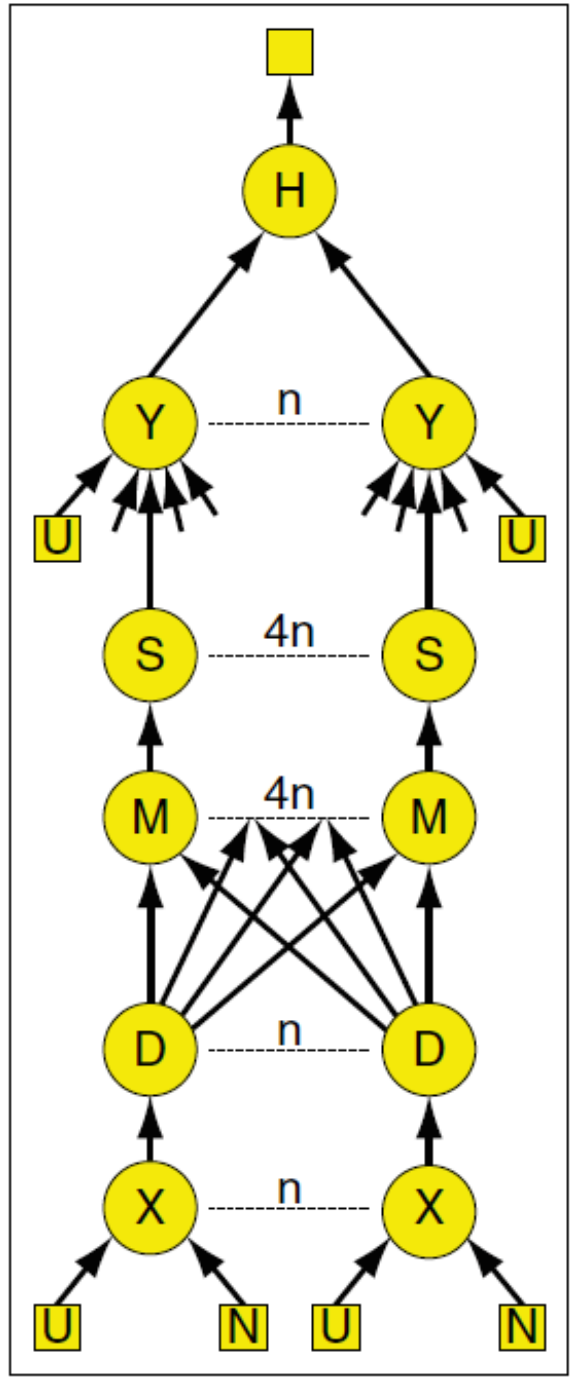
  Find neighbor star with similar colors

  Join U + N to find

  T = U.color, N.neighborID WHERE U.c

  Join U + T to find

  U.objID WHEREU.objId = T.objId AND



Example

Example started from (21:46):
https://www.youtube.com/watch?v=WPhE5JCP2Ak