Response-Time Analysis of a Soft Real-time NVIDIA Holoscan Application

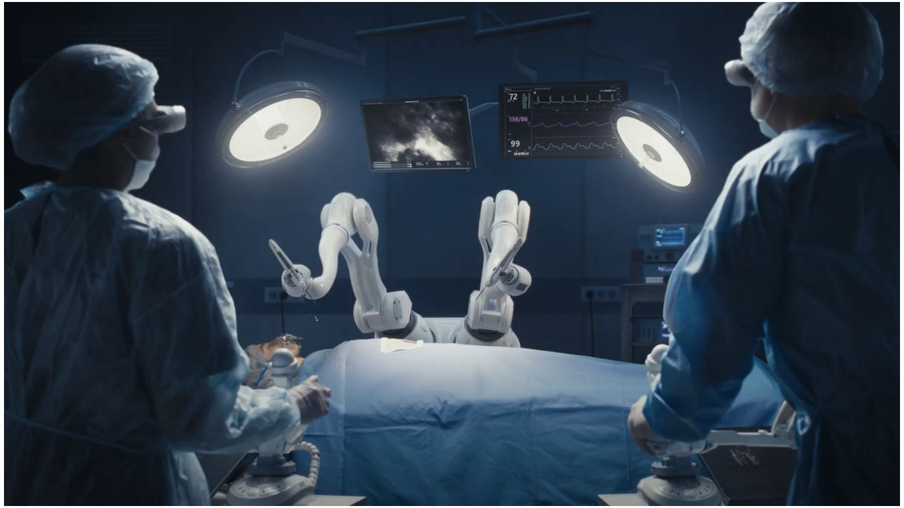
Philip Schowitz, Arpan Gujarati
University of British Columbia
Soham Sinha
NVIDIA

THE UNIVERSITY OF BRITISH COLUMBIA

NVIDIA

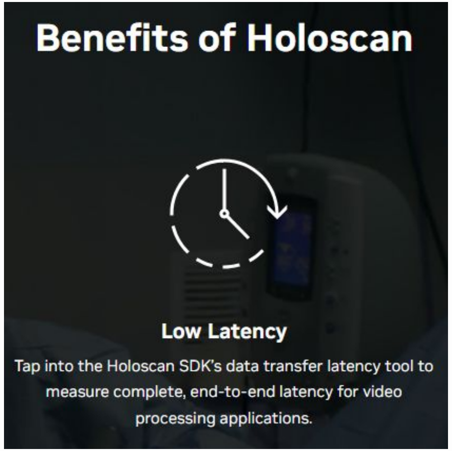
Edge Computing

- AI is fueling resource-intensive applications on the edge
- Embedded platforms become more complex
    - Harder to develop apps



Frameworks and Limitations

- Development frameworks built with latency in mind
- **NVIDIA Holoscan** promises low latency SDK for medical devices
- But what about guarantees?
  - Holoscan relies on profiling...



What's wrong with profiling?

- Profiling to learn timing properties has many issues
  - **The response time bound may be unsafe**
  - **Application development must be finished**
  - **Profiling can be costly in time and compute**

Research Question:

Can we develop a response time bound for any Holoscan application, given information about it?

2

Holoscan

Holoscan Basics

- Holoscan apps are made up of ***operators***
  - Blocks of code that run on CPU threads
  - Can call on the GPU
- Operators scheduled by an executor

Operator

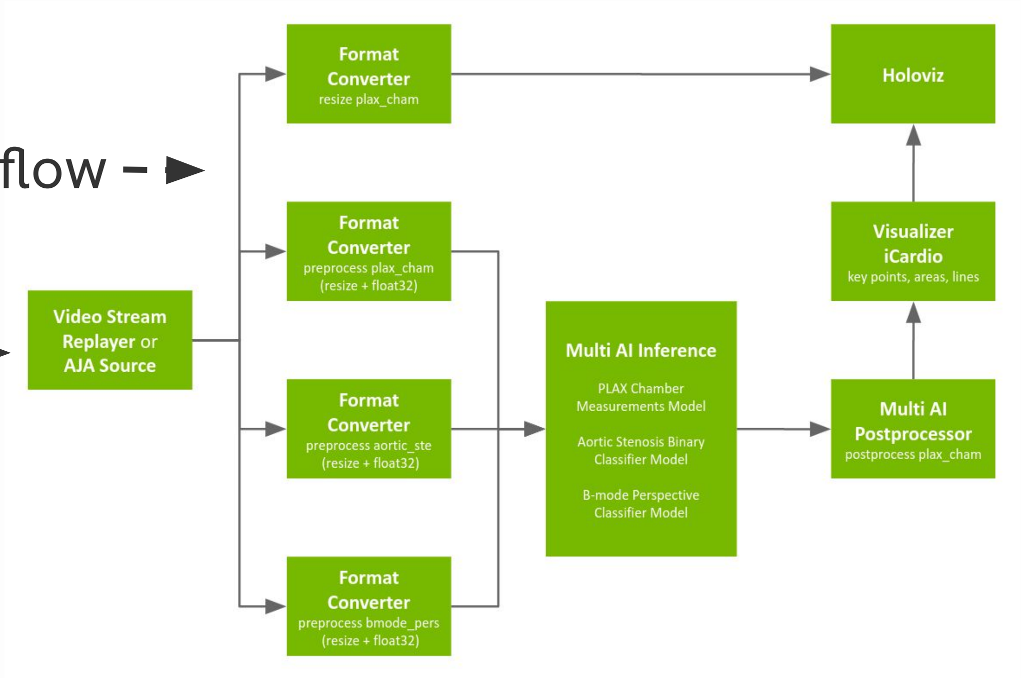
(Assume execution time known)

```cpp
void FormatConverterOp::compute(InputContext& op_input, OutputContext& op_output,
                                ExecutionContext& context) {

  // Process input message
  auto in_message = op_input.receive<gxf::Entity>("source_video").value();

  // get the CUDA stream from the input message
  gxf_result_t stream_handler_result =
      cuda_stream_handler_.from_message(context.context(), in_message);
  if (stream_handler_result != GXF_SUCCESS) {
    throw std::runtime_error("Failed to get the CUDA stream from incoming messages");
  }

  // assign the CUDA stream to the NPP stream context
  npp_stream_ctx_.hStream = cuda_stream_handler_.get_cuda_stream(context.context());

  nvidia::gxf::Shape out_shape{0, 0, 0};
  void* in_tensor_data = nullptr;
  nvidia::gxf::PrimitiveType in_primitive_type = nvidia::gxf::PrimitiveType::kCustom;
  nvidia::gxf::MemoryStorageType in_memory_storage_type = nvidia::gxf::MemoryStorageType::kHost;
  int32_t rows = 0;
  int32_t columns = 0;
  int16_t in_channels = 0;
  int16_t out_channels = 0;
  std::vector<nvidia::gxf::ColorPlane> in_color_planes;

  // get Handle to underlying nvidia::gxf::Allocator from std::shared_ptr<holoscan::Allocator>
  auto pool =
      nvidia::gxf::Handle<nvidia::gxf::Allocator>::Create(context.context(), pool_->gxf_cid());

  // Get either the Tensor or VideoBuffer attached to the message
  bool is_video_buffer;
  nvidia::gxf::Handle<nvidia::gxf::VideoBuffer> video_buffer;
  try {
    video_buffer = holoscan::gxf::get_videobuffer(in_message);
    is_video_buffer = true;
  } catch (const std::runtime_error& r_) {
    HOLOSCAN_LOG_TRACE("Failed to read VideoBuffer with error: {}", std::string(r_.what()));
    is_video_buffer = false;
  }

  if (is_video_buffer) {
    // Convert VideoBuffer to Tensor
    auto frame = video_buffer.get();
```

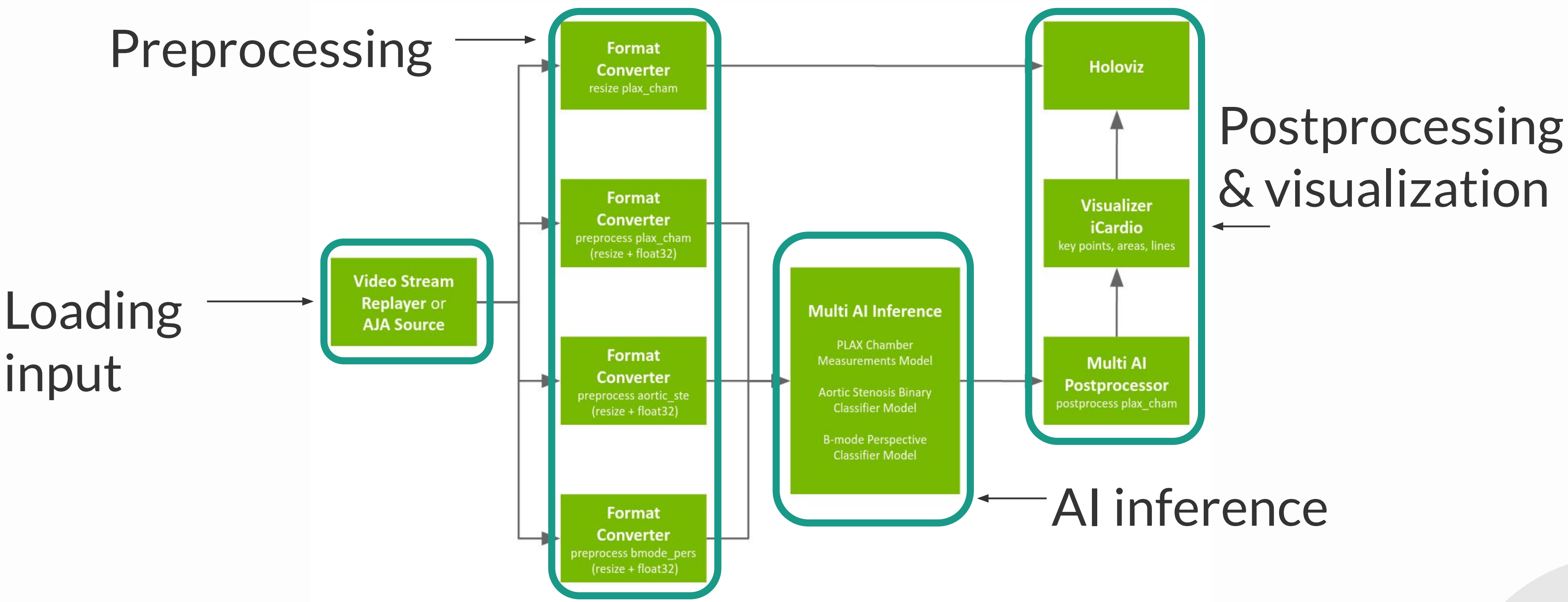
What do Holoscan Applications Look Like?

Data flow –►

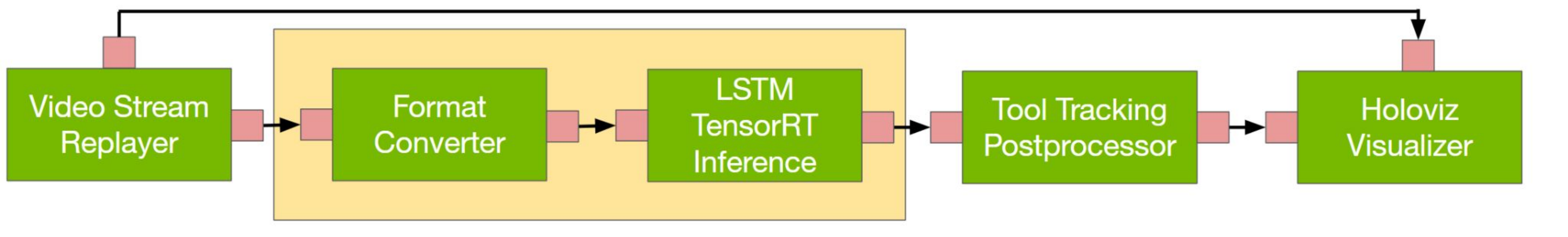
Operator –►



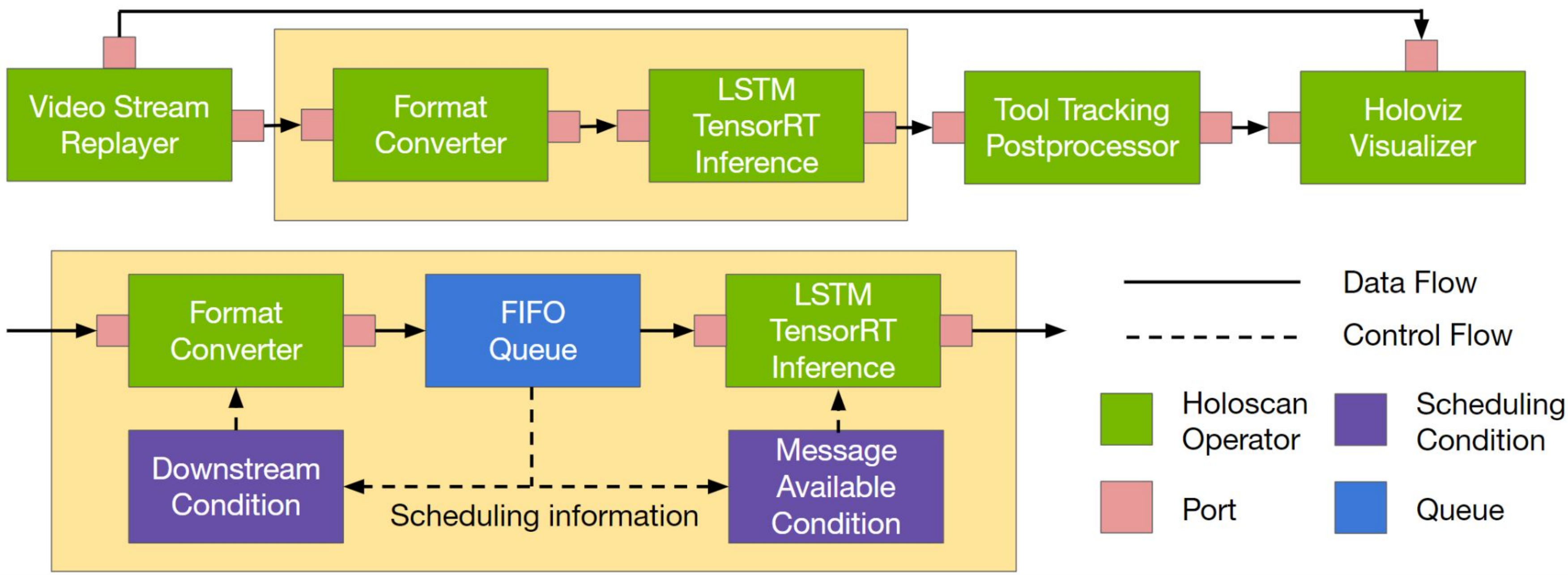
What do Holoscan Applications Look Like?



Holoscan Internals



Holoscan Internals

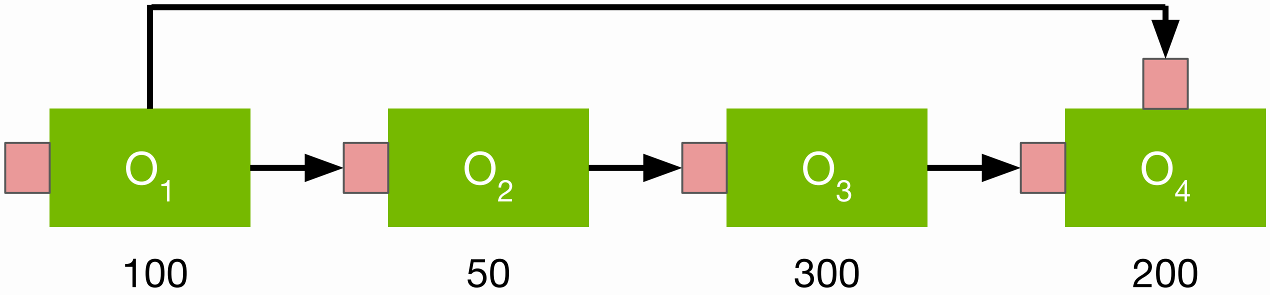


Downstream Examples

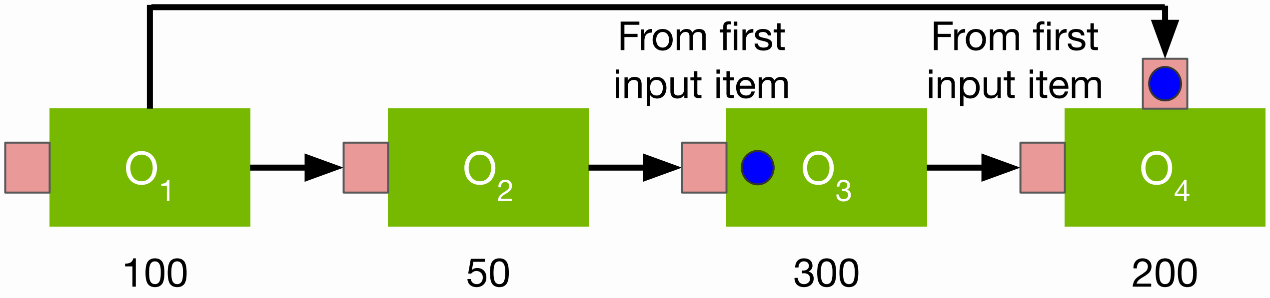
Example 1

Why do we need the downstream condition?

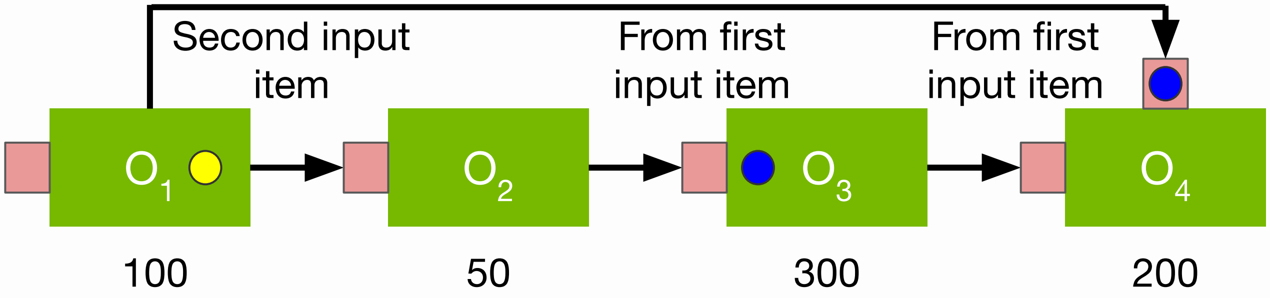
Why Downstream?



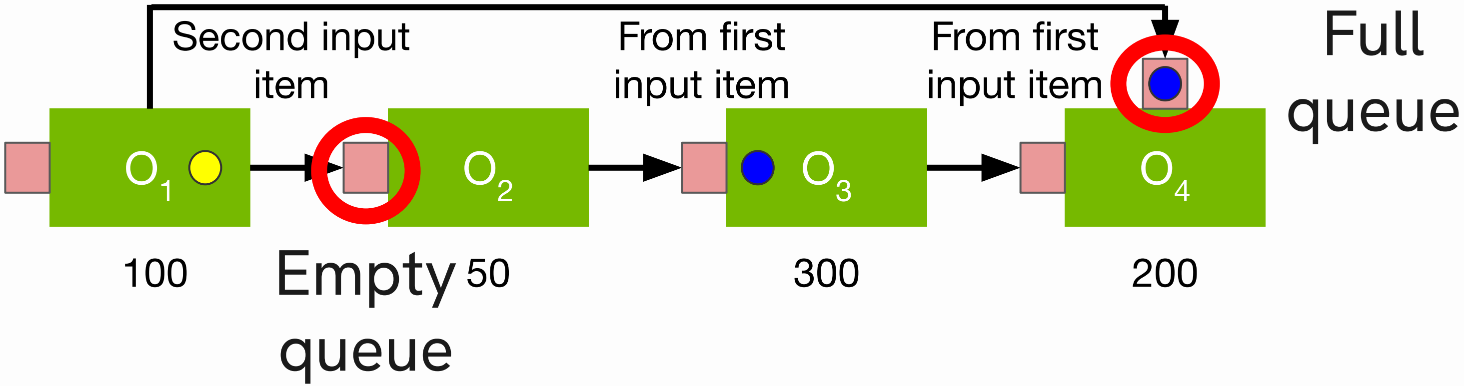
Why Downstream?



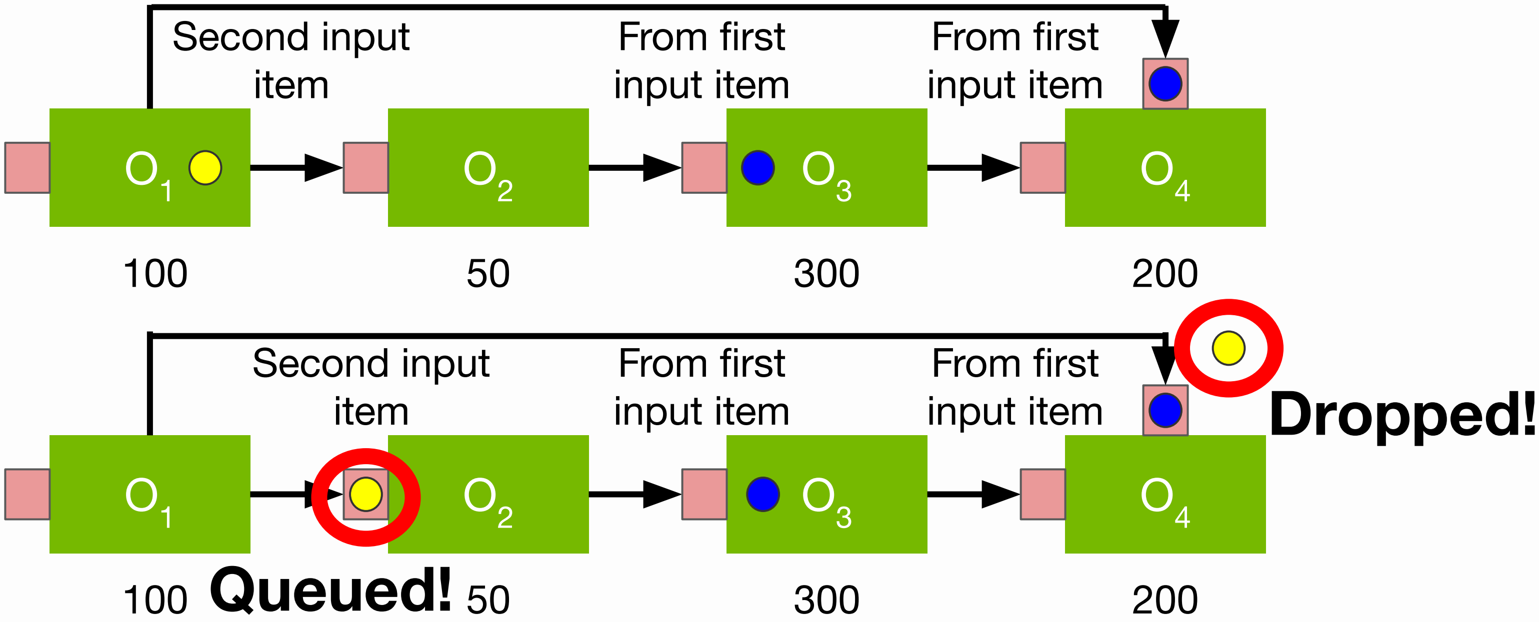
Why Downstream?



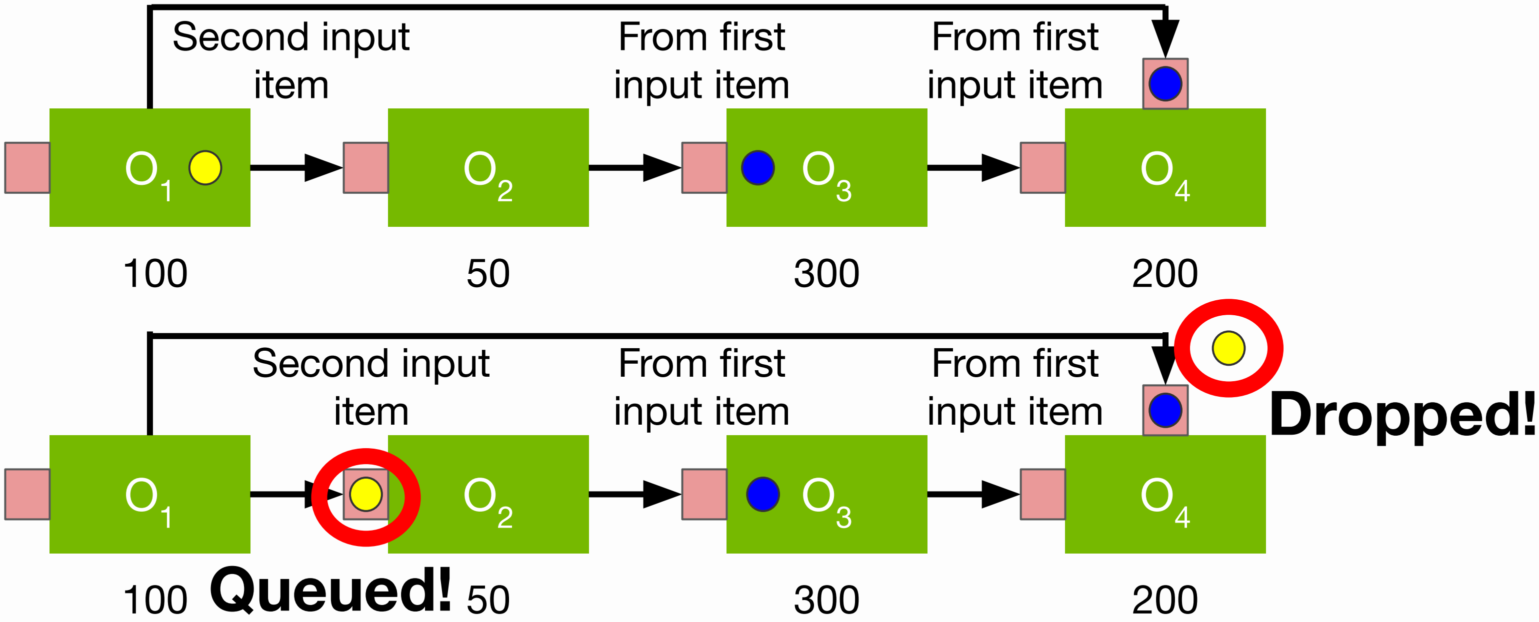
Why Downstream?



Why Downstream?



Why Downstream?



Takeaway: Violates correctness condition

Downstream Examples

Example 1

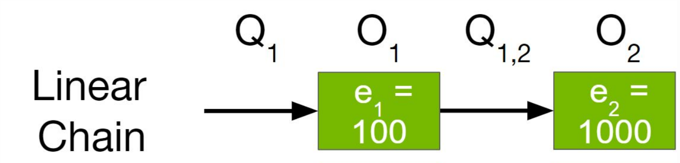
Why do we need the downstream condition?

Example 2

How does downstream affect response times?

Downstream and Response Times

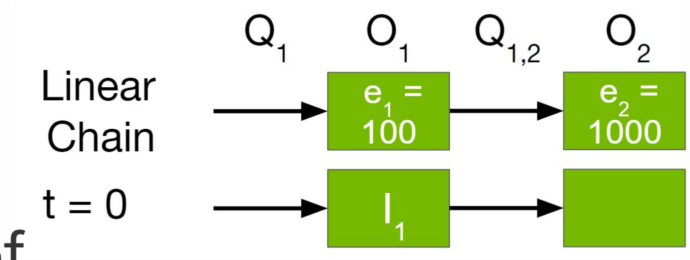
Let's assume...



- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads

Downstream and Response Times

Let's assume...

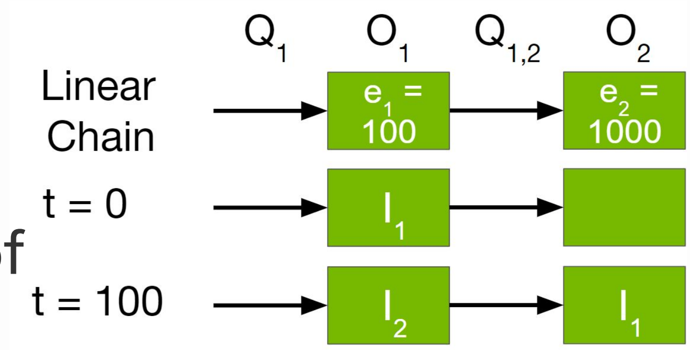


- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads

Downstream and Response Times

Let's assume...

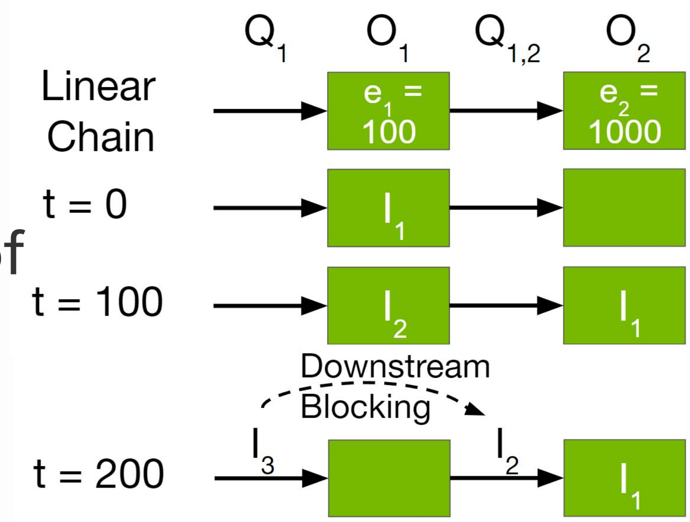
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

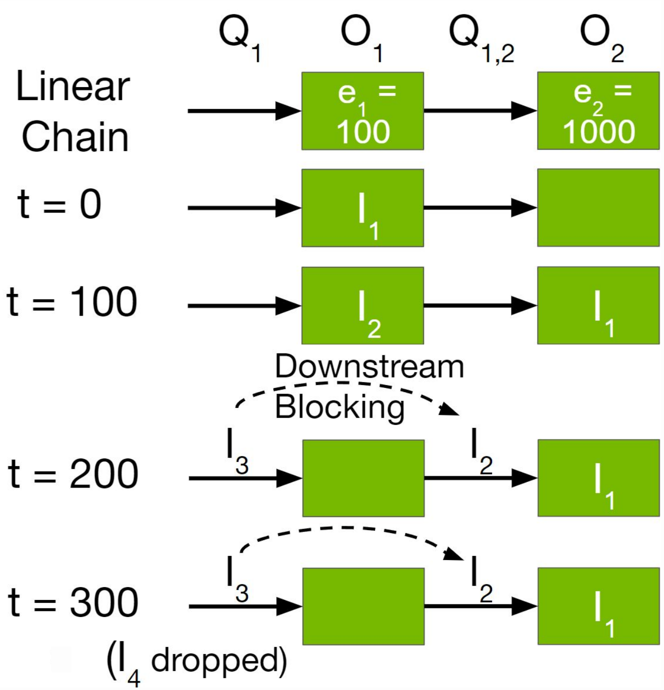
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

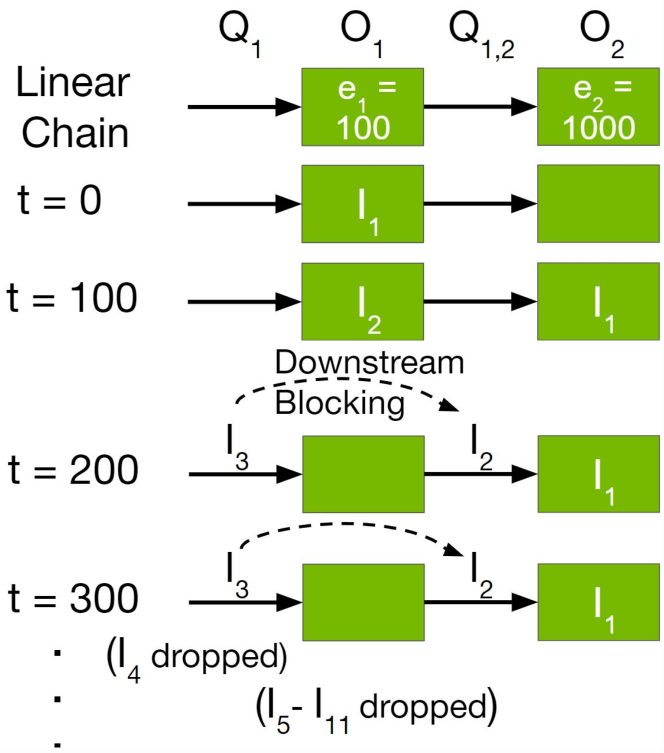
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

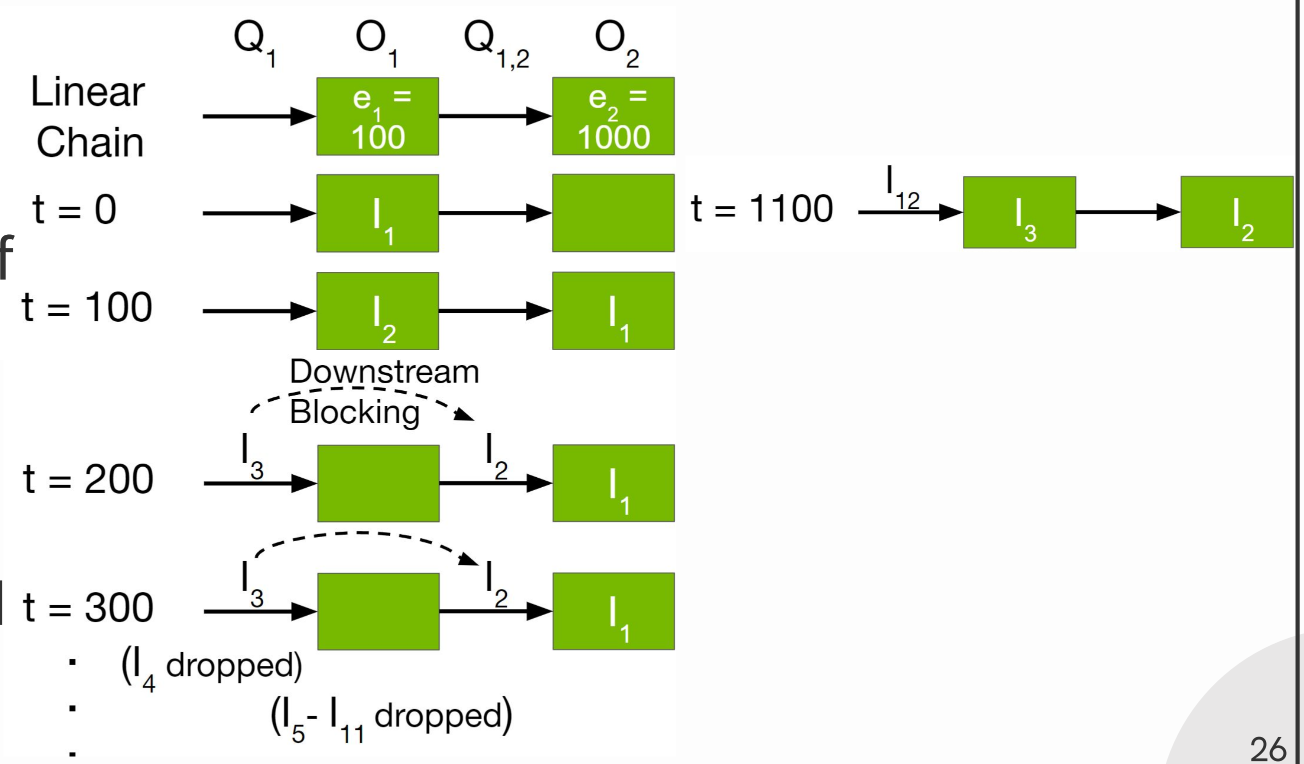
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

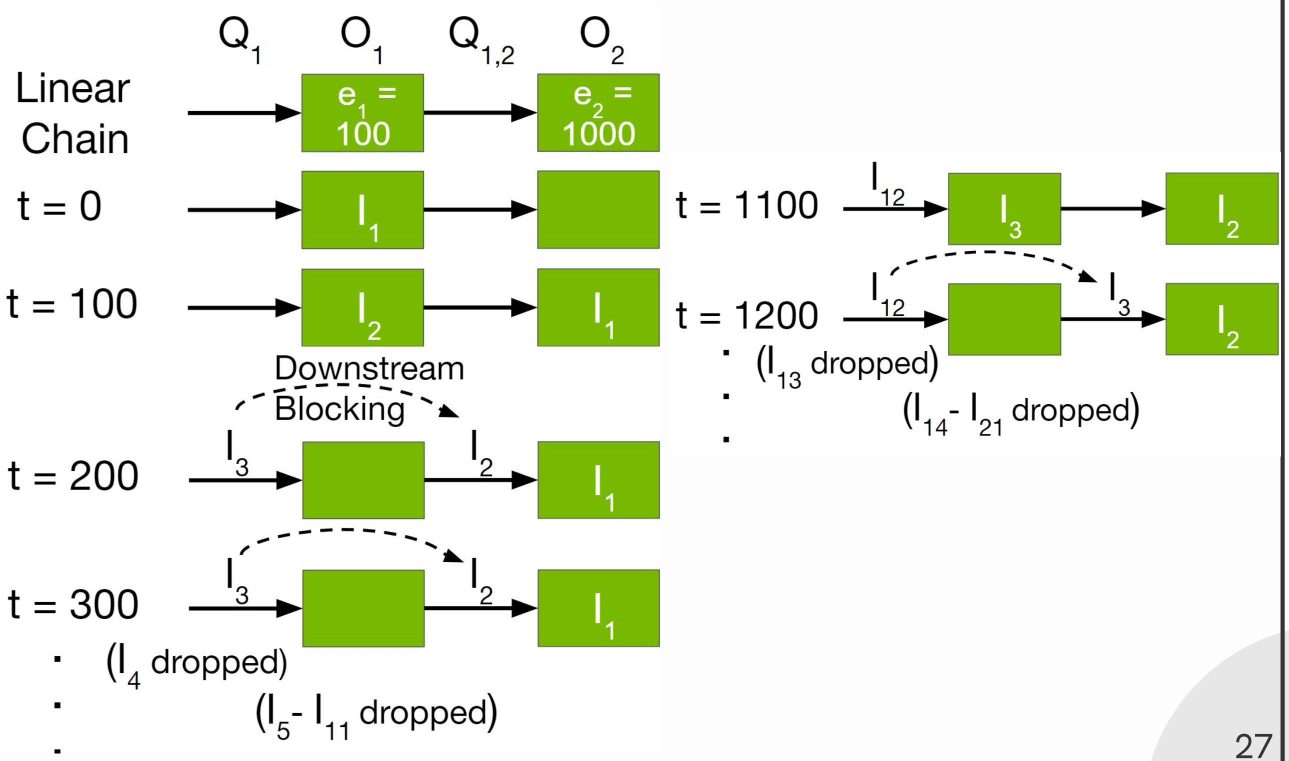
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

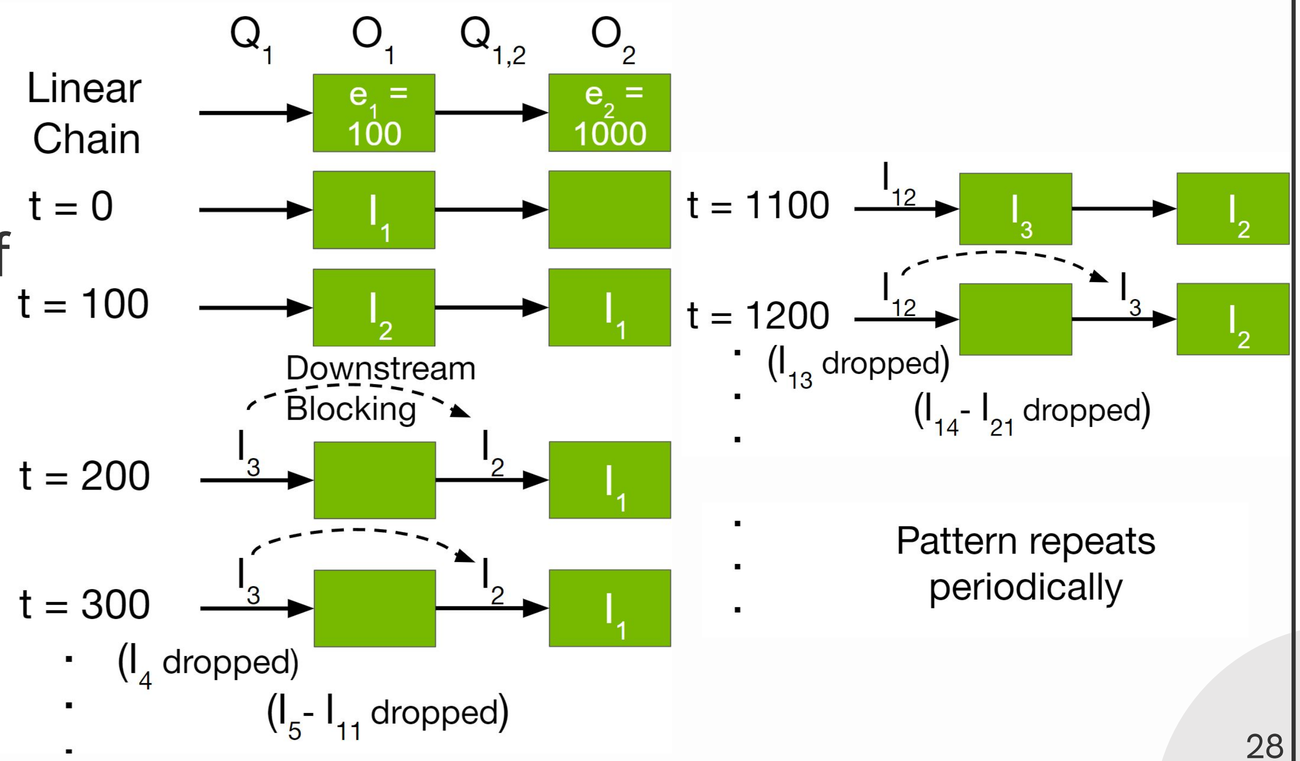
- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream and Response Times

Let's assume...

- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads

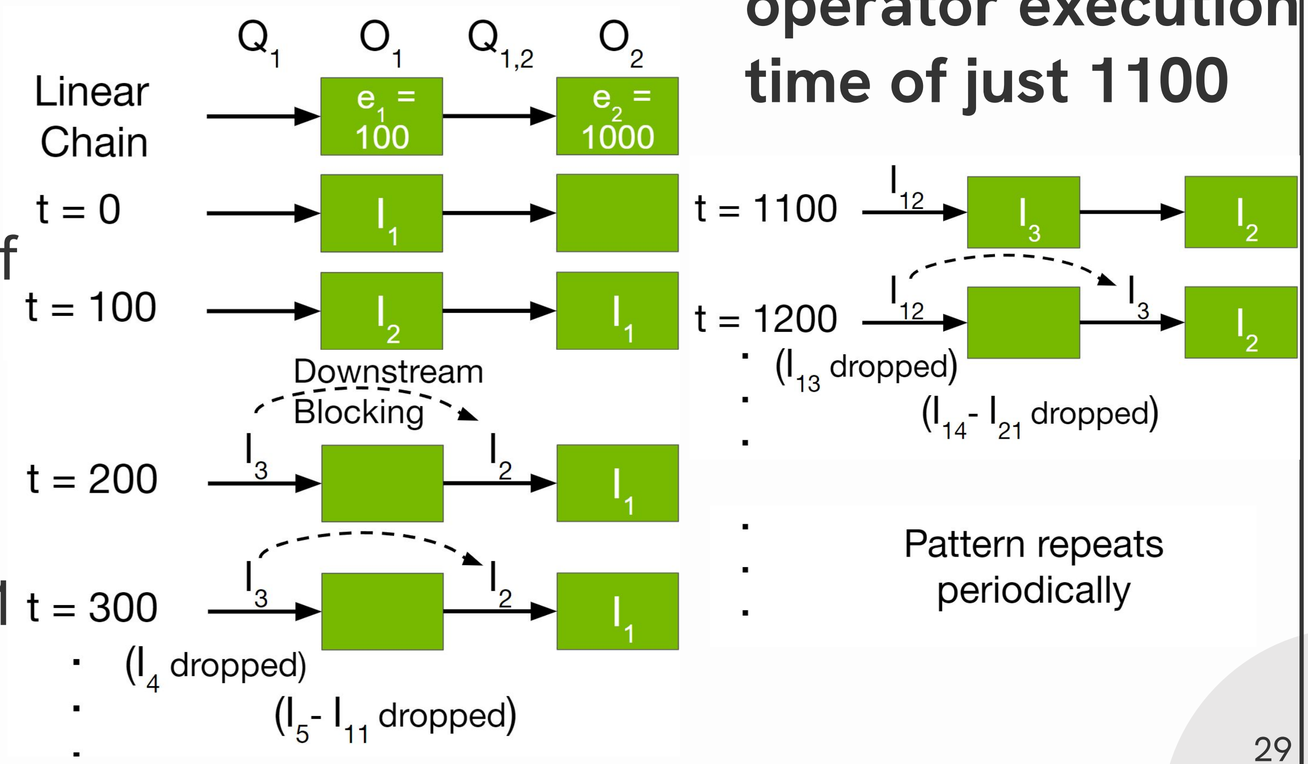


Downstream and Response Times

Takeaway: WCRT of 2900 with total operator execution time of just 1100

Let's assume...

- Linear chain of length 2
- Period = 100
- Queue size = 1
- No overheads



Downstream Examples

Example 1

Why do we need the downstream condition?

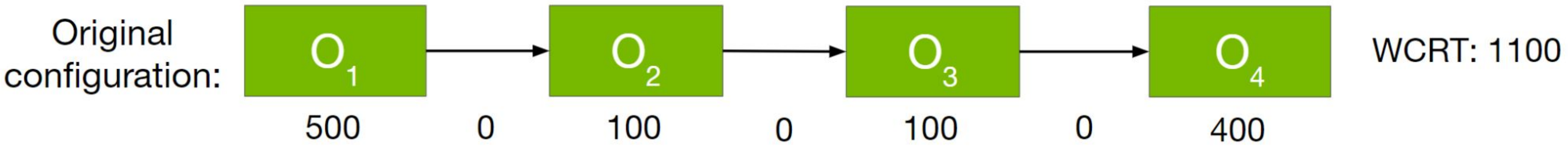
Example 2

How does downstream affect response times?

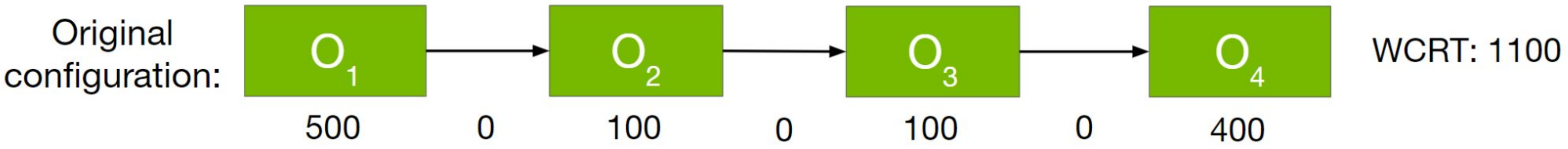
Example 3

How downstream can cause timing anomalies

Downstream Blocking Causes Timing Anomalies

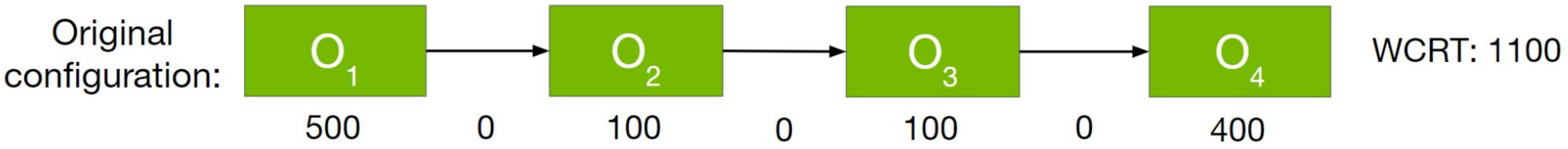


Downstream Blocking Causes Timing Anomalies

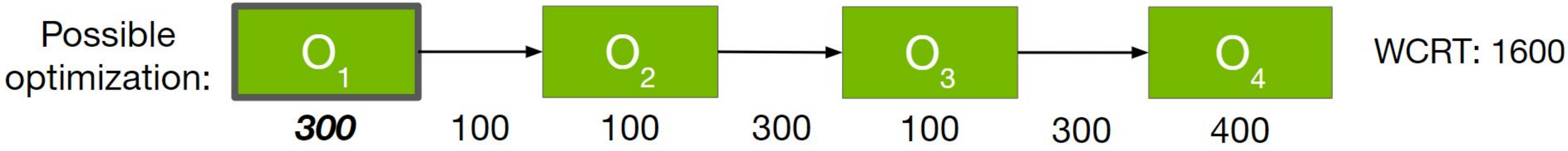


Work hard optimizing O_1 to lower execution time…

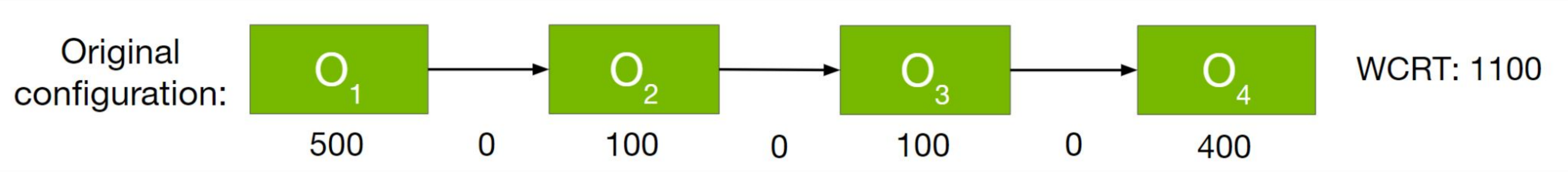
Downstream Blocking Causes Timing Anomalies



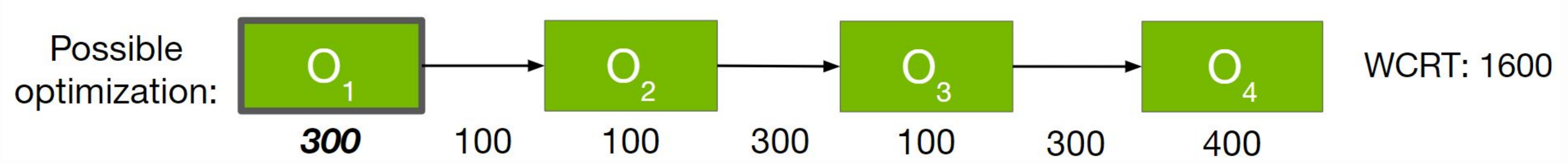
Work hard optimizing O_1 to lower execution time...



Downstream Blocking Causes Timing Anomalies



Work hard optimizing O_1 to lower execution time...



...but we encounter a timing anomaly!

3

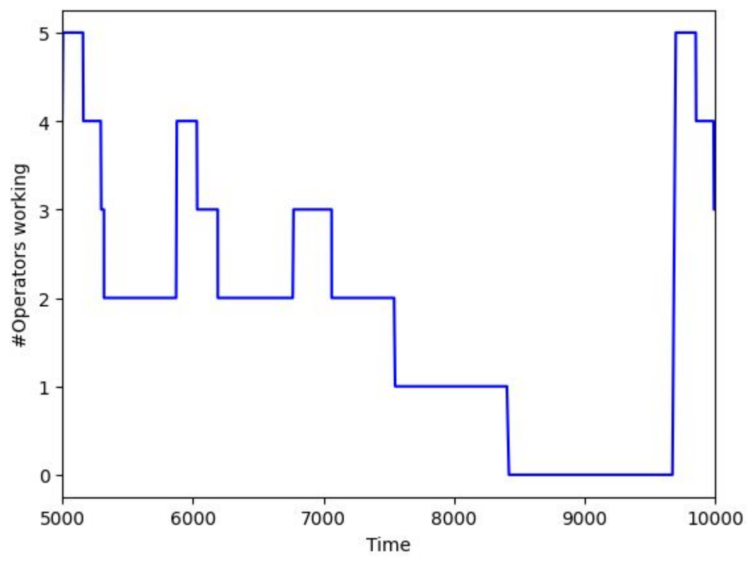
Response-Time Analysis

Approaches to RTA

- Many DAG response-time analyses already exist
  - Why not employ them?
- Consider a period T and relative deadline D...
  - Analyses commonly assume $T \geq D$
- But Holoscan wants to leverage parallelism
  - No hard deadline, maximizing throughput (T is small)

Leverage Parallelism

- Can process the first, second, third inputs simultaneously
- Holoscan geared to pipelined execution
    - Multiple jobs in same DAG



Operator parallelism over time

RTA Strategy

1. Response time bound for a linear chain
2. Why chain analysis insufficient for DAGs
3. Generalize response time bound for any arbitrary DAG

Assumptions

Ours is the first timing analysis of Holoscan

- Queue size = 1
  - Holoscan default
- All operators can run in parallel
  - NVIDIA embedded platforms have enough cores to do this
- Inputs arrive with a period as low as 0
- Operator execution time fixed throughout entire run

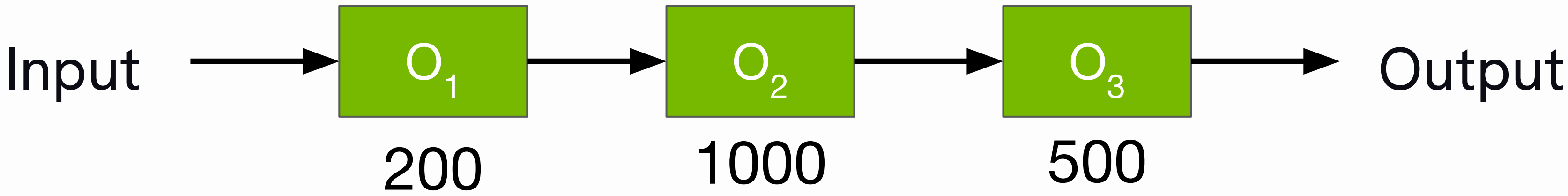
Assumptions

Ours is the first timing analysis of Holoscan

These match how the system is used

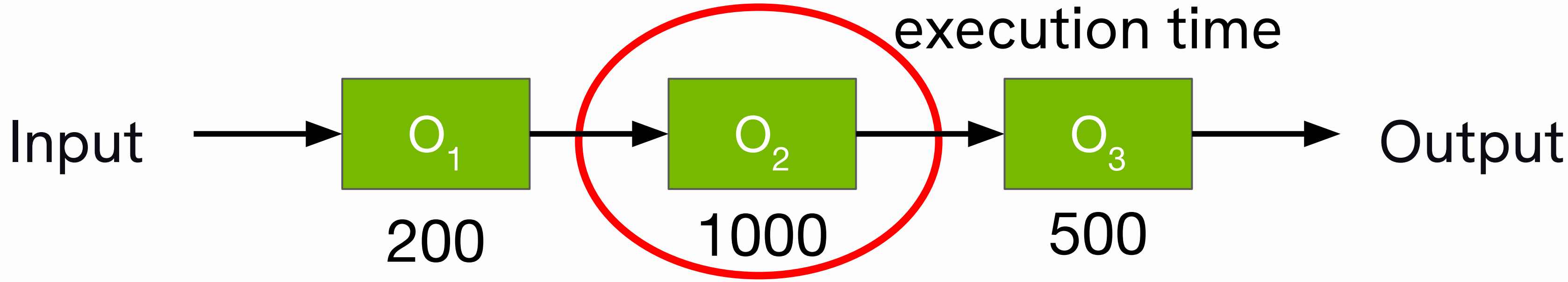
- Queue size = 1
  - Holoscan default
- All operators can run in parallel
  - NVIDIA embedded platforms have enough cores to do this
- Inputs arrive with a period as low as 0
- Operator execution time fixed throughout entire run

Linear Chain Response Time Bound



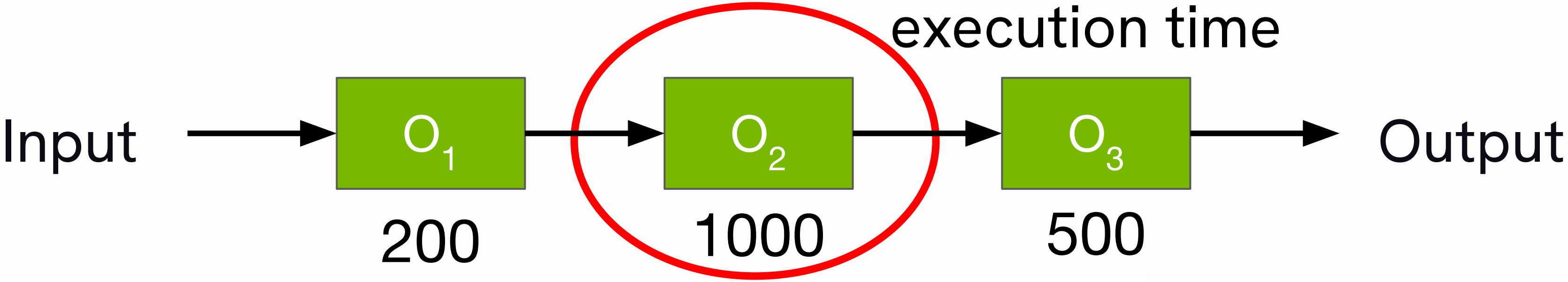
Linear Chain Response Time Bound

Key idea: Bottleneck is operator with greatest execution time



Linear Chain Response Time Bound

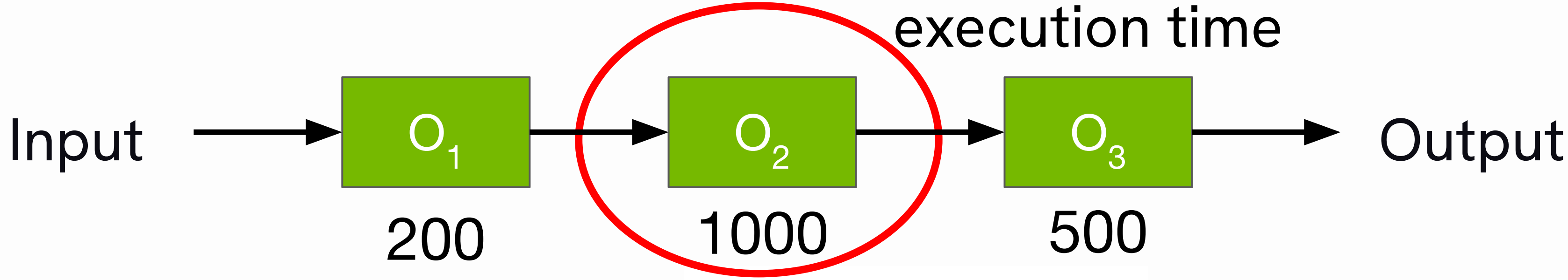
Key idea: Bottleneck is operator with greatest execution time



Upper bound: $$e_b^{ub} \cdot b + \sum_{i=b+1}^{n} e_i^{ub}$$

Linear Chain Response Time Bound

Key idea: Bottleneck is operator with greatest execution time



Upper bound: $\underbrace{e_b^{ub} \cdot b}_{} + \sum_{i=b+1}^{n} e_i^{ub}$

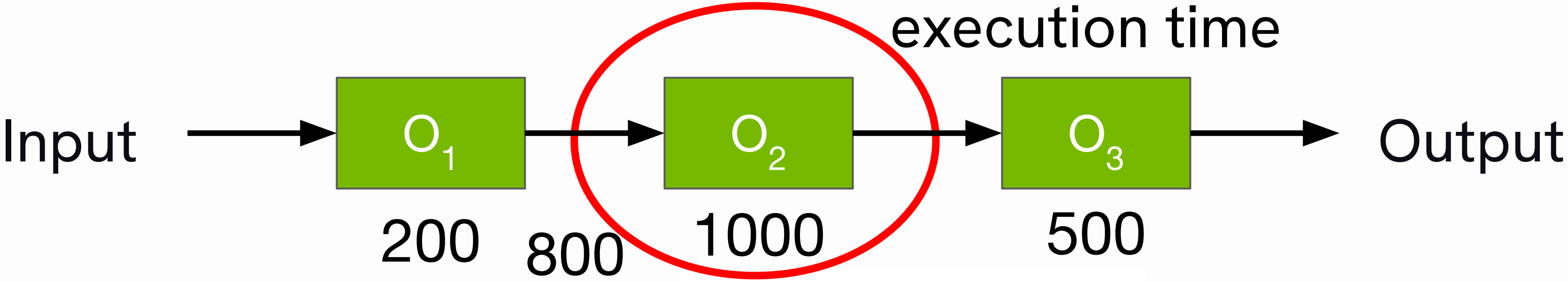
Bottleneck worst-case execution time multiplied by its index

Sum of worst-case execution times of operators following bottleneck

Linear Chain Response Time Bound

2*1000 + 500 = 2500

Key idea: Bottleneck is operator with greatest execution time

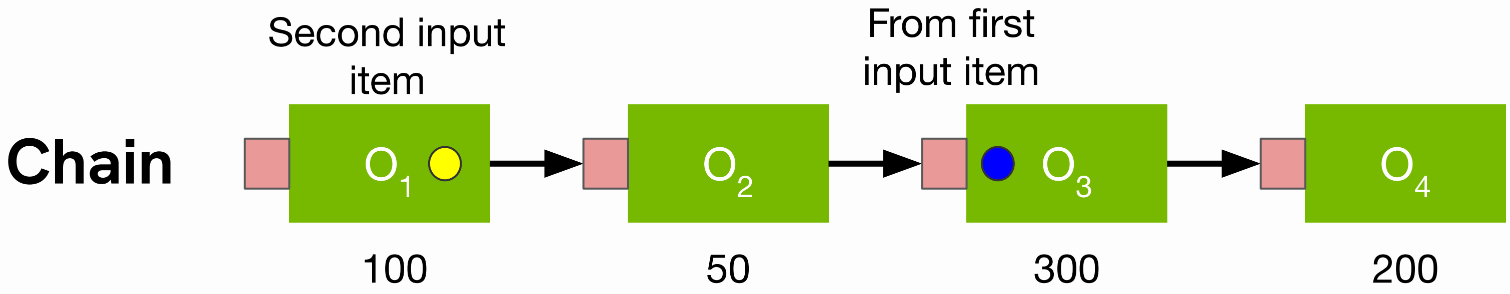


Upper bound: $$e_b^{ub} \cdot b + \sum_{i=b+1}^{n} e_i^{ub}$$

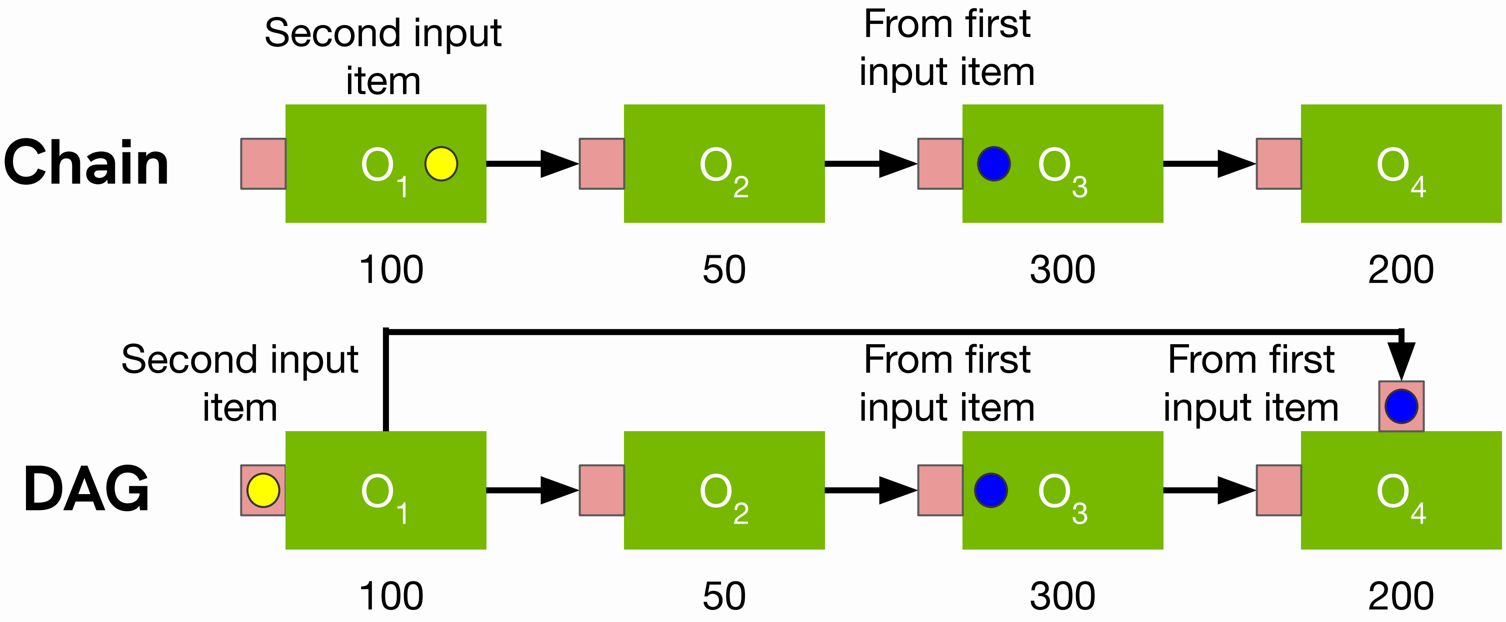
Bottleneck worst-case execution time multiplied by its index

Sum of worst-case execution times of operators following bottleneck

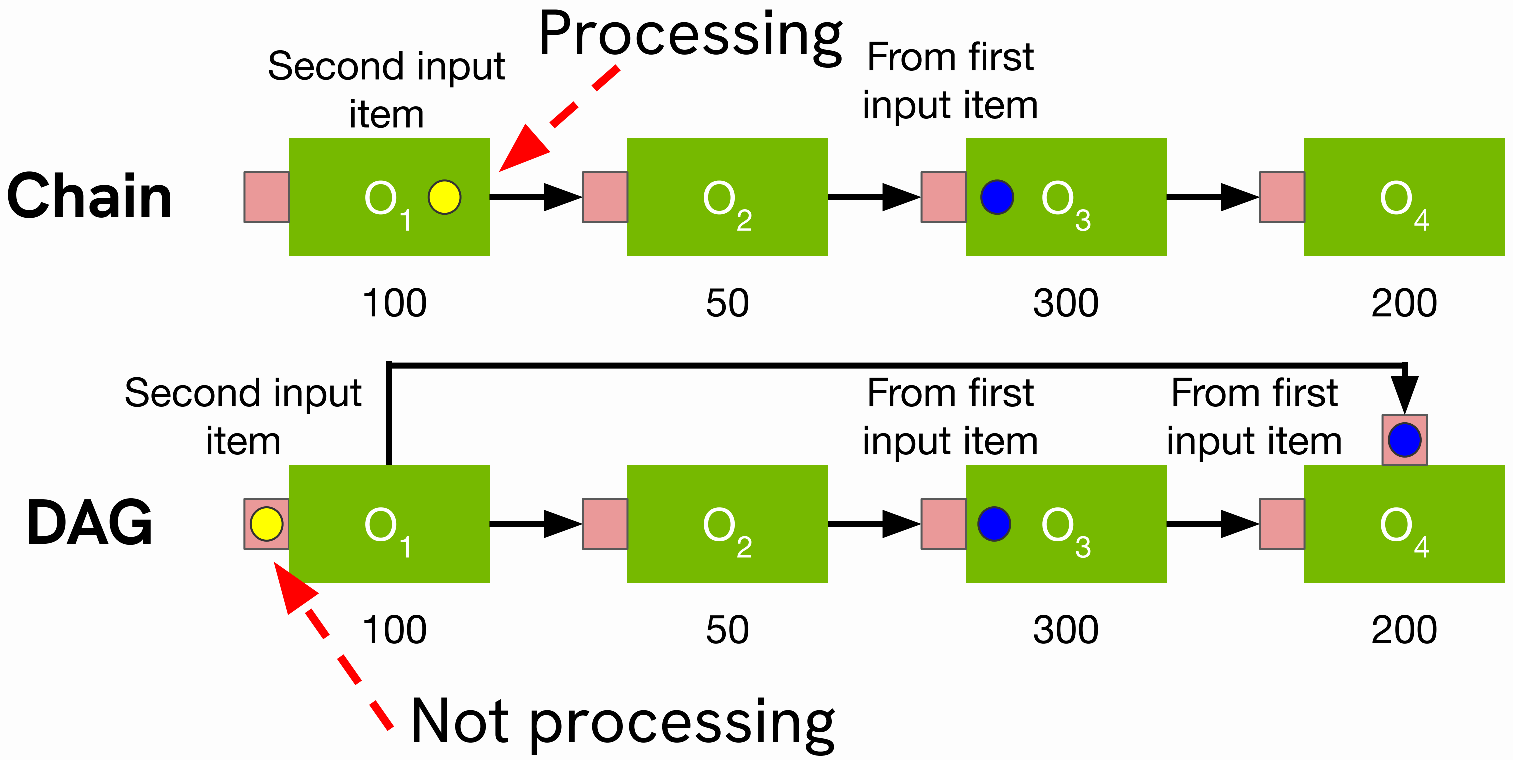
Chains vs DAGs



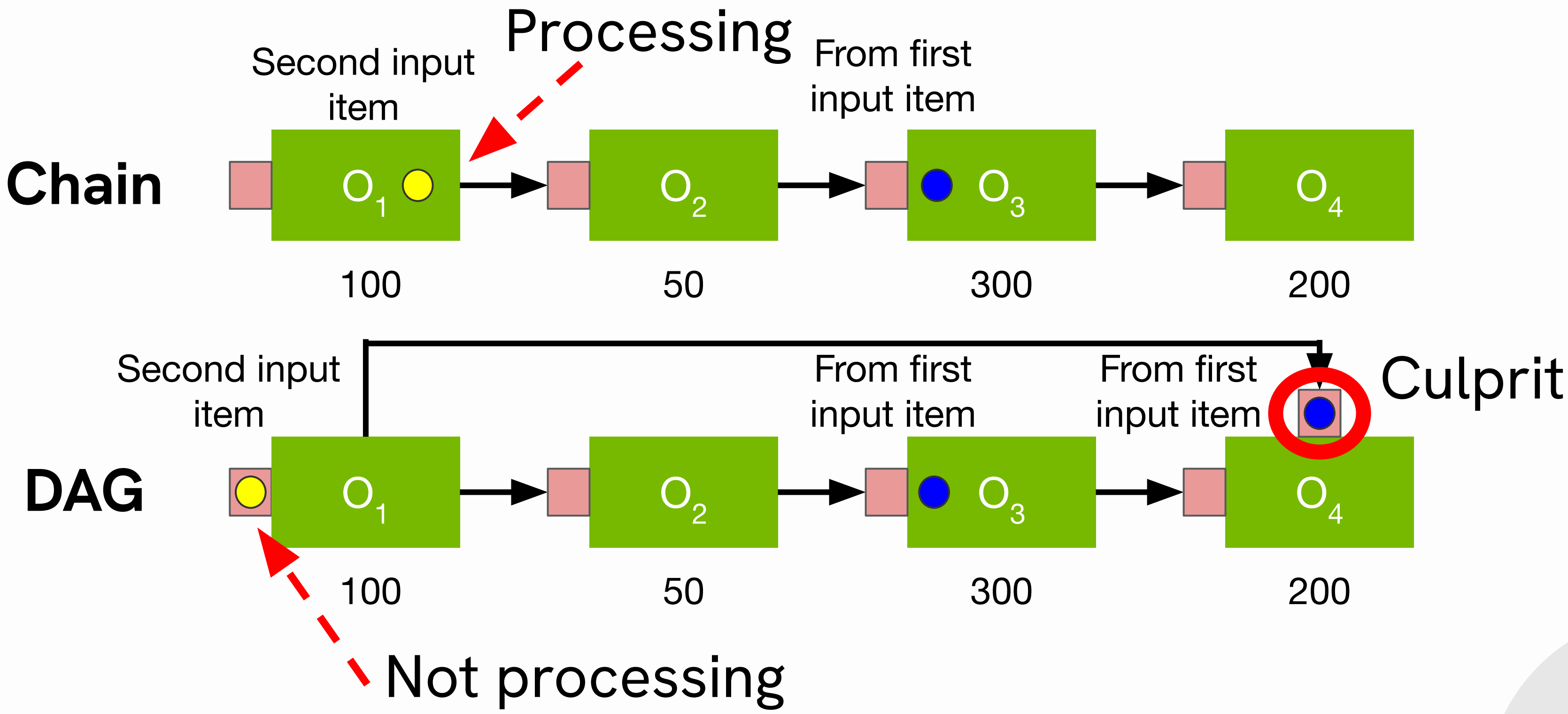
Chains vs DAGs



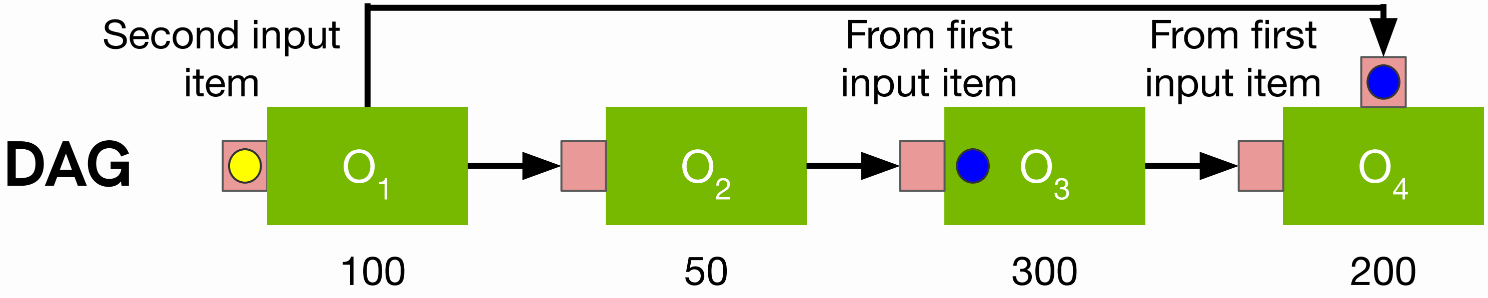
Chains vs DAGs



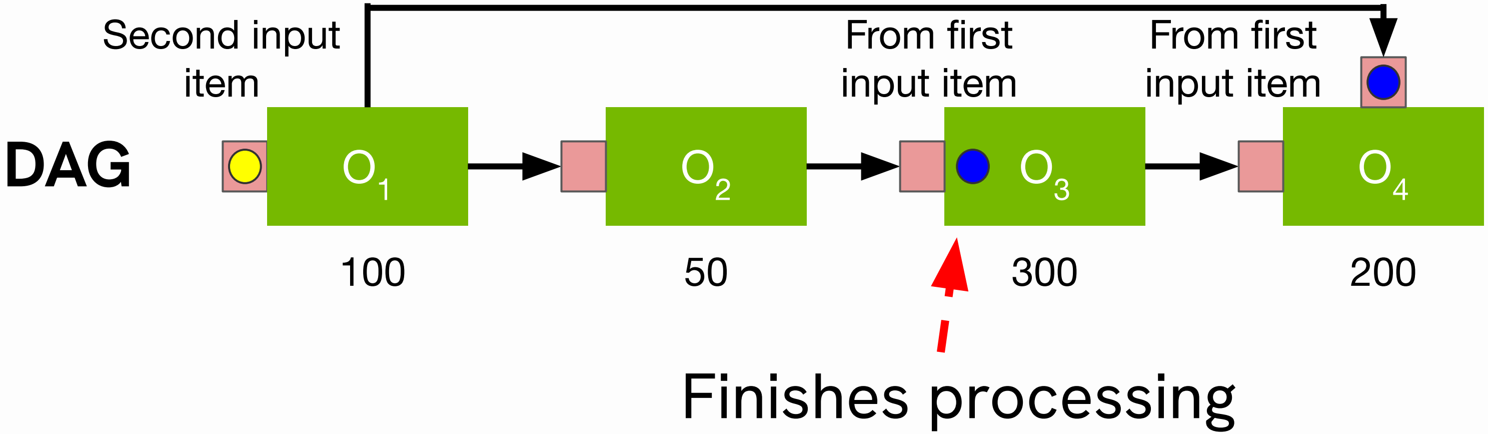
Chains vs DAGs



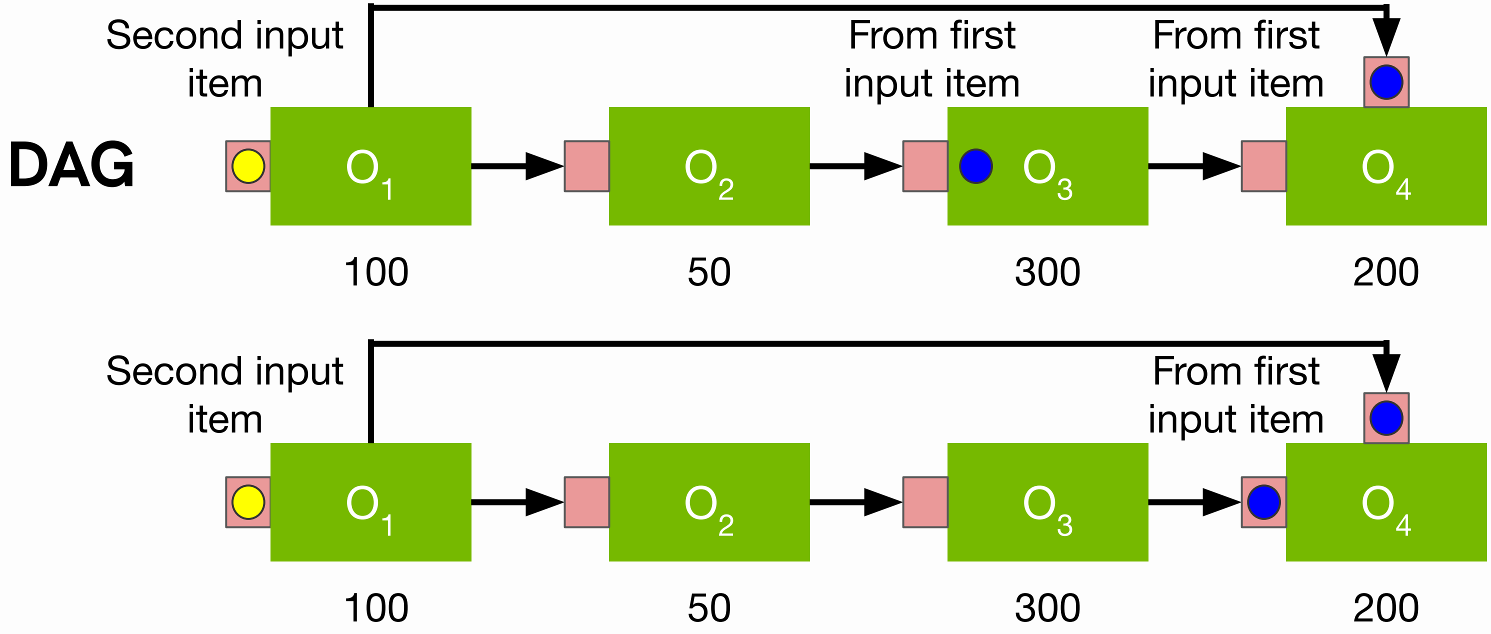
Chains vs DAGs



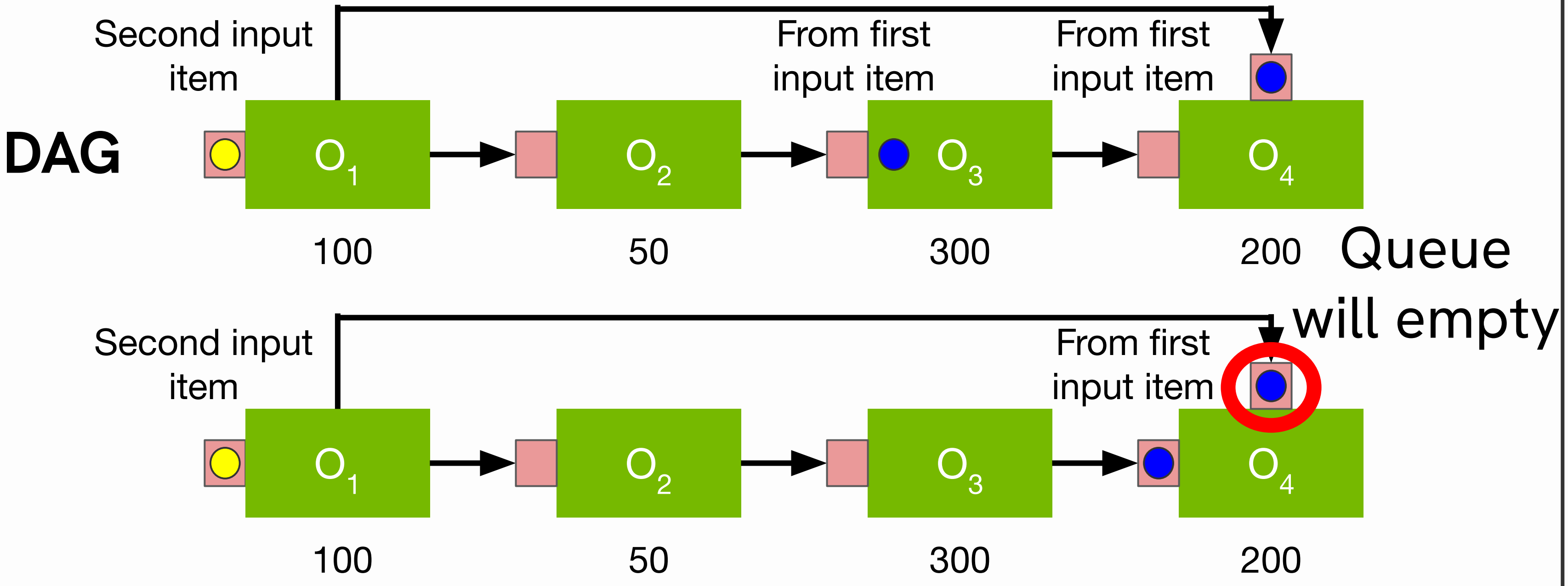
Chains vs DAGs



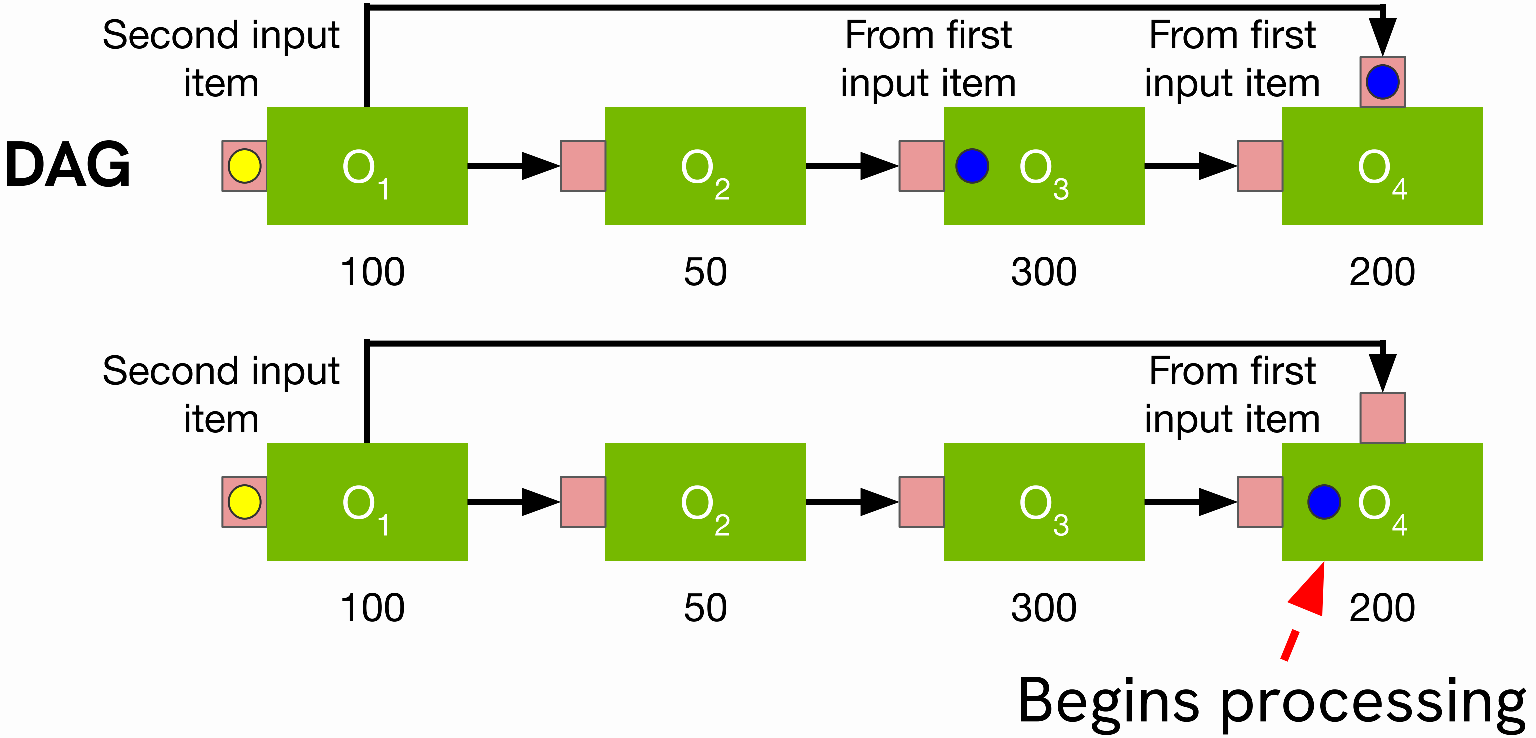
Chains vs DAGs



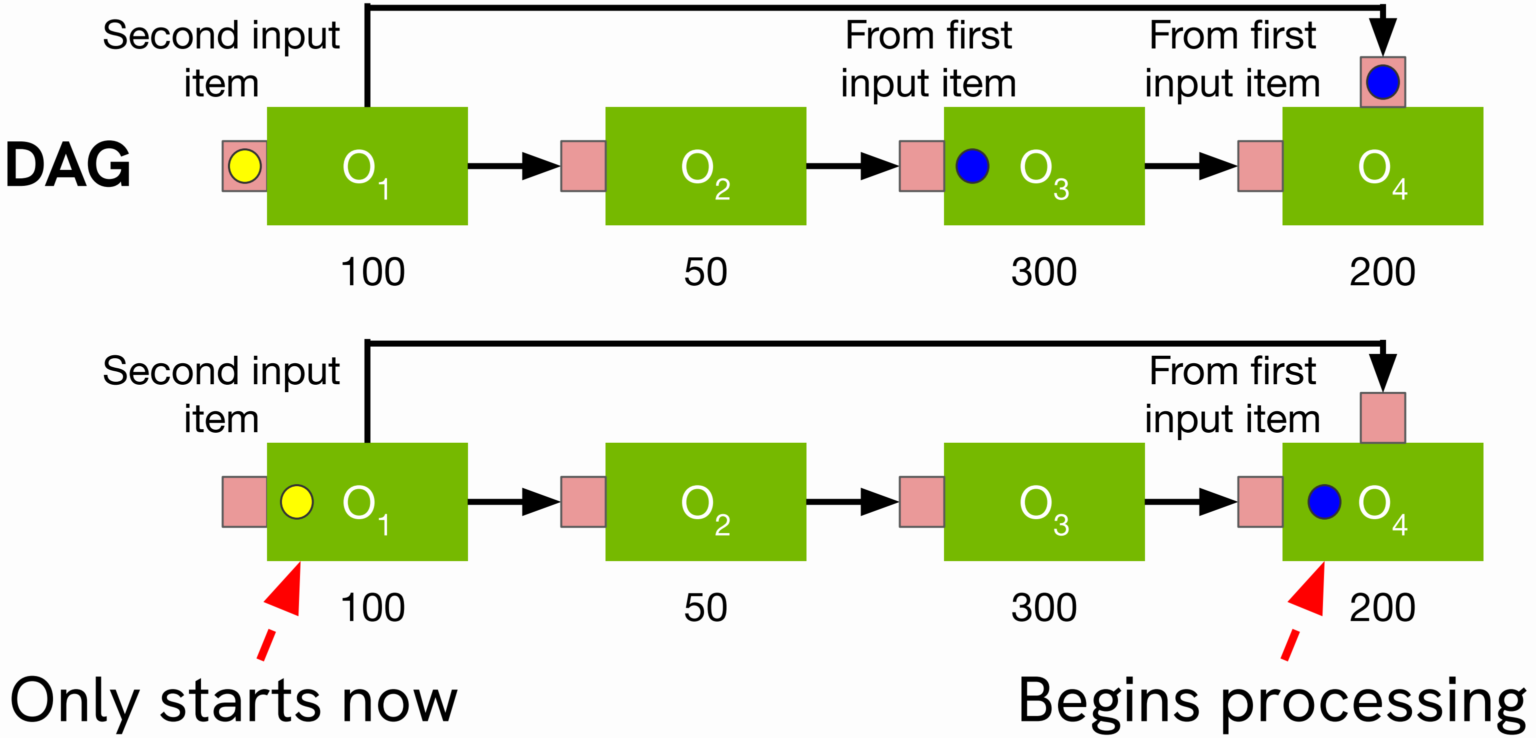
Chains vs DAGs



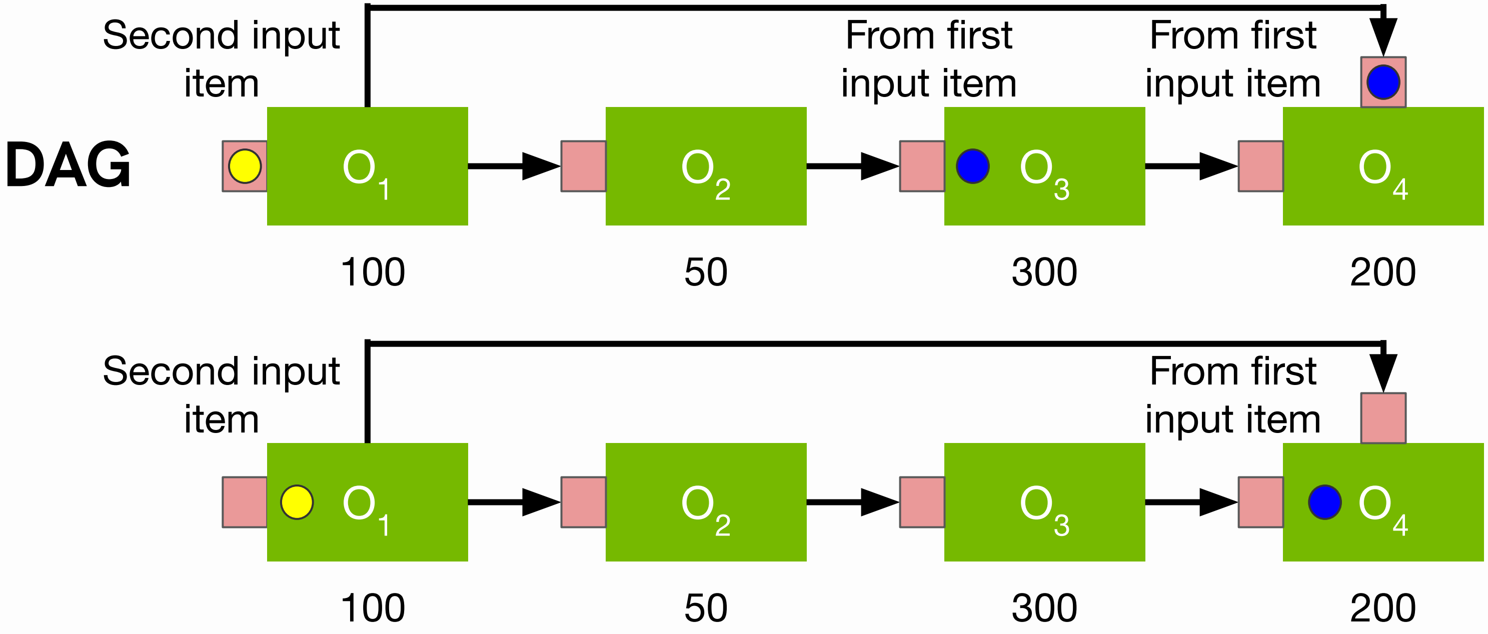
Chains vs DAGs



Chains vs DAGs



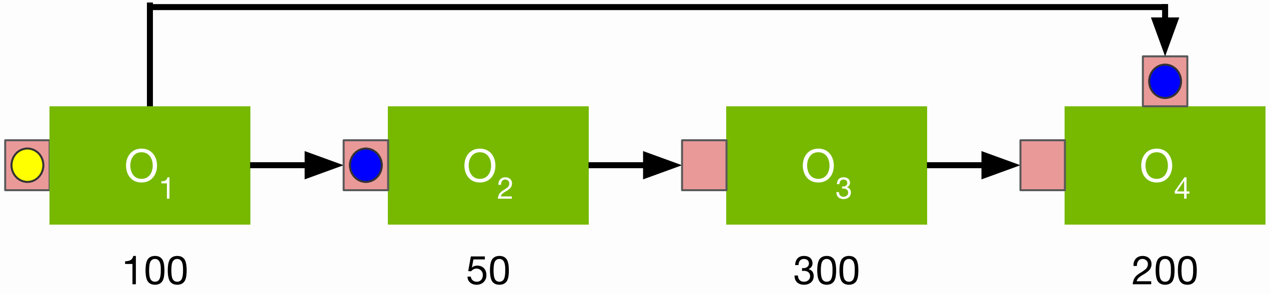
Chains vs DAGs



Takeaway: DAG had to wait longer than chain!

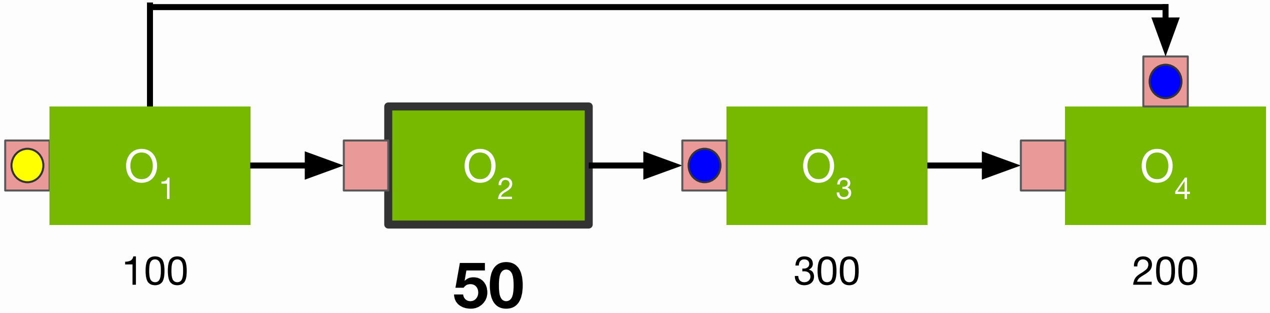
Inter-processing Delay

- The maximum time that can pass between two of an operator's consecutive outputs.



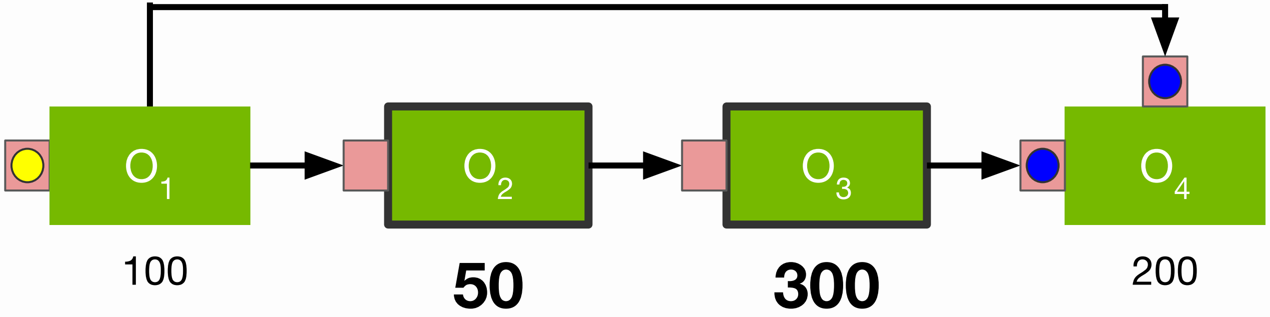
Inter-processing Delay

- The maximum time that can pass between two of an operator's consecutive outputs.



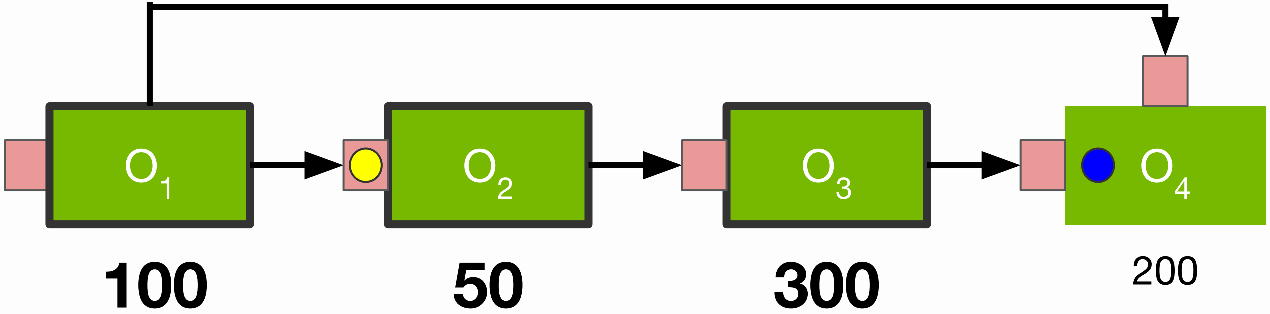
Inter-processing Delay

- The maximum time that can pass between two of an operator’s consecutive outputs.



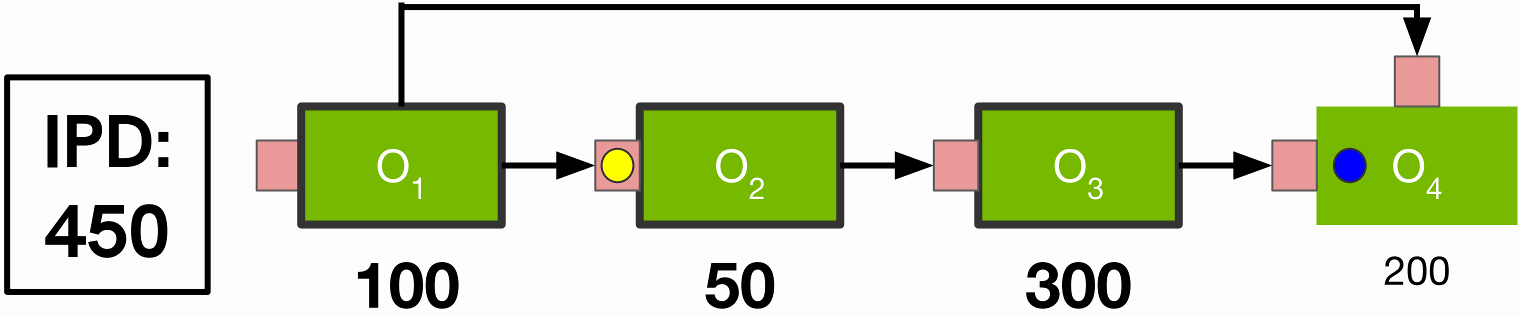
Inter-processing Delay

- The maximum time that can pass between two of an operator’s consecutive outputs.



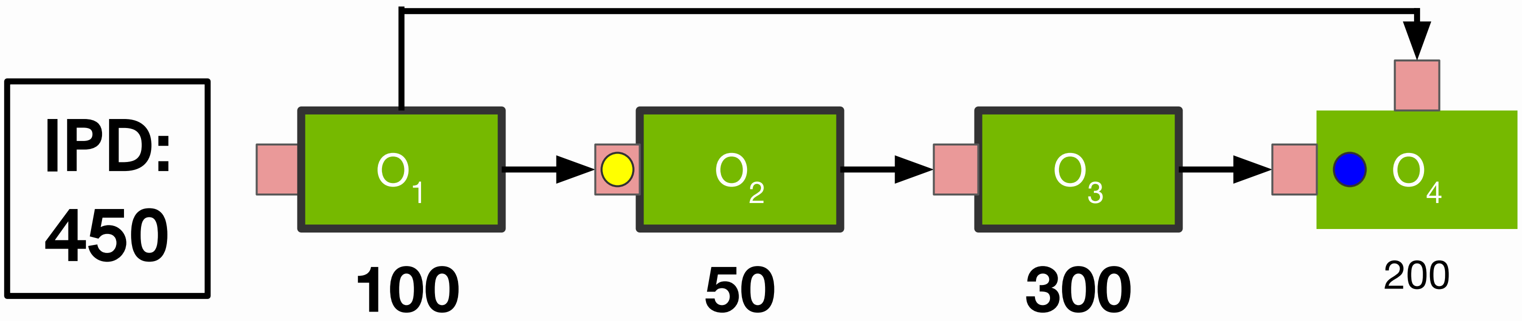
Inter-processing Delay

- The maximum time that can pass between two of an operator's consecutive outputs.



Inter-processing Delay

- The maximum time that can pass between two of an operator's consecutive outputs.

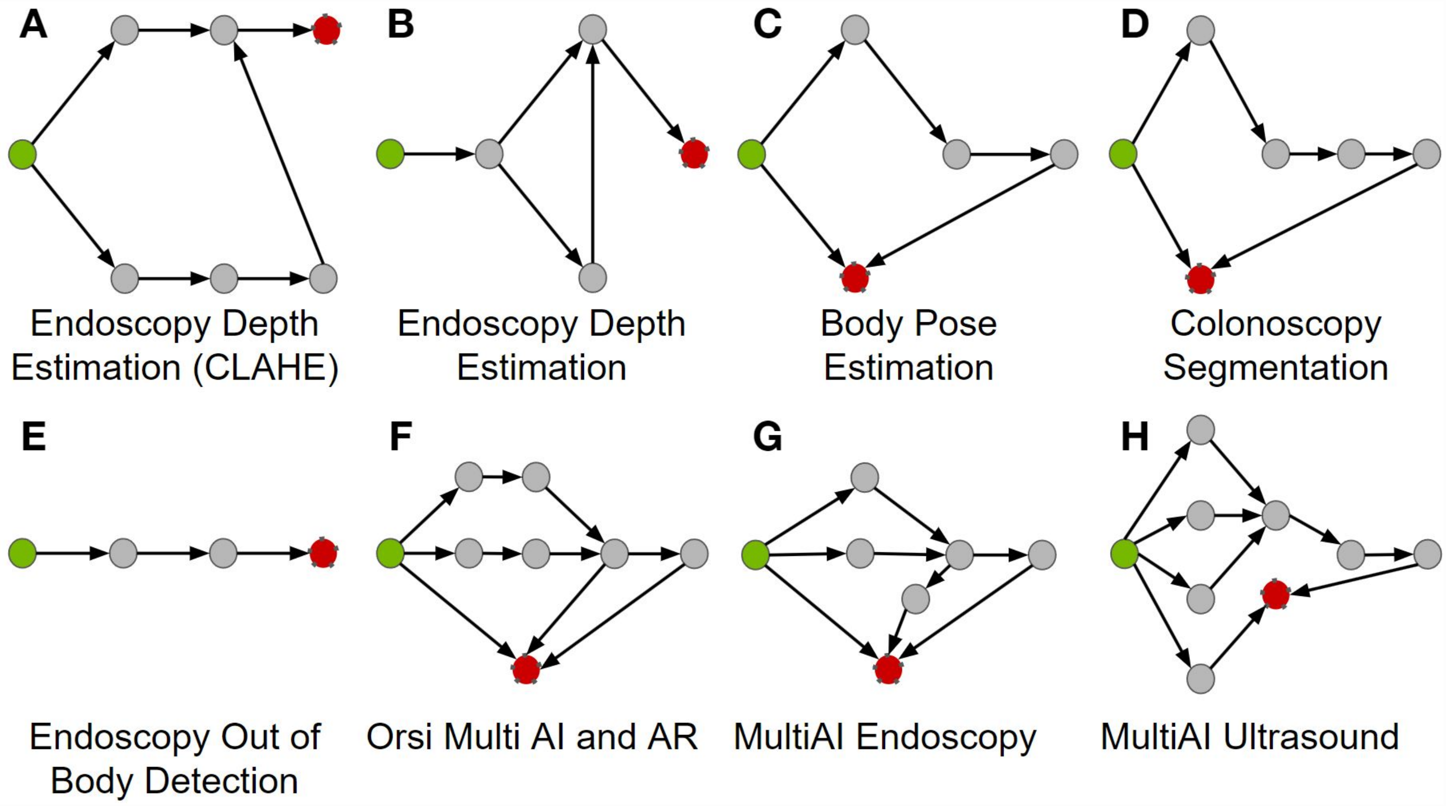


Key idea: use inter-processing delay term to generalize our linear chain bound to DAGs

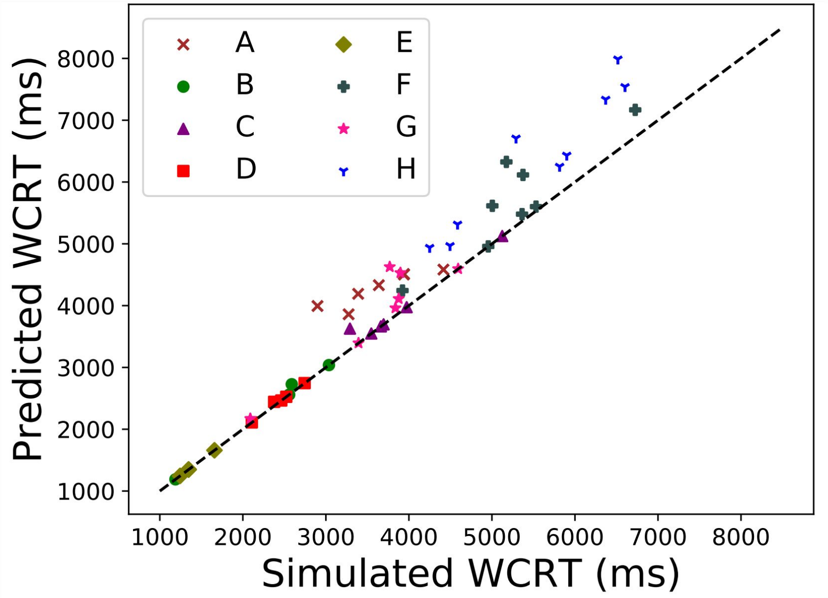
4

Evaluation

Evaluation: HoloHub Graph Structures



Evaluation: Bounds vs Sim and Profiled



- Compare theoretical WCRT to simulated and real executions
- Pessimism: The IPD we calculate may not be possible in practice

Takeaway: Closely bound most graph variations

5

Conclusion

What’s wrong with profiling?

- Profiling to learn timing properties has many issues
  - **The response time bound may be unsafe**
  - **Application development must be finished**
  - **Profiling can be costly in time and compute**

What’s wrong with profiling?

- Profiling to learn timing properties has many issues
  - **~~The response time bound may be unsafe~~**
  - **Application development must be finished**
  - **Profiling can be costly in time and compute**

What's wrong with profiling?

- Profiling to learn timing properties has many issues
  - **~~The response time bound may be unsafe~~**
  - **~~Application development must be finished~~**
  - **Profiling can be costly in time and compute**

What's wrong with profiling?

- Profiling to learn timing properties has many issues
  - ~~**The response time bound may be unsafe**~~
  - ~~**Application development must be finished**~~
  - ~~**Profiling can be costly in time and compute**~~

Takeaways

- First safe response-time bound for NVIDIA Holoscan
- Is applicable to arbitrary DAGs, hardware agnostic
  - Scalability experiments in paper
- Can help developers account for timing anomalies!
  - Observe directly how change in execution time corresponds to increase or decrease in response time

Future Research Areas

- Relax fixed execution time assumption
- Extend to core-constrained setting
- Fine-grained GPU-aware execution time analysis
- Independent applications running in parallel
- Transferring RTA results across different hardware

https://github.com/nvidia-holoscan/holohub/pull/600

Takeaways

- First safe response-time bound for NVIDIA Holoscan
- Is applicable to arbitrary DAGs
  - Scalability in paper
- Can help developers account for timing anomalies!
  - Observe directly how change in execution time corresponds to increase or decrease in response time
- **Available on NVIDIA HoloHub repository**

https://github.com/nvidia-holoscan/holohub/pull/600

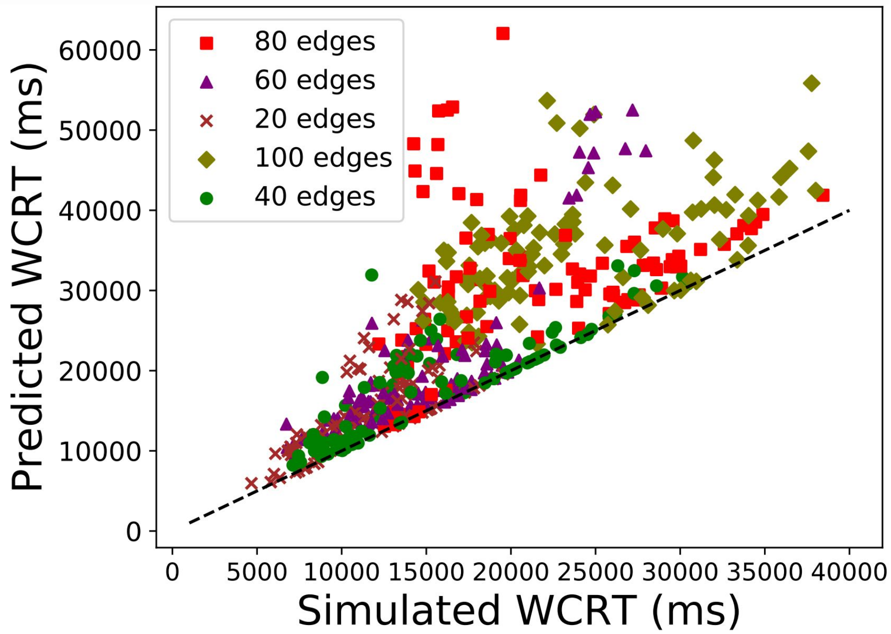
Takeaways

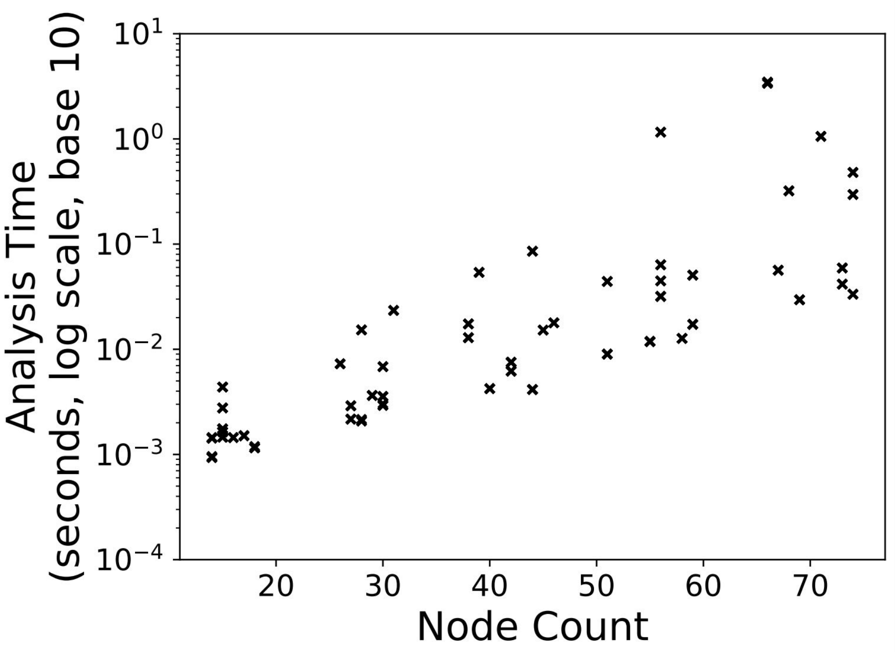
Thank you for listening! Questions?

- First safe response-time bound for NVIDIA Holoscan
- Is applicable to arbitrary DAGs
  - Scalability in paper
- Can help developers account for timing anomalies!
  - Observe directly how change in execution time corresponds to increase or decrease in response time
- Available on NVIDIA HoloHub repository

Backup

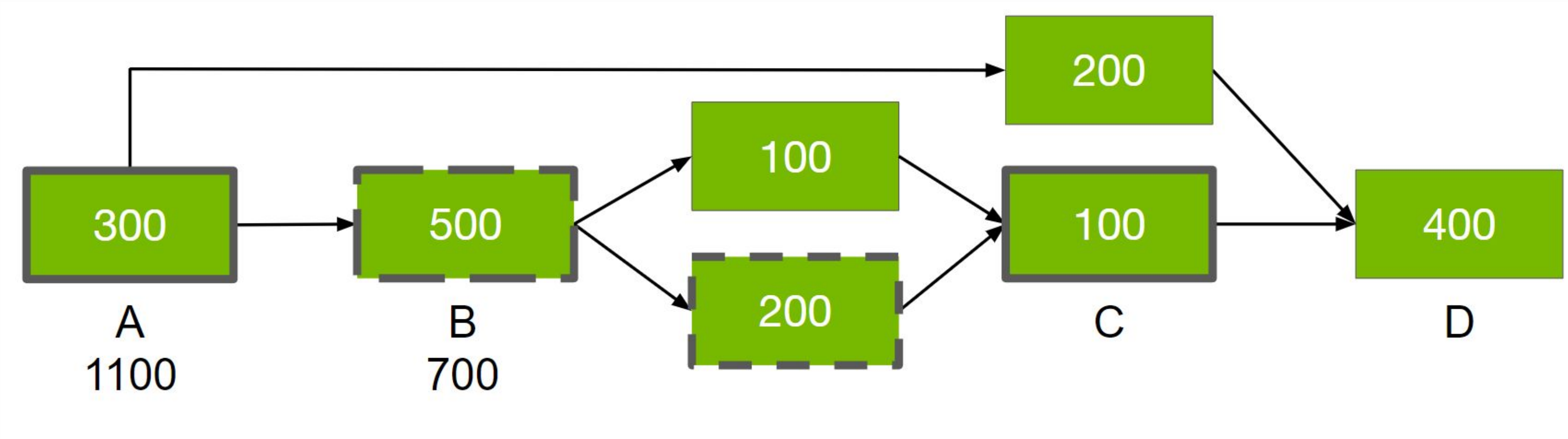
Evaluation: Scalability





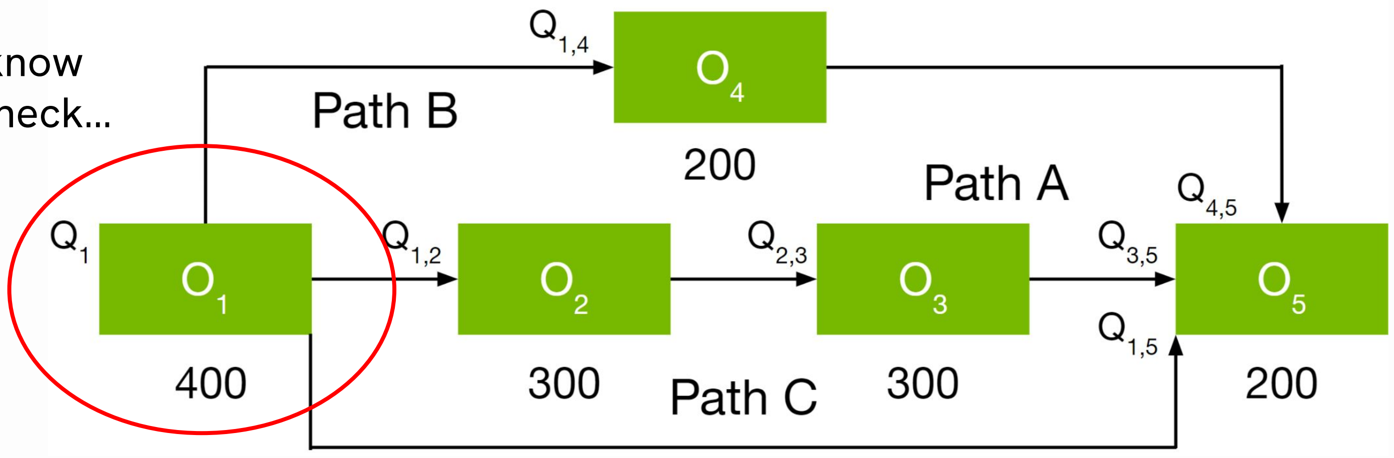
Takeaway: Small subset of graphs scale poorly

Inter-processing Delay Example



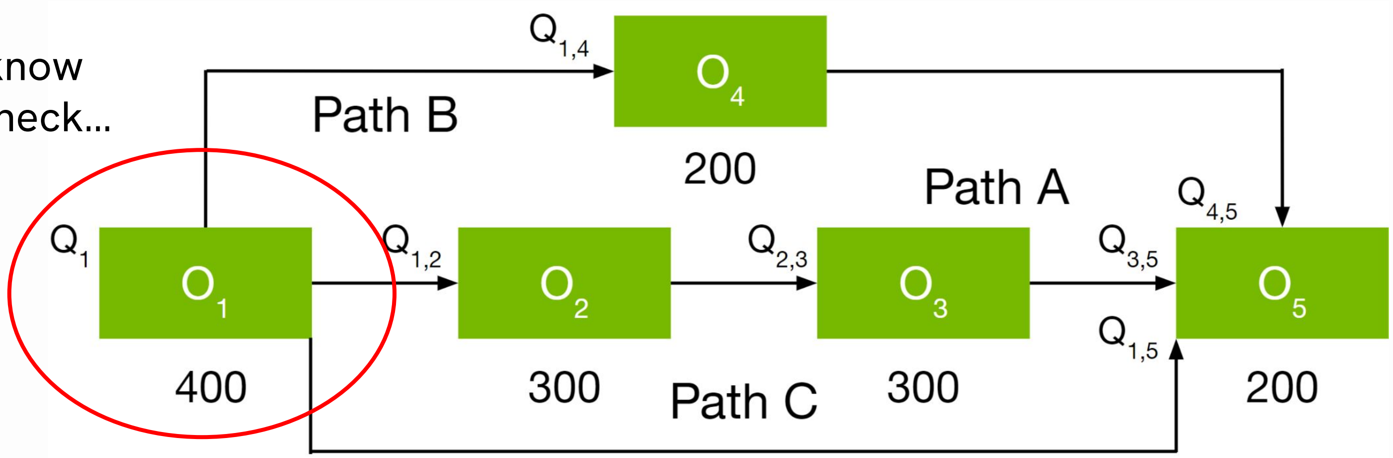
DAG Response Time Bound

If we know bottleneck...



DAG Response Time Bound

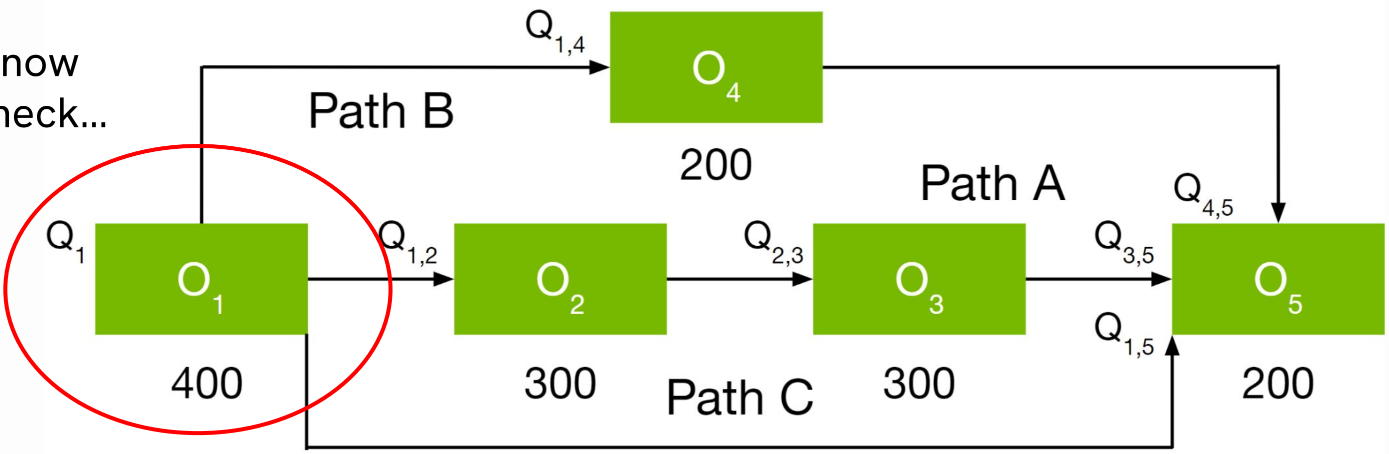
If we know bottleneck...



Upper bound: $\delta_b^{ub} \cdot (m_y + 1) + \sum_{k=1}^{n_x} e_{x_k}^{ub}$

DAG Response Time Bound

If we know bottleneck...



Upper bound: $\delta_b^{ub} \cdot (m_y + 1) + \sum_{k=1}^{n_x} e_{x_k}^{ub}$

Sum of worst-case execution times of operators following bottleneck

Bottleneck worst-case inter-processing delay multiplied by the length of shortest path to the source