上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

SJTU

DVABatch: Diversity-aware Multi-Entry Multi-Exit Batching for Efficient Processing of DNN Services on GPUs

Weihao Cui[1], Han Zhao[1], Quan Chen[1], Hao Wei[1], Zirui Li[1], Deze Zeng[2], Chao Li[1], and Minyi Guo[1]

[1]Shanghai Jiao Tong University, Emerging Parallel Computing Center
[2]China University of Geosciences

饮 水 思 源 • 爱 国 荣 校

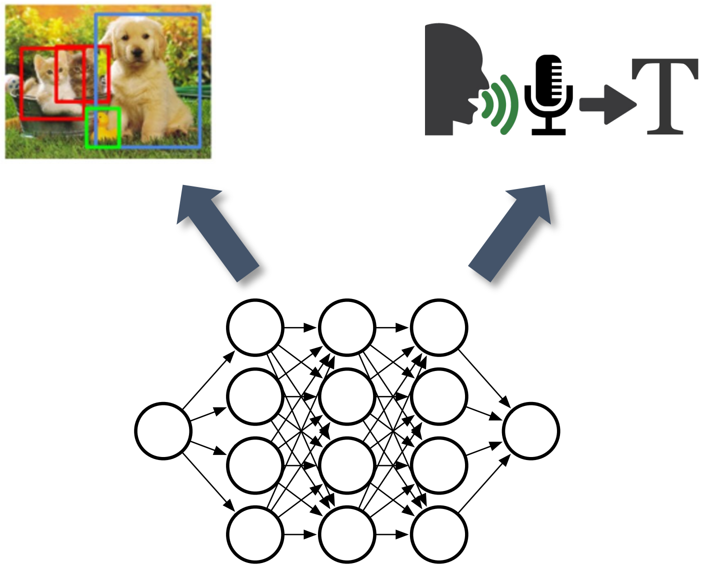
01

Background & Motivation

Deploying DNN Services On GPUs

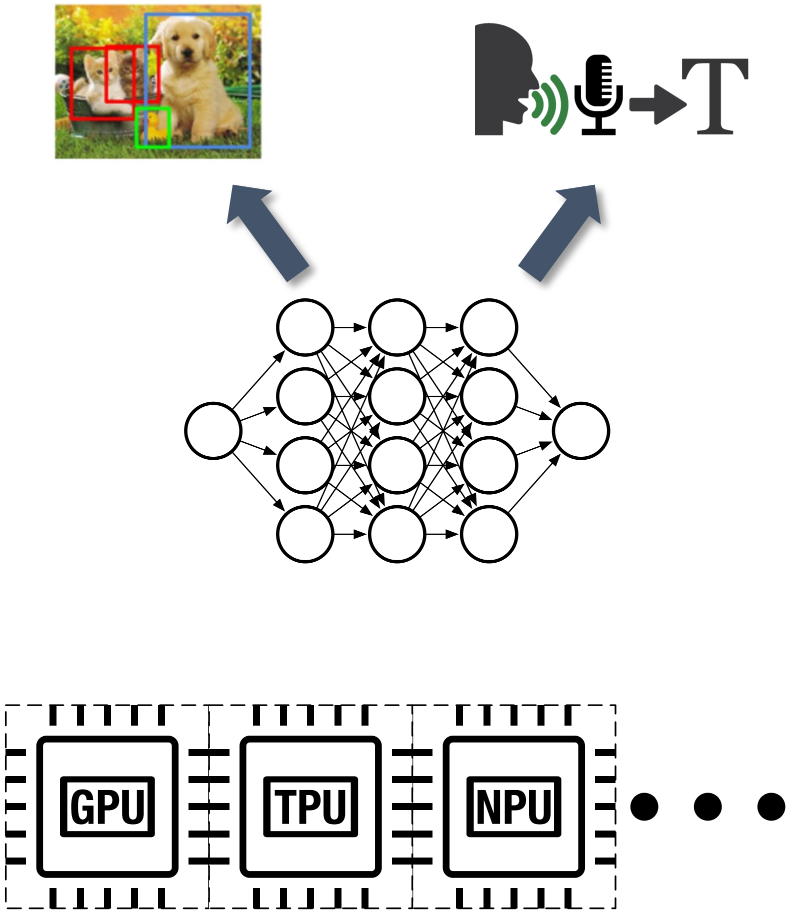
Deploying DNN Services On GPUs

- DNN techniques are powering cloud services.



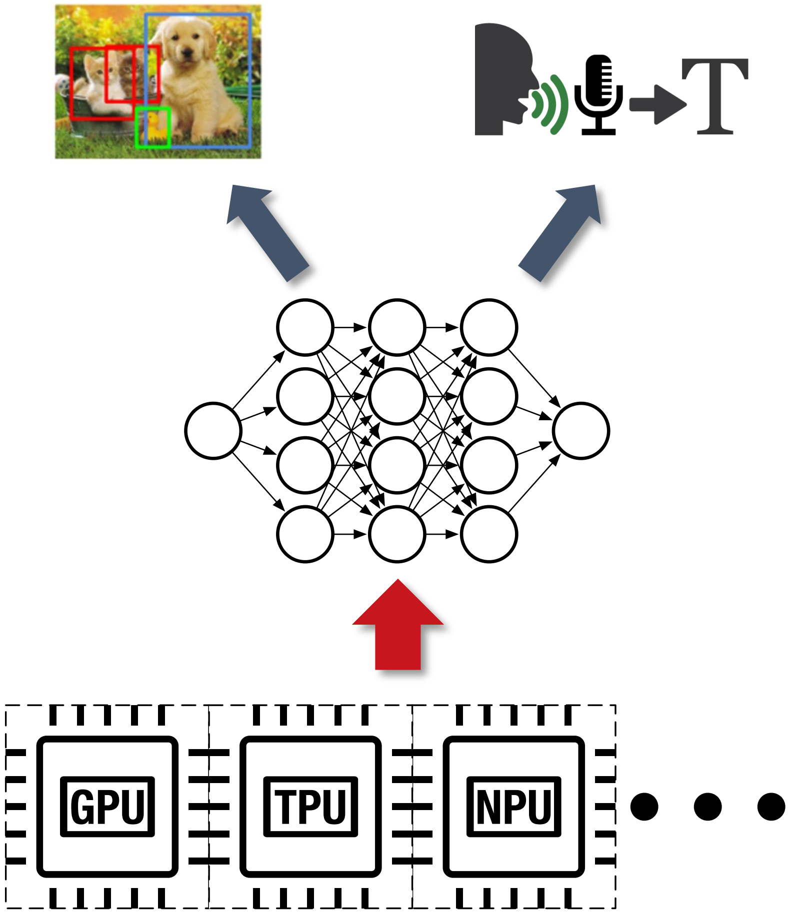
Deploying DNN Services On GPUs

- DNN techniques are powering cloud services.
- Dedicated accelerators like GPUs speed up DNN inferences.



Deploying DNN Services On GPUs

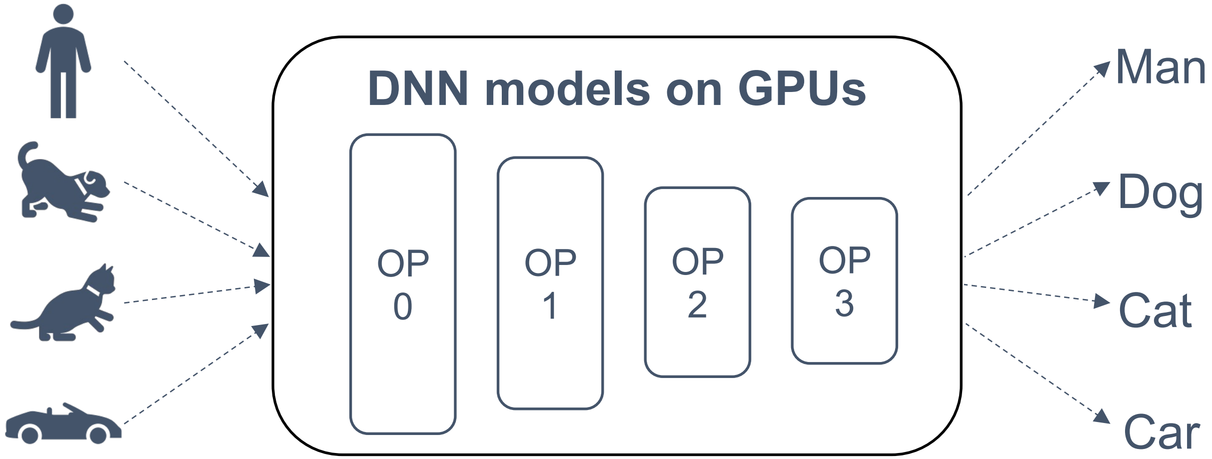
- DNN techniques are powering cloud services.
- Dedicated accelerators like GPUs speed up DNN inferences.
- **Important to process DNN-based services efficiently on GPUs.**



Batching Improves Efficiency

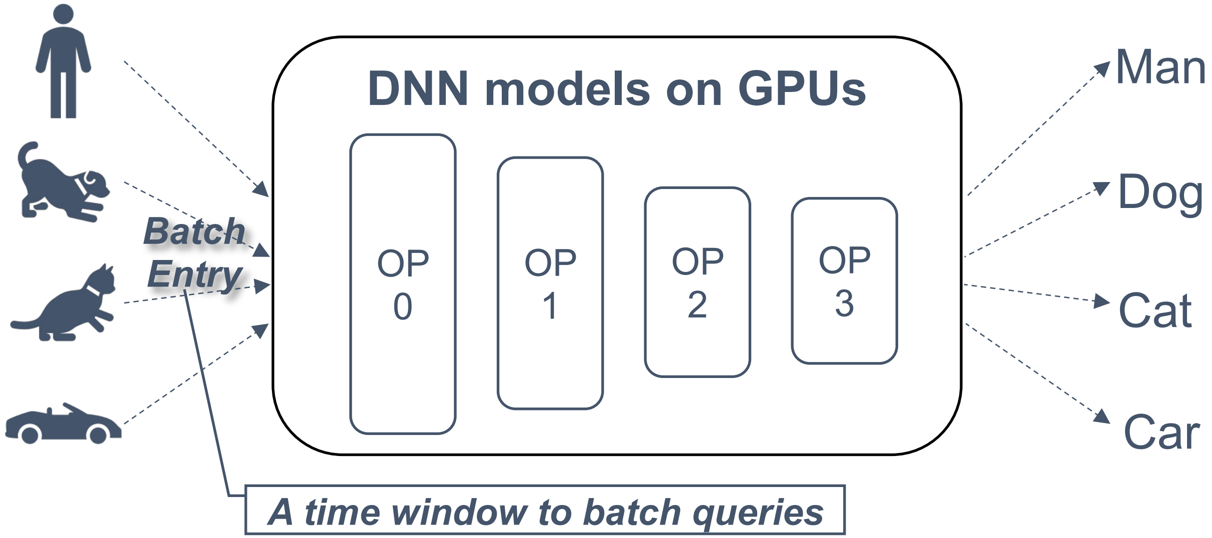
Batching Improves Efficiency

- A simple example: batched image classification.



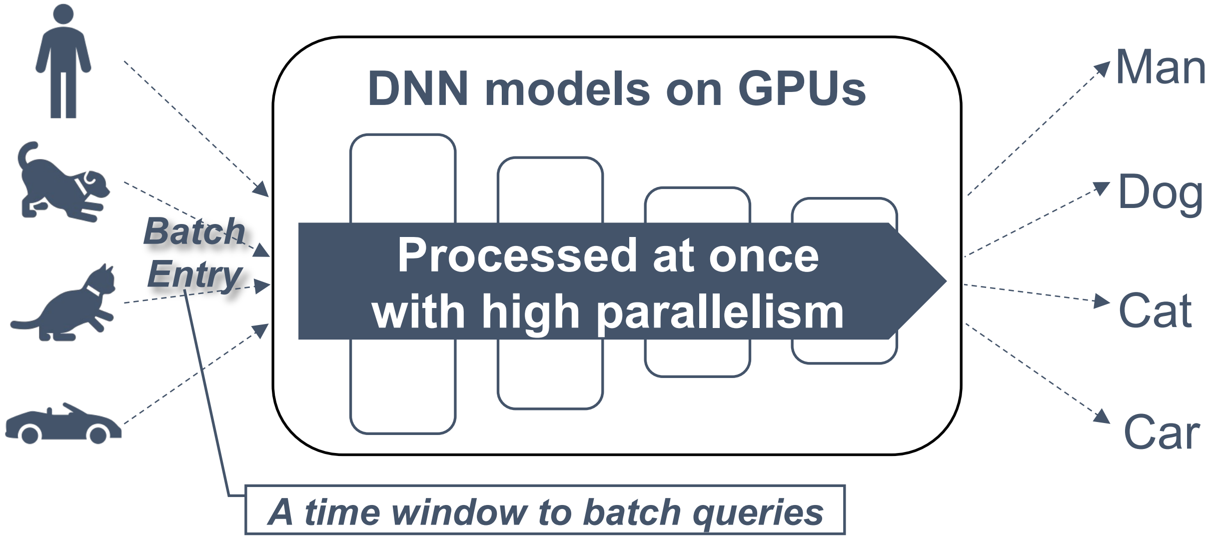
Batching Improves Efficiency

- A simple example: batched image classification.



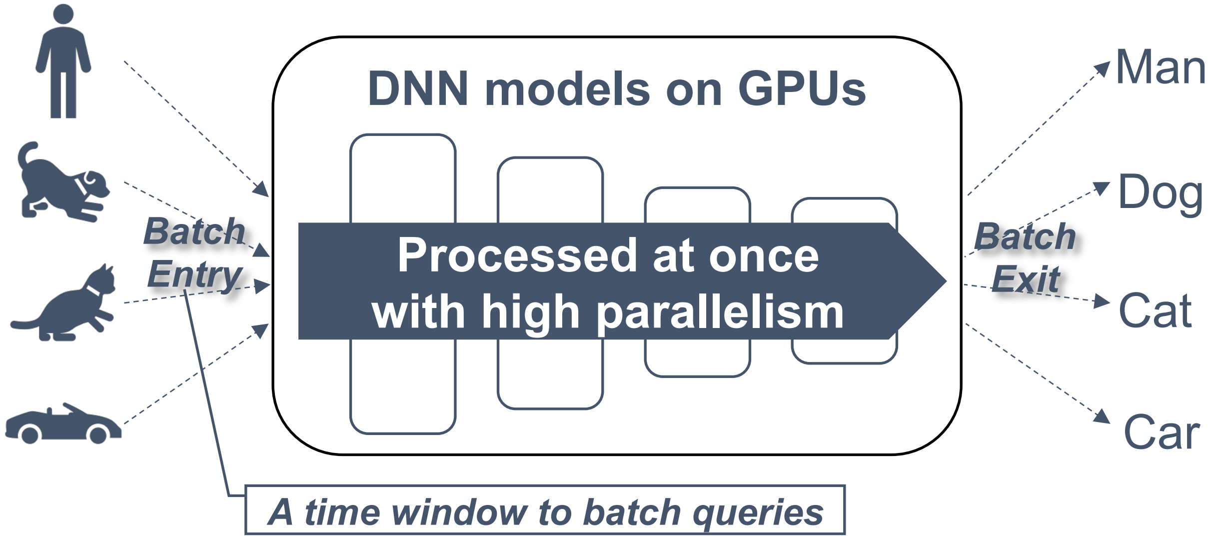
Batching Improves Efficiency

- A simple example: batched image classification.



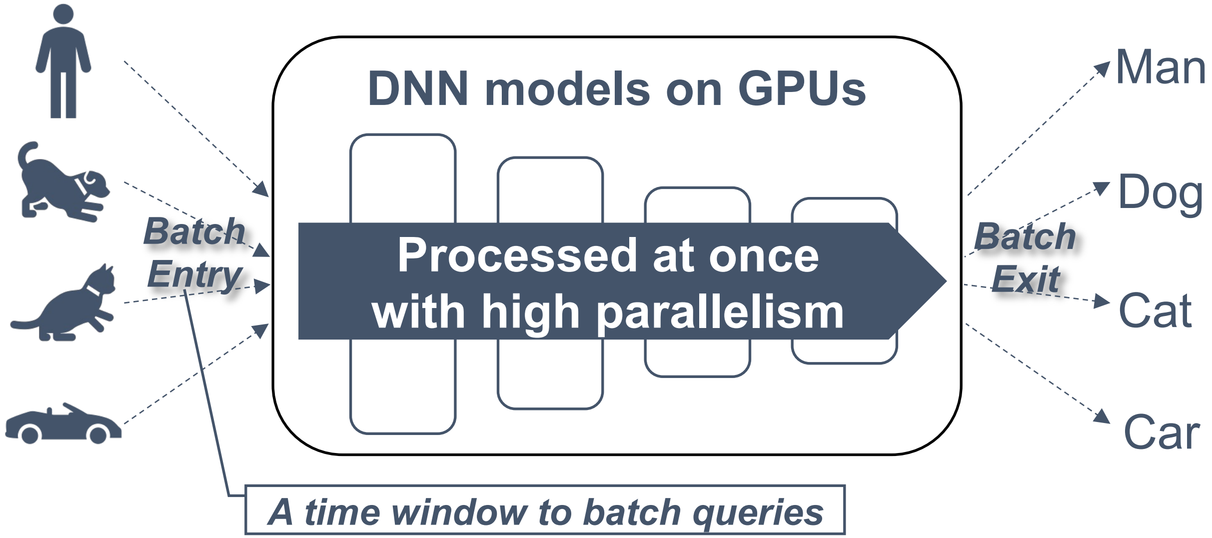
Batching Improves Efficiency

- A simple example: batched image classification.



Batching Improves Efficiency

- A simple example: batched image classification.



Serving diversities <u>diminish</u> the efficiency of the single-entry single-exit scheme.

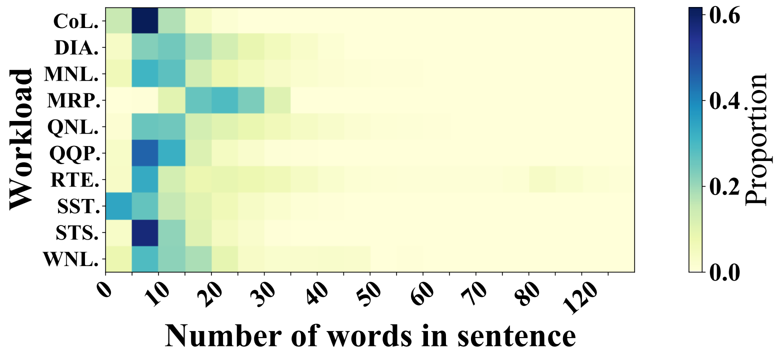
Inefficiency due to Diversities

Inefficiency due to Diversities

- Input diversity: DNN model accept inputs with different shapes.

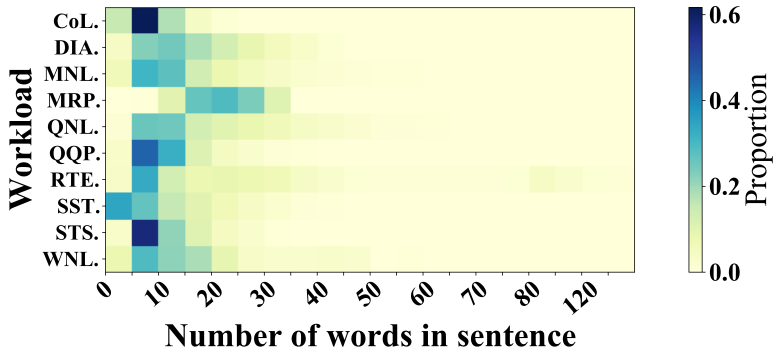
Inefficiency due to Diversities

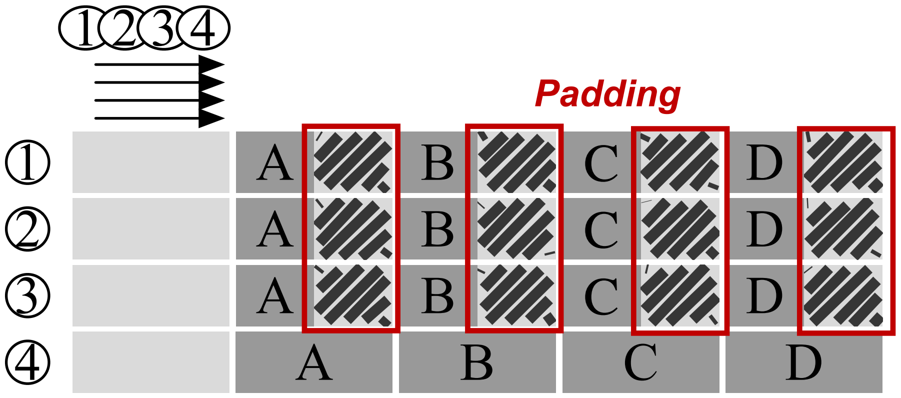
- Input diversity: DNN model accept inputs with different shapes.



Inefficiency due to Diversities

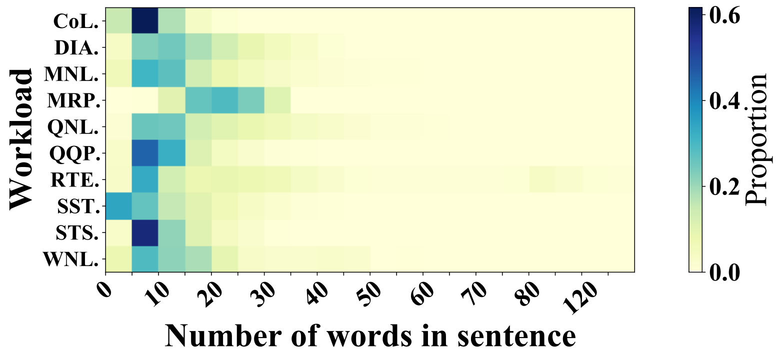
- Input diversity: DNN model accept inputs with different shapes.

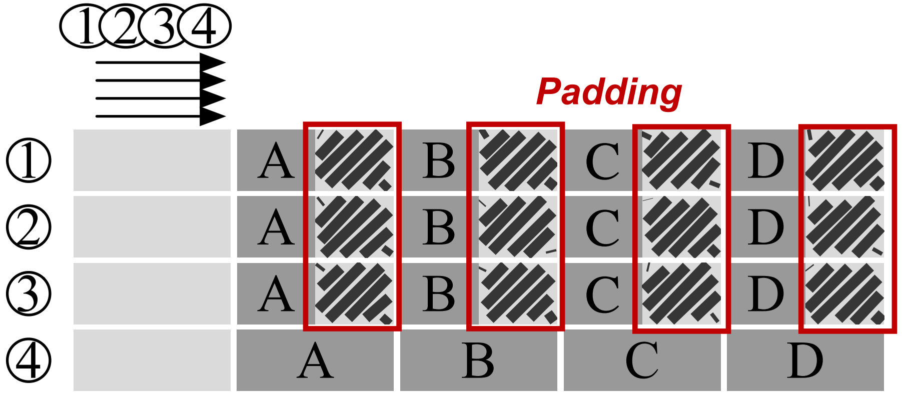




Inefficiency due to Diversities

- Input diversity: DNN model accept inputs with different shapes.

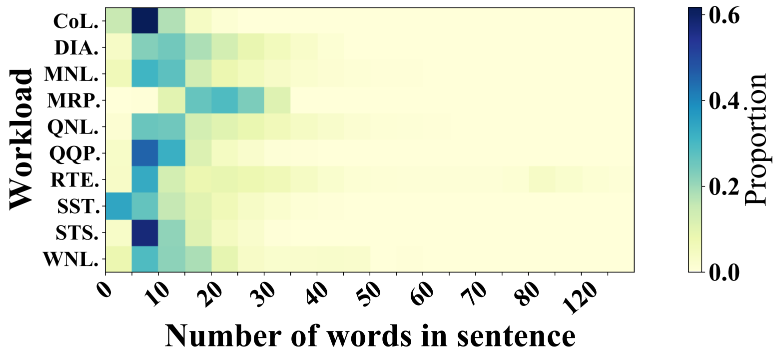


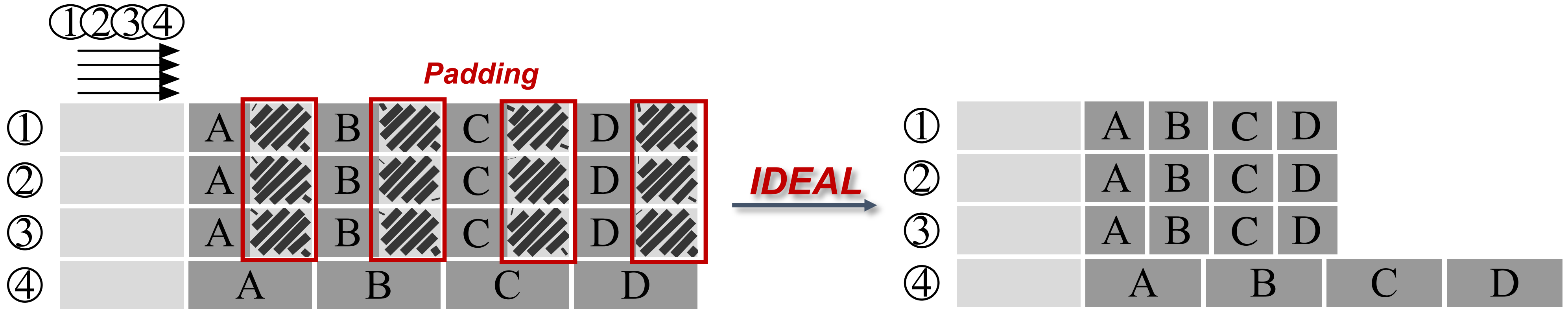


- ***Extra padding for input diversity results in wasted computation.***

Inefficiency due to Diversities

- Input diversity: DNN model accept inputs with different shapes.

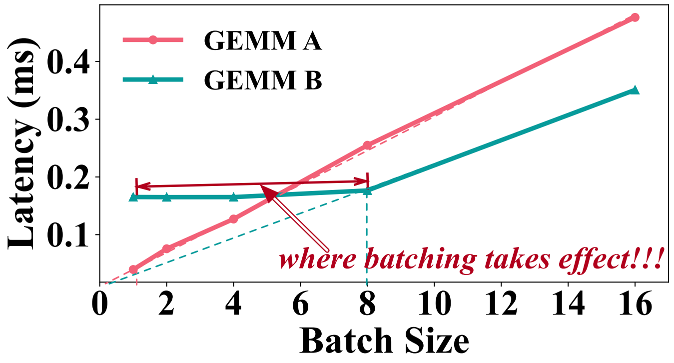




- ***Extra padding for input diversity results in wasted computation.***

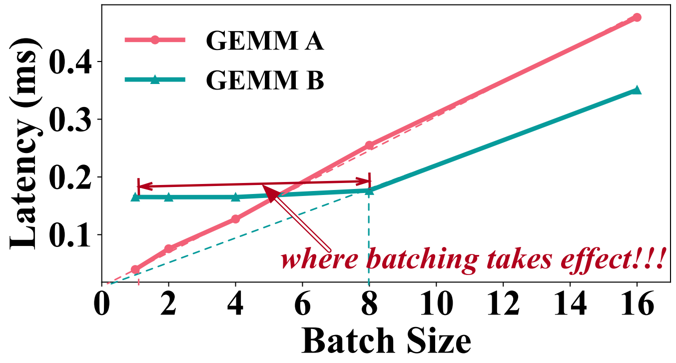
Inefficiency due to Diversities

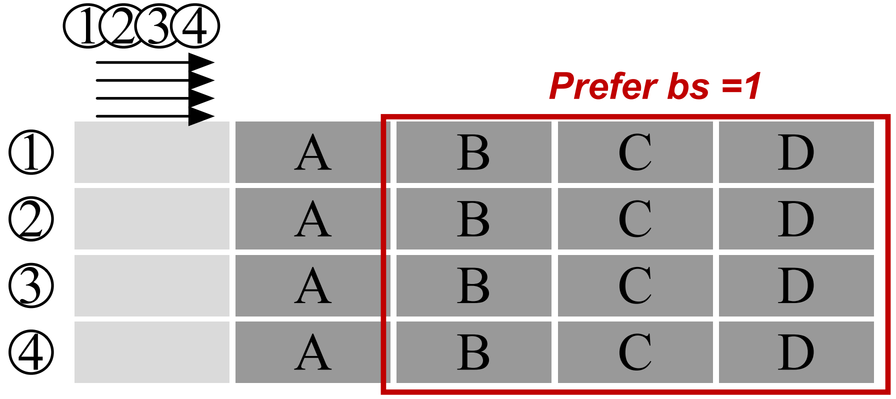
- Operator diversity: operators require different BS to fully utilize the GPU.



Inefficiency due to Diversities

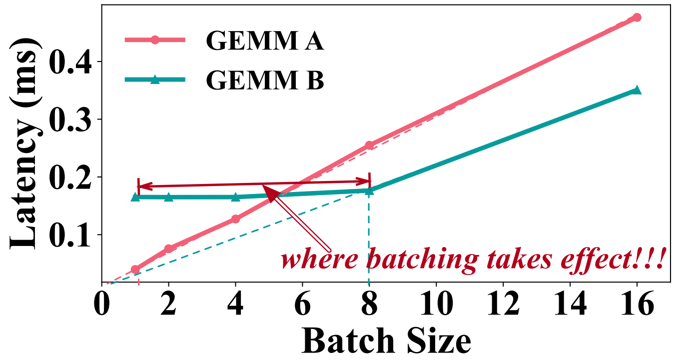
- Operator diversity: operators require different BS to fully utilize the GPU.

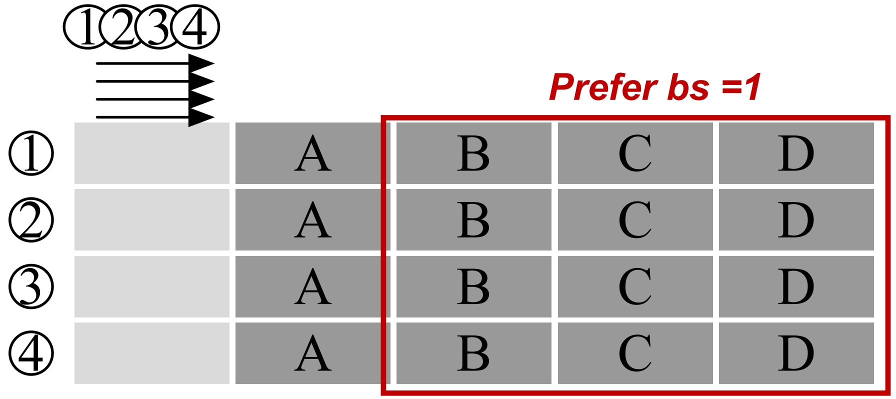




Inefficiency due to Diversities

- Operator diversity: operators require different BS to fully utilize the GPU.

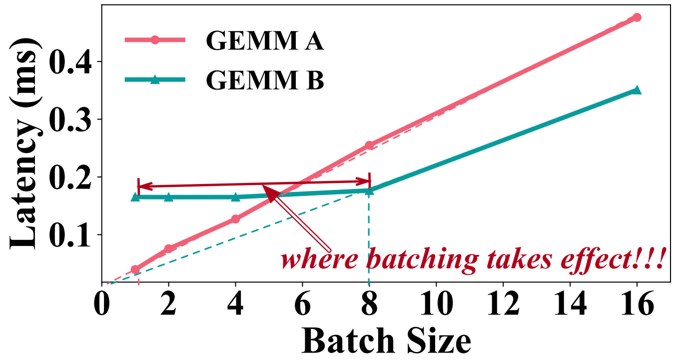


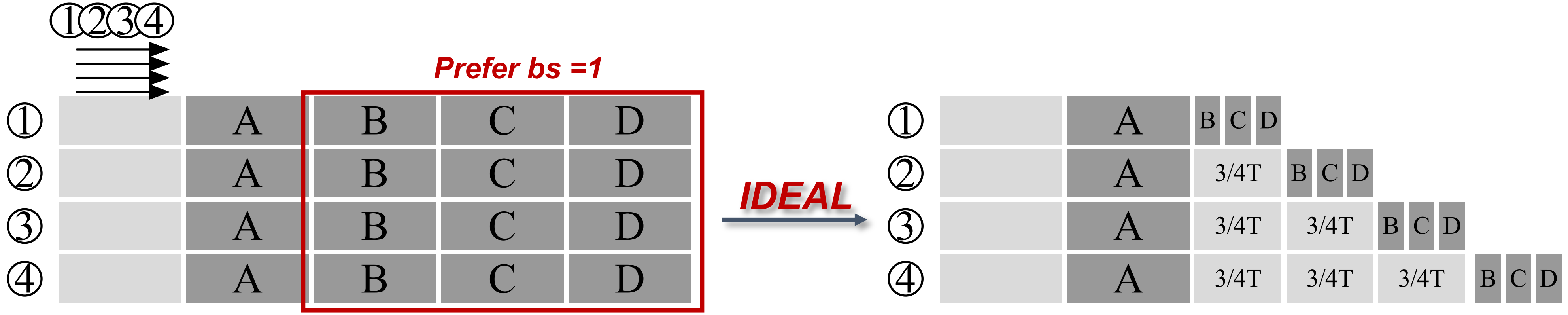


- ***Longer latency without throughput improvement***

Inefficiency due to Diversities

- Operator diversity: operators require different BS to fully utilize the GPU.

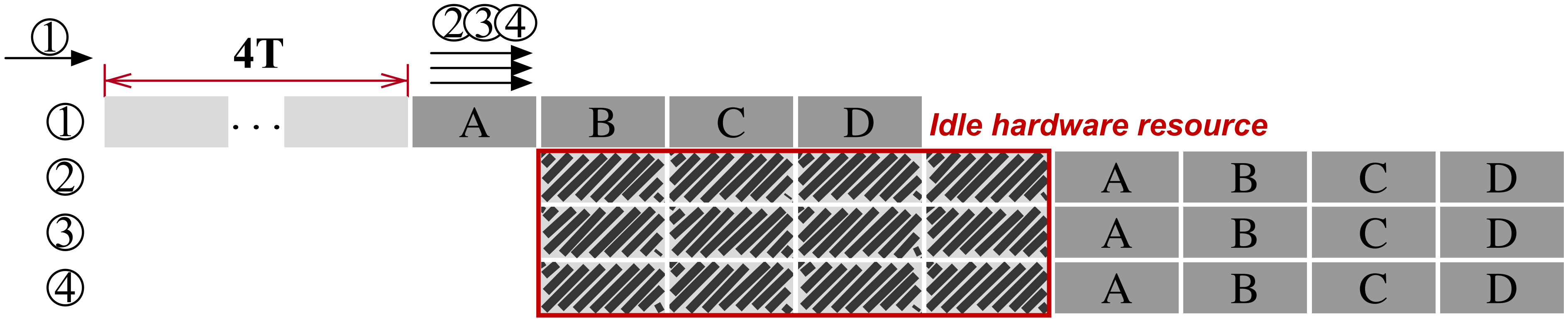




- ***Longer latency without throughput improvement***

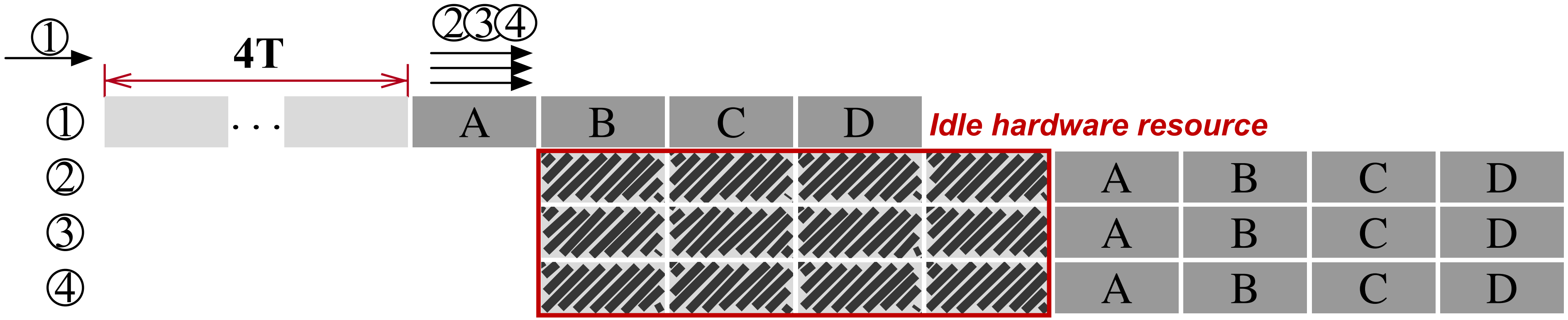
Inefficiency due to Diversities

- Load diversity: insufficient batch due to load variation.



Inefficiency due to Diversities

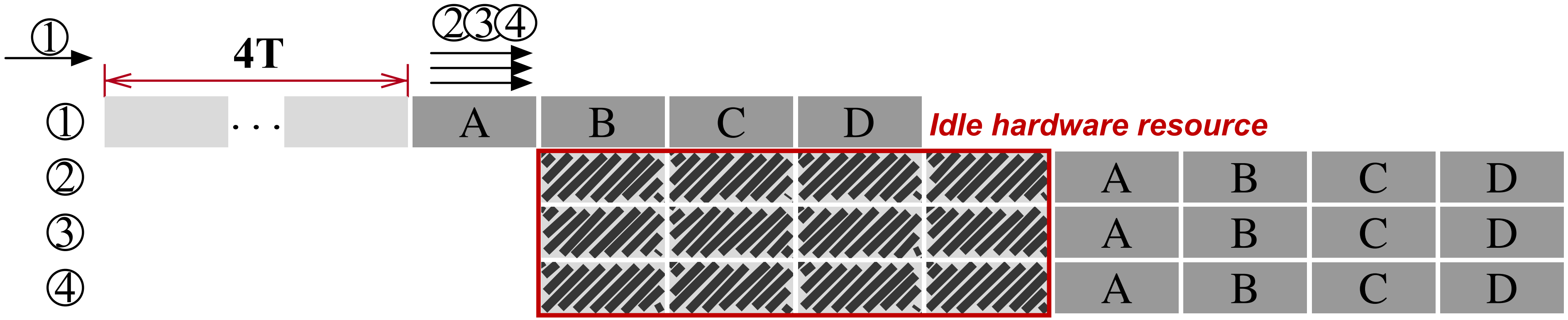
- Load diversity: insufficient batch due to load variation.



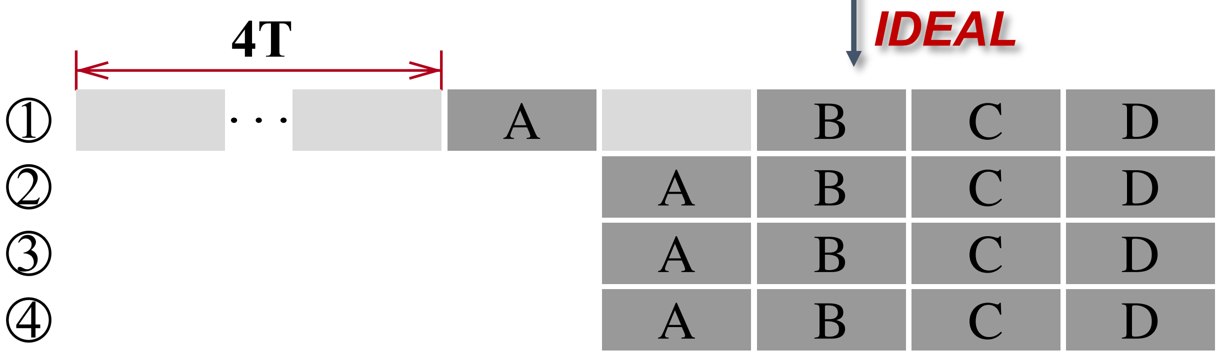
- ***Hardware is idle while there are queries for processing***

Inefficiency due to Diversities

- Load diversity: insufficient batch due to load variation.



- ***Hardware is idle while there are queries for processing***



Weakness of Single-entry Single-exit

- Assume that batch always have positive effects.
- An ongoing batch cannot be interrupted for adjustment.
- Cannot be configured to support various diversities.

Weakness of Single-entry Single-exit

- Assume that batch always have positive effects.
- An ongoing batch cannot be
- Cannot be configured to support various diversities.

A multi-entry multi-exit batch scheme?

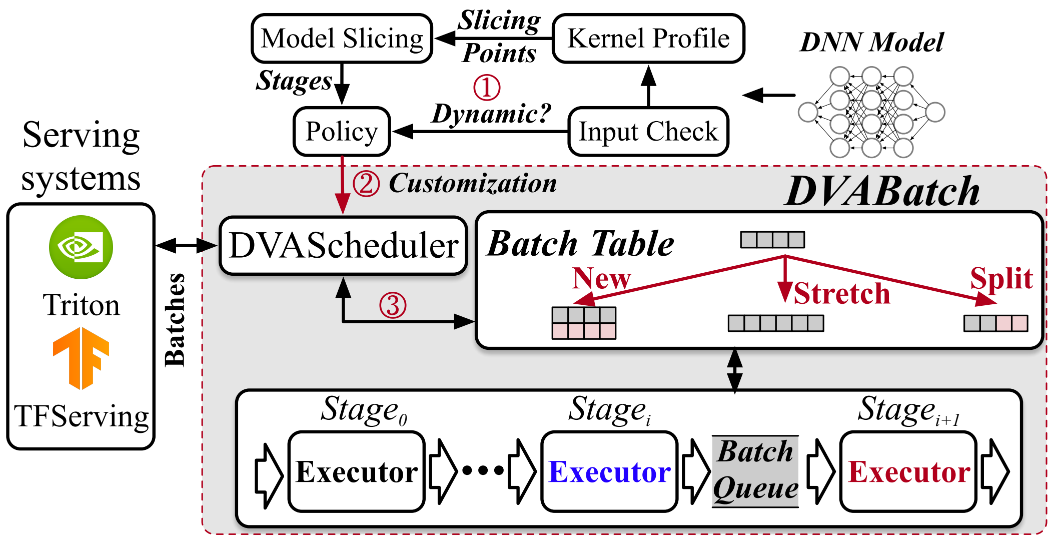
SJTU

02

Methodology

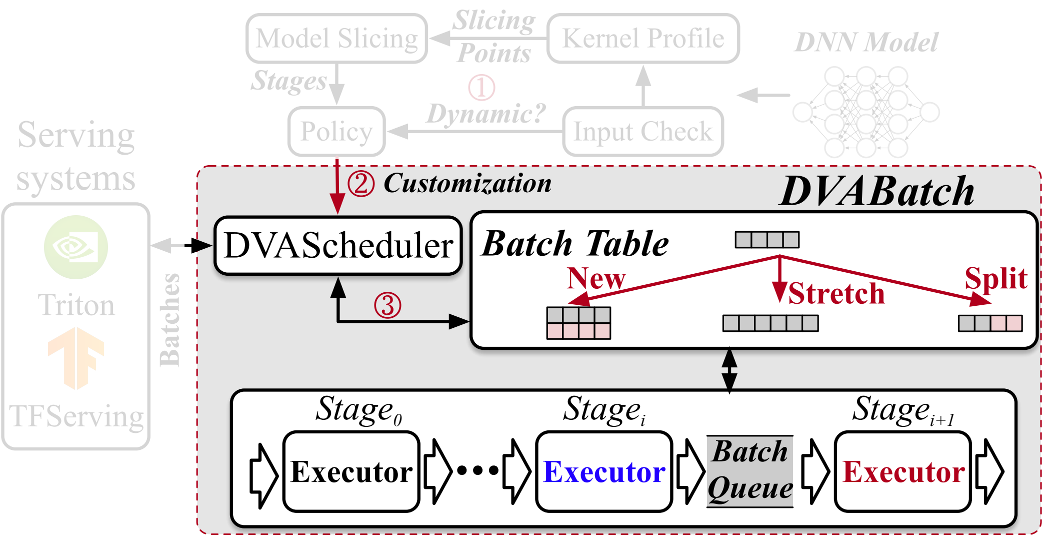
The Goal of DVABatch

- A runtime batch scheduling system integrated into existing frameworks
- A multi-entry multi-exit scheme for adjusting the batch on the fly.
- A holistic solution for different serving diversities.



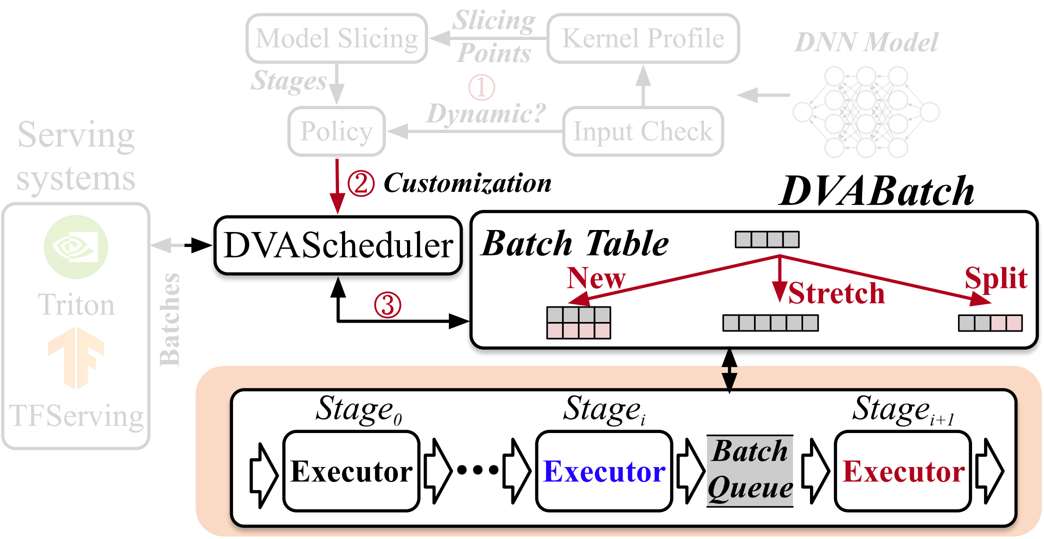
Overview of DVABatch

- Holistic multi-entry and multi-exit scheme



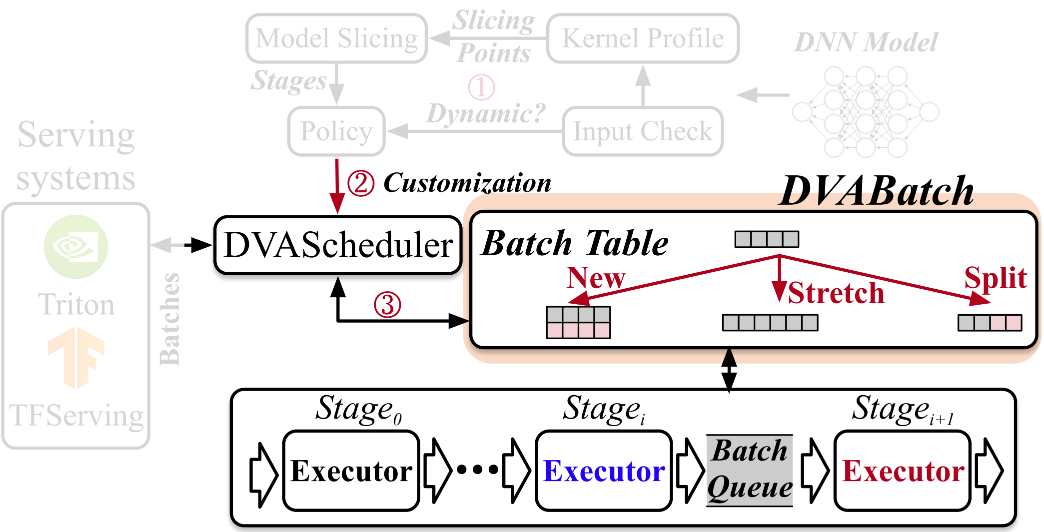
Overview of DVABatch

- Holistic multi-entry and multi-exit scheme
  - **Slice** DNN model into multiple stages (executors connected by queue).



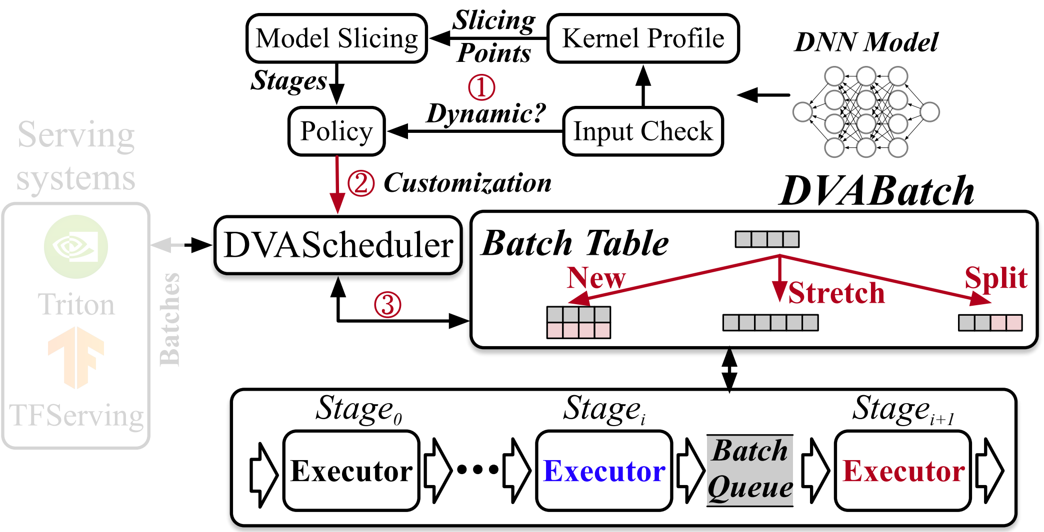
Overview of DVABatch

- Holistic multi-entry and multi-exit scheme
  - **Slice** DNN model into multiple stages (executors connected by queue).
  - **Three meta-operations** for adjusting the batching.



Overview of DVABatch

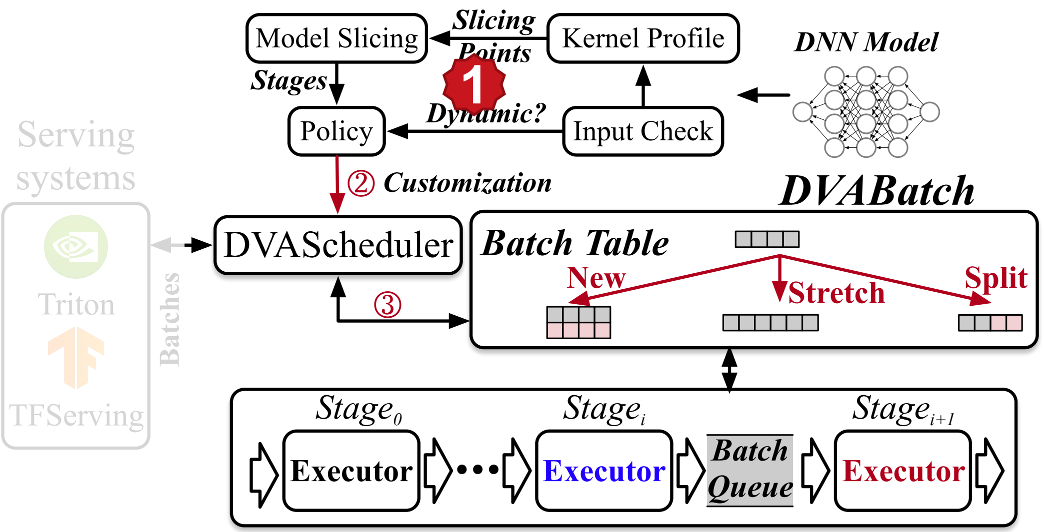
- Workflow for a new DNN services



Overview of DVABatch

- Workflow for a new DNN services

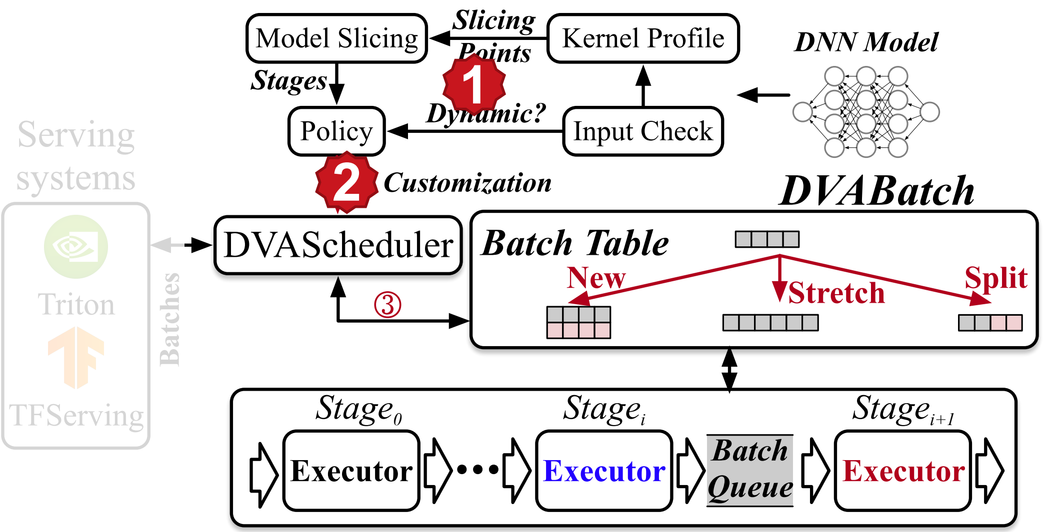
1 Identify the serving diversities through and slice model.



Overview of DVABatch

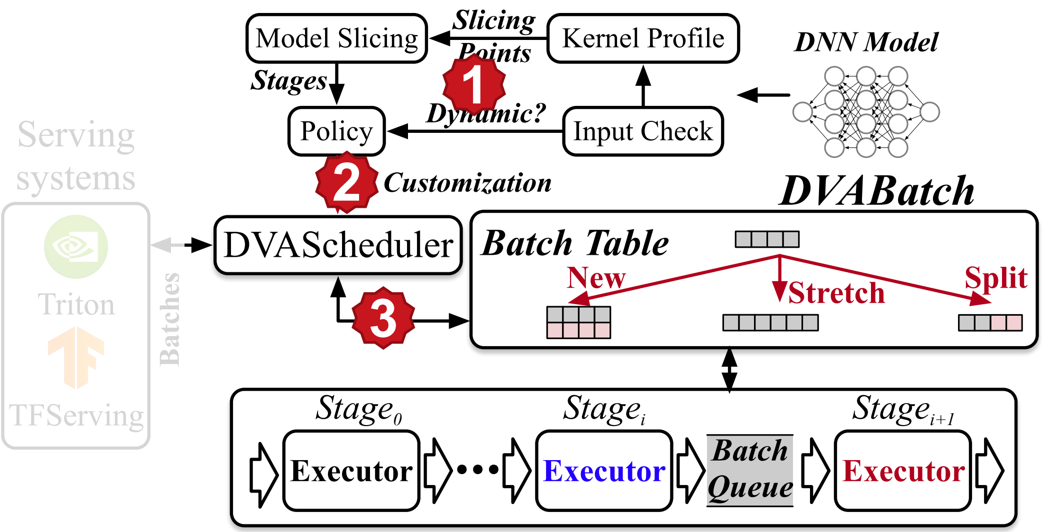
- Workflow for a new DNN services

1. Identify the serving diversities through and slice model.
2. Load models and customize the scheduler.



Overview of DVABatch

- Workflow for a new DNN services
  1. Identify the serving diversities through and slice model.
  2. Load models and customize the scheduler.
  3. Exploit meta operations for efficient processing.

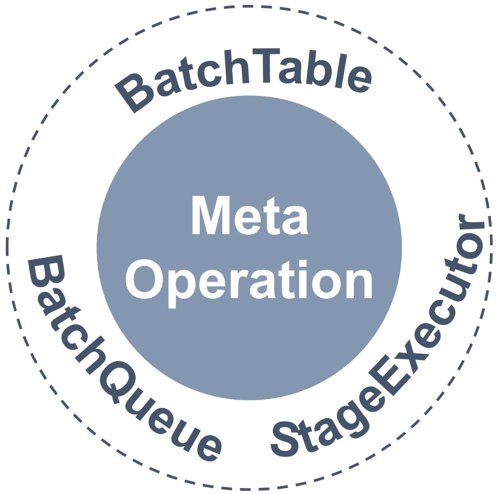


Holistic Multi-entry Multi-exit Scheme

- Key abstraction: meta operations to adjust an on-going batch

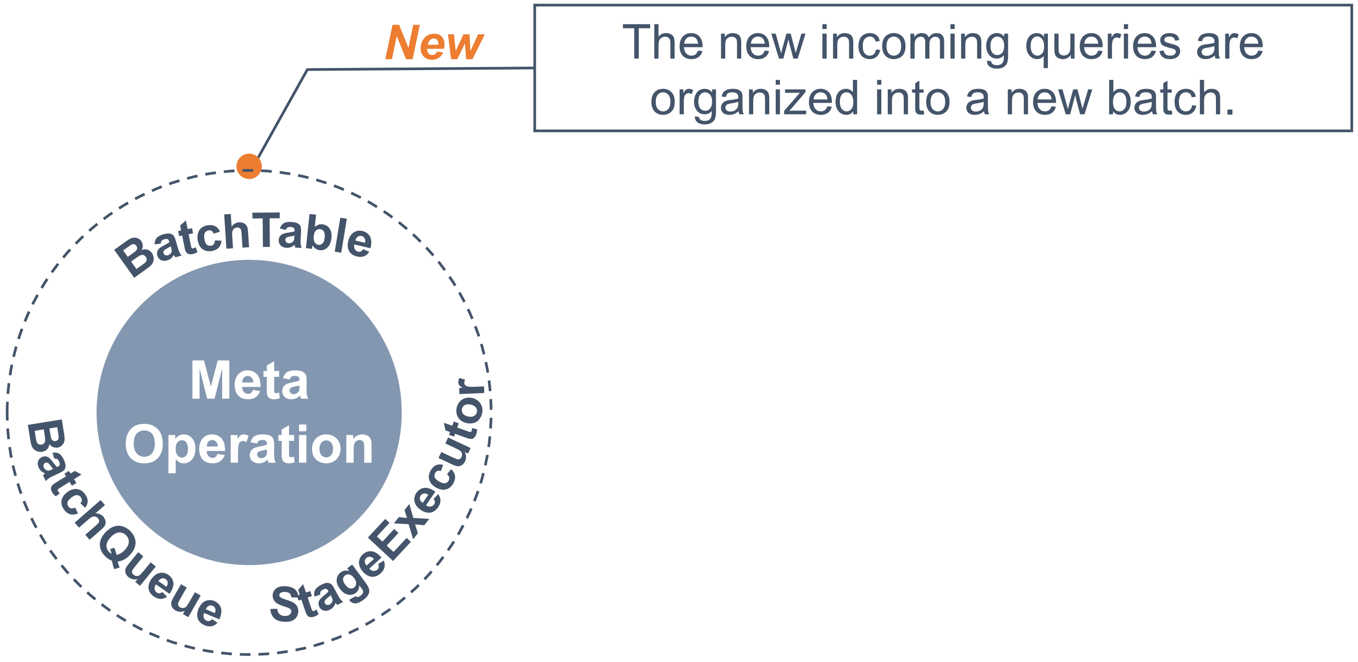
Holistic Multi-entry Multi-exit Scheme

- Key abstraction: meta operations to adjust an on-going batch



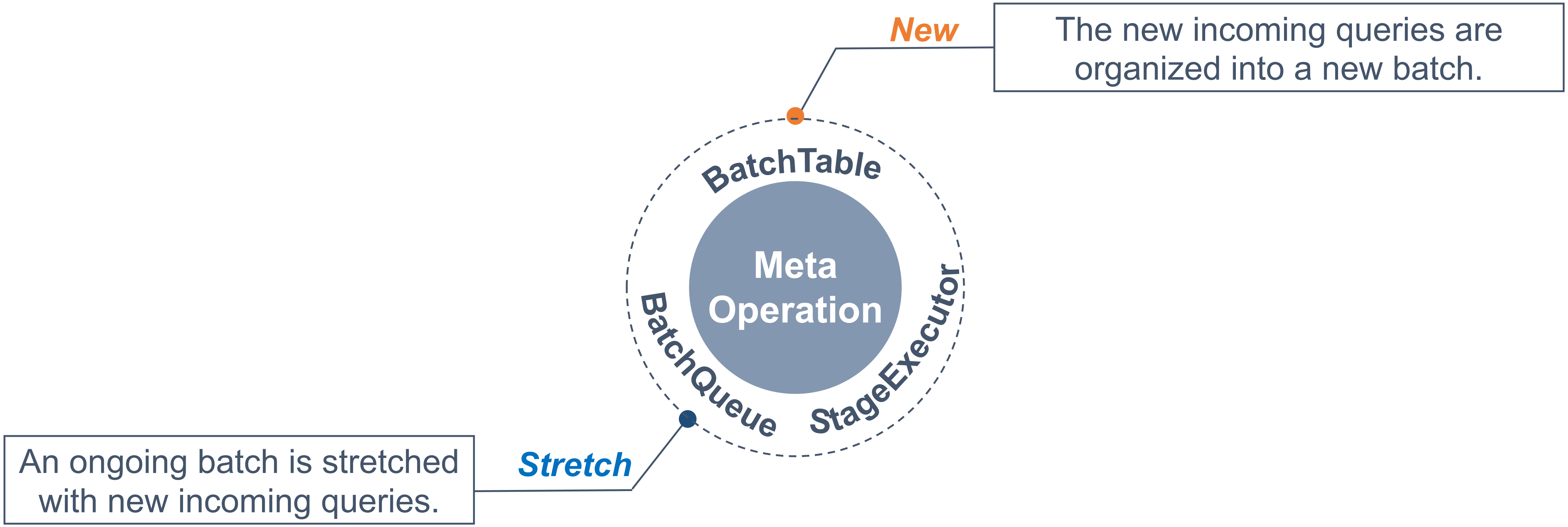
Holistic Multi-entry Multi-exit Scheme

- Key abstraction: meta operations to adjust an on-going batch



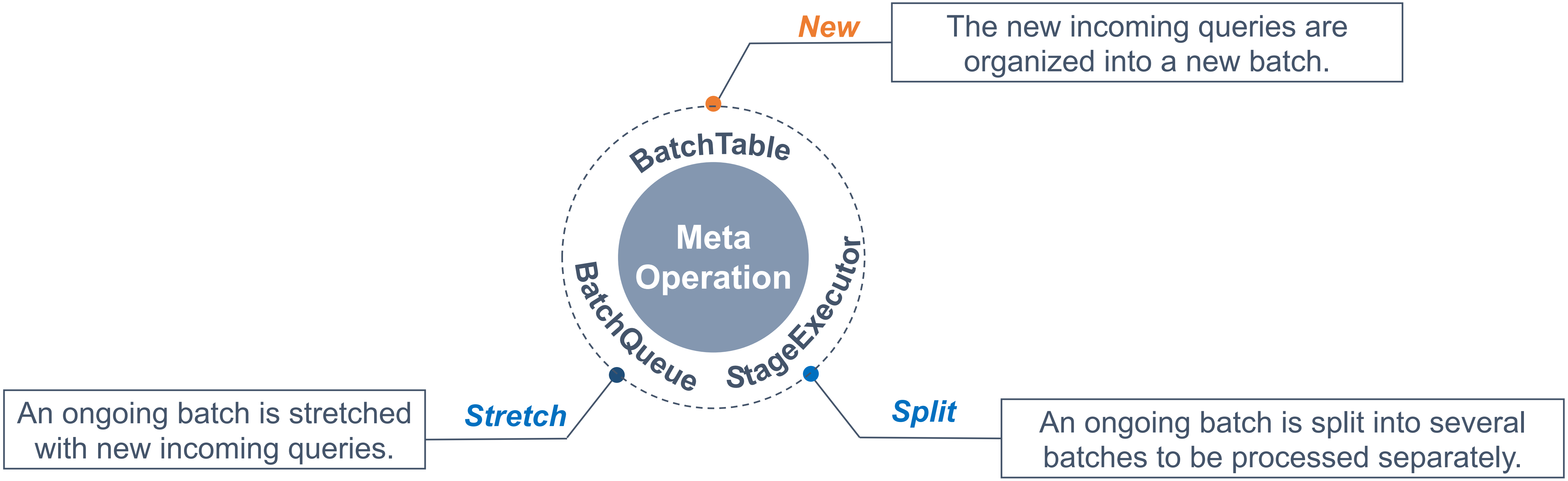
Holistic Multi-entry Multi-exit Scheme

- Key abstraction: meta operations to adjust an on-going batch



Holistic Multi-entry Multi-exit Scheme

- Key abstraction: meta operations to adjust an on-going batch

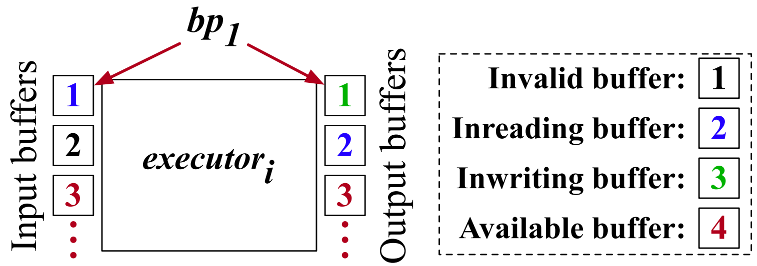


Management of stage executors

- Process with multiple buffer pairs

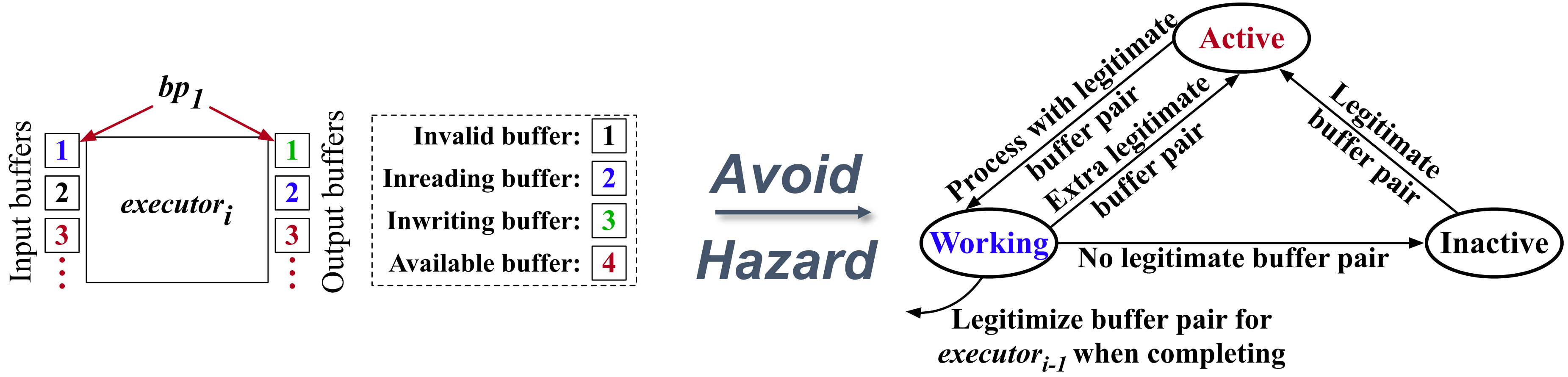
Management of stage executors

- Process with multiple buffer pairs



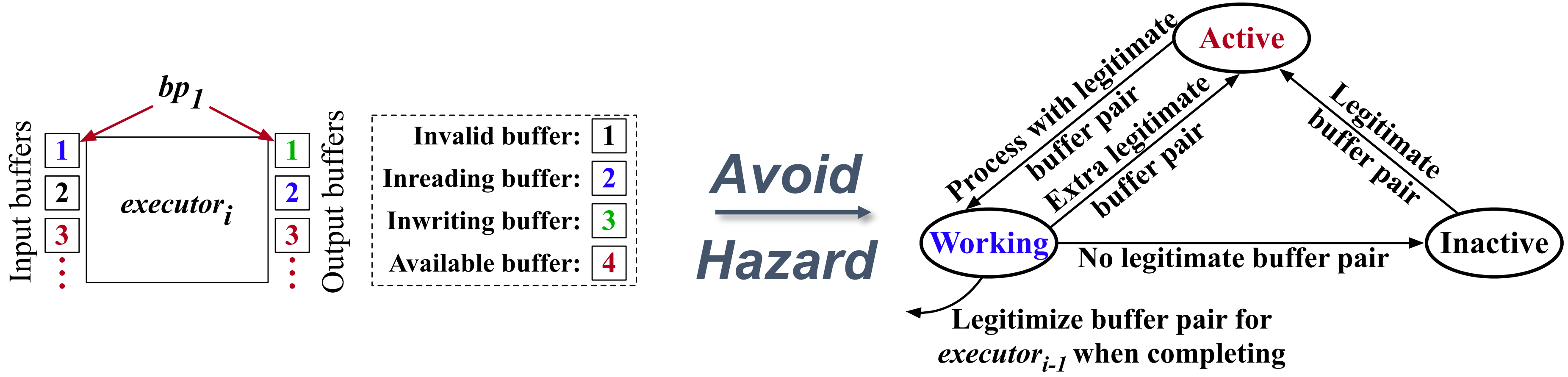
Management of stage executors

- Process with multiple buffer pairs



Management of stage executors

- Process with multiple buffer pairs



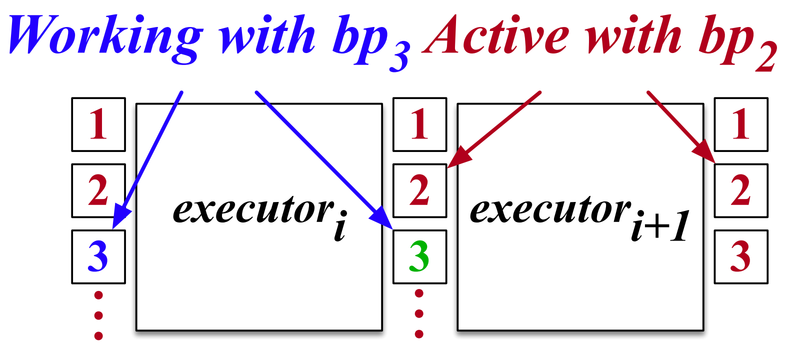
Work well for the single-entry single-exit pipeline

Management of stage executors

- Validity problem for the multi-entry multi-exit pipeline.

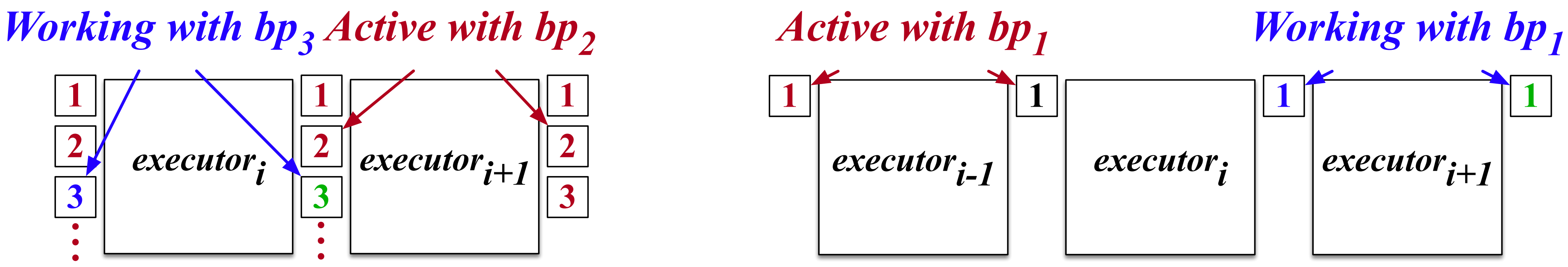
Management of stage executors

- Validity problem for the multi-entry multi-exit pipeline.
  - Scrambled access order due to Stretch and Split



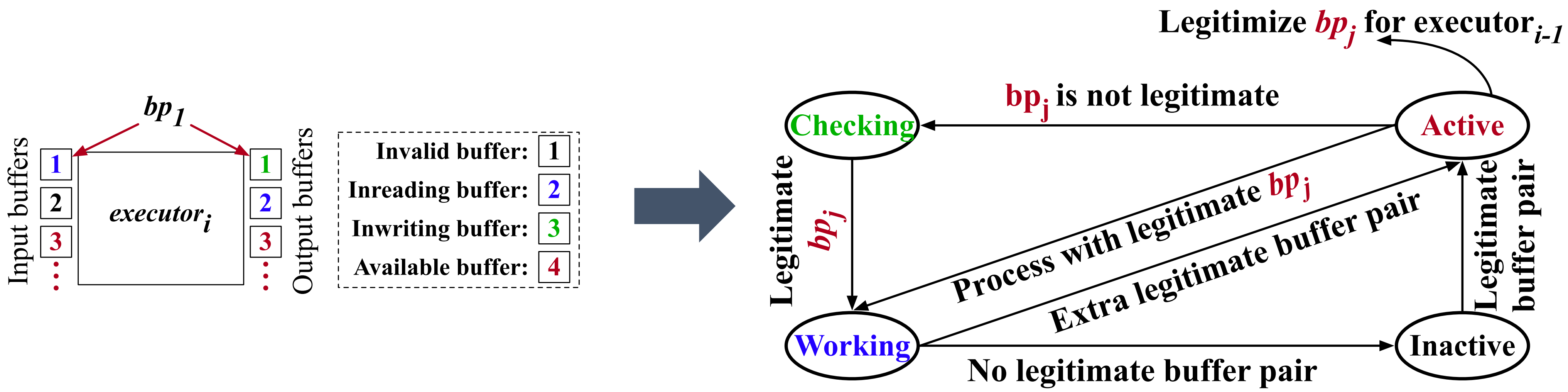
Management of stage executors

- Validity problem for the multi-entry multi-exit pipeline.
  - Scrambled access order due to Stretch and Split
  - Work in parallel even with one buffer pair



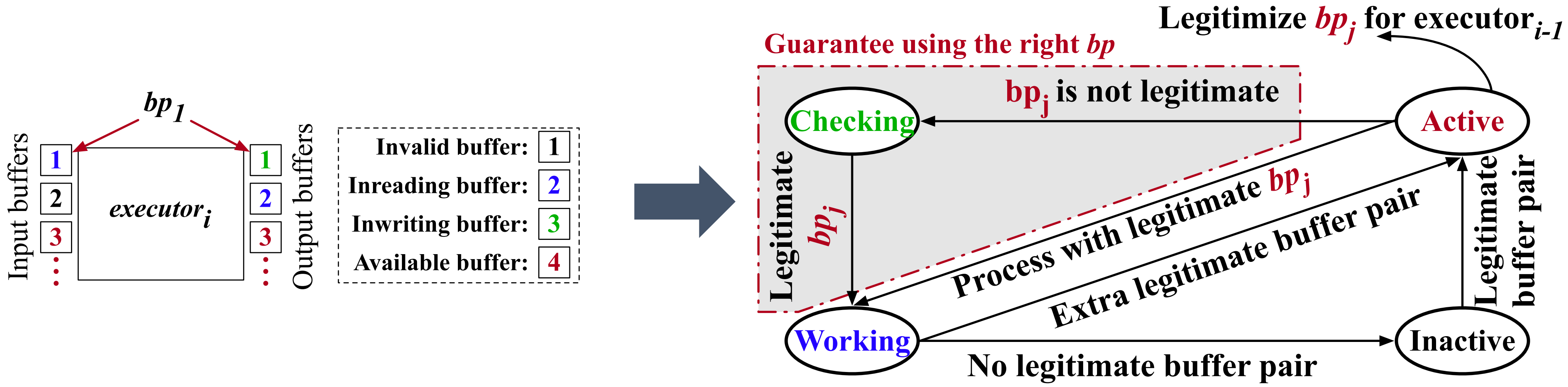
Management of stage executors

- Manage stage executors with state transition rules.
  - Process with multiple buffer pairs without hazard.
  - Ensure correct access order of buffer pairs.
  - Support switching between serial and parallel work manner.



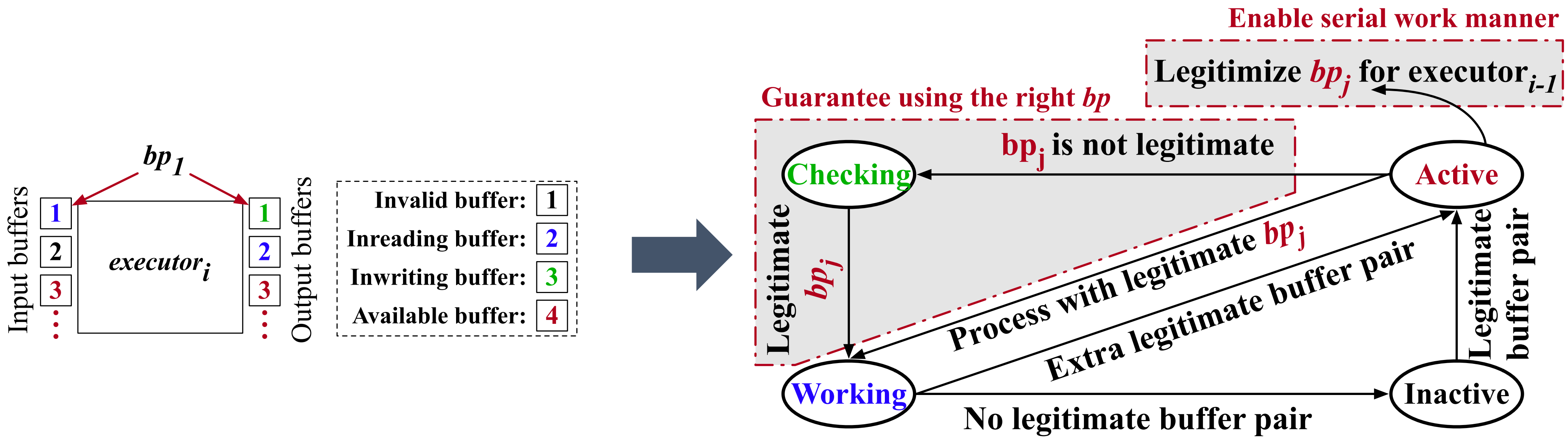
Management of stage executors

- Manage stage executors with state transition rules.
  - Process with multiple buffer pairs without hazard.
  - Ensure correct access order of buffer pairs.
  - Support switching between serial and parallel work manner.



Management of stage executors

- Manage stage executors with state transition rules.
  - Process with multiple buffer pairs without hazard.
  - Ensure correct access order of buffer pairs.
  - Support switching between serial and parallel work manner.



Interface for Creating Policies

```
//stage executors run within a while loop
void Run():
  Batch& inBatch = BatchQueue.Get();
  CheckBuffer(inBatch);//Check buffer pair
  Execute(inBatch);
  Schedule(inBatch, outBatches);
  for (auto& batch : outBatches):
    nextBatchQueue.Push(batch)
    getBuffer(); //get legitimate buffer pair
  updatePrExecutor();//update preceding executor
//call Schedule to perform meta operations
void Schedule(Batch& inBatch, vector<Batch>&
    outBatches):
  BatchTable.update(inBatch);
  if userDefined1:
    outBatches = BatchTable.New(inBatch);
  else if userDefined2:
    outBatches = BatchTable.Stretch(inBatch);
  else if userDefined3:
    outBatches = BatchTable.Split(inBatch);
  else:
    outBatches.Copy(inBatch);
```

Interface for Creating Policies

```
//stage executors run within a while loop
void Run():
  Batch& inBatch = BatchQueue.Get();
  CheckBuffer(inBatch);//Check buffer pair
  Execute(inBatch);
  Schedule(inBatch, outBatches);
  for (auto& batch : outBatches):
    nextBatchQueue.Push(batch)
    getBuffer(); //get legitimate buffer pair
  updatePrExecutor();//update preceding executor
//call Schedule to perform meta operations
void Schedule(Batch& inBatch, vector<Batch>&
     outBatches):
  BatchTable.update(inBatch);
  if userDefined1:
    outBatches = BatchTable.New(inBatch);
  else if userDefined2:
    outBatches = BatchTable.Stretch(inBatch);
  else if userDefined3:
    outBatches = BatchTable.Split(inBatch);
  else:
    outBatches.Copy(inBatch);
```

- Executors continue to execute the accepted batches and push new batches into the next batch queue.

Interface for Creating Policies

```
//stage executors run within a while loop
void Run():
  Batch& inBatch = BatchQueue.Get();
  CheckBuffer(inBatch);//Check buffer pair
  Execute(inBatch);
  Schedule(inBatch, outBatches);
  for (auto& batch : outBatches):
    nextBatchQueue.Push(batch)
    getBuffer(); //get legitimate buffer pair
  updatePrExecutor();//update preceding executor
//call Schedule to perform meta operations
void Schedule(Batch& inBatch, vector<Batch>&
     outBatches):
  BatchTable.update(inBatch);
  if userDefined1:
    outBatches = BatchTable.New(inBatch);
  else if userDefined2:
    outBatches = BatchTable.Stretch(inBatch);
  else if userDefined3:
    outBatches = BatchTable.Split(inBatch);
  else:
    outBatches.Copy(inBatch);
```

- Executors continue to execute the accepted batches and push new batches into the next batch queue.
- Users can define conditions to adjust the on-going batch with three meta operations.

Policies for Three Diversities

- Input Diversity: split the accepted batch into small batches with similar sequence length and run them in parallel.

- Operator Diversity: split the accepted batch at specific stage and run them in serial.

- Load Diversity: stretch the insufficient batch with newly arrived queries.

03

Evaluation

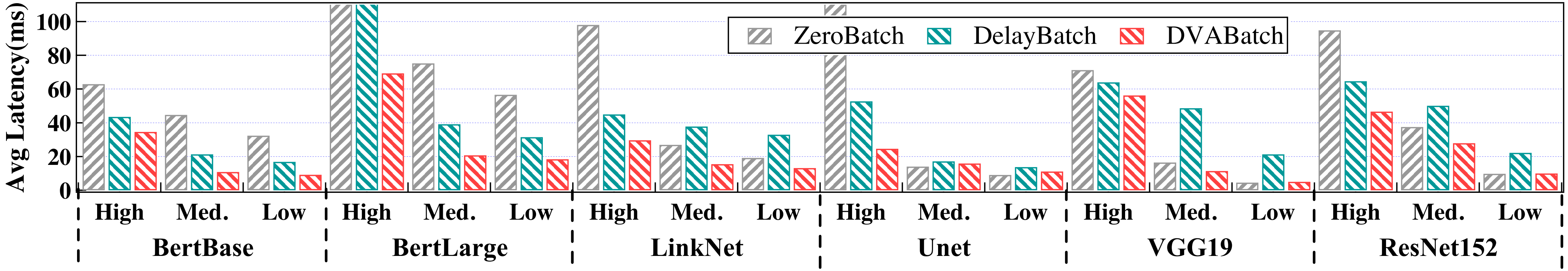
Experiment Setup

- Benchmarks:
  - BertBase, BertLarge: input and load diversity.
  - Unet, LinkNet: operator and load diversity
  - VGG19, ResNet152: load diversity
- Load Generating:
  - Arrival pattern: Poisson distribution
  - Input pattern: GLUE dataset

Hardware	CPU: Intel Xeon E5-2620, GPU: Nvidia Titan RTX
OS & Driver	Ubuntu: 18.04.6 (kernel 4.15.0); GPU Driver: 470.57
Software	CUDA: 11.4; TensorRT: 8.03; Triton 21.10
Benchmarks	Unet [56]; LinkNet [16];BertBase; BertLarge [27]; VGG19 [57]; ResNet152 [37]
Dataset	GLUE [59]

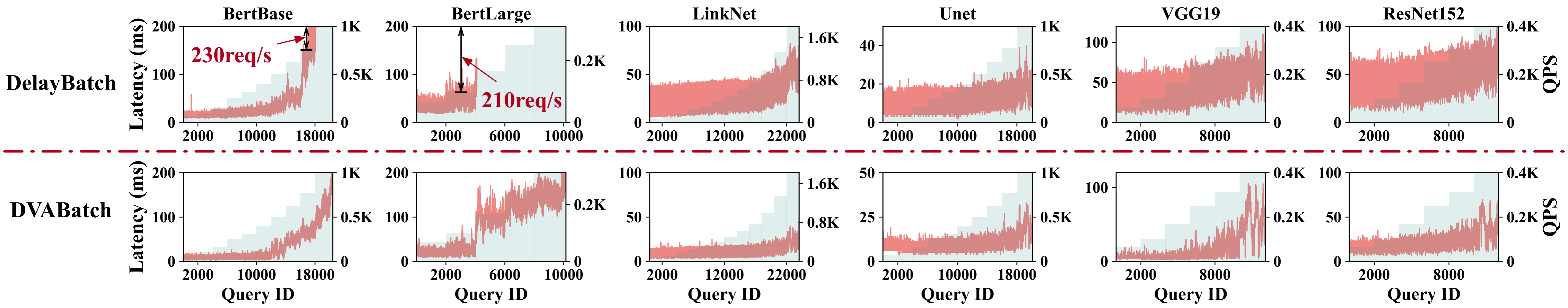
End to End Results

- Baselines:
  - ZeroBatch: batching with no extra time window.
  - DelayBatch: batching with a hand-tuned time window.



- Compared with ZeroBatch: 16.1%/39.0%/57.7% average latency reduction under high, medium, and low load.
- Compared DelayBatch: 35.4%/47.3%/48.5% average latency reduction under high, medium, and low load.

Robustness at Stepping Load



- All the benchmarks have lower latency with DVABatch than with DelayBatch in all cases.
- DVABatch increases 46.81% peak throughput for BertBase, 1.37× peak throughput for BertLarge.

04

Conclusion

Contributions

- We propose a multi-entry multi-exit batching scheme for efficient DNN service processing on GPUs.
- We provide a general scheduling mechanism that leverages meta operations, and state transition diagram to create policies for different serving diversities.
- We reduce 46.4% average latency and achieve up to 2.12× throughput improvement for the involved serving diversities.

Implications

- Omnipresent Diversity.
  - Existing DNNs may have a dynamic architecture in depth, width, and routing. As more dynamic attributes emerge, diversity spreads across new DNNs.
  - DVABatch is flexible to tackle diversities mentioned above, like layer-skip, early-exiting models.

- Intra-model Scheduling.
  - As DNNs grow larger and show more diversity, the execution of DNNs cannot be treated as a single function call.
  - Intra-model scheduling is a trend for future DNN inference of large models.

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

Thanks! Q&A

Weihao Cui: weihao@sjtu.edu.cn

饮 水 思 源 爱 国 荣 校