

中山大學
SUN YAT-SEN UNIVERSITY

USENIX ATC '25

Obscura: Concealing Recomputation Overhead in Training of Large Language Models with Bubble-filling Pipeline Transformation

Yuzhou Huang[1], Yapeng Jiang[1], Zicong Hong[2], Wuhui Chen[1], Bin Wang[3],

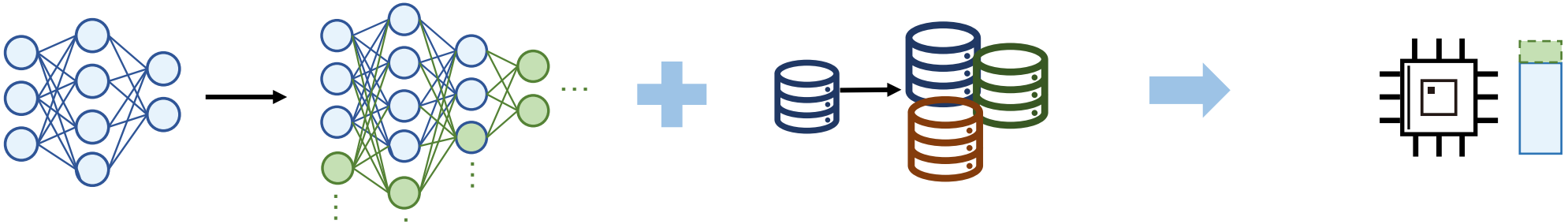
Weixi Zhu[3], Yue Yu[4], Zibin Zheng[1]

[1]Sun Yat-sen University
[2]Hong Kong University of Science and Technology
[3]Huawei Technologies
[4]Peng Cheng Laboratory

Contact: huangyzh57@mail2.sysu.edu.cn

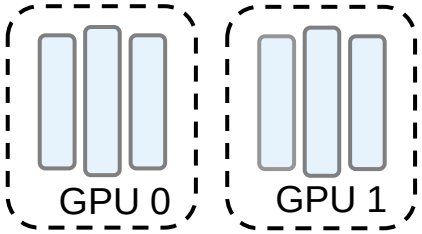
Motivation

- **Parallelism Strategy**
  - The training of large language models exceeds the memory capacity of a single GPU.

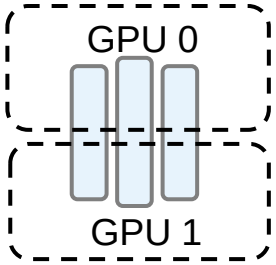


The increasing model size | The expanding dataset | Single-GPU memory limitation

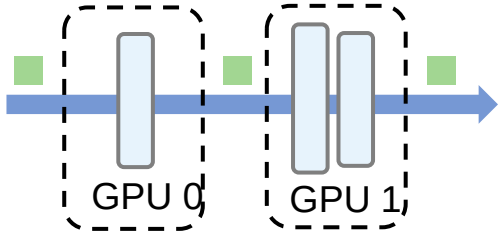
  - Adopting coordinated multi-GPU parallelism to distribute computation across multiple GPUs.



Data parallelism



Tensor parallelism

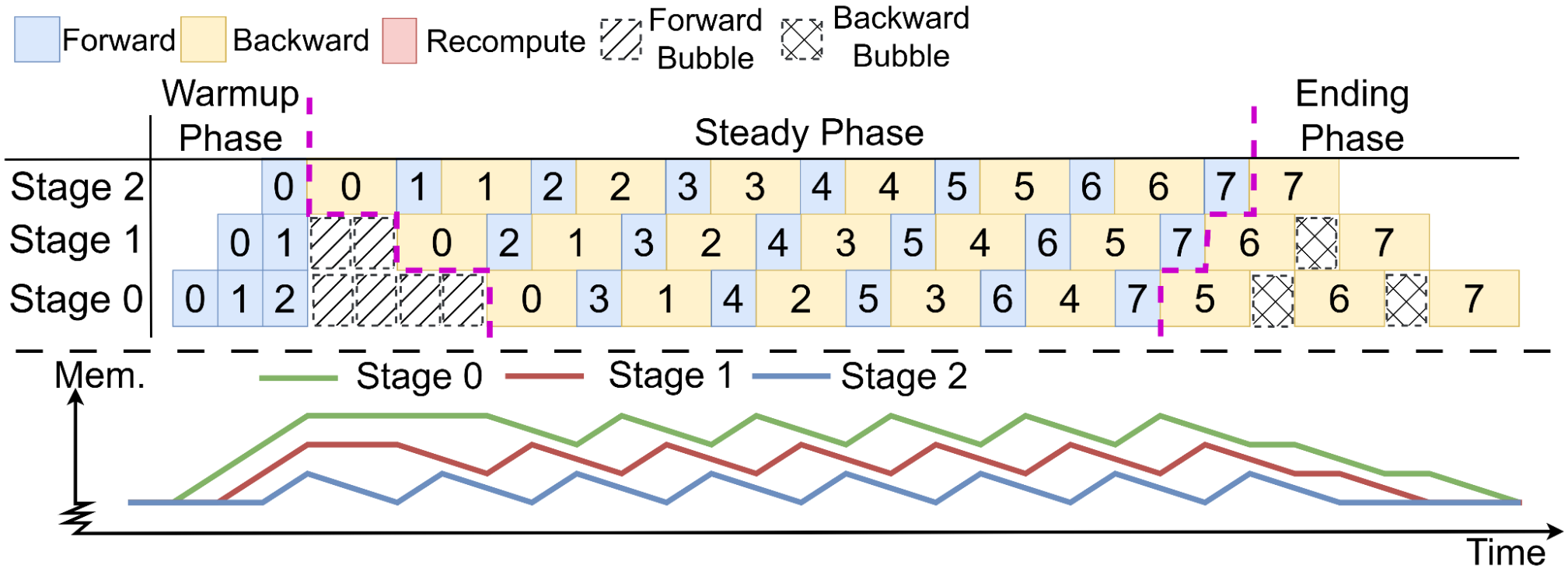


Pipeline parallelism

Pipeline parallelism stands out!

Motivation

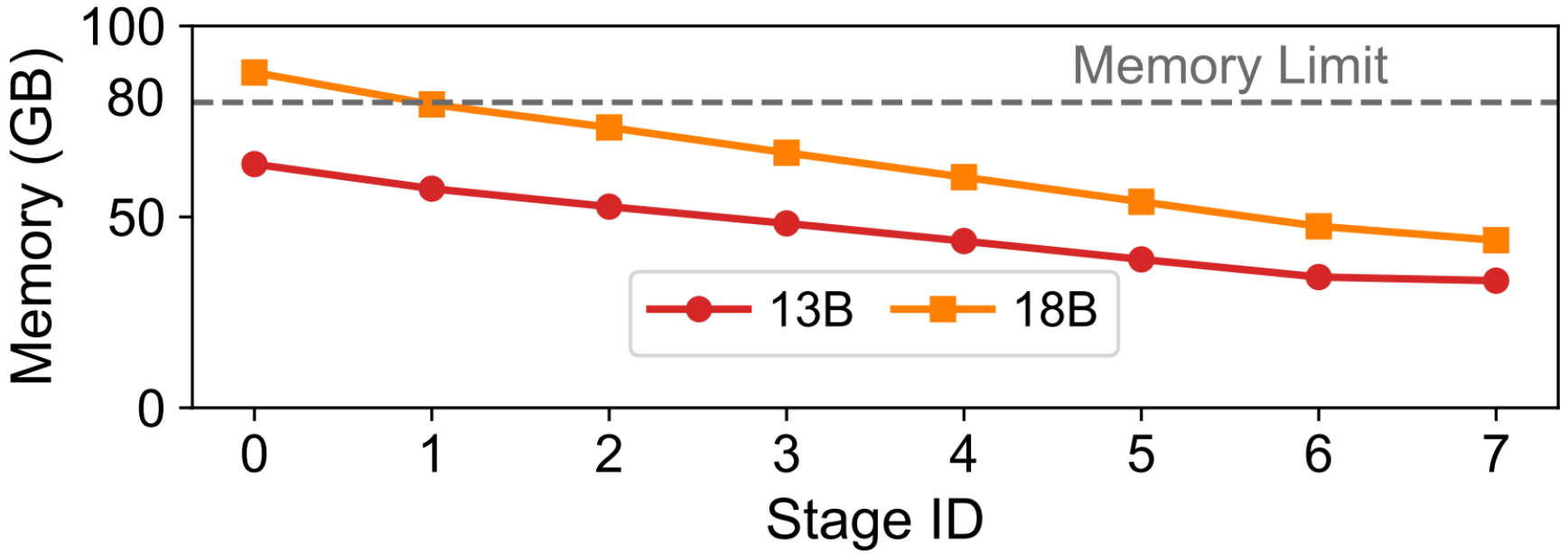
- **1F1B Pipeline is popular**



- Additionally, we define two types of bubbles: **forward bubbles** (diagonal blocks), occurring between the last forward and first backward passes, **and backward bubbles** (crossed-line blocks), occurring between the first and last backward passes.

Motivation

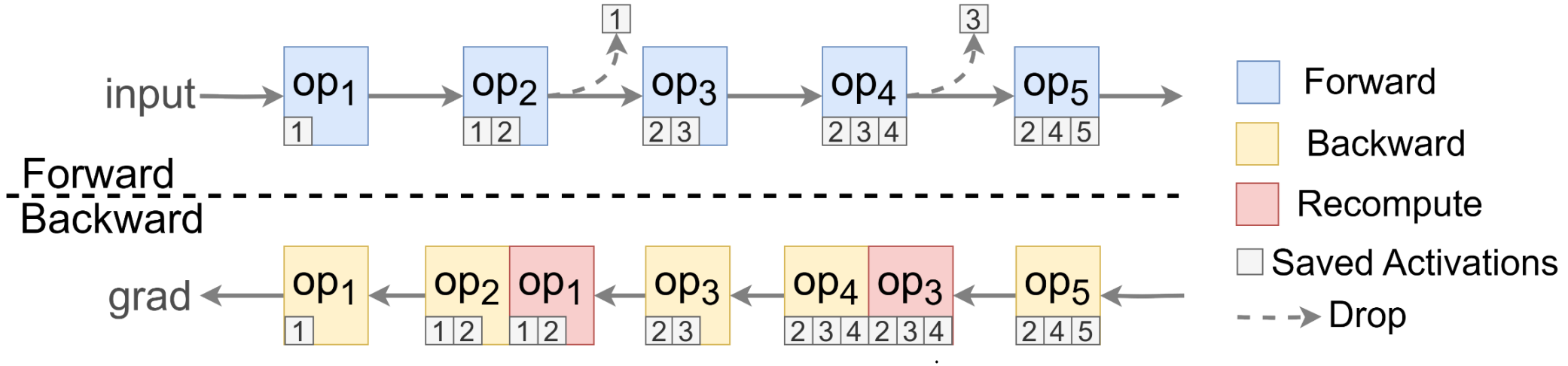
- **1F1B Pipeline is popular, but it faces a memory bottleneck in the early stages**



➢ Stage s out of p stages must store activations for up to $p - s$ micro-batches. As a result, earlier stages ($s \rightarrow 0$) experience higher memory usage.

Motivation

- **Recomputation is a direct and effective method.**



- Although this approach effectively lowers memory consumption, ***it introduces additional computation overhead!***

Motivation

- **Recomputation is a direct and effective method.**

	Non	Full	Self-Attn	CMB	MLP
Memory (GB)	63.83 (100%)	31.65 (-50%)	62.50 (-2%)	48.81 (-23%)	41.86 (-34%)
Iteration Time (s)	9.23 (100%)	12.20 (+32%)	9.49 (+3%)	9.53 (+3%)	10.60 (+15%)

➢ Although this approach effectively lowers memory consumption, ***it introduces additional computation overhead!***

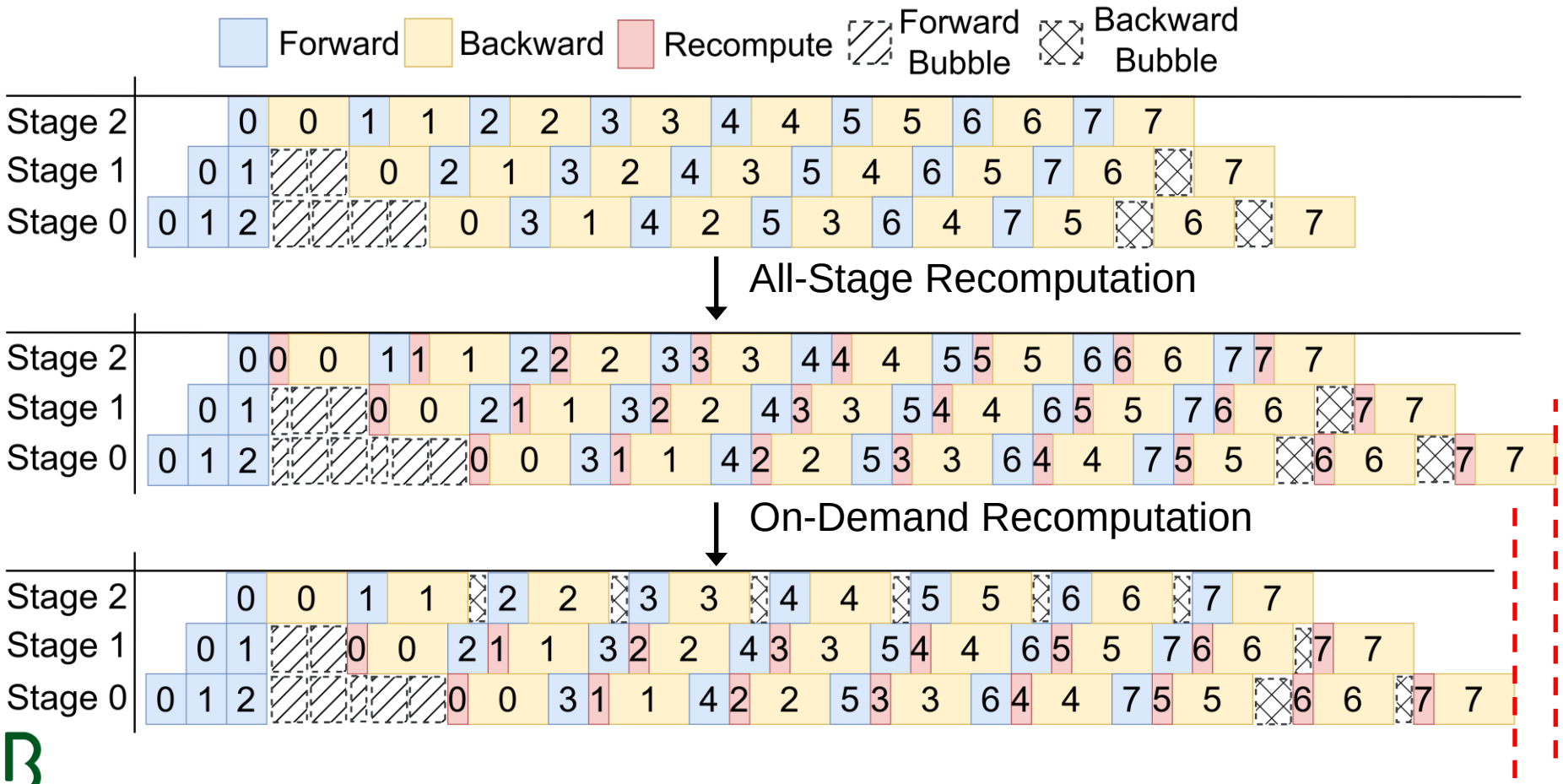
[1] Korthikanti, Vijay Anand, et al. "Reducing activation recomputation in large transformer models." *Proceedings of Machine Learning and Systems* 5 (2023): 341-353.

[2] Yuan, Tailing, et al. "Accelerating the training of large language models using efficient activation rematerialization and optimal hybrid parallelism." *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 2024.

Motivation

Observation

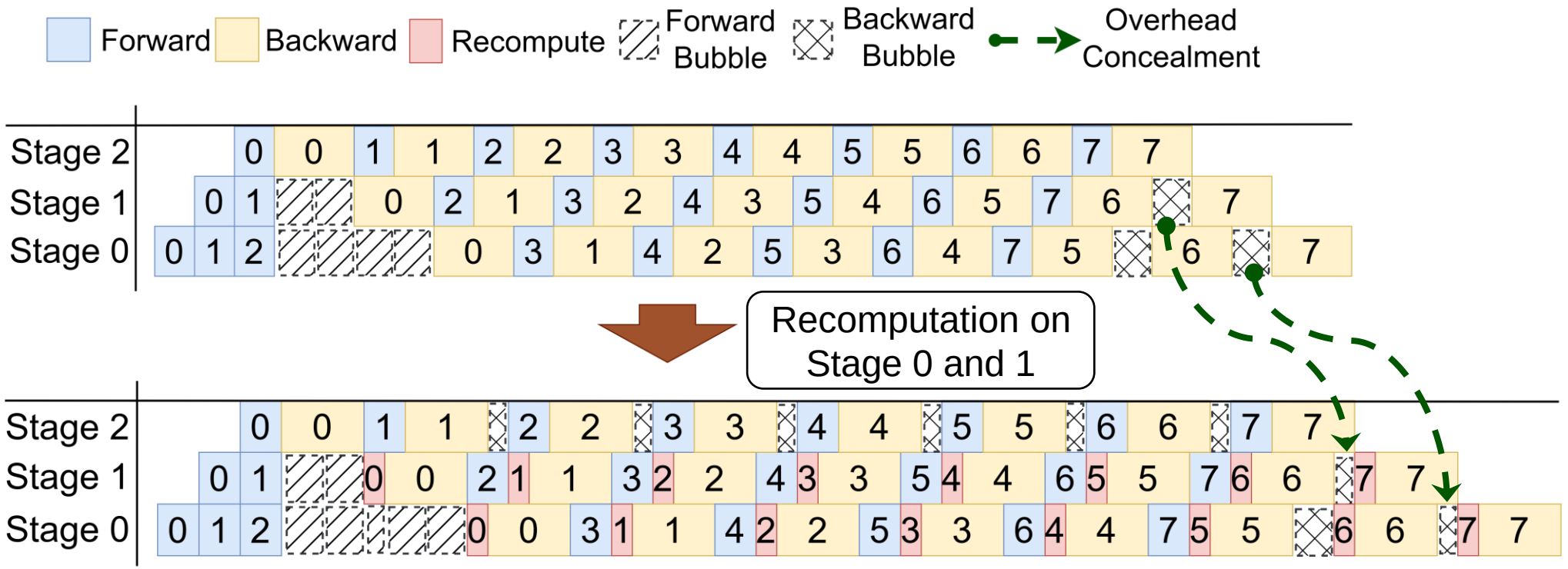
- On-Demand Recomputation Achieves Better Performance than All-Stage Recomputation.



Motivation

Observation

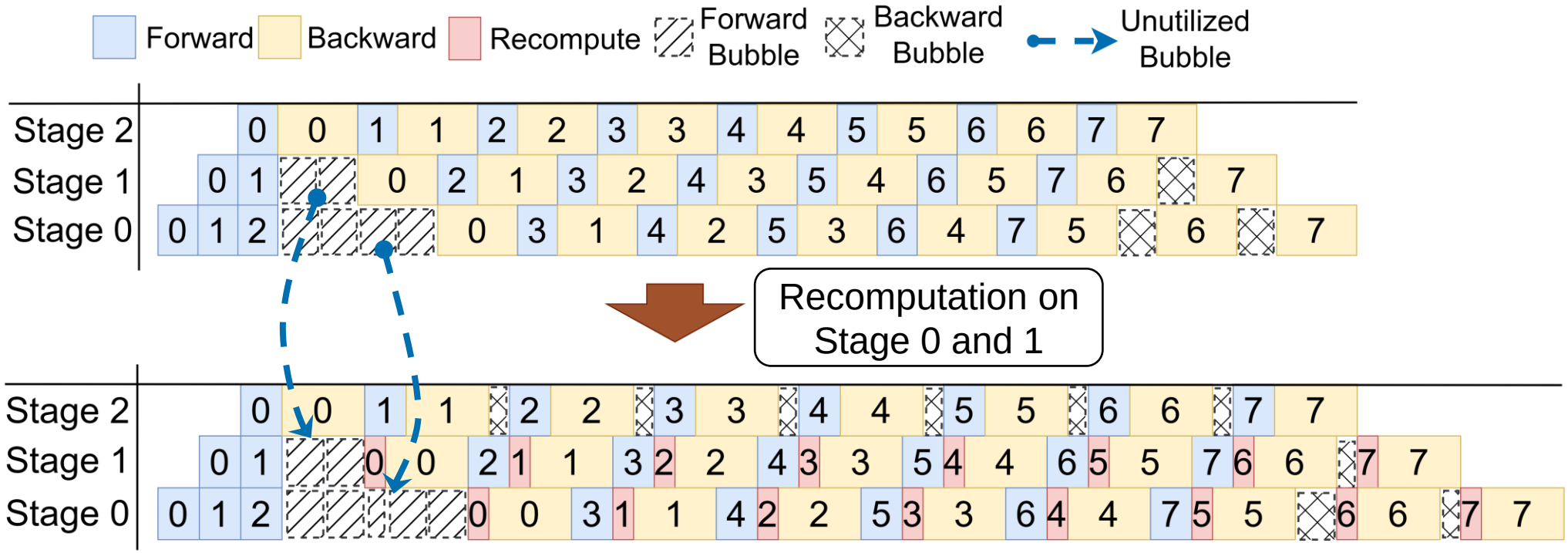
- On-Demand Recomputation Achieves Better Performance than All-Stage Recomputation.
- Backward Bubbles Can Conceal Recomputation Overhead.



Motivation

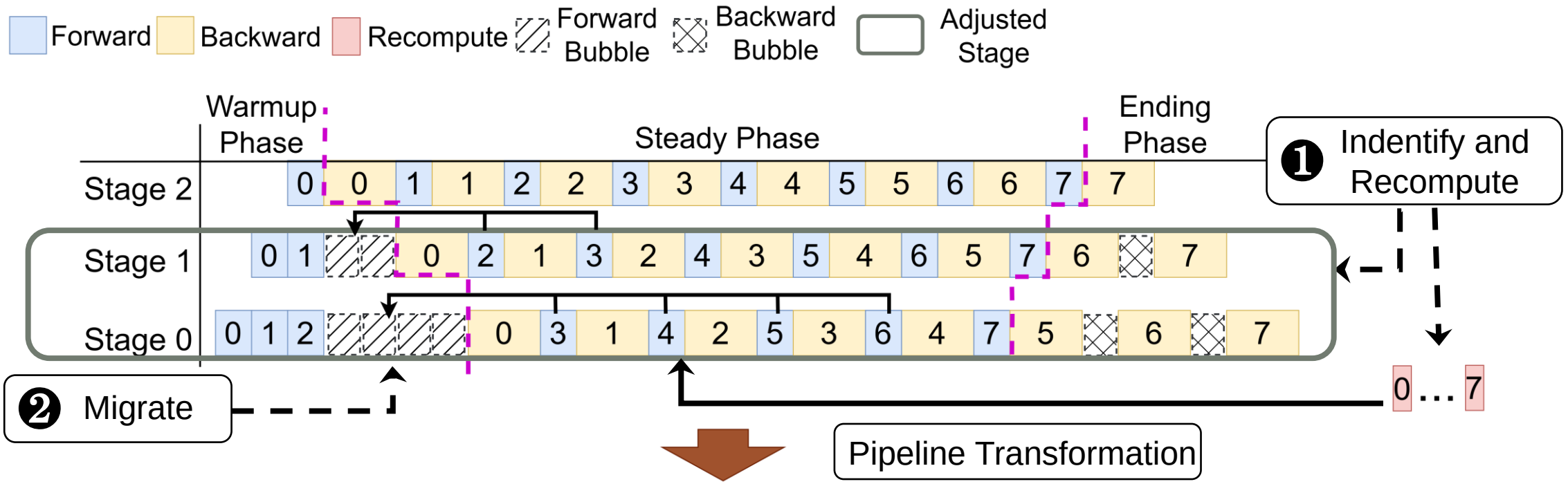
Observation

- On-Demand Recomputation Achieves Better Performance than All-Stage Recomputation.
- Backward Bubbles Can Conceal Recomputation Overhead.
- Forward Bubbles Are Unutilized.



Motivation

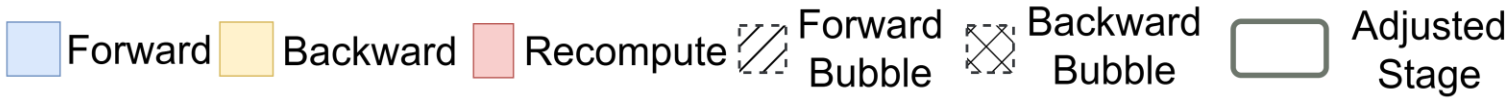
- **Strawman**



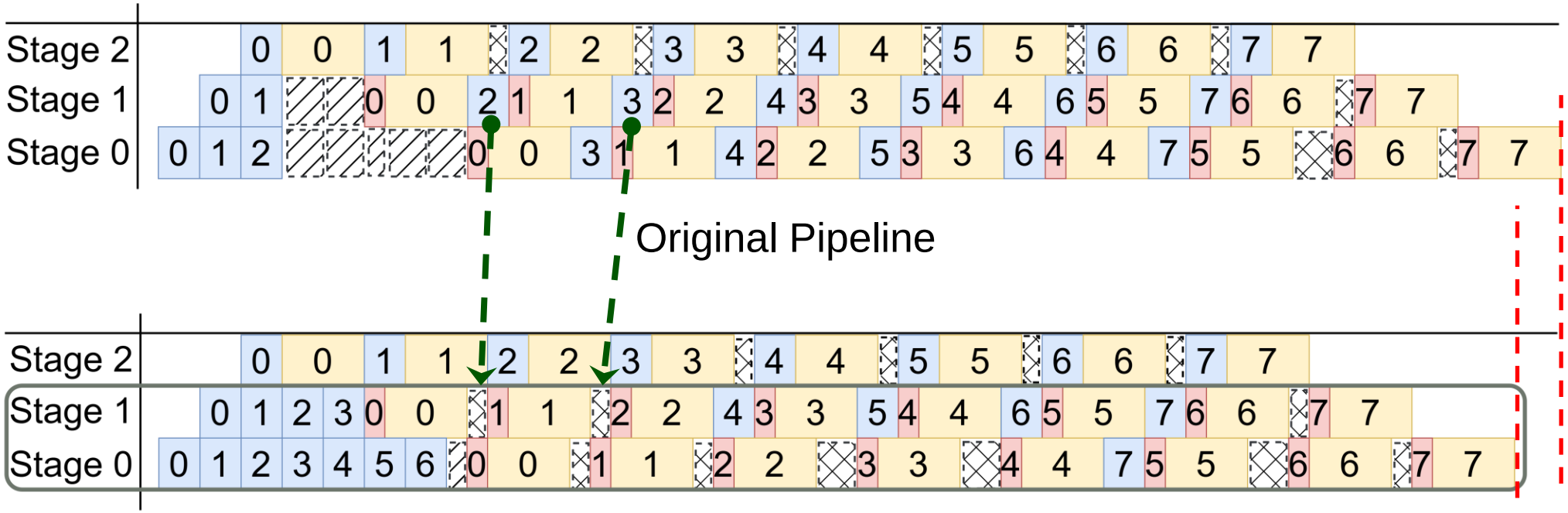
❶ Identify stages that exceed the memory limit as adjusted stages and apply recomputation to them.
❷ Migrate the forward passes in the steady phase of adjusted stages leftward into forward bubbles as much as possible.

Motivation

- **Strawman**



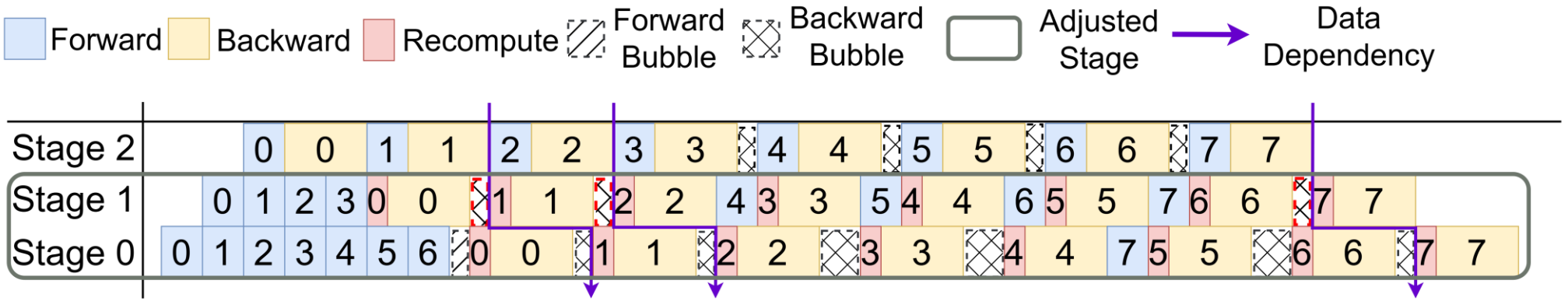
The newly introduced backward bubbles effectively hide the recomputation overhead!



Strawman Pipeline

Motivation

- **Challenges**

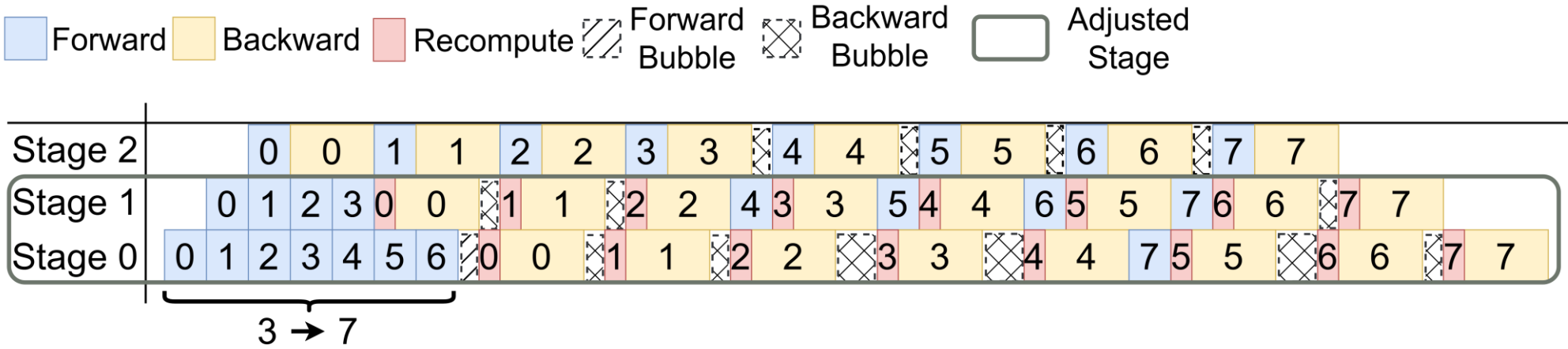


➢ How to Fully Utilize Backward Bubbles?

Tight dependencies between adjusted and non-adjusted stages limit backward bubbles from hiding recomputation.

Motivation

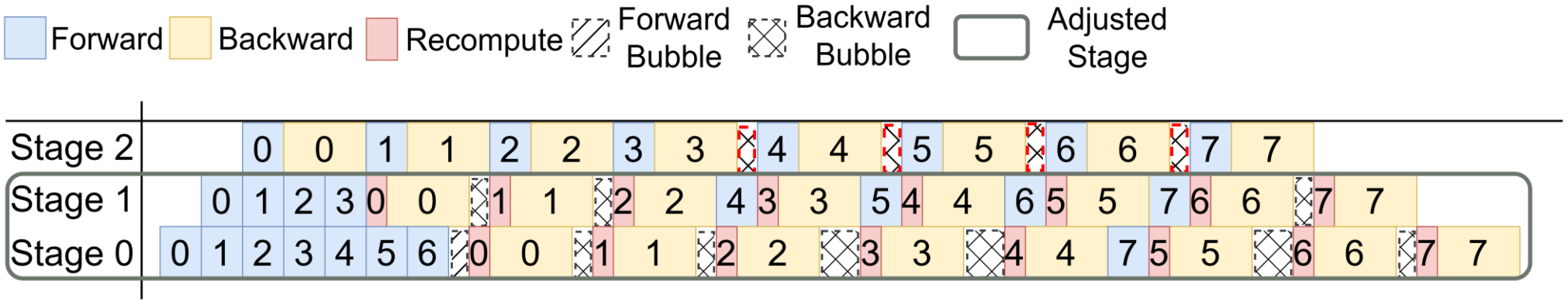
- **Challenges**



- How to Fully Utilize Backward Bubbles?
  - Tight dependencies between adjusted and non-adjusted stages limit backward bubbles from hiding recomputation.
- How to Reduce Extra Memory Usage?
  - Migration increases the number of forward passes required by adjusted stages during the warmup phase.

Motivation

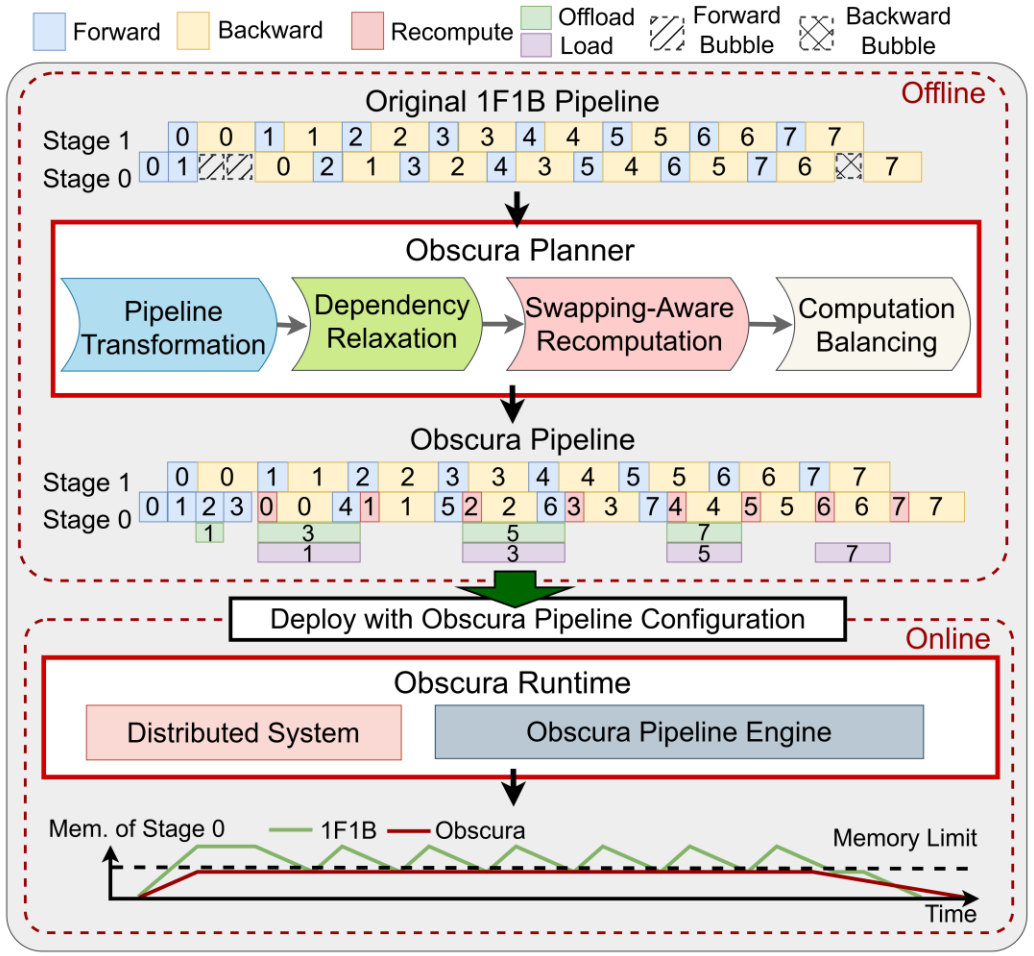
- **Challenges**



- How to Fully Utilize Backward Bubbles?
  - Tight dependencies between adjusted and non-adjusted stages limit backward bubbles from hiding recomputation.
- How to Reduce Extra Memory Usage?
  - Migration increases the number of forward passes required by adjusted stages during the warmup phase.
- How to Address the Computation Imbalance?
  - On-demand recomputation introduces workload imbalances between adjusted and non-adjusted stages.

Obscura

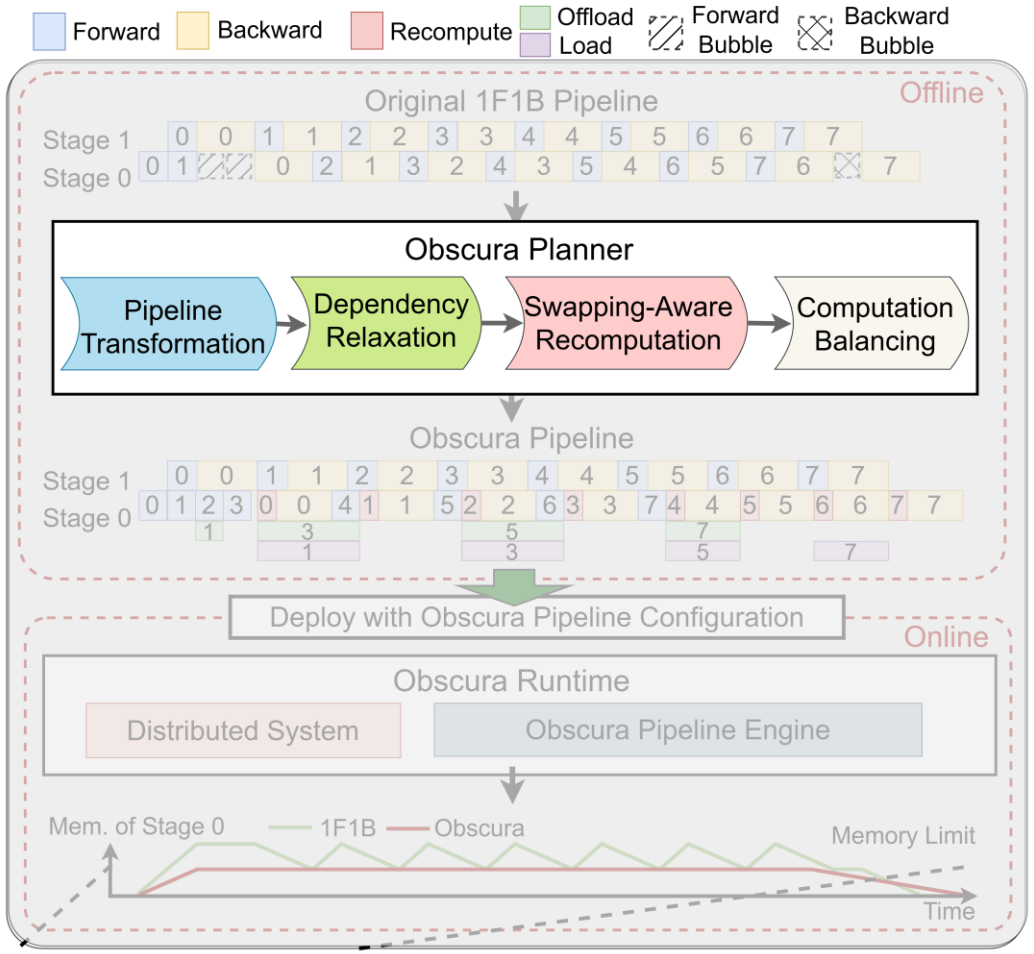
Overview



Overview of Obscura.

Obscura

Overview

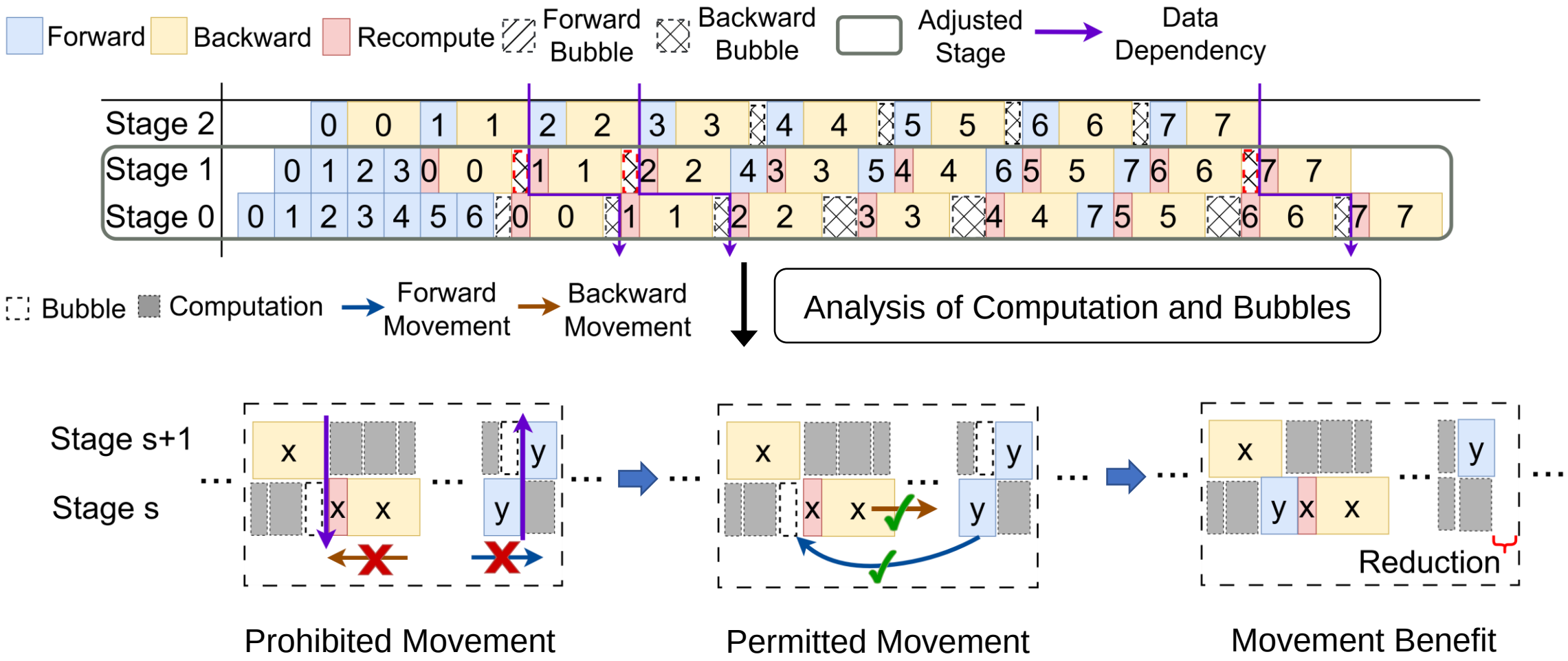


Overview of Obscura.

1. Pipeline Transformation Module
2. Dependency Relaxation Module
3. Swapping-Aware Recomputation Module
4. Partition Adjustment Module

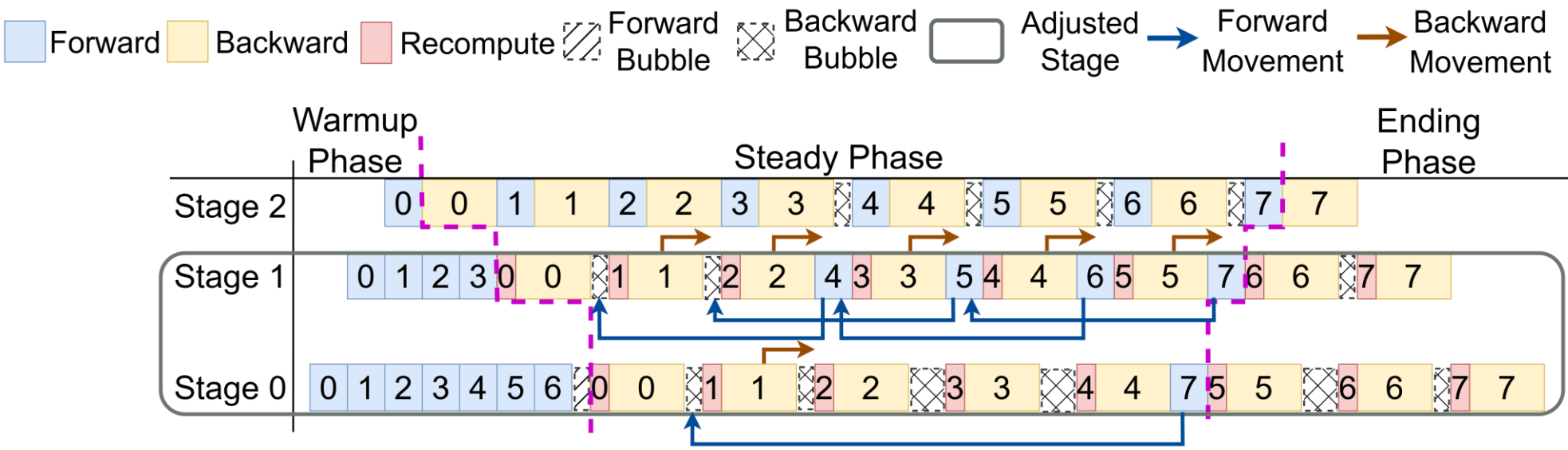
Obscura

Dependency Relaxation



Obscura

Dependency Relaxation

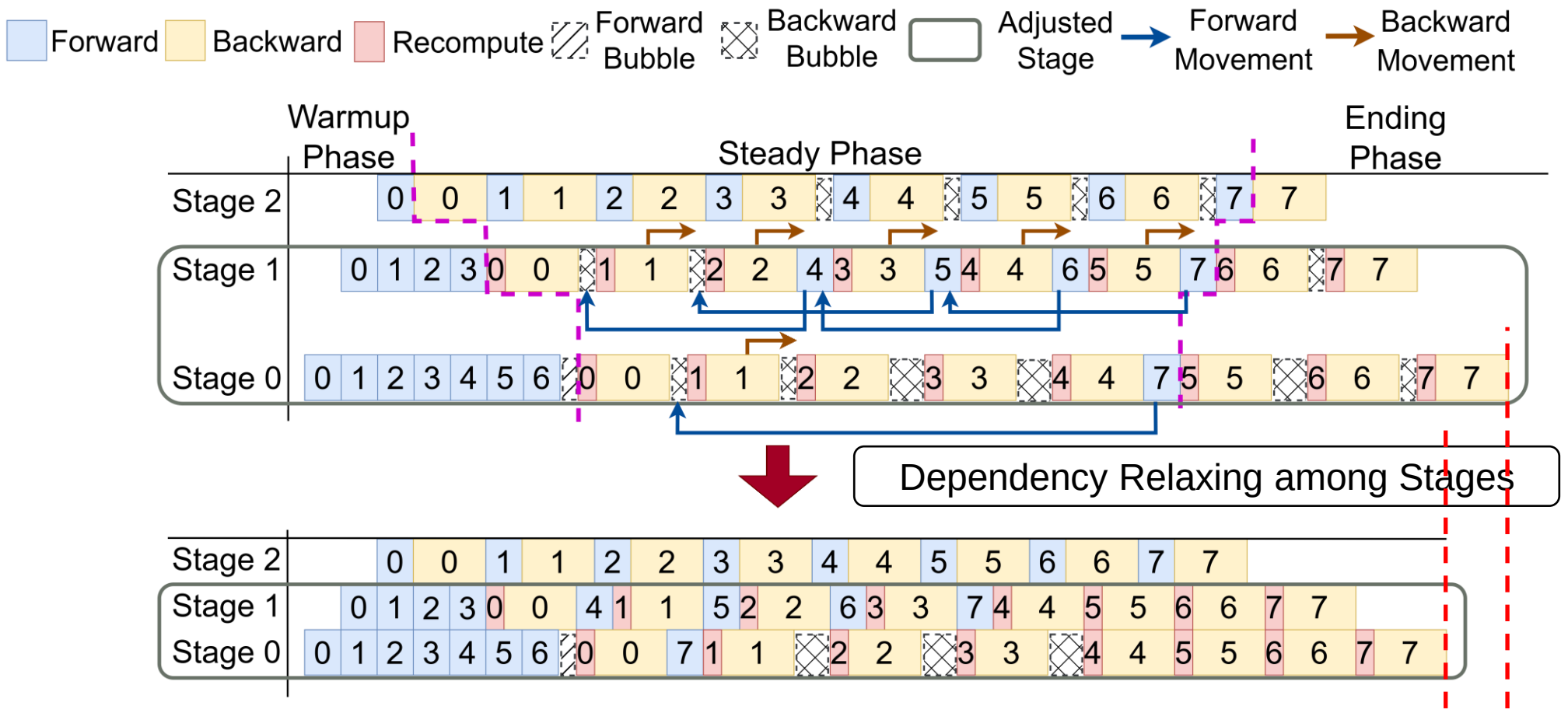


Step 1: Shift the remaining forward passes in the steady phase leftward, interleaving them with backward passes.

Step 2: Shift the backward passes rightward to fill the bubbles left by the migration forward passes.

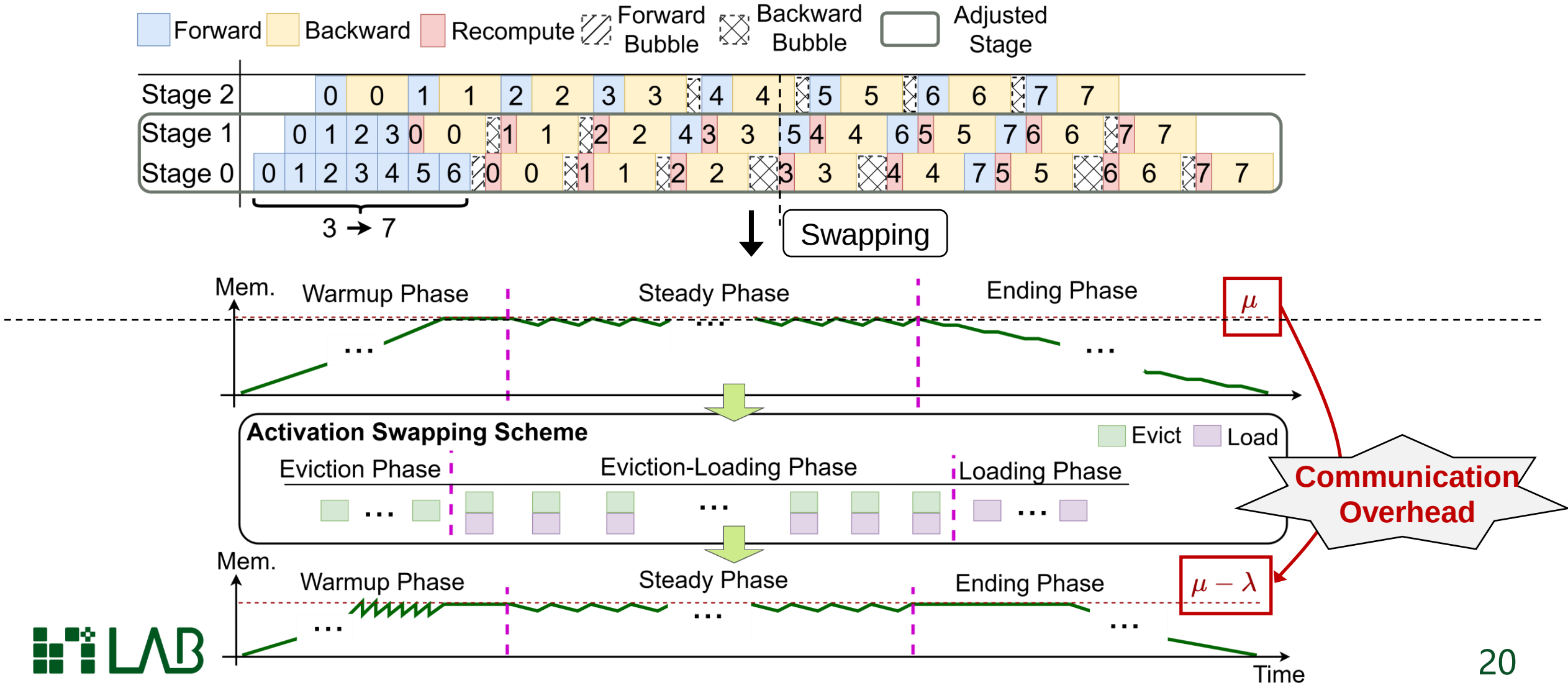
Obscura

Dependency Relaxation



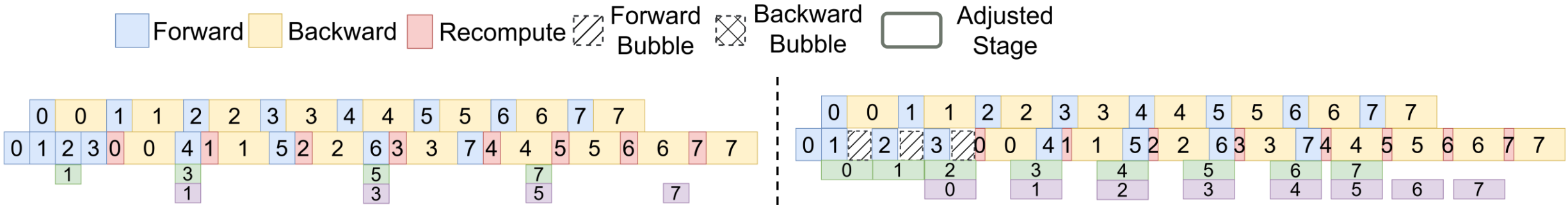
Obscura

- **Swapping-Aware Recomputation**



Obscura

Swapping-Aware Recomputation



Execution Time:

$$T_s = \begin{cases} 3mf + 3(p-s-1)f, & s > k \\ \max(T_{s+1} + \gamma, \gamma m + (k-s)\gamma + T_s^C), & s \le k \end{cases}$$

Memory Usage:

$$M_s = W_s + A_s \qquad A_s = \begin{cases} (p-s) \times a, & s > k \\ A_s^{adj}, & s \leqslant k \end{cases}$$

Recomputation Strategy:

$$r = \sum_{i=0}^{n-1} R_{op}^{0-1}[i] \times f_{op_i}$$

$$a^{adj} = a - \sum_{i=0}^{n-1} R_{op}^{0-1}[i] \times a_{op_i}$$

Optimization Objective:

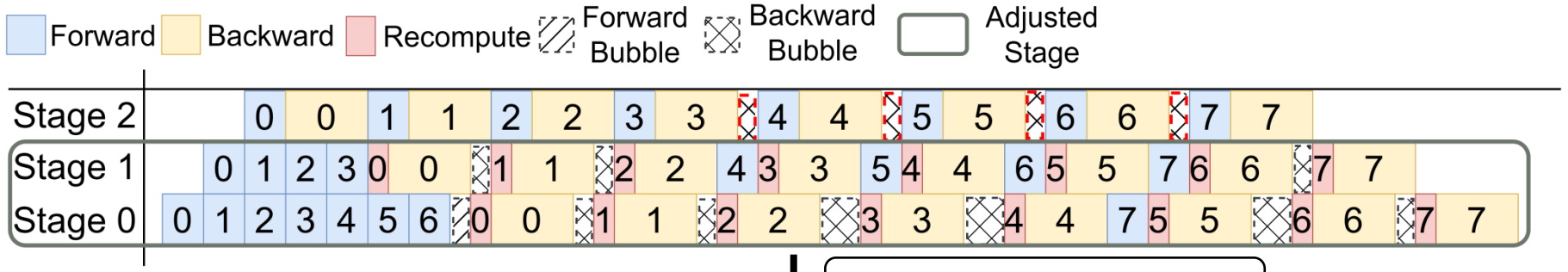
$$min\ T_0$$

$$s.t.\ M_s \le G, s \in [0, p-1]$$

$$a^{adj}\beta \le (3f + r) \times B$$

Obscura

Partition Adjustment



Partition Ajdustment

Modeling in Uneven Stage Partition:

Computation Time:

$$C_{adj} = \max_{s \le k}(3f_s + r_s)$$

$$C_{non} = \max_{s > k}(3f_s)$$

Activation Size:

$$a_s$$

Execution Time:

$$\hat{T}_s \approx \begin{cases} T_s(C_{non}), & s > k \\ T_s(C_{adj}), & s \le k \end{cases}$$

Memory Usage:

$$\hat{M}_s = M_s(a_s)$$

Algorithm 1: Partition Adjustment

Input: D_{non} and D_{adj}
Output: Updated D_{non} and D_{adj}.

Initialize $old_D_{non} \leftarrow D_{non}$, $old_D_{adj} \leftarrow D_{adj}$
Compute $\hat{T}_k$, $\hat{T}_{k+1}$, C_{adj} based on D_{non}, D_{adj}
while $\hat{T}_k - C_{adj} > \hat{T}_{k+1}$ **do**
  $old_D_{non}, old_D_{adj} \leftarrow D_{non}, D_{adj}$
  $a \leftarrow \arg\max_{s \le k}(3f_s + r_s)$
  $b \leftarrow \arg\min_{s > k \wedge \hat{M}_s \le G}(3f_s)$
  $D_{adj}[a] \leftarrow D_{adj}[a] - 1$
  $D_{non}[b] \leftarrow D_{non}[b] + 1$
  Update $\hat{T}_k$, $\hat{T}_{k+1}$, C_{adj} based on D_{non} and D_{adj}
$D_{non}, D_{adj} \leftarrow old_D_{non}, old_D_{adj}$

- Greedily transfers model layers from the slowest adjusted stage to the fastest non-adjusted stage.

Evaluation

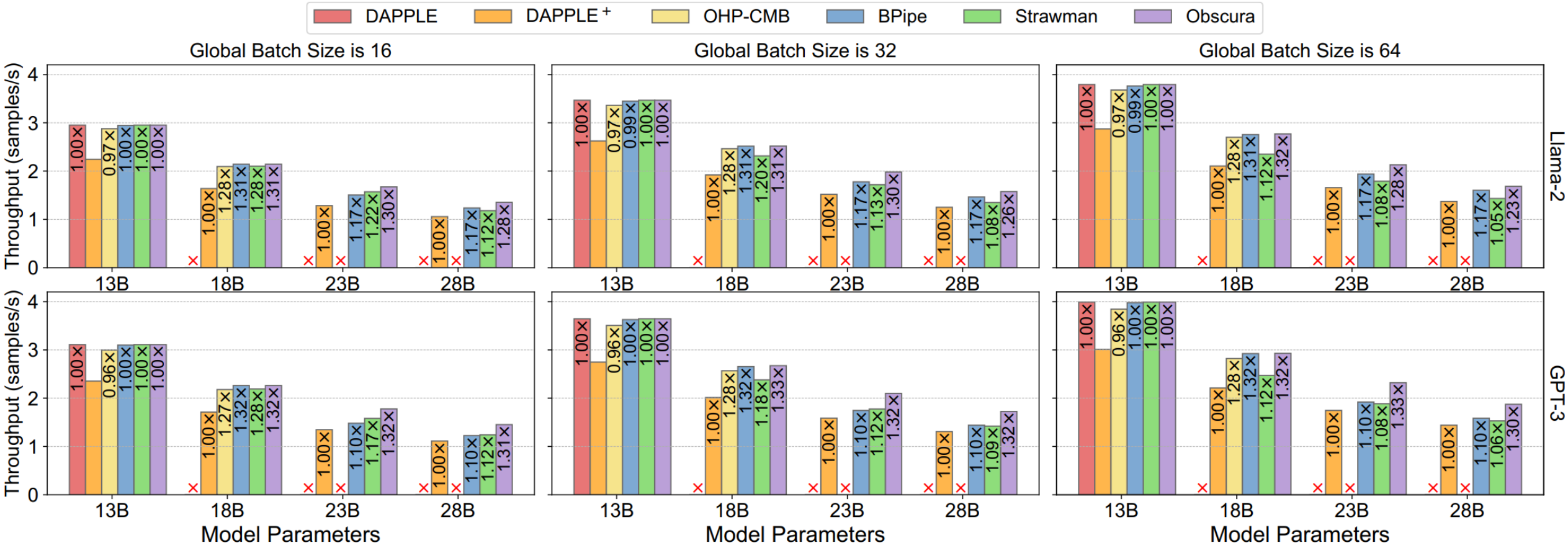
Setup

- **Hardware**: NVIDIA A100-SXM-80GB (A100*8, Fully connected NVLink 600GB/s, PCIe 4.0 x16)
- **Framework**: Deepspeed, PyTorch
- **Model**: Llama 2, GPT-3, including 13B, 18B, 23B, and 28B parameters for both base models
- **Baselines**:
  - DAPPLE: The state-of-the-art 1F1B pipeline schedule with even stage partition.
  - DAPPLE+: We evaluate DAPPLE using the *Full* recomputation strategy applied to all stages, which reduces memory usage and enables the training of larger models.
  - OHP-CMB: We apply the recomputation strategy proposed in the work [1] that drops activations of specific operators, including *Norm*, *SiLU/GELU*, and *Mul*.
  - BPipe [2]: It eliminates the memory bottleneck by transferring activations from earlier stages to later ones using an activation balancing method. As it encounters OOM issues with 23B and 28B models, we apply the *MLP* recomputation strategy for these larger models.
- **Main metric**: Throughput

[1] Yuan, Tailing, et al. "Accelerating the training of large language models using efficient activation rematerialization and optimal hybrid parallelism." *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 2024.
[2] Kim, Taebum, et al. "Bpipe: Memory-balanced pipeline parallelism for training large language models." *International Conference on Machine Learning*. PMLR, 2023.

Evaluation

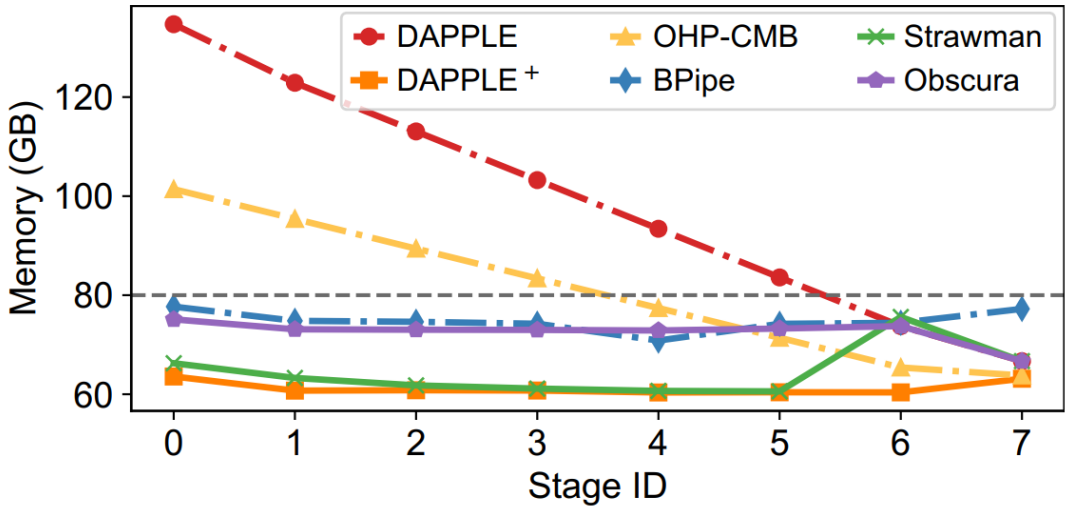
End-to-End Throughput



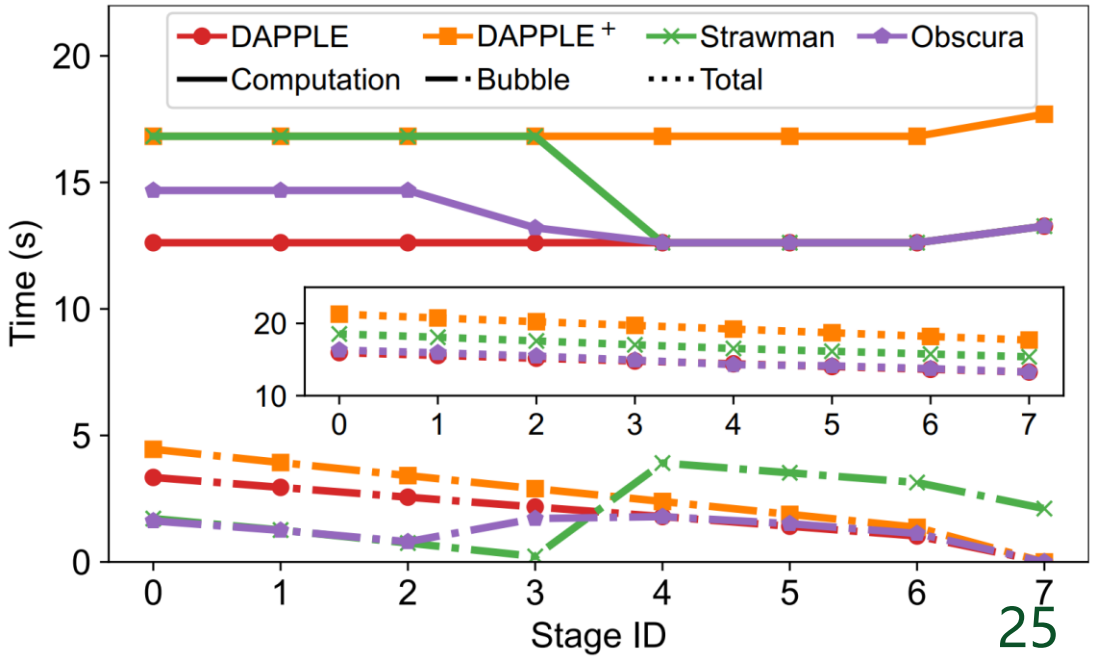
- Obscura consistently delivers leading performance compared to other methods, achieving a 22%–33% speedup over DAPPLE+ for model sizes ranging from 18B to 28B.

Evaluation

Memory, Computation and Bubble Analysis



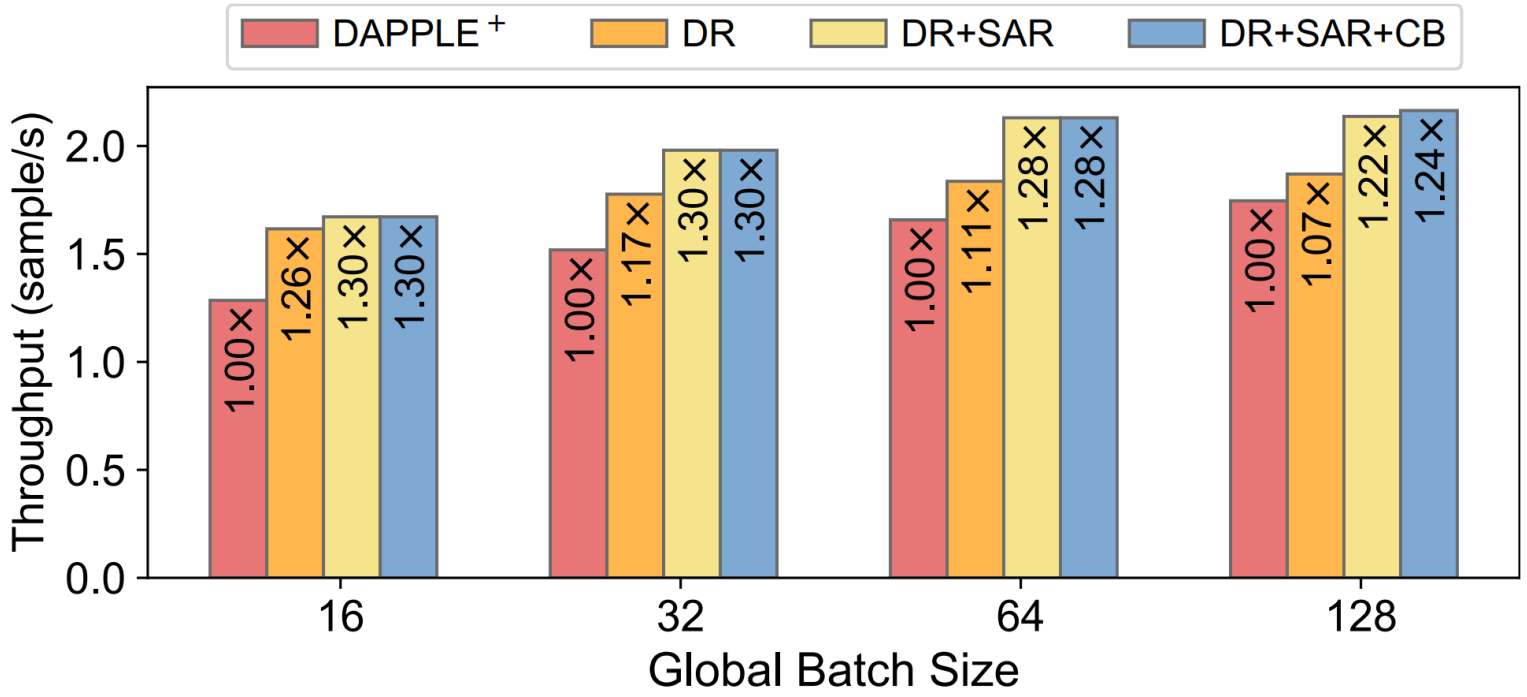
- Obscura achieves better memory efficiency than DAPPLE and OHP-CMB, reaching a memory balance comparable to BPipe.



- Obscura effectively utilizes pipeline bubbles and, with the integration of swapping, adopts a cost-efficient recomputation strategy that results in shorter computation time than DAPPLE+.

Evaluation

- **Sensitivity Analysis**



DR: Dependency Relaxation

SAR: Swapping-Aware Recomputation

CB: Computation Balancing

➢ The impact of the three key modules in Obscura designed to address the three challenges.

Conclusion

- We propose Obscura, a computationally efficient pipeline training system for LLMs that leverages pipeline transformation to better conceal recomputation overhead.
- We implement Obscura on DeepSpeed, demonstrating a throughput improvement of up to **1.33×** compared to widely used recomputation strategies on NVIDIA GPUs.

