DOPPLER: Dual-Policy Learning for Device Assignment in Asynchronous Dataflow Graphs

Xinyu Yao[1] Daniel Bourgeois[1] Abhinav Jain[1] Yuxin Tang[1] Jiawen Yao[1]
Zhimin Ding[1] Arlei Silva[1,2] Christopher Jermaine[1]

[1]Rice University [2]Rice Ken Kennedy Institute

ICLR

RICE KEN KENNEDY INSTITUTE
Responsible AI and Computing for Global Impact

NIH

Motivation

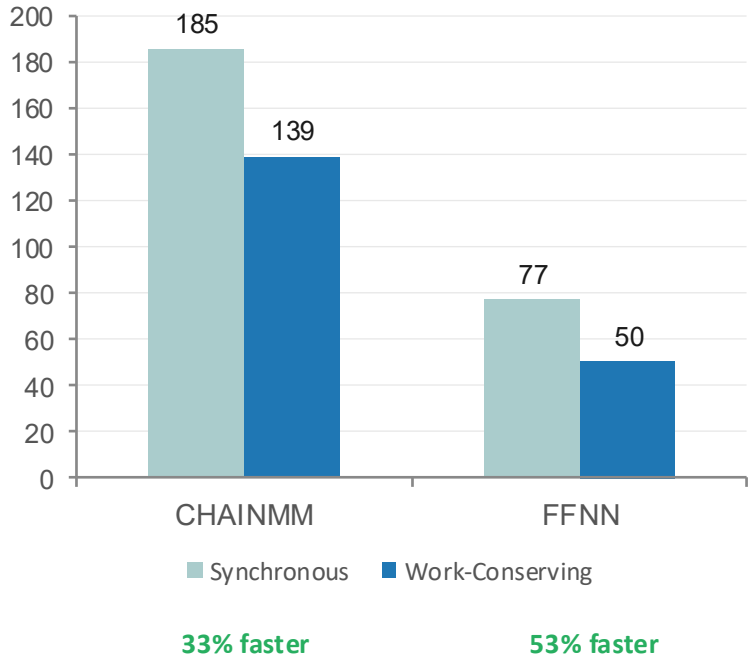
❑ Bulk-synchronous frameworks (PyTorch, JAX)

Execute level-wise with barrier synchronization — GPUs sit idle waiting for the slowest kernel before any collective can begin.

❑ Work-Conserving (WC) schedulers fix this

A WC scheduler never willingly leaves resources idle — it launches any task as soon as its inputs are available, fully overlapping compute and communication.

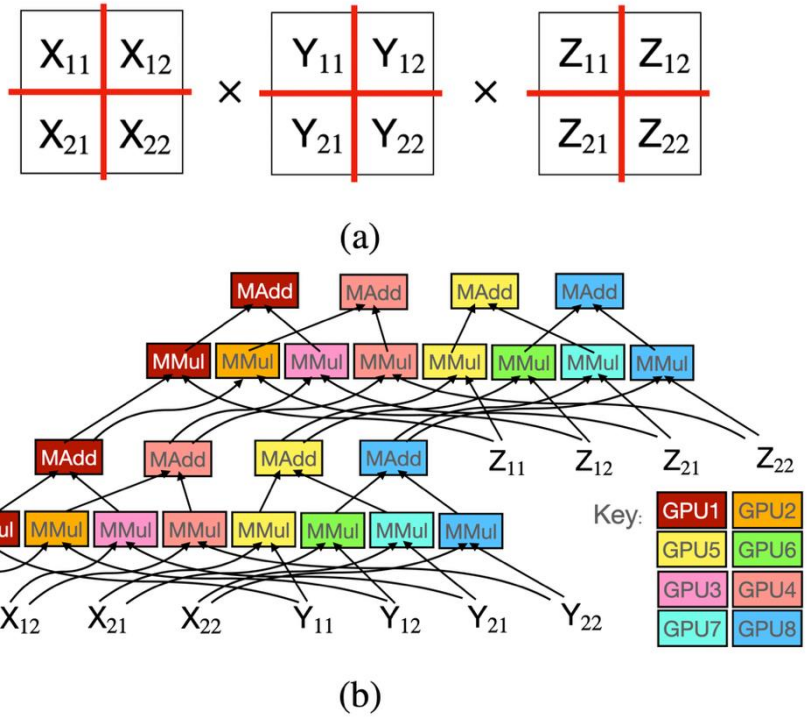
Synchronous vs. Work-Conserving Execution
Execution Time (milliseconds)



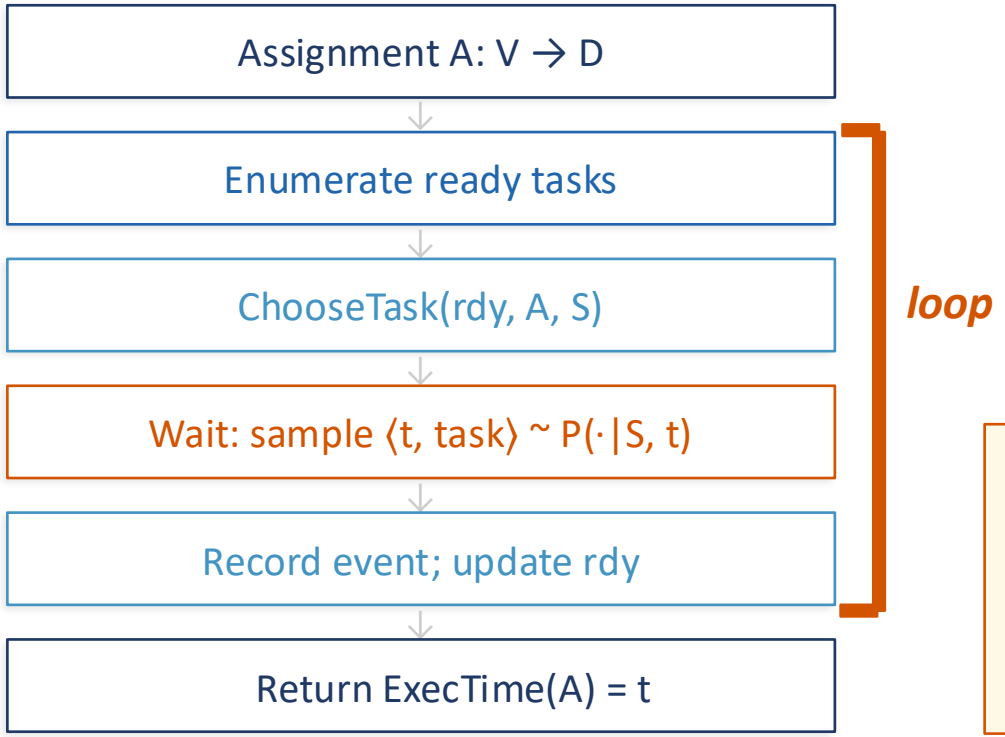
Device Assignment in Work-Conserving Systems

- **But device assignment in WC systems is hard**

  Without global synchronization, execution order is stochastic (PCIe contention, GPU jitter) — traditional placement methods designed for synchronous systems break down.



Minimizing ExecTime(A)



Key: same assignment A can produce different runtimes across runs — stochastic GPU/PCIe behavior means no closed-form formula for ExecTime(A)

Formulate as a Combinatorial Multi-Arm Bandit Problem

$D^{|V|}$ arms → 2^{300} possible assignments

(8 GPUs, 100-node graph)

Minimize regret:

$$\rho = \Sigma_t \left(\mathbb{E}[R^*] - r_t \right)$$

✗ **Classic bandits fail:** UCB / Thompson sampling require enumerating or sampling from 2^{300} arms — intractable.

✓ **DOPPLER's approach**

- Similar assignments → similar rewards (exploit structure)
- Dual-policy MDP: sequential SEL + PLC decisions
- Policy gradient RL with ExecTime reward signal

DOPPLER: Dual-Policy Framework

Dual-Policy Assignment (Algorithm 3)

Dataflow Graph G

$V = \{v_1 \ldots v_n\}$: kernel calls
E: data dependencies
D: device set
A: V → D (assignment)

→

SEL Policy (SEL_θ)

Selects the next node to assign. Learns a latent traversal order approximating the stochastic execution flow through G.

→

PLC Policy (PLC_θ)

Places the selected node on a GPU. Minimizes cross-GPU communication and balances compute load.

Three-Stage Training Framework

Stage I: Imitation Learning

Offline

Supervised: replicate decisions of the CRITICAL PATH heuristic. Provides a warm start — accelerates convergence in later RL stages.

→

Stage II: Simulation-Based RL

Offline

Reward = −ExecTime from a WC simulator. Safe exploration before deployment. Policy gradient update (REINFORCE) with baseline subtraction.

→

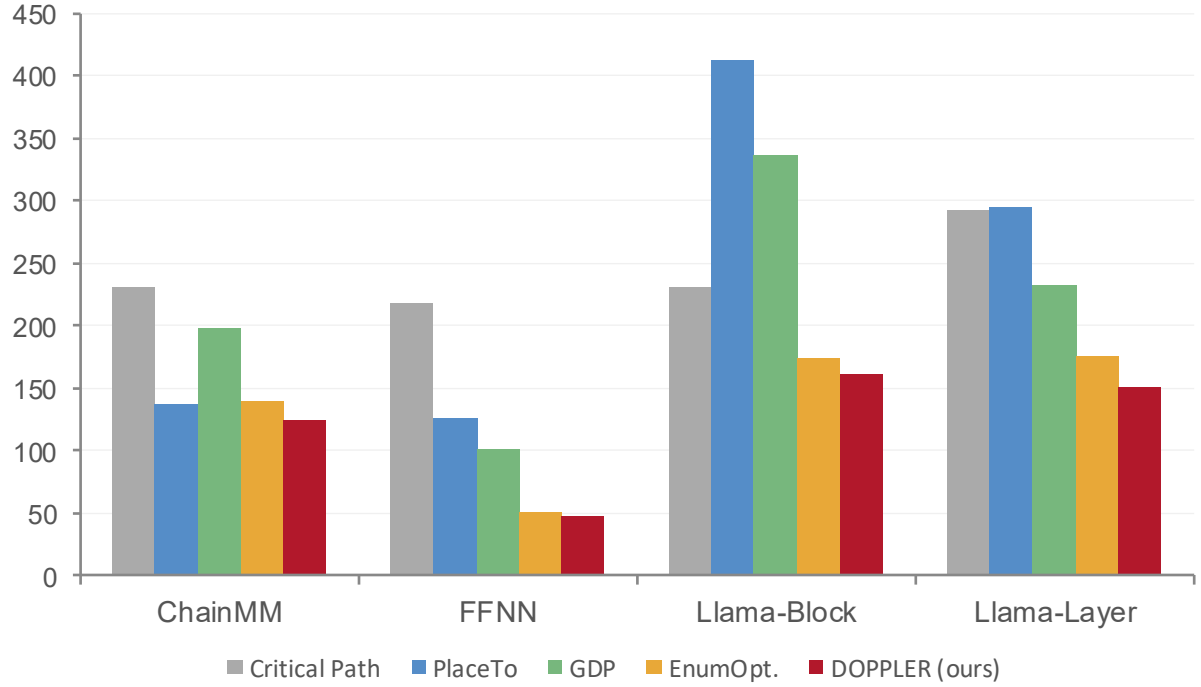
Stage III: Real-System RL

Online (Continuous)

Reward from observed production runtimes — free signal. Fine-tunes the policy on the target hardware while serving users.

Experimental Results

Real engine execution time on 4 GPUs (ms, lower is better)



4× NVIDIA Tesla P100 (16 GB) · Averaged over 10 runs · Std. dev. in paper

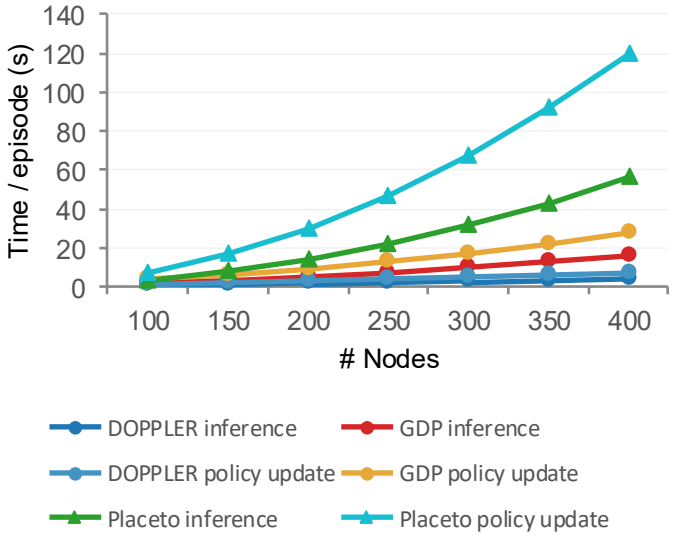
Transfer Learning & Scalability

Few-shot transfer to Llama graphs (ms)

Train → Target	Zero-Shot	2K-Shot	4K-Shot	Full Train
FFNN → Llama-Block	251.0	165.3	**159.4**	160.3
ChainMM → Llama-Block	242.3	184.9	**174.0**	160.3
FFNN → Llama-Layer	206.1	158.2	**155.8**	150.6
ChainMM → Llama-Layer	338.2	164.4	**156.4**	150.6

4K fine-tuning episodes (<50% of full training) matches full training performance — DOPPLER generalizes across architectures

Training & inference time vs. graph size



DOPPLER scales linearly;

Summary

1 **Work-conserving scheduling removes idle time**

WC systems outperform synchronous baselines by 33–53%, but require smarter device assignment because execution order is stochastic.

2 **Dual-policy decomposition makes the problem tractable**

Separating node selection (SEL_θ) from device placement (PLC_θ) decomposes the $D^{|V|}$-arm bandit into a learnable MDP.

3 **Three-stage training: imitation → simulation → real-system**

Each stage builds on the last: Stage I seeds from a heuristic, Stage II trains safely offline, Stage III refines in production at zero extra cost.

4 **Up to 52.7% runtime reduction over the best prior baseline**

DOPPLER outperforms Critical Path, PlaceTo, GDP, and our own EnumerativeOptimizer across ChainMM, FFNN, and Llama workloads .