

Habitat

A Runtime-Based Computational Performance Predictor for Deep Neural Network Training

Geoffrey X. Yu, Yubo Gao, Pavel Golikov, Gennady Pekhimenko

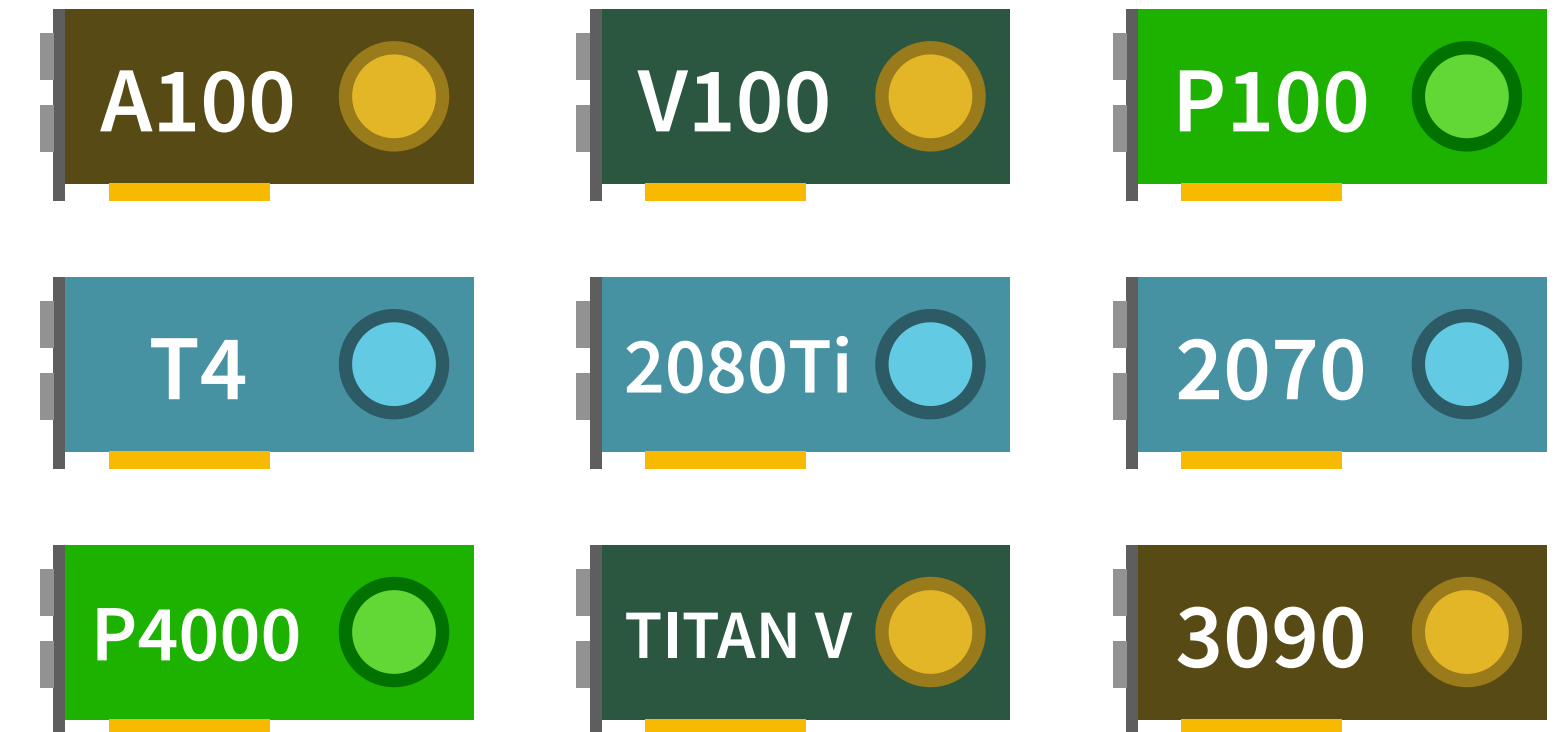


 Get started: github.com/geoffxy/habitat

What this talk is about

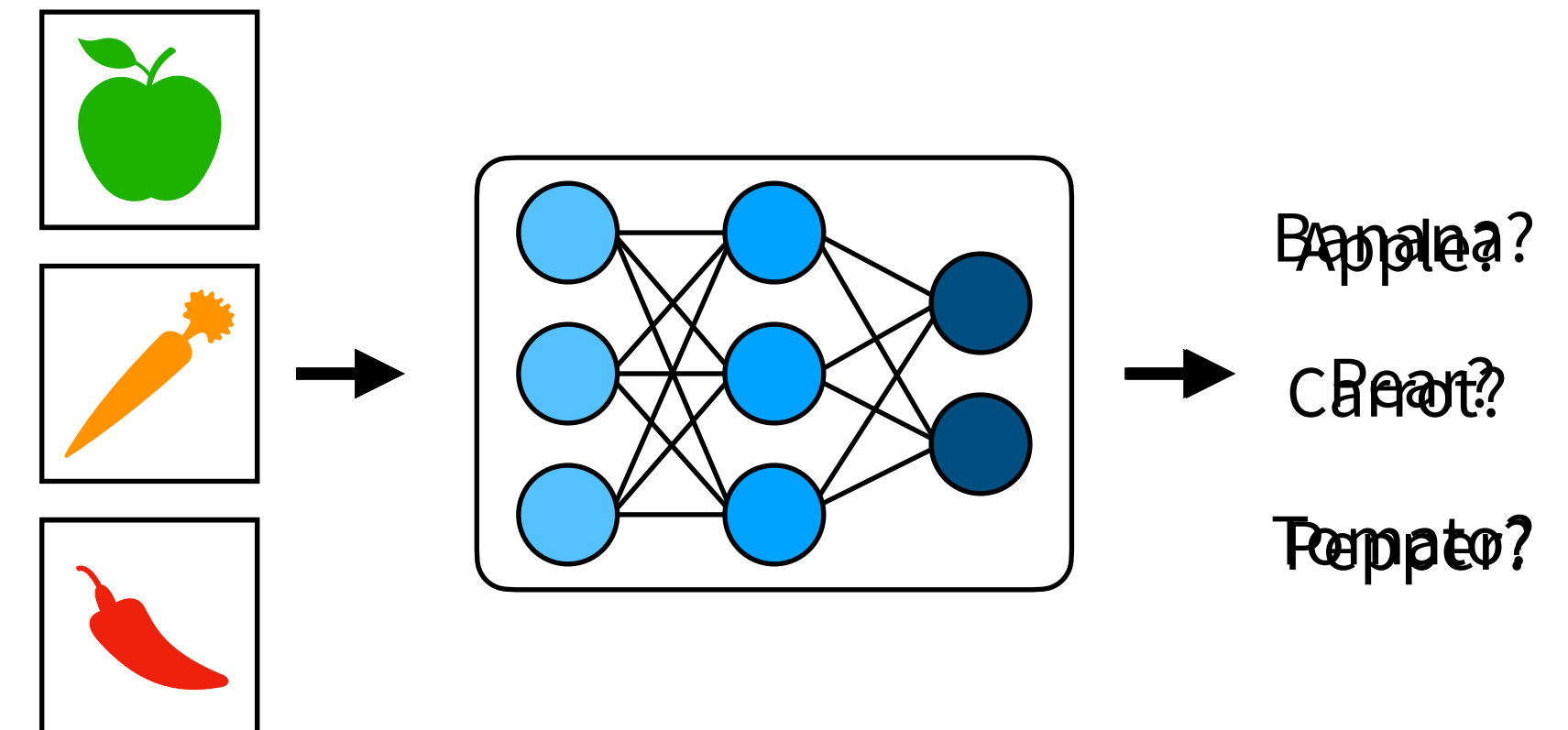
The problem:

- Many GPUs available for deep neural network (DNN) training
 - Each has a different 💰 cost and 🔥 performance
 - Which should a user choose for training?



Key observations:

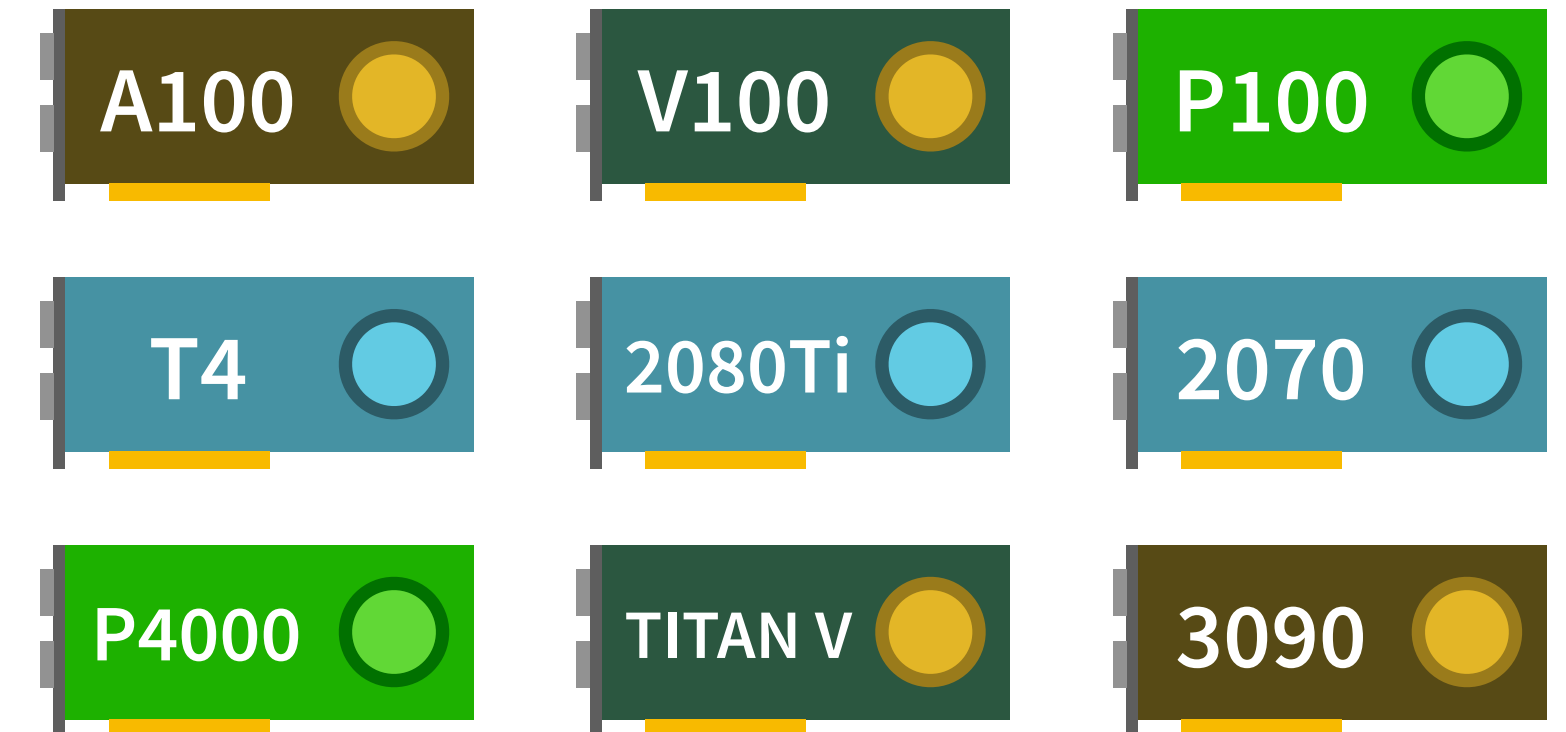
- DNN training computation is ↻ highly repetitive
- Predict a GPU's training performance by 🌟 predicting the execution time of a single iteration



What this talk is about

The problem:

- Many GPUs available for deep neural network (DNN) training
 - Each has a different 💰 cost and 🔥 performance
 - Which should a user choose for training?



Our work:

- Use an existing GPU to 🌟 predict execution times on a different GPU using 🌊 wave scaling and 🧠 pre-trained multilayer perceptrons (MLPs)
- Implement ideas in a new tool called ✨ Habitat (open source, supports PyTorch)
- Show 🔍 two case studies where Habitat leads users to the ✅ correct GPU choice

Deep neural networks (DNNs) are everywhere

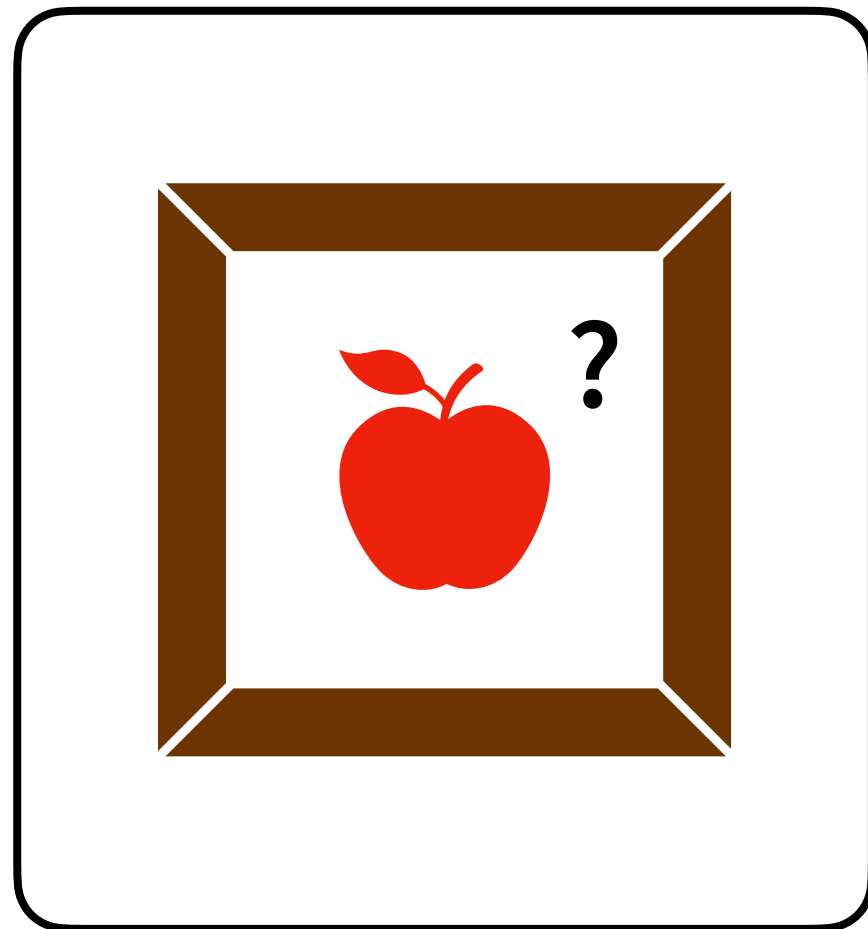
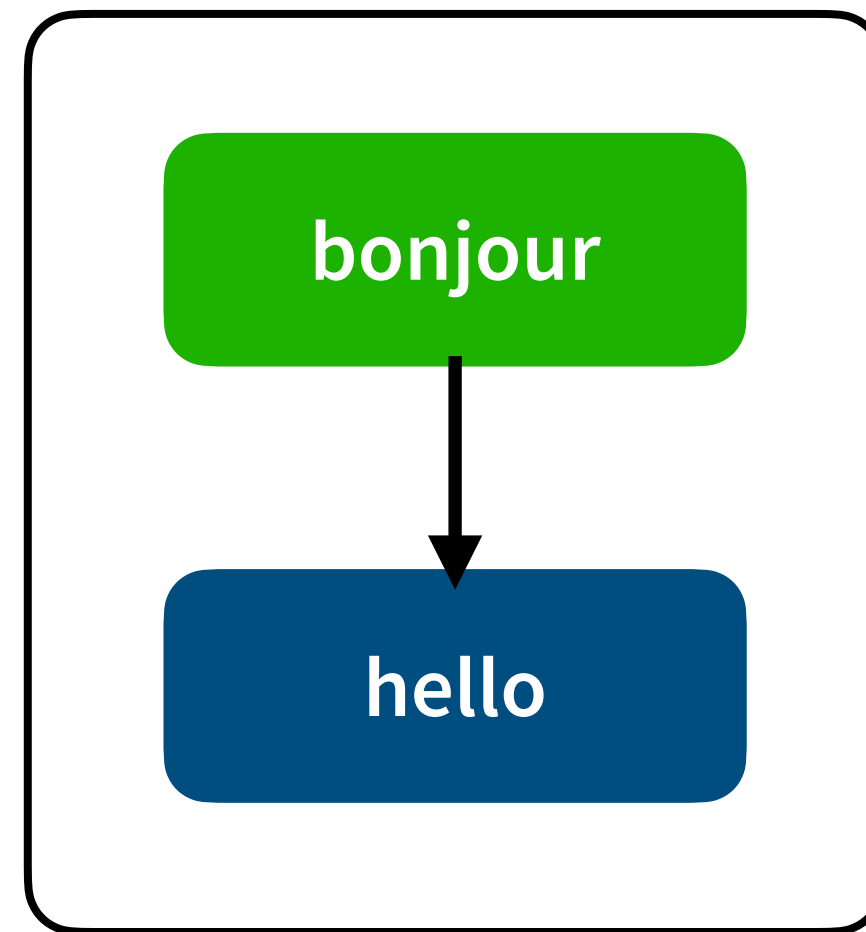


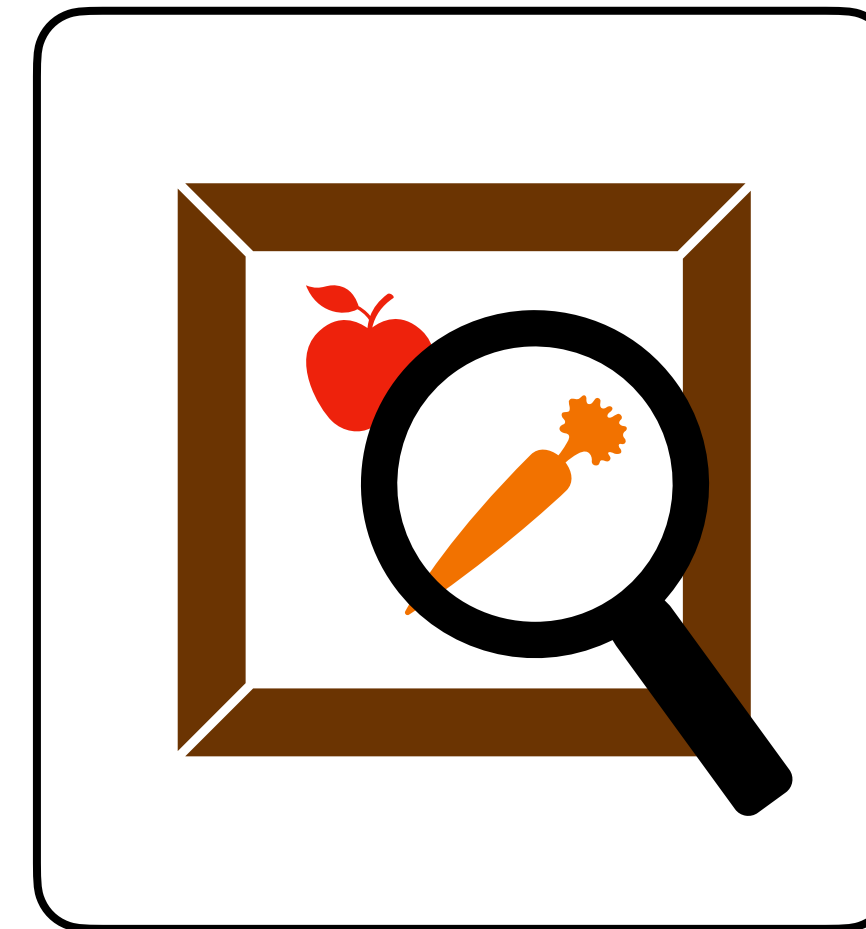
Image Classification

ResNet [CVPR'16]
VGG [ICLR'15]
AlexNet [NeurIPS'12]



Machine Translation

Transformer [NeurIPS'17]
Seq2Seq NMT [NeurIPS'14]



Object Detection

YOLO [CVPR'16]
SSD [ECCV'16]
Fast R-CNN [ICCV'15]

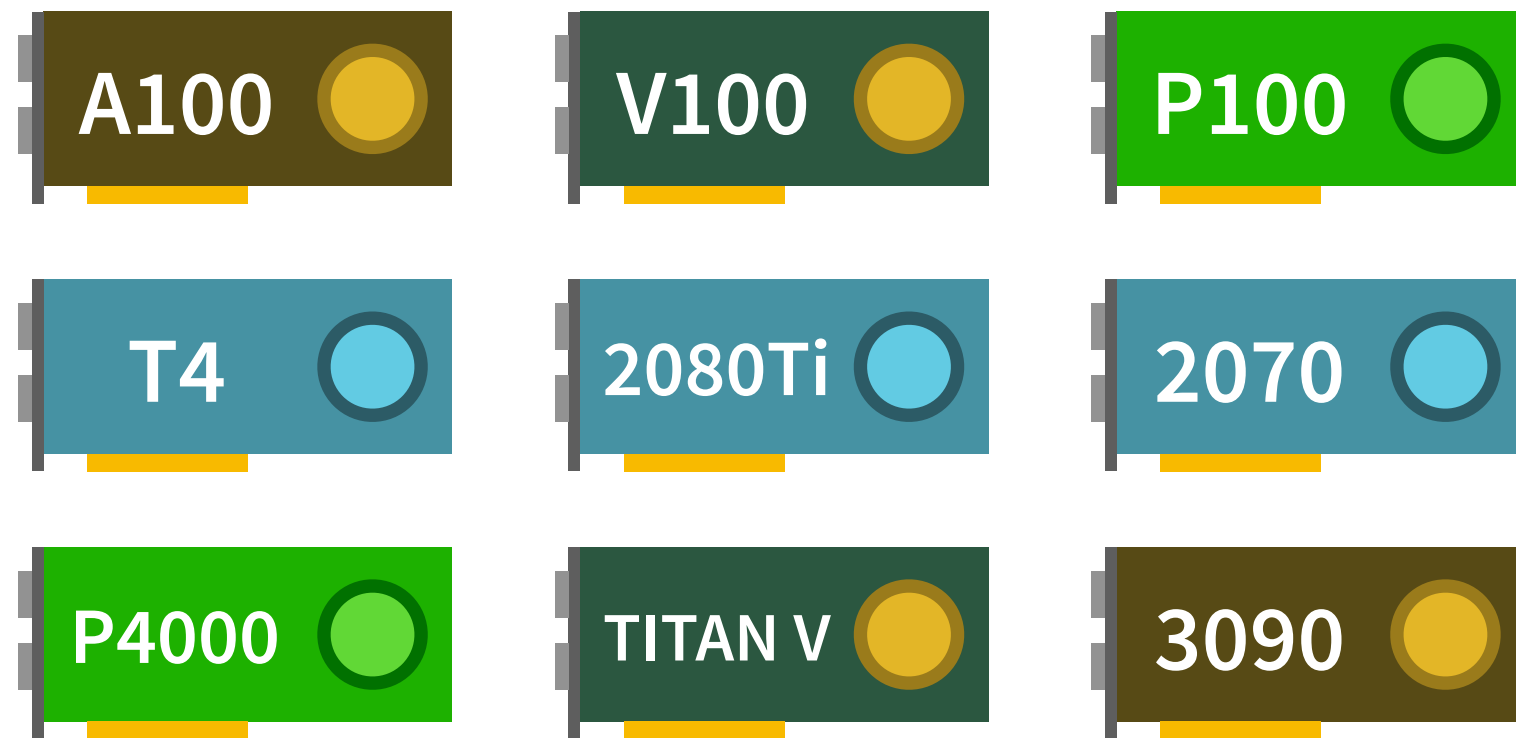


Speech Recognition

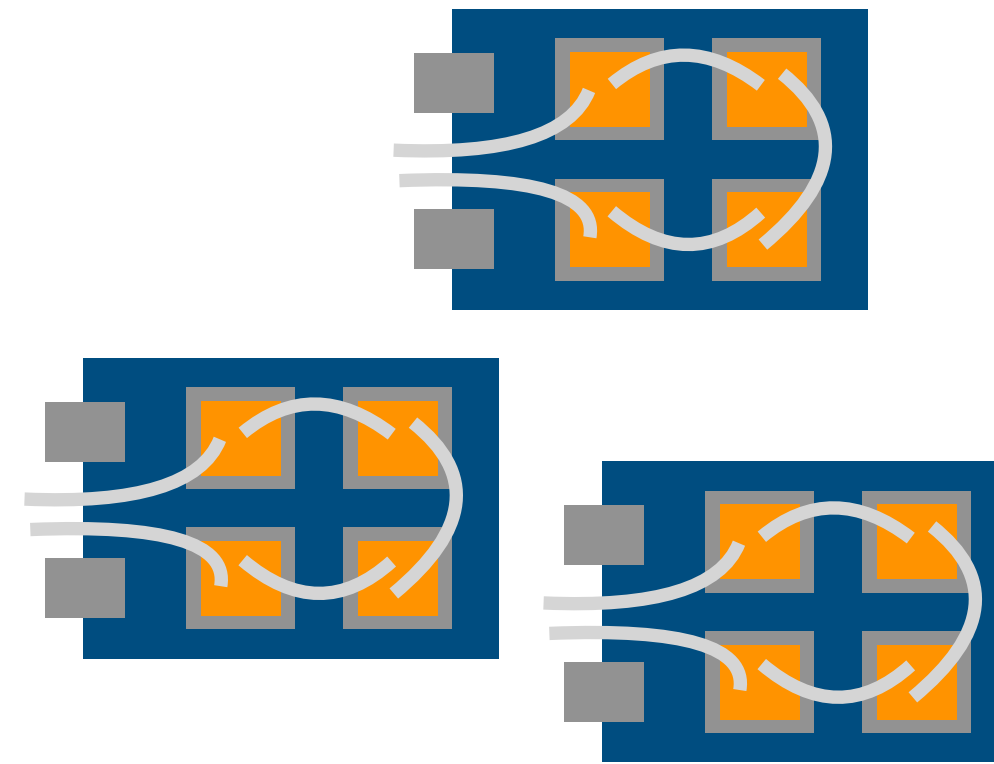
Deep Speech 2 [ICML'16]
End-to-End w/ RNNs [ICML'14]

But they are often **computationally expensive** to train!

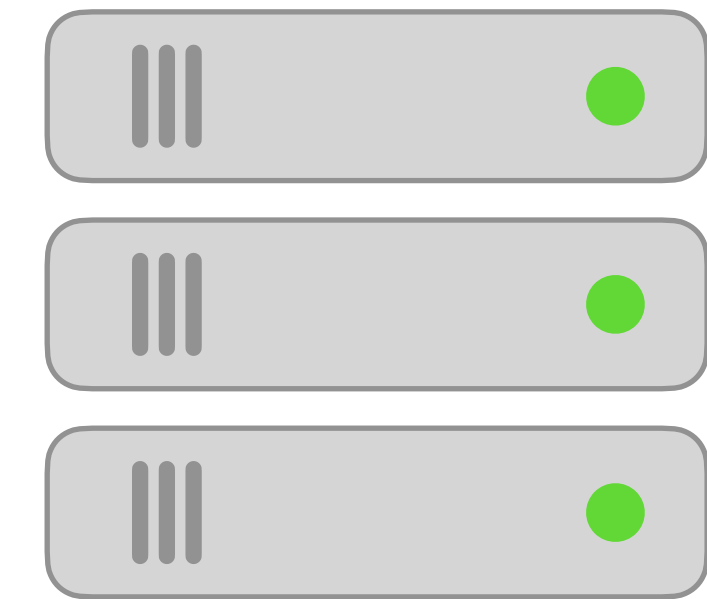
A Cambrian explosion in hardware for training



GPUs (workstation, cloud)



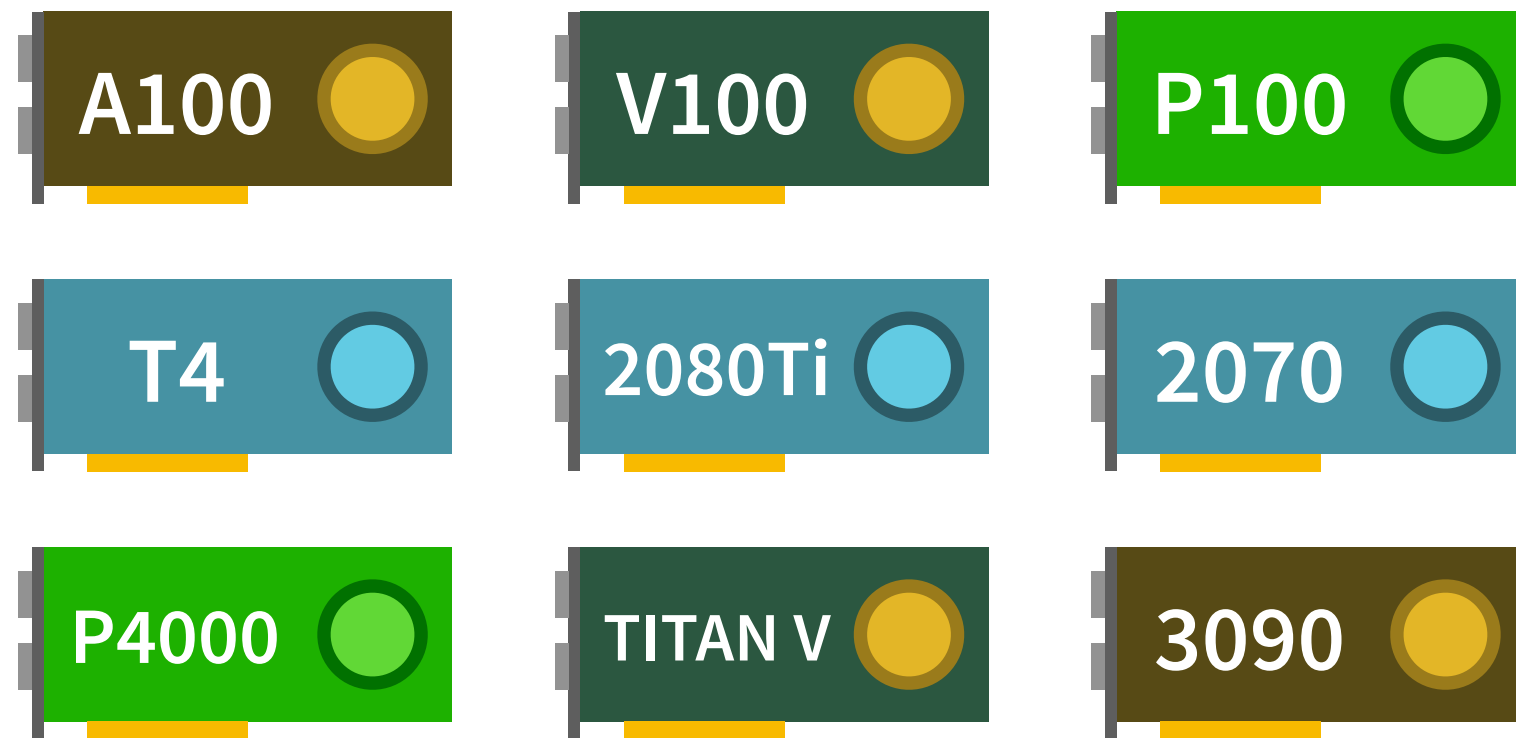
TPUs (v2, v3, v4)



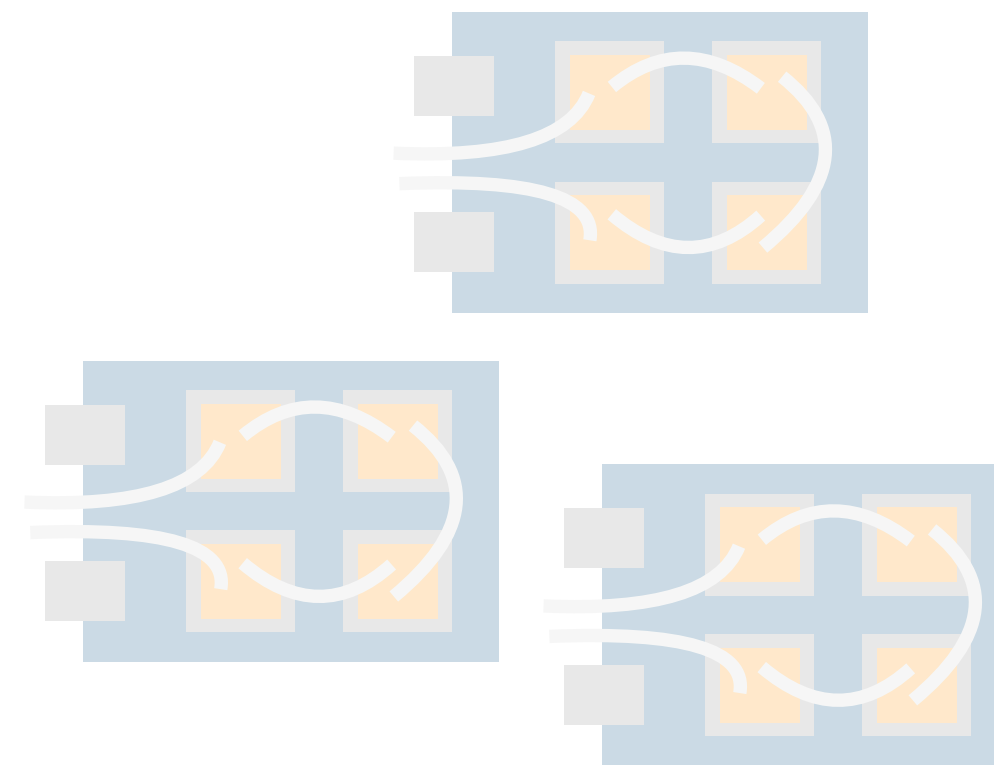
Other Emerging Accelerators
Cerebras WSE, Habana Gaudi,
AWS Trainium

Which accelerator should you use?

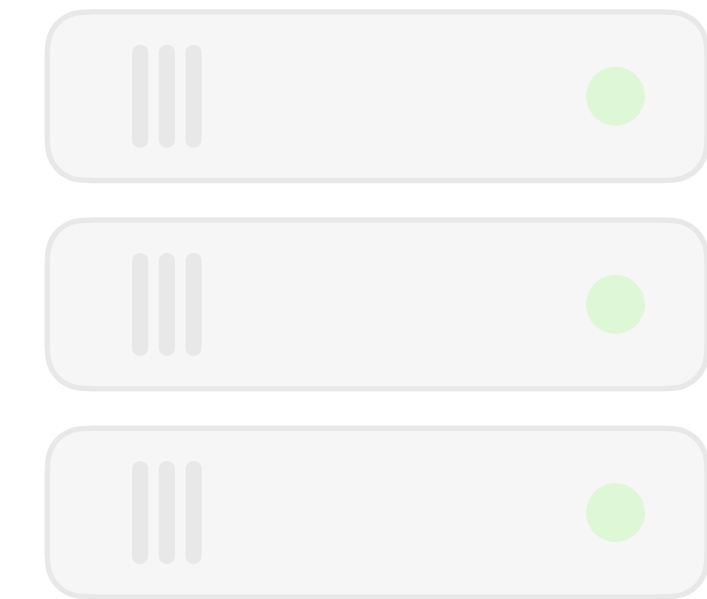
A Cambrian explosion in hardware for training



GPUs (workstation, cloud)



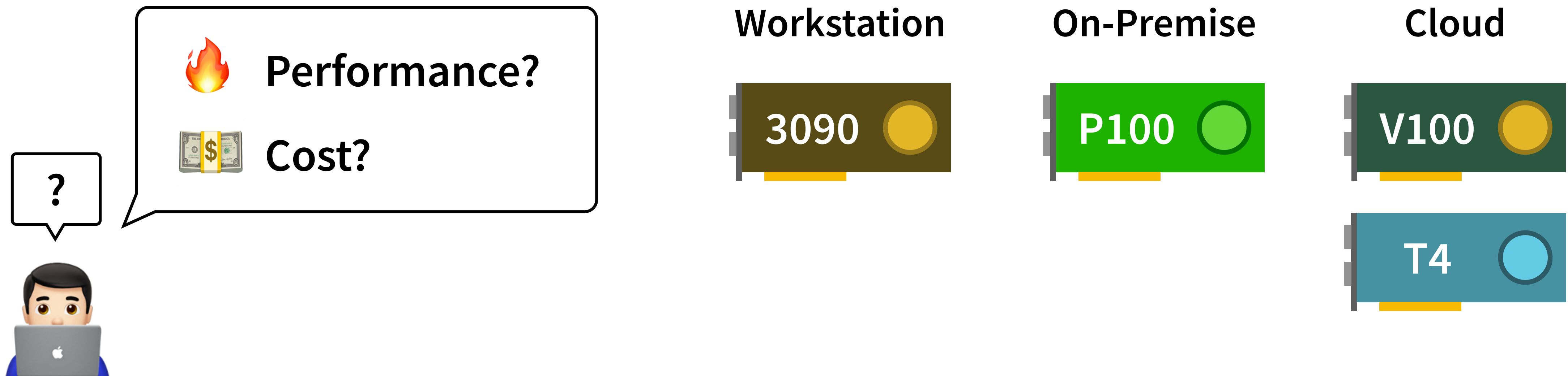
TPUs (v2, v3, v4)



Other Emerging Accelerators
Cerebras WSE, Habana Gaudi,
AWS Trainium

Which **GPU** should you use?

Choosing a GPU: The paradox of choice

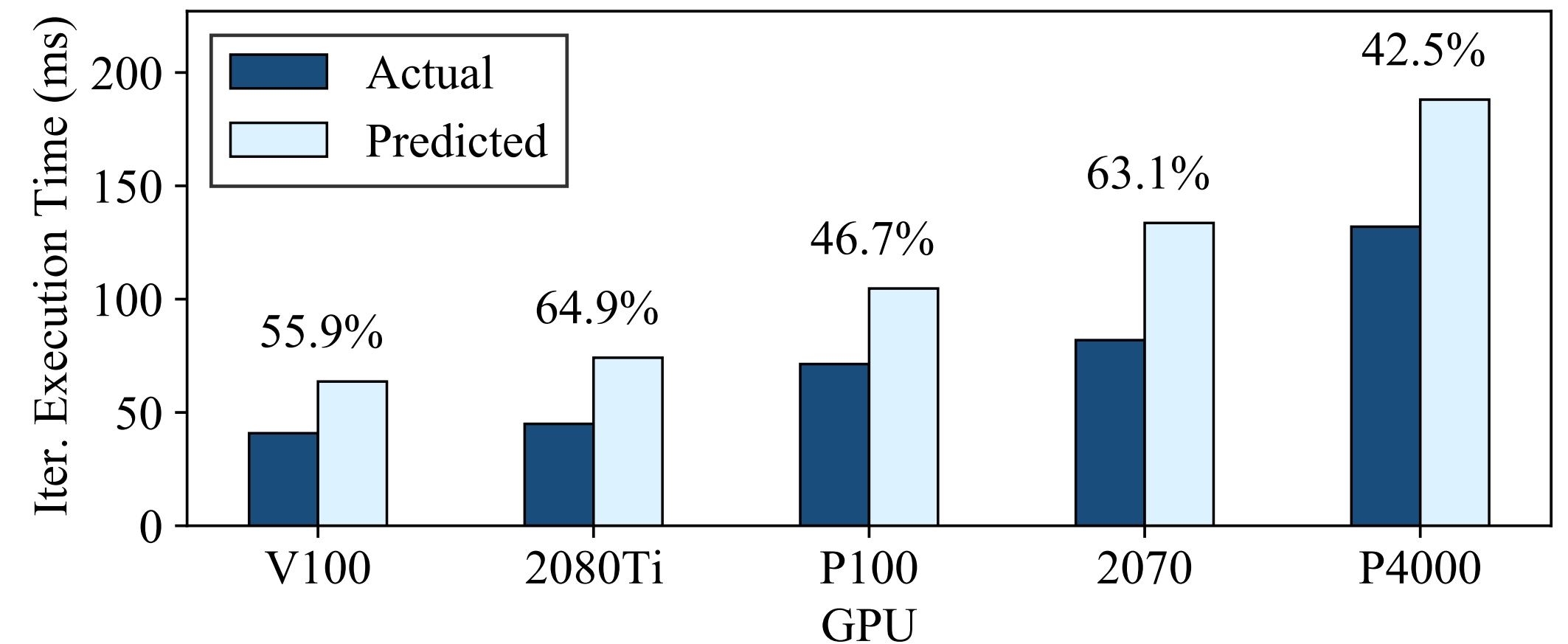


No one-size-fits-all choice. The “correct” choice depends on the user’s needs!

Why not just...


- Measure directly?
 - 💰 Need to pay to access the GPU(s)
 - 😴 Tedious to repeat for many models
- Use existing benchmarking results?
 - ⚠️ Not available for all models / GPUs
- Use simple heuristics?
 - ❌ Do not always work

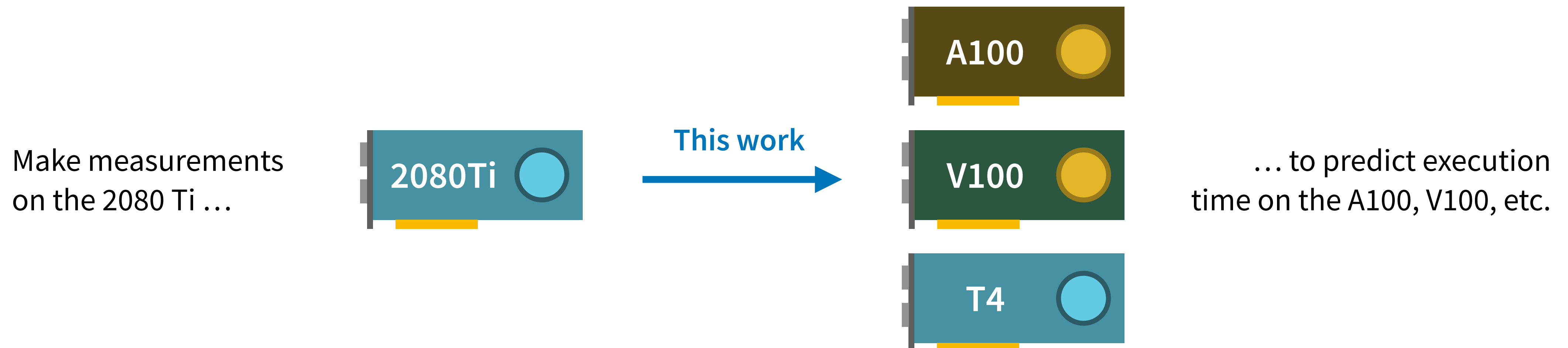
DCGAN Iteration Execution Time Predictions
(Peak FLOPS Ratios using the T4)




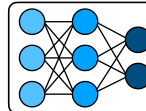
Simple heuristics can lead to **high** (> 43%) prediction errors!

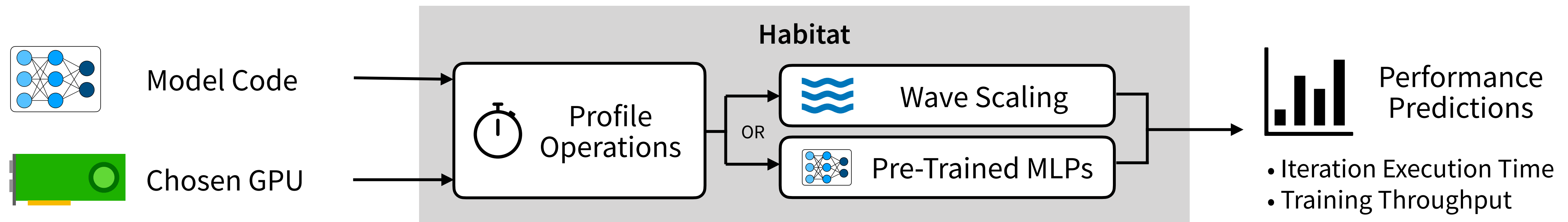
Key observations

- Deep learning users may already have an **existing GPU**
- DNN training is a  **repetitive process** (short training iterations)
- Use **existing GPU** to make iteration execution time **predictions** for **other GPUs**



Habitat: A runtime-based performance predictor

1. Profile all operations in a training iteration on an existing GPU
2. Predict each operation using  wave scaling or a  multilayer perceptron (MLP)
3. Add predictions together to get an iteration execution time prediction

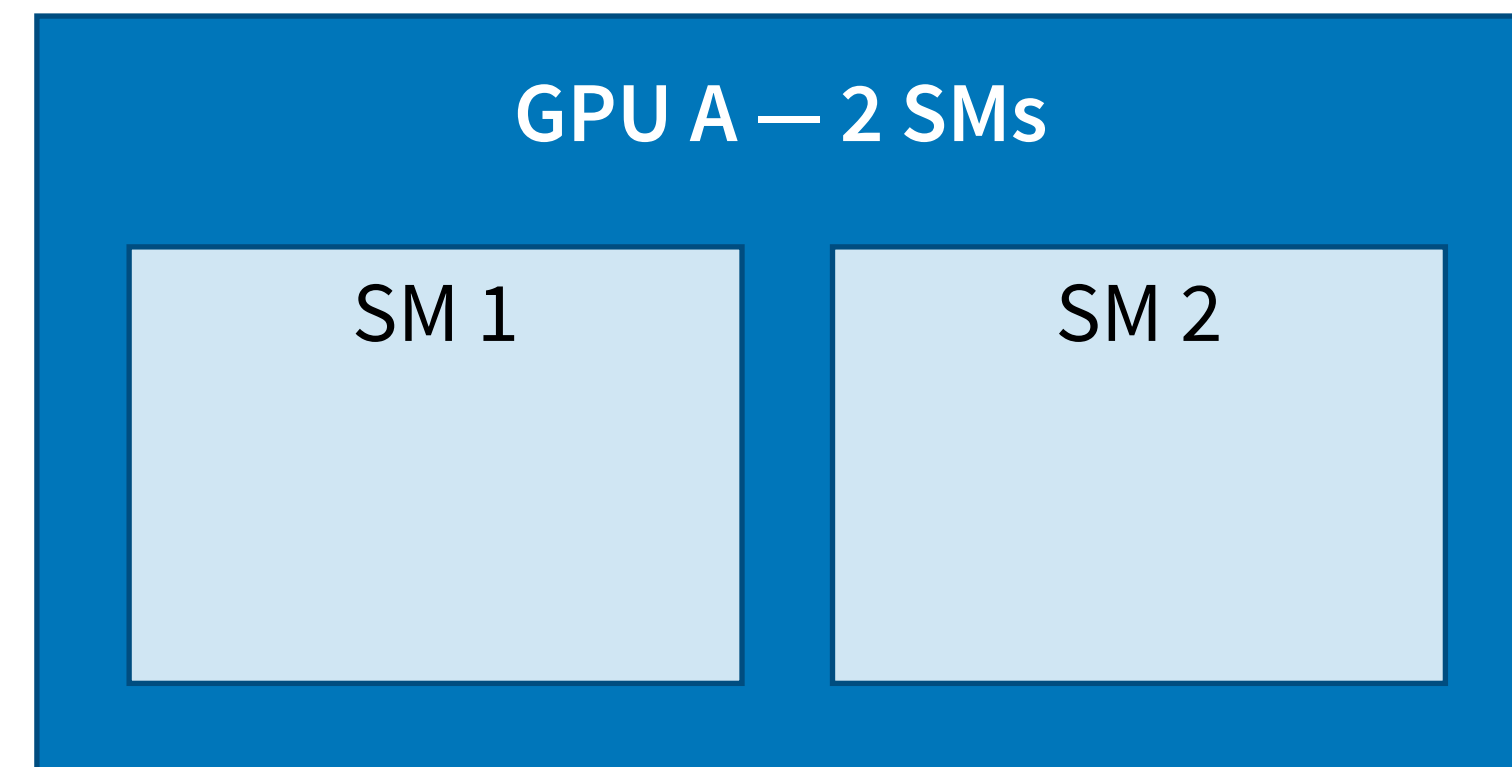
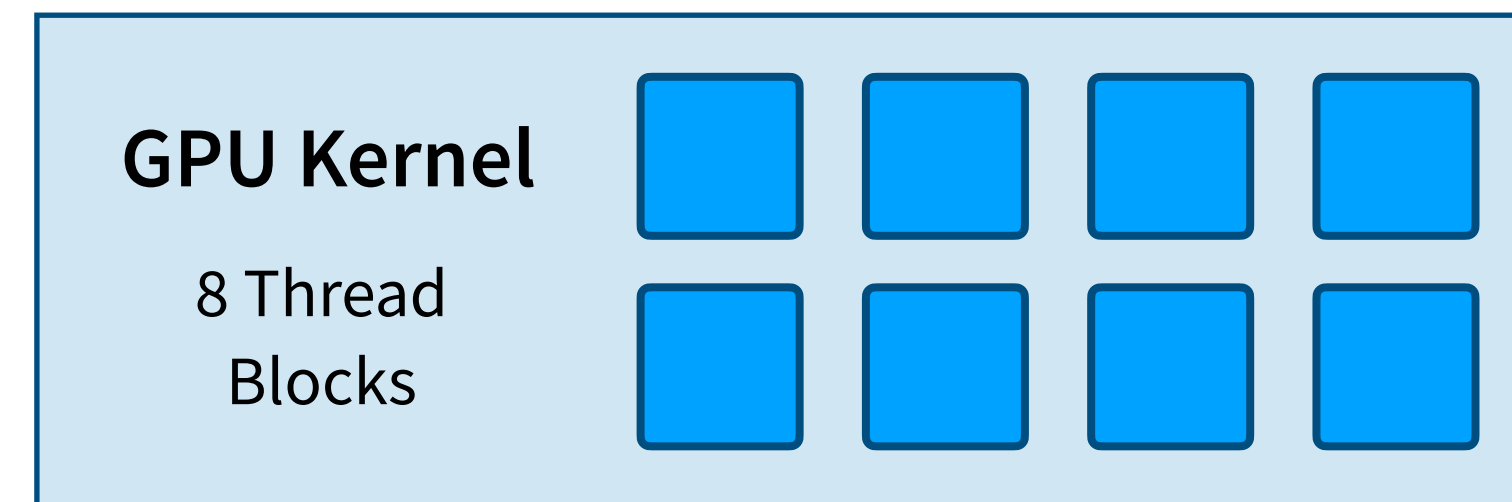


Habitat is an open source Python library; it supports PyTorch 1.4.0

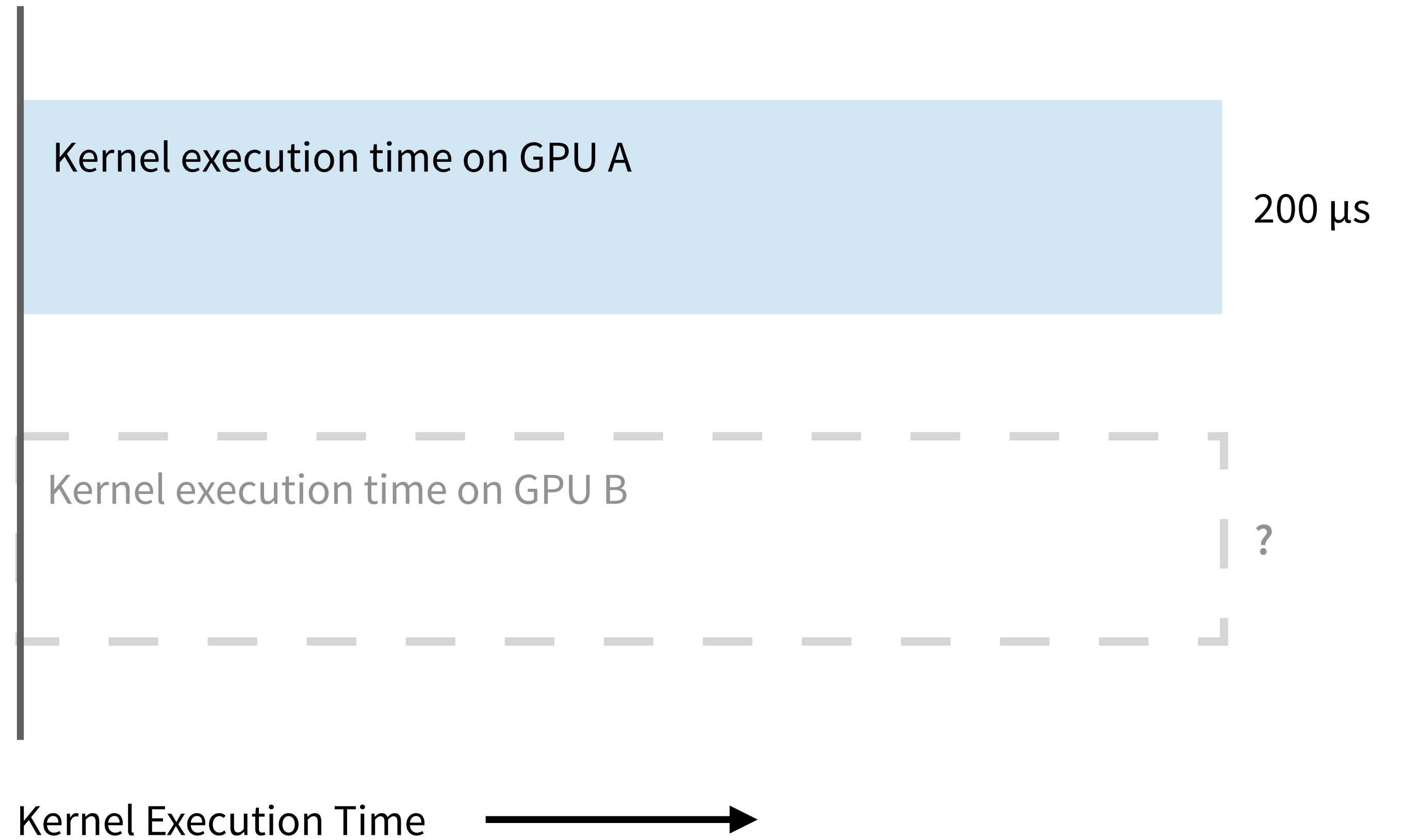
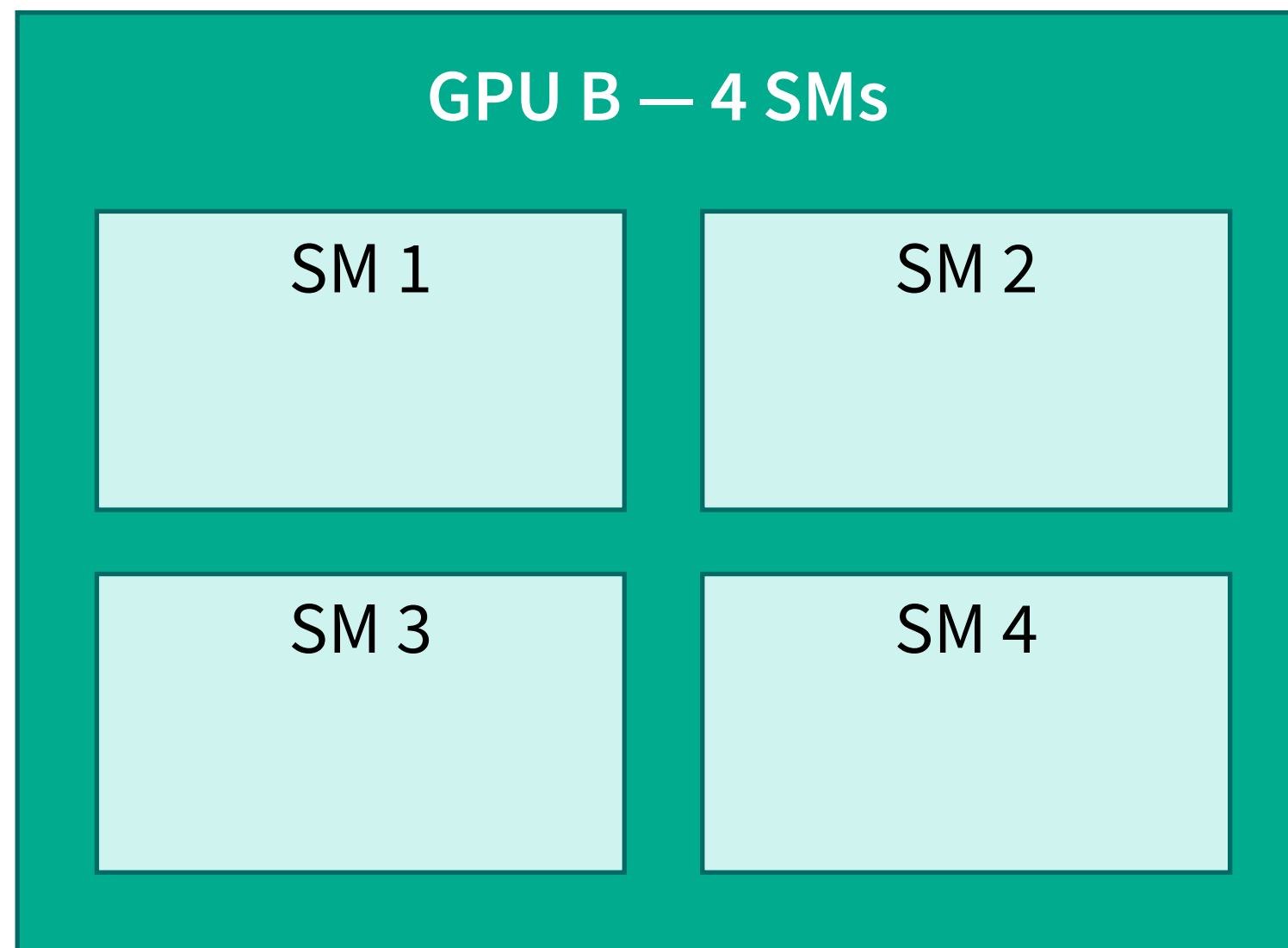
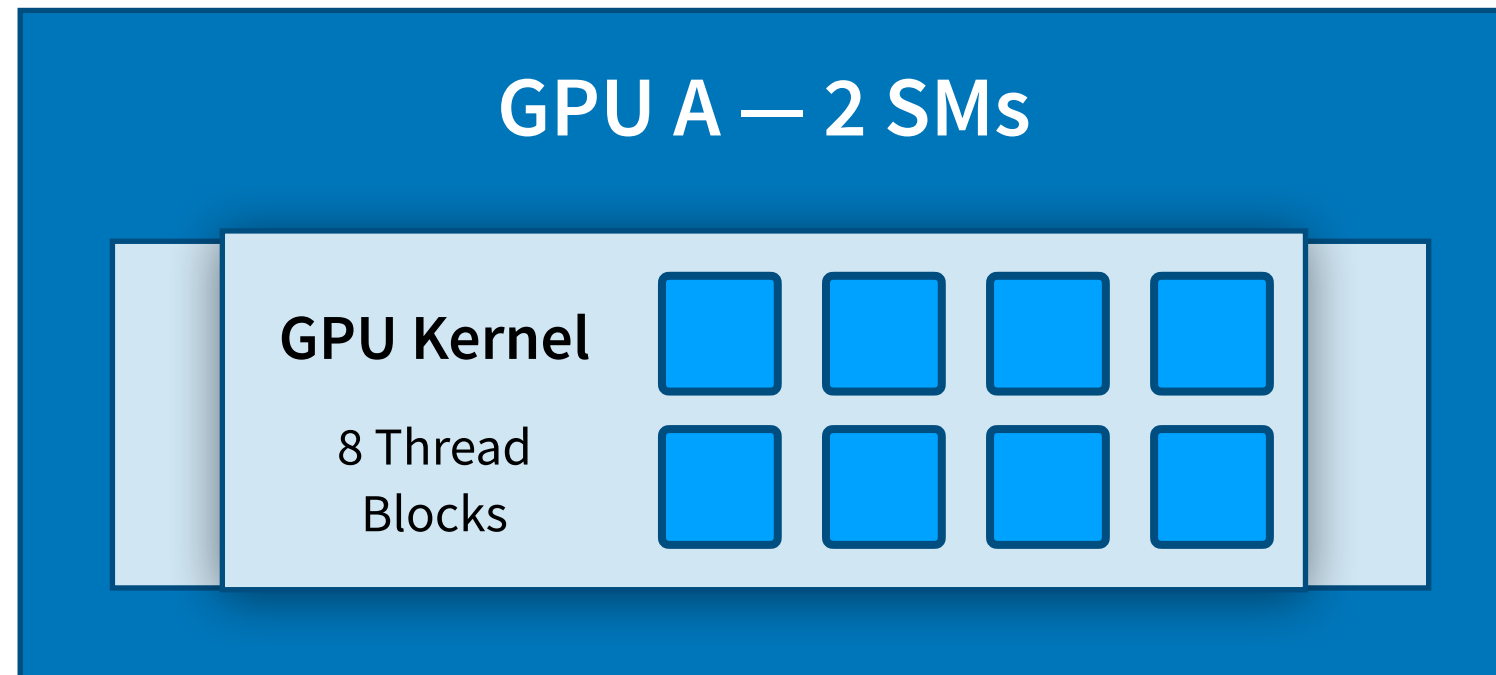
How does Habitat work?

Background: GPU execution model

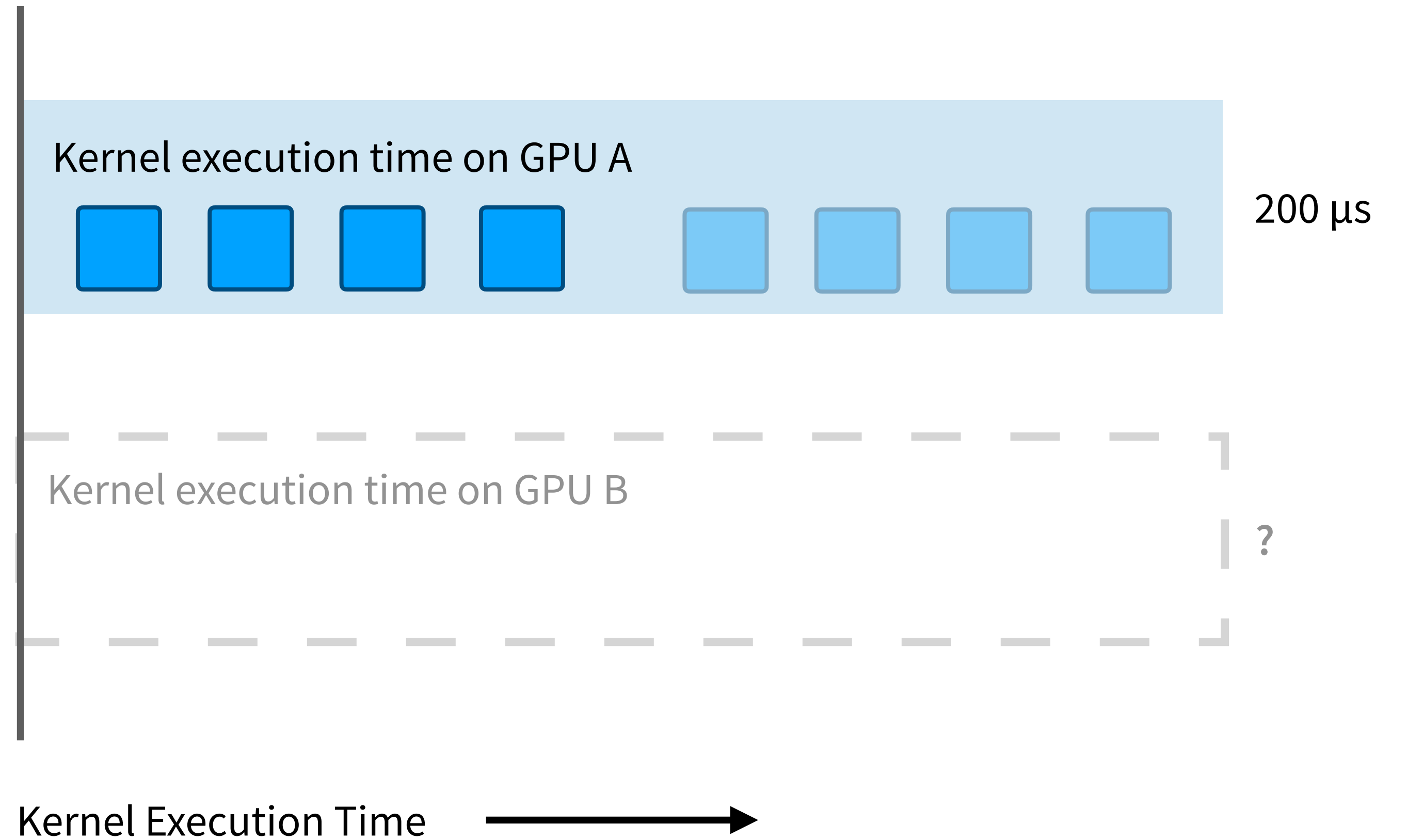
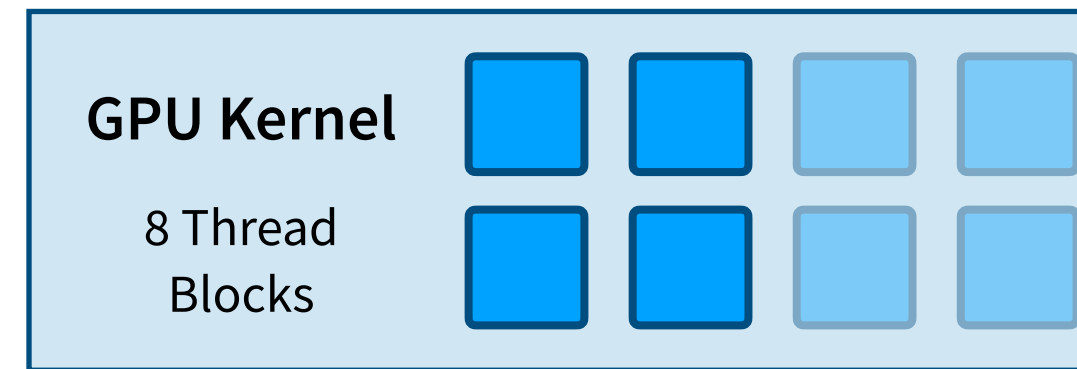
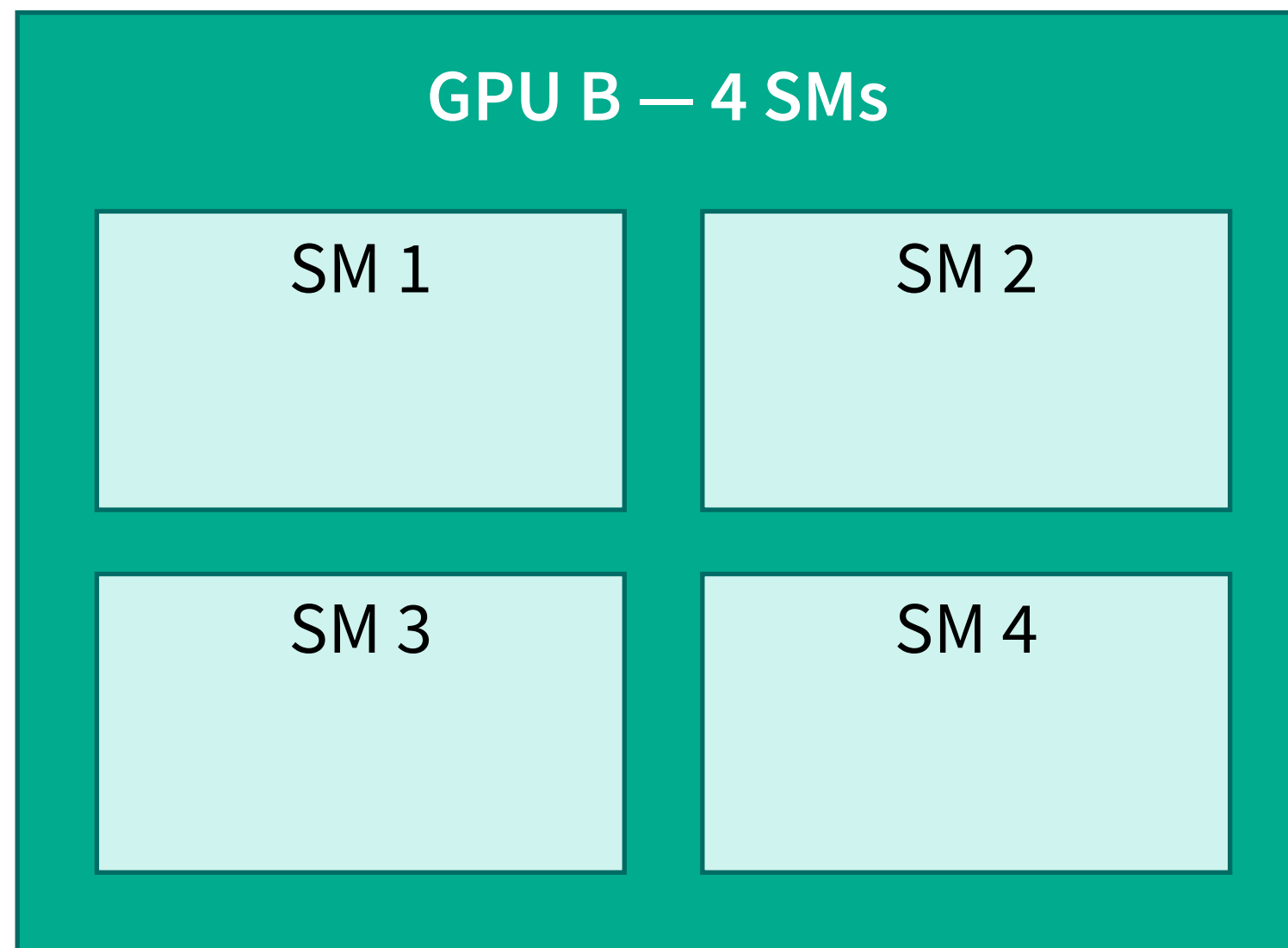
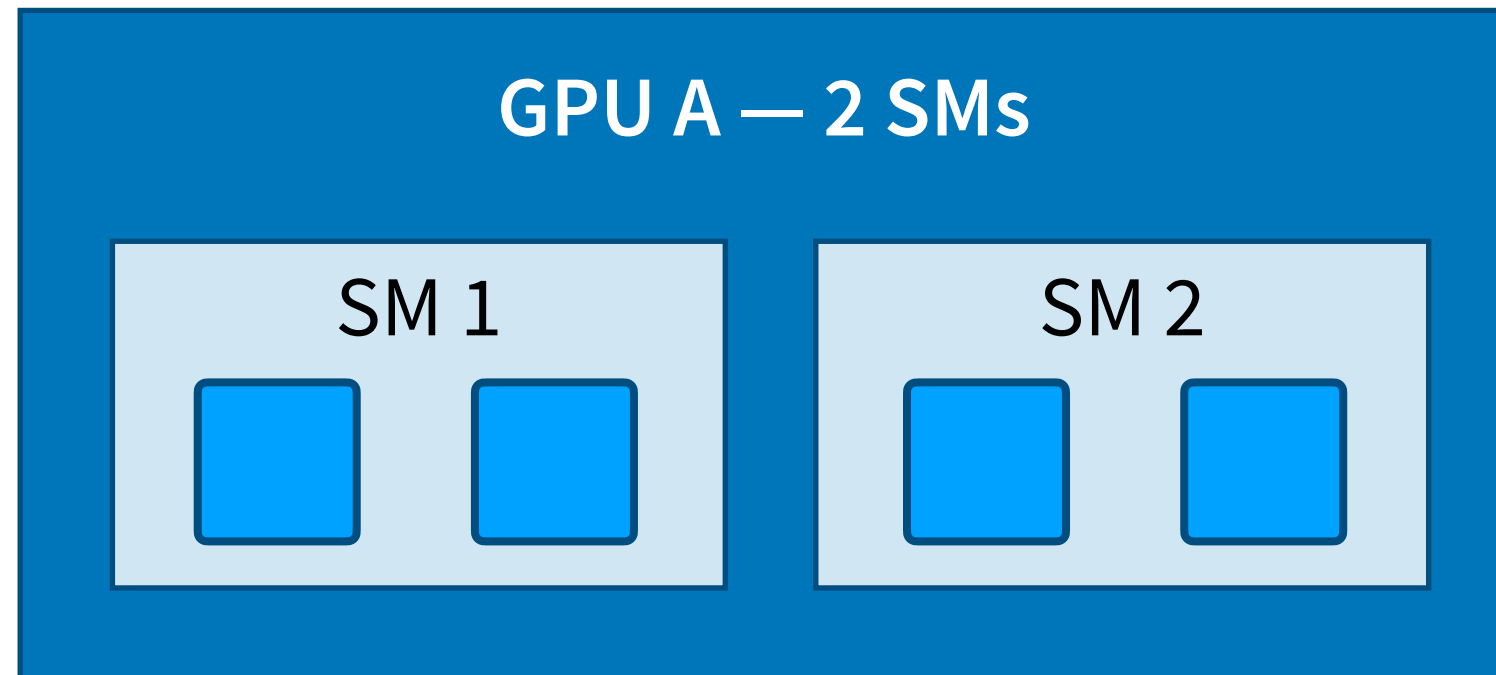
- GPU kernels: “work” divided into **thread blocks** (same code, different data)
- Streaming multiprocessors (SMs) run a **finite number** of blocks concurrently
- Blocks **round-robin** scheduled onto SMs
- GPU kernels execute in “**waves**” of thread blocks



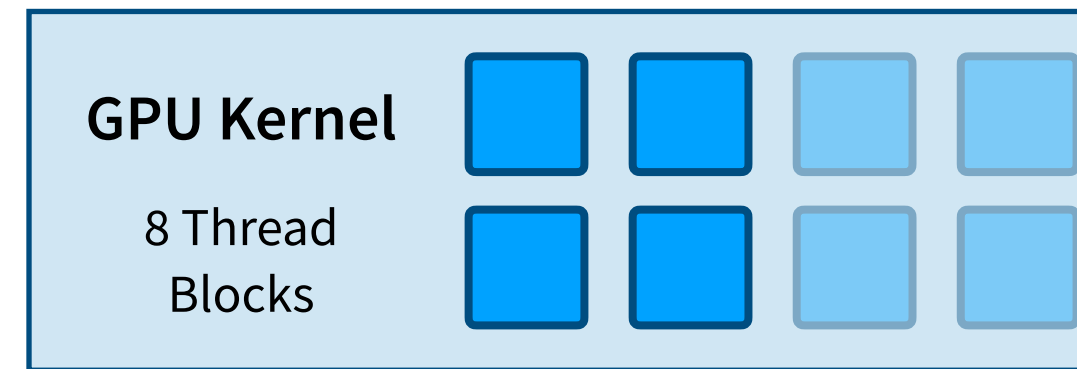
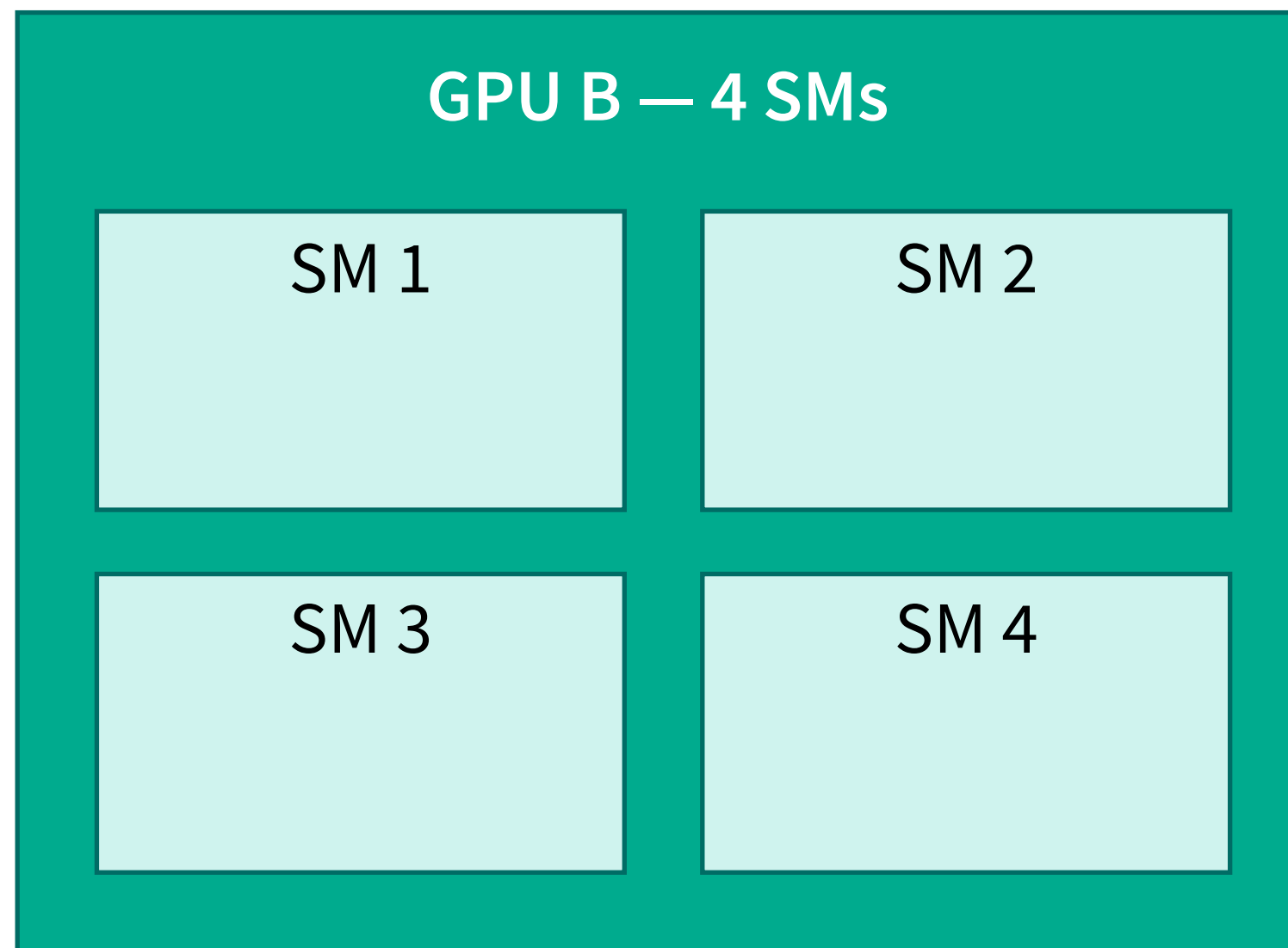
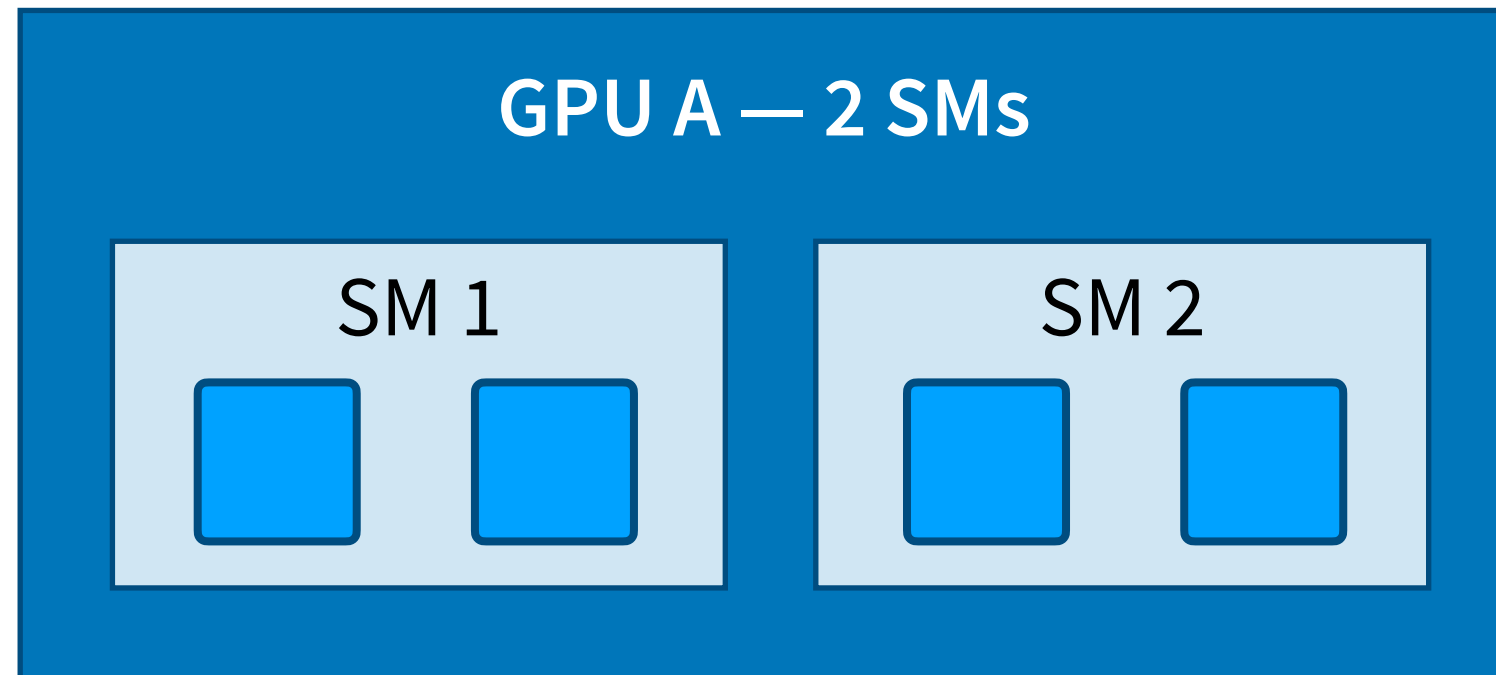
Wave scaling



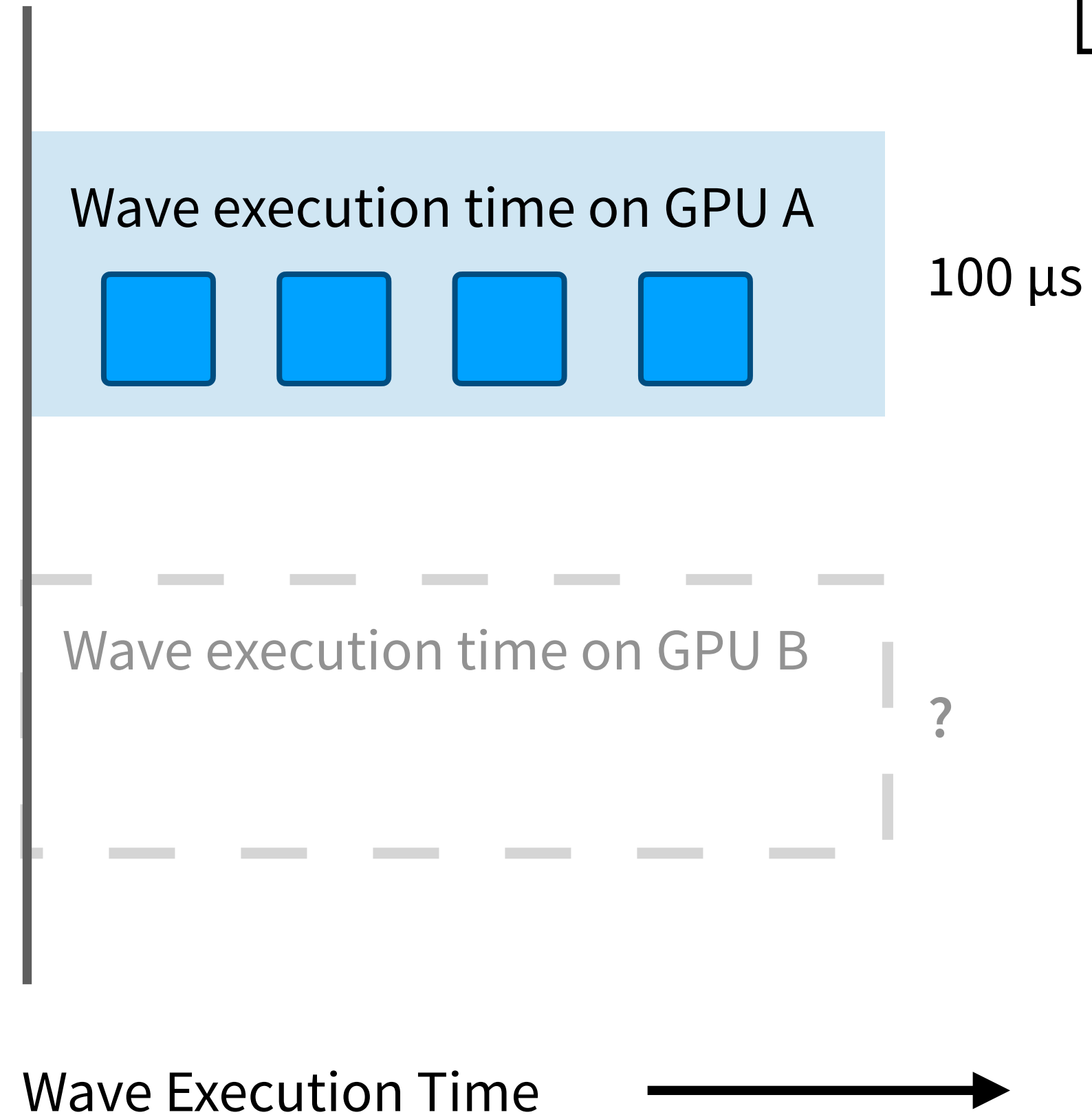
Wave scaling



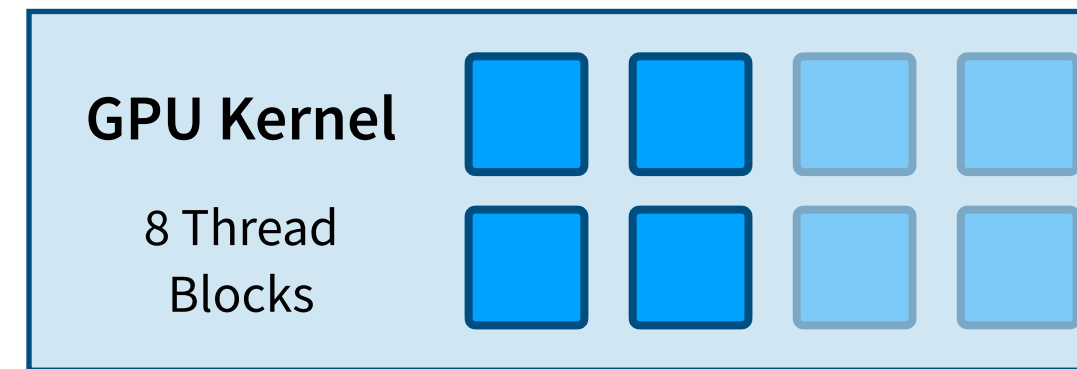
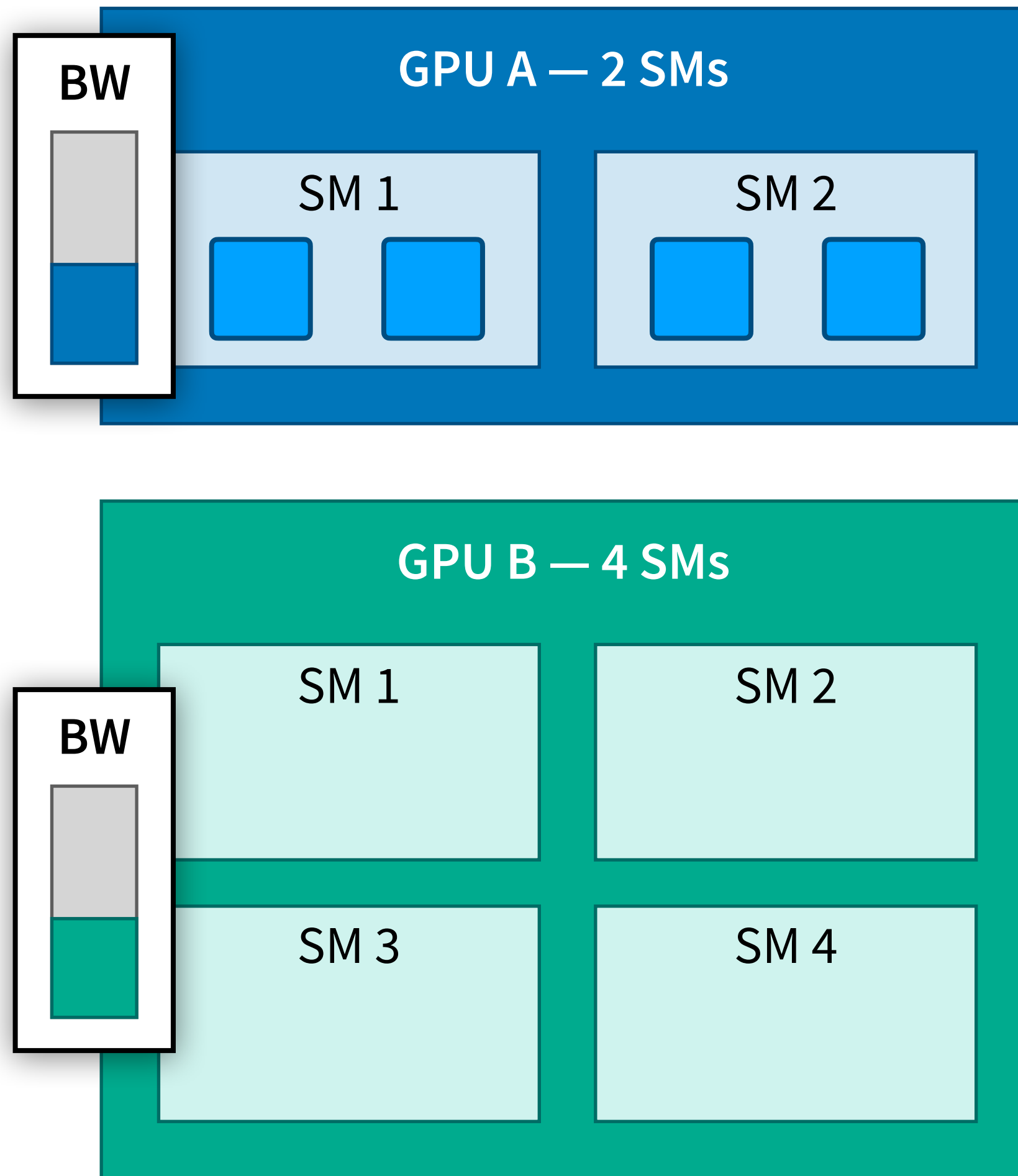
Wave scaling



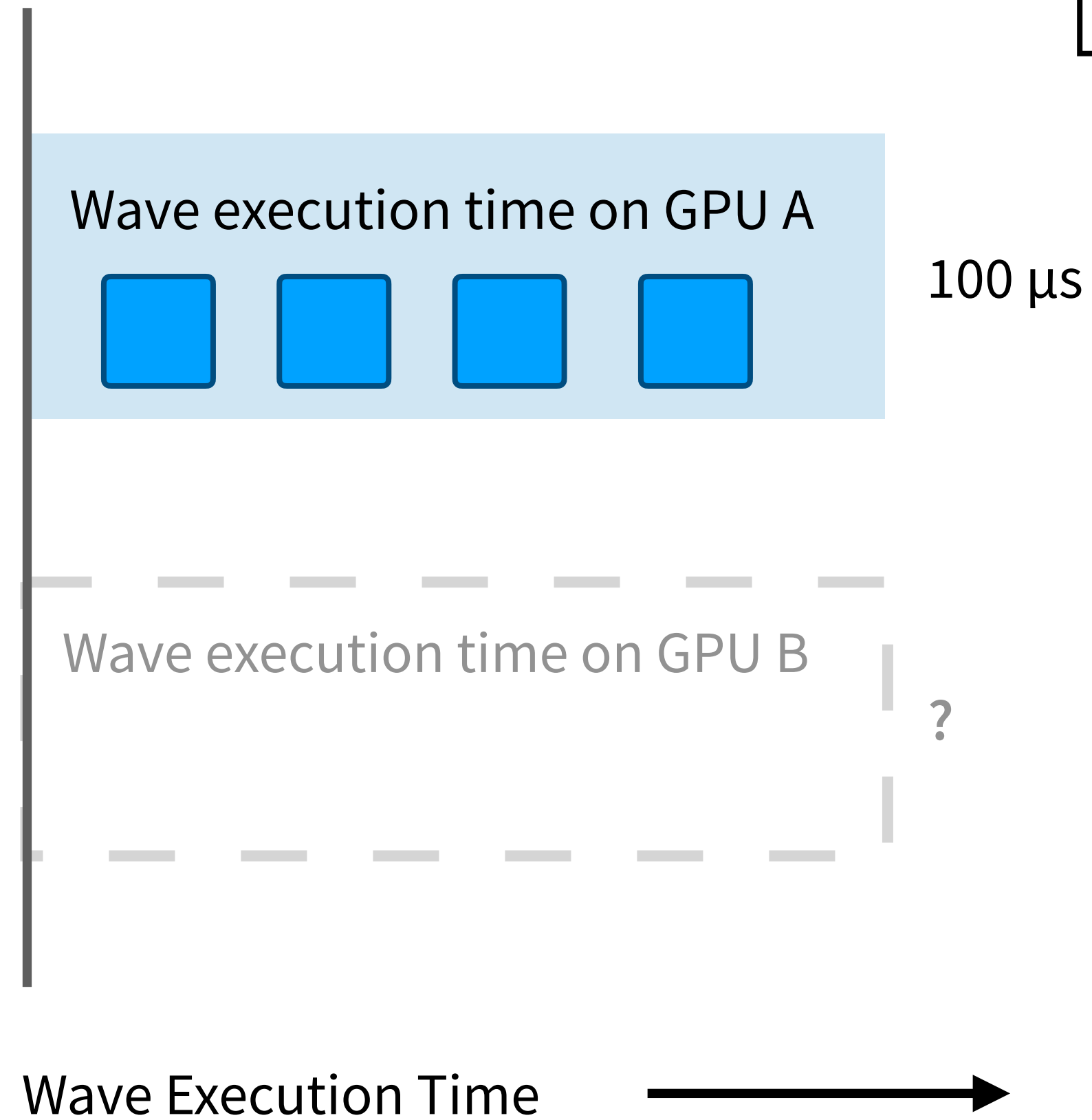
- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



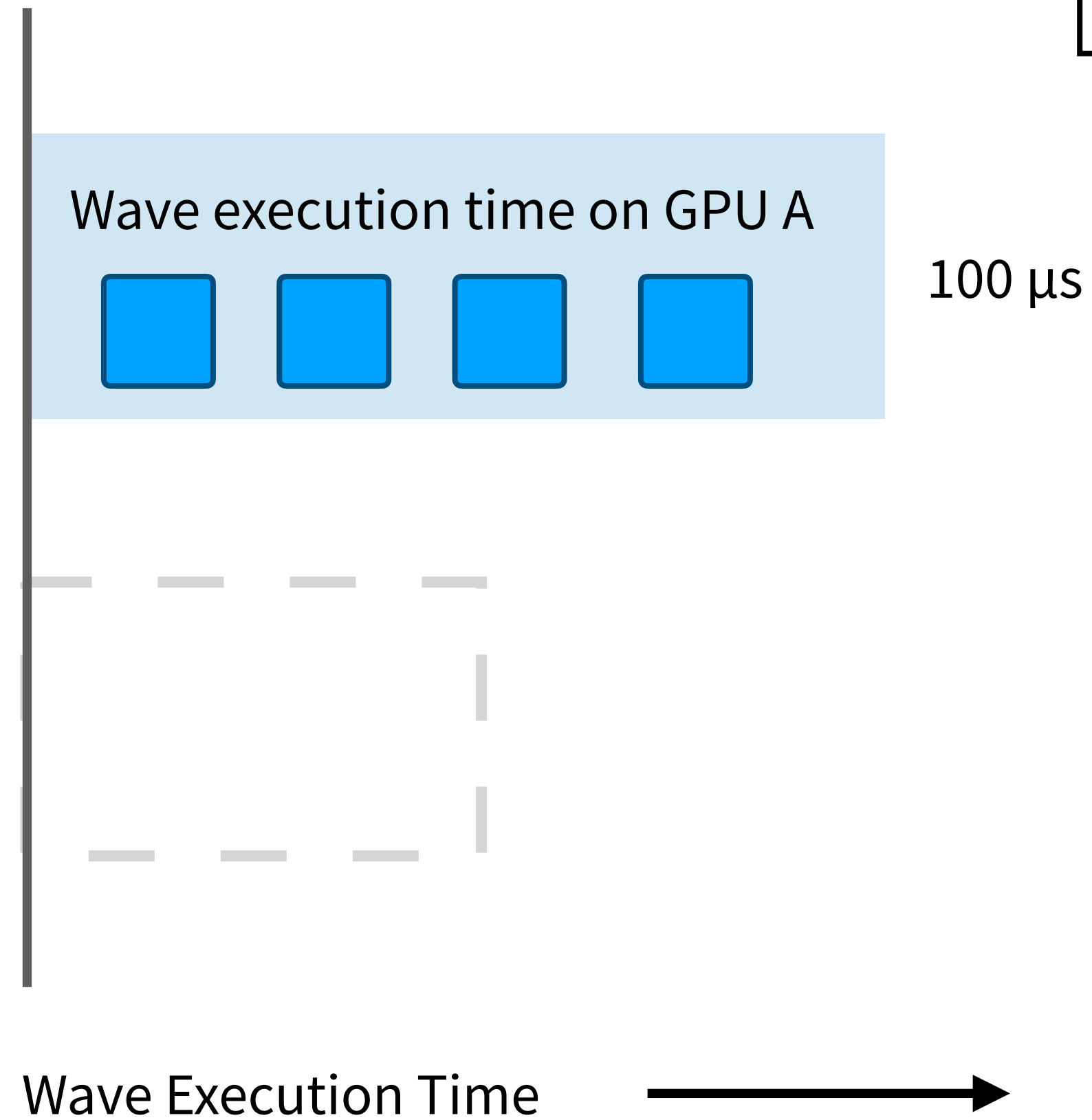
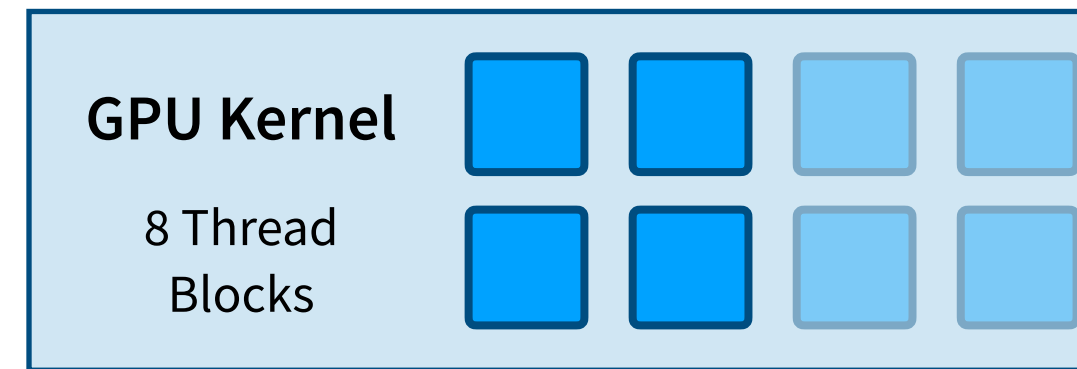
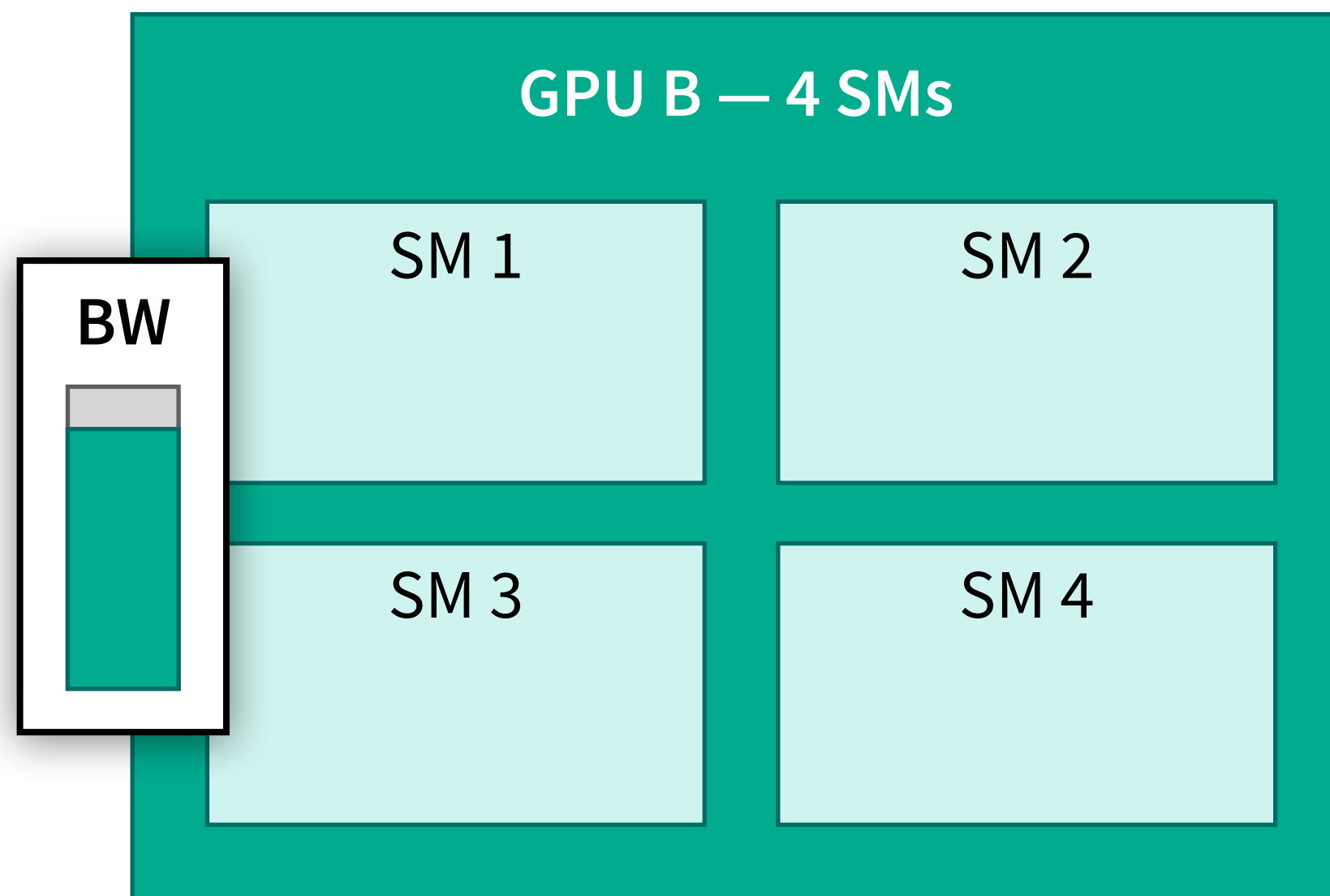
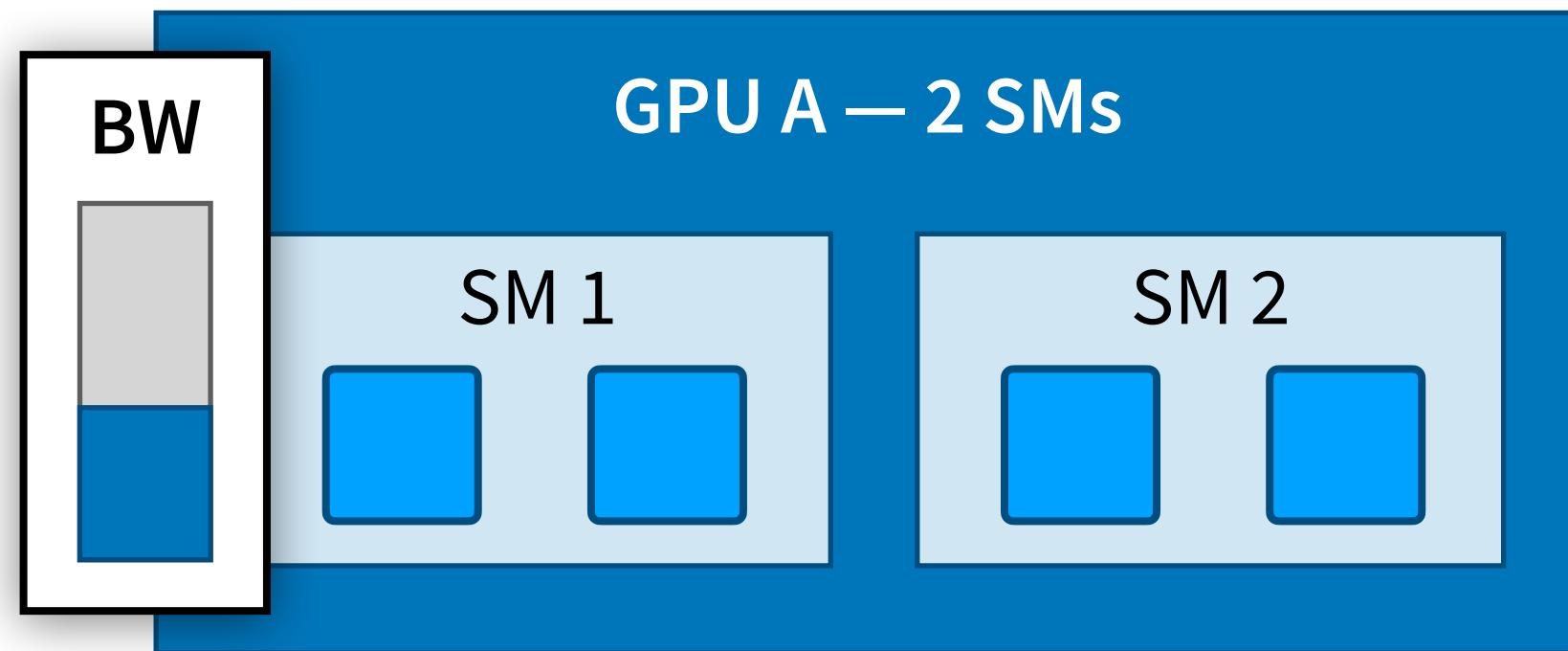
Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

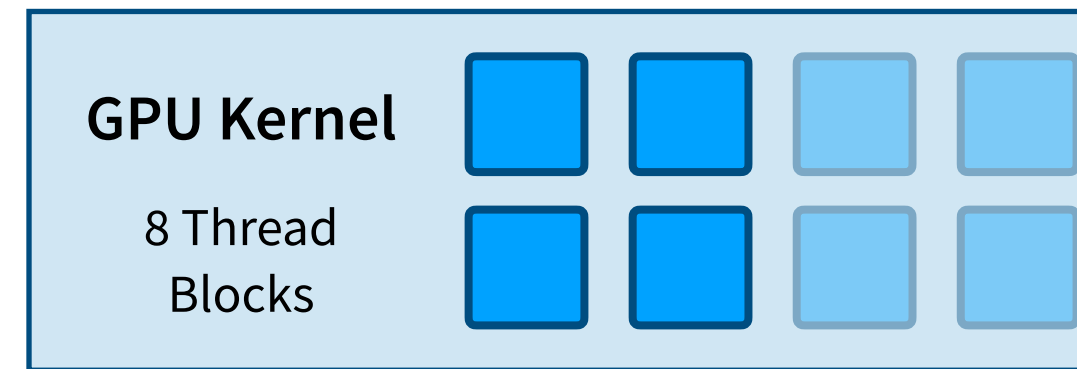
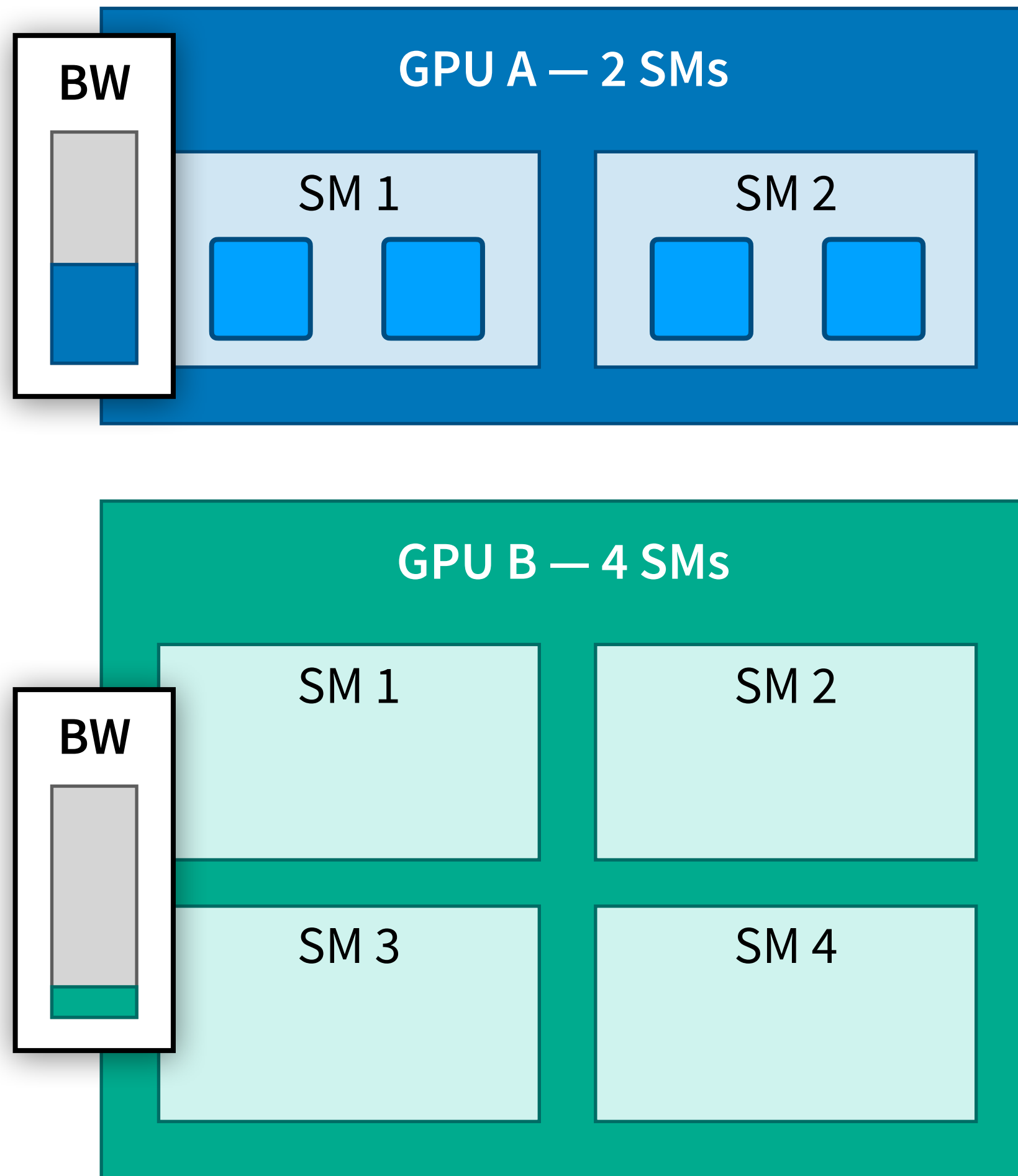


Wave scaling

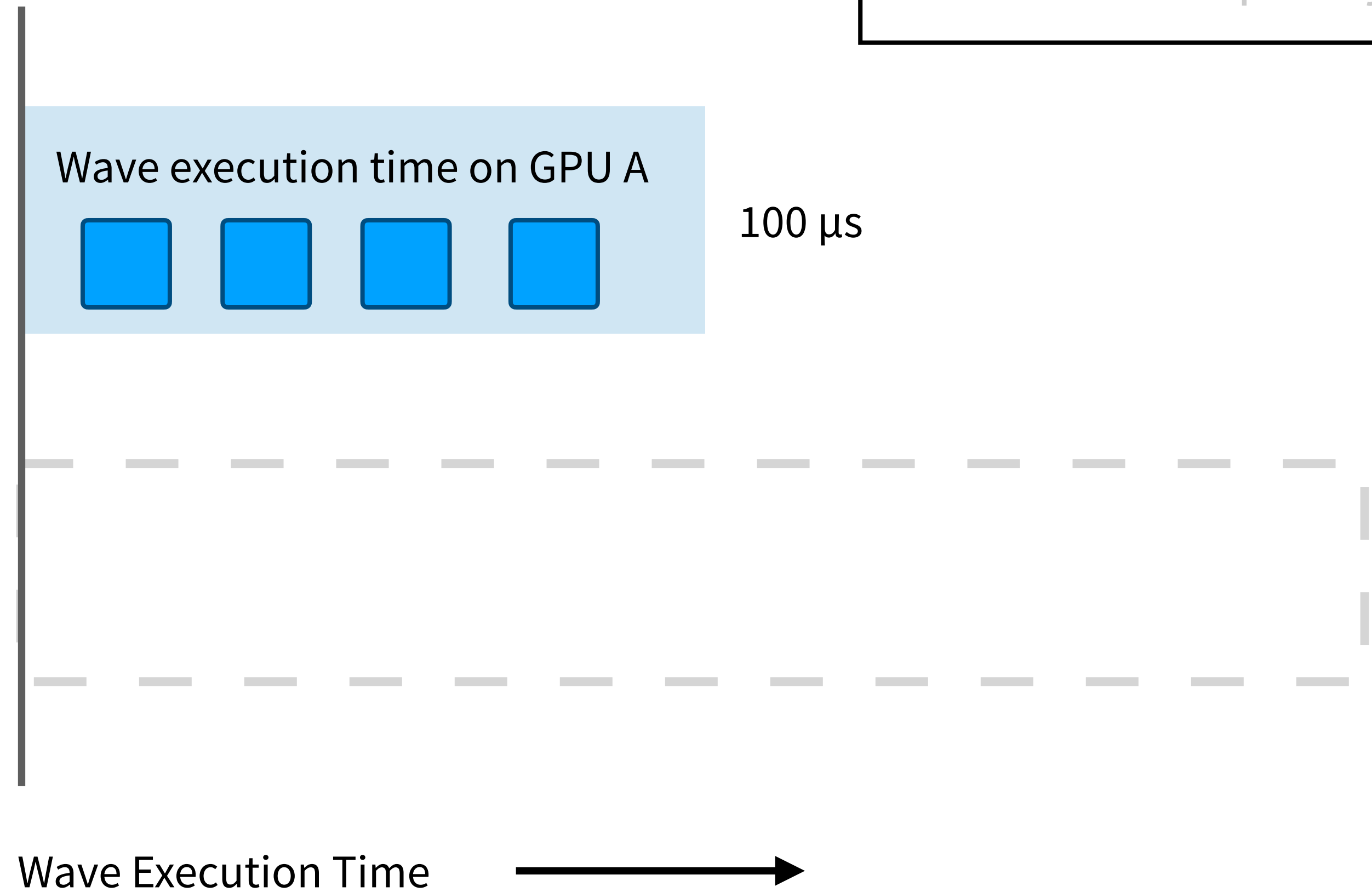


- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

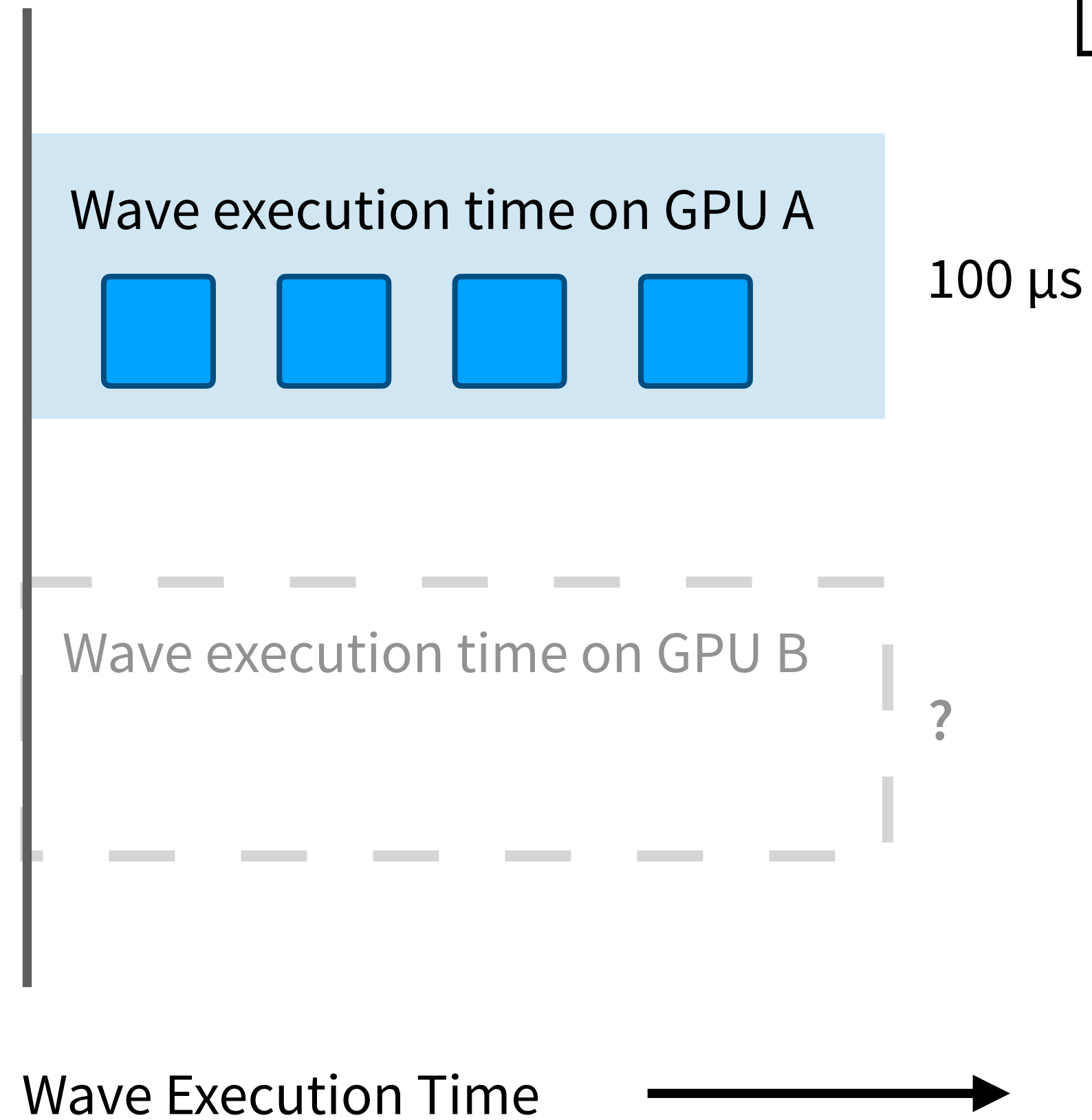
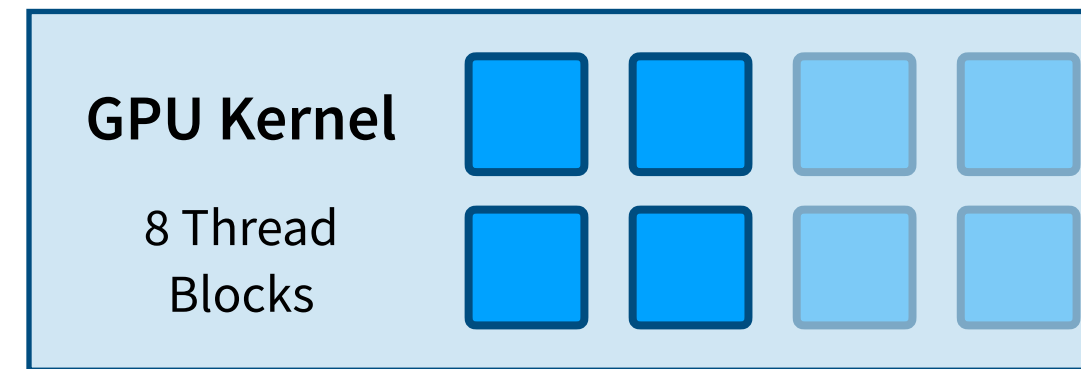
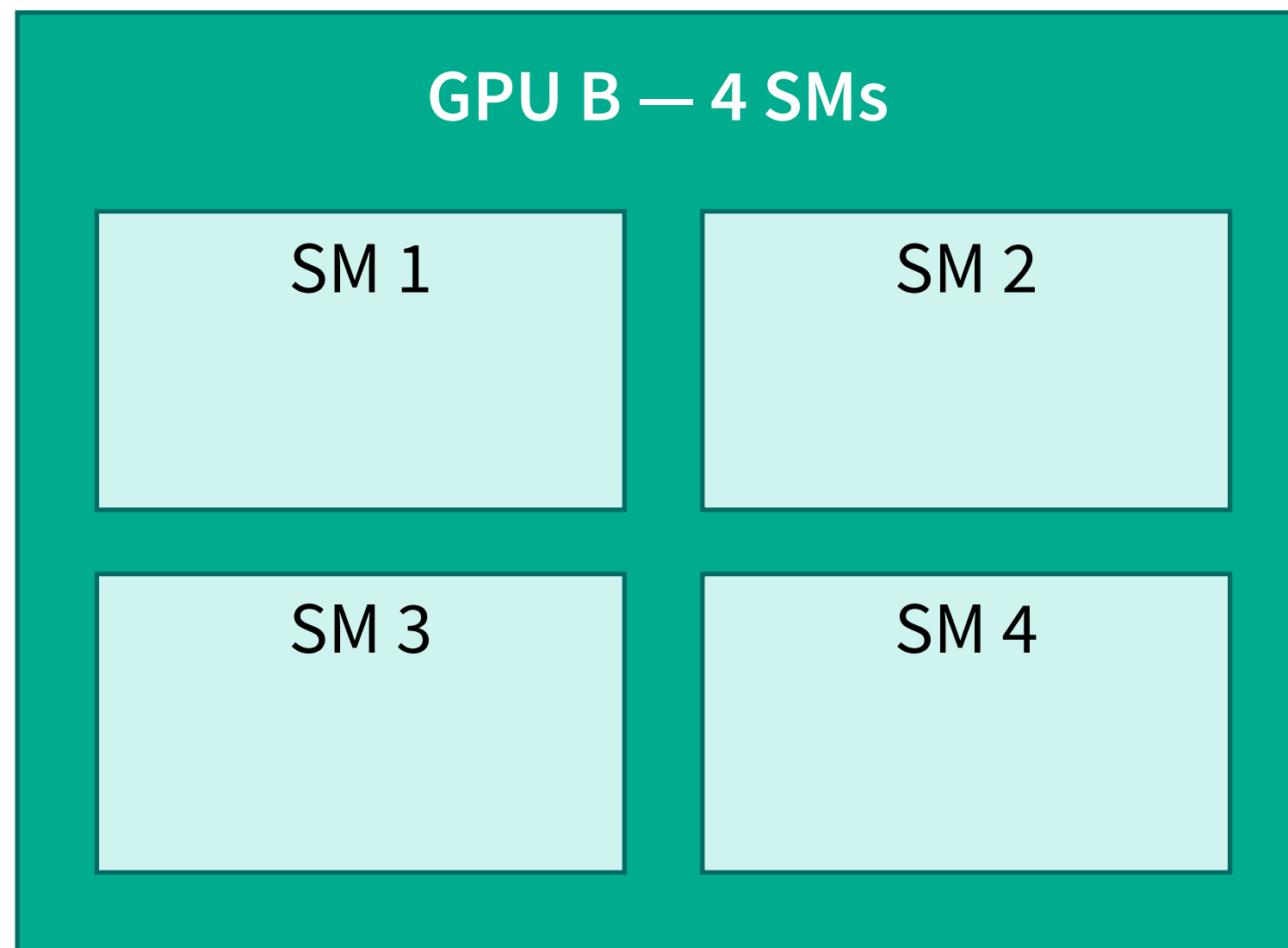
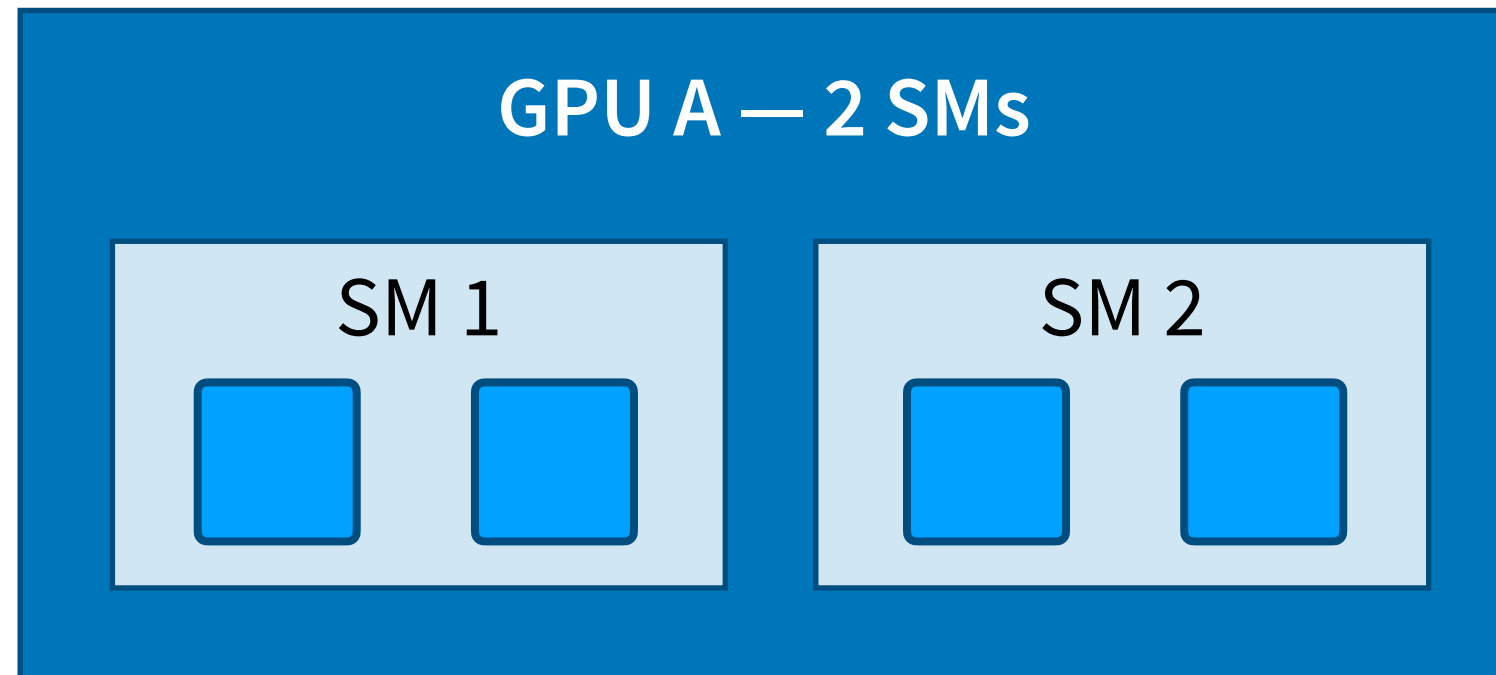
Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



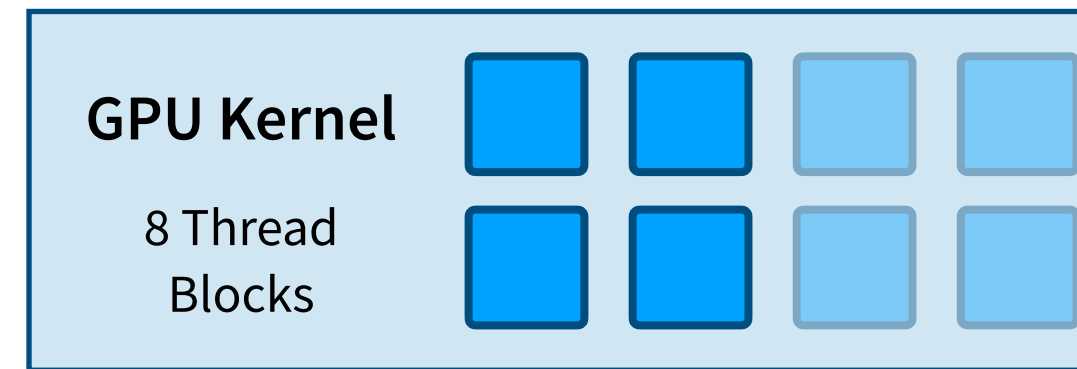
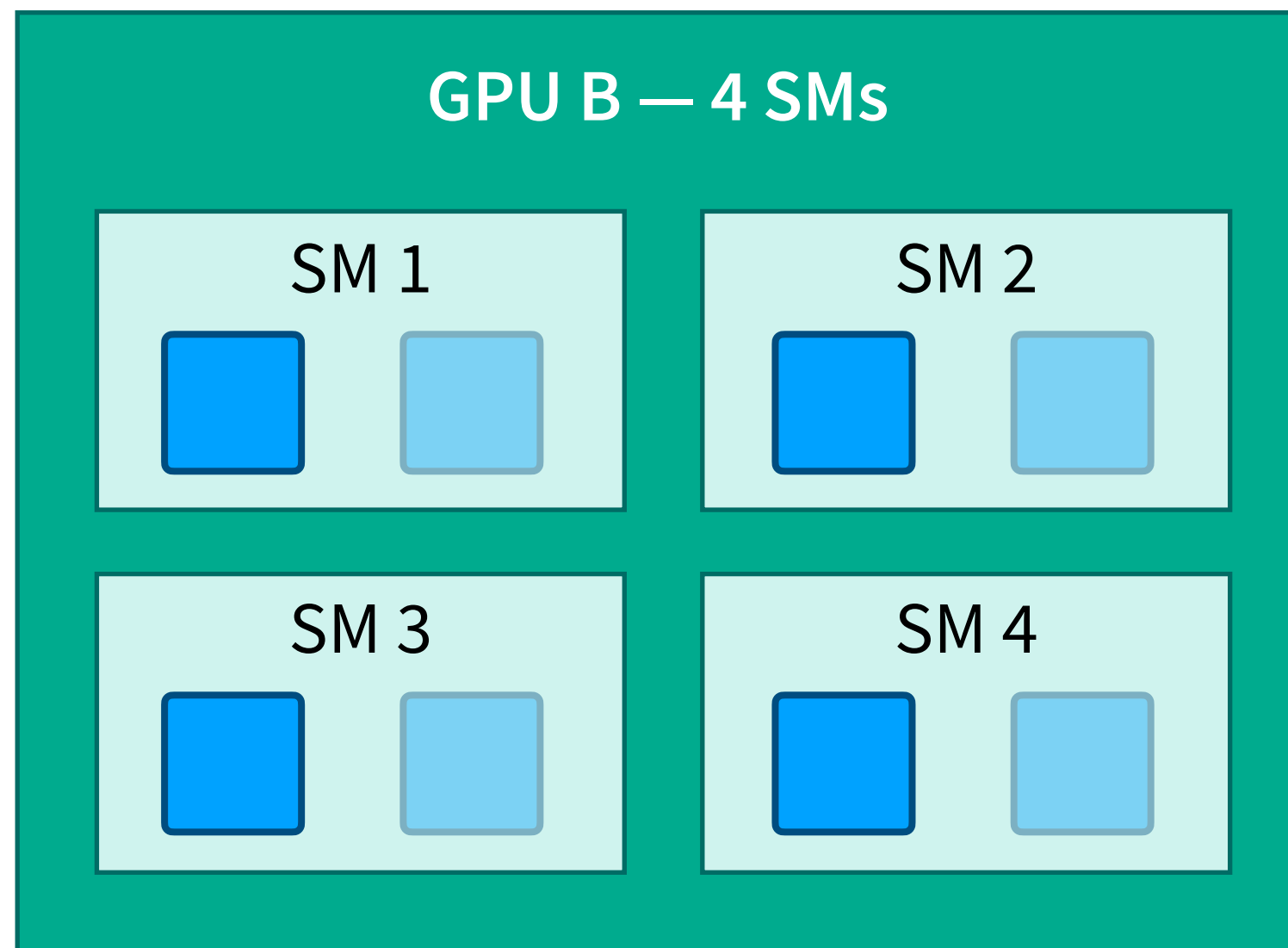
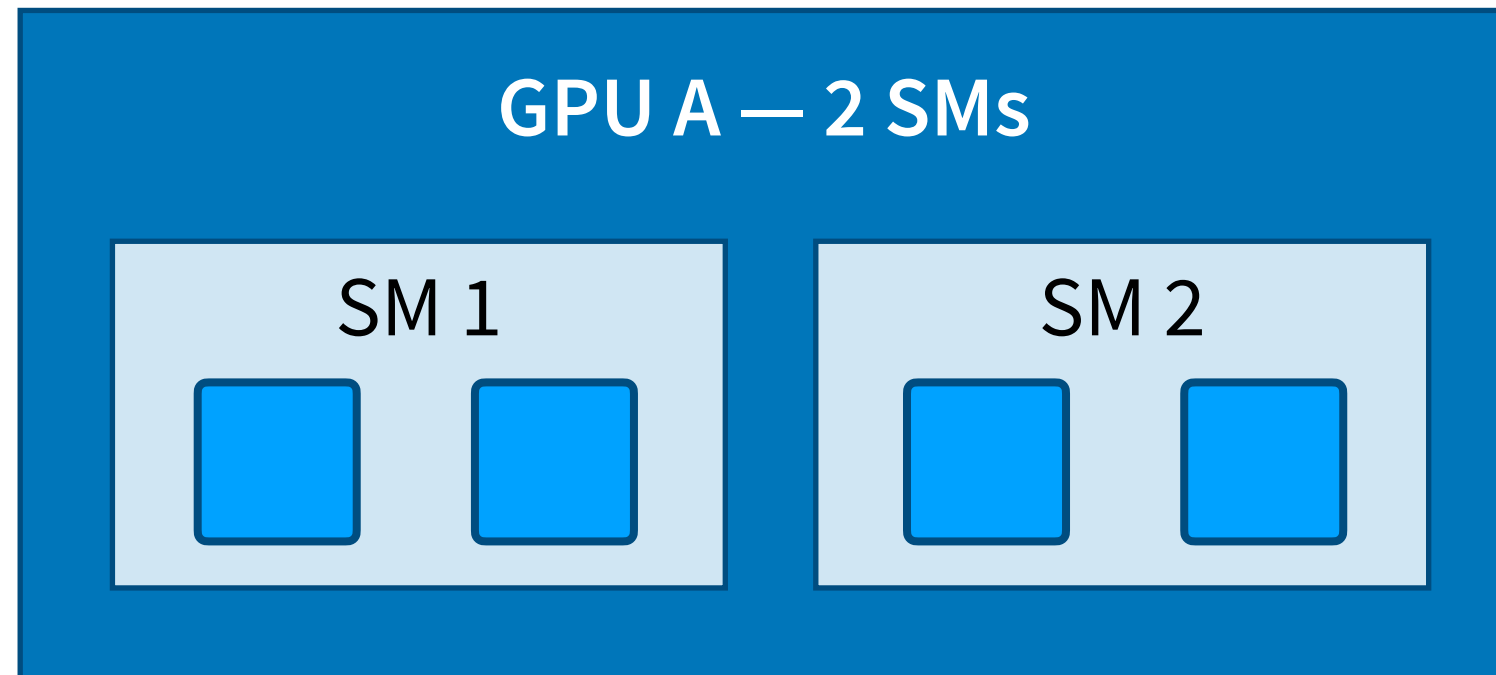
Wave scaling



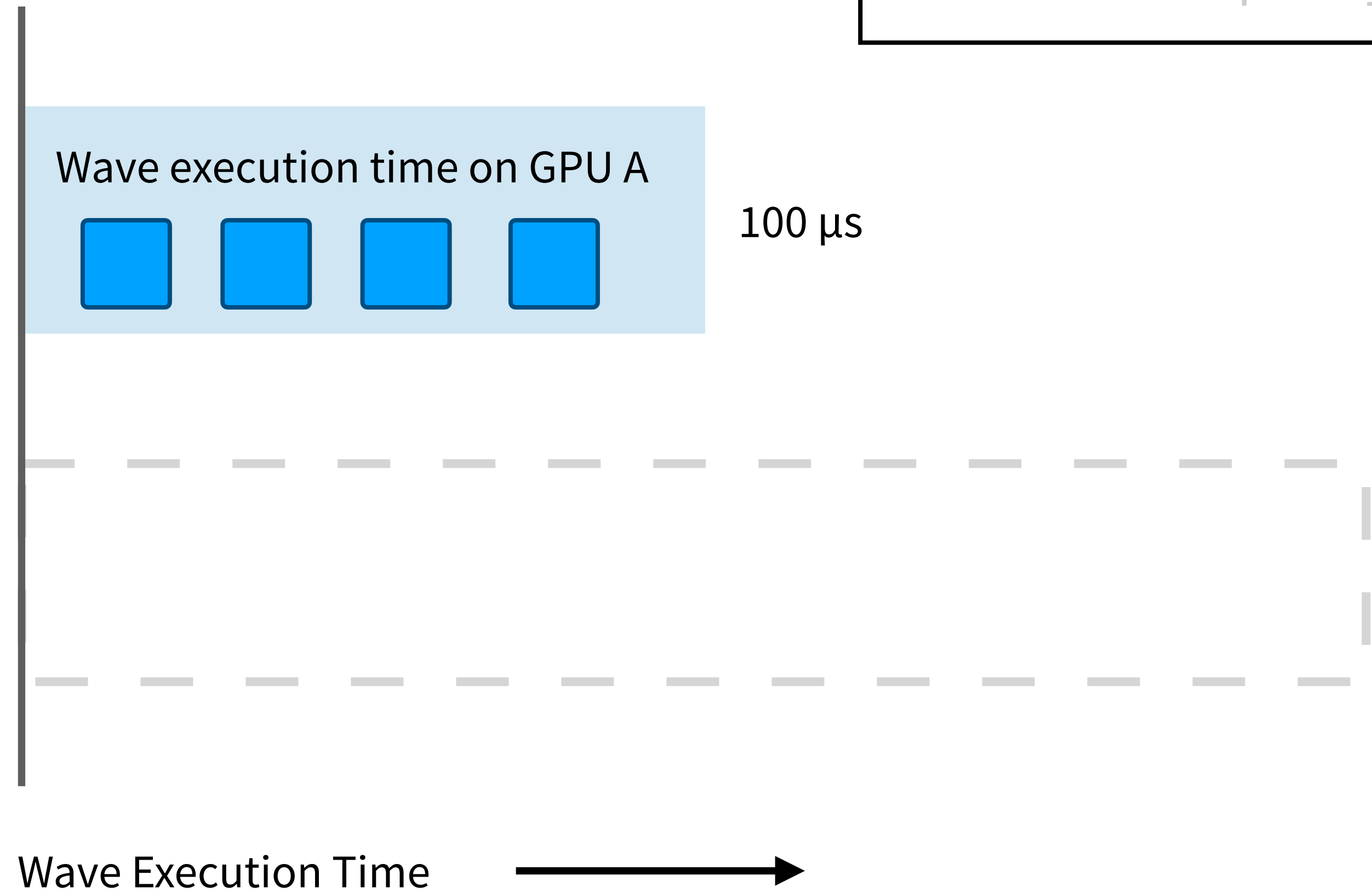
Scaling Factors

- Memory bandwidth
- Wave size
- Clock frequency

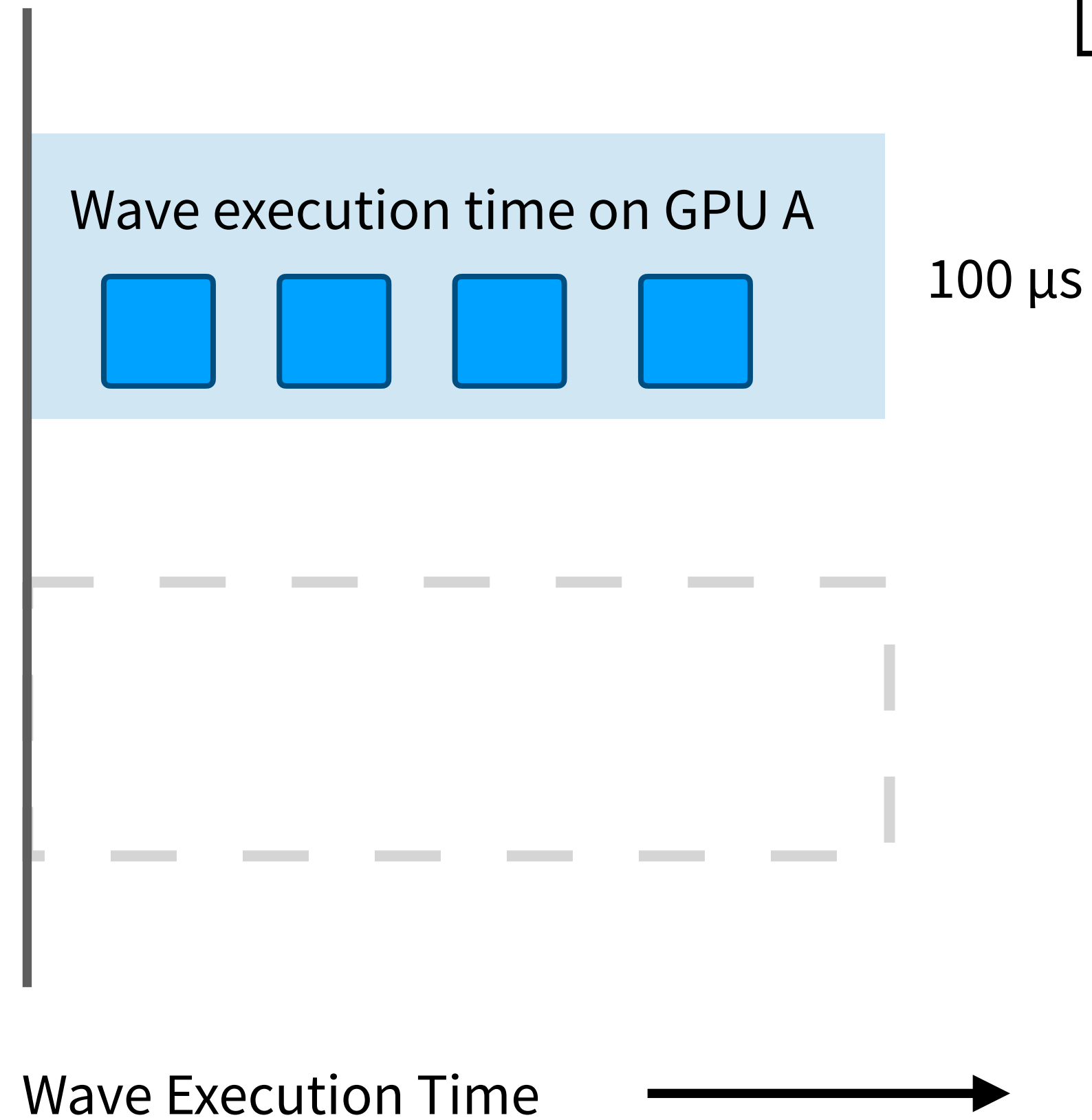
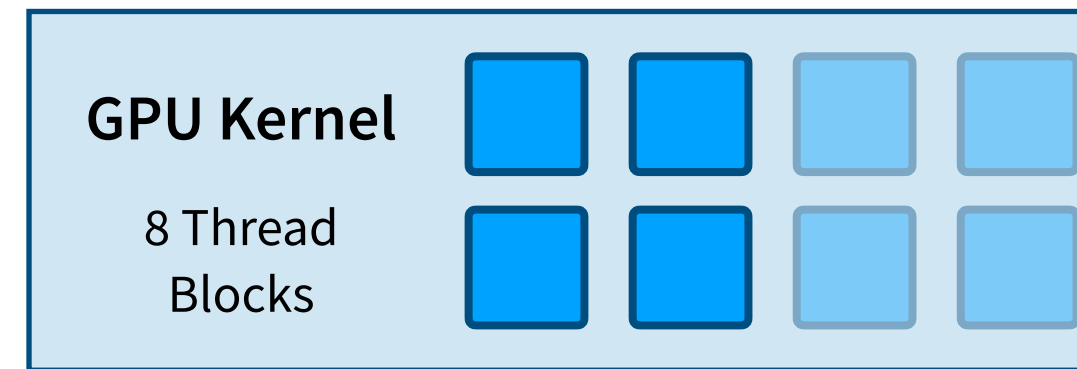
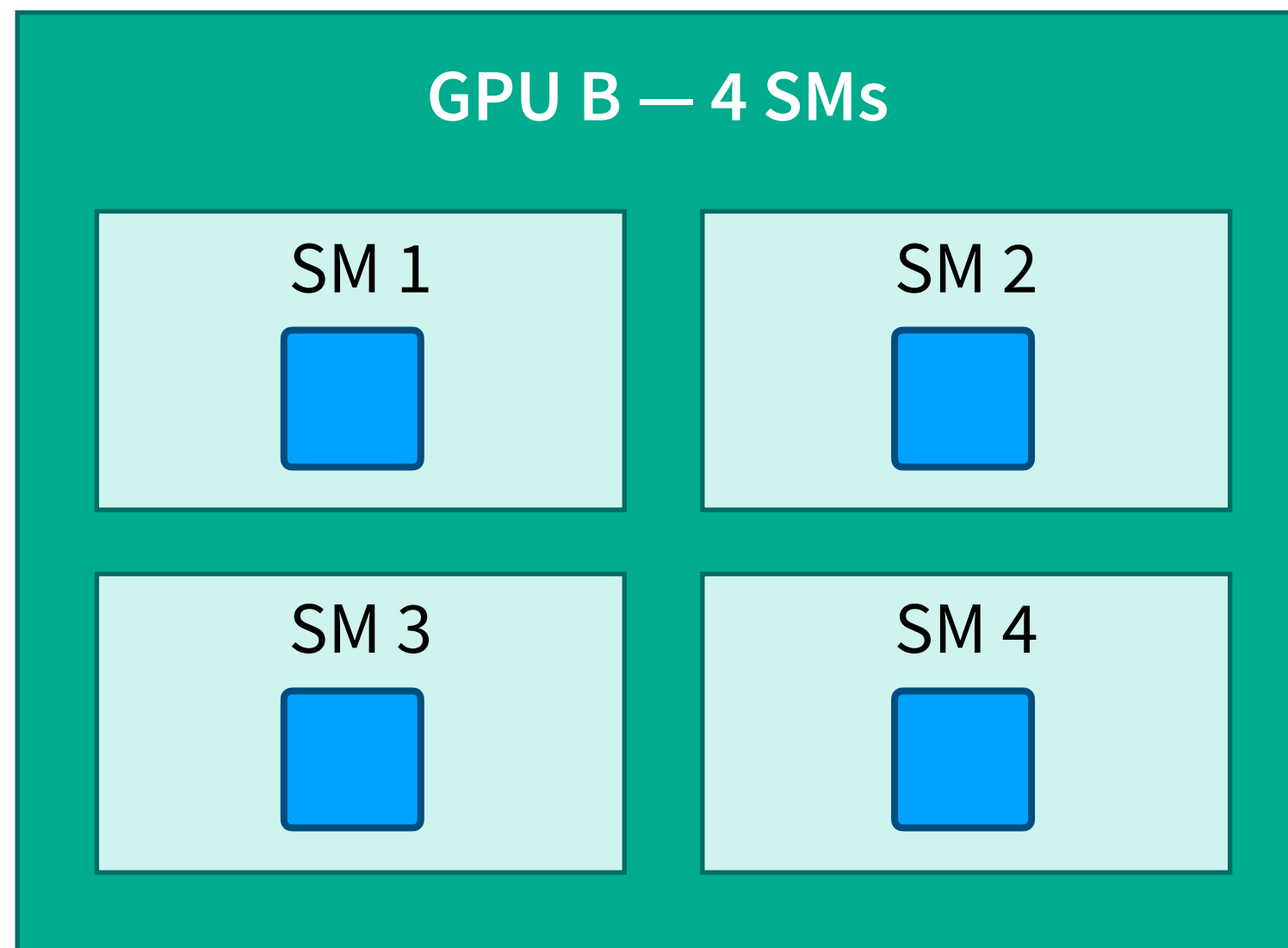
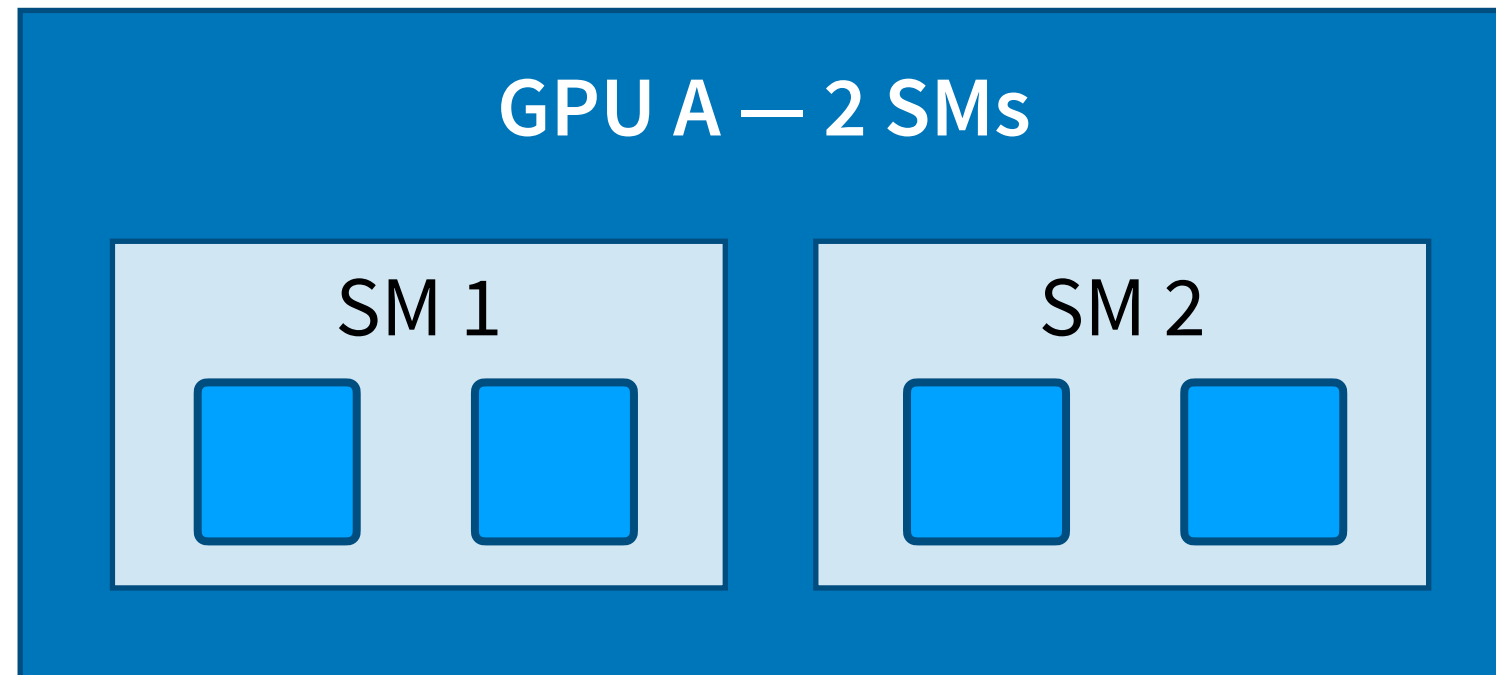
Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



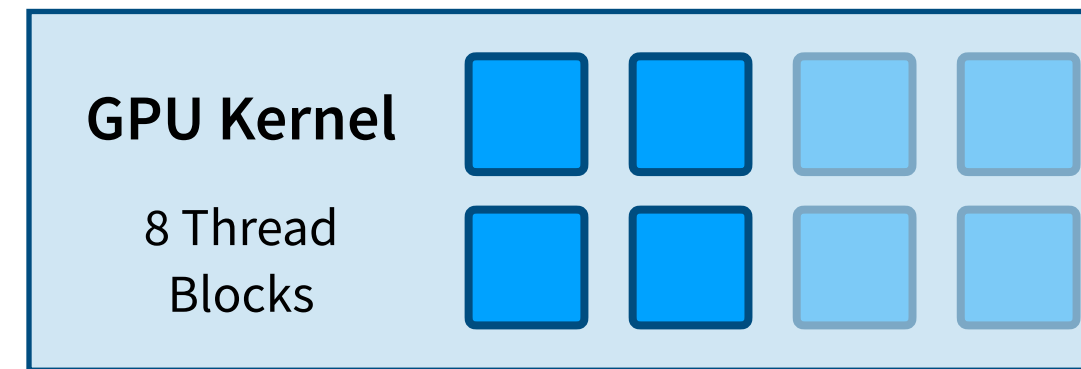
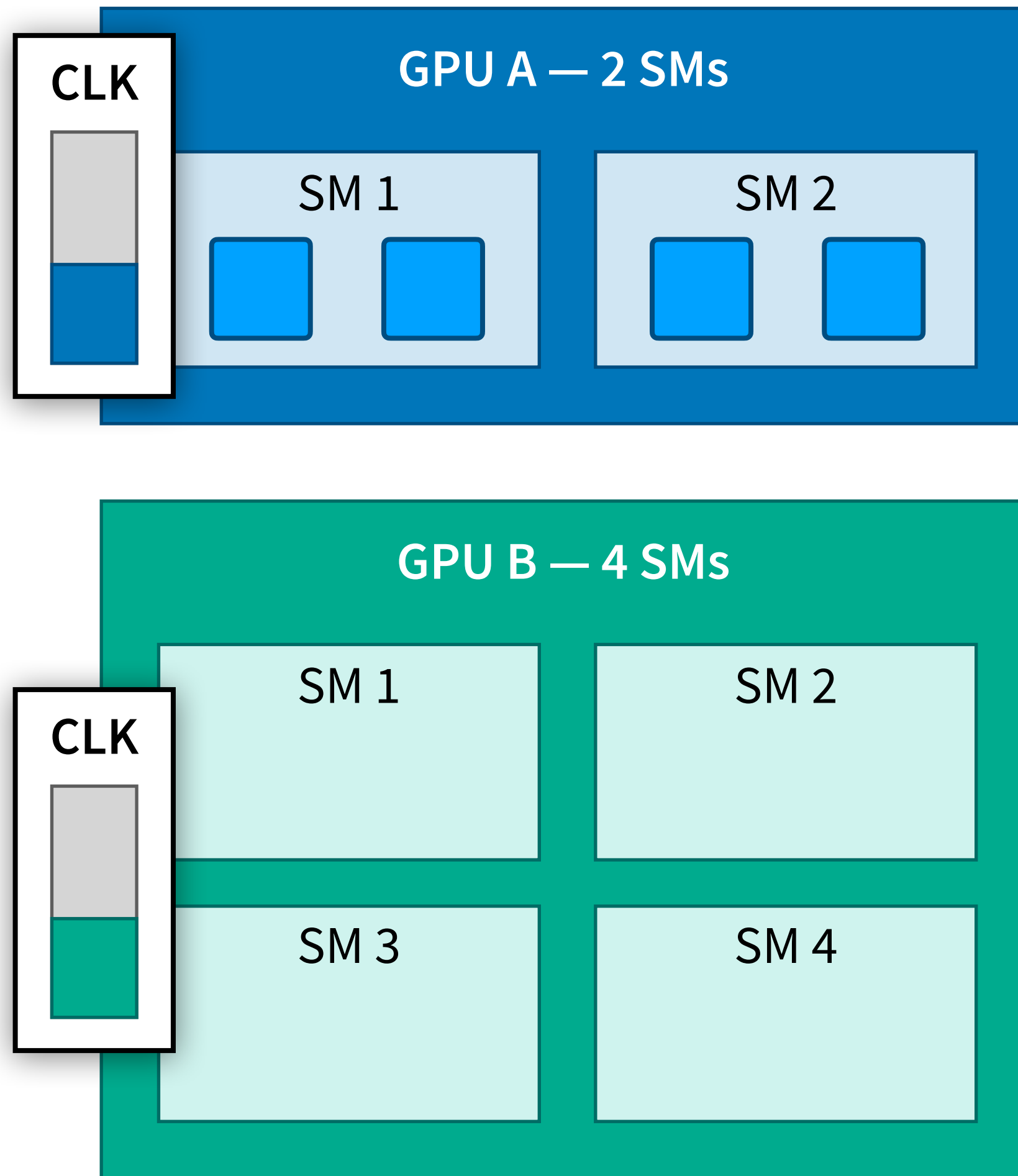
Wave scaling



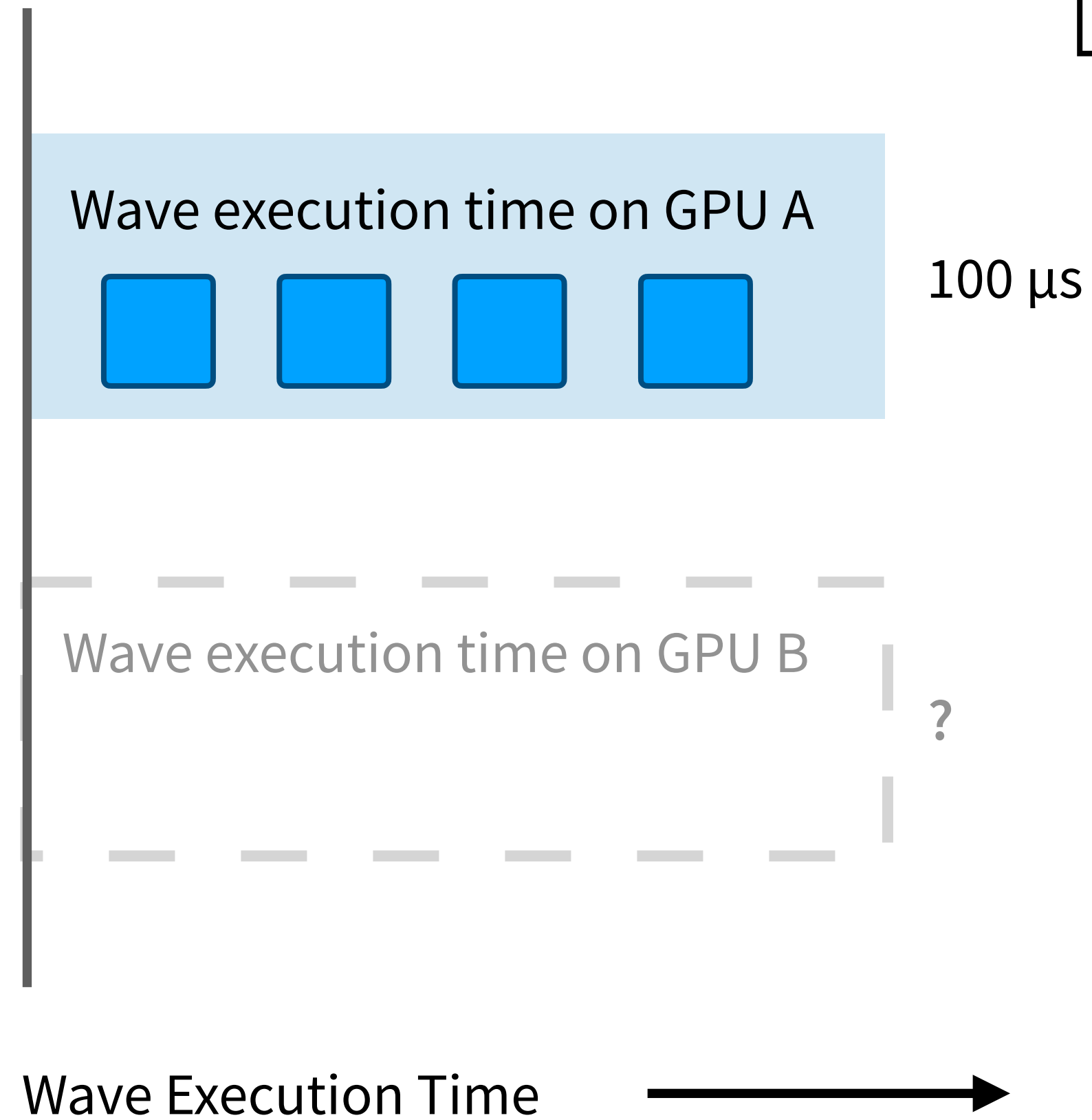
Scaling Factors

- Memory bandwidth
- Wave size
- Clock frequency

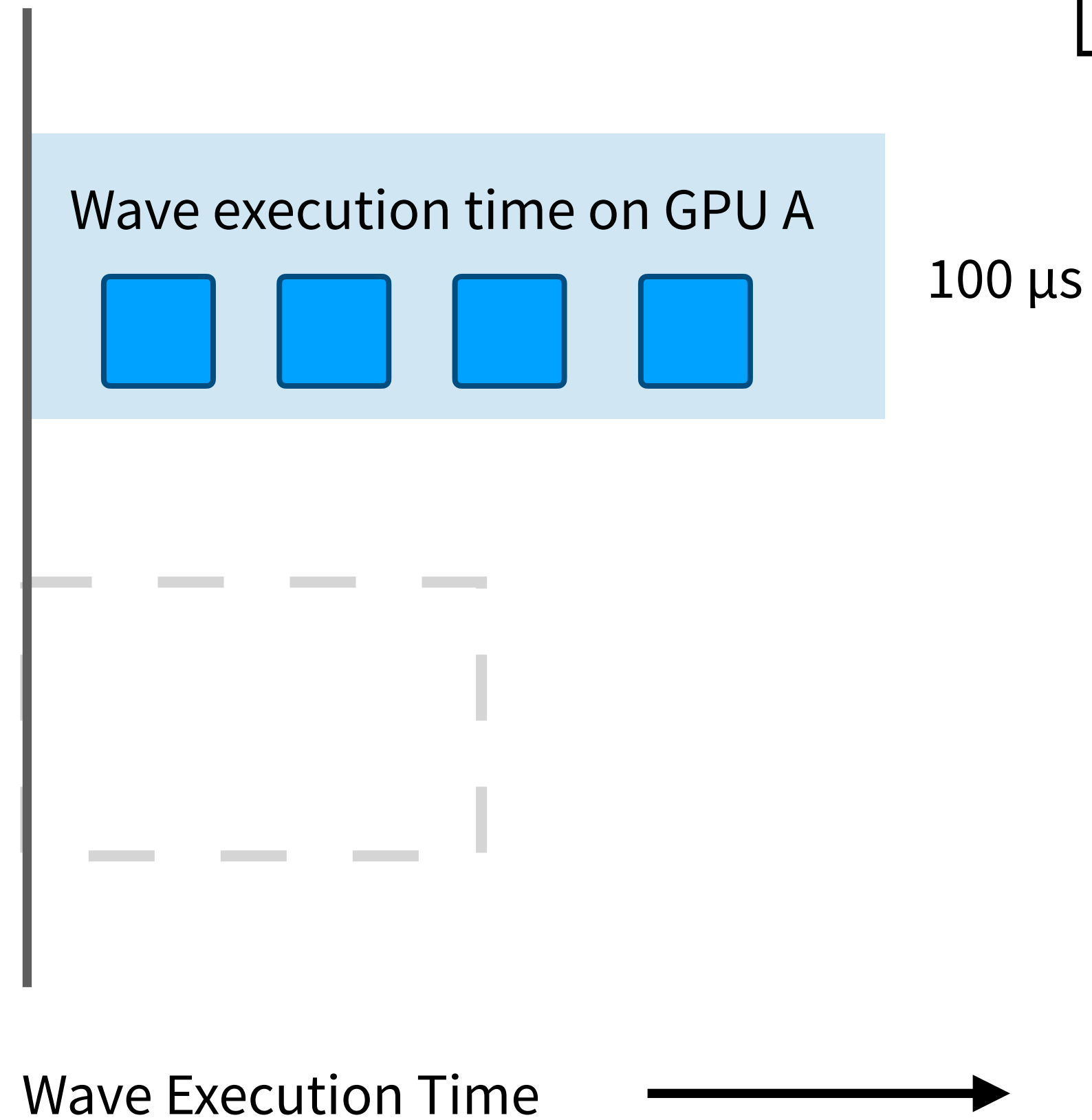
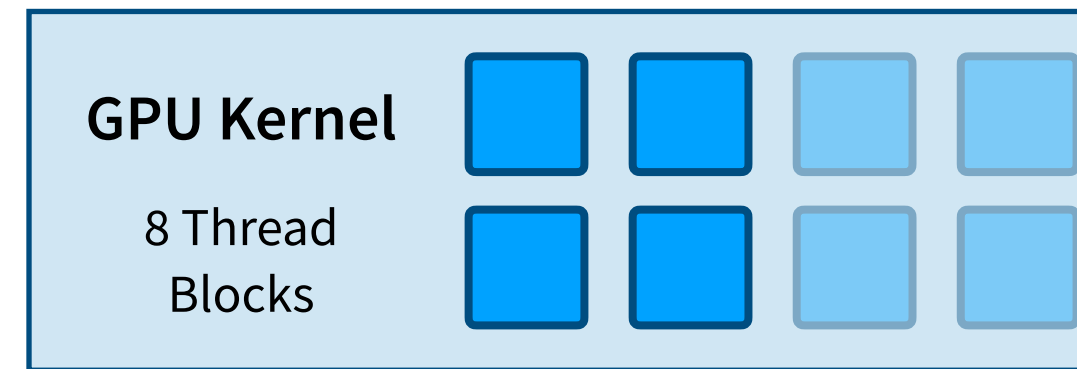
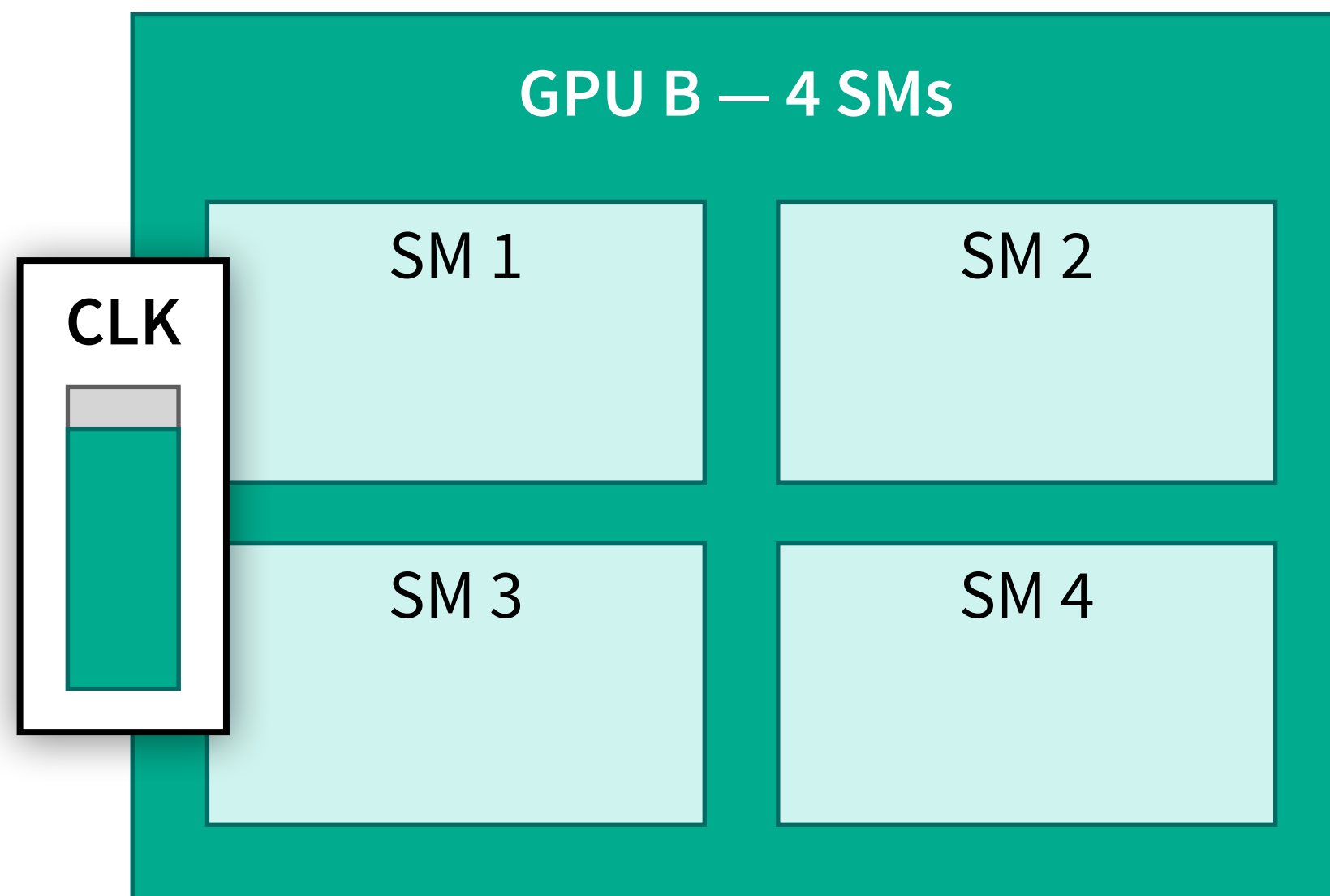
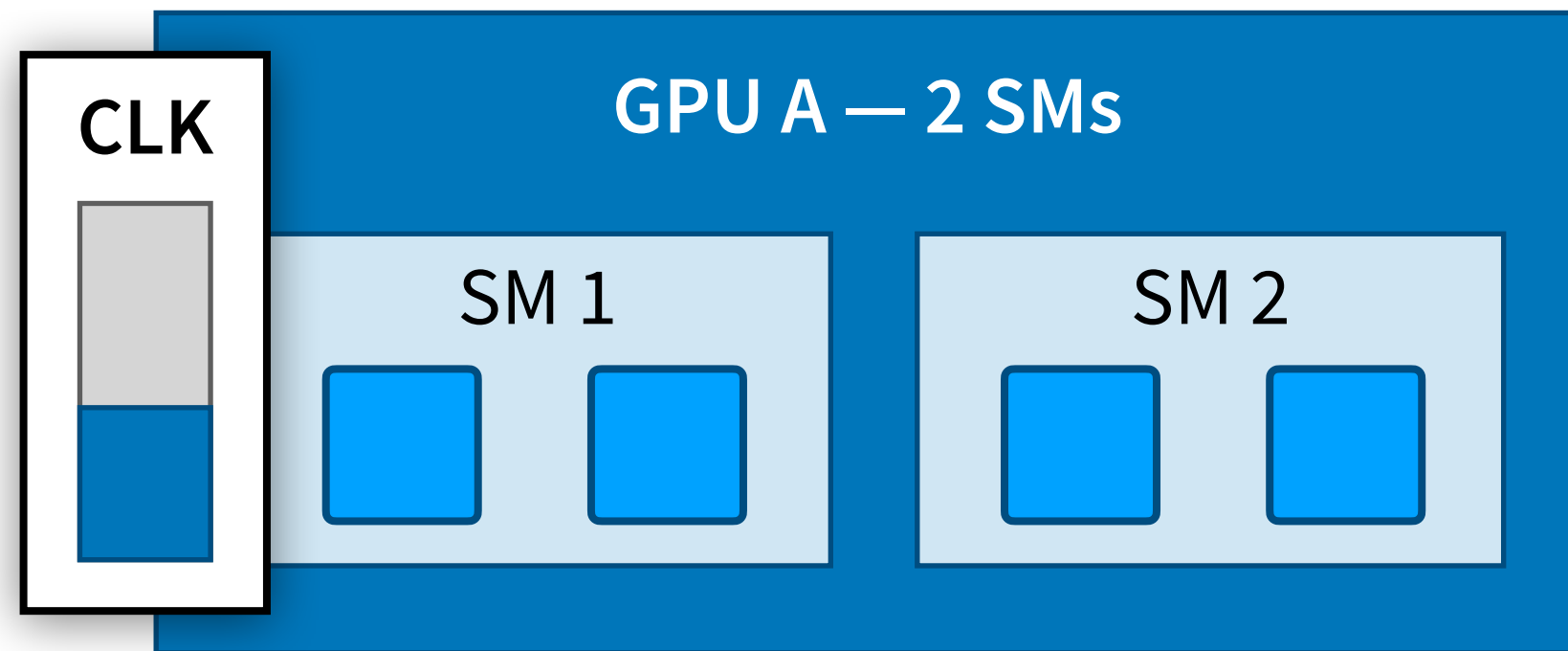
Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

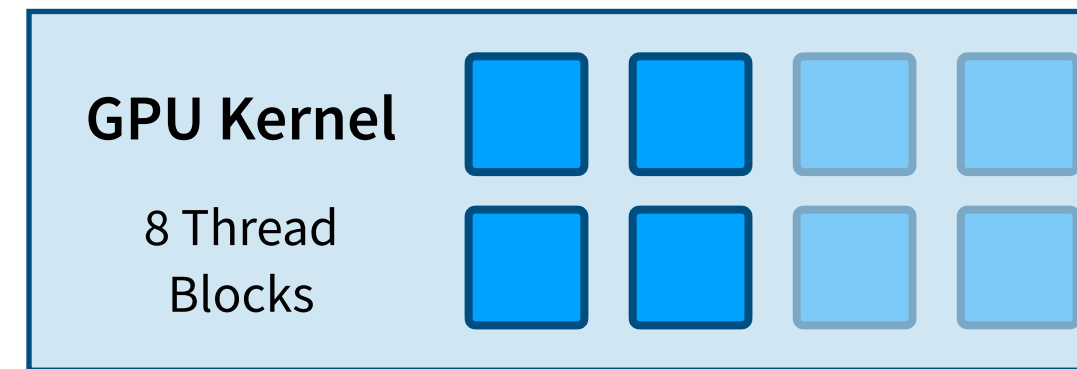
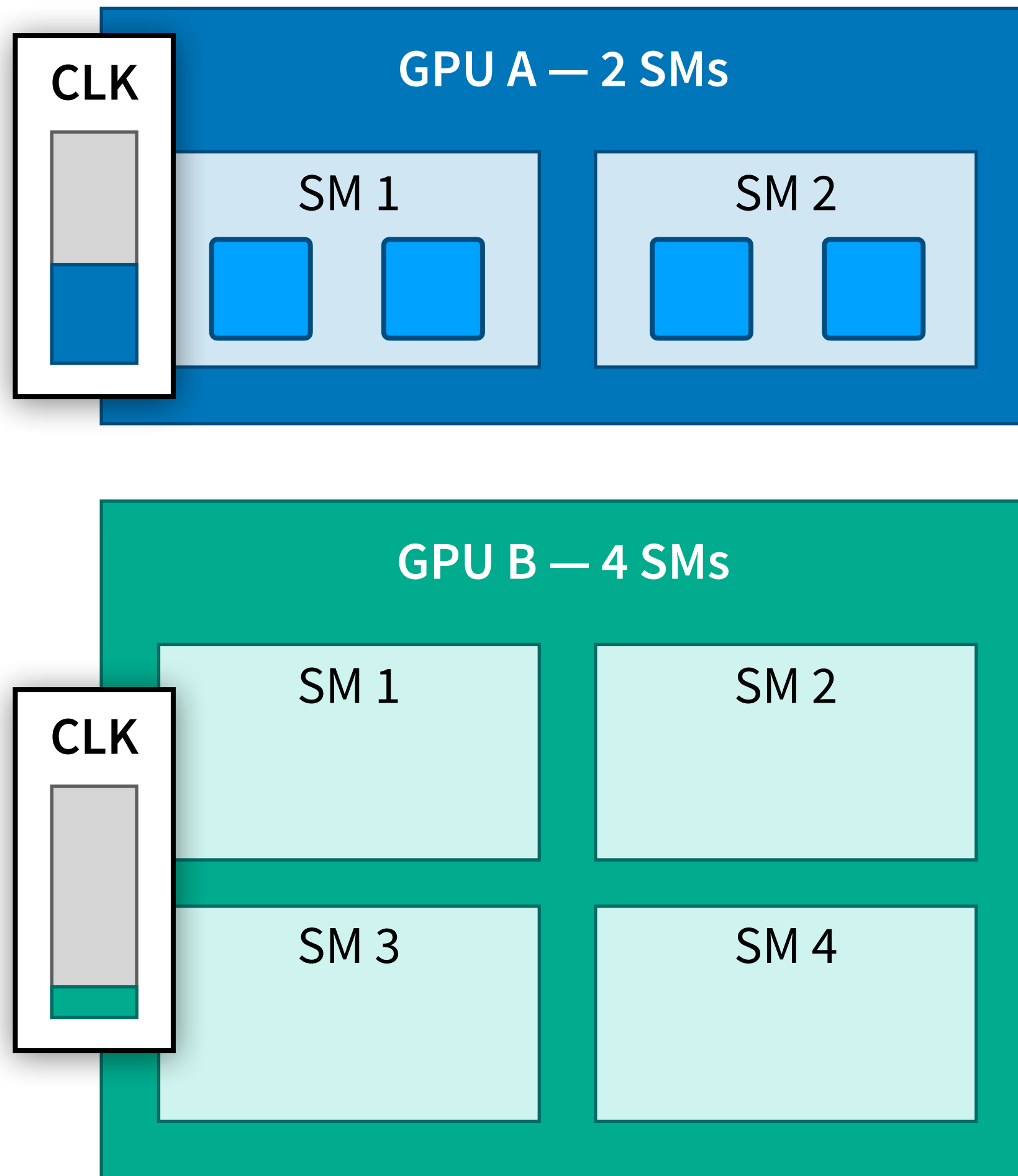


Wave scaling

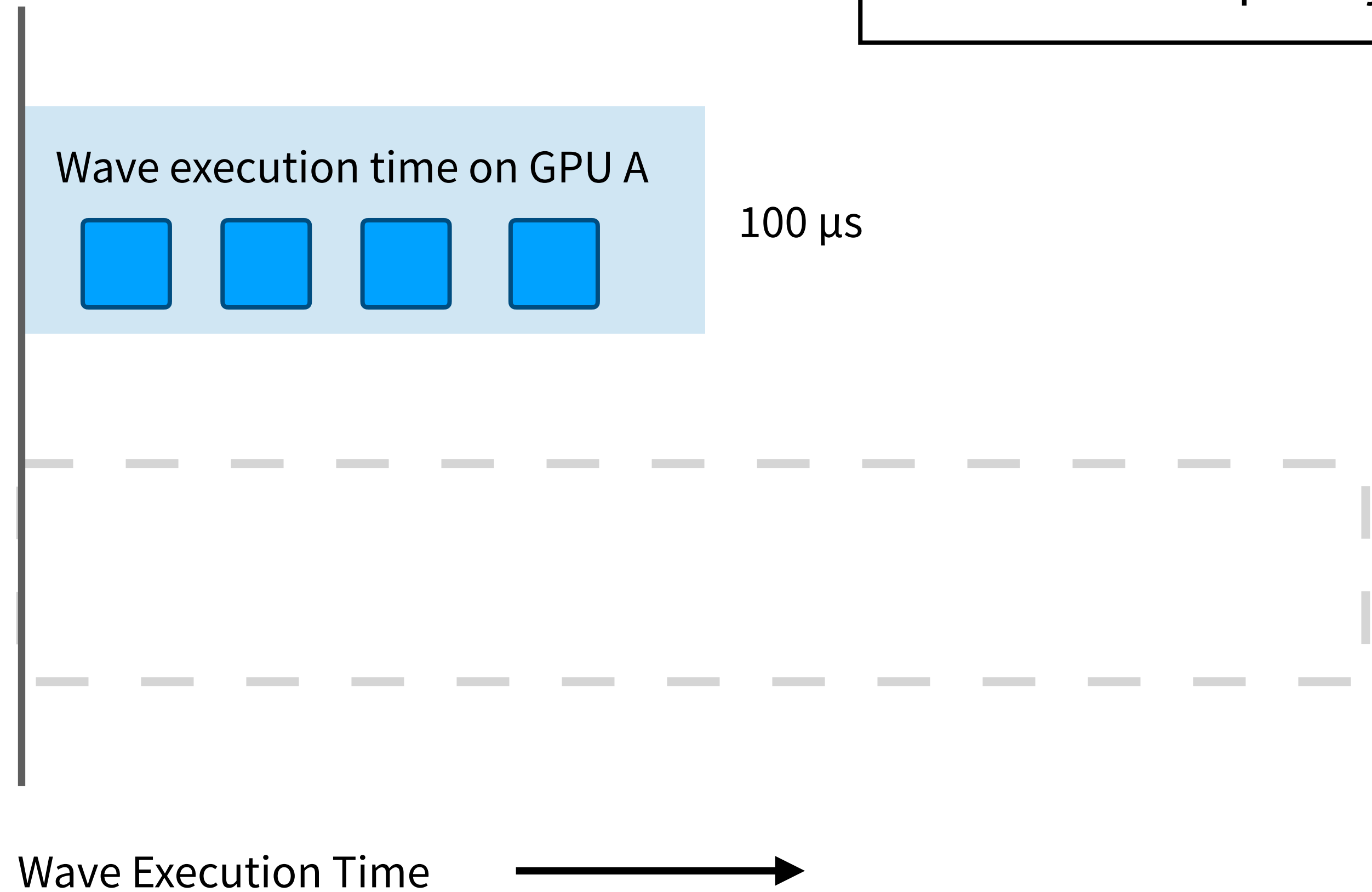


- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

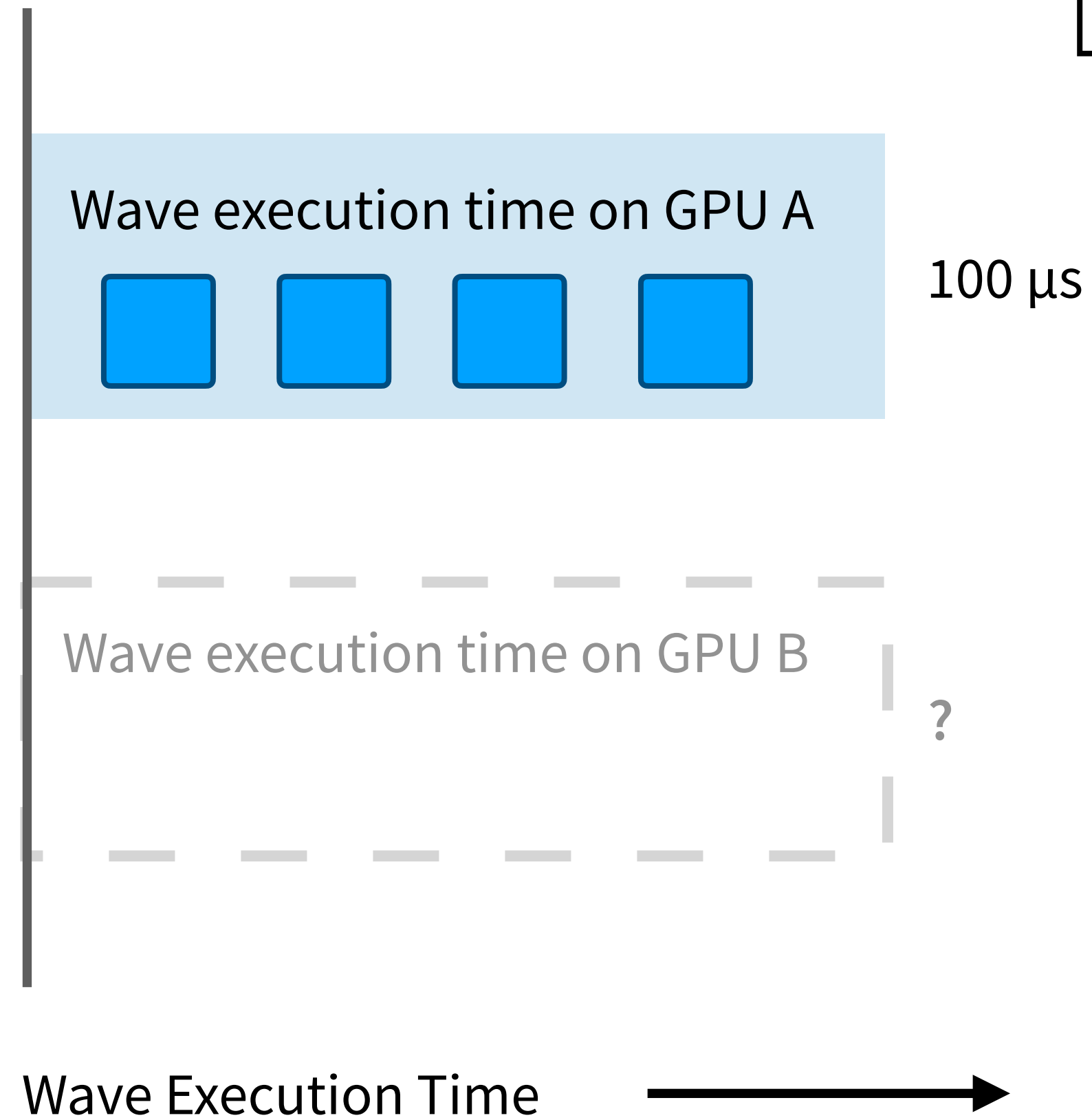
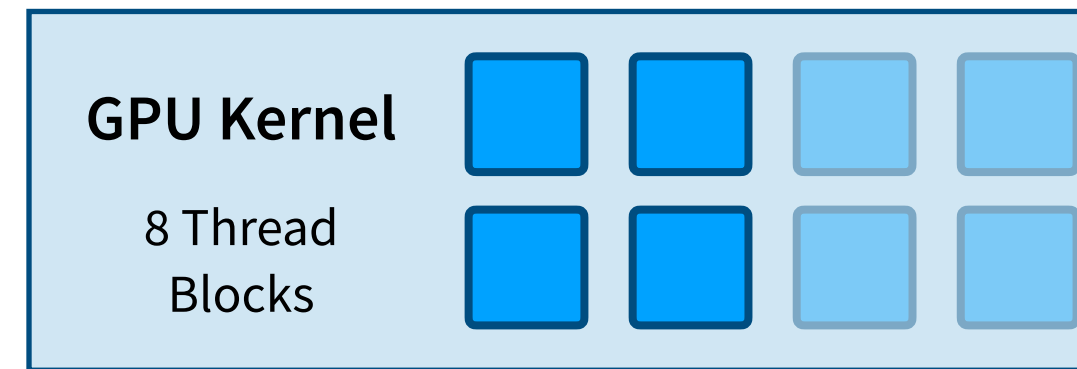
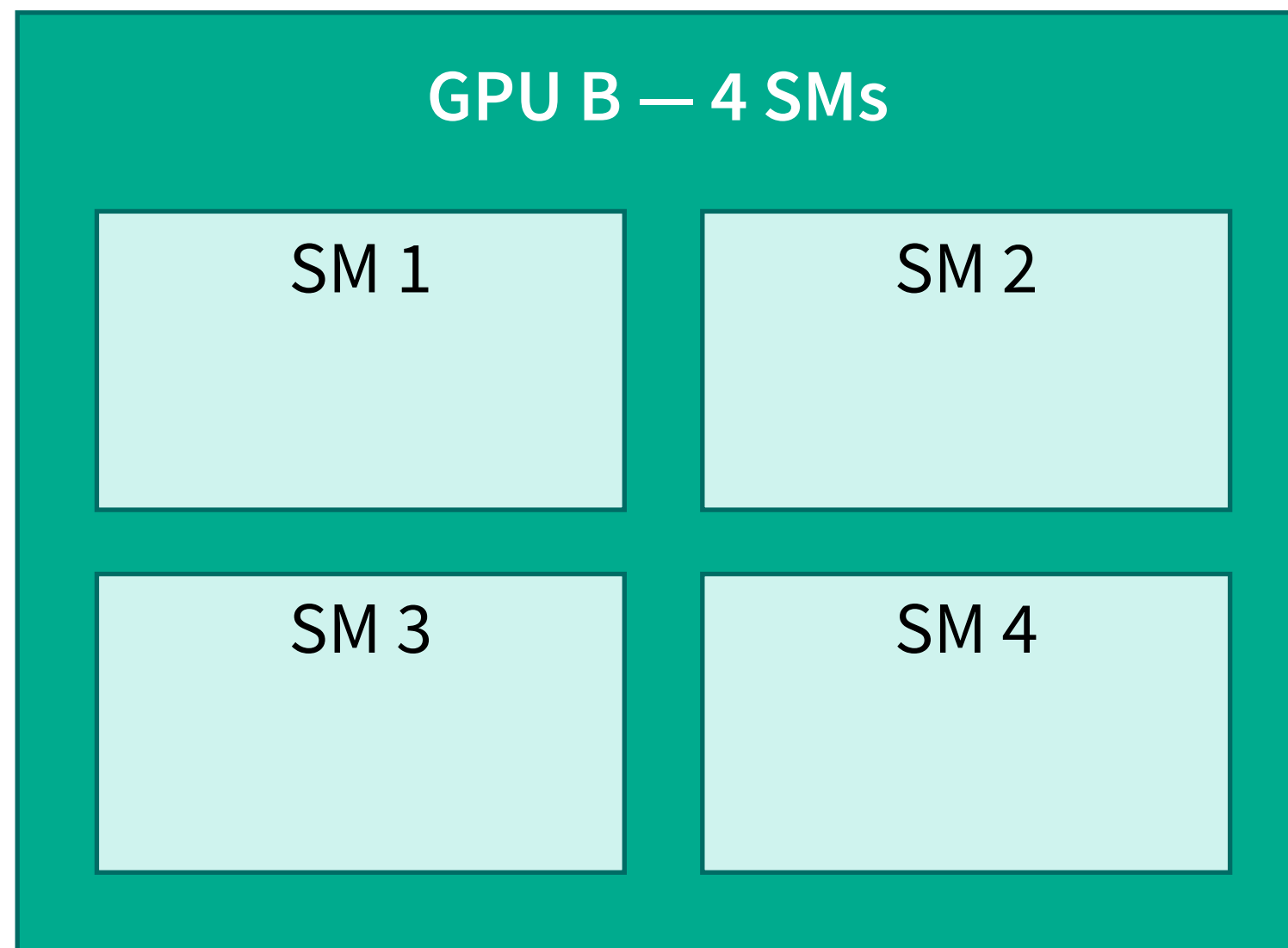
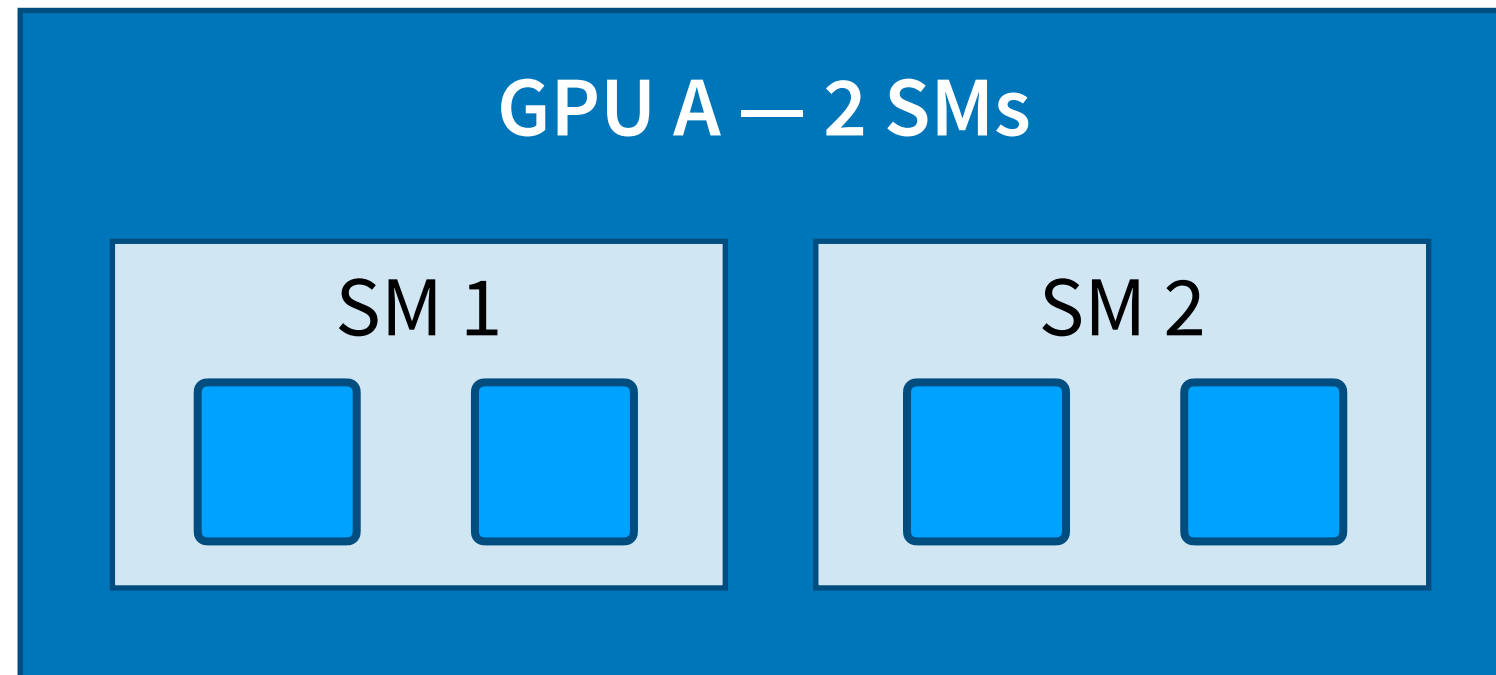
Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

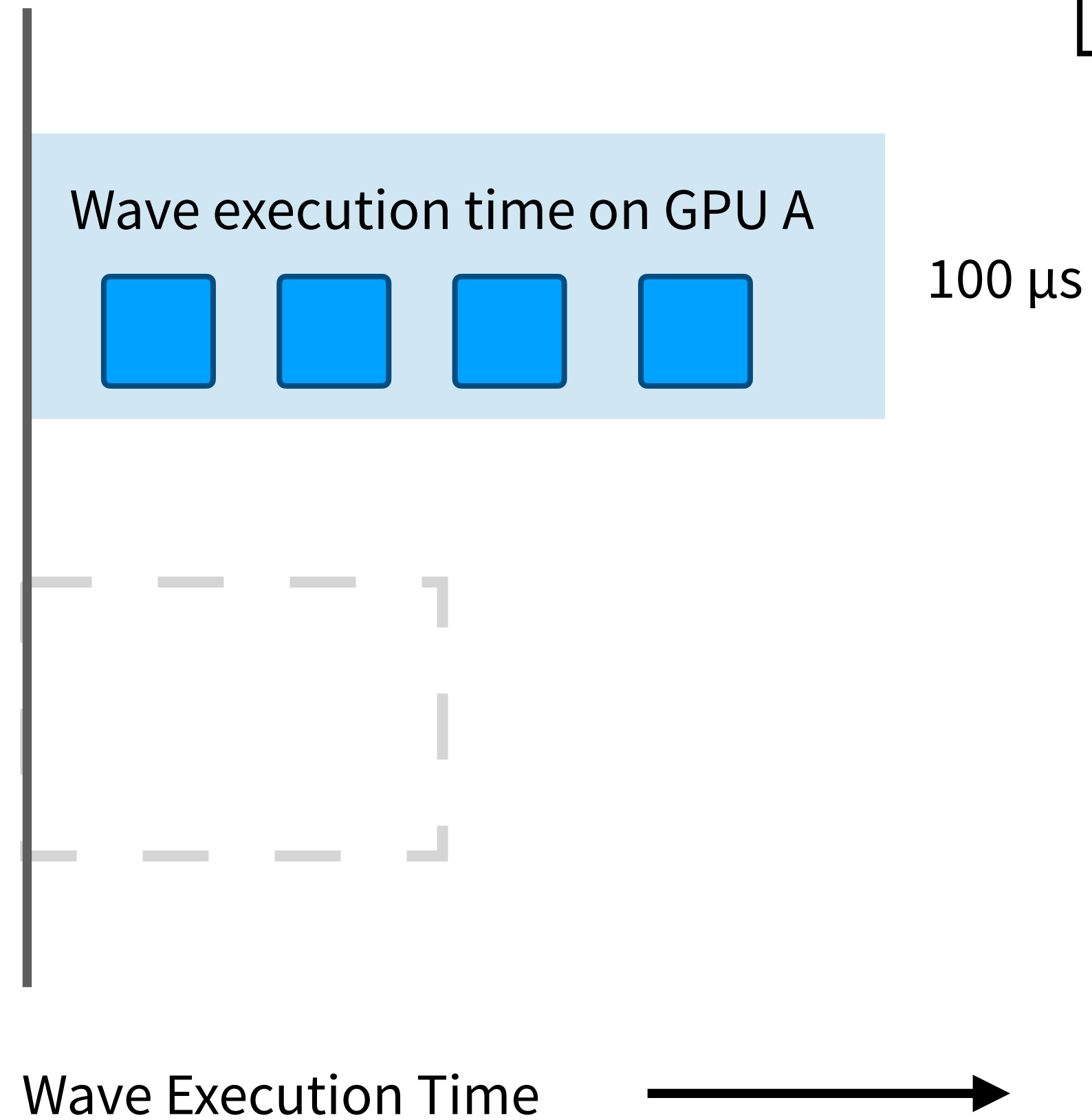
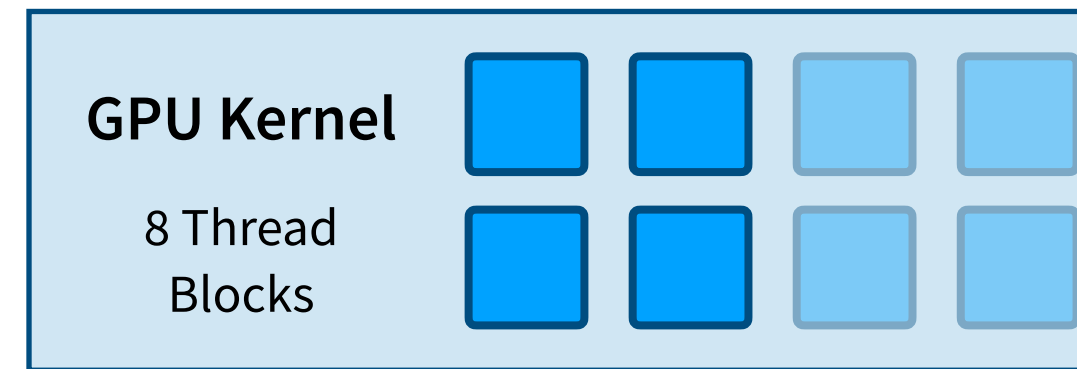
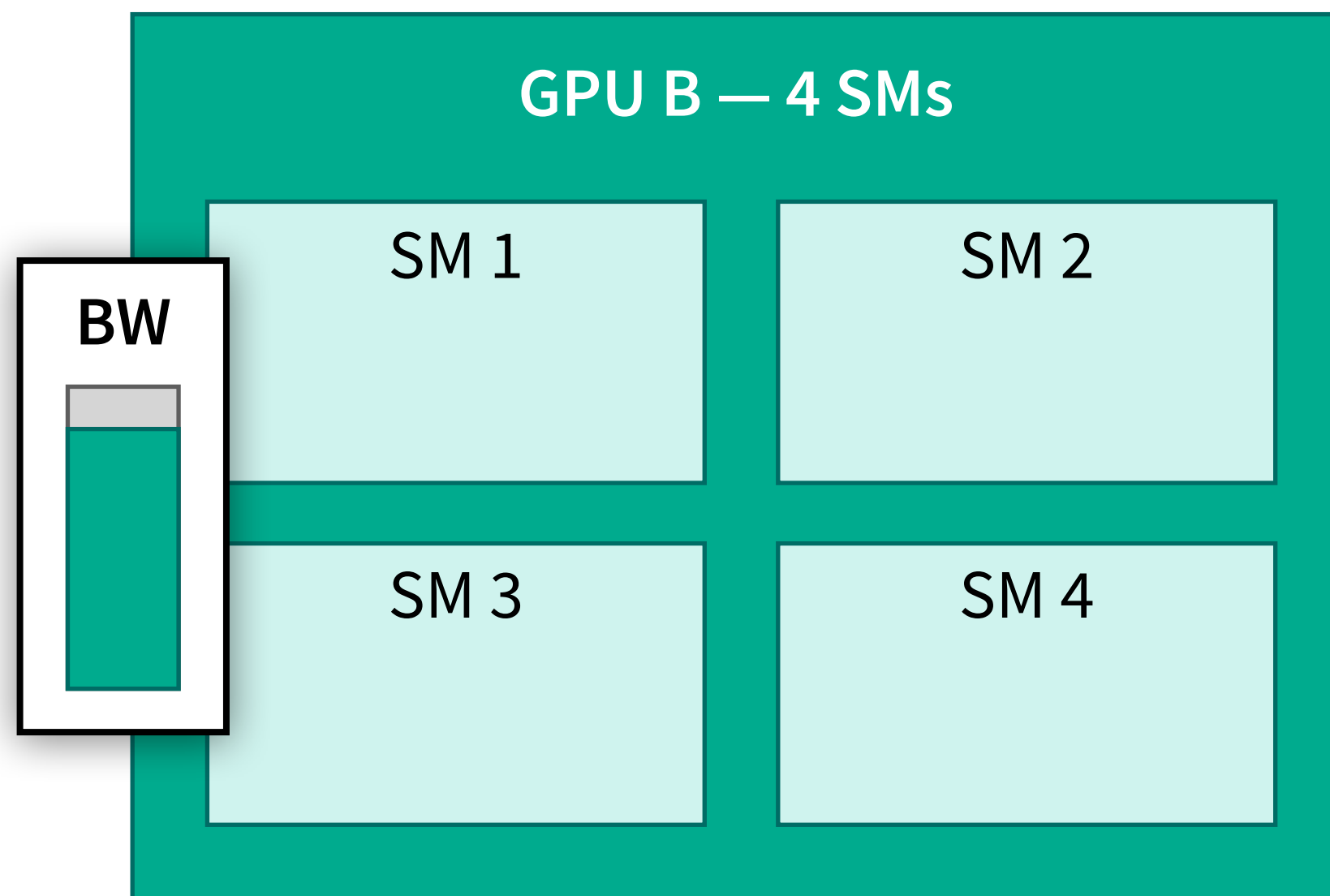
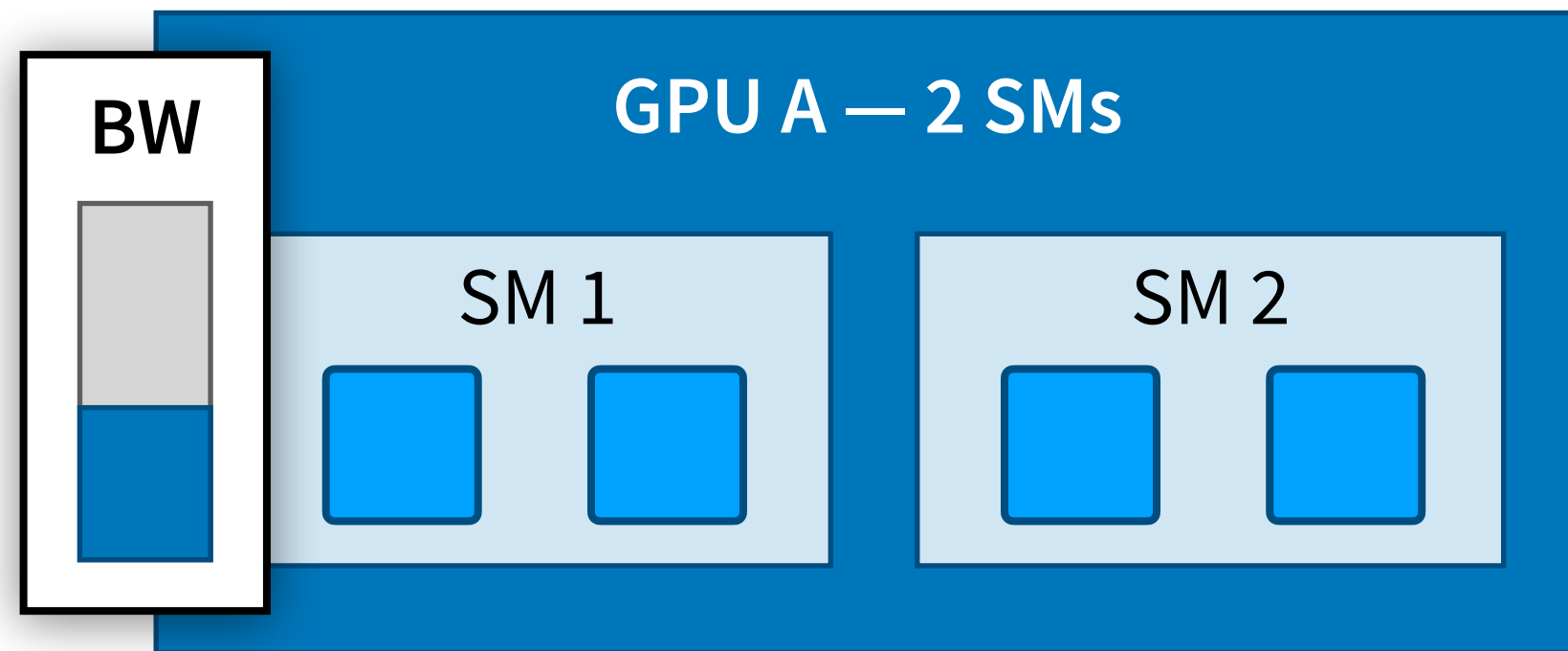


Wave scaling



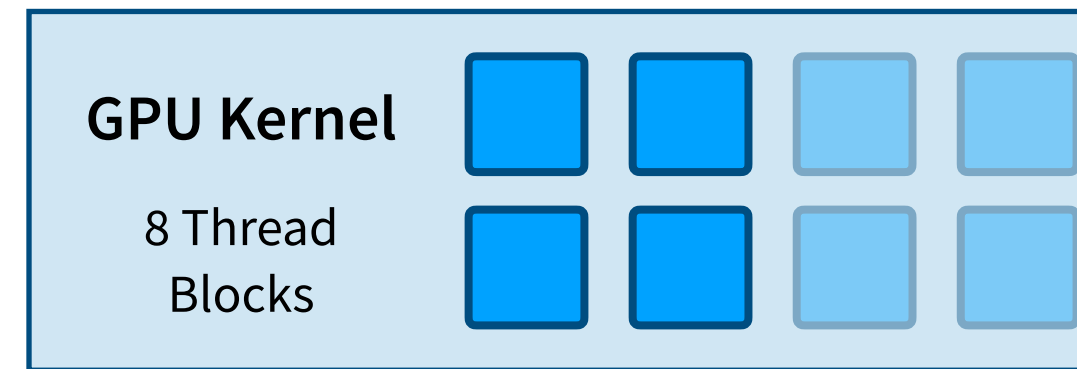
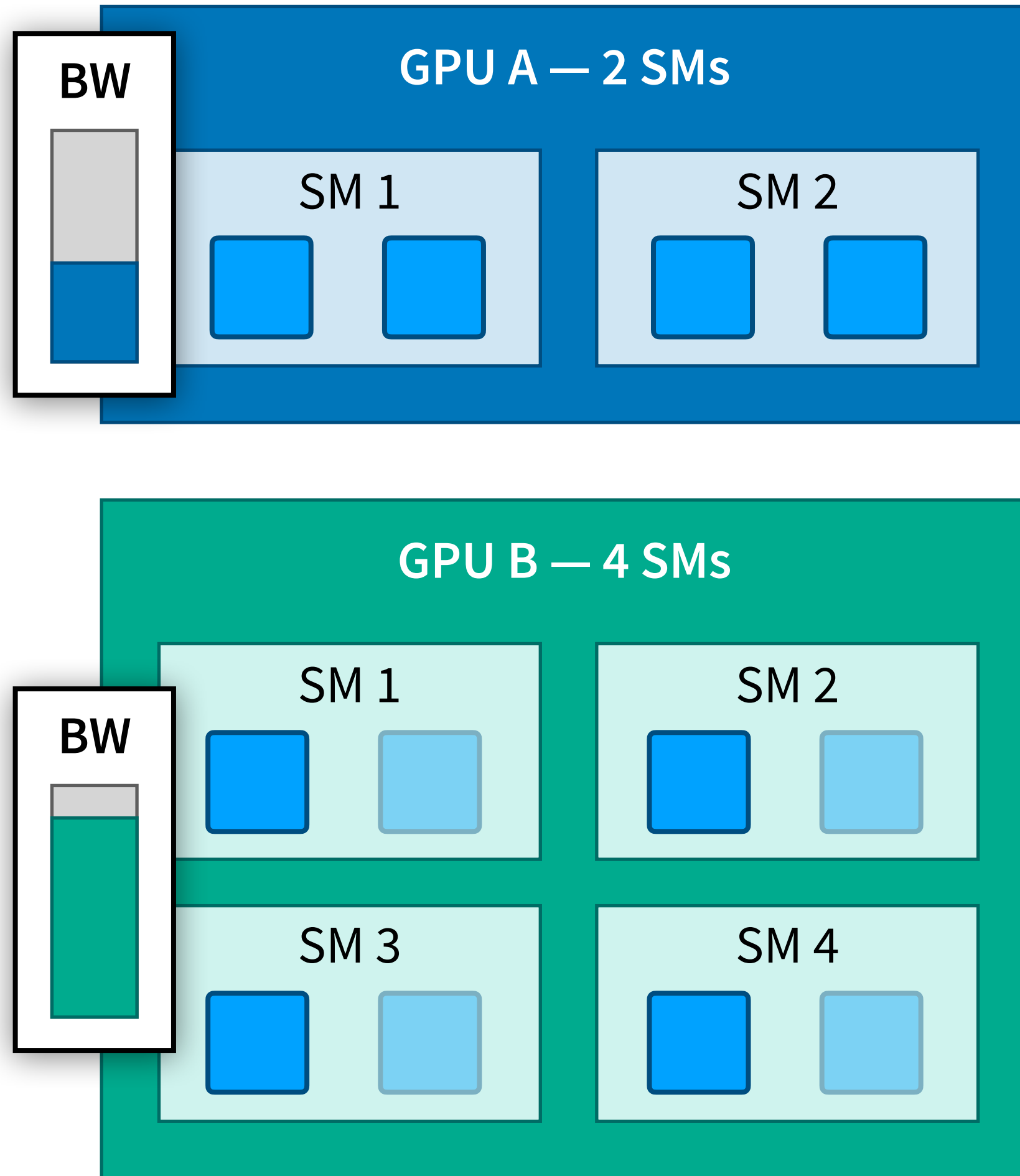
- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

Wave scaling

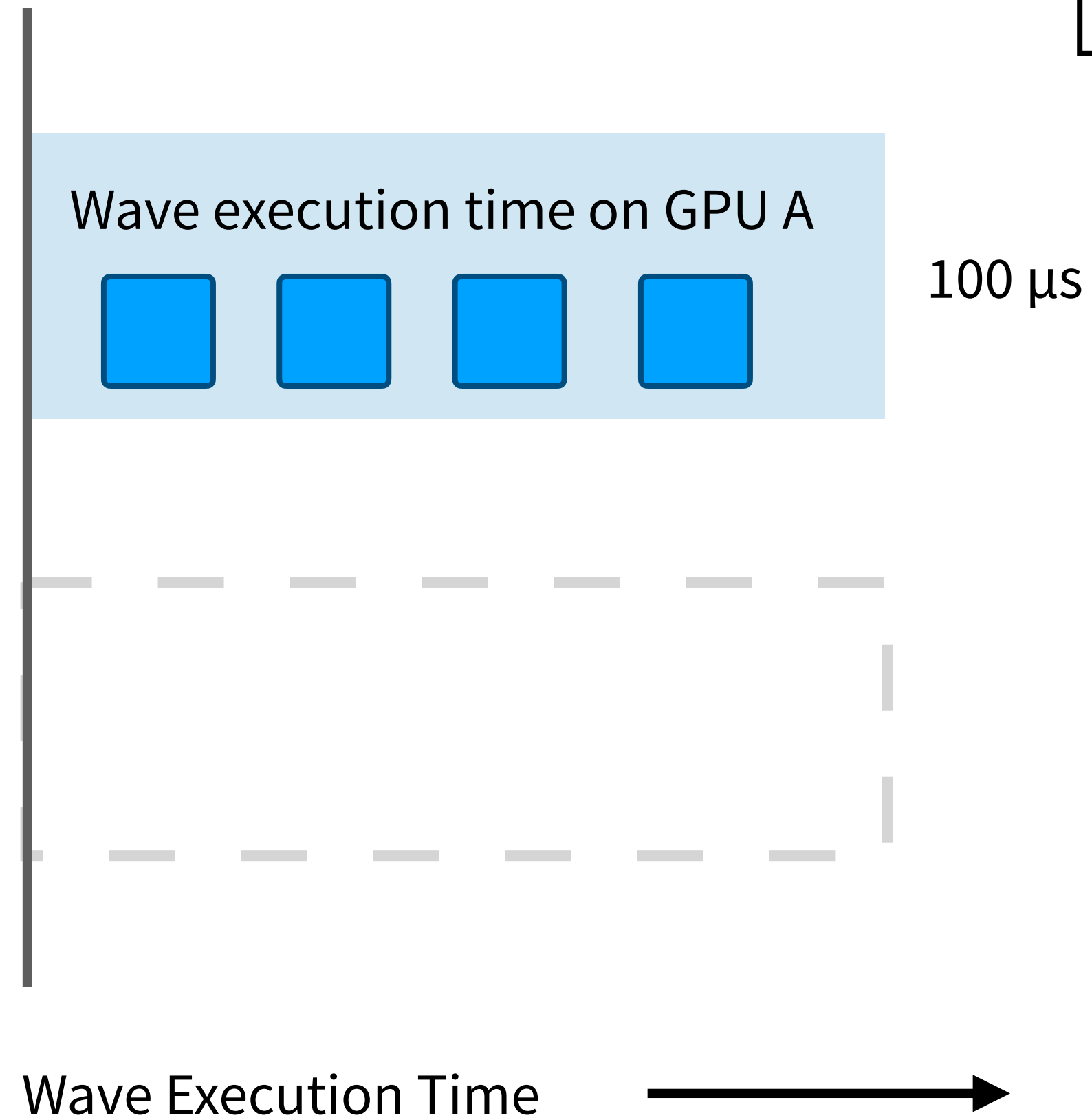


- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency

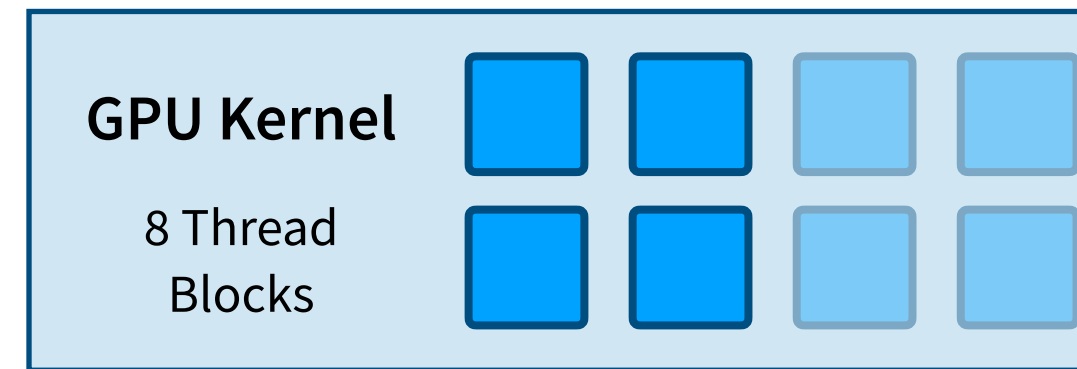
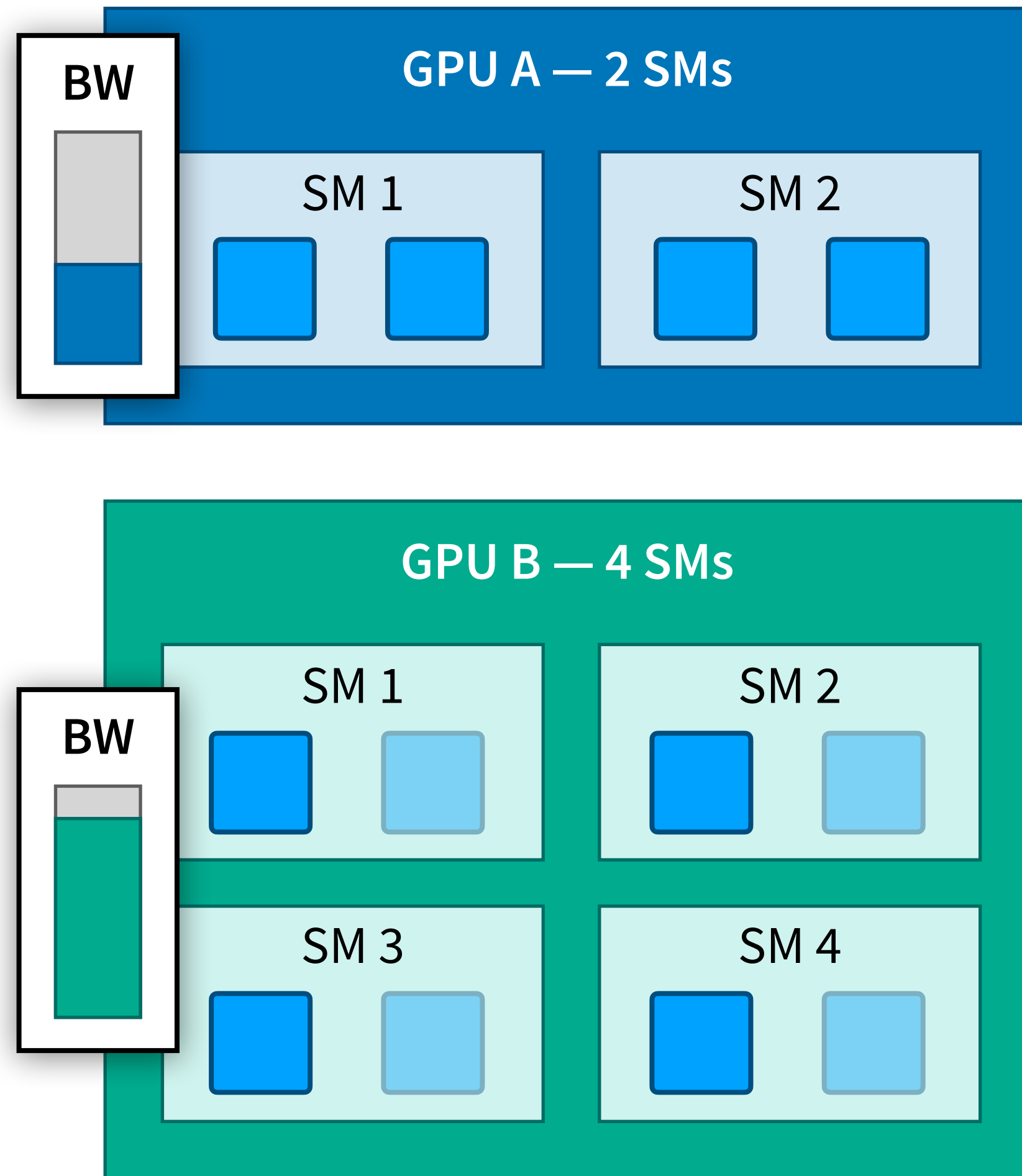
Wave scaling



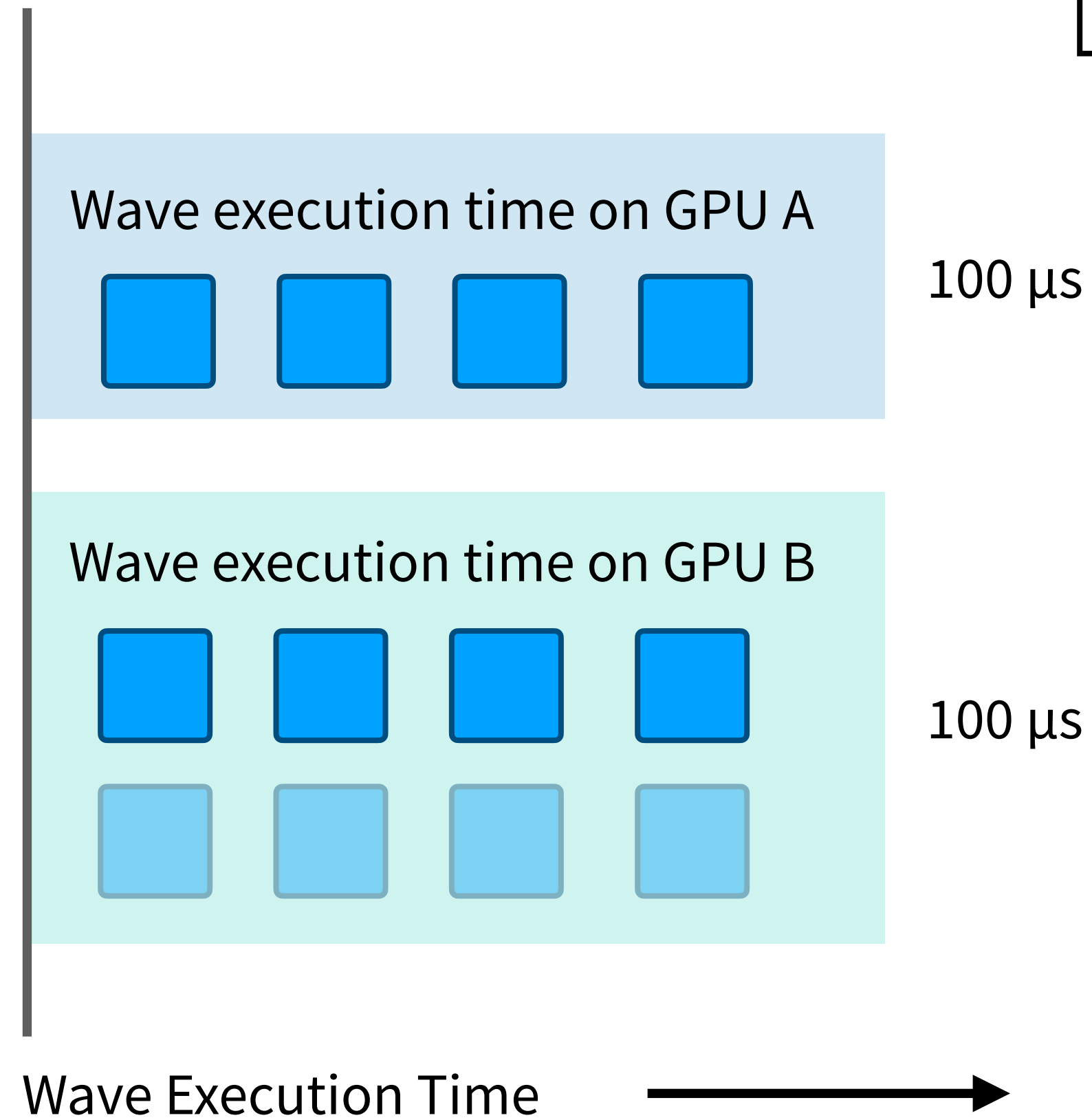
- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



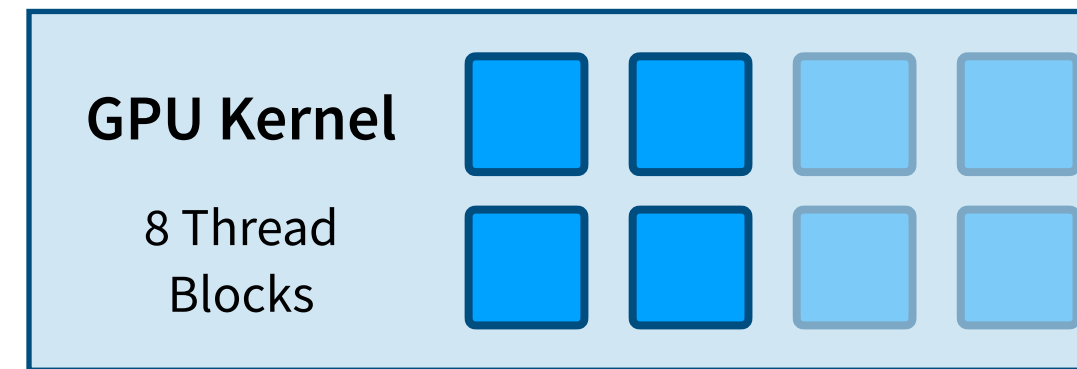
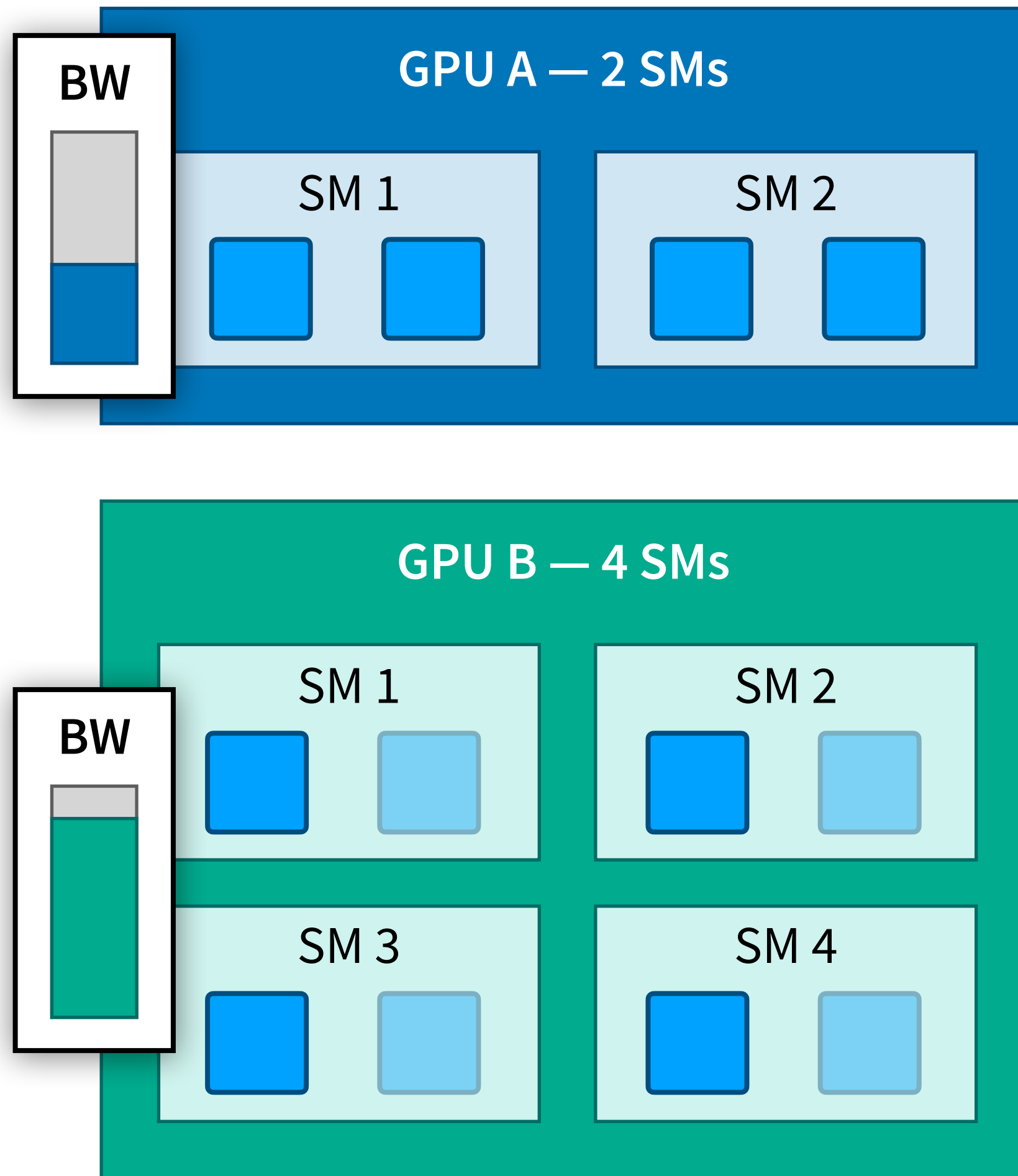
Wave scaling



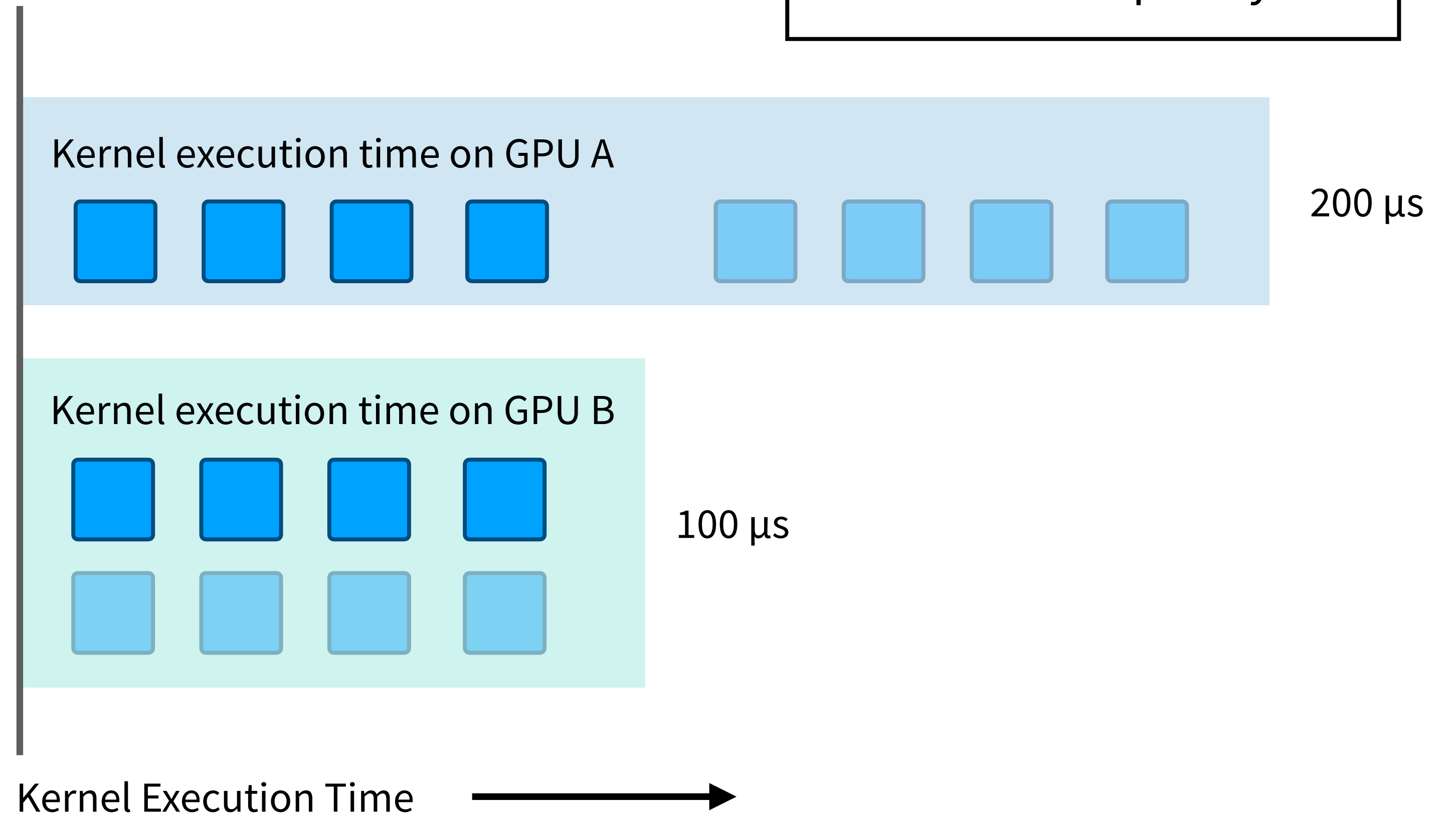
- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



Wave scaling



- ### Scaling Factors
- Memory bandwidth
 - Wave size
 - Clock frequency



One last wrinkle: Kernel-varying operations

- 🌊 Wave scaling assumes the same kernel is used across GPUs
- ⚠️ A few DNN operations use **architecture-specific** kernels (“kernel-varying”)
 - Convolutions, linear (dense) layers, LSTMs
- 💡 Habitat uses pre-trained **multilayer perceptrons (MLP)** for these operations

Evaluation

- 🔍 How **accurate** are Habitat's predictions?
- ✅ Does using Habitat lead to making **“correct”** decisions?

- **Six GPUs** (spanning three generations):



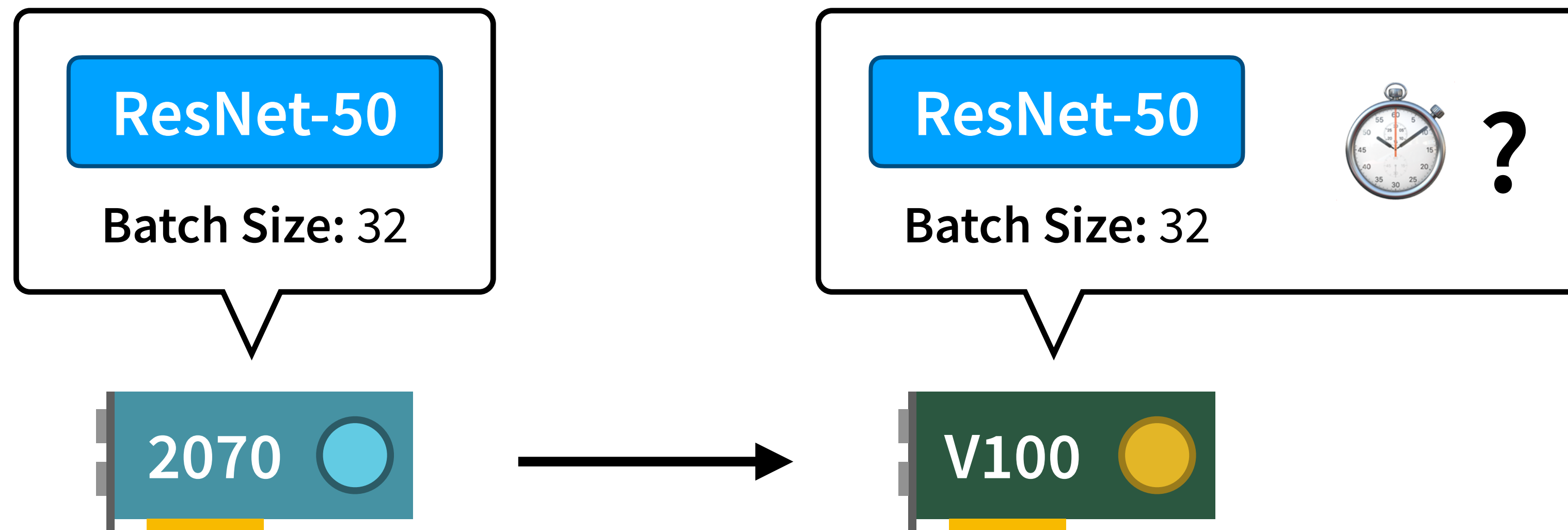
- **Five models:**



- PyTorch 1.4.0

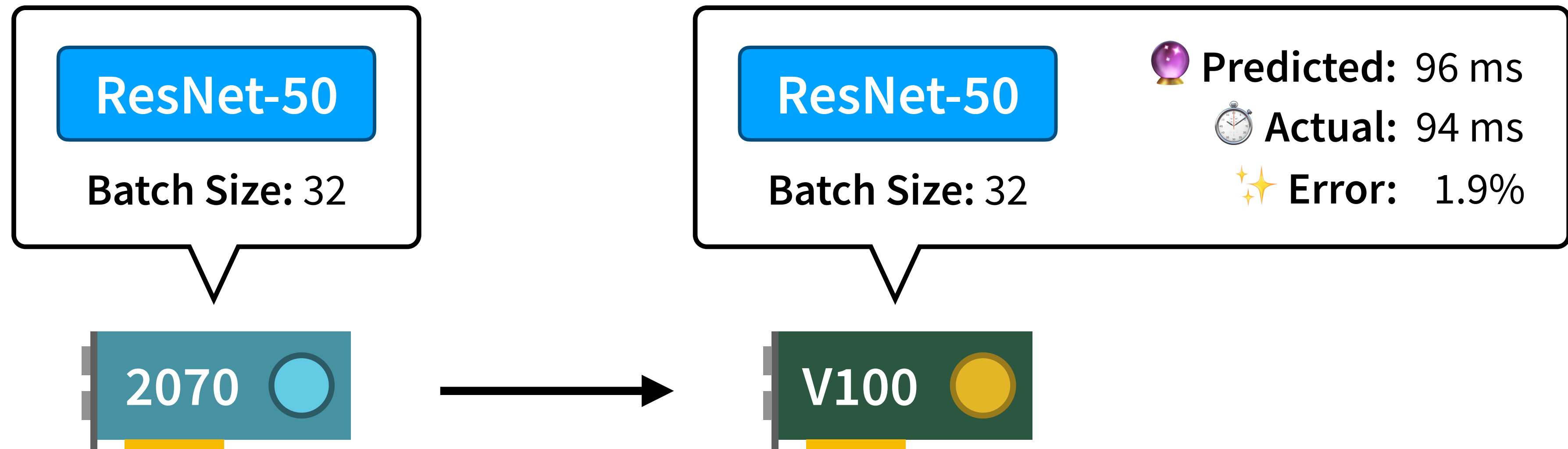
How accurate is Habitat?

- Predict iteration execution time (GPU, model, batch size)

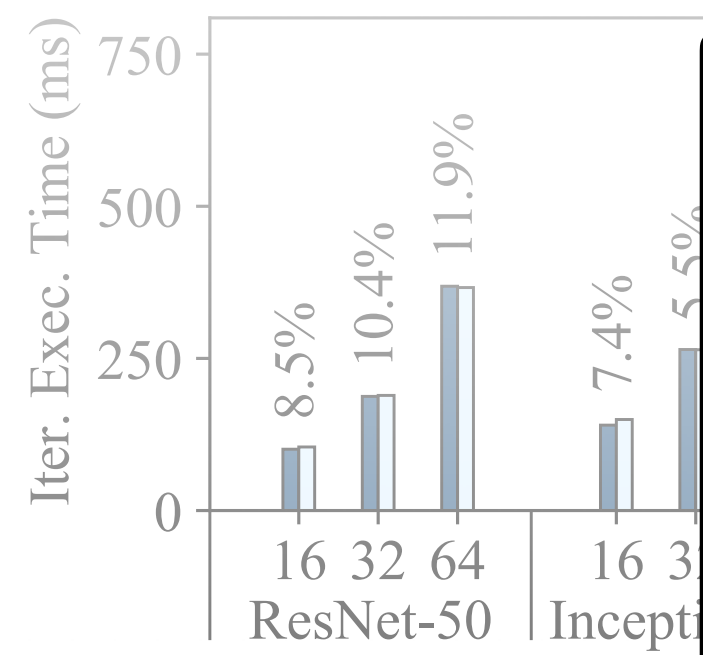


How accurate is Habitat?

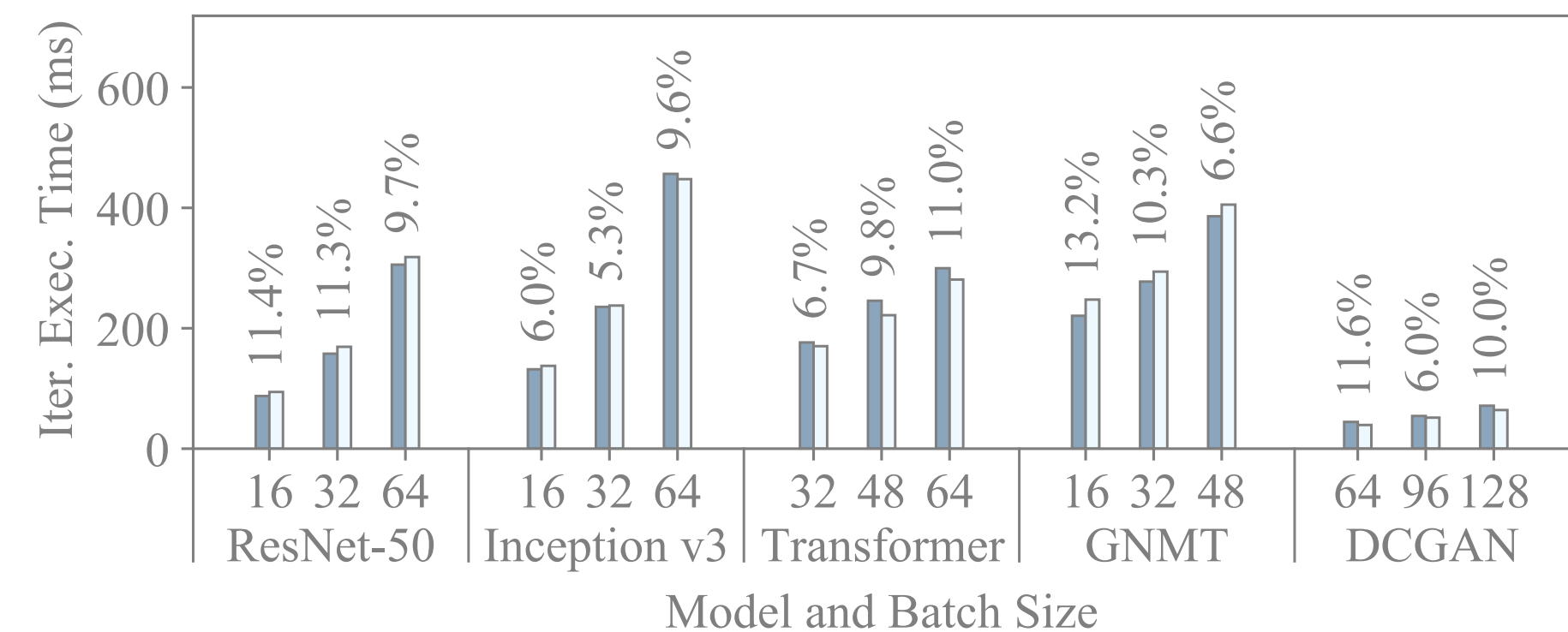
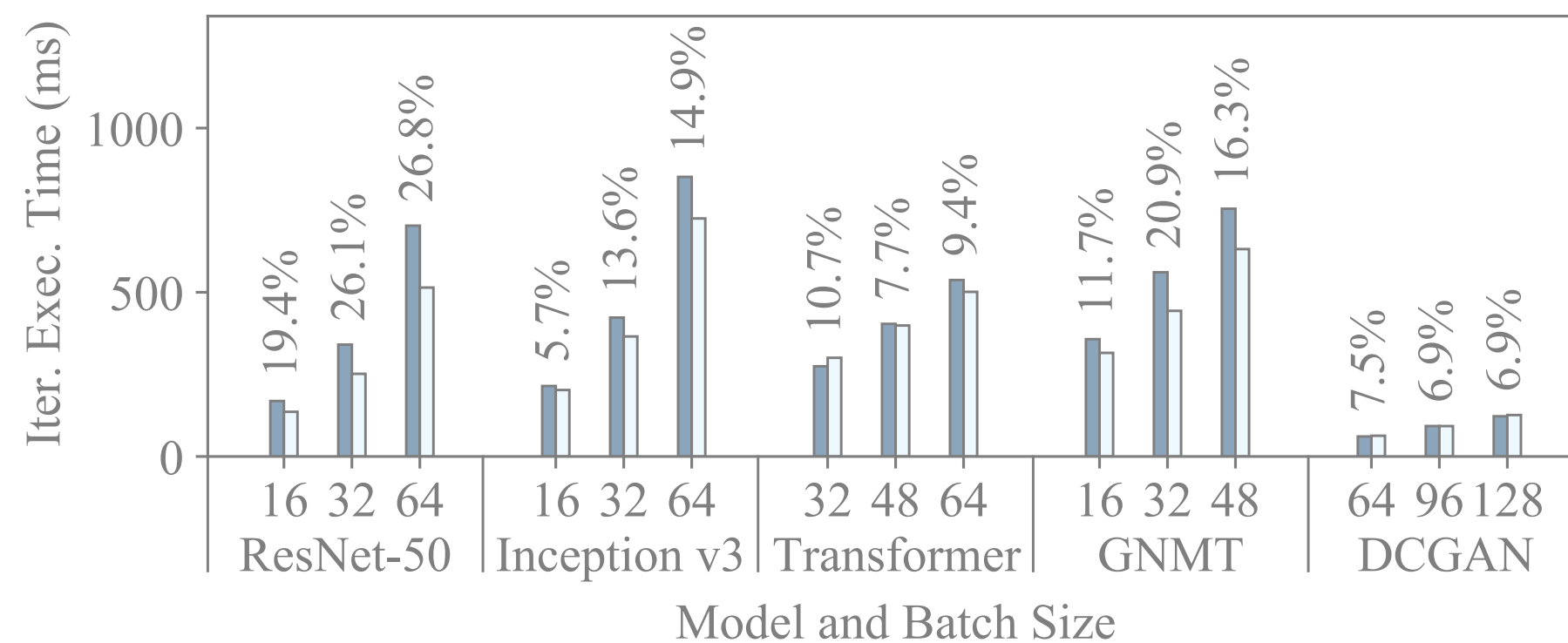
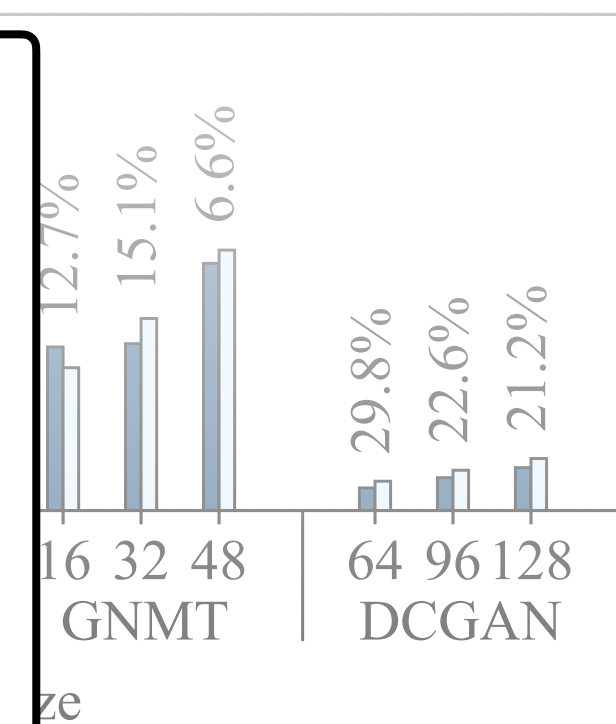
- Predict iteration execution time (GPU, model, batch size)



How accurate is Habitat?



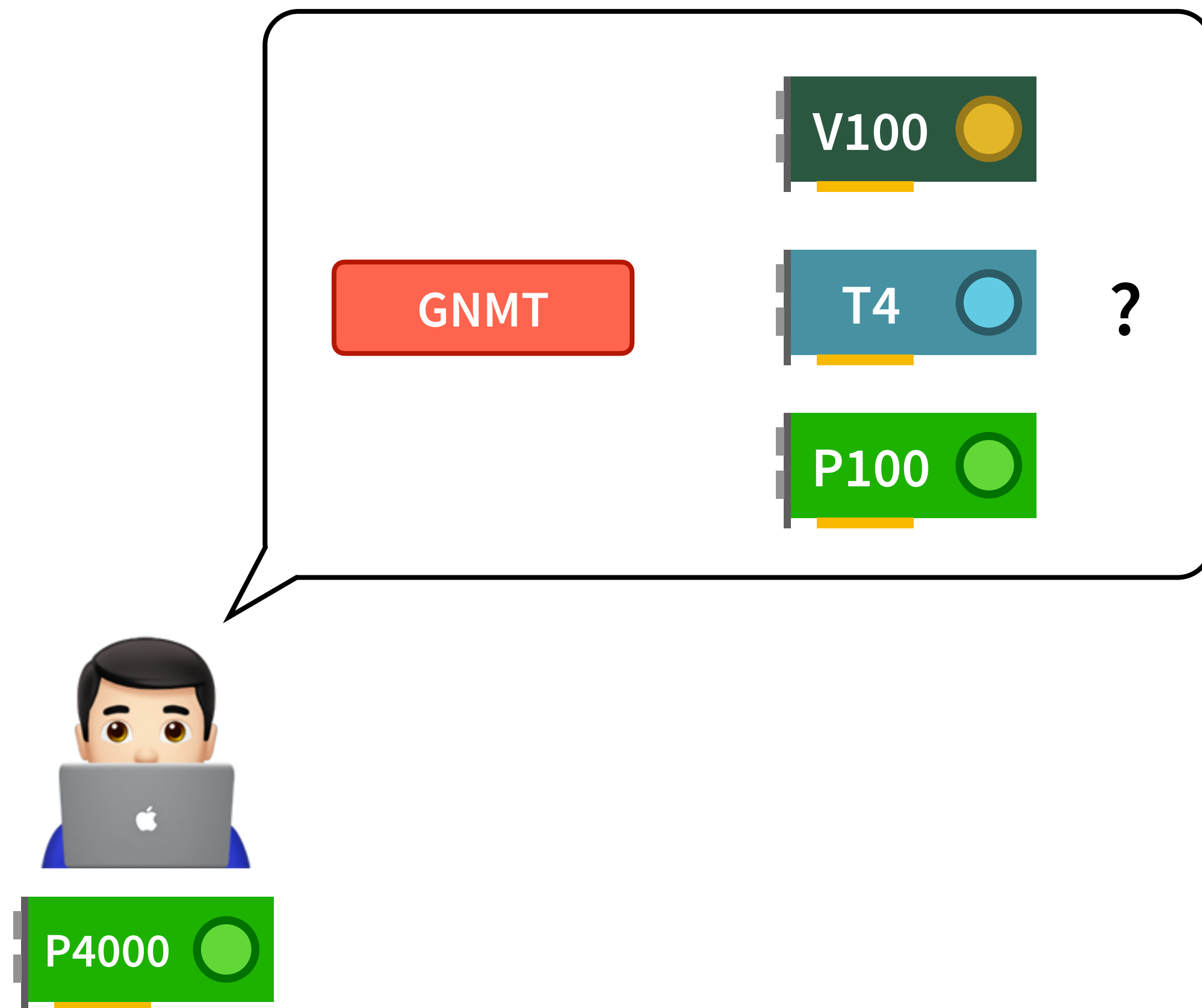
Habitat makes accurate predictions, with an average error of **11.8%** across all configurations (30 GPU pairs x 5 models x 3 batch sizes).



**Does Habitat lead to the “correct”
decision?**

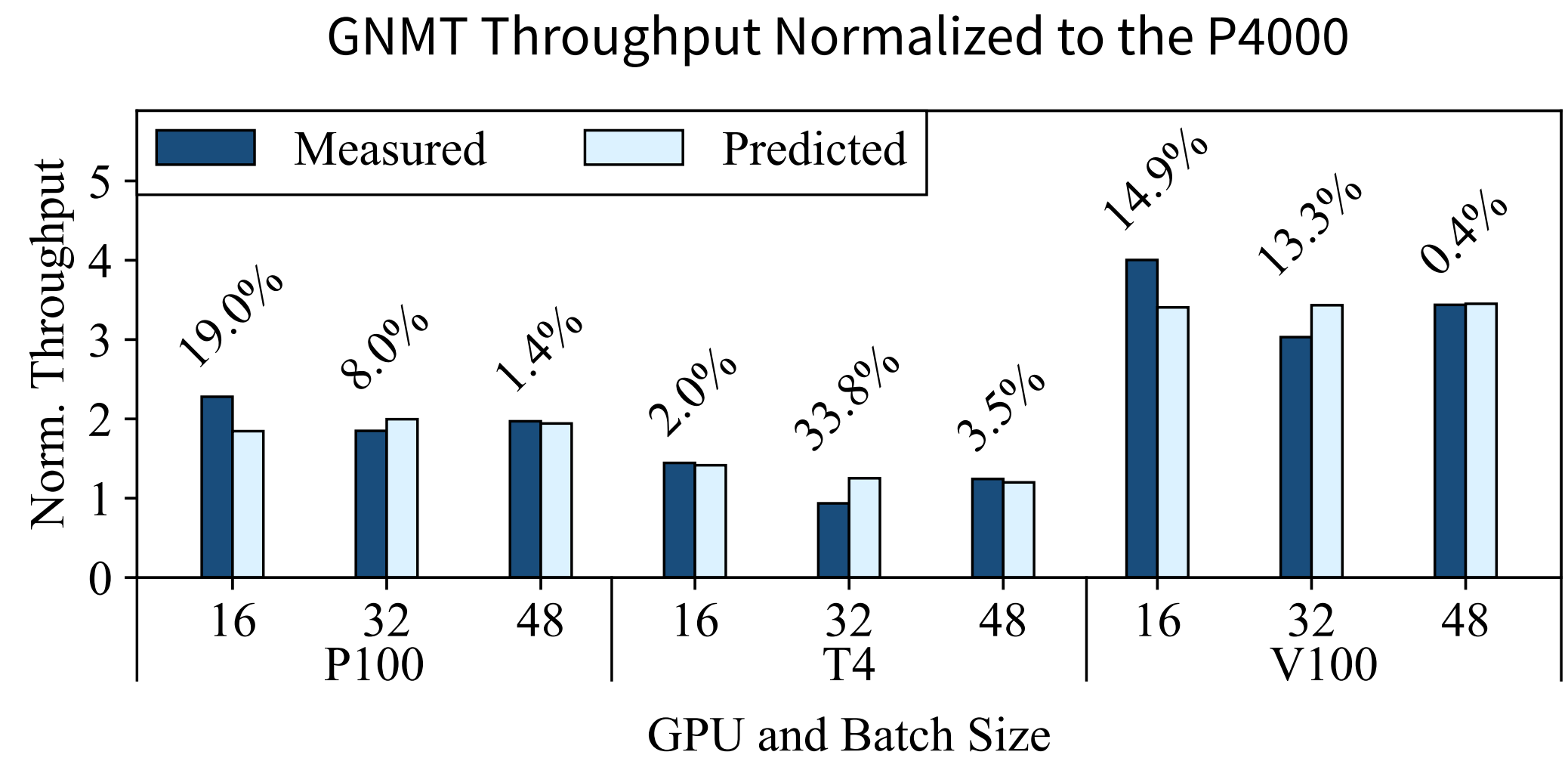
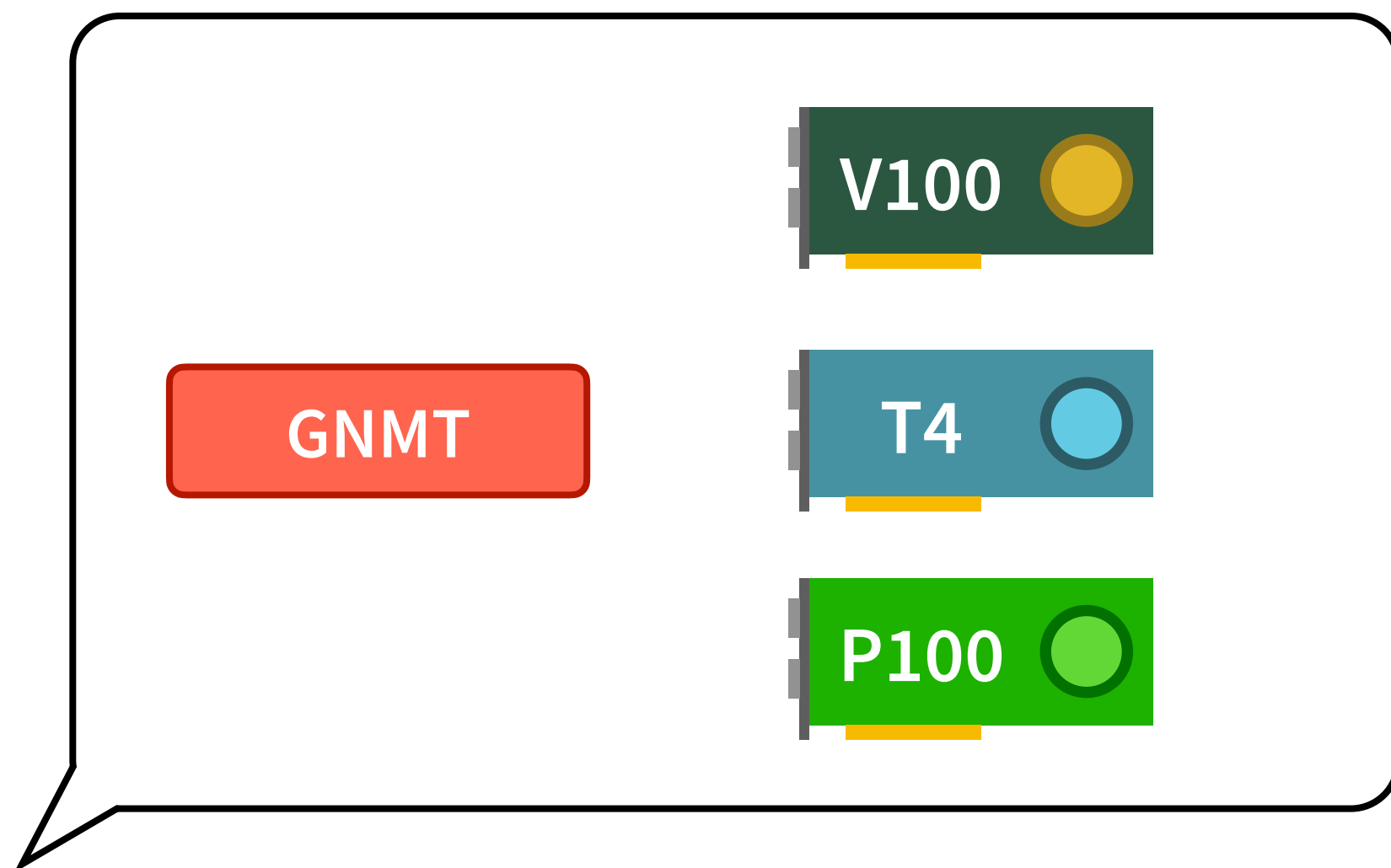
Rent a GPU in the cloud?

Scenario: Want to train GNMT, have access to a P4000. Which cloud GPU to use, if any?



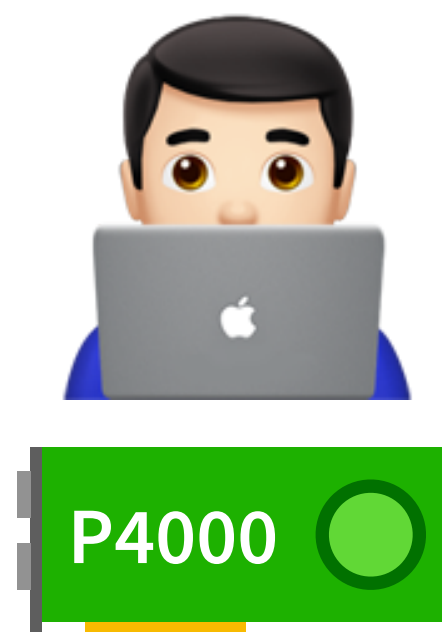
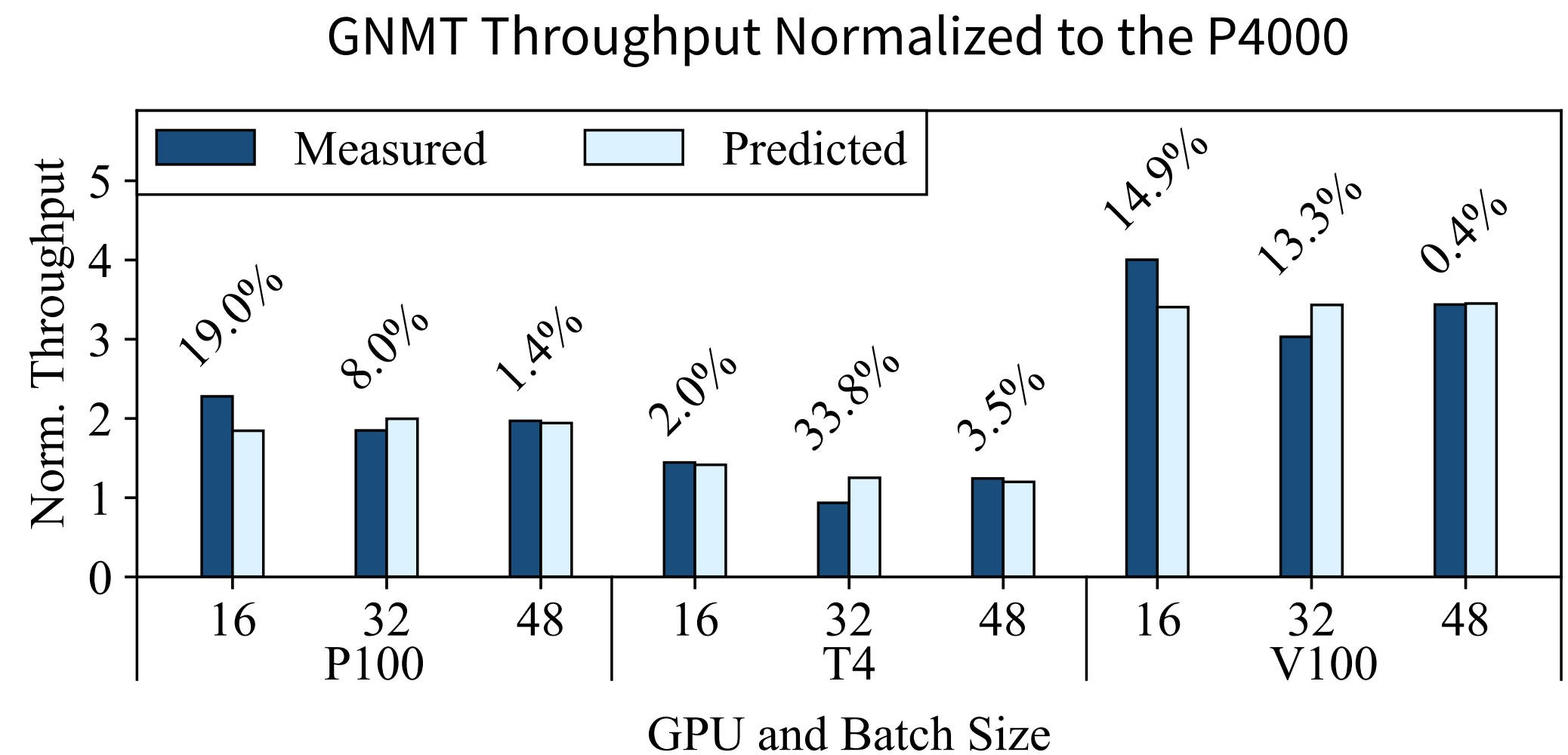
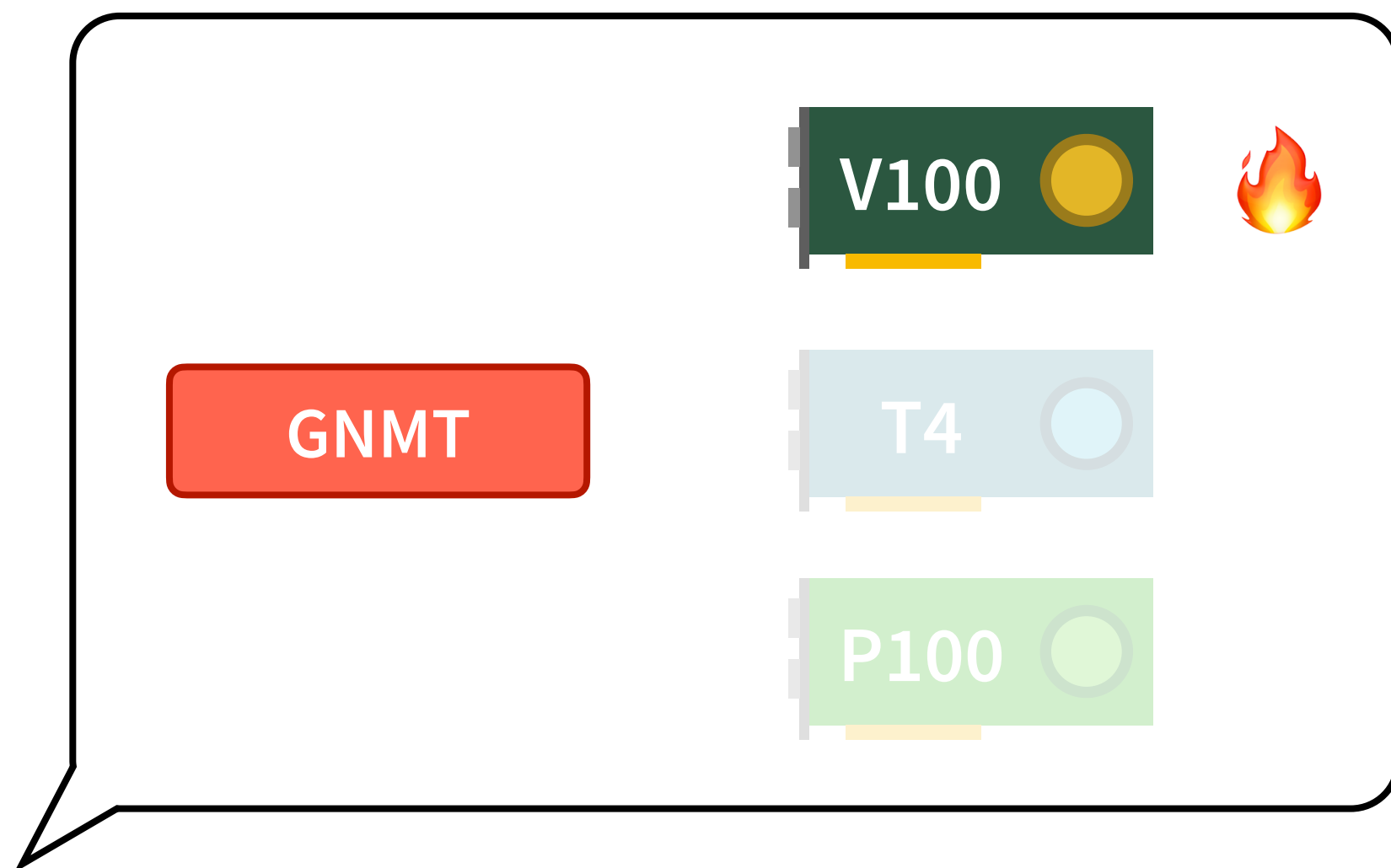
Rent a GPU in the cloud?

Scenario: Want to train GNMT, have access to a P4000. Which cloud GPU to use, if any?



Rent a GPU in the cloud?

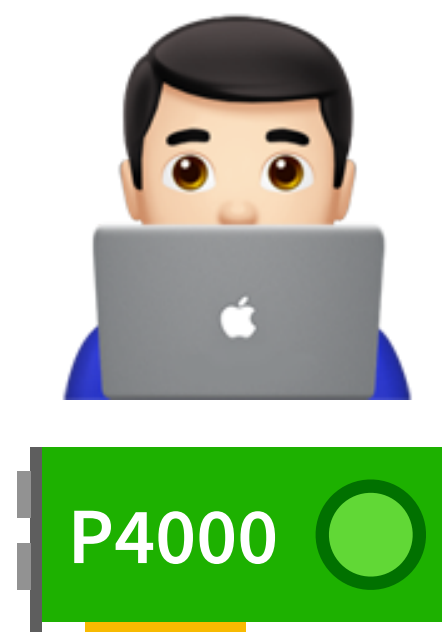
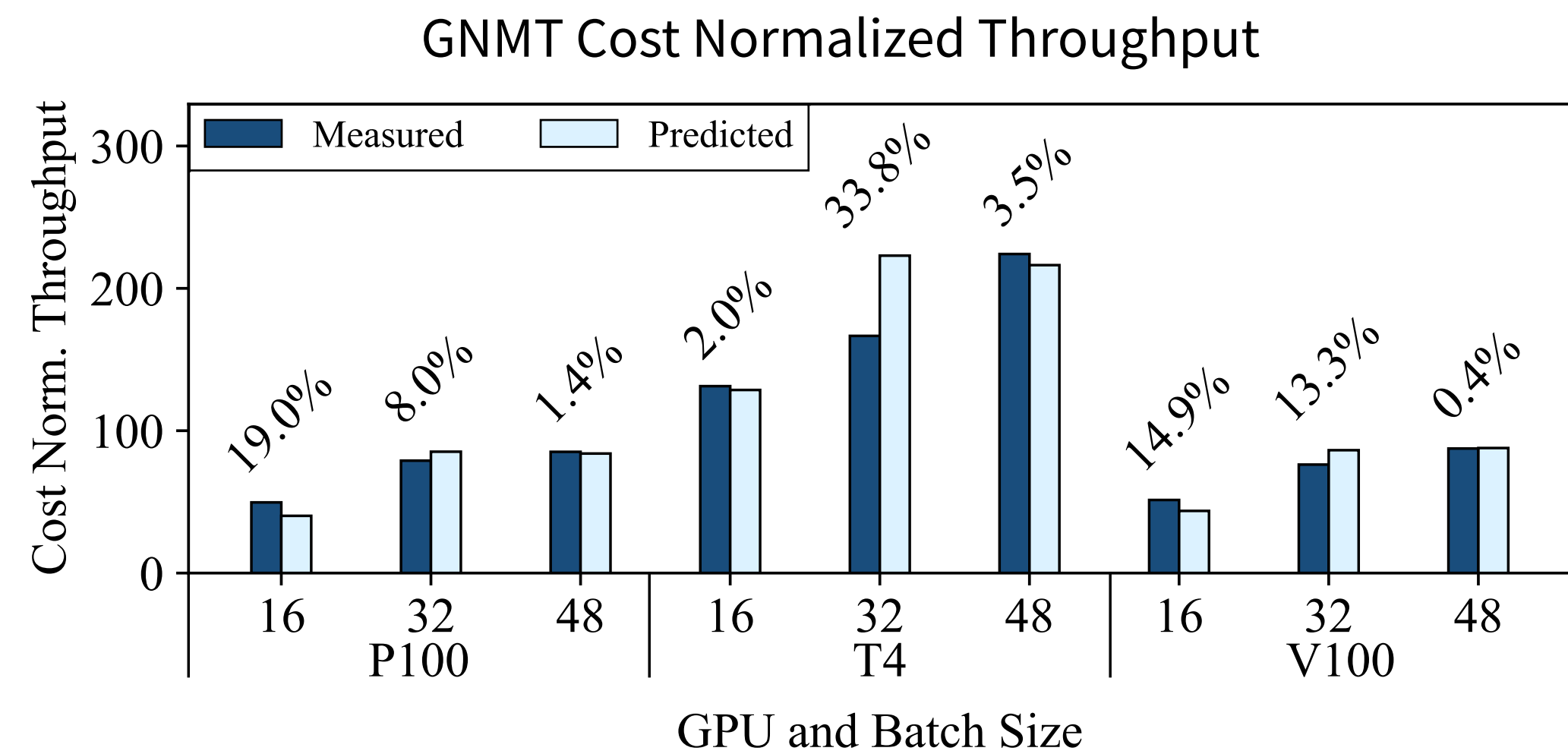
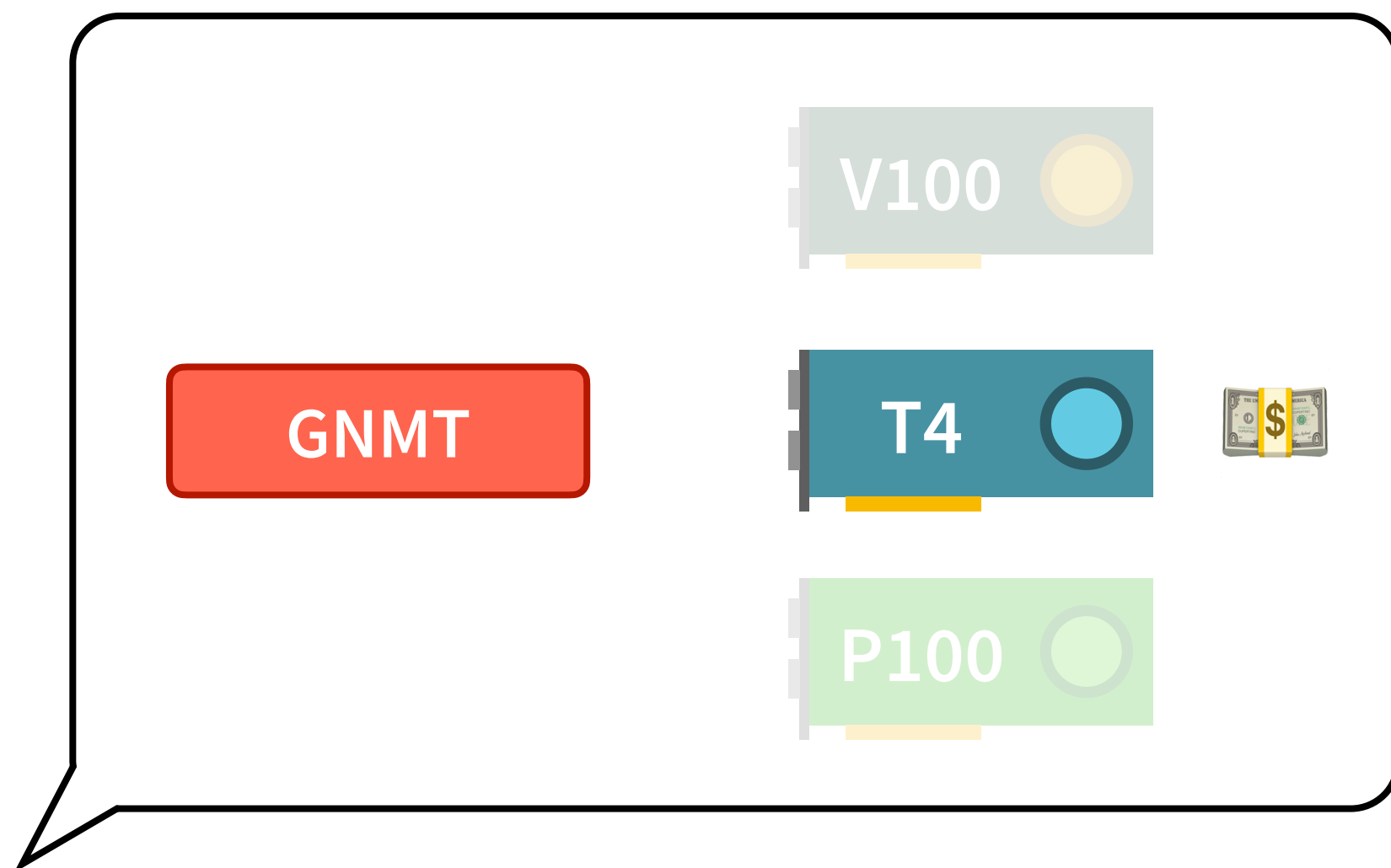
Scenario: Want to train GNMT, have access to a P4000. Which cloud GPU to use, if any?



Habitat correctly predicts that the **V100** is the best choice for **performance**.

Rent a GPU in the cloud?

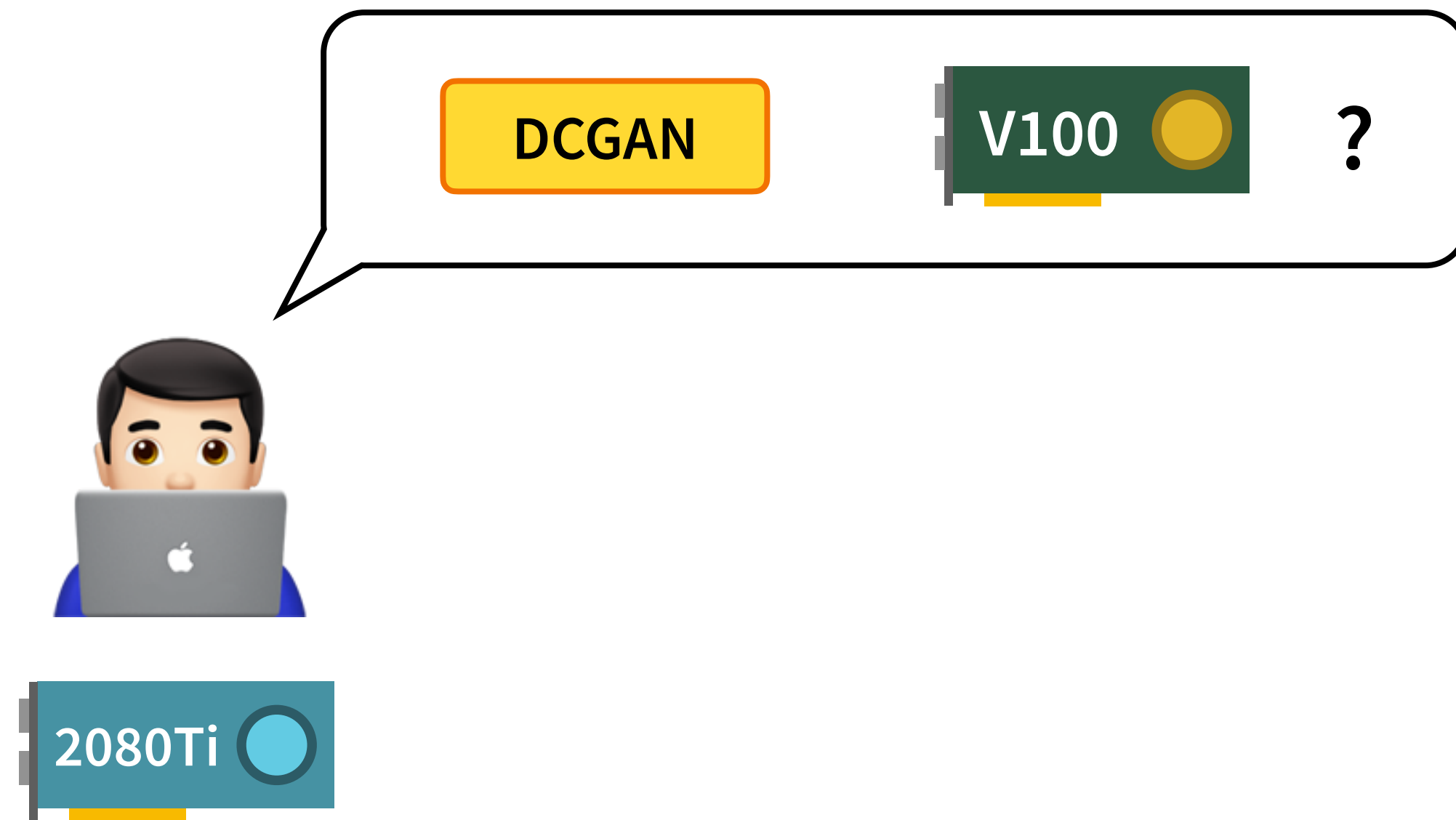
Scenario: Want to train GNMT, have access to a P4000. Which cloud GPU to use, if any?



Habitat correctly predicts that the T4 (or P4000) is the best choice for **cost**.

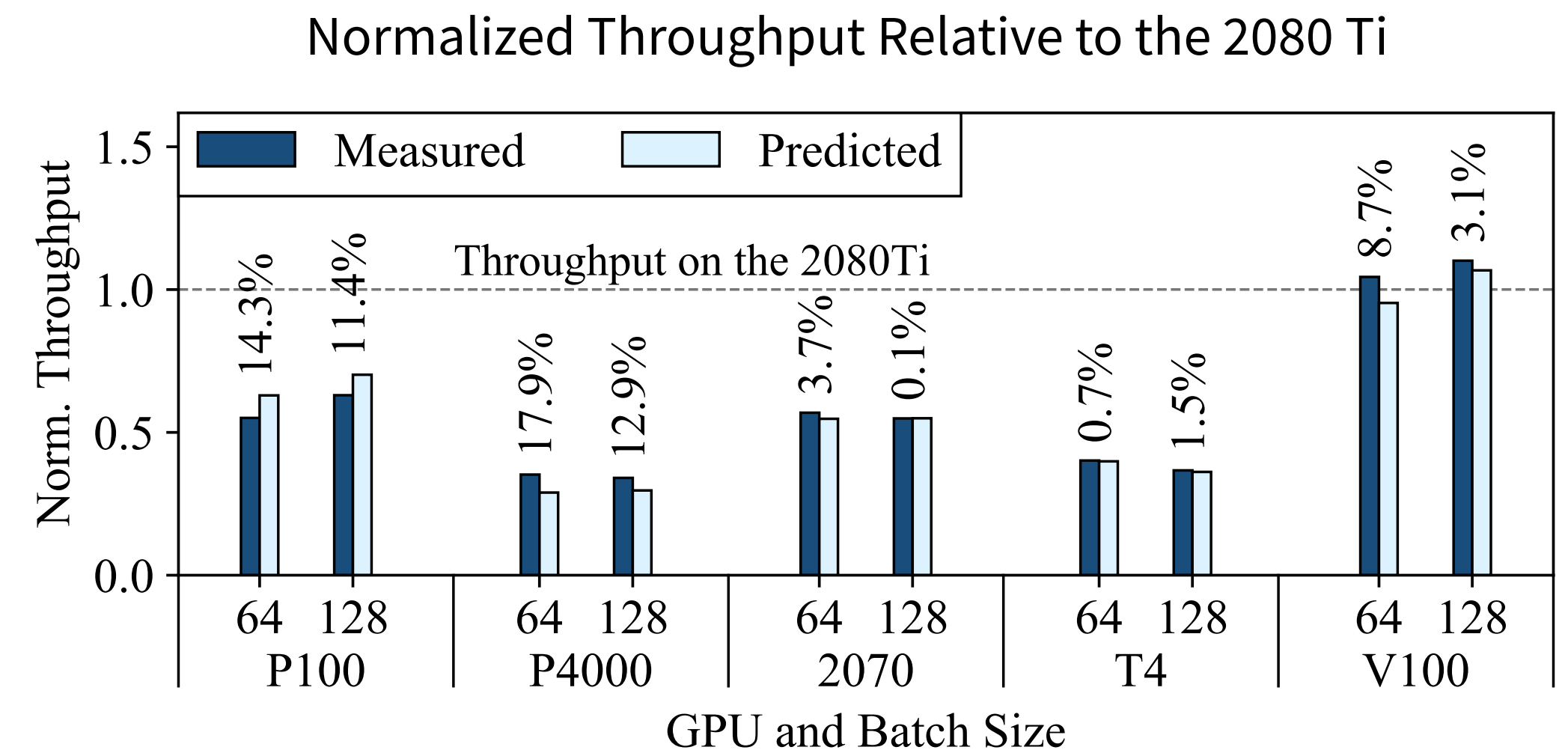
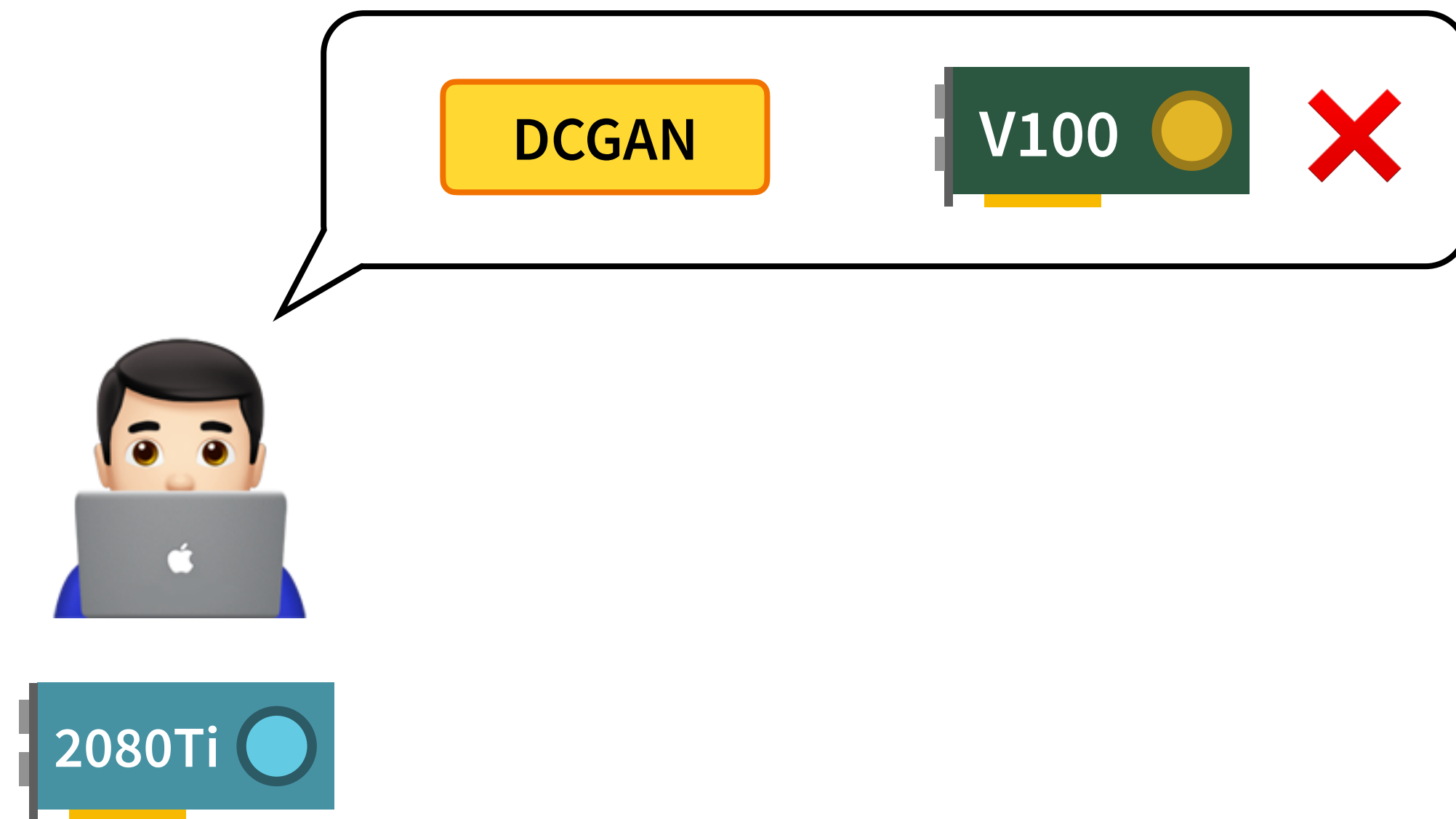
Always use the “best” GPU?

Scenario: Want to train DCGAN, have access to a 2080 Ti. Use the V100?



Always use the “best” GPU?

Scenario: Want to train DCGAN, have access to a 2080 Ti. Use the V100?



Habitat correctly predicts that the V100 only offers a marginal improvement (1.1x) over the 2080 Ti.

More details and results in the paper

- 🔍 Prediction breakdowns
- 🔬 MLP sensitivity study
- 🌟 Predictions onto additional GPUs
- 💬 Discussion about extensibility
 - Distributed training
 - Mixed precision training
 - Other types of hardware

Habitat: A Runtime-Based Computational Performance Predictor for Deep Neural Network Training

Geoffrey X. Yu
University of Toronto
Vector Institute

Yubo Gao
University of Toronto

Pavel Golikov
University of Toronto
Vector Institute

Gennady Pekhimenko
University of Toronto
Vector Institute

Abstract

Deep learning researchers and practitioners usually leverage GPUs to help train their deep neural networks (DNNs) faster. However, choosing *which* GPU to use is challenging both because (i) there are many options, and (ii) users grapple with competing concerns: maximizing compute performance while minimizing costs. In this work, we present a new practical technique to help users make informed and cost-efficient GPU selections: make performance *predictions* with the help of a GPU that the user already has. Our technique exploits the observation that, because DNN training consists of repetitive compute steps, predicting the execution time of a single iteration is usually enough to characterize the performance of an entire training process. We make predictions by scaling the execution time of each operation in a training iteration from one GPU to another using either (i) wave scaling, a technique based on a GPU’s execution model, or (ii) pre-trained multilayer perceptrons. We implement our technique into a Python library called Habitat and find that it makes accurate iteration execution time predictions (with an average error of 11.8%) on ResNet-50, Inception v3, the Transformer, GNMT, and DCGAN across six different GPU architectures. Habitat supports PyTorch, is easy to use, and is open source.¹

1 Introduction

Over the past decade, deep neural networks (DNNs) have seen incredible success across many machine learning tasks [26, 37, 39, 50, 93, 96, 99]—leading them to become widely used throughout academia and industry. However, despite their popularity, DNNs are not always straightforward to use in practice because they can be extremely computationally-expensive to train [23, 53, 95, 109]. This is why, over the past few years, there has been a significant and ongoing effort to bring *hardware acceleration* to DNN training [10, 16, 35, 36, 45, 78, 80].




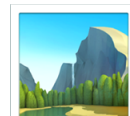
training. These options range from desktop and server-class GPUs (e.g., 2080Ti [70] and A100 [78]) all the way to specialized accelerators such as the TPU [45], AWS Trainium [10], Gaudi [36], IPU [35], and Cerebras WSE [16]. Having all these options offers flexibility to users, but at the same time can also lead to a paradox of choice: which hardware option should a researcher or practitioner use to train their DNNs?

A natural way to start answering this question is to first consider CUDA-enabled GPUs. This is because they (i) are commonly used in deep learning; (ii) are supported by all major deep learning software frameworks (PyTorch [86], TensorFlow [1], and MXNet [19]); (iii) have mature tooling support (e.g., CUPTI [76]); and (iv) are readily available for rent *and* purchase. In particular, when considering GPUs, we find that there are many situations where a deep learning user needs to *choose* a specific GPU to use for training:

- **Choosing between different hardware tiers.** In both academia and industry, deep learning users often have access to several *tiers* of hardware: (i) a workstation with a GPU used for development (e.g., 2080Ti), (ii) a private GPU cluster that is shared within their organization (e.g., RTX6000 [84]), and (iii) GPUs that they can rent in the cloud (e.g., V100 [66]). Each tier offers a different *cost, availability, and performance* trade-off. For example, a private cluster might be “free” (in monetary cost) to use, but jobs may be queued because the cluster is also shared among other users. In contrast, cloud GPUs can be rented on-demand for exclusive use.

- **Deciding on which GPU to rent or purchase.** Cloud providers make many different GPUs available for rent (e.g., P100 [62], V100, T4 [71], and A100 [78]), each with different performance at different prices. Similarly, a wide variety of GPUs are available for purchase (e.g., 2080Ti, 3090 [82]) both individually and as a part of pre-built work-

Key takeaways

-  DNN computation is special (**repetitive**), enabling new analysis opportunities.
-  Use **runtime-based information** to make iteration execution time predictions.
-  Habitat leads to the **correct decision** in the case studies.
-  The hardware landscape is growing; users need help **choosing effectively!**

 **Habitat is open source:**
github.com/geoffxy/habitat

Habitat

A Runtime-Based Computational Performance Predictor for Deep Neural Network Training

Geoffrey X. Yu, Yubo Gao, Pavel Golikov, Gennady Pekhimenko



 Get started: github.com/geoffxy/habitat