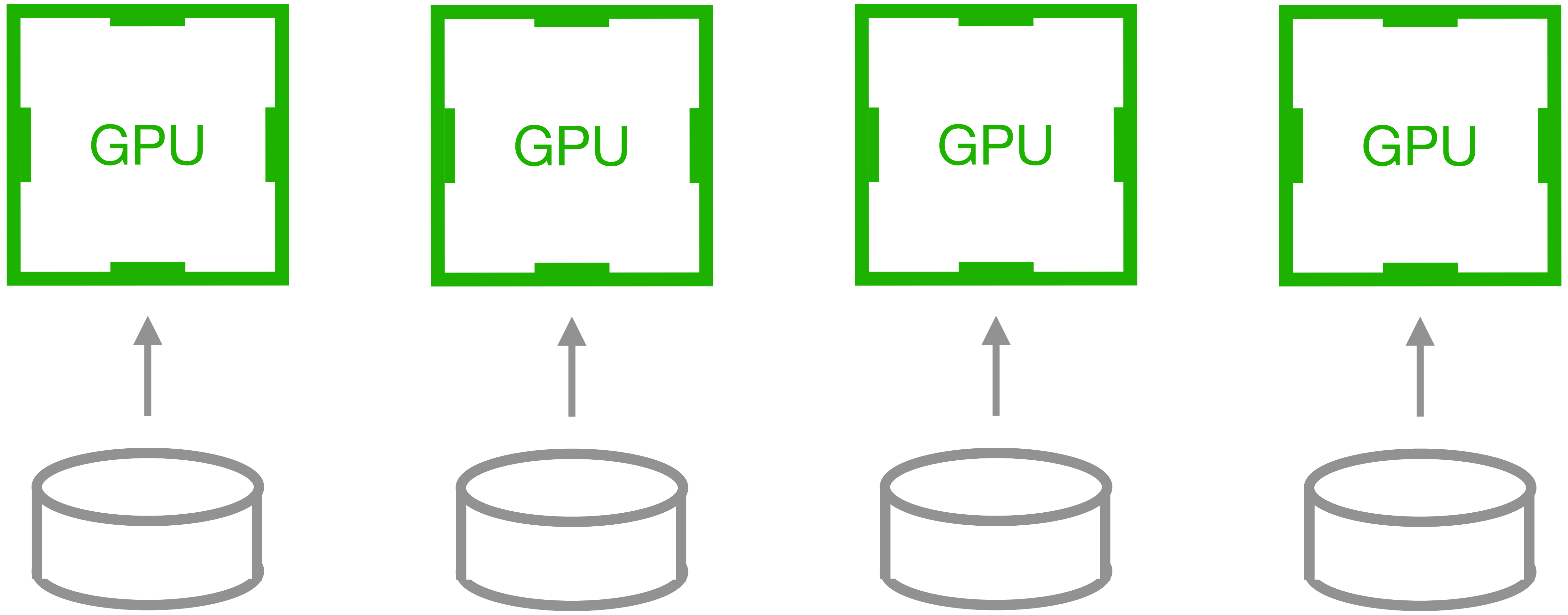
# Hi-Speed DNN Training with Espresso: Unleashing the Full Potential of Gradient Compression

**Zhuang Wang**, *Haibin Lin, *Yibo Zhu and T. S. Eugene Ng

*

# Distributed deep learning

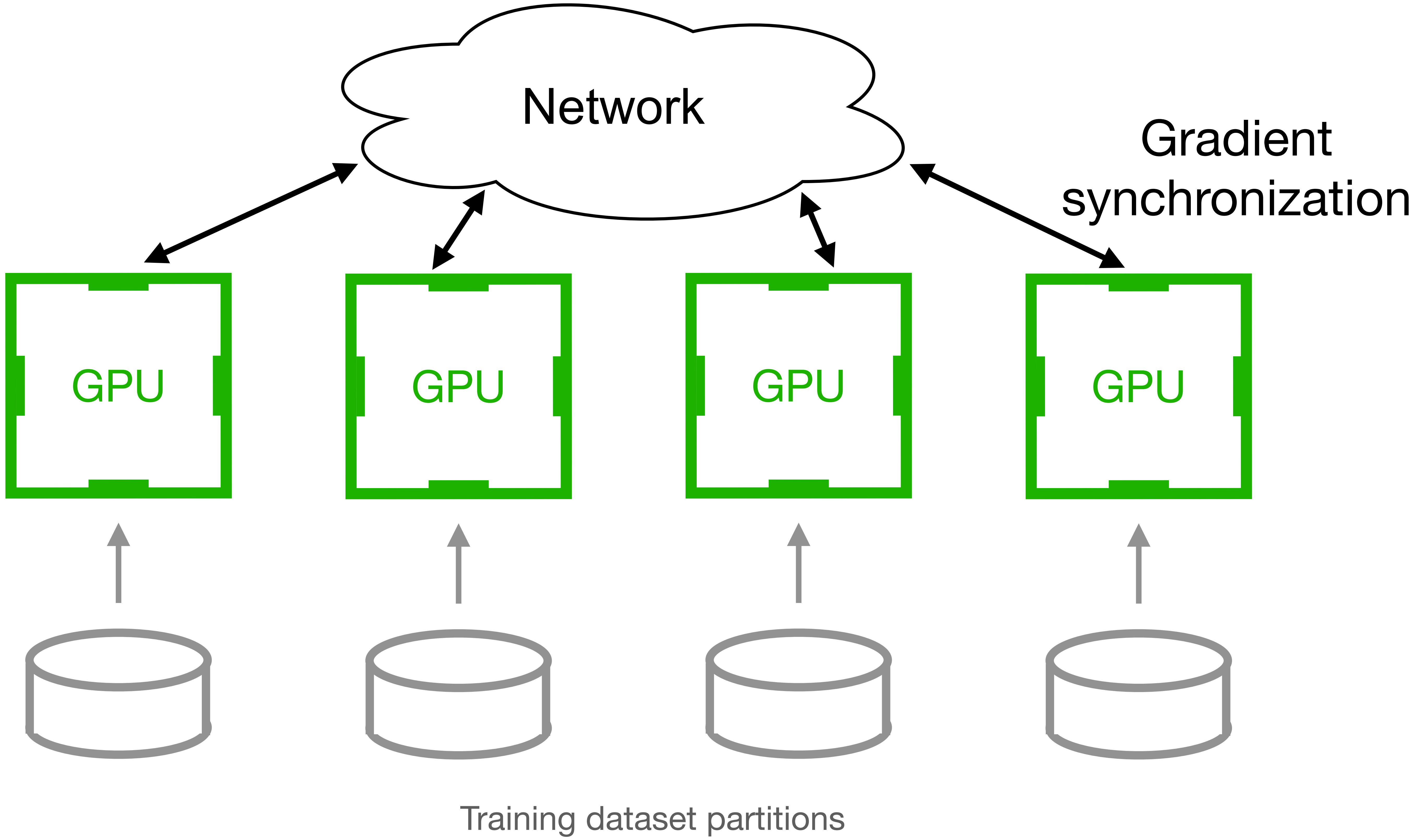
- Scale out training with multiple GPUs




Training dataset partitions

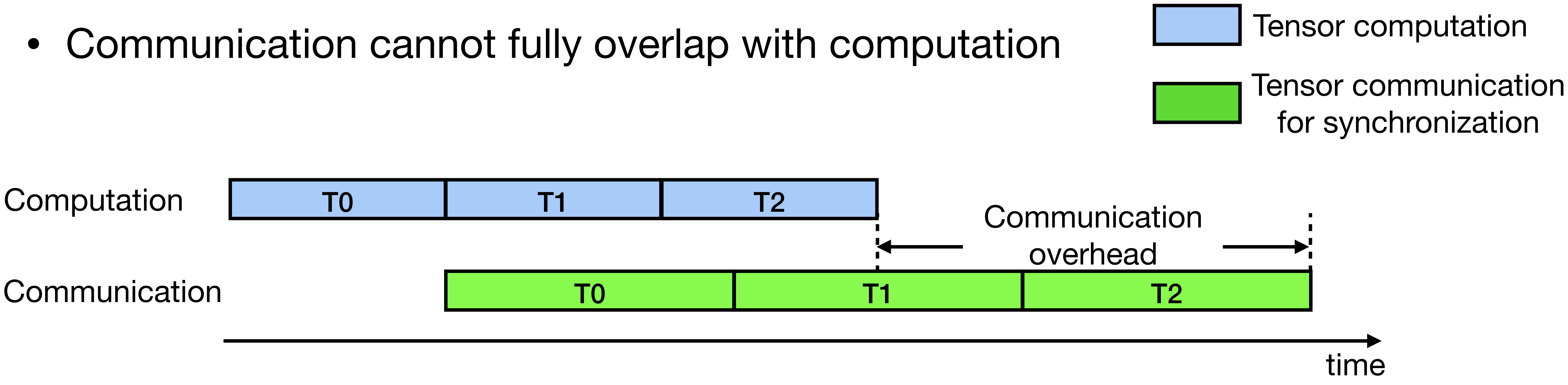
# Distributed deep learning

- Gradient synchronization among GPUs



# Communication is bottleneck

- Communication cannot fully overlap with computation



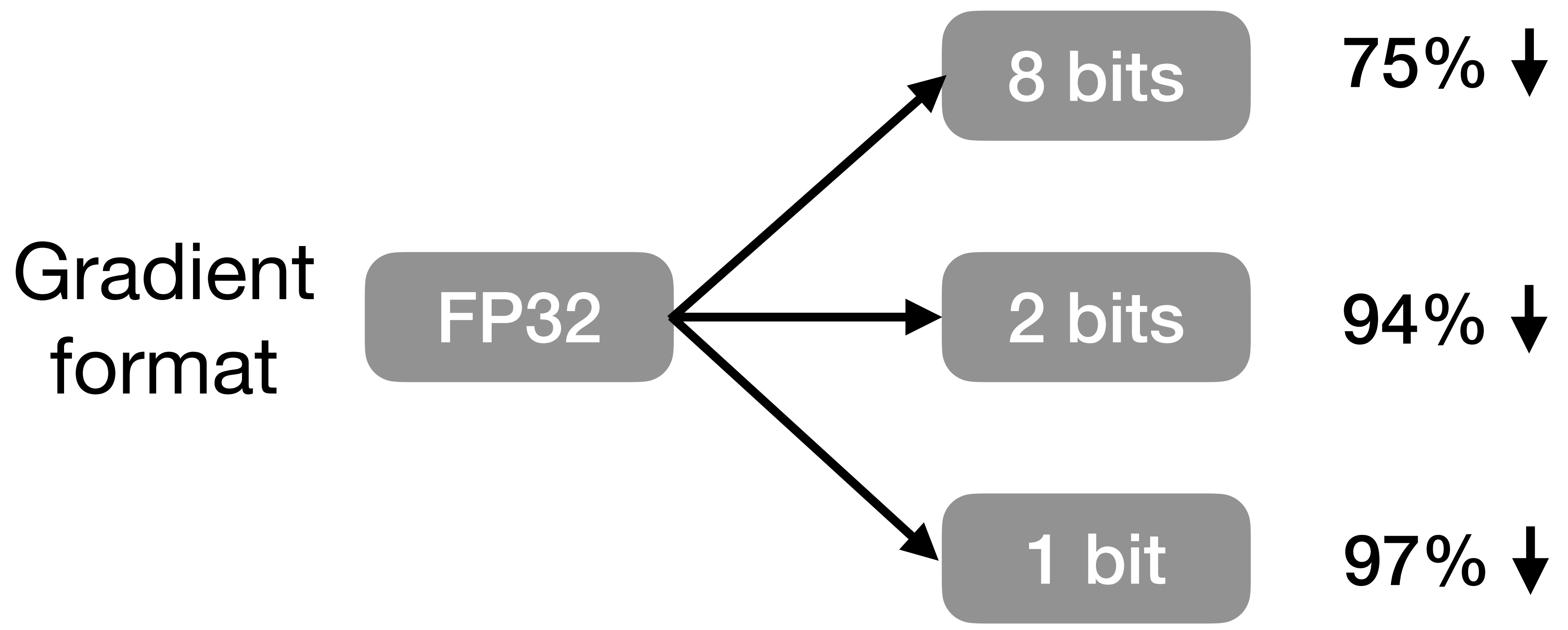

- Communication overhead can account for more than **50%** of training time [1]

[1] Gradient Compression Supercharged High-Performance Data Parallel DNN Training, SOSP ’21

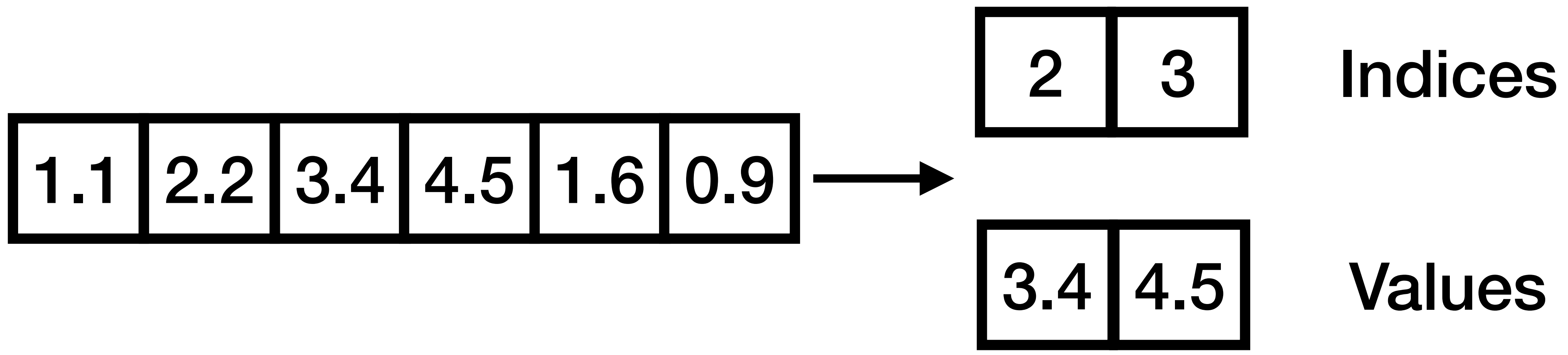
# Gradient compression (GC)

- GC shrinks communicated traffic volume
  - has negligible impact on model accuracy [1]

- Quantization



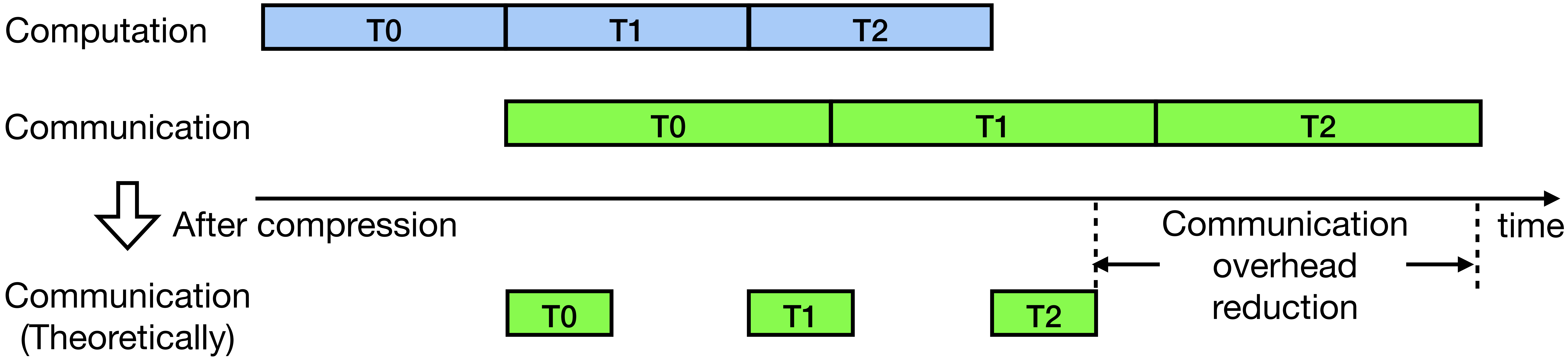

- Sparsification
  - A subset of gradients
  - Save > 99% traffic volume [2]




[1] GRACE: A compressed communication framework for distributed machine learning, ICDCS ‘21
[2] DRAGONN: Distributed Randomized Approximate Gradients of Neural Networks, ICML ‘22

# Gradient compression (GC) in theory

- GC reduces communication overhead

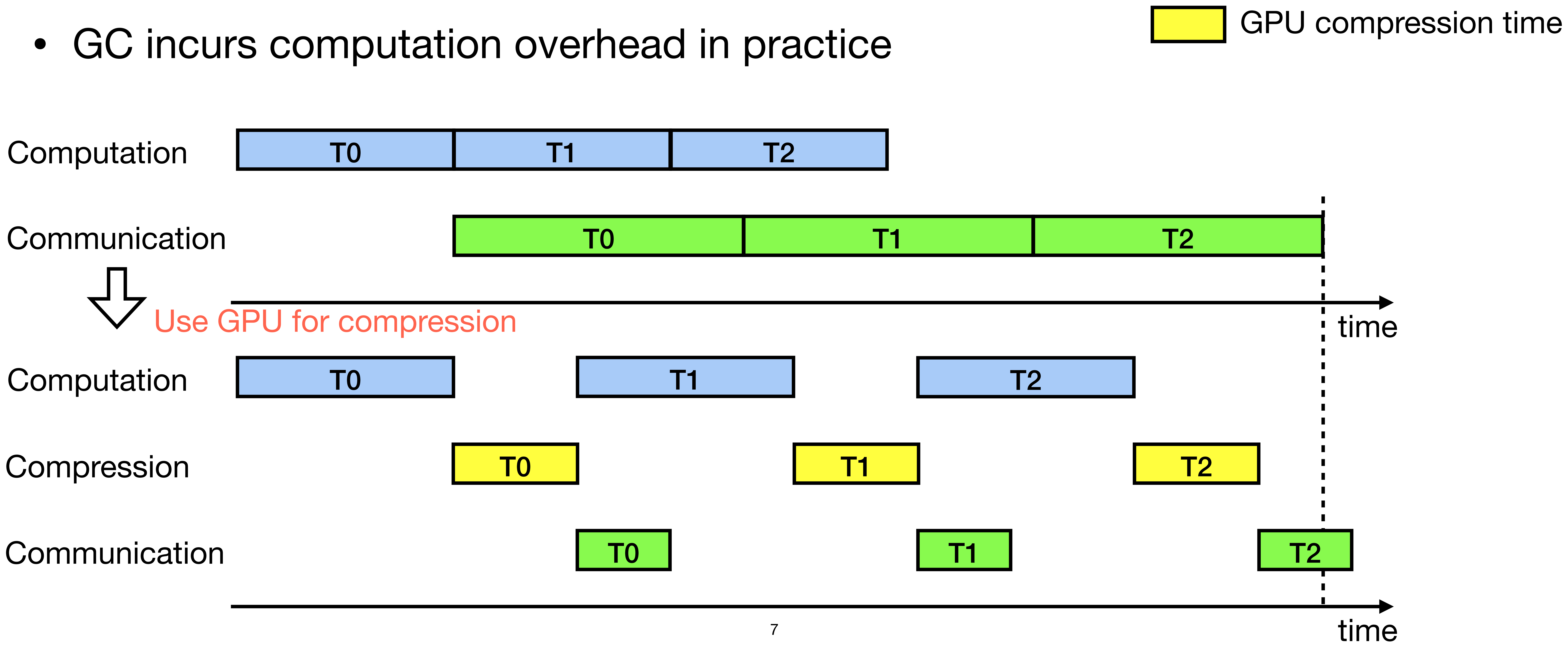



- However, GC algorithms are designed from an algorithmic perspective

# Gradient compression (GC) in reality

## Use GPU for compression

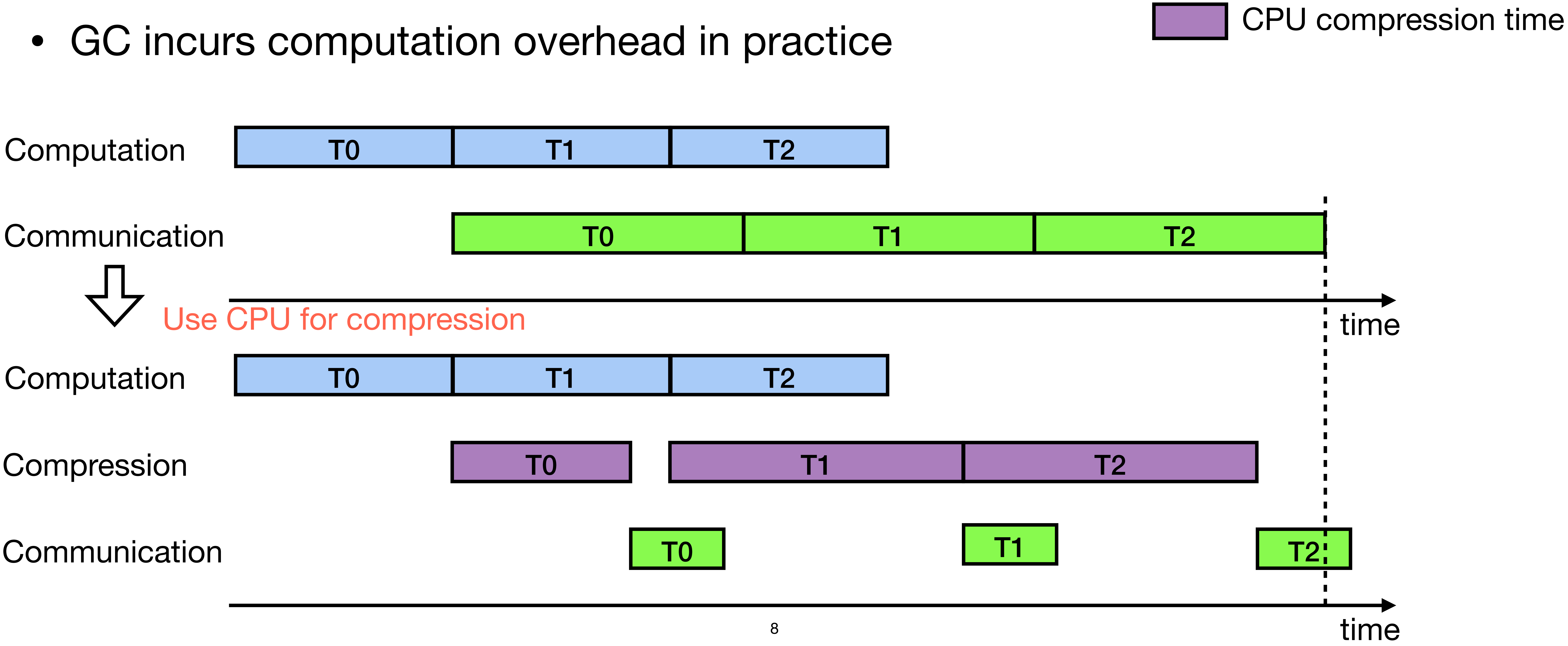
- GC incurs computation overhead in practice



# Gradient compression (GC) in reality

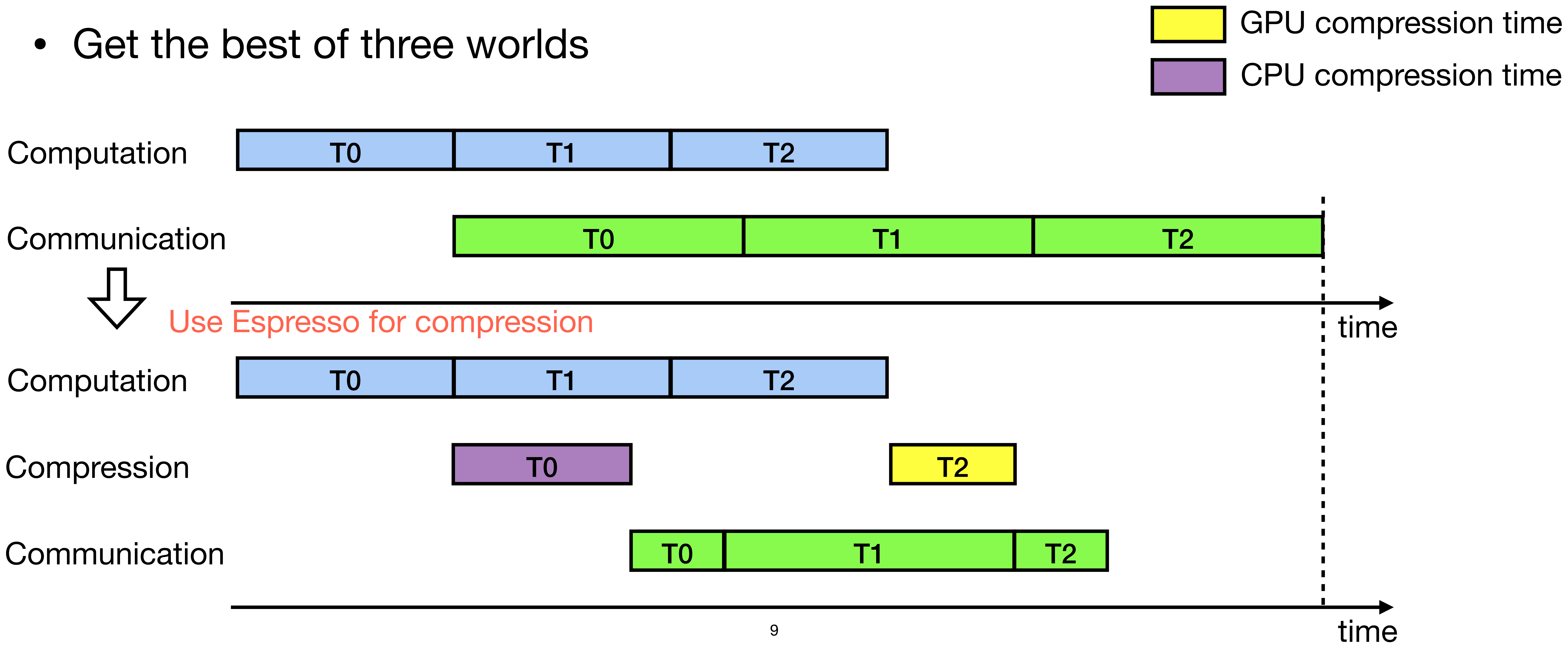
## Use CPU for compression

- GC incurs computation overhead in practice



# Gradient compression (GC) with Espresso

- Get the best of three worlds



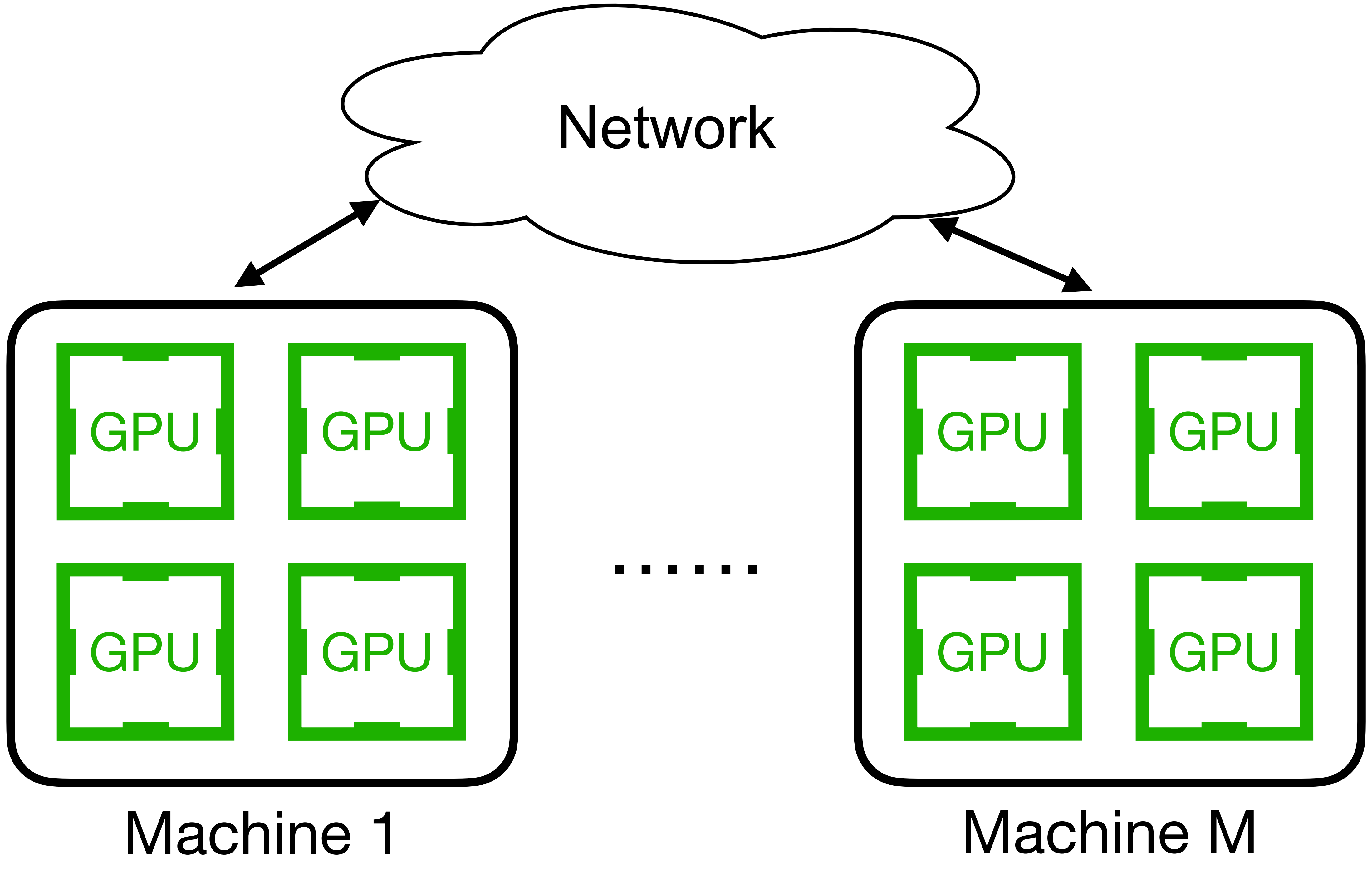
# How to choose communication schemes?

- Dimensions of communication schemes

Hierarchical communication ? Flat communication

Indivisible communication ? Divisible communication

Which communication operation to use?




- These decisions have critical impacts on training throughput

# How to find the optimal compression strategy?

## Maximize the training throughput

- Challenges

#1: How to describe the search space of compression strategies?

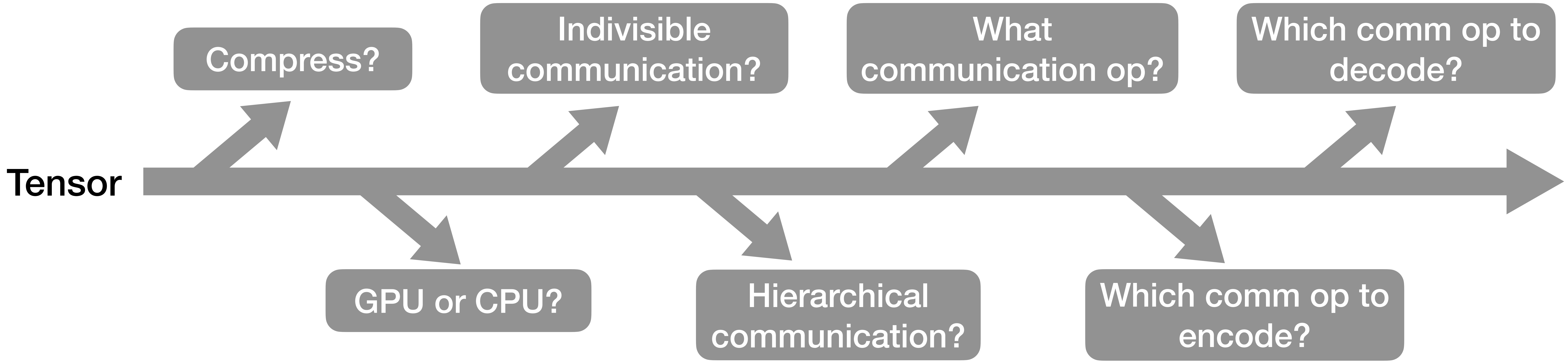
#2: How to evaluate the performance of a compression strategy?

#3: How to quickly determine a good compression strategy?

# Challenge #1

## Describe the search space of compression strategies

- Many dimensions of decisions for each tensor

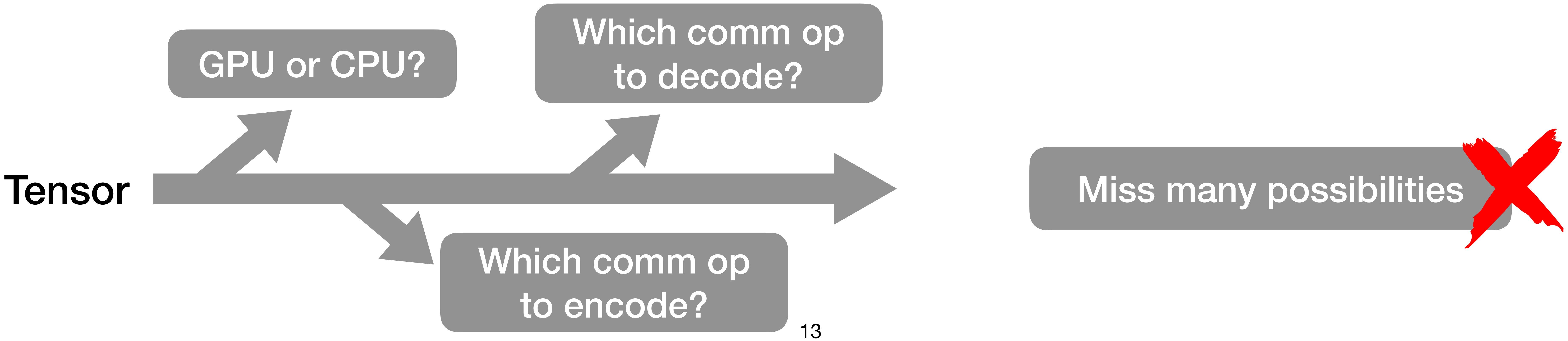


# Challenge #1 (cont'd)

## The decision order matters

- Some decisions have strict logical dependencies


- Some decision orders lead to bad choices



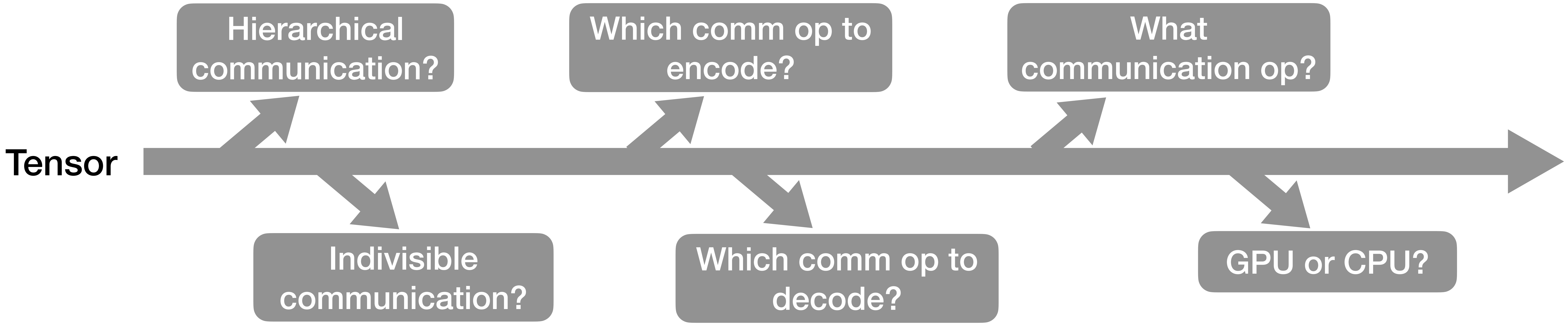
# Solution

## A decision order to maximize the number of possibilities

- Four steps
  - Step 1: determine the number of communication operations
  - Step 2: determine which operations for encoding and decoding
  - Step 3: determine what specific communication operations to use
  - Step 4: determine GPU or CPU for compression

Consider the dependencies

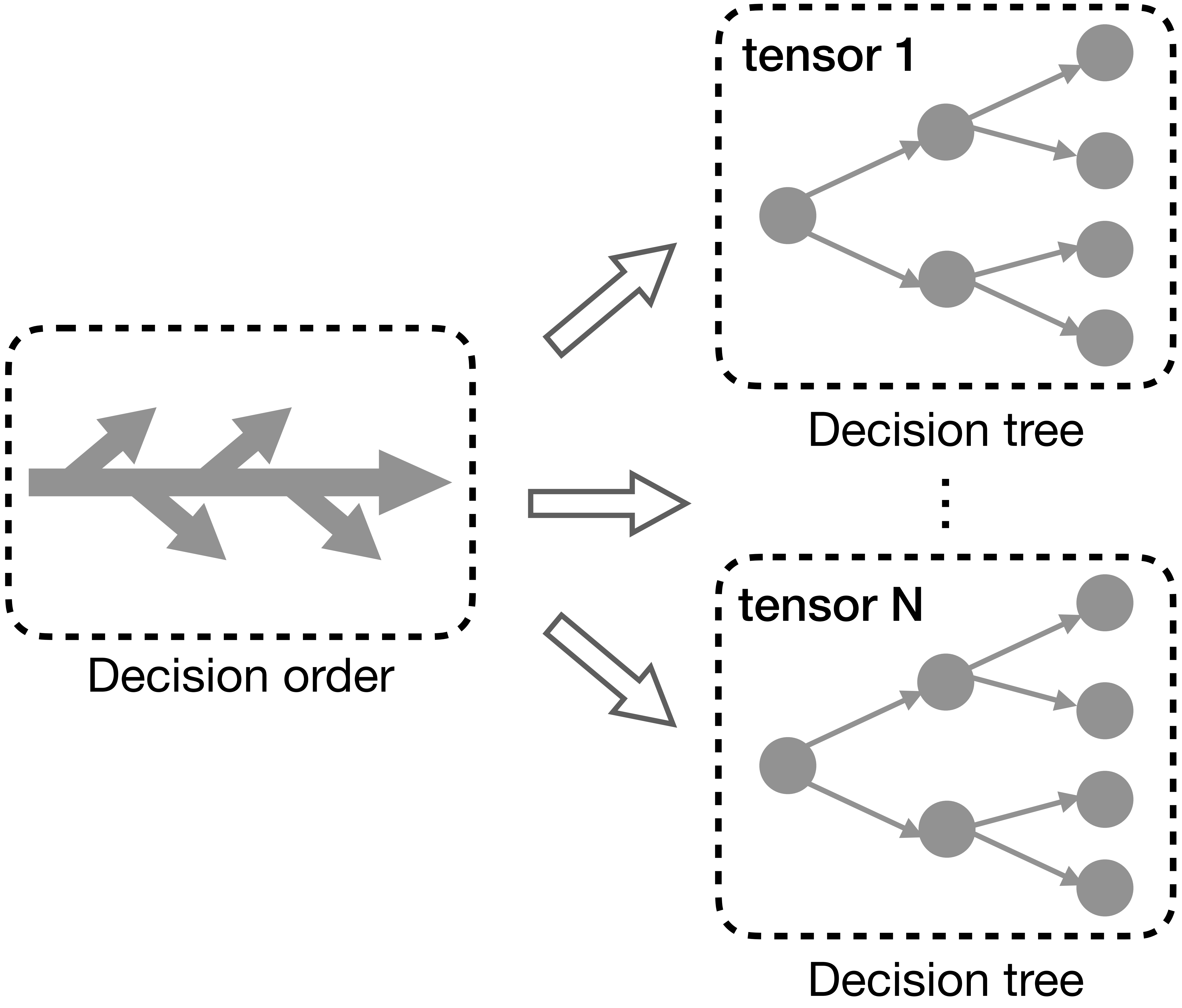
Consider all possibilities




Compress? is implies by which comm op to encode?

# Solution (cont'd)

## A tree abstraction




A decision tree describe all possible compression options of each tensor

The compression options of all tensors form a compression strategy

All decision trees together describe the search space of strategies

# Challenge #2

## Evaluate the performance of a compression strategy

- Compare different compression strategies
  - Expensive to run end-to-end training with compression strategies
- Our solution
  - Use measurements from real testbed to model training process

Tensor computation time | Tensor compression time | Tensor communication time

  - Derive the timeline of training with any strategy
  - More details in the paper

# Challenge #3

## Quickly determine a good compression strategy

- Extremely large search space

Thousands of compression options for each tensor

Hundreds of tensors in a DNN training model

- Different training jobs have different optimal strategies

Different training models

Different GC algorithms

Different hardware settings

- How to avoid searching the whole search space?
    - Minimize the computational time to determine a good strategy

# Solution

## Compression with GPUs or CPUs?

- Compressing with GPUs and CPUs have different properties

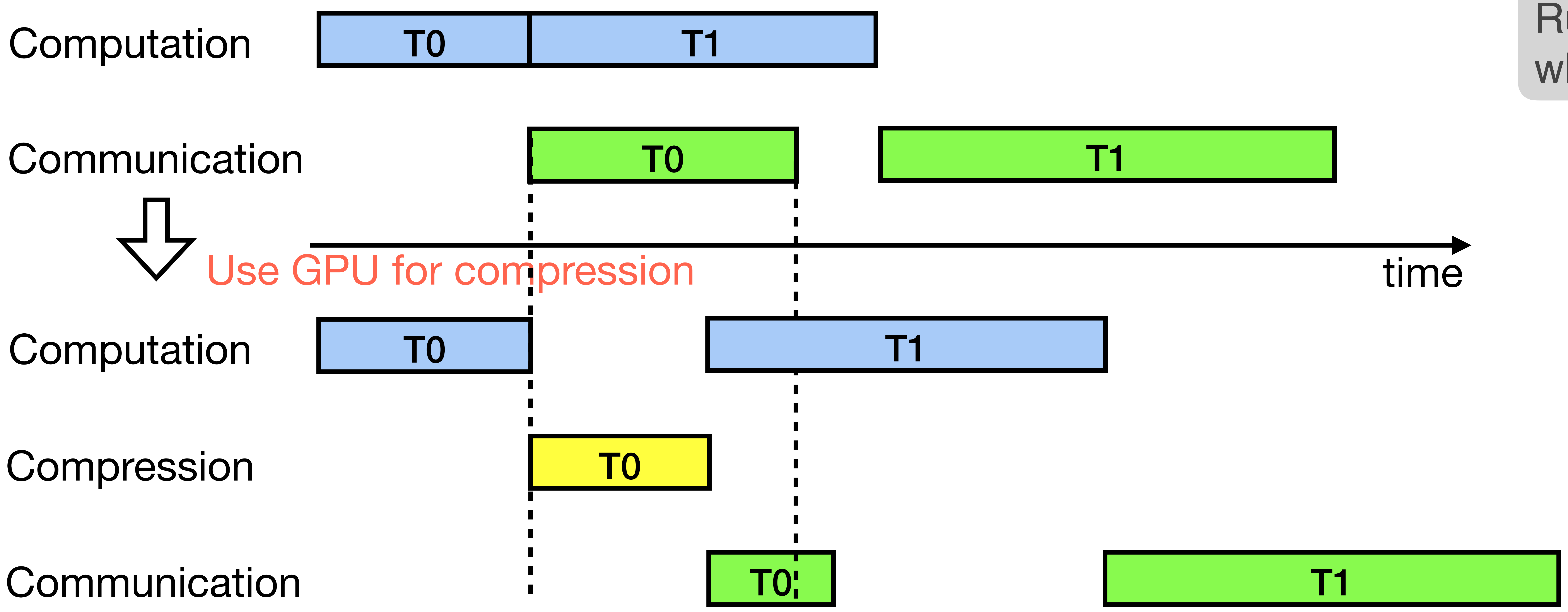
| Compress tensors with GPUs | Compress tensors with CPUs |
| --- | --- |
| • Fast compression | • Slow compression |
| • Delay computation | • Delay communication |

- A two-step decision algorithm
  - Step 1: compress tensors with GPU only
  - Step 2: offload GPU compression to CPUs to minimize compression overheads

# Solution (cont'd)

## Rule out sub-optimal strategies

- Step 1: Compress tensors with GPU only



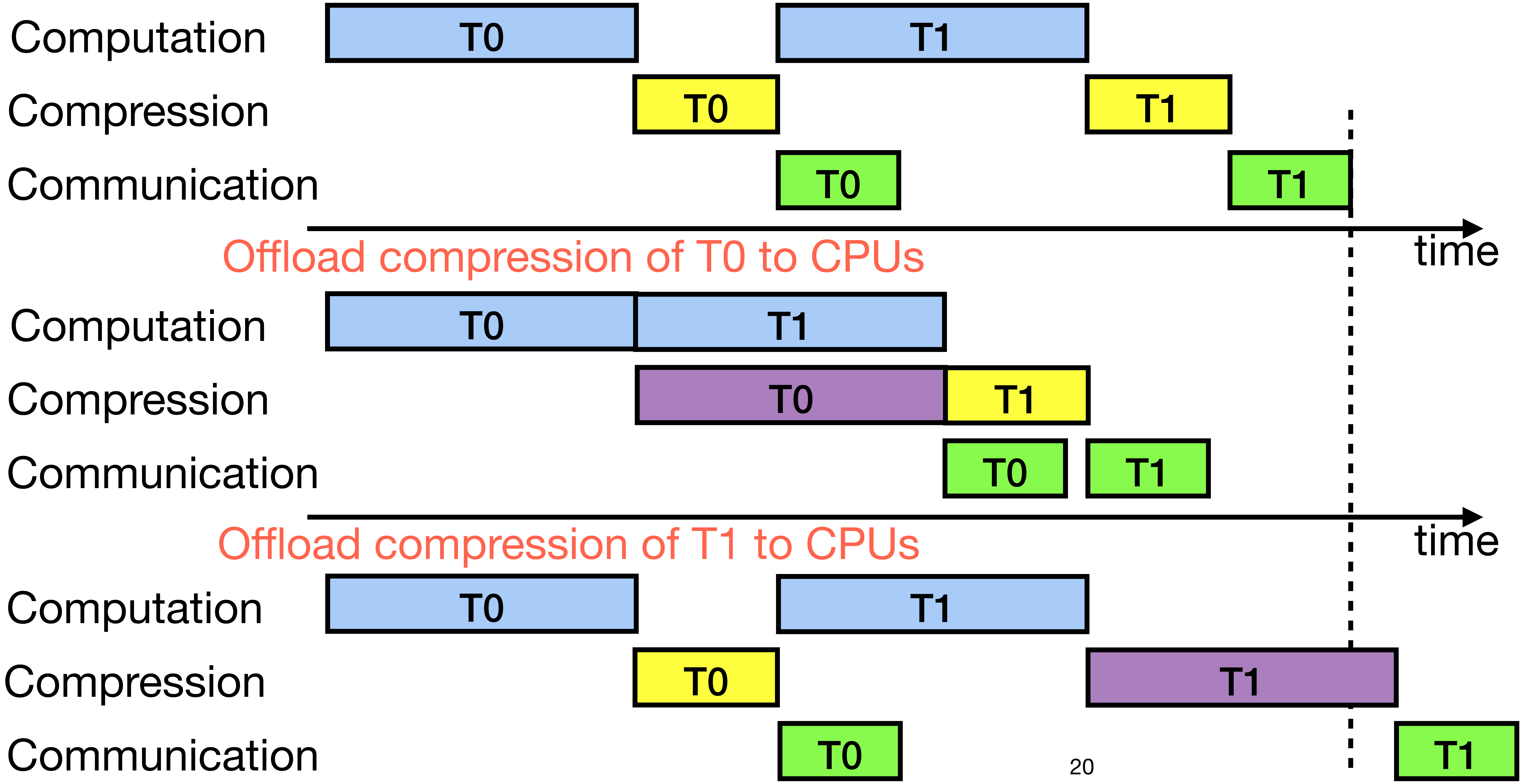

Rule out tensors with which GC cannot benefit

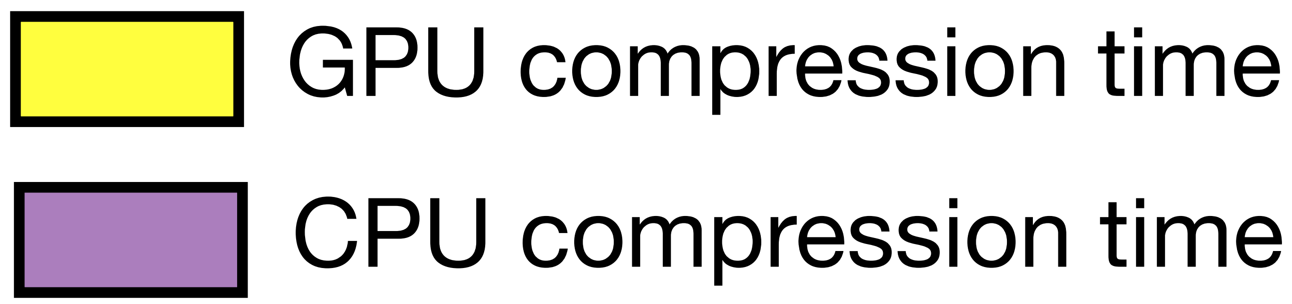
Compressing T0 with GPU is sub-optimal

# Solution (cont'd)

## Rule out sub-optimal strategies

- Step 2: Offload GPU compression to CPUs



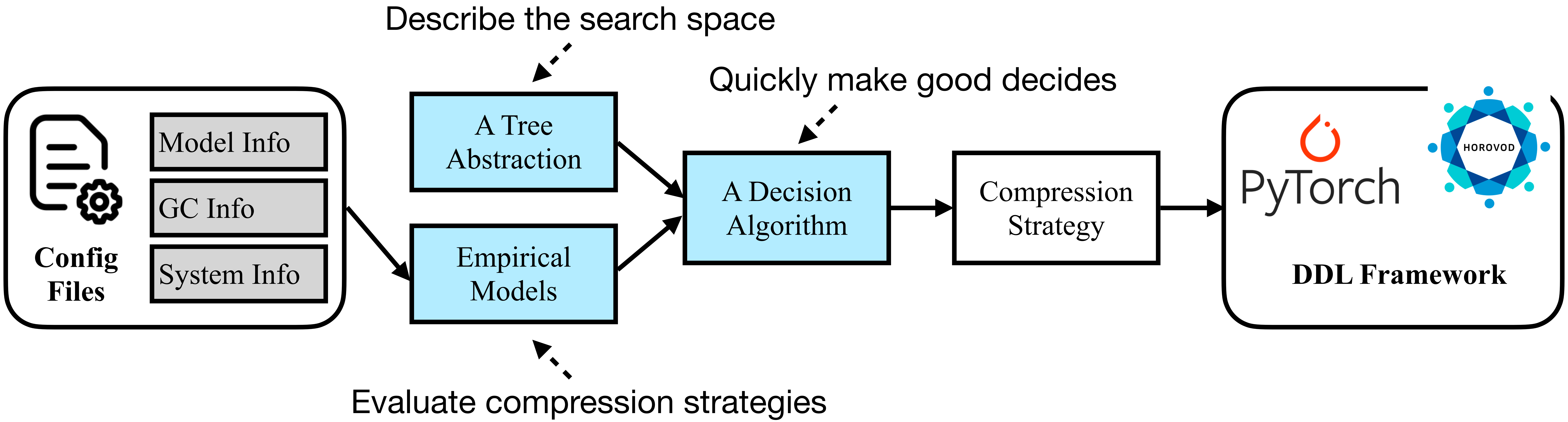




Offloading tensors earlier is better than later due to more overlapping time

An algorithm that provably finds the best CPU offloading quickly

# Espresso

- System implementation



# Results

## 25Gbps network, PCIe

- Each machine has 8 V100 GPUs


**(a)** VGG16+Randomk



**(b)** LSTM+EFSignSGD



**(c)** ResNet101+DGC

Up to 77% improvement



More evaluations in the paper

# Summary

- Fundamentally analyze the challenges of applying GC
- A tree abstraction to express the search space of compression strategies
  - Expressiveness
  - Extensibility
- A two-step decision algorithm to determine compression strategies
  - Select a near-optimal strategy in milliseconds

Thank you!

( Zhuang Wang: zhuang.wang@rice.edu )

https://github.com/zhuangwang93/Espresso