micropp

GPipe

What is GPipe?

- GPipe **splits large minibatches into smaller “microbatches”, and process them through model partitions (stages) running on different devices (GPUs)**.
-

How does it work?

划分为更小的微批次

- **Split the Minibatch**: Each input minibatch of size `B` is split into `m` equal microbatches (`B/m` samples per microbatch).
- **Pipelined Execution**: Process microbatches in order. A microbatch flows forward through stage 0, then stage 1, …, until the last stage. Once a stage finishes one microbatch, it immediately starts the next.

Gradient Accumulation

- Each microbatch is trained and its gradients computed independently. Gradients from all microbatches are *accumulated* locally before the optimizer update—giving the same effect as if training on the full batch at once.

Limitations & Details

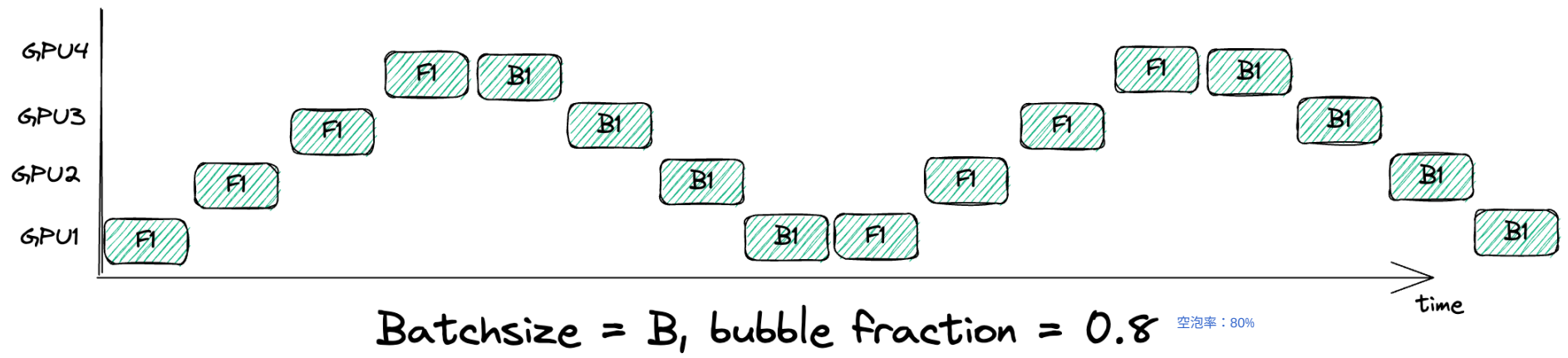
1. Pipeline Bubbles:

- The “fill” and “drain” phases of the pipeline (when not all devices are occupied) create idle slots or “bubbles”.
- The fraction of time lost to bubbles:

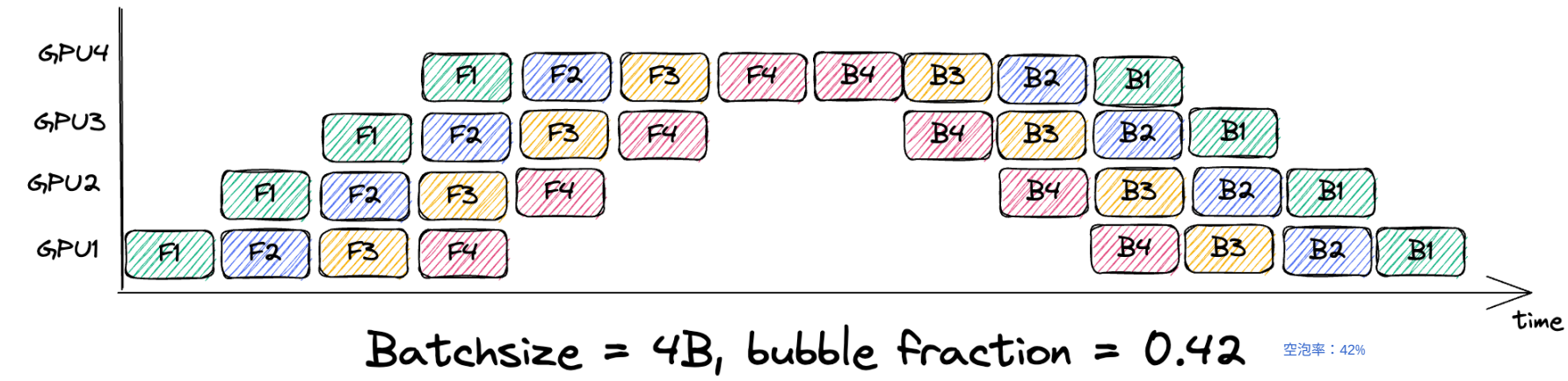
空泡的比例

$$1 - \frac{2nm}{2n(m+n-1)} = 1 - \frac{m}{m+n-1}$$

  - m为微批的个数，n为时间片数
- Increasing the number of microbatches m (flooding the pipe), is necessary for making the bubble fraction small.



空泡率：80%



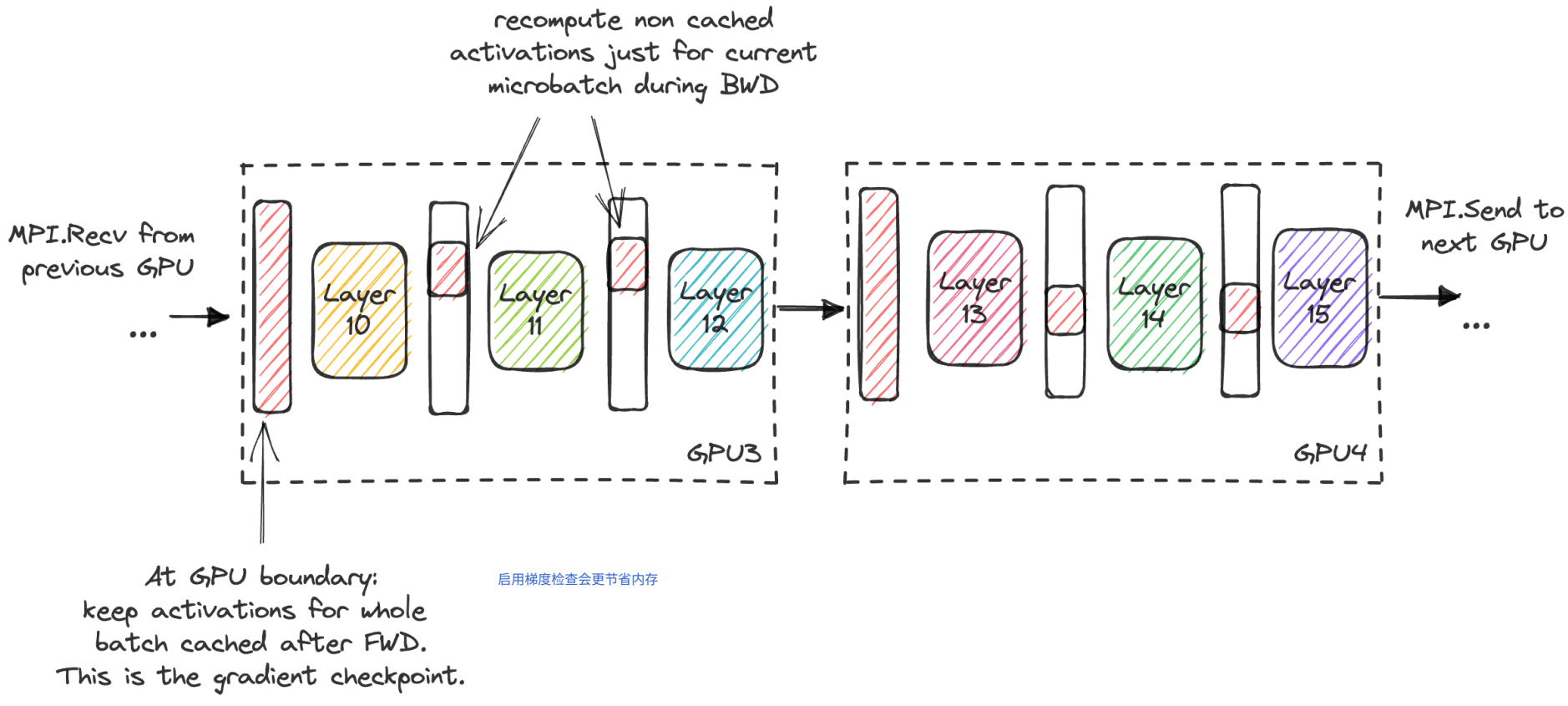
空泡率：42%

-
- Top plot is 0.75 not 0.8!

2. Memory Demand:

- Each microbatch requires storing its activations until its backward pass is finished; using more microbatches increases parallelism at the cost of memory consumption.
- In GPipe, we need to cache the activations for each microbatch from the time it was forward‘ed until the corresponding backward.

- *Gradient checkpointing* can be used to trade extra computation for reduced activation memory: instead of storing all intermediate activations, some are recomputed during backward.



启用梯度检查会更节省内存

-

3. BatchNorm:

需要统计所有批次的均值与方差，但LLM中一般不用BN

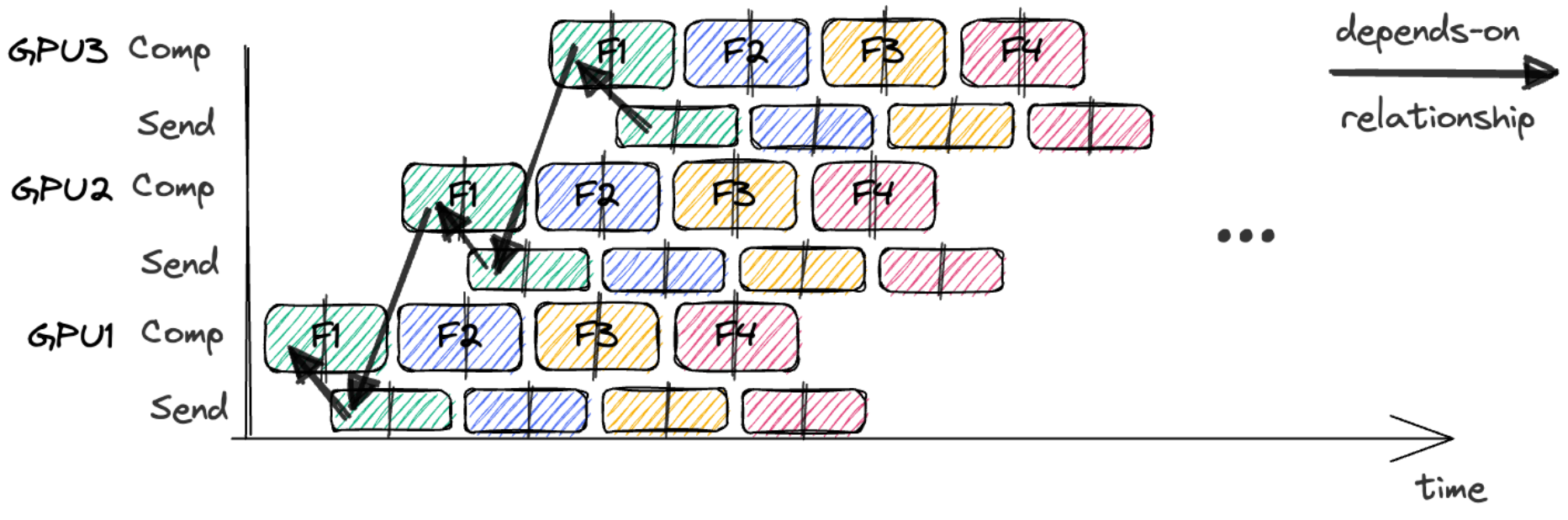
- Standard BatchNorm layers (which compute stats over the full minibatch) break the microbatch independence assumption.

4. Communication-Compute Overlap:

通信与计算重叠

- Classic GPipe does not really overlap communication (transfers between GPUs) with compute, because each stage must finish its chunk before passing it!

假设batch分为两个micro_batch,gpu先接收到m1,后才能计算m1的forward,然后发送给gpu2,gpu2再进行forward



-