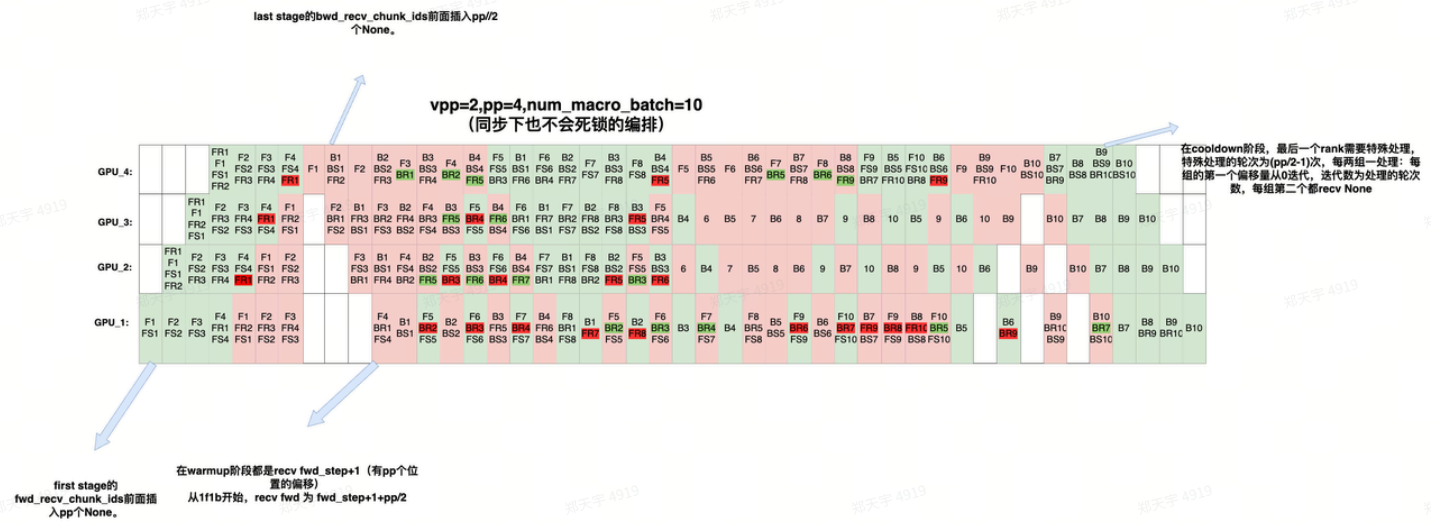
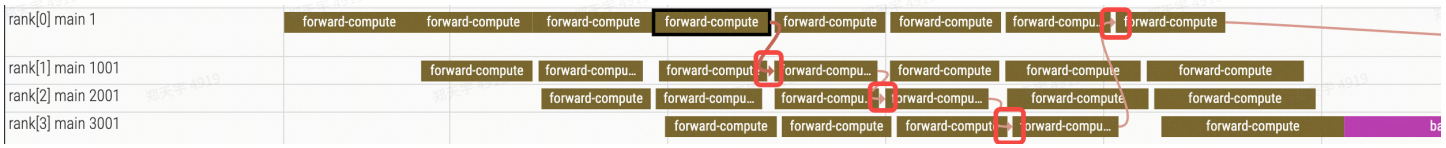
Min_Memory_Any1F1B overlap(forward&backward)



1.forward_overlap

- warmup阶段



首先可以看到在warmup阶段，有许多间隙，这是因为我们做forward的时候是即发即收，因此存在通信的延迟，因此可以做通信和计算的overlap

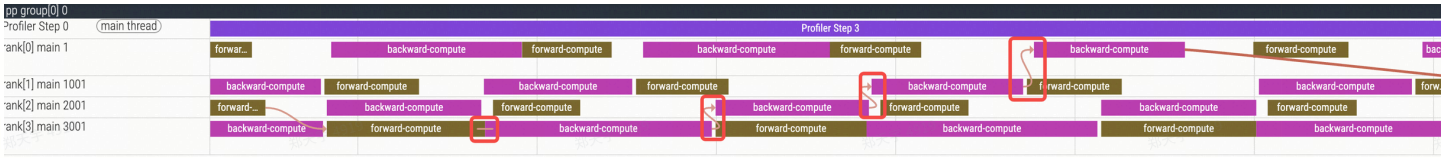
其次在1F1B阶段，因为rank0有自己的recv逻辑，都是提前接收，因此backward做完后，会立即做forward，而其他rank是backward计算完后，才做forward的接收，因此也可以做overlap。

overlap解决方法：

1. 在warmup阶段，所有pp_rank均是即发即收，因此可以提前一步进行recv_fwd，来实现overlap，但是注意，pp_0的最多recv到1F1B的第一个fwd（pp_0的1F1B有自己的recv逻辑），而其他rank可以recv到1F1B的第二个fwd（因为是先compute，后recv，因此是第二个fwd）。
2. 在1f1b阶段 pp_0需要有自定义的recv_fwd的时机（已经overlap），因此只有其它非pp_0的rank可以提前一步进行recv_fwd

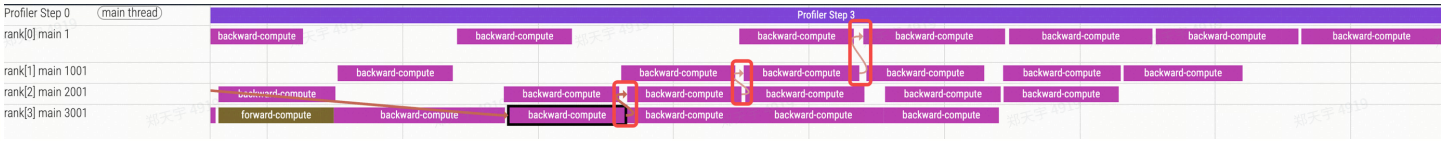
2.backward_overlap

- 1F1B阶段



由于last_rank的recv_bwd本身就会提前recv，因此forward做完后，会立即做backward，不需要提前，而其他rank都是即收即算，因此会有一个recv等待的时间，所以可以提前1步做recv。

- cooldown阶段



同样last_rank的recv_bwd本身就会提前recv，不需要提前recv。而其他rank是即收即算，因此会有一个recv等待的时间，所以可以提前1步做recv。

1. 在1F1B阶段和Cooldown阶段

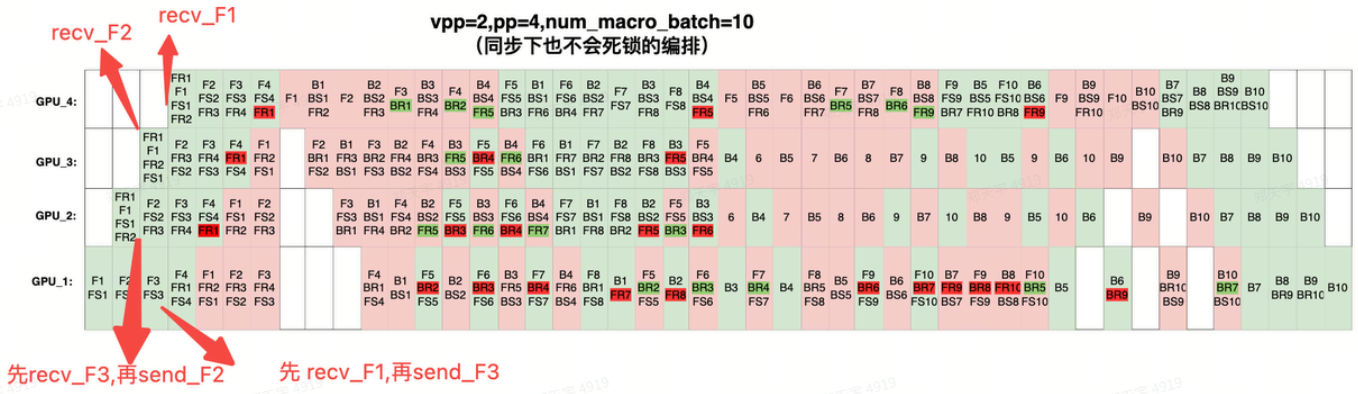
最后一个rank的轮换处理recv_bwd，改为每一步都直接recv即可（这样last rank在cooldown阶段是天然就做了overlap的）。这里需要改一下原来的编排，因为此时last_rank一定不会再向first_rank send_fwd，所以不会存在 发送fwd信息和接收bwd信息出现交错hang的现象。

其它rank提前recv一步。

3.讨论是否可以添加overlap

注意：若未开启warmup_overlap，则一定可以开启CUDA_LAUNCH_BLOCKING,但若开启了warmup_overlap，此时forward和backward的recv交错进行，需要检验是否在同步下会hang，以下也顺便做了验证。

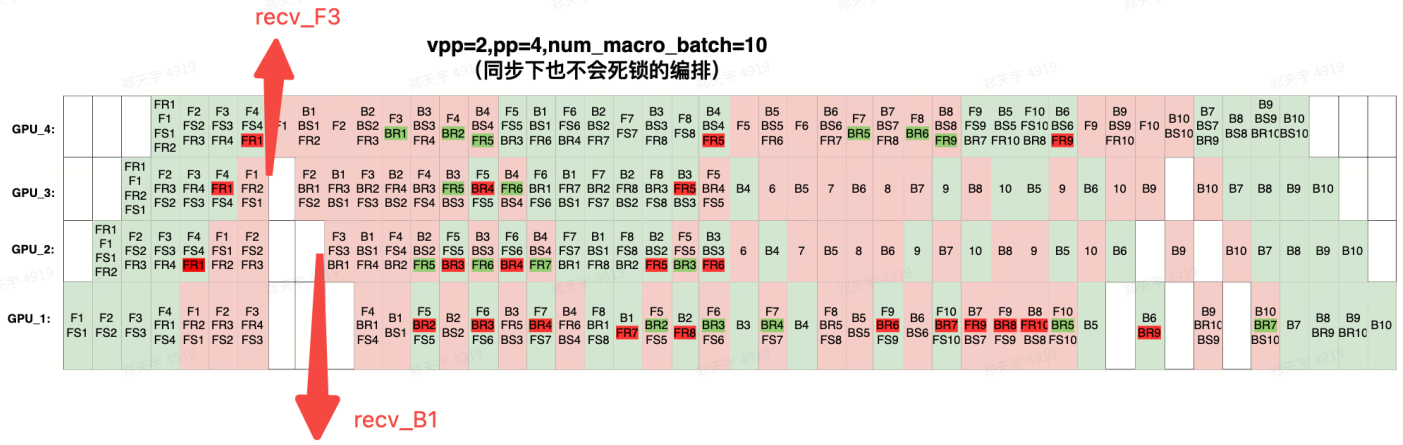
1. warmup阶段的recv_forward提前：



这里由于recv提前，会出现如上现象，此时若开启CUDA_BLOKING（强制同步通信），则出现：GPU1（等待recv_F1） -> GPU2(等待recv_F3) -> GPU3(等待recv_F2) -> GPU4(等待recv_F1),则此时所有rank都在等待，hang住。

所以不能开启CUDA_BLOKING。这里的循环hang，出现的原因是send_forward被强制延迟于recv_forward发送，而实际二者在torch封装两条通信stream里： send_backward_recv_forward和send_forward_recv_backward，相互是独立的，因此可以用异步通信解决，即不开启CUDA_BLOKING，即可实现forward的异步提前通信，从而和计算进行overlap。

2. 1F1B阶段的recv_forward与recv_backward提前：

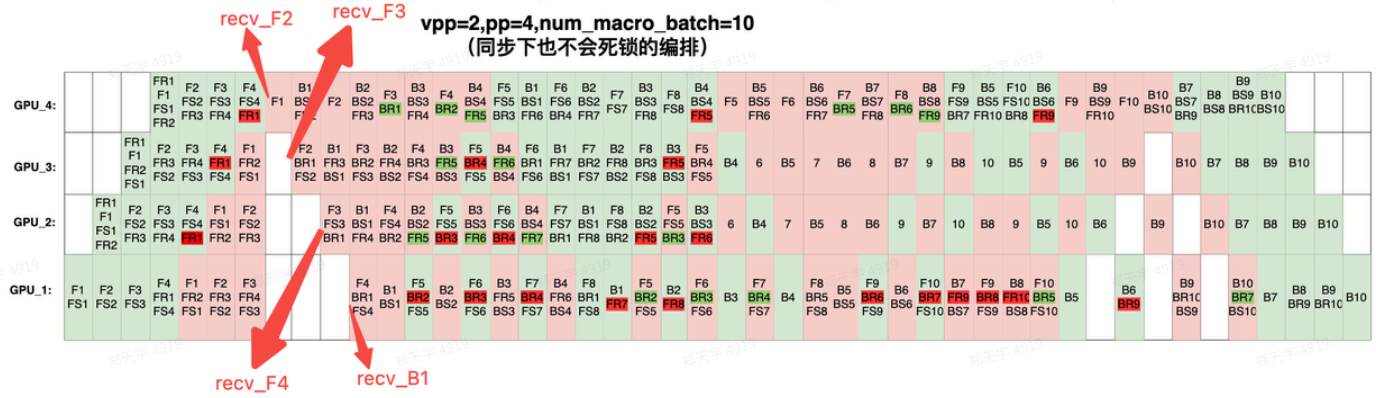


这里由于forward与backward均提前，导致GPU_3在recv_F3,而GPU_2在recv_B1,此时若开启CUDA_BLOKING（强制同步通信），GPU_3执行顺序是： recv_F3 -> send B1 ,GPU_2执行顺序是：recv_B1 ->send F3 ,因此双方都在等待，导致hang。

所以不能开启CUDA_BLOKING。那么再讨论异步解决，此时关闭CUDA_BLOKING，使用异步通信，但我们知道torch底层封装了两条stream： send_backward_recv_forward和send_forward_recv_backward。所以即使开启异步，recv_forward和send_backward，以及recv_backward与send_forward，在上述情况下实际是两组string里是串形执行，因此仍然要等待前者执行完才能执行后者，所以，即使开启异步，也仍然会hang，因此在1F1B阶段，不能同时提前forward与backward的recv。

考虑能否单独提前forward或者backward?

先回答结论：不行，因为编排有着严格的奇偶交错通信顺序，即recv_forward,recv_backward,send_forward,send_backward在每个rank上需要遵守固定的顺序才不会hang。下面以一个例子说明



如图所示，在1F1B阶段提前除了pp_0的所有forward，则 GPU_4 在等待GPU_3的F2，而GPU3在等待GPU2的F3,GPU2在等待GPU1的F4，而F4在send之前需要recv_B1,而此时的B1在GPU_4上卡着无法发出，导致通信hang。

可以看出来这里仍然是recv_fwd-send_bwd 以及 recv_bwd - send_fwd 的通信hang，因此即使开了异步，最终仍然会hang住，所以无法提前forward，而backward也同理。

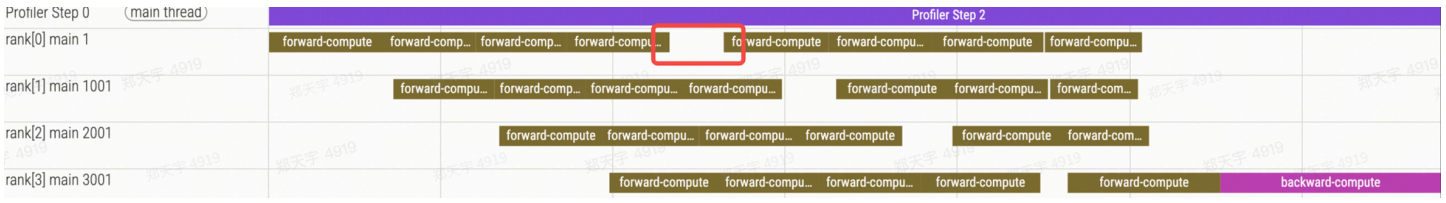
3. cooldown阶段的recv_backward提前：

cooldowen阶段的recv_backward与warmup阶段的recv_forward提前类似，均可以通过开启异步通信解决hang的问题，因此也可以做支持。

4.效果展示

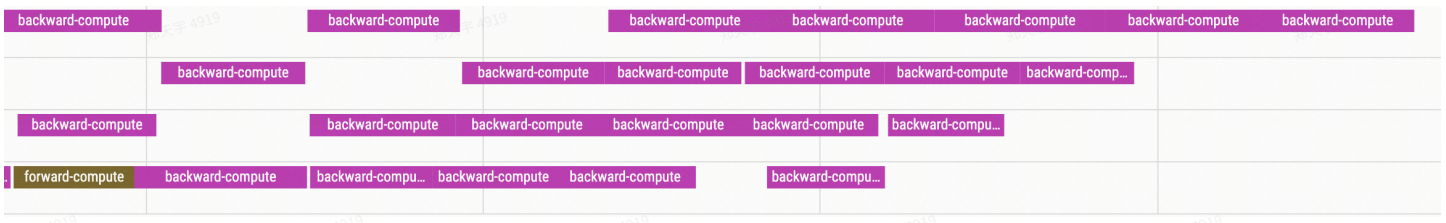
编排图：

1. warm_up阶段：



可以看到所有rank的forward都基本上紧密排列，图中存在的红框标注的bubble属于是正常的，因为rank0需要等待rank3计算完后将结果传回来，而这部分时间，是必须等待的。

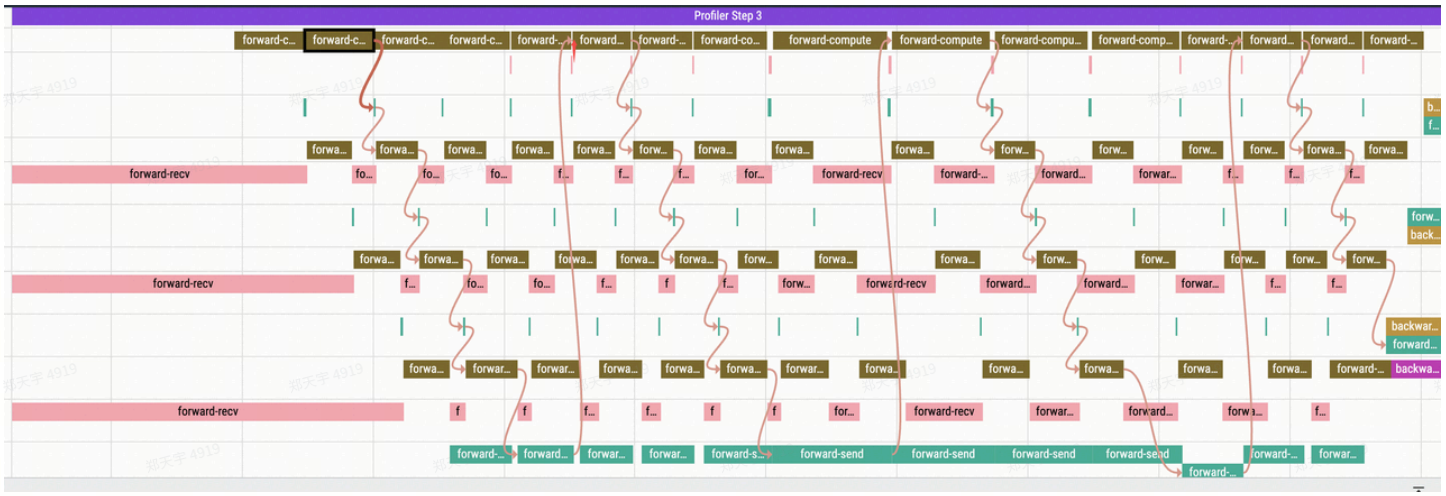
2. cool_down阶段：



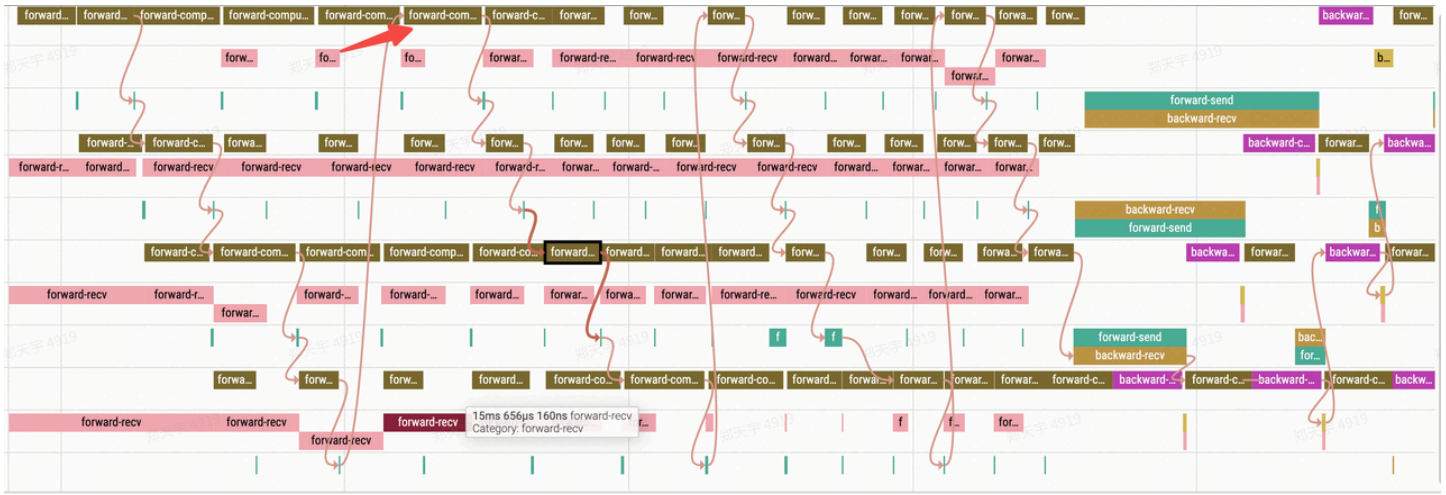
backward效果也与forward一样,每个rank最后的bacward都紧密编排。

编排+通信图：

非overlap



Overlap



可以看到通信已经overlap住了

5.实验结果

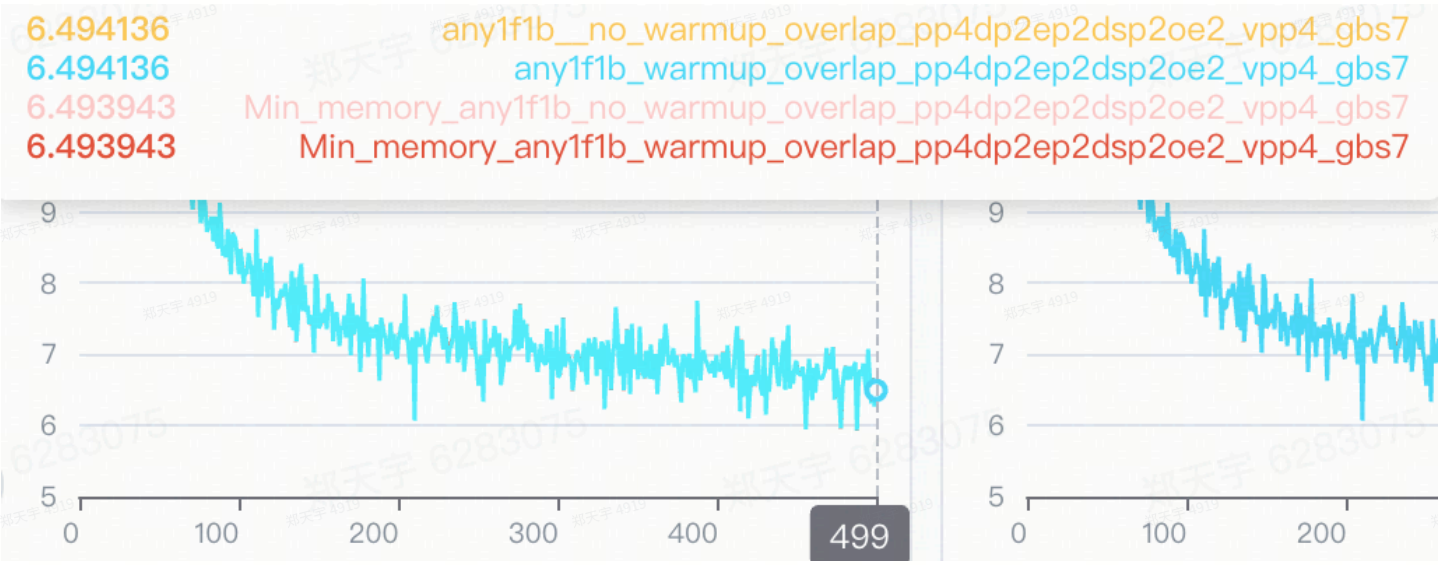
1～4为混合并行，与any1f1b类似，开启warmup时，反而导致性能下降，通信和计算会抢占带宽，导致forward计算时间延长。

1.pp4dp2ep2dsp2oe2测试：pp_size=4,vpp_size=4,micro_batch=7

1. **loss对齐：**

Min_memory的loss，开关warmup_overlap，**loss bitewise对齐，符合预期。**
https://ml.bytedance.net/experiment/tracking/detail?

Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_2473bd5c



2. **mfu性能：**

https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_76597bcd

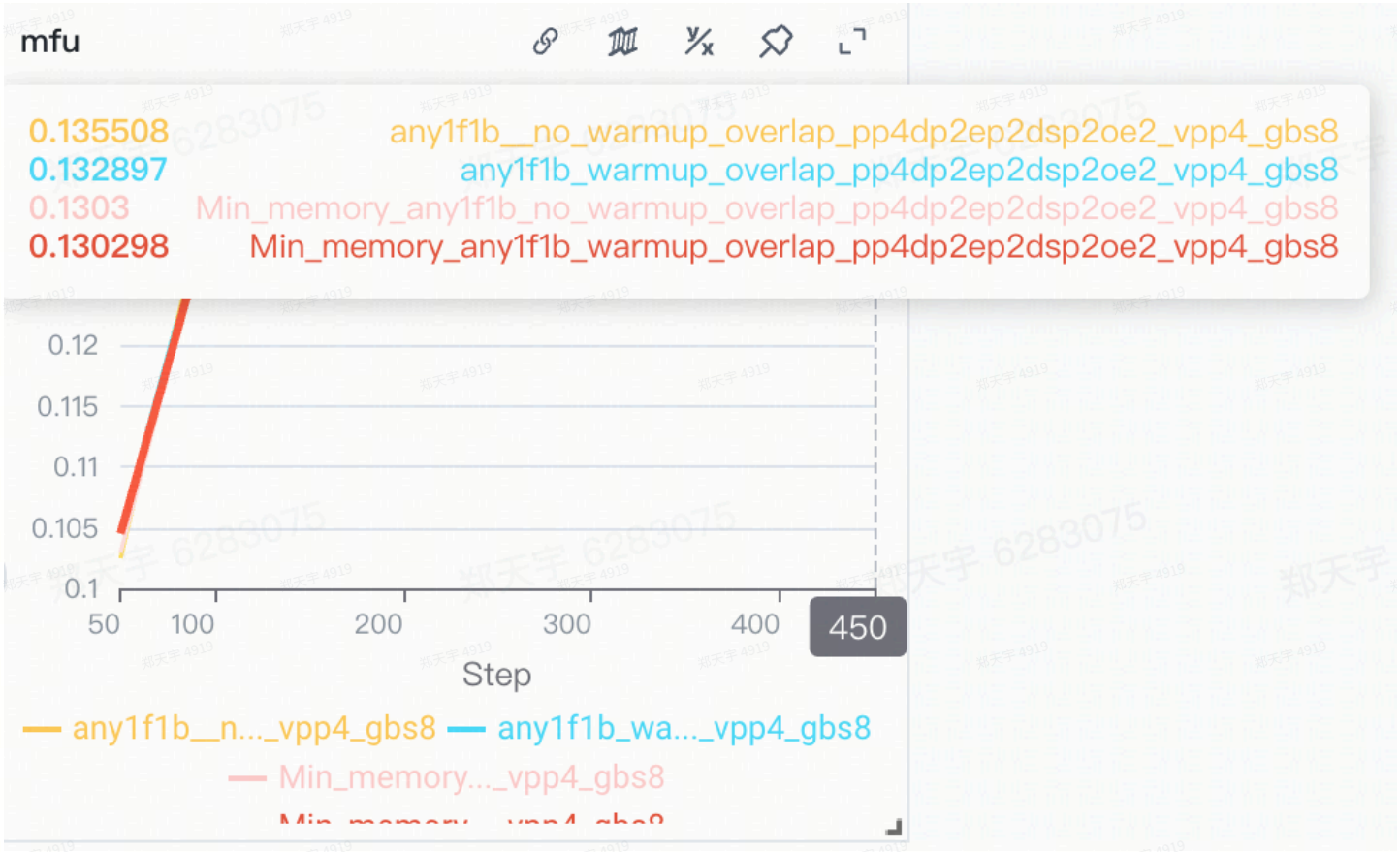
2.pp4dp2ep2dsp2oe2测试：pp_size=4,vpp_size=4,micro_batch=8

1. **loss对齐：**

Min_memory_any1f1b的loss 开关warmup_overlap 和any1f1b 开关warmup_overlap **均loss bitewise对齐，符合预期。**https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_38ca10b0

2. **mfu性能：**

https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_97ca122b



3.pp4dp2ep2dsp2oe2测试：pp_size=4,vpp_size=4,micro_batch=9

1. **loss对齐：**

Min_memory的loss，开关warmup_overlap，**loss bitewise对齐，符合预期。**
https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_808d7182

2. **mfu性能：**

https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_6ad37afd

4.pp4dp2ep2dsp2oe2测试：pp_size=4,vpp_size=4,micro_batch=10

1. **loss对齐：**

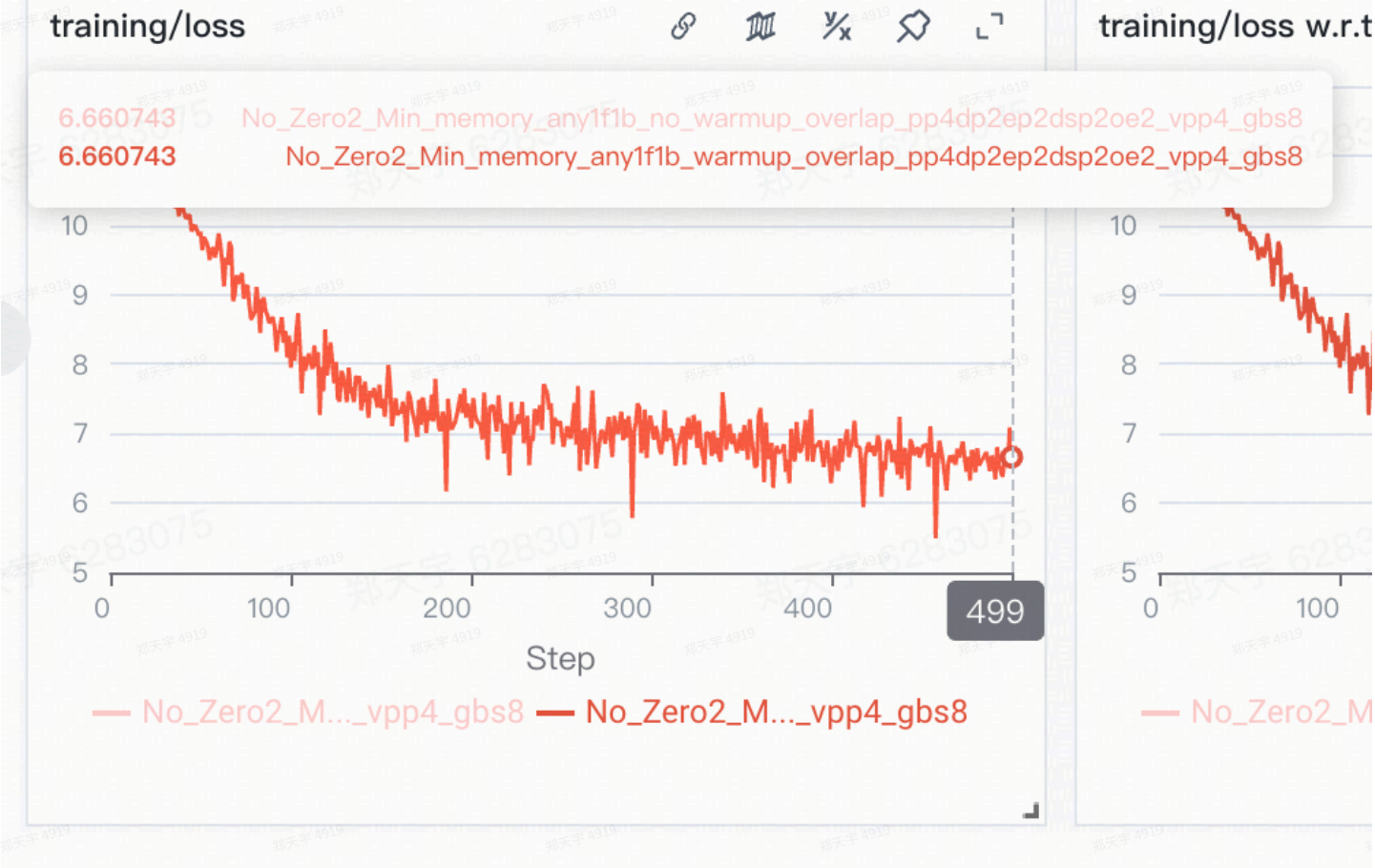
Min_memory的loss，开关warmup_overlap，**loss bitewise对齐，符合预期。**
https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_e37ba23e

2. **mfu性能：**

https://ml.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_87875e9f

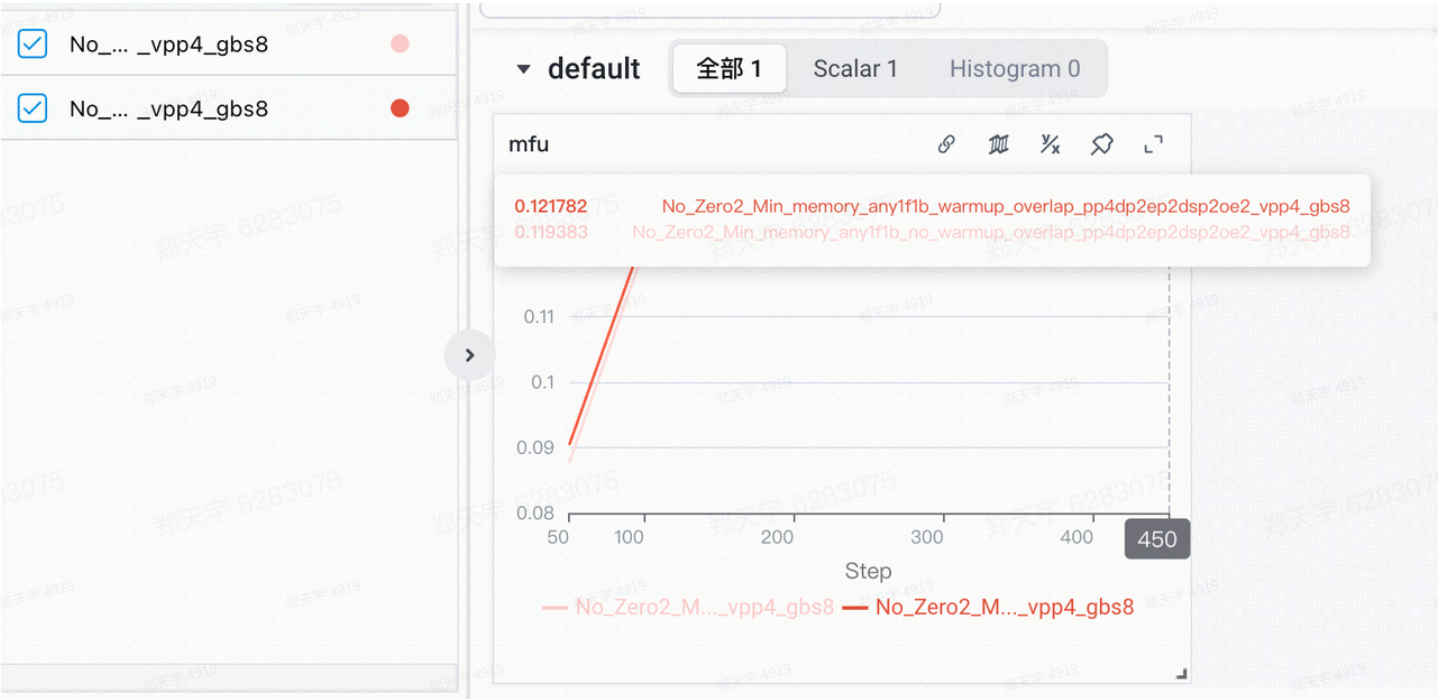
6.尝试关闭zero2

loss： loss bitewise对齐 https://seed.bytedance.net/experiment/tracking/detail?id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_5ea35198



mfu： 开了overlap效果比不开overlap效果好

https://seed.bytedance.net/experiment/tracking/detail?Id=project_20250731_ee2582db&selectedTrial=&tab=CHART&viewId=view_20260422_1407e403



目前感觉还是抢占资源，比如计算和通信分同一份带宽。