H20 两机 PD 分离 + Prefill Pipeline Parallel 测试报告

测试时间：2026.01.12

测试人：谢威宇

1. 测试环境

1.1 硬件配置

硬件名称	配置信息	数量
GPU	NVIDIA H20-3e (96GB)	8 卡/台
网络	InfiniBand (mlx5_1-4)	4 端口/台

服务器数量：2 台

- qjhs0 (172.31.0.5)：Prefill 节点
- qjhs2 (172.31.0.4)：Decode 节点

1.2 软件配置

软件名称	版本信息
操作系统	Ubuntu 22.04.5 LTS
Python	3.11.14
NVIDIA Driver	580.95.05
SGLang	0.5.6.post2
测试模型	DeepSeek-V3.2

1.3 推理配置

配置项	值
KV Cache 传输后端	Mooncake (IB RDMA)
Attention Backend	FlashAttention 3
Context Length	65535
Page Size	64
Chunked Prefill Size	16384
Memory Fraction Static	0.9

1.4 网络环境变量

```
# NCCL 配置
export NCCL_SOCKET_IFNAME=eth1
export NCCL_IB_HCA=mlx5
export GLOO_SOCKET_IFNAME=eth1
export NCCL_IB_GID_INDEX=3
export NCCL_IB_TIMEOUT=22
export NCCL_IB_RETRY_CNT=7
export NCCL_IB_SL=5
export NCCL_IB_TC=136
export NCCL_IB_QPS_PER_CONNECTION=8

# NVSHMEM 配置
export NVSHMEM_IB_GID_INDEX=3
export NVSHMEM_IB_TRAFFIC_CLASS=16
export NVSHMEM_IB_SL=5
export NVSHMEM_HCA_PE_MAPPING="mlx5_1:1:2,mlx5_2:1:2,mlx5_3:1:2,mlx5_4:1:2"
export NVSHMEM_ENABLE_NIC_PE_MAPPING=1

# IB 设备列表
export IB_DEVICE_LIST="mlx5_1,mlx5_2,mlx5_3,mlx5_4"
```

2. 测试配置

2.1 配置 A：Prefill Pipeline Parallel (PP=2)

目的：测试 Prefill 阶段使用 Pipeline Parallelism 的可行性

组件	并行策略	说明
Prefill	TP=4, PP=2	8 卡分为 2 个 pipeline stage，每 stage 4 卡
Decode	TP=8	8 卡 tensor parallel

Prefill 节点启动命令 (qjhs0):

```
python -m sglang.launch_server \
    --model-path /home/xwy/shared/DeepSeek-V3.2 \
    --disaggregation-mode prefill \
    --disaggregation-transfer-backend mooncake \
    --disaggregation-ib-device mlx5_1,mlx5_2,mlx5_3,mlx5_4 \
    --disaggregation-bootstrap-port 8998 \
    --host 172.31.0.5 \
    --port 30000 \
    --tp-size 4 \
    --pp-size 2 \
    --disable-overlap-schedule \
    --trust-remote-code \
    --page-size 64 \
    --attention-backend fa3 \
    --mem-fraction-static 0.9 \
    --chunked-prefill-size 16384 \
    --context-length 65535 \
    --enable-cache-report \
    --dist-timeout 3600
```

Decode 节点启动命令 (qjhs2):

```
python -m sglang.launch_server \
    --model-path /home/xwy/shared/DeepSeek-V3.2 \
    --disaggregation-mode decode \
    --disaggregation-transfer-backend mooncake \
    --disaggregation-ib-device mlx5_1,mlx5_2,mlx5_3,mlx5_4 \
    --disaggregation-prefill-pp 2 \
    --host 172.31.0.4 \
    --port 30001 \
    --tp-size 8 \
```

```
    --trust-remote-code \
    --page-size 64 \
    --attention-backend fa3 \
    --mem-fraction-static 0.9 \
    --context-length 65535 \
    --enable-cache-report \
    --dist-timeout 3600
```

2.2 配置 B：纯 Tensor Parallel (TP=8)

目的：作为对照组，测试传统 TP 配置的性能

组件	并行策略	说明
Prefill	TP=8	8 卡 tensor parallel
Decode	TP=8	8 卡 tensor parallel

Prefill 节点启动命令 (qjhs0):

```
python -m sglang.launch_server \
    --model-path /home/xwy/shared/DeepSeek-V3.2 \
    --disaggregation-mode prefill \
    --disaggregation-transfer-backend mooncake \
    --disaggregation-ib-device mlx5_1,mlx5_2,mlx5_3,mlx5_4 \
    --disaggregation-bootstrap-port 8998 \
    --host 172.31.0.5 \
    --port 30000 \
    --tp-size 8 \
    --trust-remote-code \
    --page-size 64 \
    --attention-backend fa3 \
    --mem-fraction-static 0.9 \
    --chunked-prefill-size 16384 \
    --context-length 65535 \
    --enable-cache-report \
    --dist-timeout 3600
```

Decode 节点启动命令 (qjhs2):

```
python -m sglang.launch_server \
    --model-path /home/xwy/shared/DeepSeek-V3.2 \
    --disaggregation-mode decode \
    --disaggregation-transfer-backend mooncake \
    --disaggregation-ib-device mlx5_1,mlx5_2,mlx5_3,mlx5_4 \
```

```
    --host 172.31.0.4 \
    --port 30001 \
    --tp-size 8 \
    --trust-remote-code \
    --page-size 64 \
    --attention-backend fa3 \
    --mem-fraction-static 0.9 \
    --context-length 65535 \
    --enable-cache-report \
    --dist-timeout 3600
```

2.3 Router 启动命令

```
python -m sglang_router.launch_router \
    --pd-disaggregation \
    --prefill http://172.31.0.5:30000 8998 \
    --decode http://172.31.0.4:30001 \
    --host 0.0.0.0 \
    --port 8000 \
    --worker-startup-timeout-secs 1200 \
    --worker-startup-check-interval 30
```

---

3. 测试用例

3.1 性能基准测试

测试目的：验证 PD 分离 + Prefill Pipeline Parallel 功能正常，对比性能表现

测试命令：

```
python perf_bench.py \
    --model DeepSeek-V3.2 \
    --tokenizer-path ~/shared/DeepSeek-V3.2/ \
    --ip 172.31.0.5 \
    --port 8000 \
    --parallel 16 \
    --number 32 \
    --input-length 6000 \
    --output-length 1000
```

测试参数：

- Input Length: 6000 tokens
- Output Length: 1000 tokens
- Concurrency: 16
- Number of Requests: 32

测试结果：

配置	Prefill 策略	TTFT (s)	吞吐量 (tok/s)	TPOT (ms)	平均延迟 (s)
PP=2	TP=4, PP=2	5.01	492.9	26.4	31.35
TP=8	TP=8	3.59	488.7	27.6	31.15

分析：

- **TTFT**：PP=2 配置的 TTFT 较高 (5.01s vs 3.59s)，这是预期的，因为 pipeline parallel 需要填充 pipeline 阶段
- **吞吐量**：两种配置吞吐量相近 (492.9 vs 488.7 tok/s)，PP=2 略高
- **TPOT**：两种配置 TPOT 接近 (26.4ms vs 27.6ms)，PP=2 略优

---

4. 测试结论

✅ Prefill Pipeline Parallel 方案可行

性能特征：

指标	PP=2 vs TP=8	说明
吞吐量	+0.9%	492.9 vs 488.7 tok/s，PP 配置略优
TPOT	−4.3%	26.4 vs 27.6 ms，PP 配置略优
TTFT	+39.6%	5.01 vs 3.59 s，PP 配置较高

结论：

- Prefill Pipeline Parallel 可提升吞吐量，代价是 TTFT 增加

- 适用于对吞吐量敏感、对首 token 延迟容忍度较高的场景（如批量处理、离线推理）
- 对于实时交互场景，纯 TP 配置的低 TTFT 更有优势