昇腾 910B4 2卡 + 鲲鹏 920 A+K 混合推理方案（内部版）

趋境科技

1. 概述

1.1 场景背景

要素	描述
目标场景	DeepSeek-R1 671B 大模型在昇腾平台的低成本推理部署
核心痛点	8卡方案成本高、资源门槛高；单机 2 卡无法独立承载 671B 模型
目标用户	成本敏感型客户、边缘部署场景、资源受限环境
竞品方案	单机 8 卡纯 NPU 方案（MindIE）、x86 + GPU 混合方案

1.2 价值主张

> 通过 NPU-CPU 异构协同推理 + 分层量化策略（NPU W8A8 + CPU INT4），实现 DeepSeek-R1 671B 在 2 卡昇腾 910B4 + 鲲鹏 920 上的可用推理，使客户能够以 1/4 的 NPU 硬件成本获得大模型推理能力，且单卡效率超越 8 卡方案。

核心差异点：

- 成本优势：2 卡 vs 8 卡，NPU 成本降低 75%
- 单卡效率领先：TPS/卡 达到 8.6，超越 8 卡方案的 6.5（提升 **33%**）
- 资源复用：充分利用服务器现有 CPU 算力和大容量内存
- 长上下文支持：支持最长 40K tokens 输入

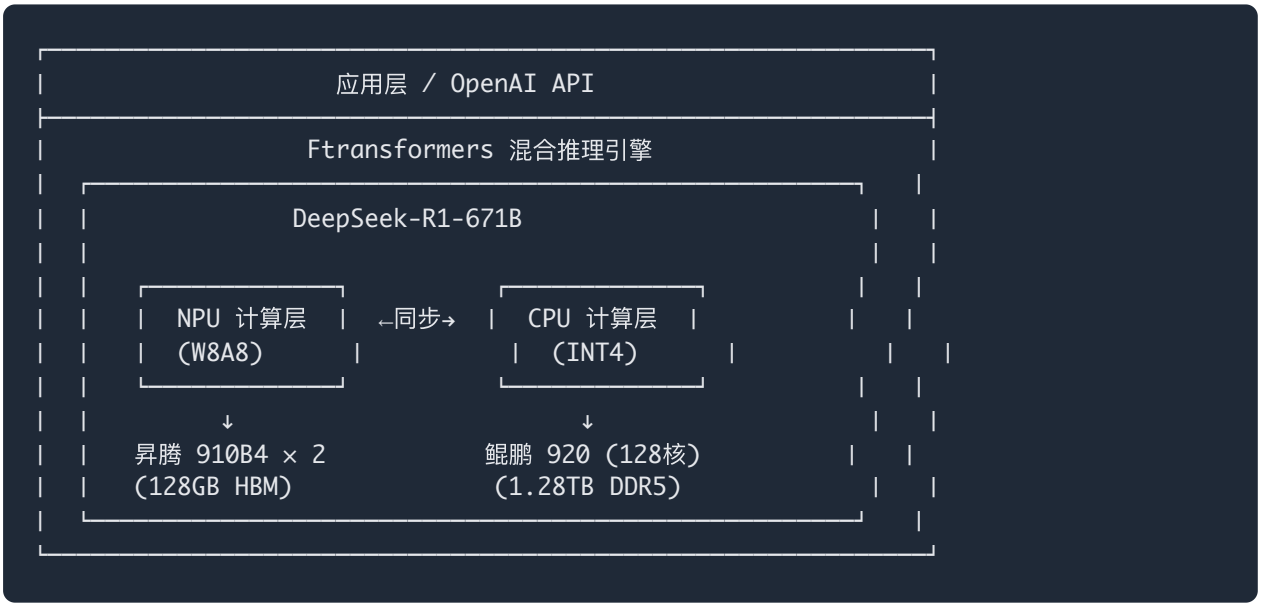
1.3 方案定位

维度	8 卡纯 NPU 方案	A+K 混合方案（本方案）
定位	性能优先	成本优先
NPU 需求	8 × 910B	2 × 910B4
适用场景	高并发、低延迟	低并发、成本敏感
单卡效率	6.5 TPS/卡	**8.6 TPS/卡**
绝对性能	TPS 51.8	TPS 17.3

---

2. 技术方案

2.1 整体架构



2.2 关键技术点

技术点 1：分层量化策略（NPU W8A8 + CPU INT4 Per-Channel）

- **是什么**：NPU 和 CPU 采用不同的量化精度
  - **NPU 权重**：W8A8（权重 8bit，激活 8bit）

  - **CPU 权重**：INT4 Per-Channel（权重 4bit，按通道量化）
- **为什么这么设计**：
  - NPU 使用 W8A8 保证计算精度和速度
  - CPU 使用 INT4 减少内存占用和带宽压力
  - Per-Channel 量化比 Per-Tensor 精度更高，减少量化误差
- **内存需求**：
  - 671B 模型 FP16 需要 ~1.3TB
  - 分层量化后：NPU 部分 ~100GB + CPU 部分 ~200GB

技术点 2：NPU-CPU 异构层级划分

- **是什么**：将模型层按计算特性划分，分配到 NPU 和 CPU
- **划分策略**：

```
NPU 计算 (W8A8):
- Attention 层 (计算密集、访存带宽敏感)
- MoE Gate 层 (需要快速路由决策)
- Embedding / LM Head

CPU 计算 (INT4 Per-Channel):
- MoE Expert 层 (161 个 Expert, 可并行)
- 全部 Expert 卸载到 CPU (num-gpu-experts=0)
```

- **为什么有效**：
  - NPU 擅长矩阵计算和高带宽访存（Attention）
  - CPU 128 核可并行处理 161 个 Expert
  - Expert 计算相对独立，适合 CPU 多线程

技术点 3：Chunked Prefill 优化

- **是什么**：将长序列的 Prefill 阶段分块处理
- **配置**：`--chunked-prefill-size 8192`
- **为什么有效**：
  - 避免长序列一次性计算导致的内存峰值
  - 平衡 Prefill 和 Decode 的资源占用
  - 支持更长的上下文输入

技术点 4：延迟解码（Defer）优化

- **是什么**：`--enable-defer` 启用延迟解码策略
- **作用**：优化 NPU–CPU 之间的数据同步时机
- **效果**：减少空闲等待，提升整体吞吐

2.3 部署配置

```
# 启动命令
export HCCL_OP_EXPANSION_MODE=AIV

python -m ftransformers.launch_server \
  --host 0.0.0.0 \
  --port 8001 \
  --model-path /home/model/DeepSeek-R1-W8A8 \
  --attention-backend ascend \
  --chunked-prefill-size 8192 \
  --tensor-parallel-size 2 \
  --device npu \
  --trust-remote-code \
  --mem-fraction-static 0.92 \
  --disable-radix-cache \
  --max-total-tokens 190000 \
  --cpu-weight-path /home/model/deepseek-int4 \
  --cpuinfer 128 \
  --subpool-count 4 \
  --num-gpu-experts 0 \
  --quantization w8a8_int8 \
  --served-model-name DeepSeek-R1-W8A8 \
  --enable-defer
```

关键参数说明：

参数	值	说明
`--tensor-parallel-size`	2	2 卡 TP 并行
`--num-gpu-experts`	0	Expert 全部卸载到 CPU
`--cpuinfer`	128	使用 128 核 CPU 推理
`--quantization`	w8a8_int8	NPU 使用 W8A8 量化
`--cpu-weight-path`	deepseek-int4	CPU 使用 INT4 权重
`--chunked-prefill-size`	8192	Prefill 分块大小
`--max-total-tokens`	190000	最大 token 数
`--mem-fraction-static`	0.92	HBM 静态分配比例

3. 测试环境

3.1 硬件配置

项目	配置
AI 加速芯片	华为昇腾 910B4（64 GiB）× 2
CPU	华为鲲鹏 920 7263Z（128 核）
内存	DDR5 64GB × 20 = 1280GB
NPU–CPU 互联	PCIe

3.2 软件配置

项目	版本
操作系统	openEuler 22.03
内核版本	5.10.0-182.0.0.95.oe2203sp3.aarch64
推理引擎	Ftransformers v3.3.1
测试工具	evalscope v1.0.1

3.3 模型配置

项目	配置
模型	DeepSeek-R1（671B）
NPU 权重量化	W8A8
CPU 权重量化	INT4 Per-Channel

---

4. 性能数据

4.1 测试说明

- **输出长度**：标准测试固定 512 tokens，长输出测试 3000 tokens
- **测试工具**：evalscope v1.0.1
- **测试模式**：关闭 Radix Cache（`--disable-radix-cache`）

4.2 短输入场景（128 tokens 输入，512 tokens 输出）

并发	端到端延迟(s)	系统吞吐(tok/s)	TTFT(s)	TPOT(ms)	TPS	平均单路吞吐(tok/s)
1	28.94	17.69	2.38	60	17.27	17.69
2	44.56	22.98	2.65	90	11.26	11.49
4	77.82	26.32	4.72	160	6.24	6.58
8	169.07	24.23	10.93	320	3.12	3.03
16	230.64	35.52	19.07	470	2.12	2.22

性能亮点：

- 单并发 TPS **17.27 tok/s**，满足基本交互需求
- 系统最大吞吐 **35.52 tok/s**（16 并发）

4.3 中等输入场景（1024 tokens 输入，512 tokens 输出）

并发	端到端延迟(s)	系统吞吐(tok/s)	TTFT(s)	TPOT(ms)	TPS	平均单路吞吐(tok/s)
1	38.44	13.32	13.53	50	20.41	13.32
2	59.00	17.36	17.38	80	12.33	8.68
4	106.93	19.15	34.21	180	5.65	4.79

观察：

- 输入增长导致 TTFT 显著增加（Prefill 阶段）
- Decode 阶段 TPOT 保持稳定（50ms）

4.4 长输入场景（512 tokens 输出）

输入长度	并发	端到端延迟(s)	系统吞吐(tok/s)	TTFT(s)	TPOT(ms)	TPS
2K	1	47.84	10.70	24.47	50	21.93
2K	2	74.63	13.72	34.01	120	8.43
2K	4	144.96	14.13	73.87	170	5.78
4K	1	57.39	8.92	33.83	50	20.75
4K	2	106.01	9.66	67.73	150	6.71
4K	4	212.03	9.66	143.80	170	6.02
8K	1	88.64	5.78	64.78	50	21.51

观察：

- Decode 阶段 TPOT 稳定在 **50ms**（单并发），与输入长度无关
- Prefill 阶段 TTFT 与输入长度正相关

4.5 超长输入 + 长输出场景（3000 tokens 输出）

输入长度	并发	端到端延迟(s)	系统吞吐(tok/s)	TTFT(s)	TPOT(ms)	TPS
10K	1	222.63	13.48	79.55	50	20.96
20K	1	311.26	9.64	163.35	50	20.28
40K	1	479.15	6.26	327.33	50	18.55

性能亮点：

- 支持最长 **40K tokens** 输入
- 超长输入下 Decode 阶段 TPS 仍保持 **18–21 tok/s**

4.6 性能总结

场景	指标	数据
单请求性能	TPS	17–21 tok/s
单请求延迟	TPOT	50–60ms
首 token 延迟	TTFT (128in)	2.38s
系统最大吞吐	吞吐量 (16并发)	35.52 tok/s
长上下文支持	最大输入	40K tokens

---

5. 对比分析

5.1 与 8 卡纯 NPU 方案对比

指标	A+K 混合 (2卡)	纯 NPU (8卡)	A+K / 8卡	说明
NPU 硬件成本	2 卡	8 卡	25%	成本降低 75%
TTFT (128in, 1并发)	2.38s	0.273s	8.7x	Prefill 较慢
TPOT (128in, 1并发)	60ms	19.3ms	3.1x	Decode 较慢
TPS (128in, 1并发)	17.27	51.8	33%	绝对性能 1/3
系统最大吞吐	35.52	806.7	4.4%	高并发差距大
单卡效率 (TPS/卡)	8.6	6.5	133%	单卡效率更高

5.2 成本效益分析

关键发现：单卡效率 A+K 方案领先 33%

```
A+K 方案：17.27 TPS / 2 卡 = 8.6 TPS/卡
8卡方案：51.8 TPS / 8 卡 = 6.5 TPS/卡

单卡效率比：8.6 / 6.5 = 1.33x
```

成本效益计算：

- 假设 910B 单卡成本为 C
- A+K 方案：2C 成本，获得 17.27 TPS，成本效率 = 17.27/2C = **8.6 TPS/C**
- 8卡方案：8C 成本，获得 51.8 TPS，成本效率 = 51.8/8C = **6.5 TPS/C**
- **A+K 方案性价比更高**

5.3 场景推荐

场景	推荐方案	理由
高并发生产环境	8 卡纯 NPU	绝对性能优先，延迟敏感
低并发、成本敏感	**A+K 混合**	成本降低 75%，性价比更高
开发测试、PoC	**A+K 混合**	快速验证，低投入
边缘部署	**A+K 混合**	硬件门槛低
长文档处理	**A+K 混合**	支持 40K 长输入，延迟可接受
实时对话（低延迟）	8 卡纯 NPU	TTFT 要求 < 1s

---

6. 总结

6.1 核心结论

1. **成本优势显著**：
   - NPU 硬件成本降低 **75%**（2卡 vs 8卡）
   - 充分利用服务器现有 CPU（128核）和大容量内存（1.28TB DDR5）
2. **单卡效率领先**：
   - A+K 方案单卡效率 **8.6 TPS/卡**
   - 超越 8 卡方案的 6.5 TPS/卡，提升 **33%**
   - 证明 NPU–CPU 异构协同的价值
3. **性能满足低并发场景**：

- 单请求 TPS **17–21 tok/s**，Decode 阶段稳定
- 首 token 延迟 **2.38s**（128 tokens 输入），适合非实时场景
- 支持最长 **40K tokens** 输入

4. **技术方案成熟**：
   - 分层量化（NPU W8A8 + CPU INT4 Per-Channel）
   - Expert 全部卸载到 CPU，充分利用 128 核算力
   - Chunked Prefill + Defer 优化

6.2 已知局限

- **首 token 延迟较高**：TTFT 2.38s（128in），不适合实时对话场景
- **高并发性能受限**：系统吞吐最高 35.52 tok/s，远低于 8 卡方案
- **Prefill 瓶颈**：长输入时 TTFT 增长明显（40K 输入需 5+ 分钟）

6.3 适用场景总结

推荐使用 A+K 混合方案：

- 成本敏感的部署环境
- 低并发（1–4 并发）应用场景
- 开发测试和 PoC 验证
- 长文档分析（延迟不敏感）
- 边缘部署或资源受限环境

不推荐使用 A+K 混合方案：

- 高并发生产环境
- 实时对话（TTFT < 1s 要求）
- 延迟敏感的在线服务

附录

A. 原始数据文件

```
data/
└── test.md     # 完整测试数据
```

B. 测试脚本

使用 `evalscope` 进行性能测试：

```
python perf_via_es10x.py \
  --ip 127.0.0.1 \
  --port 8001 \
  --parallel 1 2 4 8 \
  --number 1 2 4 8 \
  --model DeepSeek-R1-W8A8 \
  --tokenizer-path /home/model/DeepSeek-R1-W8A8 \
  --input-length 128 \
  --output-length 512
```

C. 更新记录

日期	版本	更新内容
2025-12-29	v1.0	完成性能测试
2025-12-30	v1.1	整理内部版报告