北京智源人工智能研究院
BEIJING ACADEMY OF ARTIFICIAL INTELLIGENCE

跨芯片训练与推理自适应优化技术

跨芯片训练自适应优化技术

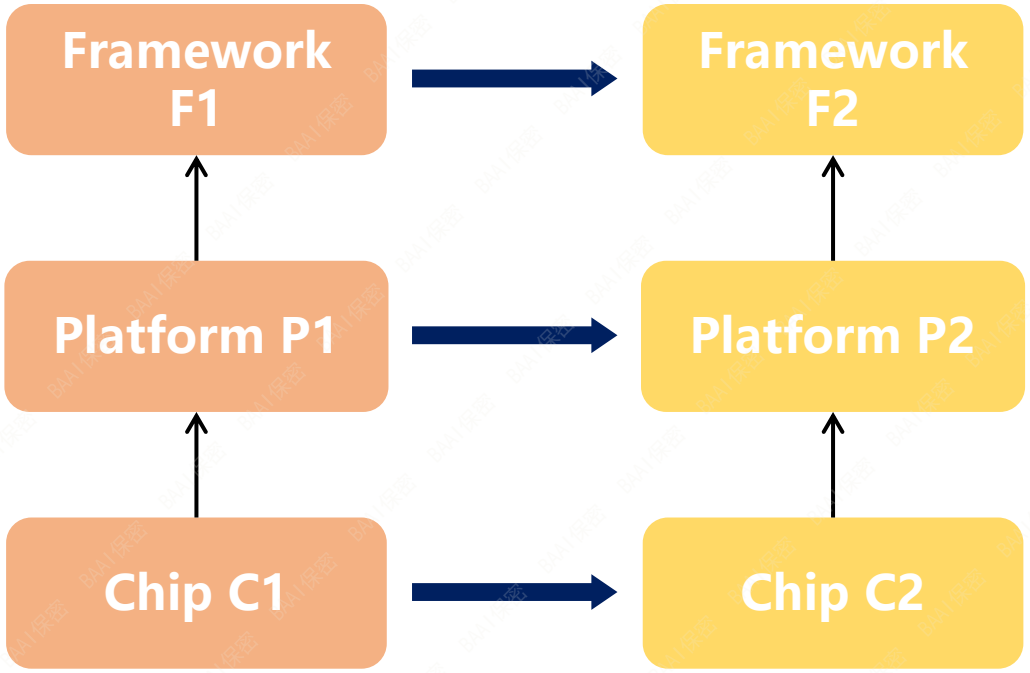
背景

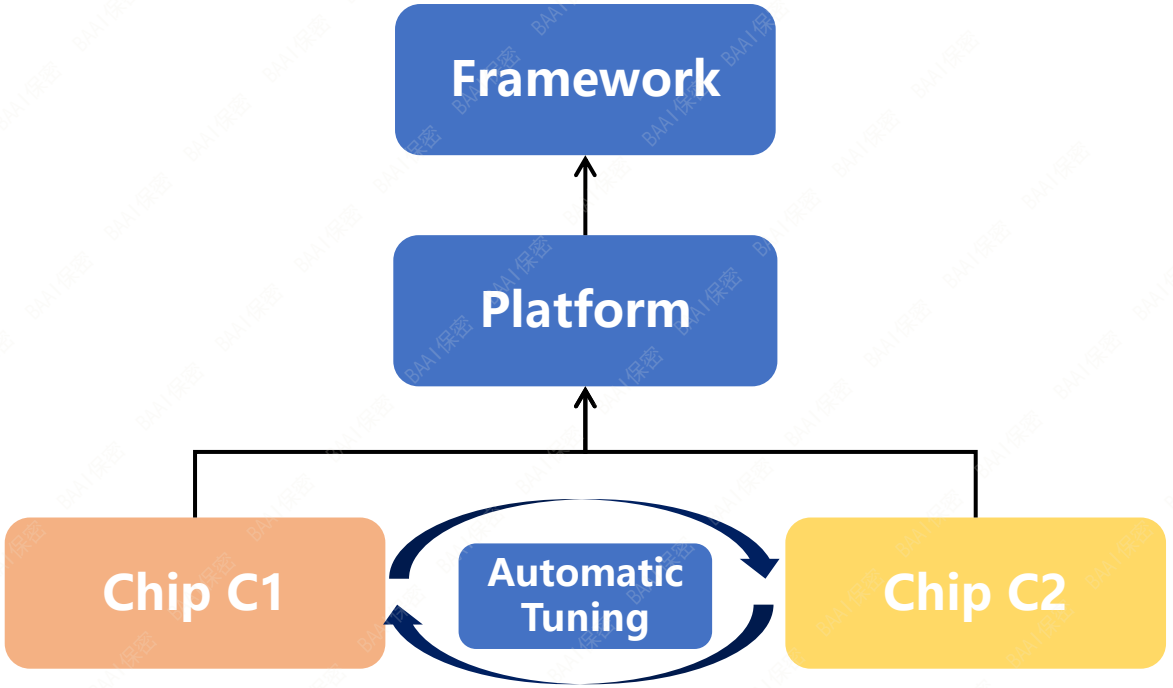
- 多元算力时代已经到来

NVIDIA AMD intel 天数智芯 Iluvatar CoreX Cambricon 寒武纪 昆仑芯 KUNLUNXIN 摩尔线程 MOORE THREADS METAX 沐曦 HYGON Ascend

- 不同模态、不同大小的模型

跨芯片迁移挑战





- 框架与平台的变更需要投入大量额外的学习与开发成本。
- 针对不同芯片的并行化与优化策略需要适配调整，过程高度依赖专家经验。

- 使用兼容多种硬件的框架与平台，可避免代码适配工作并降低学习成本。
- **自动调优机制可在新芯片训练过程中自动选取最优的并行化与优化方案。**

确定最优并行策略的挑战

✗ **挑战1:**专家策略缺乏普适性

- 机器类型发生变化，原并行策略可能无法使用或者不是最优
- 机器数量发生变化，原并行策略可能无法使用或者不是最优

✗ **挑战2:** 专家策略非全局最优

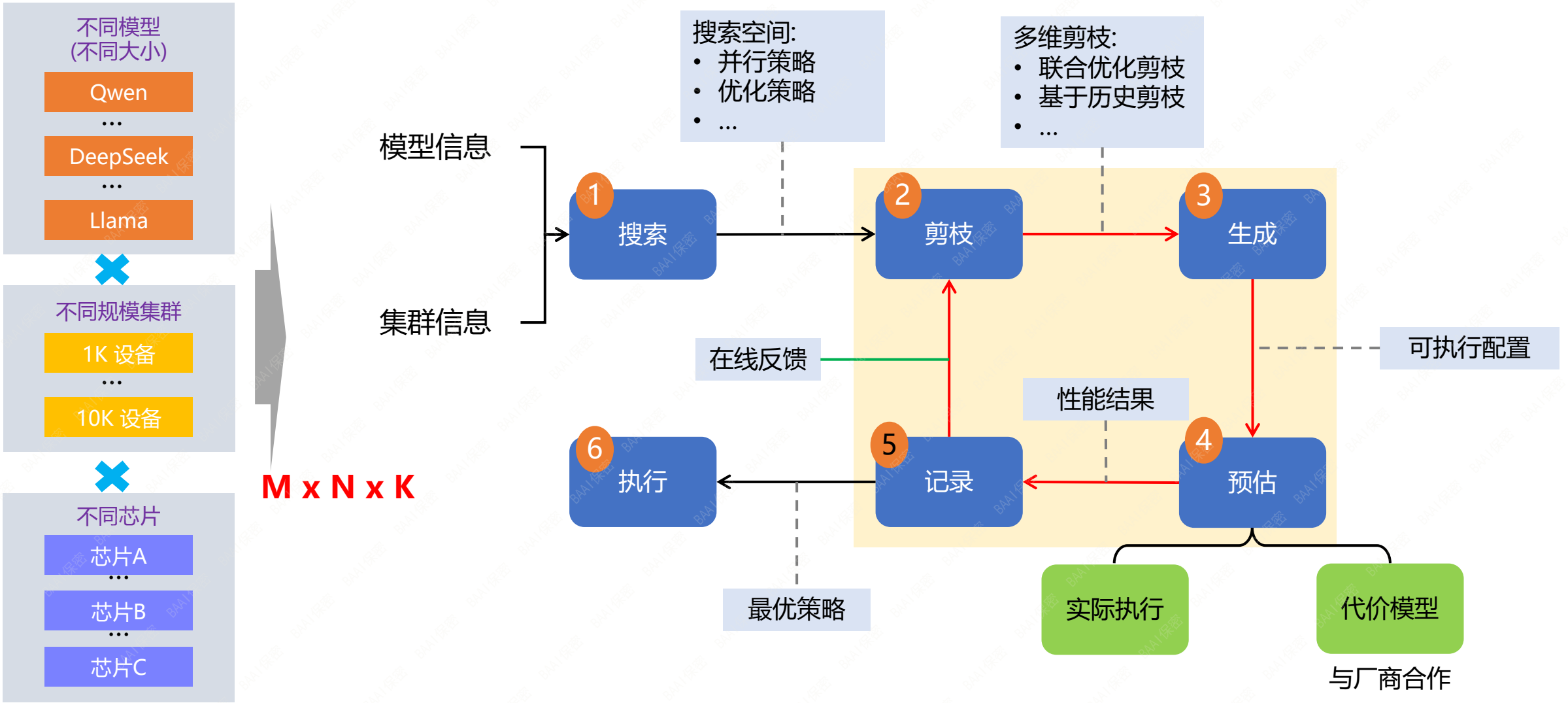
- 人工经验手动设计有限个实验，很难取得最优性能
- 大模型大集群训练下的小幅度性能提升，也会带来很大的成本节省

✗ **挑战3:** 搜索空间大，确定最优策略时间长

- 并行模式正交并且可排列组合
- 没有考虑并行策略之间的联系

✓ **解决方案: FlagScale AutoTuner Based on Memory model and Historical Pruning**

自动调优机制


不同模型
(不同大小)
Qwen
…
DeepSeek
…
Llama
不同规模集群
1K 设备
…
10K 设备
不同芯片
芯片A
…
芯片B
…
芯片C
M x N x K
模型信息
集群信息
搜索空间:
• 并行策略
• 优化策略
• …
多维剪枝:
• 联合优化剪枝
• 基于历史剪枝
• …
1
搜索
2
剪枝
3
生成
4
预估
5
记录
6
执行
在线反馈
可执行配置
性能结果
最优策略
实际执行
代价模型
与厂商合作

搜索空间

- 数据并行
  - DataParallel
  - Distributed Optimizer
- 张量并行
  - TensorParallel
  - SequenceParallel
- 流水线并行
  - 1F1B
  - Interleaved 1F1B
- 上下文并行
- 专家并行
- 梯度累加
- 细粒度重计算
  - Selective Recompute
  - Full Recompute
  - ...

```yaml
auto_tuner:
  space:
    data_parallel_size: "auto"
    use_distributed_optimizer: [true, false]
    tensor_model_parallel_size: [2, 4, 8]
    sequence_parallel: [true]
    pipeline_model_parallel_size: "auto"
    num_layers_per_virtual_pipeline_stage: [1]
    context_parallel_size: "auto"
    expert_model_parallel_size: [1]
    micro_batch_size: "auto"
    use_recompute: [true]
    recompute_method: "auto"
    recompute_granularity: "auto"
    recompute_num_layers: "auto"
  control:
    max_time_per_task: 300
    train_iters: 5
    max_time: 600
```

细粒度显存预估

- 峰值显存 = 常驻显存 + 激活张量显存
- 常驻显存（模型参数+优化器参数）为分配到每张卡上的经过切分后的模型参数和优化器参数总和
- 激活张量为执行反向前未被释放的激活张量的总和

Item	Params	Activation (Peak Mem with Parallelism)
GQA/MLA QKV Proj	✓	✓
GQA/MLA Core Attn	-	✓
Out Proj	✓	✓
FFN Up&Down	✓	✓
Experts	✓	✓
Shared Experts	✓	✓
Layernorms	✓	✓
MTP Module	✓	✓

DeepSeekV3-16B-A3B			
	per-layer param(B)	num layers	params(B)
Embedding	0.3111	1	0.3111
MLA	0.0138	27	0.3726
MoE	0.5711	26	14.8486
MLP	0.0692	1	0.0692
Output	0.3111	1	0.3111
Total			15.9126

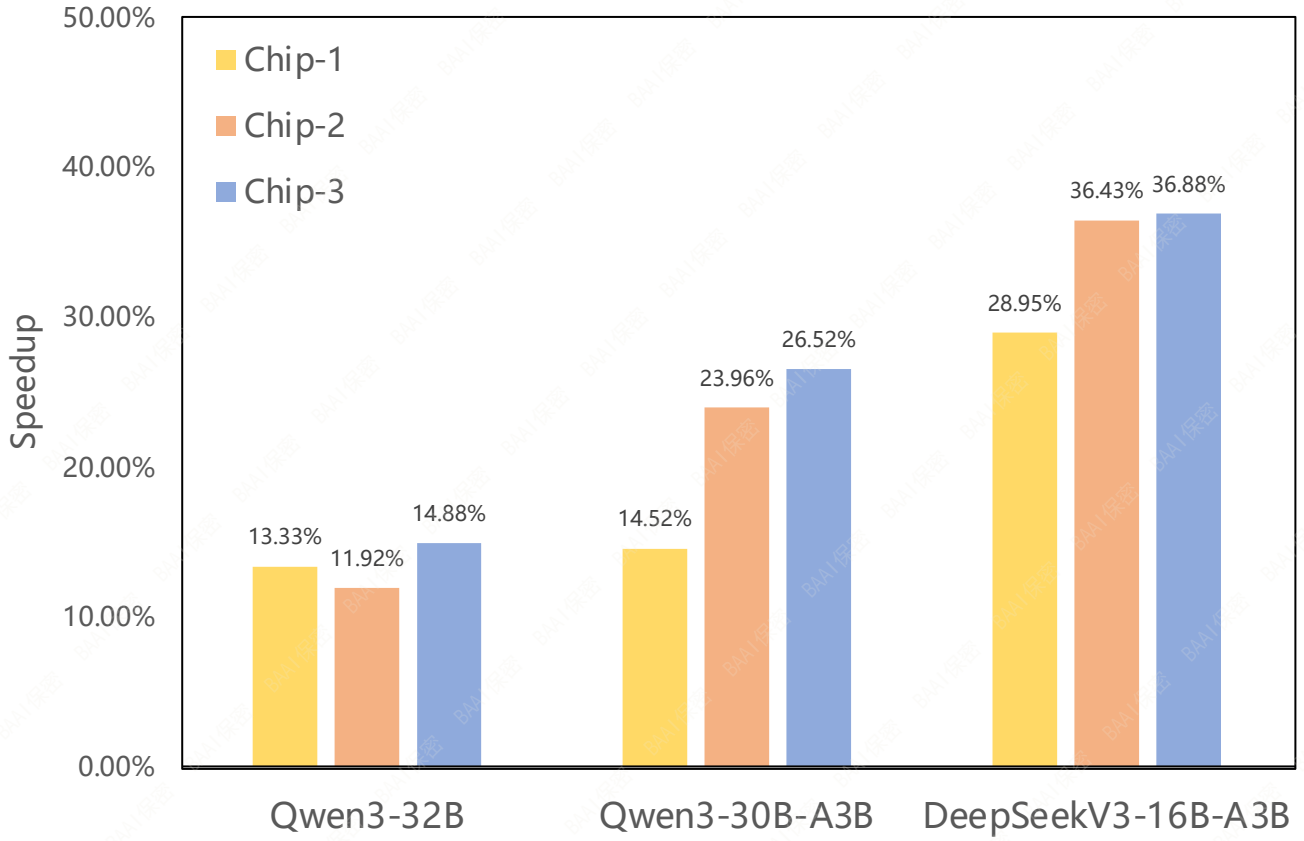
```
Theoretical memory footprints: weight and optimizer=41733.77 MB, activation=18653.03 MB, total=60386.80 MB

ocation, Total memory used after allocation: 60.7GiB (65123222165 bytes),
```

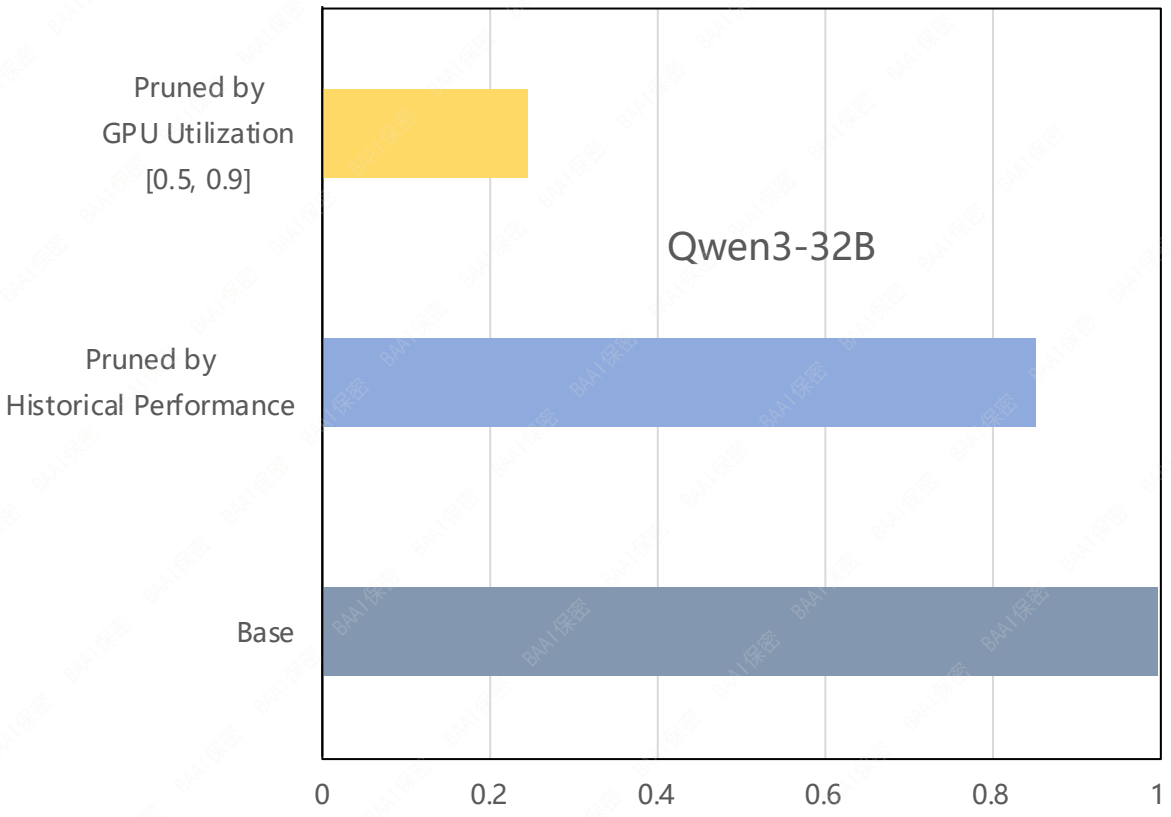
基于历史结果的剪枝

- 在线反馈
  - 记录剪枝结果和运行结果
  - 与历史结果进行对比

- 剪枝规则
  - 显存
    - 如果开启recompute都OOM，其它策略不变，recompute不开直接被剪枝
    - 如果开启Distributed Optimizer都OOM，其他策略不变，纯数据并行直接被剪枝
    - …
  - 性能
    - 如果关闭recompute都可以运行，其它策略不变，recompute开启可以直接被剪枝。
    - …

实验数据





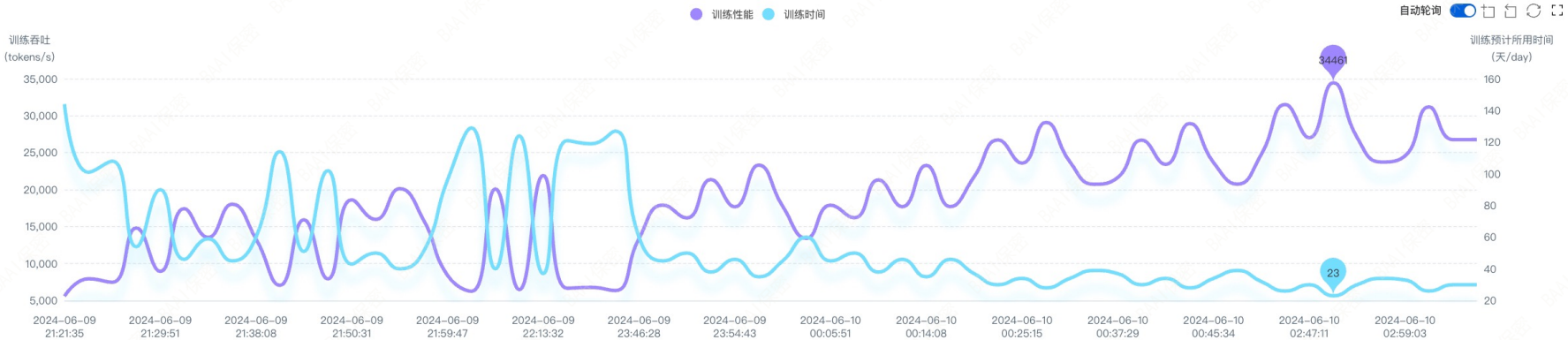
- 在不同的芯片上，不同模型和不同大小下均能获取比专家经验更优的配置。
- 最大能加速36.88%，平均能够加速23%。

- 结合历史信息、内存模型和在线反馈等多维剪枝，能够将搜索空间减少76%。

上线智源九鼎平台

模型训练 > job列表 > job详情

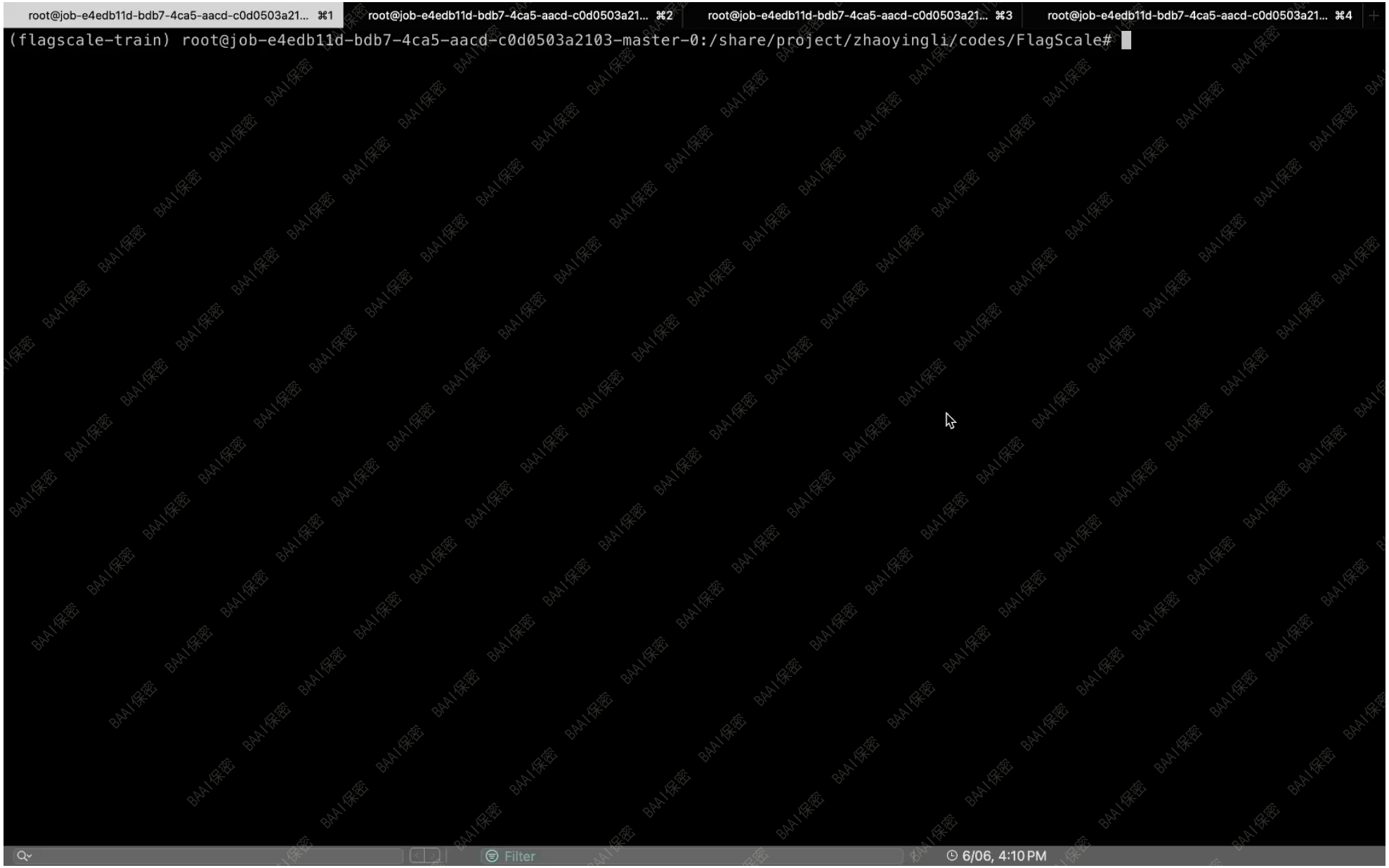
job详情　生命周期　策略寻优



时间	DP	SP	TP	PP	CP	DistOpt	MBS	ReM	ReG	ReN	训练吞吐 tokens/s
2024-06-10 03:06:36	4	false	1	4	1	false	4	uniform	full	8	26787
2024-06-10 03:02:39	4	false	1	4	1	false	2	uniform	full	8	26770
2024-06-10 03:00:45	4	false	1	4	1	false	1			0	31203
2024-06-10 02:59:03	4	false	1	4	1	false	1	uniform	full	8	24562
2024-06-10 02:54:55	4	false	1	4	1	true	8	uniform	full	8	23739
2024-06-10 02:50:47	4	false	1	4	1	true	4	uniform	full	8	27052
2024-06-10 02:48:54	4	false	1	4	1	true	2			0	34461
2024-06-10 02:47:11	4	false	1	4	1	true	2	uniform	full	8	27032
2024-06-10 02:45:18	4	false	1	4	1	true	1			0	31526
2024-06-10 02:43:35	4	false	1	4	1	true	1	uniform	full	8	24796

共 60 条　10条/页　1 2 3 4 5 6　前往 1 页

演示视频

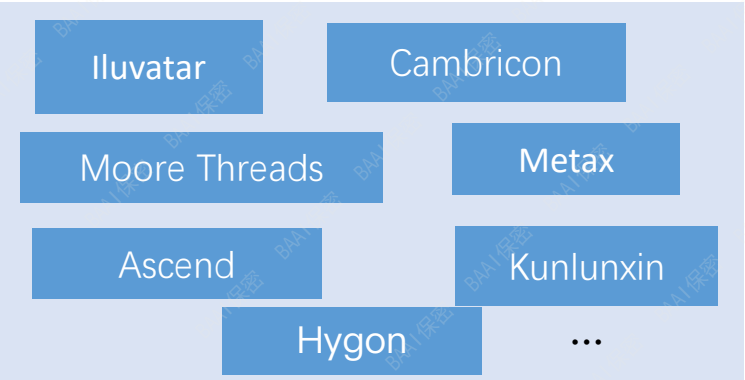

root@job-e4edb11d-bdb7-4ca5-aacd-c0d0503a21... ⌘1
root@job-e4edb11d-bdb7-4ca5-aacd-c0d0503a21... ⌘2
root@job-e4edb11d-bdb7-4ca5-aacd-c0d0503a21... ⌘3
root@job-e4edb11d-bdb7-4ca5-aacd-c0d0503a21... ⌘4
(flagscale-train) root@job-e4edb11d-bdb7-4ca5-aacd-c0d0503a2103-master-0:/share/project/zhaoyingli/codes/FlagScale#
Filter
6/06, 4:10 PM

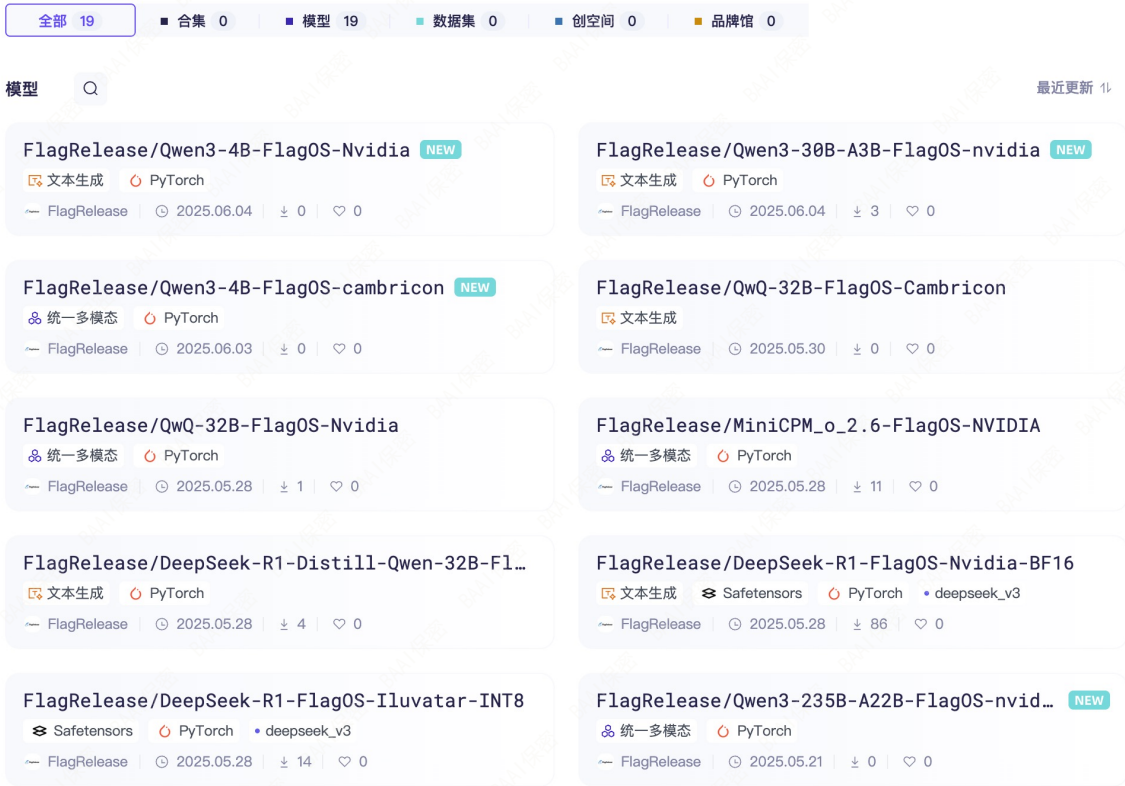
跨芯片推理自适应优化技术

多芯片一键推理部署

✗ 挑战：

- 部署芯片多，不同芯片配置不同
- 多实例部署资源如何配置，如何进行负载均衡
- 模型部署流程复杂，上手门槛高
- 不同推理后端的性能差异大，部署后端如何选择





flagscale serve <Model>

多硬件多模型多场景高效部署的挑战

✗ 挑战1：模型层出不穷、不同硬件芯片怎么推理效率最高

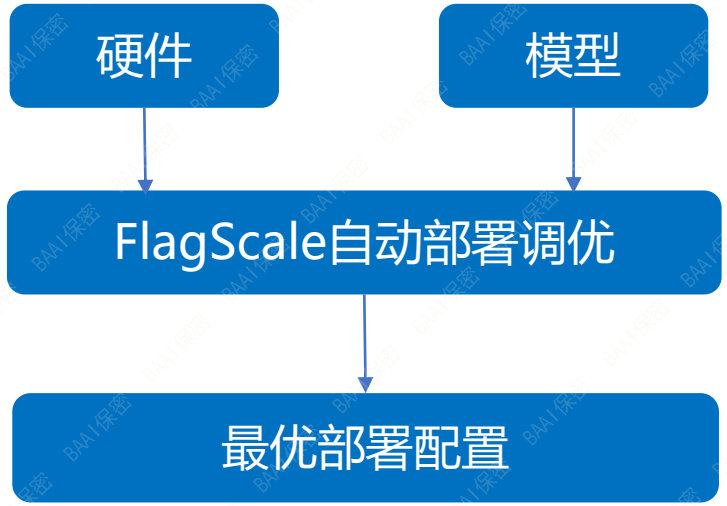
- 现有的工具配置针对英伟达芯片设置，不同芯片适用最优部署参数不同。

✗ 挑战2：推理框架多，各个框架超参多，不同模型要怎么配置最优

- 分析不同推理框架，不同超参数对推理时延和吞吐的影响，针对不同模型和芯片都自动找到最优配置方案。

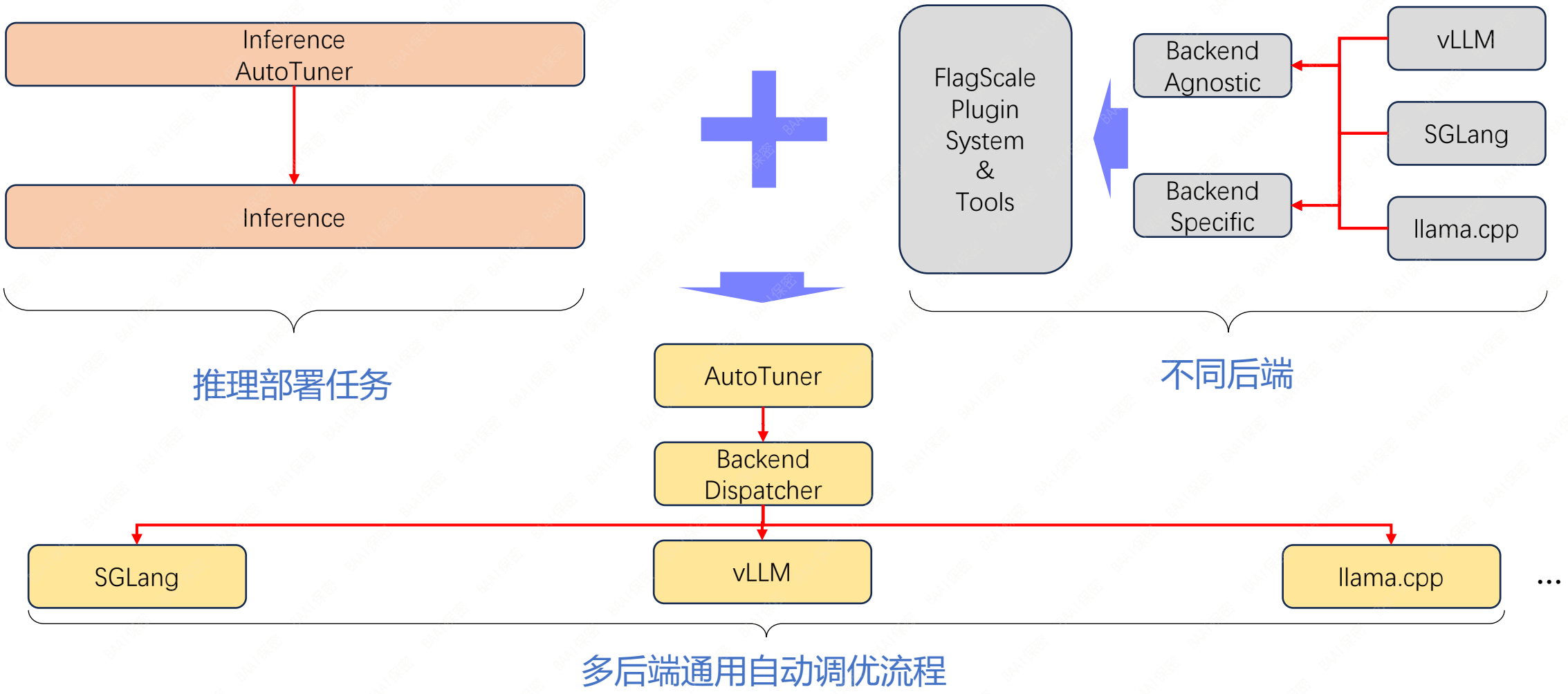
✗ 挑战3：资源数量固定，怎么分配资源达到最优部署

- 探索适合的部署实例数量，在固定资源条件下实现吞吐量的最大化，确保推理效率最优。

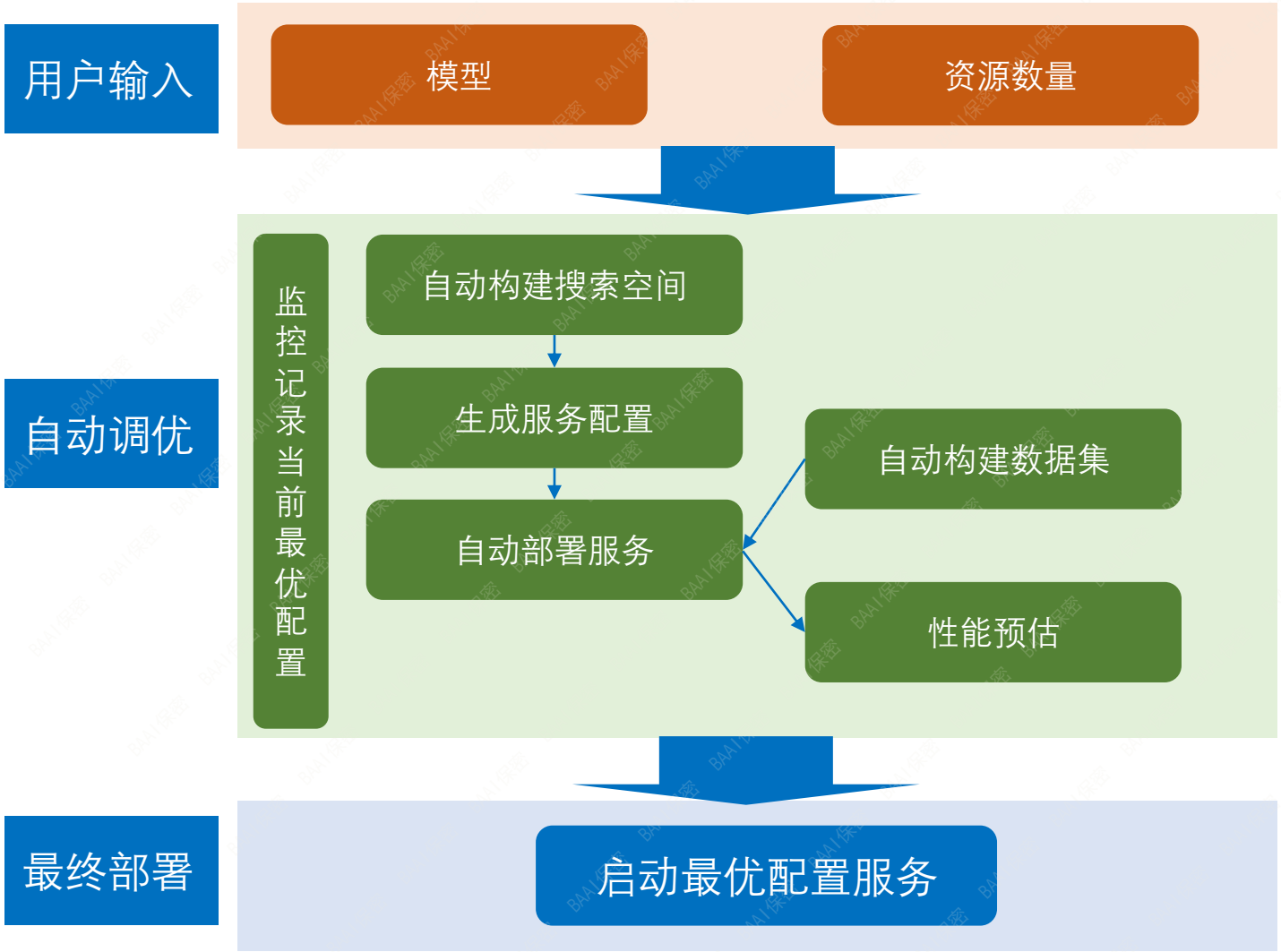


NVIDIA AMD 天数智芯 Iluvatar CoreX Cambricon 寒武纪
昆仑芯 KUNLUNXIN 摩尔线程 MOORE THREADS METAX 沐曦 HYGON Ascend

多推理后端调优功能


Inference
AutoTuner
Inference
推理部署任务
FlagScale
Plugin
System
&
Tools
Backend
Agnostic
Backend
Specific
vLLM
SGLang
llama.cpp
不同后端
AutoTuner
Backend
Dispatcher
SGLang
vLLM
llama.cpp
...
多后端通用自动调优流程

多推理后端调优功能

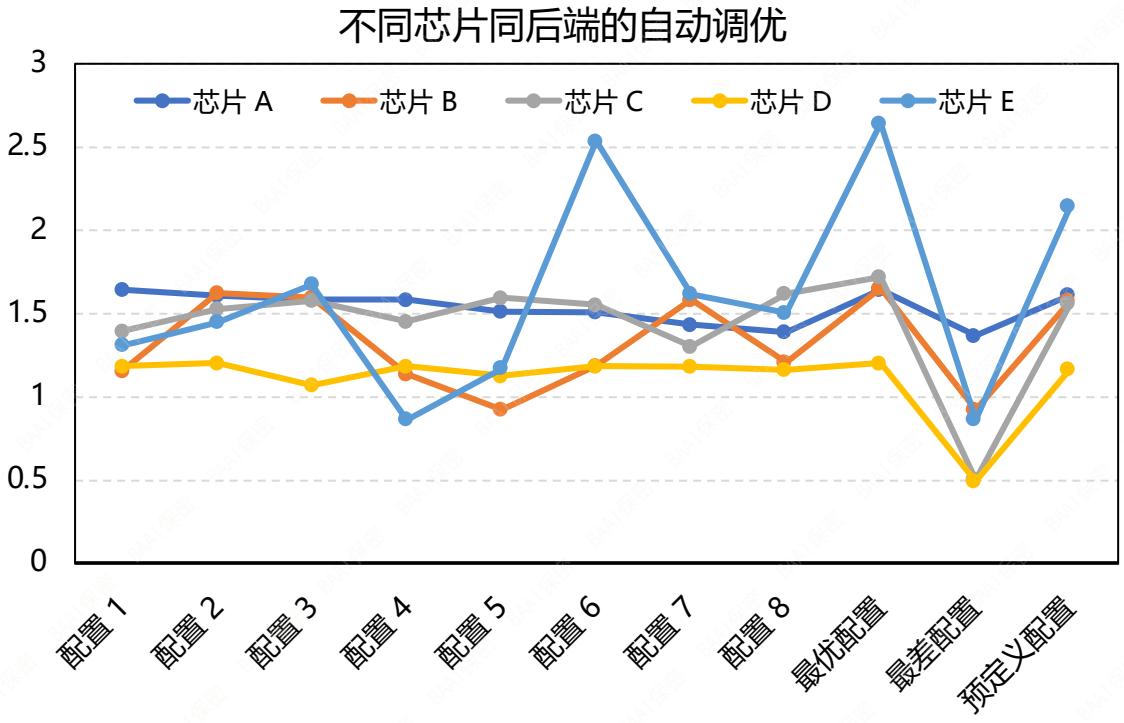


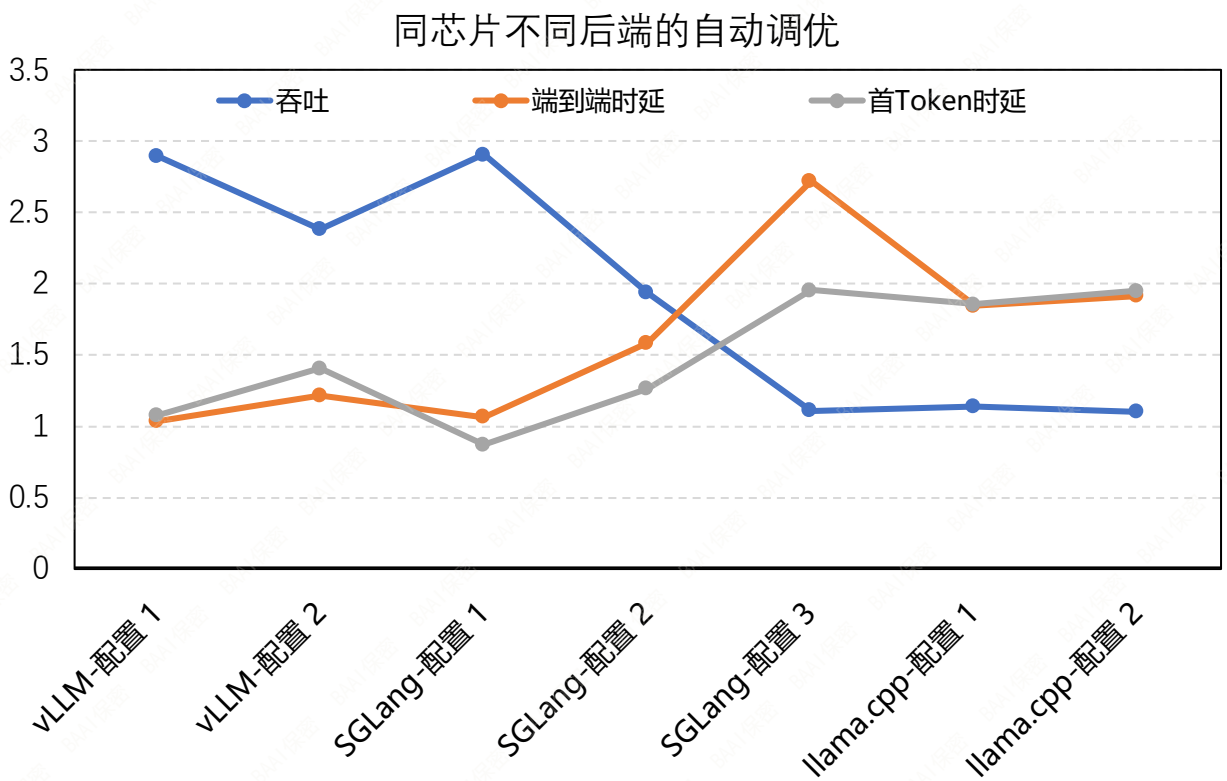
- 支持多芯片一键部署调优，无需编写代码，只需在 YAML 文件中设置 action: auto_tune 即可启动调优。
- 为不同芯片和模型的推理部署提供更加精准、高效的策略。
  - ✓ 时延调优：不同并行策略和超参的时延不同，可以找到单实例情况下时延最低的并行策略和配置。
  - ✓ 吞吐调优：不同实例数量，对吞吐的影响较大，在固定资源量的前提下，对并行策略和实例数调优，找到最优吞吐。

搜索空间

vLLM参数	SGLang参数	llama.cpp参数	说明
tensor_model_parallel_size	同vLLM	无	TP，受到总卡数的限制
pipeline_model_parallel_size	同vLLM	无	PP，受到总卡数的限制
instance	同vLLM	无	部署的实例数，受到总卡数的限制
block_size	page_size	无	KV Cache存储调度的最小粒度
max_num_batched_tokens	无	无	单次迭代中最大的token数量
max_num_seqs	max_running_requests	无	单次迭代中最大的sequence数量
无	无	n_gpu_layers	把模型的多少层卸载到GPU
无	无	parallel	sequence的并行度
无	无	threads	生成过程使用多少线程

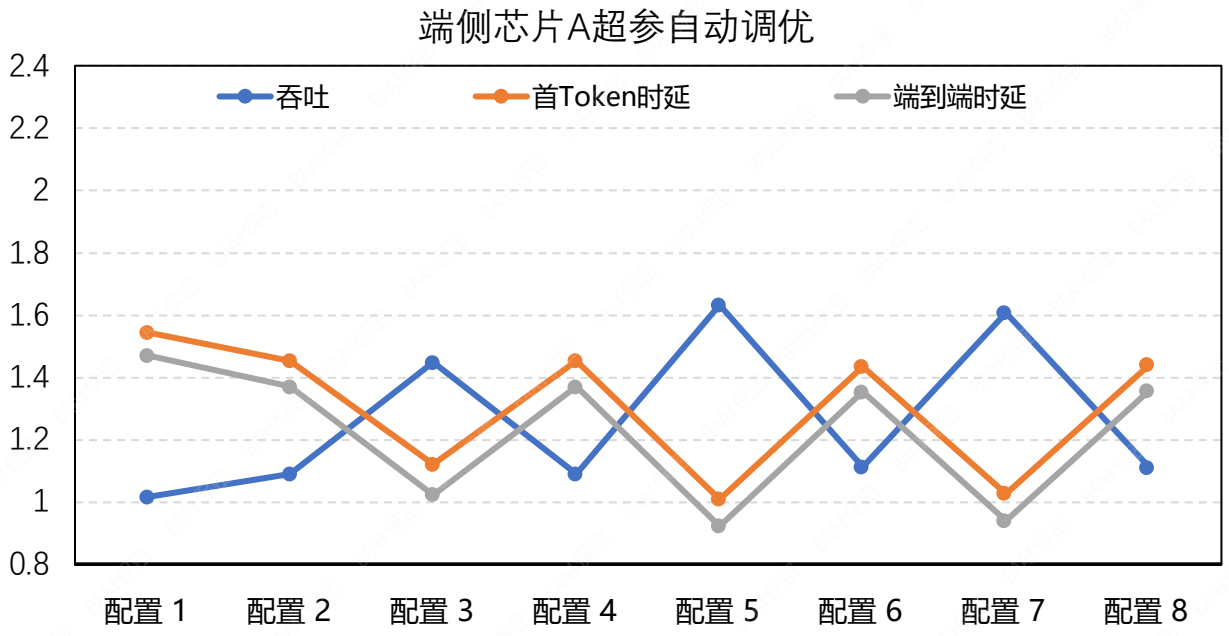
服务端和多后端推理框架调优结论

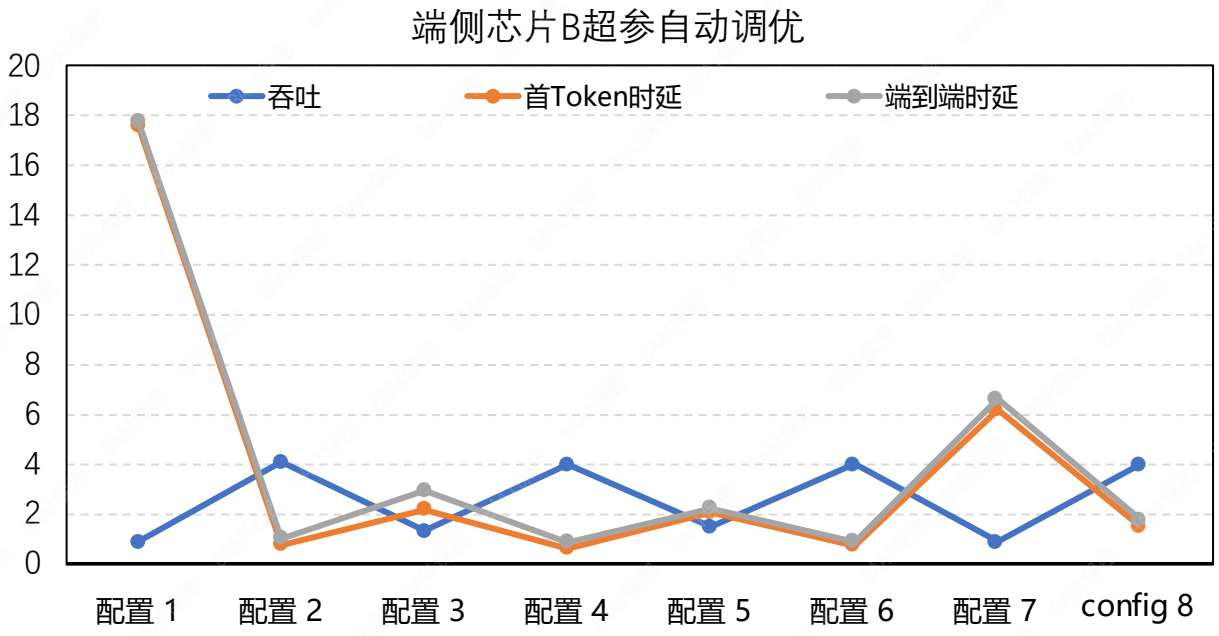




- 不同配置对吞吐的影响最高可达3倍。
- 最优超参相比框架预定义配置吞吐均有2%-20%的提升。
- 不同后端针对不同优化目标有不同的表现。

端侧调优结论

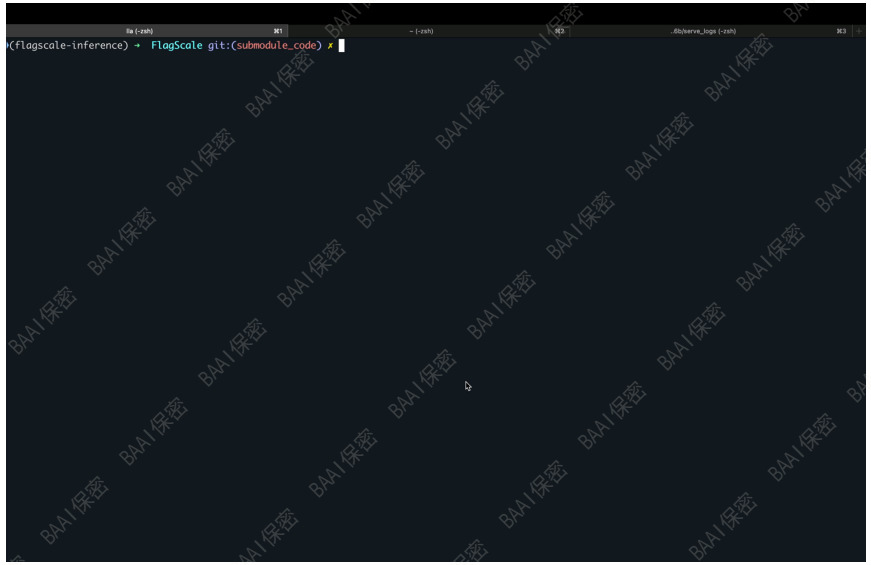


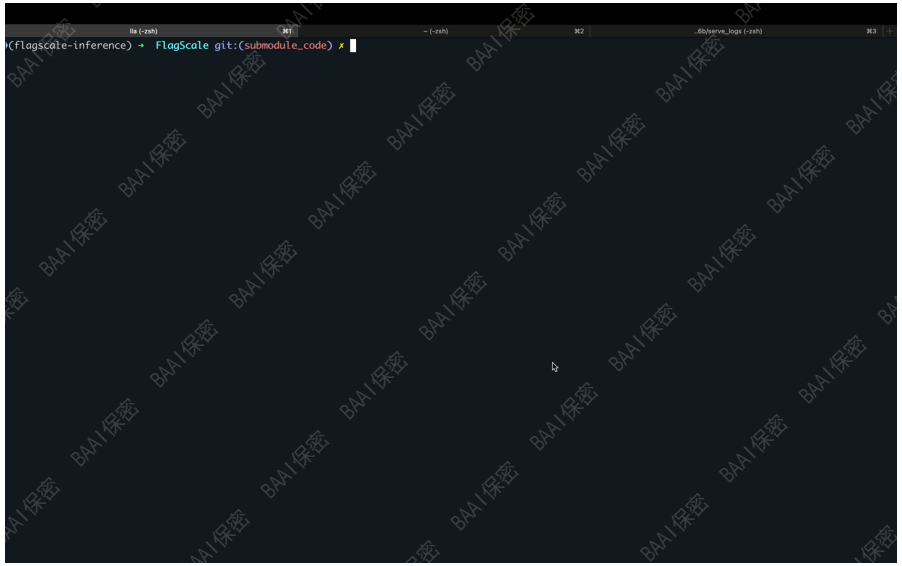


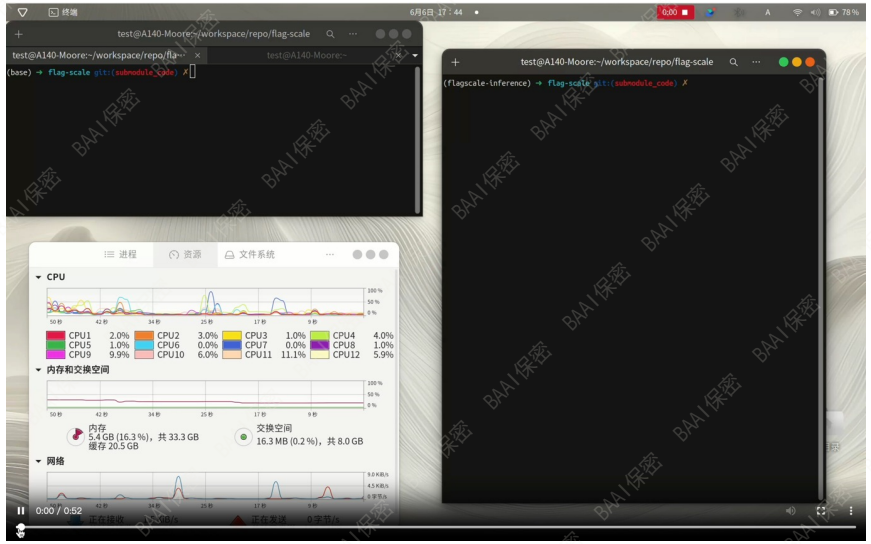
- 端侧芯片A上最优吞吐相比最差吞吐有 61% 的提升，最优端到端时延相比最差端到端时延有 38% 的下降。

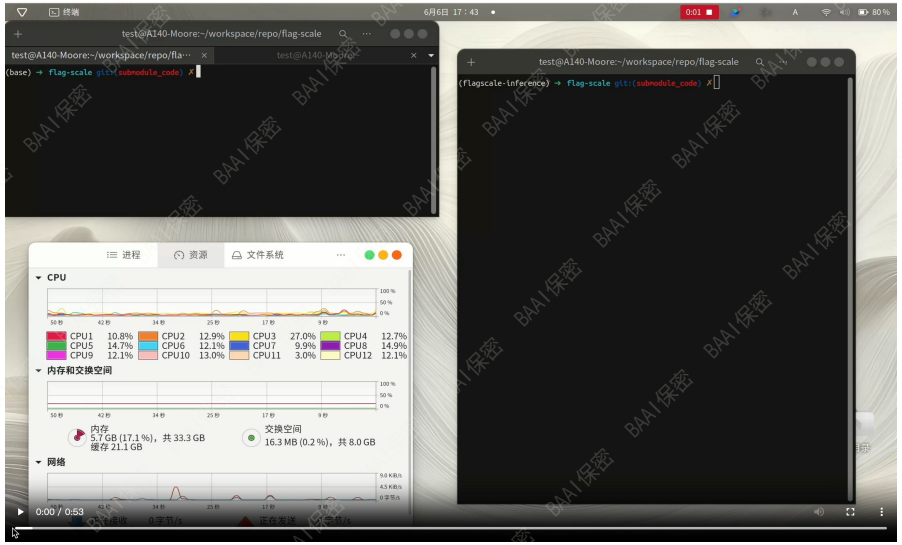
- 端侧芯片B上最优吞吐相比最差吞吐有 471% 的提升，最优端到端时延相比最差端到端时延有 95% 的下降。

视频演示


(flagscale-inference) → FlagScale git:(submodule_code) ✗


(flagscale-inference) → FlagScale git:(submodule_code) ✗


终端
test@A140-Moore:~/workspace/repo/flag-scale
(base) → flag-scale git:(submodule_code) ✗
(flagscale-inference) → flag-scale git:(submodule_code) ✗
进程
资源
文件系统
CPU
内存和交换空间
网络
0:00 / 0:52


终端
test@A140-Moore:~/workspace/repo/flag-scale
(base) → flag-scale git:(submodule_code) ✗
(flagscale-inference) → flag-scale git:(submodule_code) ✗
进程
资源
文件系统
CPU
内存和交换空间
网络
0:00 / 0:53

推理自动调优示例

- 下载测试镜像

docker pull flagrelease-registry.cn-beijing.cr.aliyuncs.com/flagrelease/flagrelease:flagrelease_metax_qwen3

- 启动测试容器

docker run -it --device=/dev/dri --device=/dev/mxcd --group-add video --name test_serve_autotune --device=/dev/mem --network=host --security-opt seccomp=unconfined --security-opt apparmor=unconfined --shm-size '100gb' --ulimit memlock=-1 -v /usr/local/:/usr/local/ -v /nfs:/nfs flagrelease-registry.cn-beijing.cr.aliyuncs.com/flagrelease/flagrelease:flagrelease_metax_qwen3 /bin/bash

- 安装modelscope并下载模型

pip install modelscope
modelscope download --model Qwen/Qwen3-0.6B --local_dir /nfs/Qwen3-0.6B/

- 下载FlagScale

git clone https://github.com/FlagOpen/FlagScale.git

推理自动调优示例

- 配置搜索空间

examples/qwen3/conf/serve_atmb.yaml

```yaml
defaults:
  - _self_
  - serve: 0_6b

experiment:
  exp_name: qwen3_0.6b
  exp_dir: outputs/${experiment.exp_name}
  task:
    type: serve
  deploy:
    use_fs_serve: false
  runner:
    hostfile: null
    nproc_per_node: 1
    nnodes: 1
    host: 0.0.0.0
  auto_tuner:
    engines: vllm
    space:
      vllm:
        tensor_model_parallel_size: [1]
        block_size: [32, 64]
        max_num_batched_tokens: [512]
        max_num_seqs: [128]
    cards: 1
    control:
      interval: 10
      run_best: False
  envs:
    GEMS_VENDOR: metax
action: auto_tune

hydra:
  run:
    dir: ${experiment.exp_dir}/hydra
```

examples/qwen3/conf/serve/0_6b.yaml

```yaml
- serve_id: vllm_model
  engine: vllm
  engine_args:
    model: /nfs/Qwen3-0.6B/
    max_model_len: 4096
    max_num_seqs: 4
  engine_args_specific:
    vllm:
      tensor_parallel_size: 1
      pipeline_parallel_size: 1
      gpu_memory_utilization: 0.9
      enforce_eager: true
      trust_remote_code: true
      enable_chunked_prefill: true
  profile:
    prefix_len: 0
    input_len: 1024
    output_len: 128
    num_prompts: 128
    range_ratio: 1
```

- 执行

python run.py --config-path ./examples/qwen3/conf --config-name serve_atmb action=auto_tune

谢谢聆听，
欢迎合作与建议。

https://github.com/FlagOpen/FlagScale