性能测试与分析文档

Valor-go Team

小组知乎文章：https://zhuanlan.zhihu.com/p/1923813736247964768

1. LLM部署相关的性能指标列表

1. 首Token返回延迟(Time to First Token)
    1. 定义：从用户发送请求到收到第一个输出Token的时间
    2. 合理性：直接影响用户体验的“响应感知”速度，对于交互场景十分重要
2. 输出吞吐量（输出速度, Tokens per Second）
    1. 定义：模型每秒生成的Token数量（排除首Token延迟时间）
    2. 合理性：决定长文本生成的流畅度。低吞吐量会导致长回答“卡顿式输出”，影响用户体验
3. 并发处理能力(Concurrent Requests Handled)
    1. 定义：系统在保证SLA（Service Level Agreement，如延迟<2秒）下能同时处理的请求数
    2. 合理性：反映系统的实际服务容量，直接影响部署成本和业务扩展性（如应对流量高峰）
4. 显存占用(GPU Memory Use)
    1. 定义：推理时GPU显存占用
    2. 合理性：过高的显存占用会导致OOM错误，限制并发能力；过低则可能意味着资源浪费，需优化模型分片或量化策略
5. 错误率(Error Rate)
    1. 定义：请求失败比例（如超时、崩溃、返回非法内容）
    2. 合理性：直接影响服务可靠性；需区分暂时性错误（可重试）和系统性错误（如显存泄漏）
6. 每请求成本(Cost per Request)
    1. 定义：单次请求消耗的计算资源成本（如GPU秒数）
    2. 合理性：直接关联商业可行性，尤其在面向C端的高频调用场景（如AI写作助手）

2. 设计测试任务与性能指标选择

2.1 性能指标选择

1. 输出吞吐量(Tokens per Second)：可直接通过llama-bench的输出查看
2. 显存占用(Memory Use): 运行llama-bench时，在另一个终端使用指令`nvidia-smi --query-gpu=timestamp,utilization.gpu,utilization.memory,memory.total,memory.used --format=csv -l 1 > gpu_log.csv`，其中的`memory.used`便是显存占用

2.2 测试任务

使用llama-bench进行测试(llama-bench的详细使用请参考：https://github.com/ggml-org/llama.cpp/tree/master/tools/llama-bench)

```
usage: llama-bench [options]

options:
  -h, --help
```

```
  --numa <distribute|isolate|numactl>       numa mode (default: disabled)
  -r, --repetitions <n>                     number of times to repeat each
test (default: 5)
  --prio <0|1|2|3>                          process/thread priority
(default: 0)
  --delay <0...N> (seconds)                 delay between each test
(default: 0)
  -o, --output <csv|json|jsonl|md|sql>      output format printed to
stdout (default: md)
  -oe, --output-err <csv|json|jsonl|md|sql> output format printed to
stderr (default: none)
  -v, --verbose                             verbose output
  --progress                                print test progress indicators

test parameters:
  -m, --model <filename>                    (default: models/7B/ggml-
model-q4_0.gguf)
  -p, --n-prompt <n>                        (default: 512)
  -n, --n-gen <n>                           (default: 128)
  -pg <pp,tg>                               (default: )
  -d, --n-depth <n>                         (default: 0)
  -b, --batch-size <n>                      (default: 2048)
  -ub, --ubatch-size <n>                    (default: 512)
  -ctk, --cache-type-k <t>                  (default: f16)
  -ctv, --cache-type-v <t>                  (default: f16)
  -dt, --defrag-thold <f>                   (default: -1)
  -t, --threads <n>                         (default: system dependent)
  -C, --cpu-mask <hex,hex>                  (default: 0x0)
  --cpu-strict <0|1>                        (default: 0)
  --poll <0...100>                          (default: 50)
  -ngl, --n-gpu-layers <n>                  (default: 99)
  -rpc, --rpc <rpc_servers>                 (default: none)
  -sm, --split-mode <none|layer|row>        (default: layer)
  -mg, --main-gpu <i>                       (default: 0)
  -nkvo, --no-kv-offload <0|1>              (default: 0)
  -fa, --flash-attn <0|1>                   (default: 0)
  -mmp, --mmap <0|1>                        (default: 1)
  -embd, --embeddings <0|1>                 (default: 0)
  -ts, --tensor-split <ts0/ts1/..>          (default: 0)
  -ot --override-tensors <tensor name pattern>=<buffer type>;...
                                            (default: disabled)
  -nopo, --no-op-offload <0|1>              (default: 0)

Multiple values can be given for each parameter by separating them with
','
or by specifying the parameter multiple times. Ranges can be given as
'first-last' or 'first-last+step' or 'first-last*mult'.
```

1. 吞吐能力测试:
    1. 不同batch_size下的吞吐能力测试
        1. 指令`./llama-bench -n 0 -p 1024 -b 128,256,512,1024 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf`;

    2. 其中的`-b 128,256,512,1024`用于测试不同batch_size的性能效果；
    3. `-n 0 -p 1024`指不生成token, 提示词长度为1024，这是常见的前向性能基准测试方法，常用于模型部署时评估吞吐能力
2. 不同threads_num下的吞吐能力测试
    1. `./llama-bench -n 0 -n 16 -p 64 -t 1,2,4,8,16,32 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf`
    2. `-t 1,2,4,8,16`即指定不同的线程数量
    3. 使用 prompt 长度 64（-p 64）
    4. 不生成 tokens 的测试：-n 0 → pp64
    5. 生成 16 tokens 的测试：-n 16 → tg16
3. 不同GPU层数下的吞吐能力测试
    1. `./llama-bench -ngl 10,20,30,31,32,33,34,35,99 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf`
    2. `-ngl`指`n-gpu-layers`，即放置在GPU上的模型层数
4. 不同预填充上下文下的吞吐能力测试
    1. `./llama-bench -d 0,512 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf`
    2. 这里的`-d`就是指定预填充上下文的长度

2. 显存占用测试： 只需在使用llama-bench测试时，在另一个终端使用指令`nvidia-smi --query-gpu=timestamp,utilization.gpu,utilization.memory,memory.total,memory.used --format=csv -l 1 > gpu_log.csv`，其中的`memory.used`便是显存占用 为了方便，我编写了一个脚本`run_benchmark.sh`在使用llama-bench时同时触发上述指令 收集数据时仅选择最大值（峰值）
3. 测试任务说明：

- `pp(num)` 指prompt processing, 也就是提示词长度 = num,用于测评纯前向推理速度，比如总结/编码阶段的瓶颈。
- `tg(num)` 指text generation, 也就是生成token 数 = num，用于测评生成速度，反映整体响应时间中的实际 token 生成效率

3. 测试数据、分析以及优化

- 选择最佳参数的原则：
    - `Tokens per Second`越大越好，这样吞吐更快
    - `GPU Memory Use`适中比较好

3.1.1 不同batch-size下的测试

`./llama-bench -n 0 -p 1024 -b 128,256,512,1024 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf`

model	size	params	backend	ngl	n_batch	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	128	pp1024	2457.54 ± 79.48	1374

model	size	params	backend	ngl	n_batch	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	256	pp1024	2419.84 ± 30.16	1454
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	512	pp1024	2375.89 ± 16.96	1614
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	1024	pp1024	2408.87 ± 10.85	1614

观察以上数据可知：随着`batch_size`增加，`Tokens per Second`标准差逐渐稳定，`GPU Memory Use`逐渐升高直至最大`1614`， 因此我们对`batch_size`的最优化选择是128，此时`Tokens per Second`最大，且`GPU Memory Use`最小

3.1.2 不同threads_num下的测试

```
./llama-bench -n 0 -n 16 -p 64 -t 1,2,4,8,16,32 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf
```

model	size	params	backend	ngl	threads	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	1	pp64	2586.35 ± 1034.14	1224
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	1	tg16	34.37 ± 3.76	1208
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	2	pp64	2167.59 ± 42.58	1228
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	2	tg16	33.36 ± 0.34	1228
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	4	pp64	2164.58 ± 78.53	1228
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	4	tg16	33.73 ± 2.35	1224
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	8	pp64	2146.30 ± 55.10	1226
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	8	tg16	33.07 ± 1.15	1224
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	16	pp64	2084.91 ± 50.26	1228

model	size	params	backend	ngl	threads	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	16	tg16	32.79 ± 1.14	1228
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	32	pp64	2090.97 ± 47.05	1228
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	32	tg16	32.79 ± 1.37	1228

观察数据可知：

- 在pp64测试中，随着线程数增加，`Tokens per Second`均值逐渐减小、方差起伏不定，`GPU Memory Use`保持稳定
- 在tg16测试中，和pp64测试类似
- 综上，线程数的最优化选择是1

3.1.3 不同GPU层数下的测试

```
./llama-bench -ngl 10,20,30,31,32,33,34,35,99 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf
```

model	size	params	backend	ngl	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	10	pp512	2439.33 ± 98.51	998
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	10	tg128	44.79 ± 2.06	998
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	20	pp512	2840.46 ± 272.70	1304
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	20	tg128	56.58 ± 4.67	1304
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	30	pp512	2509.46 ± 378.98	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	30	tg128	33.80 ± 0.53	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	31	pp512	2641.39 ± 136.02	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	31	tg128	33.09 ± 0.26	1550

model	size	params	backend	ngl	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	32	pp512	2773.88 ± 174.51	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	32	tg128	32.74 ± 0.05	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	33	pp512	2580.63 ± 142.67	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	33	tg128	32.59 ± 0.15	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	34	pp512	2764.08 ± 129.42	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	34	tg128	32.84 ± 0.09	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	35	pp512	2571.51 ± 127.89	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	35	tg128	32.61 ± 0.04	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	pp512	2340.46 ± 479.29	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	tg128	32.39 ± 0.09	1550

- 在pp512测试中：随着`ngl`增加，`Tokens per Second`在`ngl=20`时表现最佳，且`GPU Memory Use`在`ngl=30`及以后达到最大值1550
- 在tg128测试中，`Tokens per Second`中表现最佳的也是`ngl=20`，且它的`GPU Memory Use`并没有达到最大
- 综合考虑我们的最优化选择为`ngl=20`

3.1.4 不同预填充上下文下的测试

```
./llama-bench -d 0,512 -m ./models/ggml-org_Qwen3-1.7B-GGUF_Qwen3-1.7B-Q4_K_M.gguf
```

model	size	params	backend	ngl	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	pp512	2707.51 ± 196.63	1550
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	tg128	33.38 ± 0.33	1550

model	size	params	backend	ngl	test	t/s	GPU Memory Use(MiB)
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	pp512 @ d512	2173.30 ± 21.61	1614
qwen3 1.7B Q4_K - Medium	1.19 GiB	2.03 B	CUDA	99	tg128 @ d512	31.40 ± 0.55	1614

- 无论是pp512还是tg128，`d=0`时`Tokens per Second`均大于`d=512`时，且`d=0`时可以节省`GPU Memory`
- 所以最优化选择是`d=0`

4. 关键优化操作分析

4.1 影响最大的三项优化操作

根据测试数据，按照对性能指标和LLM输出质量的**影响程度从大到小**排序：

4.1.1 GPU层数优化(ngl)

数据：

- `ngl=10`: pp512测试为2439.33 t/s，GPU内存998MiB
- `ngl=20`: pp512测试为2840.46 t/s，GPU内存1304MiB
- `ngl=99`: pp512测试为2340.46 t/s，GPU内存1550MiB

影响分析：

- **性能影响**：从ngl=10到ngl=20，吞吐量提升16.4%（2439→2840 t/s），这是所有测试中最显著的性能提升
- **资源效率**：ngl=20相比ngl=99节省246MiB GPU内存，同时保持更优性能
- **输出质量影响**：GPU层数直接影响模型推理精度。过少的GPU层会导致部分计算在CPU上进行，可能影响数值精度和模型输出的一致性

原因解释：

- ngl=20时达到了GPU计算能力和内存使用的最佳平衡点
- 过高的ngl值会导致GPU内存碎片化和调度开销增加
- 适中的GPU层数确保了关键的注意力计算和前馈网络在GPU上高效执行

4.1.2 批处理大小优化(batch_size)

数据：

- `batch_size=128`: 2457.54 t/s，GPU内存1374MiB
- `batch_size=256`: 2419.84 t/s，GPU内存1454MiB
- `batch_size=1024`: 2408.87 t/s，GPU内存1614MiB

影响分析：

- **性能影响**：batch_size=128相比1024提升约2%的吞吐量，同时节省240MiB内存
- **并发能力**：较小的batch_size为并发请求预留更多GPU内存空间
- **输出质量影响**：batch_size主要影响推理效率，对单个请求的输出质量影响较小

原因解释：

- 较小的batch_size减少了GPU内存碎片，提高了内存访问效率
- 避免了大batch导致的缓存失效和内存带宽瓶颈

4.1.3 线程数优化(threads)

数据：

- `threads=1`: pp64测试为2586.35 t/s
- `threads=2`: pp64测试为2167.59 t/s
- `threads=32`: pp64测试为2090.97 t/s

影响分析：

- **性能影响**：单线程相比多线程有显著优势，threads=1比threads=32提升约24%
- **资源利用**：多线程在GPU加速场景下反而引入了额外开销
- **输出质量影响**：线程数不直接影响模型输出质量，主要影响推理速度

原因解释：

- GPU计算为主的场景下，CPU多线程并行的收益有限
- 线程间同步和调度开销超过了并行带来的收益

4.2 最优参数组合

基于以上分析，推荐的最优参数组合：

- **ngl=20** (GPU层数)
- **batch_size=128** (批处理大小)
- **threads=1** (线程数)
- **d=0** (预填充上下文)

预期效果：

- 最大化吞吐量性能
- 优化GPU内存使用效率
- 为并发请求预留资源空间
- 保持模型输出质量稳定性