Dump logits k fuse kernel

Bitonic Sort、**Radix Select** 和 **Qrita** 三种算法在 Top-K 场景下：

1. Bitonic Sort（双调排序）

核心思想

双调排序是一种**并行排序网络**。它通过一系列预定义的“比较-交换”操作，将无序序列转换为双调序列（先增后减或先减后增），再通过反复合并得到完全有序序列。用于 Top-K 时，通常会对整个数组排序后取前 K 个。

2. Radix Select（基数选择）

核心思想

不排序，只筛选。利用基数排序的按位处理思路，从最高位到最低位逐位统计候选元素的桶分布，从而确定第 K 大的元素的数值（阈值），然后一次性收集所有大于等于阈值的元素。

3. Qrita

核心思想

“统计估算 + 自适应主元搜索”。

1. **高斯 σ-截断**：利用 logits 近似高斯分布，用 $\mu + \alpha \cdot \sigma$ 快速过滤掉明显不可能是 Top-K 的低分值，大幅缩小候选集。
2. **四元主元搜索**：在缩小后的候选集上，通过一次检查两个主元的二分搜索策略，快速定位精确的第 K 大阈值。

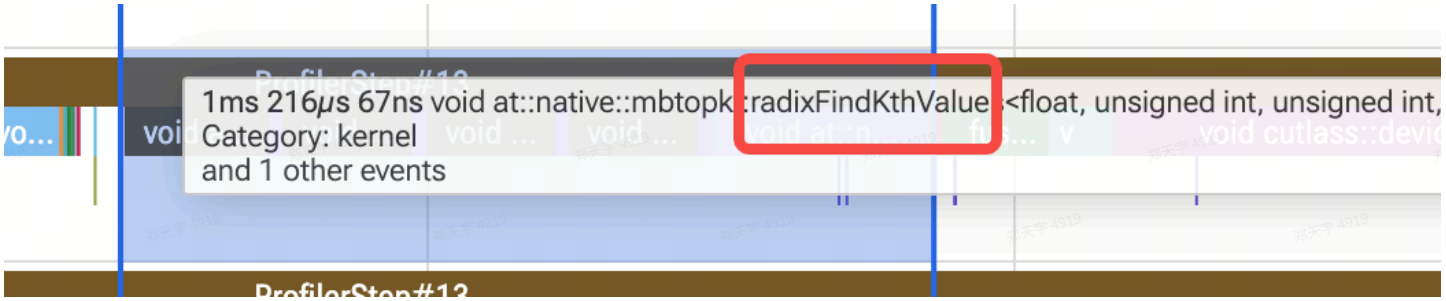
论文链接：https://arxiv.org/abs/2602.01518

仓库链接：https://github.com/vllm-project/vllm/blob/main/vllm/v1/sample/ops/topk_topp_triton.py

算法	核心思想	复杂度	是否需要全局排序	对 K 敏感度
Bitonic Sort	并行排序网络	O(N $\log^2$ N)	是	低（必须排序）
Radix Select	按位淘汰	O(N·B)，B≈32	否	不敏感
Qrita	统计截断 + 双主元二分	O(N) + O(log N) 迭代	否	不敏感

当前实现：

当前使用torch.topk,可以看到底层选用的是radix方法。



单机实验debug效果：

1. Redix select

当前的histogram不内部被triton_distribute支持（编译失败），导致只能自己写分桶计数的逻辑，导致耗时加重

All fuse：105.638

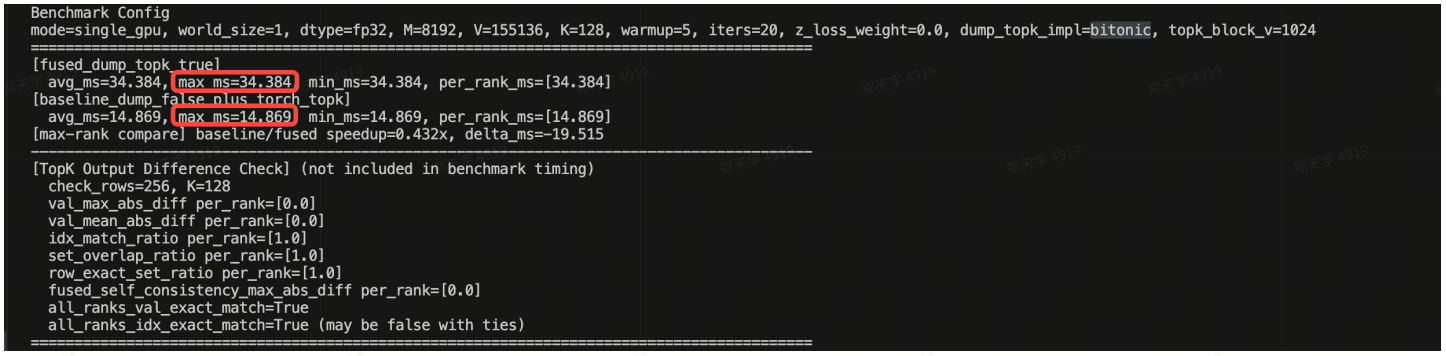
Fuse loss + top-k: 14.867

```
================================================================================
Benchmark Config
mode=single_gpu, world_size=1, dtype=fp32, M=8192, V=155136, K=128, warmup=5, iters=20, z_loss_weight=0.0, dump_topk_impl=radix
================================================================================
[fused_dump_topk_true]
  avg_ms=105.630, max_ms=105.630, min_ms=105.630, per_rank_ms=[105.63]
[baseline_dump_false_plus_torch_topk]
  avg_ms=14.867, max_ms=14.867, min_ms=14.867, per_rank_ms=[14.867]
[max-rank compare] baseline/fused speedup=0.141x, delta_ms=-90.764
--------------------------------------------------------------------------------
[TopK Output Difference Check] (not included in benchmark timing)
  check_rows=256, K=128
  val_max_abs_diff per_rank=[0.0]
  val_mean_abs_diff per_rank=[0.0]
  idx_match_ratio per_rank=[1.0]
  set_overlap_ratio per_rank=[1.0]
  row_exact_set_ratio per_rank=[1.0]
  fused_self_consistency_max_abs_diff per_rank=[0.0]
  all_ranks_val_exact_match=True
  all_ranks_idx_exact_match=True (may be false with ties)
```

2. Bitonic sort

All fuse：34.484

Fuse loss + top-k: 14.869



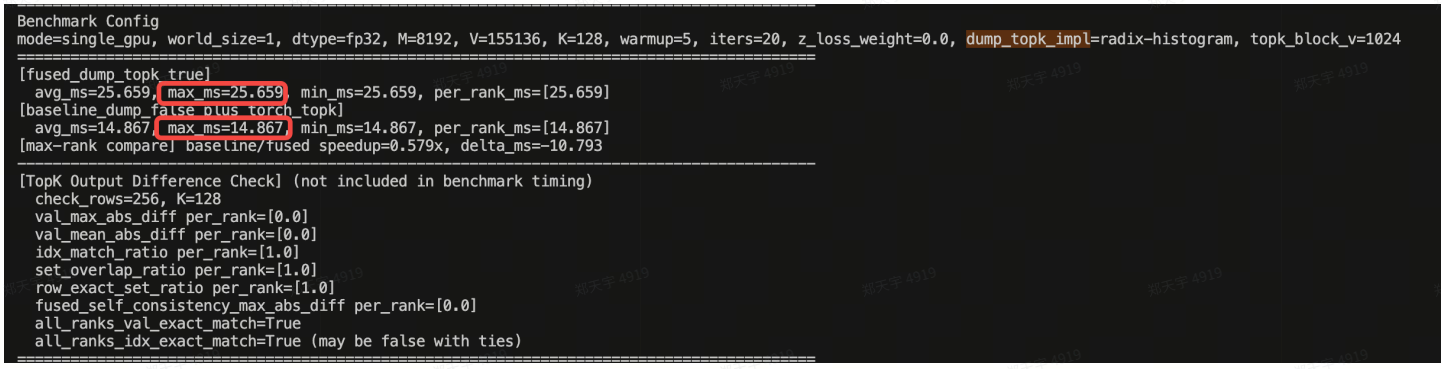
耗时稍微好一点，但是由于当vocabsize较大时，Bitonic sort效果会明显弱于redix select，因此时间还是很长

3. Radix-*histogram*

看了一下github，当前triton3系列明显是支持tr.*histogram的，因此直接用python pip装原装的triton==3.5尝试实现一版tr.histogram的，发现可以正常编译。尝试用histogram实现。*

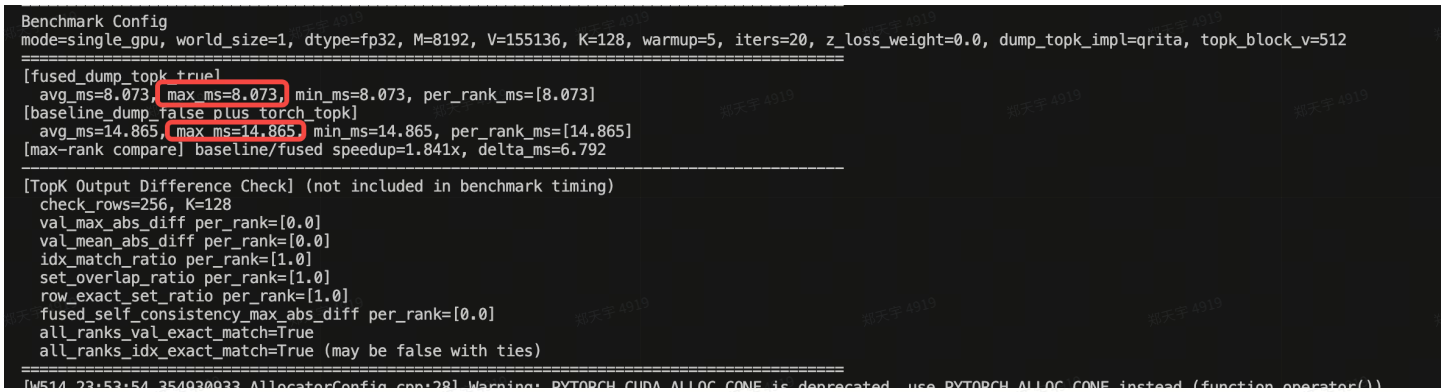
All fuse：25.659

Fuse loss + top-k: 14.867



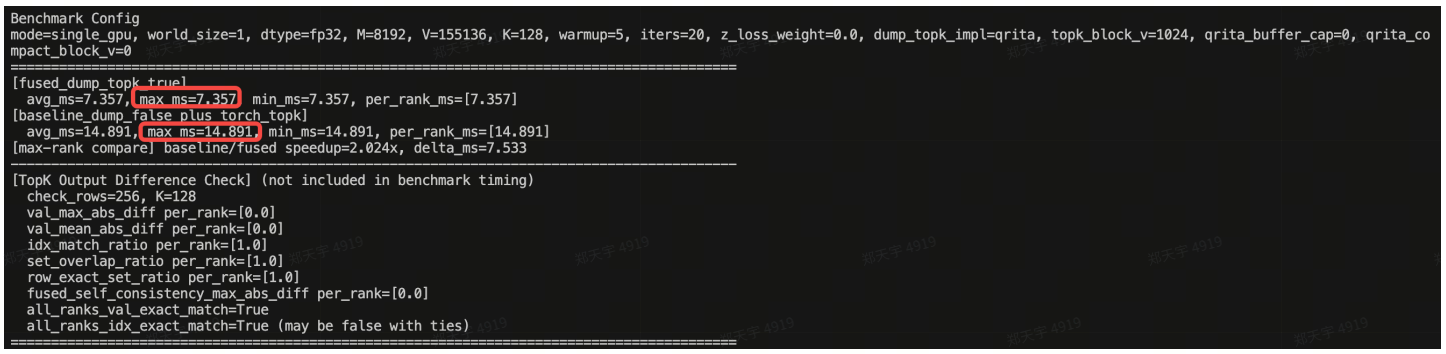
但仍然相差10ms（感觉这里可以用cuda kernel写，triton实现radix还是太不友好了）

4. Qrita



效果显著，缩了一倍的时间。同时单机单卡logtis是对齐的。

单机4卡测试：



也是对齐的，而且时间结论一样。

当前跟fuse CE 合并的好处：

1. fused CE 主循环读 logits 一遍

顺便算：

- sum_logits
- sum_sq_logits
- row_min / row_max

其中，qrita需要：

mean = sum / V

std = sqrt(sum_sq / V - mean^2)

row_min / row_max

这部分就可以和原来celoss那一遍循环一起算，提高效率。

2. 第二遍遍历大的logits过滤
3. 遍历小的buffer做search/gather(双 pivot 的三分查找)

- Feedback

多机实验效果：