论文1

Efficient Memory Management for Large Language Model Serving with PagedAttention

（SOSP’23）

作用：写作思路值得学习；对大模型推理过程中制约的因素有了新的思考，以前大家认为计算是制约，实际上由于英伟达显卡显存就是比较小，所以在桌面计算机上为了跑起来大模型，对内存管理的优化就显得尤为重要。

研究问题：大模型在推理过程中存在严重的内存碎片化和过度分配的问题，拉低了服务的吞吐量，文章主要是优化大模型推理过程中的内存管理，以在多批量输入的情况下提高服务的吞吐率。

研究层次：大模型部署过程中的显存浪费问题，只考虑显存管理问题。

研究重要性：被工业界（OpenAI等）广泛使用

研究差异：不是量化，对精度没有一点损失

挑战：

（1）目前在推理过程中广泛使用了KVCache的机制来加快推理速度（llama2.c也用了），但是对于每一个prompt，都会产生大量的KVCache，甚至占到了全部内存占用的30%以上，但是这部分的内存是动态迅速变化的，如何采取好的管理策略是一个挑战。

解决方案：

（1）借鉴操作系统内存管理的分页机制，达到几乎零冗余的KVCache；（2）在同一请求或跨请求共享KVCache；（3）开源了高效transformer推理框架vLLM。

论文2

PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU

（OSDI’24在投；论文2实际上就是论文4的翻版）

作用：对剪枝有一点启示，由于LLM通常会经过激活层，经过之后某些参数的作用就大大减弱，这部分参数可以被认为是不重要的。

研究问题：大模型在推理过程中将GPU的工作卸载到CPU上运行。

研究层次：大模型部署过程中工作负载卸载到CPU上运行，系统的工作，关心在桌面级处理器和GPU上运行超大模型，是不含显示剪枝的剪枝方法。

研究重要性：桌面级计算机运行LLM的推理；为剪枝提供新的依据（尽管没有数学上的依据）。

研究差异：纯炼丹，没有一点数学依据；但是工程量大、代码量大，做到开箱即用，已经做成了很完善的框架，下载下来就能用。

挑战：

（1）消费级硬件上两件事造成的性能影响：CPU和GPU之间的带宽、GPU的显存；（2）CPU和GPU异构推理的问题。

解决方案：

（1）先用若干数据推理（离线，LLM Profiler and Policy Solver），找出对结果影响大的（和激活相乘之后通过激活函数，如果值依然很大，说明这个权重很重要）；（2）推理时，（在线，Neuron-aware LLM Inference Engine）对结果影响大的认为是少部分，放到GPU上计算，这部分是稀疏计算；（3）对结果影响小的，不计算；（4）有一个动态的预测器预测哪些权重应该被放到GPU上计算。

论文3

SparseTIR: Composable Abstractions for Sparse Compilation in Deep Learning

（ASPLOS’23）

作用：机器学习编译。类似于TVM的一种稀疏范式，可以支持不同格式和转换方式的组合，在GPU上取得一致的性能提升。

研究问题：稀疏编译在深度学习中的可组合抽象+Domain Specific Language。

研究层次：端到端机器学习编译，现在加入考虑对稀疏的支持，可以理解为是稀疏版TVM。

研究重要性：在深度学习中，需要可组合的格式和转换来加速稀疏操作，因为单一的稀疏格式无法最大程度地提高硬件效率，而单次编译器无法跟上最新的硬件和系统进展。

研究差异：现在发稀疏的文章的基本上还停留在算子优化上，试图通过对某个算子优化取得在一系列benchmark上的结果，但是benchmark并不一定是适用的，并且算子很多时候也是针对不同代GPU的体系结构然后做各种方法（换行，压缩，etc）的组合。这篇文章想要做的是混合式的稀疏算子（在不同情形下采用不同的稀疏格式），并且随着计算过程的进行或者GPU硬件更新，通过重新编译模型就能得到更好的稀疏加速性能。

挑战：

（1）高性能稀疏运算符的开发既困难又繁琐，而且现有的库无法满足新算子不断升级的需求——交给编译器来做选择；（2）一次编译不能永远解决问题——那就多次。

解决方案：

（1）首先提供一组可使用的稀疏格式和转换方法，将其抽象为中间表示（IR）；（2）编译器自己选择具体的方法，将其编译为GPU核函数；（3）提供一组API供机器学习编译的程序员使用；（4）目标平台定制的代码生成（从而改善不同代GPU之间的体系结构不同带来的问题，用传统写算子的方法没法全部覆盖）。

论文4

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

作用：LLM MoE（Mixture of experts）开山之作。MoE为不同的输入选择某一小部分权重进行激活，从而在模型迅速变大的时候计算量能够保持相对不变。

研究问题：提出了Switch Transformer模型，针对深度学习中参数重用的问题，提出了一种稀疏激活模型的解决方案。另外，通过简化MoE路由算法和设计改进模型，降低了通信和计算成本。

研究层次：所有的剪枝都是MoE中的一个专家（expert），而且大概率是静态的，所以大多数剪枝的目的是为了找到一个好专家。但是，论文4作为LLM MoE的开山之作，在数学上为MoE提供了支持。然后讨论了并行性。

研究重要性：为专门的架构（TPU）支持；实际上也是模型稀疏化。

研究差异：这篇文章是重新设计了网络模型，通过训练来实现和剪枝相同的效果；其他剪枝手段是通过在已经训练好的大模型上设计算法来剪枝。

挑战：

（1）模型量迅速增长，计算量也迅速增长，希望能够尽可能减缓计算的增长；（2）如何正确选择专家。

解决方案：

（1）将FFN MoE化；（2）将Attention MoE化；（3）用Top-K来选择专家；（4）实际上是使用大型的稀疏模型来创建更小的、稠密的模型，然后这些模型也可用。

论文5

GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints

作用：很短的文章，仅仅是提出了一种炼丹的方法，然后现在大家都在用。

研究问题：训练和推理时都采用多查询注意力（MQA）和分组查询注意力（GQA）的方法，提高推理速度，switch transformers的变种。

研究层次：改进模型，并提供了将目前的multi-attn转换为multi-query模型的方法。

研究重要性：提高模型的可扩展性，和switch transformers基本一致；但是通过分组查询的方式能在不怎么损失精度的情况下加快推理速度。总之，什么方面似乎都可以做剪枝，而且剪枝并不意味着只是简单的把参数量减小，可能模型结构也会有很大的影响。

研究差异：这篇文章考虑的又是关于attention head的剪枝了（相当于），现在只保留一个attention head，因此在qkv相乘的时候，kv都分别只有1个。

挑战：

（1）加载key和value的内存带宽开销。感觉仅仅只为了解决这一个问题。

解决方案：

（1）既然加载key和value都需要很大的内存带宽开销，那就只使用1个，这是模型上的改进；（2）提供了将多个key和value剪枝为1个的方法，将多个key和value取平均值然后做池化；（3）如果不想太激进剪到1个，也可以剪到多个，这就是grouped-query。