LLM??AI??????????

????

1. 普遍使用 MOE架构降低算力需求
2. 高度定制化的集成度高的大算子
    1. 定制化的核心 Attention加速算子: FlashAttention
3. KVcache的压缩、加速等: Deepseek的 Flash MLA
4. 混合精度及量化
5. 低精度支持及累加精度保证
6. 多卡互联技术，包括快速的分布式 all to all的性能，通信异步化，不占用计算核，最大化带宽利用率
    1. 更低的 latency，更高的 throughput
    2. 异步通信，动态执行
7. 复杂的存储地址控制，复杂的 MMU系统
8. Atomic的支持，用于复杂算法的 reduction运算支持

同步和动态

??&??

1. 通过传递依赖 /同步信息到存储器（ memory barrier）
    1. 优点: 避免调用 fence和 trigger一个任务的 leading和 tailing latency
    2. 缺点:
        1. 为了异步调用和挂起，需要保存抽象的线程信息
        2. 占用大量的调度逻辑和流水线存储资源，动态调度的代价
2. 源头统一调度和控制（ mailbox 、 global sync）

????

1. 通过预调度（ prefetch）等预估流水线的 latency进行提前 trigger，减少 latency
2. 专用的异步进行同步的信号，同步信号和存储请求分离，同步信号可以不随流水线传递
3. 把存储相关的程序指令存储到存储器端，控制器直接 launch，类似一个可以执行指令的 DMA
4. 所有的单元（ 1D、2D、DMA、NoC）都变成可以执行程序的单元、可同步单元，通过 launch 和同步来控制
    1. 需要解决可执行程序的 prefetch/preload需求
    2. 可以把整个程序通过计算图来抽象，明确所有 launch的时机
5. 通过预编译的 “动态计算图 ”表达所有的 launch和 join，非常细粒度的调度
    1. 专用的 “图 ”执行硬件单元，负责执行图、 launch、同步、配置参数计算 /生成和传递
    2. 标准的 launch参数传递和配置方法，负责动态 launch参数的配置
    3. 标准的标量流水线，通过专用硬件单元（专用指令）进行加速

???

1. 大容量、低延迟（高速）、随机访问
    1. 推理阶段，每个对话就必须加载全部的模型参数，只为了计算一个 token 的输出，算力利用率特别低
    2. 为了低延迟，不能使用低速存储替代，容量和速度的需求比例和传统 DRAM不一致，可能需要 1T带宽 1T容量 ？
2. KV Cache的机制对存储容量和带宽的需求
3. 跨界点通信开销

??????

1. 动态全局随机访问的需求
    1. MOE算法需求
    2. 稀疏算法

?????

1. 定义复杂指令，单指令支持操作数的 Dequent， GEMM计算
2. 设计很大的 L1甚至是 L0，用于存储 Tensor计算的左右值

?????

1. 超大的 dot变成多个小 dot的并行
2. 固定尺寸的 dot，和输入无关的 dot
3. 所有模型使用一种固定的层，同一种组件，不要很多种类，很多奇怪的 linear

Revision #17
Created 2025-02-24 05:36:42 UTC by Colin
Updated 2026-04-29 07:33:23 UTC by Colin