AI软件栈的一些思考

https://zhuanlan.zhihu.com/p/378389269

None Sun Jun, 06 16:06

最近有更多精力花在底层技术的思考和跟进上，也有了一些想法，记录下来，看看后续会有什么有趣的演化。

1. **对于Google当时为TPU选择了XLA的技术路径，自己现在有了更深刻的理解和认识。**

比较早以前，就有不止一位朋友告诉我XLA是最先为TPU服务才搭建起来的，后来发现居然在GPU这类异构硬件上也有收益(**fusion带来的访存和kernel launch开销的减少**)，所以后来加入了XLA GPU乃至CPU的支持。在AI新硬件领域，也知道有不止一家AI硬件公司选择了XLA的技术路径。

现在再回头来看，会觉得这个技术决策有很强的历史必然性，有如下几个原因：

1). 新硬件要获得跟existing硬件相比的差异化（性能，性价比），几乎是一定要走Domain Specific Architecture的路线。因为在摩尔定律放缓的前提下，特别是绝大多数公司做硬件都是fabless的模式，只靠工艺带来硬件性能，性价比提升已经是不可能的事情。所以结合自身的需求，以及对场景的理解，进行领域相关的硬件架构设计才可能带来红利空间。**我的观点是，只走市场主流成熟标准(比如CUDA)的兼容，会是一个稳妥性的过渡，但是是不可能真正获得像样的市场空间的，因为标准背后往往决定了软硬件架构，在别人的标准下去设计新的软硬件系统，留给自己的腾挪发挥空间就太小了。**

2).DSA架构的设计注定了会存在编程的不友好性。这不仅仅是硬件架构师的能力是否到位，而是因为DSA架构往往意味着和成熟硬件相差较大的programming model，在用户进行大量的适配学习之前，只能存在事实上的'不友好“。我们现在说NV的CUDA programming model对GPU开发者比较友好，是因为经过了大量时间的打磨布道，NV上的开发者已经接纳了CUDA的编程范式。而一个之前没接触过CUDA的工程师，一上来接触CUDA，在材料不全的情况下，也往往会觉得有些反直觉。事实上，NV V100之后引入的TensorCore就可以视为和CUDA SIMT不同的programming model，能够用好TensorCore指令进行编程开发的CUDA工程师，在NV之外，其实并不算多。不过，TensorCore带来的另一个潜在变化是，计算密集算子的重要性可能会下降，原因是硬件做了更多之前由软件来做的事情，未来如果NV在AI编译这条技术栈上投入更多资源，我想应该不是很意外的事情，因为当计算密集算子的开发工作量通过专用硬件进行显著缓解之后，瓶颈就可能转移到别处了。

3).DSA在带来架构红利的同时，也带来了可编程性的下降，那么如何减小上层应用和DSA硬件之间的鸿沟呢。这就是引入XLA这种技术栈的必要性了。XLA的核心我认为并不在于Fusion&Codegen，而是在于**HLO这层IR的抽象定义**。原因是，HLO把纷繁复杂的上层workload

抽象成了底层100多个原子算子，而再复杂的模型都可以通过这些原子算子拼装起来。对于XLA TPU团队来说，把这100多个原子算子实现出高效的codegen模板，再基于XLA fusion&codegen的框架进行组合拼装，就在相当程度上缓解了DSA架构红利和可编程性下降的矛盾了。当然，从工程实现层面，我还是很欣赏XLA里关于Fusion&Codegen的整体架构设计的，但我们评价一个系统，还是需要抓他的关键点，所以我会重点highlight HLO的意义和价值。

4).即便有了XLA TPU的支持，也听到不止一次有人提到，Google TPU走得并不算顺。这从目前为止Google仍然是NV GPU几乎最大的买家可就可以看出来了。这里面的根本原因，因为我并没有在XLA团队内部工作，所以并不得而知，仅从Google公开的一些材料和论文，我的猜测是，TPU的架构设计可能过于domain specific了一些，在硬件和软件之间的那层分界线的切割，以及programming model选取得还是不太合适，使得哪怕有了XLA这个工具支持，仍然在模型通用性上存在比较大的挑战。软件做得再NB，如果硬件架构设计本身存在先天的局限，也就决定了整体软硬件系统的上界了。

5).再稍微做一些延展性的思考。国内的AI新硬件系统，目前做的最好的我觉得还是首推华为的昇腾平台。当然昇腾平台是投入了海量资源进行开发，靠大兵团拱出来了当前的格局（圈子里几乎都知道昇腾平台消耗的人力是显著超过了其他的AI硬件公司的，人效其实并不算高）。两年多前，第一次了解到昇腾硬件的AI编译器是基于TVM来做的，当时还好奇为什么没有考虑XLA之类。现在再回头来看，其实选择TVM还是XLA对于昇腾的区别可能并不大，**因为这里最本质的挑战是，在NV GPU上，我们为每个原子算子开发完codegen模板之后，再进行装配所需的effort是远小于在昇腾硬件上的。原因是DSA在带来更强domain相关算力的同时，也提出了更高的要求，比如对fusion的饥渴度（一个不是很精确的比方，如果说NV GPU上不fusion的性能损耗是1，那么在昇腾这类AI硬件上的损耗可能就是3甚至5），以及第一代硬件架构为开发原子算子带来的挑战都使得无论是使用TVM，还是XLA，还是MLIR，其实本质上都是差不多的。AI编译的不同框架解决的是脚手架的问题，而新硬件最核心的还是硬件架构的设计所寻找的那个性能和可编程性的trade-off是否足够得好，这决定了在上面构建一套新的软件系统的天花板。**当然，如果华为有机会再进行迭代的话，我相信以华为的执行能力，昇腾会找到性能和可编程性的更好的trade-off，但可惜现在被限制生产，也是有些唏嘘了。

6).如果时光倒流，假设YY我能够有机会为TPU再做一次底层软件的技术选型。我会怎么考虑呢？我想大概率我不会选择XLA了，因为XLA在其诞生的时代，有其非常优秀的体现，但同时也能看到其limitation。我们不去说XLA先天的dynamic shape的不足，在XLA里想进行协同开发，或做一些扩展，往往需要对整条链路有E2E的了解才可能高效推进起来，并且扩展对系统的侵入性也比较强。如果是没有任何包袱的话，我会觉得MLIR的技术栈更为优雅一些，值得作为新硬件的软件基础层。当然MLIR社区大神太多，观点太多，也比较发散，反而不如TVM社区更加凝聚，这是我觉得MLIR社区做得还不够好的地方。一个组织，一个系统，缺乏一两个灵魂人物来制订统一的原则，其实容易产生类布朗运动，既可能带来新的创新空间，也可能带来技术演进的损耗，在此不表。

但是，但是，有一点一定要强调，MLIR，XLA, TVM本身并不解决新硬件上软件栈建设最核心的问题。最核心的问题其实是对新硬件软硬件边界的理解，这涉及到ISA，微架构，编程模型。从这个角度来说，其实MLIR反而是术层面的东西了。如果硬件的软硬件边界划分不得当，用再优雅的技术框架，也其是与事无补，徒唤奈何了。

7).TVM社区过去一年花了很大精力做底层重构。现在来看，我其实不太拿得准的是，天奇推进这次大重构，是想解决之前已经看到的新硬件上AI编译效率性能的问题，还是期望capture未来新硬件上编译效率性能的问题，如果是前者我觉得可能还好，如果是后者我其实觉得可能存在一定的risk。因为新硬件为了追求性能和可编程性的trade-off，往往会引入一些独特的设计理念，而在之前我的观察里，TVM并没有能够capture到这样的策略空间。**毕竟TVM脱胎于Halide，在Halide时代，编程模型还是以SIMT或SIMD为主，而到了DSA时代，编程模型其实在发生剧烈的演化和试错，远未收敛，这其实给试图cover新硬件的通用软件栈的设计（类似于TVM)带来了挑战 。不过TVM过去有一个重要的feature BYOC，我倒觉得这是一种比较务实的，基于框架思维缓解这类挑战的策略了。**

2.软硬件协同设计是一个非常有意思的技术topic。

当有能力把握、影响到全栈系统设计的时候，一个业务功能点的支持，在整个系统栈里是往硬件多挪一点，还是给软件多派点活，这是一个充满探究可能性的事情。在完成软硬件分工拆解之后，再来考虑将硬件能力以什么样的programming model暴露给软件，也是一种有趣的体验。某种程度上，你会体验到一种仿真的“上帝”视角，决定了在你构建的这个比特世界里，运行的法则是什么，物理定律是什么。如果最终这个工作还能带来可观的商业结果，那会是巨大的成就感。之前和朋友聊天，说到对AI的推动，NV起到了至关重要的作用，就是类似的成就感了。如果能够有机会在初始阶段就参与到类似Ion Buck那样的工作里，并见证这个工作的开花结果，那会是一件多么幸福且幸运的事情了。