杨军

蒋力 18:15  
有非常多的问题和思考，然后一会大家想要加入雪峰总这边讨论可以可以微调过去。然后我们下一位请杨军，以前是深度参与阿里PAI平台的研发，有非常多的思考，也是我们群里的或者是大网红是不是？好，杨军老师欢迎。

杨军 18:45  
然后谢谢蒋老师邀请了，然后我 slides做的比较简单，画了一页图，麻烦切换到下一页。

【P2】  
对，可能我这里会说的稍微多一些，更多希望有一些互动，因为可能在slides里边我不会说太多很具体的一些工作内容。然后这个图的话因为刚才雪峰其实已经提了非常好的一个背景，包括AI编译，这里边其实我想我应该有一些不同的输入给出来，可能是对大家可能会有一些启发性的。

其实整个从架构来说的话，大家都会把AI编译分成大概我们前端、中端、后端三个layer。不同layer的话，他们所面临的问题，他们的取舍也都是不一样的，所以我会沿着三个layer的话分别来阐述一下一些我的一些思考。

另外就是说可能除了这三个layer以外的话，当我们有机会从软硬协同看系统设计的话，哪些放在硬件里面，哪些放在软件里面，我也会给出我的一些可能比较premature的一些思考。对，那么首先前端，其实现在我觉得前端是比较有代表性的东西，我们说国外的框架的话，那么可能看TF、PyTorch、JAX算是目前比较hot的框架。当然国内的话包括像MindSpore、Paddle、OneFlow、MegEngine，其实会有很多的一些我们会做的非常好的工作，包括计图。对，那么其实如果我们从它整个的问题的原则性来归约的话，我会遇到包括几类比较大的问题。

第一个就是说我的算子颗粒度的问题，是个从Caffe时代慢慢演化过来，到了TensorFlow时代的话，我们看到了一个问题。那么在Caffe时代的话，我们可能可以非常方便的去魔改Caffe，我们加入一个layer，但是我发现每当我的算稍微复杂一点的话，我就加一个新的layer，这样的话其实对于整个我的model sharing和我的开发的效率不是很高。于是就从TensorFlow开始的话，可能会是用以更细的算子粒度来描述模型，再进行编译执行，作为一种模式了。

当然TensorFlow的model其实它的模式不是原创的，其实在那之前的话 Theano已经做过类似的事情，只是没有像TensorFlow推广那么hot而已。这个方式它带来表述灵活性的同时的话，也带来了相应的代价，比如说我们带来的这种kernel launch的开销以及访存的开销。那么这个问题可能我自己个人判断思考是说在硬件加速器流行之前的话，还是可以接受的，因为我一次访存一次launch的开销，跟我的整个加速器的gap没有那么庞大，那么它的灵活性我是可以去mitigate的。但是随着GPU，随着V100的tensor core，随着现在的像A100，包括明年NV的下一代硬件，包括现在很多AI芯片的出现的话，我们发现算子颗粒度过细的话会带来很严重的问题，我可能大量时间花在了访存和等待上面去。那么这也相应的会有很多的一些软件硬件的手段，希望解决的问题，后面会稍微提一下下。那么这是第一part算子颗粒度带来的一些优势和一些invitation。

那么第二个就是说我们算子覆盖度的问题，那么这个问题的话其实也echo一下雪峰的观点，就是图灵完备其实是我们对一个AI编译框架希望的要求，但是挑战就是说我对于新架构来说，我怎么能够做到图灵完备，举一个简单的例子，可能像Texion2是做语音比较popular的一个model，那么这个model比较有代表性，它除了典型的一些深度学习的算子以外的话，包括MT，包括solver那么这些东西都需要我们去提供支持，那么每当我更换了我们的backend以后的话，我的硬件以后，我都需要考虑是要cover所有这些东西，那么这种覆盖度的话，其实对于这种编译器的支持的话会带来蛮大的一个挑战，这是第二个part。

第三part它就是eager mode。Eager mode本身的话，其实在Chainer这个时代的话就比较流行的一种模式，那么其实对于我们去所见即所得的交互式开发非常有效率。但这个模式其实在加速器越来越流行，硬件越来越powerful的时代，其实非常有挑战的一个事情。

举个例子，前段时间跟一个朋友聊天，不能说是哪个公司的同事，他了说句话。他说那些只会用eager mode的人，不配用A100以上的卡，非常的激进。原因是因为在A100的话，我们打开tensor core以后，我的性能是很powerful的，导致我大量的时间在交互里面去轮转，我其实受不了。那么包括像有些硬件公司的话，为了去解决这样的矛盾的话，会选择一些看起来不是最优雅或者不是最scalable，但对这个硬件来说比较行得通的方案。比如说可能像某家芯片公司的话，它的每次kernel launch的开销很大，因为它是要把它整个kernel做一个在线的编译，dispatch到很多的tile上面去。那么做这个工作的话，一次开销可能会是以秒级为单位。对于这样的场景的话，他恨不得说把整个图切换到我的加速器里执行完。于是他就选择了一个看起来比较特别的路径，他可能PyTorch的话会把PyTorch的前向图给抠出来，自己做完所有的auto diff，做完所有的model。这能保证这个图对我全吃进来能够获得好的性能，但这样相应挑战就是说损失的我在算子的灵活性、覆盖度和通信这块的一些优势，导致我在换个场景的话，它消耗大量是为了工作，那么这里面本身也是说eager mode在带来我们前面表述灵活性同时带来的一些对于软硬件的挑战。

第四part的话就是dynamic shape。其实上午已经不止一位嘉宾老师都提过这个问题了，那么dynamic shape其实是一个在这种AI现在的模型里边非常常见的一个场景，那么我一个推理的batch size的变化是比较simple的一个case，那么我做个detection models，我一些bounding box的变化都会导致我的整个的计算模式不是那么的规则，那么做优化其实有一个特点，其实我们希望假设这个问题是固定的。因为固定的问题的话，我们是能做很多specialized的一些优化，但是当我的shape不断的变化的话，那么就会break很多的一些假设。所以当我在记得三四年前，我们看到FC的static shape假设的话，非常不理解，就很奇怪，等到自己有机会开始进入到更多硬件的系统，会发现这是个非常nature的符合硬件架构师的一种思维。我非常希望这个事情是固定的。当它固定以后的话，我能做极致的优化，但是软件的思维跟这个可能是不一样的，于是需要在中间找一些mitigation。

当然可能像MLIR出现以后的话提供了一个路径，说我们能够在表示层描述dynamic shape行为，但是fundamentally他没有解决说我们shape变化本身怎么在灵活性和性能获得好的trade off，那么这是另外一个问题。

第五part的问题是说dynamic control flow。上午其实林老师也提到的，我的模型非常flexible，我可能根据我的不同样本，我之间的分支不一样，那么它本身在我的单个机器的执行会带来一个挑战。那么对于我多个机器的话，可能会带来隐式的这种struggle问题，导致我的计算量的分配不是那么规则，于是对我的整个策略的设计，我的优化会带来很多一些麻烦的地方，所以这就是在我来看的话，是整个前端可能带来的一些有趣的现象和一些挑战，所以在这里面先做一个大概的summary。

然后第二part就是我们在中端看到的一些问题。因为前端是面向用户的，那么前端面向用户的话，通常需要转换成一个相对来讲更为干净的一个中端的表示，便于我们做一系列的变换和优化，那么这个里面我大概把它称之为是一个hardware-agnostic的优化。当然划分可能不是那么的权威，是我自己按照我的理解划分的，他大概是说我们希望在这个里面做一些跟硬件不是那么相关的一些优化，但实际整个系统拆解我们不能做到完全那么的干净，所以在这个里边难免会融入一些硬件的信息，甚至某些底层的hardware信息的话，需要通过某些机制去透传到中端来来定义做一些事情。

那么这里边我大概可能列了4类，我觉得比较有意思的问题。

第一类是在中端做策略探索，比如说我们希望去做自动并行的策略。那么关于这类工作的话，包括像GShard的，包括像MindSpore都做了很多工作，其实像阿里这边之前的话也做很多工作，像刁岚松是我的前同事，他们在XC里面的话围绕着并行策略做了很多的工作，所以如果大家有兴趣的话可以去找岚松去讨论，岚松很喜欢讨论这样的问题。

那么除了并行策略以外的话，包括上午有一位同学问到了layout的问题，那么其实也是本质上叫做策略选择。那么只要是策略选择的话，都很可能会变成一个NP hard的问题。那么在NP hard的场景下，我们怎么能够获得一个相对好的性能跟探索速度的trade off。

第三个是混合精度，混合精度这个问题其实我想今天可能没有寒武纪同学在，如果有寒武纪同学可能会有感触一些，因为当我们会想通过硬件追求一些非常卡丁爱的极限系统的话，或者克服我的工艺限制的方面，就会发现就会带来对我的整个的策略更多的要求。

那么寒武纪的这种低精度的话，其实在我来看是这样一种探索，虽然可能有不同的一些观点comment这个事情，但从技术本身来说的话，我认为是一个非常有意思的探索。对那么在策略探索这块终端的一些事情。

那么第二个可能会比较的耳熟能详一些等价图变换，可能像TASO这样的工作就是应该非常好的代表，包括你有些别的工作这块我就不去展开了，我觉得可能大家会更有些common的sense，对。

那么第三块就是关于fusion的问题，因为对于这种现代的新硬件的话，往往我们需要做很激进的fusion来缓解kernel的dispatch以及访问开销的问题，那么我们该怎么样做fusion？Fusion本身其实他确实是说像上午有一位同学提问问到的，包括是尹老师提到的，不是越大越好，当我fuse粒度很大的话，我的整个memory的消耗，我的SRAM的消耗，我的寄存器的消耗会导致我的并行度可能被限制，可能反而导致说我虽然获得了data reuse，我虽然减少kernel launch开销，但是我的整个硬件的利用率反而下降了。所以其实这里面存在一个trade off， 我们要看到我们到底对不同的模型不同的子图，我们应该在fusion plan空间里边做什么样的一些探索。那么这个就引申出来说一般来说为了做这个问题，我们需要有一个cost model来定义 cost function，帮助我们去做一些枚举。

一般来说在这一层的cost model的话，我们可能不希望做的非常的复杂，原因是因为这个空间比较庞大，所以我们希望能够做得小快灵得一些，那么这块其实之前在阿里巴巴的话，我们同时有一些工作，可能大家如果有兴趣可以refer。大概是我们在XLA的中端的话，我们基于cost model做了一些fusion plan的探索，是一个启发式的类似DP的算法。那么这个中端可能我看到了一些有趣的问题。

那么最后到了后端，这可能是跟hardware跟芯片相关的东西了，那么因为我不是硬件的专家，所以我就非常粗暴，我把硬件分成了GPGPU、DSA、CPU这么几类，其实如果分得更清晰的话，我们可能还可以说什么CGRA这样是一个特别的流派之类的，这些我通通就不管了，我就先把它简化成这个情况，可能对于我描述问题。那么在backend这个层面的话，我们需要做非常多的跟hardware相关的一些优化。那么这里面我也列举了大概是5类我觉得比较有意思的问题，或者说我认为目前没有完全解决非常好的问题。

第一个就是说我们怎么样去描述张量计算的空间，因为在我来看的话，Halide的也好或者别的一些或TVM的之前的版本也好，他们诞生的时代张量计算没有那么hot，所以其实整个的抽象的话是以寄存器，以这种非张量指令为粒度做调度优化的。那么这种调度优化的策略，当我们在张量计算都会发现，它的整个的灵活性探索的便利性也不会非常的方便，所以其实目前我知道的或者我看到一些情况，有不止一家硬件公司在做这种张量相关的codegen的话，都会选择不使用TVM的策略，也没有选择走向MLIR或者说是FC的策略，原因是因为确实在快速演化的过程中里面的抽象不是那么容易做，或者说目前的抽象的工具做的不够好，我需要我结合我的hardware的特性来做相应的抽象。

那么另外一个问题就是说本身硬件在张量计算这块的话也存在讲究，那么我可能是不是引进一些更近的 buffer，我需要选什么size，导致可能本身这个策略空间的话也有些不一样。所以在这个地方的话，其实大家会发现会有一些好玩的一些要在这个里面做抽象探索的问题的发生。

那么第二块是我自己的观察，可能开源社区相对关注少的问题就是张量跟向量交互的问题。因为一个AI的模型不是只有张量计算，其实上午林老师提到的我特别有感触，因为从算法层面来看的话，我们很多时间花在张量以外的一些building block里面。但是当你不断优化你的张量计算以后的话，可能你的向量部分是那些非常flexible部分的话，就是会成为瓶颈的。那么他们之间怎么进行交互，怎么进行并行，怎么进行传输，怎么构成一些比较高效的生产者消费者的关系，其实这里边需要去考虑的问题。目前我看到的一些主流的工作可能还没有cover这个topic，可能是我觉得后面也许会有意思，催生一些好玩的工作的一个地方。

那么第三个还是fusion codegen。因为一旦到了跟芯片结合的话，我们绕不过fusion这个事情。那么fusion codegen的话，本质上其实也是一个策略探索的问题。那么我的不同的kernel，我的拆解的策略，我去loop的策略，当我组合在一起的话，本质上就是个组合爆炸的问题，那么对组合爆炸的问题，我们该选择什么样的手段能够在一个比较decent的time budget里边获得好的性能，其实在这个里面我们需要做的事情。

那接到第四点，我们一般来说为了开拓这样一个问题的话，我们会需要引入一个cost model。那么 cost model的话它会相对来说就要更精确一些，因为我们希望 cost model的capture更多硬件行为的话，可以指导我们去做比较好一些codegen的策略探索，这块其实包括在GPU上面，包括在一些芯片上面，其实都会有些探索工作再去发生这些事情，对。

第五个就是compiler和runtime的动静结合。这块也是我自己的一个感触的，就是做优化的同学有一个特点，就是我希望假设说我是在一个athletic[32:39]的环境里面做优化，这个环境我是独占的，我可以用所有的资源去做优化。但是在真正的工业生产环境里面的话往往不是这个样子的，可能会有两个workload共享一块GPU或者共享若干个CPU的core，那么他们之间串儿【32:52】很严重的。这个时候的话我们其实并不太希望把所有的优化都在静态planning做到极致，而是我们希望留出一个buffer和一个tolerance，我们在runtime里面来进行结合。这是我觉得其实google后来想去思考TensorRT的话很重要的motivation，当然这个问题当我们已经有了成熟的硬件，已经有相对好的性能以后的话，哪怕通过更好的runtime跟compiler的结合，能获得20-30%的提升，因为这种设计带来的cost可能导致说暂时不能成为domality flow【33：21】，这是另外一个问题。但从整个typical【33：24】的角度来说的话，其实我个人会觉得说compiler跟runtime他们的动静结合会是个非常有意思的一个问题。

所以我这边大概就是说从我看到的三个layer的话，来去大概说了几个问题，那么最左边是我画了两条线，就一个是往下的硬化的一个线，一个是软化的线。

那么硬化就是说当我们具备了整个系统的全栈设计能力以后，哪些功能我们觉得往下挪一挪，放在硬件层面会更好一些。举一个比较simple的例子，如果说我是一个对推荐的模型有强需求刚需的一个团队，如果我把我的embedding查表的操作部分下移到硬件里边，能不能获得更好的提升更好的性能？其实这个里边是存在着一些策略优化的探索空间在里头。

那么软化是另外的一个反例。当我有机会把软件硬件全程考虑的话， Maybe某些在硬件里面做很复杂的事情，我在软件里面可以mitigate。那么举个例子，其实最近NV刚刚放出了MLPerf的打榜的结果，他们在PyTorch的bench【34:26】里面的话，应该有了进一步的性能优化的一个提升。如果有同事有同学去看里面的细节，可能会注意到，它在PyTorch里面的话引入了CUDA graph，那么在0.7的版本的话是在MNetch【34：36】里边，那么NV引入CUDA graph在1.0里面在PyTorch这边加cuda graph，那么cuda graph的核心是说我希望用尽可能轻的软件的一些手段我去mitigate，我在这种eager mode里边kernel launch的开销，从而获得更好的一个全栈性能。在我来看的话，NV在这个地方其实他没有选择说我在硬件里面做更多的工作，希望去fix这个问题，我希望在软件system层面去解决问题，或者他所希望的一些投入跟收益的平衡点，这可能是我理解的另外一个例子。

所以说整个从这三个层面的话，我可能会大概是有这样一些的问题，包括一些思考，所以我的分享基本是这些，好，谢谢大家。

唐衫：  
我先插一个问题啊，我刚提 cost model，刚才金总说 cost model很困难。我不知道是获取 cost model的困难还是使用上的问题或者是困难。

金雪峰：

获取

唐衫：

获取，对于我们做硬件的人给提供cost model，就是你想要什么样的cost model，我们做硬件的人提供 cost model不是很难，我没太理解这难度。

金雪峰：  
你说的可能是非常细节的东西是吧？你是CA Model这种，但是这对我们来说没用，我们主要关心的是内存，比如说搬到哪里的时候，这种比较宏观意义上的。所以这个是有gap的，我理解这方面不是硬件就能够提供的。然后另外一个model上来说，分布式那个model就是硬件上用的，当然有bias。他比如说我通信的开销计算的开销是吧？包括这种分布式又是另外一回事，现在我理解有三个cost model。

第一个cost model就是用于大型的这种分布式，就是我切到多个device。第二个我假定框架需要把这个子图扔下来，这时抠出一个子图，这个也是一个很困难的事情。我拿多少个东西给你出来。然后第三个cost model是说我在做scheduling跟底下算子scheduling跟tiling的时候，要怎么来tiling，怎么来scheduling，这就需要。这三个cost model，其实说句实在的，我们在这种DSA架构基本上都不完整，或者说有些根本没有。没有的话，我就全是规则，你去看我们所有的像fusion这一块，就可能大量的其实就是规则是吧？我看到一个element-wise，然后我碰到一个reduce，我就变成一个reduce。然后甚至我们再到tiling的时候可能也是一种。

唐衫：  
之前看到一些是说反正他有各种策略跑，跑完再反过来，但是相当于是你用AI方式记下来。我理解你说这个意思，对。

说话人1：

反正我们其实也遇到比如说做一些自动并行的时候，你可能希望做一些 op的拆分，然后或者是一些比如说假如一个conv是吧？你可能希望把它放到比如说多张卡上同时进行这个conv，这个模型的方式拆分的过程中，你是怎么样去拆分 conv？其实这里面它，就相当于针对它的性能的一个预测，对吧？它就可以。