唐衫 03:04:13  
我们其实讨论了几个问题，有些没答案，有些好像有点有意思的答案。

第一个我们在讨论继龙这边刚才讲的，他们Rammer下一步的事。他们可能是想他们是希望说把硬件做一个抽象，刚才他其实那个图大概讲了一下，我觉得我之前一直对Rammer有一个疑问，就是说Rammer它对硬件的假设这个粒度是一个同构的东西，它每一个EU（element unit）同构的东西，面对我们一些至少你的SM core，加上tensor core，它就不好解决了，对吧？但是他们现在的工作可能往后，会把这个tensor core加上。然后当然相对来说，它还是对硬件有这么一个抽象，或者有一个建模，还是一个简化的模型。然后通过简化的模型导致编译器会，相对来说工作会容易做一些，强调它的一个基础。

然后我们讨论这个东西，我们讨论东西发现他有一个有意思的地方就是说，他有可能会得到一个我们一直在说一个合理的硬件架构，或者是一个从编译器比较友好的硬件的架构是什么样，我觉得虽然现在他可能刚开始的建模还是一个比较简单的建模，但是他有可能回答一些问题，比如说他现在的Stream core加上tensor core，那tensor core我可以先假设比如是4×4×4的这么一个tensor core，它workload下来编译以后可能会有一个结果对吧？可能会发现什么tensor core很忙，我是不是tensor core再增大这么大一点，对吧？因为它是个硬件的模型，或者它是一个从编译器角度来看硬件模型，我觉得这似乎可以往后走去看一下这种，其实我们做硬件是设计这些手段。我上午也讲了，我们这些可以用什么手段，其实最终就是面积用在哪。对吧？面积用在哪种东西上。划分对吧？是用在 general processor上，是用在tensor core上，还是用在 RAM上，还是用在 communication上，对吧这其实就是个比例问题。我们其实就想知道哪个比例对于我们常规的workload是合理的，所以我觉得他这个工作当然一个是编译的工作，另外一个是可以往后走一走，看看软硬件、一个更合理的硬件更合理的一个样子。

蒋力 03:07:06  
唐博，我觉得这个挺有意思的，你能一句话能总结这个问题吗？

唐衫

基于Rammer V2的软硬件接口抽象之类的吧，再往后合理的软件划分，我们一直在谈软件给硬件给软件提供什么样的接口之类的，这是一种探索吧。

蒋力 03:07:50  
我们觉得这是个problem，这是吧？

薛继龙 03:07:56  
问题可以叫成合理的软硬件接口是什么，对吧？

唐衫 03:07:59  
对，从编译器的角度去探索一下这个问题。所以回头我们再抽象一下。

第二个，我们刚才说了一下这个算子概念到底有没有意义，大家也简单讨论一下。反正这可能从这个算法逻辑概念上，它是从逻辑概念上来的，但是现在其实用的中间的包括算子的边界，但是包括融合，它其实也不是很强的一个概念了，反正我们大概讨论的是这个结果。

然后另外一个我们也还花了一些时间讨论稀疏化的问题，就是稀疏化的方向到底硬件该做到什么程度，怎么支持这个稀疏化。这块反正我们对可能也没有统一的结论，只是说反正硬件要支持不同程度的稀疏化很难。其实可能现在短期也就是这种结构化的提出先做一下呃，可但是编译器在中间可以发挥的作用应该是值得讨论的，可做的事情可能是比较多的，包括一些新的数据结构什么之类的。这是大概的结论。

然后另外一个讨论的就是说大的model今天讲了，上午也讲了，林总也讲了，大的model到底是我们老百姓怎么用。有点比较brain storming的一些讨论，包括是不是我们我们按这种网格计算的这种分布式或者联邦学习的模式，我们每人贡献一点这个东西我们能训的，大家一块训一个，用自己的电脑组合起来训一个大的model有没有可行性。然后包括他训完以后怎么落地。就是他这种很大的model他给你用的时候，现在可能大家有时候给你提供API什么的。它背后是会做一些压缩，还是会做一些 distillate什么等等这些没答案，我们也不知道各位如果有答案也可以给我们讲一下。

然后刚才又提了一个新的问题，就是我们今天可能说compiler比较多，没太说 runtime的问题，就是compiler和runtime，我们后面是不是有一些更合理的分工，或者在某些情况下什么东西放在什么地方做什么，我觉得今天没太讨论，我们也没什么结论，提出来这个问题后面也可以大家有兴趣可以再聊一下。

然后其他就没有什么太多更深的，我们其实就是第一点是讨花时间讨论挺长的，觉得也挺有意思，后面可能会期待这边一个工作，我们也看能不能有些合作在里面。

蒋力 03:11:41  
要不继龙补充一下 Rammer，这个结论能不能用更加通俗的话说一下？

薛继龙 03:11:47

我们就是想抽象以后叫做tensor primitive，然后试图来以tensor primitive作为软硬件的接口。

蒋力 03:12:00  
所以跟现在的这种软硬接口有什么区别？它是变更简单了，更加单一了吗？

薛继龙 03:12:08  
对，它的简单就是，对硬件来说只要实现这些primitive就够了。然后对软件来说，我们的编译其实很大的重要意义是可以做到白盒编译，就说是以前的编译像TVM是通过搜索去做的，你很难理解他的行为以及编译的时间很长，然后白盒编译的话就是说你可以一键编译，并且性能可以做分析，相当于有了cost model。

蒋力  
那么第二个算子概念有意义吗？我记得敏捷是不是也有问过这个问题，其他两个讨论有没有讨论过算子的概念？

说话人7 03:12:46  
这个我们有大概几个点。一个是说算子更多的是一个研究员的一个领域概念，是用来做交流，还有一些习惯，但是在真的算子优化层面不是非常必要的。然后现在大家都在做图算融合不也就是在打破边界。后来我们还探讨了一下，就是说我们假设不走这个层次，我们能不能有一个比算子更往下的，但是能够相对比较统一的IR表示，然后目前看起来很难。

蒋力

所以你刚才说在算子往下更统一的表示，跟那边的primitive有点相似。

唐衫

这边的可能就是一种回答的思路，对，至于能不能统一我们不知道。

说话人7  
对就也许可以，但是现在明显还没有一个足够标准的东西，它不像大家都看过都是一样的。

唐衫  
你们有什么讨论吗？咱们有可能讨论这个问题吗？就是讨论出相对来说大家比较有共识的。不是operator，而是再往下的IR表示，对于不管是编译还是硬件来说都更友好一点的这种表示，你刚才不是这个问题，是也想往下讨论一下吗？我不知道你们讨论到哪儿了。

说话人7

我们就停在这儿。

蒋力 03:14:21  
因为其实两个组其实都停留在这，这个问题是有共识的对吧？我们要在算子下面可能会有一个更统一的IR。

唐衫

算子本身的意义我们思路意见应该是一样的。

蒋力 03:14:36  
好，依源你看还有什么其他的。动态shape这块，是吧？

说话人7

我记得咱们是说动态shape支持的意义是吧？有没有必要支持动作shape？如何支持？

说话人8

我觉得这肯定有必要是有啊，哈哈哈哈。

蒋力 03:15:03  
那么就是说动态shape的价值到底有多大，然后如何支持是吧？

金雪峰

这个问题是不得不支持。可能这有两个问题，一个是静态图是不是要支持动态shape，这是倒是个问题。但是你从这个模型的角度，一些检测模型本来就是个动态shape的是吧？当然你也可以去强制转化成一堆的静态图去搞它，这个感觉代价太大了，所以说这里这个问题是不是说静态图里面要不要支持？我要理解一下这个问题。

唐衫

好像我们的问题是说，支持这个动态shape又保证性能，应该的途径是什么？还是说可能损失一些性能去更简单的去识别。我觉得这个和性能之间的trade off，好像是这个问题。

说话人3

是指compiler编译kernel时候的动态是吗？就是说是框架层面的动态还是编译一个算子图的时候？

唐衫 03:16:11  
我也记不太清了。

蒋力

没有关系，这个先放这。雪峰，你那边的结论就是说，这个问题重新定义，应该是说是否在静态图里面要去支持动态图？

金雪峰

我只是刚才跟跟他们check一下到底是什么问题。

唐衫 03:16:38  
我们就好像聊的多有点忘了，就到别的问题了。

蒋力 03:16:45  
好，那么我们还要看看，runtime和compiler如何更合理的分工？另外两个讨论组有没有讨论过这个问题。

说话人10 03:16:59  
好，其实之前也简单讨论了，就是说runtime，因为我们确实今天主要考虑compiler。不过我不太清楚另外一个小组怎么定义的runtime，因为我可能会更多聚焦在说我们的compiler编译出来这种binary文件以后，我们runtime要负责把binary给加载进来，然后这个时候说可能会跟芯片这种driver非常相关。就说你的driver到底是开放，比如说粗颗粒度啊或者细颗粒度的接口，包括你的memory这种接口是不是开放出来。我觉得我们讨论的runtime是这个，我不是很清楚这个小组怎么定义的runtime。

唐衫 03:17:57  
我们也没讨论太细，我们还是说哪些事儿，比如说在runtime要干，要更努力一些或者是什么，我们没有什么结论。今天好像runtime讨论比较少。

说话人10 03:18:36  
其实runtime的话，我觉得如果从芯片角度来说的话，我们runtime怎么和芯片和driver更好的一个配合。比如说芯片，现在很多加速芯片是非常大粗颗粒度的，还是可能比如相对GPU或者怎样，他们会是更细颗粒度的。然后现在很多生态的环境栈runtime确实也是更加适配细颗粒度，因为他会在软件层可以做更多的事情，这个时候怎么去做这个trade off，我觉得runtime也是需要去考虑的。

蒋力 03:19:13  
好的好的，那么辛苦志愿者记一下，然后我们一下一位就是请杨军是吧？

杨军 03:19:36  
然后我就把我们这些小组讨论一些问题，找一些跟刚才谈过分享的有些互补性的会稍微说一下，如果有些我有遗漏的，我觉得大家可以帮忙补充一下。

第一个问题就是说我们今天讨论AI编译的话，其实提到了这个软硬协同设计，那么软件的背景同学跟硬件背景同学他们的background是不一样的，在这个背景下怎么能够有一个更有效的协同设计的机制，包括怎么能够形成共同对话，可能是我们觉得在这种AI系统探索里面的问题。

那么这里边已经说了一个事情，就是说如果我们能够有一些开放的可以共享的基础的话， Maybe能够简化复杂性。举一个例子像RISC-V的话，其实它在一定程度上讲，简化了我们为一个新硬件设计ISA以及软件栈的一些工作量。那么他可能对于所有使用RISC-V的公司我们可以有蛮多可以share的东西。甚至一些我们觉得可能相对不是那么的敏感的东西的话，可以贡献回去，相当于我们把整个社区的力量和公司力量形成一个共赢。那么这是硬件层面。

那么软件层面的话目前能看到可能还是各家有各家的特点，TVM、MLIR、XLA or whatever，那么软件层面有没有可能也去类似的攒出这样一个局来，然后这些不同的硬件厂商能够往一个大家有一个共识的一个软件基础里面去贡献软件code，贡献我们理解，那么以后如果再有style【03:21:00】出来想做新的硬件的话，想做新的硬件软件栈，这样的话能够去拉入这些工作。这是我们讨论的第一个可能觉得比较有意思的一个问题啊。

蒋力 03:21:08  
所以稍微再总结说，你是觉得像在软件栈里面是不是也应该有像类似RISC-V这样的一个联盟生态？

杨军

对，MLIR可能有希望。但MLIR的出来比较的晚，目前整个成熟度和完美度的话会差一些。另外MLIR社区本身也不是那么的凝聚，它缺乏一个像类似于天奇这样的一个灵魂人物来统一drive这个方向，导致发展的比较的多样化。对，所以软件上目前反而没有那么的成熟，在这件事情上。

第二个问题其实是一个可能跟workload有关的问题，就是说我们到底在训练场景对于这个性能有多么强的需要。那么可能我们在场有很多做AI系统的同学，我们关注说20% 到30%的对训练的提升，那么从业务来说的话，可能很可能说百分之二三十的性能提升，不如说你给我更多灵活性更有帮助一些。那么在这一点上到性能到什么程度，是我们做新的系统新的硬件需要观测的点。

当然这里面有一个有点意思的事情，就是当我们说系统的话，我们不可避免需要跟NV对标，那么NV跑的速度基本是说每2-3天一次大迭代，可能到了明年10月份的话，如果没有意外的话，他们的Hopper会出现性能可能是A100的三倍。也就是说哪怕我们不考虑性能，也需要考虑说不能比他差太多，才能去考虑易用性，所以这就是我们在芯片公司里面的话不得面临的一个挑战，我觉得这是一个我觉得比较有趣的一个问题。

然后第三个问题是这种大模型。其实主要是商汤的同学来贡献的一个观点，因为他们可能离workload更近一些。大模型的训练其实会发现，我们每当换了一个model，我们需要在整个切分的策略做很多的一些工作来保证我的性能跑的更好。可能不同的model切的效率不一样，如果标准transformer的话可能会好一些，如果再复杂一点可能不一样了。那么这个里边我们对AI 编译器这种自动化的分布式策略要求会更高一些。这个可能岚松听到了会很开心，因为工作有更强的需求。对，所以我其实觉得可能是一个题外话，我就觉得我们讨论应该更多的拉一些有这种比较强的workload背景来参与进来，像上午林达华老师分享，我就特别有启发，因为来自于workload的分享的话，能更好drive我们整个系统研发的方向，包括软件和硬件，这可能第三个的问题。

其他的我觉得可能是一些相对小的点，就是因为有些问题在唐博的里面也提到了，所以我这边就不再提了，我不知道小组里面有没有什么想补充的问题没有？大家可以来帮我补充一下，也想去听他的topic。

说话人10 03:24:21  
我一个就是说我们说是大家现在做很多AI系统，说比较分散，是不是有可能大家会更聚集一些力量去做这个东西？

杨军 03:24:38  
这个主题其实中午吃饭是进辉也提到了。对，进辉提到说我们这么多人其实都是属于做这个领域的一些同行，其实都代表了不同的视角。如果能够有一个合适的策略把大家以某种方式能够assemble起来，可能是一种虚拟的committee也好，whatever能够去把这些对这个领域的理解形成一个common的可能会reuse的东西，可能对于整个行业的进展，对于自己的创业有帮助，这是一个我觉得可能也是蒋老师去攒这么一个大的seminar，可能希望慢慢去引导去促进的一个东西。所以这也是想去提的一个topic和问题，希望能持续讨论。

蒋力 03:25:19  
我们其实最后有准备一个环节，就是我们要组成讨论组一个小的committee是吧，然后名字没起好，反正叫一个执行委员会，然后我们来探讨到底要出什么样的一个report，然后去组织这个会议对吧？我们肯定会讨论。

金雪峰 03:26:10  
因为那个内容比较多，我挑挑一些讲，然后可能有些大家觉得不全的补充。

第一个就是说为什么有算子这个层次，刚才已经讨论过了是吧？所以说我就不再赘述了。

第二个就是我们讨论了一下怎么构建一个更好的编译器的生态，实际上就涉及到编译器能不能标准化是吧？能不能把上面的负载更好的跟就映射到硬件上这里面。我先说自己的观点，我先讲一下就是说编译器生态标准化或者说开放化的当前面临的一个挑战。我并不是说有说这个领域不能形成标准化，我只是说它目前存在的挑战，我觉得分两个维度。

首先，我们谈MLIR。刚才蒋力老师说我反对MLIR，其实我不反对MLIR，因为我们公司也有很多团队做MLIR。只是说我现在对它做一个标准的 AI的编译器，可能我觉得他挑战蛮大的。

第一个是说因为MLIR是一个分层的编译器是吧？但是其实在MLIR出现之前，就是我们那种商用的公司IBM，包括它里面内部的编译器，它就是一个分层的编译器。但它的分层的编译器跟MLIR的分层编译器是不一样的。就是说像比如IBM或者说我们华为内部也有一些类似的编译器，它的分层是定义好的，我是两层还是三层对吧？比如说IBM的编译器，它第一层就是一个类似于clang的一个表达，就是for【03：28：06】这种东西，然后到下面就变成codegen这样的一个。但是 MLIR它实际上是一个机制，它告诉你可以做分层，但是他没告诉你有几层每一层是啥，所以说它不是一个标准，它是一种公共的机制，所以说我要澄清一下。

第二个我们也要理解，就是当初LLVM为啥会形成一个比较好的编译器的标准，是吧？那是因为我们在芯片硬件层面上，它的ISA是相对统一了，因为它是以CPU为中心的，然后你的协处理器，你的加速器就最后一个intrinsic，你可以嵌进去。但是我们现在面临的挑战，我们硬件在AI这个领域ISA太多了，估计目前来看还是一个挺困难的事情，所以说我觉得我们在编译器生态要形成标准，要形成一个开放的生态，可能要把这两个地方解决，解决好了以后我们可能就会形成生态是吧？

然后另外有一些大家讨论的就是说我们可能需要把高层的一些model的信息，就是语义的信息怎么更好的透传到我们底下，一层层的传下去。这个也是大家讨论比较多，我觉得这也是比较重要的问题。可能我未来假定比如说去基于MLIR去构建我们生态的话，整个MLIR里面的信息的设计也是很关键。怎么把高层的语义信息透传下去，然后里面也提到了一些相当于，因为传统的编译器有很多option是吧？O0啊，O1啊，O2啊，是吧？但我们现在好像 AI的编译器没这个概念是吧？要不就打开，要不就关闭，这个可能后面也是一个非常可以就是讨论的问题。

然后第三个问题就是说我们提到整个深度学习的workload是收敛的还是发散的。因为这个还是蛮重要的，如果它是收敛的，其实我们很多策略就比较固定，我们的框架，我们的编译器，我们的芯片可能就全部标准化了。但是如果它发散的事情，而且它的发散可能很发散，我们标准化就很难做。所以说我觉得目前来看，大家我觉得主流的观点就是research是发散的，但是我们看到在生产场景有一些收敛的趋势，至少在模型的定义上已经收敛，但是它的前后端比如数据处理、数据增强，到后端的模型的应用，它可能还是比较零碎的，这个是我们的一些观点。

唐衫 03:31:00  
我问一下业务上的收敛是说不同玩家都会收敛到同样的，还是不同玩家只是他在内部会收敛到一定的架构或者什么？还是大家有共识的？

金雪峰

我的理解类似于比如说CV、NLP是吧？大家都在玩transformer，那就比较好，这个事情就比较好处理了是吧？包括你说你再怎么变化，反正就这么几个基本的backbone。至少我们如果说在大量的生产场景，比如说 CV、NLP、推荐，大家有几个主流的模型，就变成比如事实标准，或者大家用大量的这些东西，我觉得这个就比较好，当然我觉得可能会有新的领域是吧？比如科学计算新出来可能是另外一种模型。但是没关系，anyway，就是某一些我固定了，至少我们的挑战就会比较少，因为我觉得research的问题，他如果不care性能，那就好办是吧？你就可以这个还是玩。但是我生产场景如果是固定的话，对于做芯片的比如说编译器的就是一个福音是吧？我把这些生产场景常用的负载搞好，是不是就解决大量的性能的问题了。

唐衫

对，所以我下面一个问题就是MLPerf是否代表了收敛？我想知道这个答案，我做芯片的肯定是想知道这个答案。

金雪峰 03:32:41  
我觉得我们好像没有讨论。其实我的观点就是MLPerf肯定是落后的，因为他肯定是比较慢。

唐衫

对我们现在没考虑这个问题，其实我觉得其实大家可以聊一下，就是说它慢，比如说是它也在变对吧？我们可能认为它有几个 model没用了，但是有可能再加一两个model是不是就基本有代表性了？我不知道大家怎么想，是不是说如果说收敛，我理解，那也不会太多。

金雪峰 03:33:21  
对，我觉得反正这应该是两个不同的问题。我觉得收敛那个问题，可能大家看看 backbone到底是不是趋同，对吧？未来，至少从趋势看。然后MLPerf我觉得这里面有很多玩家的例子，MLPerf本来就是有一帮人去定这个标准，然后来打榜，是吧？然后来他肯定会根据他的利益去制定。这两个问题不是完全一样的。

唐衫 03:33:53  
对，其实我不care他那个东西，我也不认为它就是有代表性，我就是想知道有代表性的是什么。你也可以说他都不对。

金雪峰

我觉得至少，比如说现在NLP用transformer这个事情，大家还是基本上就实际就这么用了，反正没人会跳出来是吧？

杨军 03:34:30  
这个问题其实我们这边有过一些类似的讨论，其实应该是？？【03：34：35】发起这个事.问题就是说可能我们有些model非常flexible的时候，可能就像说一个检测模型，可能30%是在conv之外，可能在一些非常flexible算子里面去了。那么对于这样的算子的话，可能大概率是不太容易直接被加速的。如果为每个这样的算子，我们都回到CPU里面算的话不划算。现在我们可能PyTorch里面的话，我们就说一个C的programs启动起来，在里面一些分枝launch出来其他的kernel。那么这里面我们有没有一种可能说，我们对这些我们能够看到的比较重要的workload，我们对背后的算子的在一个更细的粒度来进行一个规约，这个规约可能不是在这种TensorFlow/PyTorch op的粒度，而是在一个类似于像HLO【03：35：24】类似Relay这样IR的里面归约。如果发现哪些算子，在这里面描述不是很方便的话，我们来进行扩充，最终是能够形成一个封闭集。如果封闭集存在的话，其实我们对任何一家芯片公司，我们把这个集合支持好就行。

金雪峰 03:35:38  
我们也讨论过，就说不行，哈哈哈哈。下面会讲一下，但是我觉得你这个问题可能里面有两块。一块就是说。你是属于前处理后处理，还是属于backbone的，对吧？这个可能还是要分开。另外一个就是说这个不规则算子你是给research用的还是生产场景用的，这可能也要分开。

说话人19 03:36:08  
其实我觉得这个应该好几个组应该都讨论过，包括最前面提的一个问题，有没有存在一个算子以下的一个叫primitive的东西，能够把所有的东西都拼起来。我理解其实如果要真的要全，他就需要是图灵完备的，我们讨论过。

说话人20

我可能我表示反对，因为我觉得其实 deep learning model还是比较restricted的，他不一定真的要图灵完备。

说话人19 03:36:39  
对对，是这样，如果说你想要把这个问题完全的close掉，图灵完备你就close了，咱们就不用纠结了。但实际上要做的话，我们肯定没有那么多成本能给他做到图灵完备。那事实上可行的是说我们在一个受限的范围内，让它能够表示这个就是99%的这些case，这个应该是可行的。我觉得我们是不是可以去共同努力去造这样一个东西，就是一个primitives，它能够表示目前TensorFlow或者PyTorch里面99%的case。然后有了这个case之后，这一层其实它就可以把硬件和上层的框架一定程度上做一个划分，然后大家可以标准化地去做工作。

金雪峰 03:37:32  
现在反正我记得MLIR本身有个项目叫HLO，还有ONNX本来就是做这个工作的。所以说我为啥说其实很简单，ONNX现在成功了吗？所以说你觉得如果他成功了，那就搞定了。如果说他没成功，那就是还有问题。

袁进辉 03:37:58  
他有可能用那种比较抽象一点的，比如说Jittor里面对算子做了几类，一对一的，多对一的，一对多的，是吧？然后他其实用一个所谓的重索引reindex来描述这种，他不是给一个具体的算子是什么形式，而是说reindex大概是一个好像是多对多的吧，忘了。但是你可以在里面再写个小的lambda的方式，然后这个一算的话它种类很有限是吧？

唐衫 03:38:37  
我是觉得其实本来我们发起论坛或者发起讨论的这是一个话题，但是你说ONNX成功不成功，或者他为什么成功不成功，我们其实不太关注。我们是想说通过大家讨论我们是不是能讨论出这么一个东西，如果能讨论出一个或者大家基本上是有一定共识的，我觉得就有一定代表性，对吧？如果我们这些人都讨论不出来一个共识，可能确实也比较难。我们其实还是很有代表性。

杨军

我想echo一下雪峰的ONNX成不成功这个问题，还挺有意思的。其实我自己印象，ONNX在深度学习框架没有出现两家独大的情况时就很popular了，大家都希望去接入它，去用它来做一些中转和一些连通的工作。基本上进入到PyTorch已经超过了TensorFlow以后的话，至少Facebook内部是不再去推这个东西，因为它已经没有incentive了。所以这个时候的话它出于各种各样的原因发展了torchscript。于是出现了说其实现在ONNX最大的贡献者应该是Facebook之外，如果我没记错的话，应该是以微软为主来贡献更多，这是一些现状。所以其实确实如果说作为一个中立标准的话，它不能算很成功，TensorFlow也不是那么待见他，PyTorch也不是那么待见他。但是有个有趣的现象，几乎所有的硬件公司都会先做ONNX对接，包括推理包括训练，因为大家发现它太neutral了，非常适合我们的需要，这是特别好玩的一个问题。对，所以这个问题也就是说它不光是一个技术问题，还是一个商业问题。举个例子，如果说我们想干这个事情的话，假定说是我们拉着，因为像TensorFlow和PyTorch的话属于国外的公司，如果我们拿着MindSpore来聊的话，可能也在想像MindSpore的角度会有这样的动机去干这种事情吗？我的感觉是可能动机不会太强，因为一个框架好像没有太大必要说我花太多巨大的精力去打通跟别人对接，而我是应该不断扩大我自己的市场份额，所以这个确实不是一个纯技术的 problem，我的感觉。

金雪峰 03:40:38  
杨军说的反正我觉得是一方面，另外一方面我们也讨论过，就是说其实跟深度学习产业相关，因为我们实际上我也参与过很多标准的讨论。比如说我们做4G、3G的标准，是因为大家坐下来谈是吧？讨论完以后把大家写定说好，然后我们干5年对吧？然后下一个5年来下一代再谈，但是深度学习不是这种玩的模式，就是说我明天就做多了一个op然后你加不加？你来不及加是吧？来不及加怎么办？我就自己搞，自己搞的到最后就发现，与其这样，我为啥要这样做，所以说这个是一个非常现实的问题。所以这是个挑战，我觉得不是说不行，我们今天主要是把这种挑战说出来。

唐衫 03:41:38  
我觉得我们就不谈标准，其实我们没必要，我们这些人其实很casual的讨论。我们不谈标准，我们也不想推标准，我们只是想讨论出来有没有一个合理的代表对吧？

金雪峰  
对，然后我刚才想说的是说这个东西太快了。我可能今天我们讨论完，可能明天又出来一个，这时候大家这个东西要不要遵守。然后如果不遵守大家可能自己参数自己弄完，然后结果是两家不一样，然后又放在一起又来argue一下。这是个问题，不是说我们愿不愿意做。

唐衫

我们不是把它作为标准，就是说我们不存在遵守不遵守的问题，我们是一个recommendation，我们是这些有做硬件的，有做软件的，大家认为反正在现阶段这个基本cover了多少多少，就形成一个反正我们讨论出来的一个东西，也没说谁要遵守，谁要不遵守。因为其实我们希望得到一个稍微稳定一点的答案，你后面加可能也没关系。但可能比如是隔一年我们再更新一下，这个没问题，我就是纯技术讨论，我们也不想说是推什么标准，那个没意义。

颜深根 03:42:54  
对，其实我觉得这个我比较赞同杨军的一个观点，就是这个事情我觉得他可能不是光技术的问题，它是一个商业的问题。而且我觉得说确实我们也没有必要就是说一定去形成一个算子的具体一个list的是吧？我觉得那个东西说实话一时半会也形成不了，但是其实我觉得我们可以做的事情就是说，有没有办法可以加快收敛的过程，是一个比较我们可以去想这个事情是吧？确实我我短期内我是肯定找不出这么一个list，但是我有可能可以做一些事，可以去加快list形成这个过程，也许最终形成的list不是由咱们来定义的，有可能是比如说某某一家大公司最后形成事实标准，它出了一个list，但是我们可能就是说在这个过程中有可能可以去出一些力，或者是做一些事情能够促成这个过程，我觉得这个可能是我们可以去做的一个事情，对。

说话人19 03:44:09  
如果说要找一个训练的就是要用到的算子的一个列表的话，其实是存在的，你把PyTorch或者跟TensorFlow的API列表拉下来，他就是一个事实标准。对，但这里面的问题是说，不管是PyTorch还是TensorFlow，他们的算子现在都是基于NV的CUDA生态来编程的。我们要的前面提到的，我在OP的基础之下给出一个新的primitive，然后所有算子这1000多个算子都能基于这些primitive来表示。

我观察到是说这里面其实是一个技术人力成本的问题。我首先假设存在这样一个 primitive，我设计出来了，我也需要把这1000多个算字在 primitive上全部重新实现一遍，这个里面的人力是没有任何一个独立的实体能够去commit的。你想PyTorch还是TensorFlow的话，都是社区这么多年这么多人往里面去贡献的，代码怎样还参差不齐。

说话人15 03:45:15  
我其实follow一下深根的建议，就是说有没有可能能够简化这个流程，让这件事情变得可能，对吧？我提一个例子就是DLPack，因为DLPack是天奇发起的，说我要让所有的框架之间的tensor的in-memory能够share，你可以讲effort也不小，right？但他为什么能够work，但至少目前很多人还是用的，我觉得有两点。第一点是它非常简单，我要加DLPack，我只要加两行，两个函数就结束了，这件事情就这样，cost非常小，这是第一。第二就是说它有很多议题是放到后面去decide。比如说DLPack现在开始去讨论说，怎么支持simple memory、stream【03：46：02】等等这些原来不cover的东西。就说他一开始的scope特别的小，这件事情就是让整个社区，当然他也是拉了当时的很多方向的创始人过来一起来adopt，但是大家觉得这个东西effort还好，可能花一个下午我也搞掉了，right？所以他就adopt，然后滚雪球就滚起来了，因为比如说我们现在去看DLPack生态里边，像Nvidia就很支持，他很多个package全部是接DLPack，这样它能够很方便的去replace一些component。然后生态起来以后，它反过来去push这些框架开发者必须去维护这个东西，没有办法，然后我再去说我在这上面再加新的功能，大家也更好接受一些。

蒋力 03:46:49  
所以这个有点像一开始给你免费使用是吧？成本上基本不要钱，粘性很高之后你得付出代价。

金雪峰

要不我继续其实还有一个我们讨论了一下，因为我们现在讨论的更多的是说编译器，或者说是op领域的接口标准，有没有可能在AI框架的接口做一个标准是吧？如果那个标准可以，那不就更一劳永逸了是吧？向下就更加通顺了，当然更难是吧？

当然我们也讲了一些路径。就是说比如说你numpy加grad，是一个最简洁的情况，因为jax就是这么做的，然后大家所有的其他的库就基于这个东西去往上扔。其实也是一个建议，当然我们大部分的人觉得，包括敏捷、欣然，包括岚松都认为就是说现有的框架的遗留系统的带来的问题比较多，或者说他这个包袱比较大，这个事情还是挺难推的。现在实际上我们AI框架的结构基本上都是PyTorch like的，就有点像PyTorch，但又不一样，绝对是这种。所以我觉得这也是一个一个问题。

然后下面刚才讲的就是在这个模型标准上，我们现在可能有两个大家认为是比较公共的，一个叫ONNX，第二个我理解MLIR也在推这个标准是吧？模型层面的标准，这一块反正我觉得如果能够统一是个好事情。因为我们最早MindSpore用的是ONNX，但是我们确实发现我刚才说的那个问题。我刚才不是说反对这个事情，而是说我说这是个挑战，非常大的挑战，因为ONNX变动太多了，最后ONNX不是大家想象的ONNX。就是你跟我讲，我给你一个ONNX，但是实际上你给的跟他给的是不一样的，这就是一个问题。对，我觉得后面确实如果说我们要定义模型的标准，是可以需要一方面技术上去讨论，第二个可能机制上也要讨论。因为现在像ONNX社区要提交一个更改，那是很漫长的事情是吧？大家要这个社区要讨论，这实际上不太work。

然后最后一个，现在提的训练框架未来是更动态还是更静态？然后有几派观点。可能从欣然的角度来讲，可能动态图从灵活性包括效率上会更高一点。然后我当时提的一个观点是说可能要考虑两个，一个是我们这个硬件厂商的生存空间的问题。如果真的全是动态图的话，其实DSA架构就发挥不出它的效率了。所以说我觉得这个问题可能最好分场景考虑，就是说我们一般可能可以把一个AI的这个过程分成研究阶段跟生产阶段。那研究阶段，你可能需要的是说更加的灵活，效率更高，你的模型变化的更快更多。另外一个就是生产阶段，生产阶段就是说你是要大规模生产，比如说像互联网厂商你的一些实时的这种推荐是吧？推荐、搜索，包括你的大量的这种语音的这种语音、语义分析的这种场景，你可能是搭了一个几万台的这种集群在跑，这时候你肯定对能效比包括性能是要求很高的，然后那个模型假定我们生产阶段的模型是能够收敛的，这两个场景就可以分开解决了是吧？我在研究阶段可以用你的动态图，性能低一点也没关系。然后你生产阶段可能等你研究阶段的模型基本上搞定了，到生产阶段部署的时候，你可以用性能更加高的动态图。同时就是说因为现在动态图主要是在V100上算，其实说白了未来GPU到了A100甚至更后的加了tensor core以后，其实它的性能调整也会非常大。

说话人1 03:51:38  
这个地方我补充一点啊我的一个想法，就说这里边提到硬件芯片欢迎静态图对吧？其实对于我们来说，我们之前有个实践的逐渐的认知。就说我们现在加速器不宜做的过大，就是说你加速器做本身是很大的，但是你里边的core不宜过大。我们一般加速器的话它都会有一些并行度，因为我们处理的tensor可能有的是做三维的，有的做二维的，有的也有一维的，就是向量计算。实际上就说每个维度，如果说我是三维的，处理的数据本身可能每一个维度都不一定正好把它凑满。比如说每个维度都用了90%，两个维度90%，一乘就成了81%，再一乘就70%多了，利用率也降得很快。所以我们就说第一点，加加速器，同样的计算能力我尽量拆成很多很小的core，然后大一点的shape我就可能用10个core，小的shape可能用3个core，anyway我就是把core一个个填满，最后的那个core可能填不满。基本上这样的话我们就觉得就是说，从硬件的计算效果来说，它多多少少能够把dynamic shape这个问题能够做得更友好一点。

蒋力 03:53:22  
由于时间关系，我们这个也到了尾声，今天其实还是意犹未尽，还有很多问题还可以值得继续讨论，那么这也是我们后面要去搞开放式的这种研讨会的动机之一。那么我们后面可能关于研讨会怎么去组织，我们可能更多还是在线下，我们可以在群里面稍微讨论一下就行了，没有必要今天把大家留在这里，非得把讨论的一清二楚。那么我们今天这个论坛就到这里结束了，然后我们最后就合影，然后大家纪念一下我们今天这次kick off的一个比较有标志性的事件，对吧？有可能将来会产生一个非常大的影响。