关于Tenstorrent的访谈思考

https://zhuanlan.zhihu.com/p/386400118

None Sun Aug, 08 23:36

ANAKDTECH在今年5月份对Tenstorrent的CEO Ljubisa Bajic和CTO Jim Keller做了一次非常细致的访谈，读完以后，有一些感悟，也记录在这里，聊作打卡。

1.关于Tenstorrent和Jim Keller的事迹介绍网上有很多，这里就不做搬运工了，接下来主要谈一些自己的思考。

2.Jim很强调在芯片设计过程中尽早考虑到软硬件协同设计的需要。在我看来，**这里的关键就是软硬件交互的边界定义**，这个边界定义通常很难一下子就达到比较好的平衡态。既然一步到位很难，那就通过多次迭代的方式来不断进行修正。这里也蕴含了敏捷开发和MVP的思想了。从一个hardware guy嘴里看到如此强调这个概念，确实让自己稍微有些surprise。也相应地能够感觉到Jim作为架构师的功力老到之处。

> make sure whatever you’re building always has software to run on it. There have been a number of AI efforts where they build a chip, and then they say ‘now we’re going to build the software’. They then find out the hardware/software interface is terrible, and it’s difficult to make that work. I think [Tenstorrent] did an interesting job on that.

稍微展开一些，为什么要尽早通过实际run software的方式来迭代软硬件接口？这实际上会增加一定的工作量。在硬件实际可以跑在FPGA上之前，想比较完备地验证软硬件接口通常需要在性能模型或RTL代码上完成，这会增加一定的开发工作量。想象一下如果想在C model上跑通一个BERT模型的前向后向流程？在device compiler还没有ready之前，仅仅是造出完整的test case可能就会稍微有些让人挠头了。在RTL代码上跑一个完整的ResNet50的mini-batch iteration？耗时之长可能会让硬件开发工程师怀疑这样做是否划算。

而如果问我的观点，我是举双手双脚支持Jim的观点。我甚至期望在硬件设计的第一天，软件的整体技术全栈的方案就已经有了一个雏形，这个软件技术栈的雏形会随着硬件设计的细化不断进行迭代演化（以及可能的大返工，如果发现软硬件接口存在严重的问题）。device compiler不ready怎么办？只要有一个基本的ISA定义，就可以通过手写汇编，或实现简易的汇编生成器的方式来为示例模型产生完整的测试代码（*如果能够把模型+AI框架+ISA + 硬件架构这几个不同层次的研发同学纠集在一起，其实针对主流模型，实现一个单设备的麻雀虽小，五脏俱全的完整神经网络训练的功能测试任务并不是非常困难，复杂性在于这里所需的技术点跨越了不同层次，需要把不同背景的同学统一在一起形成有效的配合协作*）。建立起这个软硬协同的体系以后，在硬件设计推进的过程中，ISA，C-model，上层软件栈都进行相应的变化调节，就可能形成一个紧凑的全栈迭代了。

3. Ljubisa提到了在2016年，他们在FPGA上build了一个E2E的原型系统，可以用来跑一个caffe的模型。这个E2E的原型系统的主要目的是验证他们的核心设计idea。

> **LB:**We had a box that we were taking around and showing everybody in June 2016. It was running neural networks end-to-end on Caffe on an FPGA and returning results, and so it wasn't anything that could sell but it showed that we could basically bring up this thing end-to-end relatively quickly and that we had a bunch of ideas. Ultimately it had all of our key focal points, even now, or essentially stuff we wanted from the get-go. It hasn't been a huge revolution in high-level thinking since then, it's been more just a massive amount of execution.

这也是一个我感兴趣的topic。如果是开发手机端芯片，在比较早的阶段在FPGA上跑通原型，验证想法，甚至打通电话我都不意外。如果是云端芯片，特别是大芯片，做到这一点就要challenging很多了。LB claim说Tensotorrent的核心idea和16年的FPGA原型系统表现出来的基本相同，这让我非常好奇。不过在后文，Tensotorrent提到了他们在架构设计上很关注架构层面的scalability（从tiny computing device到high throughput computing device），算是部分address了我的疑问。当然关键的细节并没有充分展开。

有一个原型系统在那里实际run的最大好处是能够比较快捷地提供基于silicon的反馈数据。比如ISA是不是需要做一些增改来适应软件编程的需要。AI框架层是不是有必要做一些图层面的处理，来简化底层优化的复杂性。workload的计算访存比是否满足预期，是否有必要增加不同层级存储介质的尺寸，或扩大其带宽。以及Tensor计算和非Tensor计算单元的资源配比是否需要进行必要的调整。这些硬件架构层面的magic number可以通过精细的analytic model和老到的架构直觉来给出，但我还是觉得如果能够花费不算非常大的代价就打造一个FPGA原型系统，提供基于实际运行的反馈数据回来，可以更有效地指导整体系统的开发。并且一定程度上，这种methodology也可以降低domain specific硬件开发的门坎。因为有一些经验可以通过实际的运行数据来弥补上了。这可能不是最typical的hardware architect的taste，带有了比较强的software guy的taste。**我映象中hardware guy比较期望“一次把事情做对做好”，而software guy会有更多的'tiral and error'的习惯。当然上层业务软件的开发同学这个倾向就更明显了：）**

4.19年 Tensotorrent完成了PoC硬件Jawbridge的流片，只包含**6个**Tensix cores，功耗为1.5W，使用的成熟工艺(>12nm)，紧接着在**2020**年完成了Grayskull的流片，包含**120个**Tensix cores，功耗为65W。间隔这么短的核心原因是用Jawbridge来进行整体研发流程的打磨。这又是小步快跑的研发思想的一个体现。这种小步快跑的方式，在我看来，能够把系统全栈打磨的周期缩短，比如Jawbridge的流片应该能够把不少硬件验证和后端物理实现的时间腾挪出来，用于全栈系统的打磨。整套系统打磨完毕以后，再推进Grayskull的研发。这还是相当考验主架构师的平衡能力，增加一个PoC迭代产品确实可以缩短反馈链路，但怎样避免这个PoC过于PoC就有讲究了。这还是一个trade-off拿捏的艺术。

LB: There were two reasons why we went about our product roadmap the way we did. One was purely risk management- it was a new team, no existing flows, no existing anything. We really had to flush the pipe and get something built so that all of the 50+ steps that you need to put in place to get it done can be done. We have it all working and we didn’t want to run a risk of sinking a pile of money in case there’s a mistake. The other motivation was that we believe pretty deeply that it’s important to have the same architecture basically span from edge to huge multi-chip, multi-computer deployments in the cloud. The main reason why we think you need an architecture that spans so widely is that as you go away from just running through a bunch of equations, to implementing the neural net, you get into more fancy things like runtime sparsity or dynamic computation or anything that tries to go away from the mindset.

Jim也补充了一个有些意思的观点，他再次emphasize了小步快跑的想法，还不客气地调侃了有些公司在硬件流片成功以后，要组建上千人的软件团队进行配套软件的落地支持（似乎有公司可以对号入座了^-^），因为缺失了靠前迭代环节软硬全栈的打磨适配，导致软硬接口的开发效率不够高效。

JK: You know, a lot of people spend a boatload of money on their first try. Grayskull (the chip shipping this year to customers) is our third-generation chip - we had that prototype in an FPGA as the first generation, and had the test chip as our second. The company has learned a shit-load on each step, both hardware and software. It’s reflected in the software stack, and our software team is pretty small.I’ve seen a lot of AI companies who have got a chip back and their plan has been that to make it work they need to hire 300 software people. That means they don't really have a plan. You can’t really overcome that mismatch in the hardware/software boundary with huge teams. Well at some level you could, if you can throw 1000 people at it, and some people are doing that. That’s not really the right way to do it, and that’s not going to be something you could expose long run to programmers, because the complexities are so high, it gets really fragile, and then the programmers can’t see how the hardware works. One of the key elements of Linux, as well as open-source software and running on x86, is that programmers could program the hardware right to the metal. They could see how it worked and it was obvious, it was robust, and it worked over time. For AI hardware that’s too fragile to be exposed to most programmers. Not all programmers, but you know, the ninjas.

5.关于目标客户的触达，我觉得JK他们的头脑还蛮清晰的。其核心观点浓缩一下就是，先不打大公司的重点核心业务，而是走农村包围城市的路线。核心业务的试错成本高，通常对于新硬件的adoption周期会比较长，所以能够看到不少新硬件公司也都在通过类似的路径进行业务的推广。

JK: I think there's another way to look at it. The three big areas of AI today are image processing, language processing and recommendation engines. Then there's also this movement from convolutional networks to transformer networks. Everybody's using the same building blocks. Then what we've noticed is a lot of people go and aim at the big hyperscalers, but the hyperscalers won’t deploy them until you can sell them a million parts. But they don't want to buy a million parts until they see a proof-point of 100,000. Those customers who want 100,000 want to see a 10,000-unit deployment. We've talked to a whole bunch of people, from the top to the bottom of the stack, and they're all interested. The ones that are easiest to talk to that are going to move the fastest, like AI start-ups, or research labs inside of big companies that own their own software. They understand their models and just get on with it. Our initial target isn't to get some huge contract, it is to get 100 programmers using our hardware, programming it, and living with it every day. That's where I want to get to in the short run, and that's basically our initial plan.But they are a fairly diverse set of people that we're talking to. There's the autonomous crowd, control systems, imaging, language. We're building this cool recommendation engine, which has an extra board and a computer with a huge amount of memory to make that model work. So it's fairly diverse, but we're looking for people who are agile, as opposed to any particular vertical.

6.LB和JK重点提到了在Tensotorrent的架构上进行分布式训练，对于上层模型开发同学会很友好，因为按照他们的说法，无论是数据并行，模型并行，流水并行，只要符合GPT-3这种transformer家族类的模型拓扑(Attention模型大行其道的说，之前团队同学的一个工作，为不同模型进行分布式策略的探索，遇到BERT，Transformer、GPT-3，T5这类模型这类相对规整的模型会更直观一些，反而是遇到了了类似GNMT，或是AmebaNet这类不太规整的模型探索其分布式策略会更讲究一些，而Transformer类模型目前的popularity反而简化了策略探索的复杂性)，都可以将自动分布式的复杂性hide在他们提供的compiler背后。这一点，其实我觉得即便是在NV GPU的硬件体系里也同样能够做到，Tensotorrent claim的优势多大程度上是其软件栈带来的，多大程度是硬件架构带来的，目前还不是很有判断。不过有一点我倒是buy-in。那就是NV GPU上在不同层次引入的同步交互机制确实存在差异，这就带来了一定的软件开发的异构性，这种异构性的复杂性管控往往需要更多的研发精力投入。**某种程度上，Tensotorrent确实是在软硬全栈地探索如何更有效支持自动分布式的需求。而在我看来，想达到比较理想的支持自动分布式的AI训练需求，需要硬件，硬件厂商的软件栈，AI框架后端，AI建模前端(类似于JAX/TF PyTorch建模API这一层)的协同设计。任何一层处理不当，都可能给整套全栈系统引入额外的复杂性。**

IC: Jim has been quoted as saying that the Tenstorrent design was 'The Most Promising Architecture Out There'. Can you shed some light on what this meant at the time Jim initially invested? Was that more about the initial Grayskull inference design, or was there vision into the Wormhole processor (next generation)?

JK: I would start with our flow. We have a compiler stack that starts with PyTorch generating a graph, and then the graph gets parallelized into smaller chunks. Those chunks have to coordinate between their computation, and then they execute kernels. That's the basic flow that everybody is doing. What we have is a really nice hardware abstraction layer between the hardware and the software that makes all that work, and that's what I really like.

[In the world of AI], if you make a really big matrix multiplier, the problem is that the bigger it gets, the less power efficient it gets because you have to move data farther. So you don't want the engine to be too big, you want it to be small enough to be efficient, but you don't want it to be too small, because then the chunk of data it's working on isn't that interesting. So we picked a pretty good size of what we call our Tensix processor. The processor is pretty programmable. It's really good at doing computation locally in memory, and then forwarding the data from Compute Engine to Compute Engine, in a way that's not software disastrous.

I've seen people say that they have a DMA engine, and you write all this code for it, but then they just end up spending their whole life debugging corner cases. [At Tenstorrent] we have a really nice abstraction in the hardware that says (i) do compute, (ii) when you need to send data to put the data in the pipe, (iii) when you push it in the pipe, the other end pulls it out of the pipe. It's very clean. That has resulted in a fairly small software team, and software you can actually understand.

So when I say that it is promising, it is because they have got a whole bunch of things right. The compute engines are the right size, it natively does matrix multiply and convolution (rather than writing for threads), and it natively knows how to communicate data around. It's very good at keeping all the data on-chip. So we're much more efficient on memory - we don't need HBM to go fast!

Then when we hook up multiple chips, the communication on a single chip compared to the communication from chip to chip is not any different at the software layer. So while the physical transport of data is different on-chip with a NOC versus off-chip with Ethernet, the hardware is built so that it looks like it is just sending data from one engine to another engine - the software doesn't care. The only thing that does is the compiler, which knows the latency and bandwidth is a little different between the two.

But again, the abstraction layers are built properly, which means you don't have to have three different software stacks. If you write on a GPU, you have to be a CUDA programmer in the thread, then you have to coordinate within the streaming multiprocessor, then coordinate on the chip, then you have NVLink which is a different thing, and then you have the network which is a different thing. There are many different software layers in that model, and if you have 1000 people, or if that's what you think is fun, that's cool.

But if you just want to write AI software, you don't want to know about all those different layers. We have a plan that actually will work.
We're also doing some other interesting things. We are adding general-purpose compute as part of the graph and future products, and we're looking at how to make it programmer-friendly. We’re also asking programmers want they want to do. I've done a lot of projects where you build the hardware, and then the software guys don’t like it because that's not what they wanted. We are trying to meet the software guys where they are at, because they just want to write code. They also want to understand the hardware so they can drive it, but they don't want to be tortured by it.

7.LB和JK提到了Tensotorrent的融资节奏。前三代硬件系统(FPGA原型+Jawbridge + Grayskull)的研发花掉了**4000万美金**，然后基于前三代硬件系统的成果，在今年5月份进一步完成了**2亿美金**的融资。感觉其融资节奏也有些小步快跑的味道:)。**4000万美金完成三款硬件迭代，支持了从16年到21年近五年时间的运作，资金使用效率感觉还是比较高的样子。基于Hotchips 20的材料，20年Tensotorrent还只有70人，这确实是一个比较精干的小团队。**

8.JK提到了他作为senior tech leader，对技术细节的关注，让自己映象深刻。需**要的时候，要能够get to the bottom of anything**。哪怕是对于senior的技术管理者和架构师，都需要具备这种素质能力。比较怕的是天天在画流程图，准备slides，而缺失了对代码实施那一环的hands-on的感知，最后就可能看起来一切都好，做出来差很大火候。比如讨论起软硬接口是否合理，最直接的验证方法之一就是操起键盘，基于现有的ISA写一段汇编码，体感一下完成一个Conv + BN + ReLU的典型fusion算子在计算访存配比，寄存器和片上缓存的尺寸&带宽，Tensor计算和非Tensor计算的交互细节方面是否存在进一步改进空间。

JK: Yeah, I don’t want to go into too many details, but when I was at Intel, people were surprised a Senior VP was grilling people on how they manage their batch queue, or what the PDK is, or what the qualification steps were, or how long the CAD flows took to run, or what the dense utilization was in the CPU, or how the register file was built. I know details about everything, and you know I care about them, I actually really care. I want them to look good.

A friend of mine said, if it doesn't look good, it isn’t good. So when you look at the layout of a processor, or how something is built, it's got to be great. You can't make it great if you don't know what you're doing. It’s bad if somebody gives you like a five-level abstraction PowerPoint slide that said, things are going good, and we're improving things by 3% per year. I've been in meetings where somebody said 5% better, and I said ‘better than what? Better than last year, or better than the best possible, or better than your first-grade project?’. They literally didn't know what - they'd been putting 5% better on the slide for so long, they forgot what it was. So yeah, you have to get the details.

If you're going to be in the technology business in any way, shape, or form, and you have to get to the details. I thought I was good at that, and then I met Elon Musk. Holy crap. For him the details started at atoms. Maybe lower, I don't know. But like, what I thought was first principle thinking wasn't close to his first principle thinking, and then I got my ass kicked seriously about doing that. But it was really great, I really like to do that. I hope that when I engage with engineering teams, they start to get that engineering is fun, and the details matter, and there's an abstraction stack - there's the high level, there's the medium, and there's a low level. Yes, you do need to know a lot about all of them, because then you can figure out how to fix things. You can't fix something simple like ‘computer too slow’ - what are you going to do if it's too slow? If you could go into 1000 details, there's all kinds of stuff to do if you know the details.

IC: I’m surprised that people would wonder why you’re asking detailed questions about, register file and cache use and such, because this is stuff that you’ve been doing for so long. It kind of seems weird to me, it’s almost as if the person you were speaking to didn’t know who you were?

JK: Yeah, but the positions get associated with technical level. My team at Intel was 10,000 people, right, so you spend a lot of time on organization charts and budgets, and all kinds of admin, and then it's easy to find yourself just doing that and saying you’ll hand off leadership for a project to this person or this person and that. But the problem is there's so many things that are cross-functional – I want to know what the fab is doing, how does the PDK work, how does the library work, how does the IP team work, how does the SoC integration work, how does the performance model work, how does the software work. Then you find out that if you can't deep dive into all those pieces, bad things happen.

My father worked in GE when Jack Welch ran GE, and he said Jack could go to any part of GE and within a day and get to the bottom of it. He got credited with a whole bunch of business innovation, but I heard from many people that Jack could get to the bottom of anything. I read his book Straight From the Gut years ago, and I thought, well, I'd like to be that kind of person you know - if I'm going to manage people, I'd like to be the kind of person to get the bottom of anything.

总的来说，这是一篇质量很高的访谈。

发布于 07-04

文章被以下专栏收录