**Triton CPU Meeting-20240418\_130844-Meeting Recording**

April 18, 2024, 6:08PM

52m 38s

 **Video-TX-AN4-C105-MTR-6** 0:03  
So we started recording.  
Hello everyone this is the second community meeting for the discussion.  
I've tried this with you back end.  
The recording can notes will be available on the selection on offer today.  
We have one topic of just a recap of the intermediate things that happened between these two meetings we met with.  
Yeah, from Microsoft discussed other approach.  
We tried to shared ohh.  
Also, Lucas and maybe presented some drawbacks that they think they're talking share project has and Lucas, can you please share the those slides on the channel or I can share better for you to share and and also we met also we met with burnt from my time discussed and a little bit of design principles and also but mentioned that matter kindly agreed to host.

 **Lukas Sommer** 0:50  
Sure.

 **Video-TX-AN4-C105-MTR-6** 1:05  
In uh, I thought labs, which we are eternally grateful for that for that.  
So there are no other options of questions.  
That's give Stage 2, Leon.  
Alright, so don't.  
This is just one topic to discuss the actually and the multiple topics and I have, I hope some decisions to make.  
So let's try to recap our water.  
What we decided to keep also not goals and I think.  
One of them is to have a portable solution and easy to use and very easy to use.  
I understand that people can run the existing Triton kernels on CPU.  
They don't need to adopt them, modify them in any way.  
Uh, it's uh, OK to not be able to provide the maximum performance, only existing code result parameters thing because people are likely to have those parameters change for GPU.  
So our goal is to achieve performance at least similar to watching Doctor currently has four CPU use with its.  
Opening P generated loops.  
And we target multiple architecture, so it's not overly excited.  
So as snow goal is to using these big end as a debugger, so don't try to make it as a debugging feature and also non goal is to try to lower everything to allow the MTR and it's OK at least it's Chung Cheng to use some external canals for example for what will make sense.  
So if someone if some months ago someone goals or so.  
Comments.  
Skype.  
You speak up?

 **Bert** 3:24  
I I wanna I clarify on the external solutions for for Jem I'm kind of assuming that would mean we just like let Jim be handled in a different level, not like we try to integrate MKL and to try it in or something like that.  
I I think that would be like not the right direction if but in the short term, if people wanna fast jam, they can just call it A10 gem which calls MKL.

 **Video-TX-AN4-C105-MTR-6** 3:52  
Yes, that's basically it's just external kernel call to vendor library or touch library, whatever.  
So the first topic I'd like to discuss.

 **Tiotto, Ettore** 4:07  
So, so sorry.  
Sorry, I have a question on the gem.  
I mean, we still need to be able to lower the gem.  
So what the solution?  
I don't understand your.  
The solution is to lower gem to a loop with FEMA operation and not use.  
You know, first library that vendor may have to do a gem operation.

 **Video-TX-AN4-C105-MTR-6** 4:30  
The solution is to lower gem to call function call.

 **Tiotto, Ettore** 4:34  
Right.  
So in those function call the the yeah.  
So there will be a external library that can implement the general functionality that so that's the way to get performance right.

 **Bert** 4:43  
Umm.  
That that wasn't that wasn't actually my understanding.

 **Video-TX-AN4-C105-MTR-6** 4:46  
Yes.

 **Bert** 4:48  
Like if you called TL dot, I think we should generate code for that and not try to integrate external libraries.

 **Video-TX-AN4-C105-MTR-6** 4:59  
Wait.

 **Tiotto, Ettore** 5:00  
Yeah.  
So there is a disconnect.  
That's why I asked the question.  
And why is that that that, you know, like we don't wanna use external library for efficient gem implementation and we want to default to just a serial loop that does FMLA, which is obviously not gonna be very efficient.

 **Bert** 5:21  
Yeah.  
If so, this is from the Pytorch perspective.  
If someone is writing a model in Pytorch, they already have access to a fast gem via via MKL.

 **Tiotto, Ettore** 5:32  
Umm hello right?

 **Bert** 5:33  
They can call torch dot M and they're gonna get a fast gym.

 **Tiotto, Ettore** 5:35  
But I mean this project is me.  
Yes, by torching as very use a very important user of Triton.  
But when you design A Triton CPU back, can you also have to think about that?  
You know, you may have any users may not go through Pytorch, right?  
So then if it doesn't go through by torch but torch may have an efficient implement may already not use Triton for that use case and it's fine, it's Pytorch prerogative to do that.  
But what about everybody else?  
Everybody.  
Potential users try it and they're not gonna be happy with the sequential loop.

 **Philippe Tillet** 6:12  
I can.  
I can speak to that because I guess I'm the one who proposed this non goal and I think the the purpose here is really to reduce the scope of the work of the CPU back end.  
It's not to say that like we wouldn't like a fast gym.  
It's more like it's like considered less important than then being like very efficient on memory bound kernels.

 **Tiotto, Ettore** 6:26  
Yeah.

 **Philippe Tillet** 6:35  
Umm, so I think that that's that's not like the perspective I'm taking like, yeah.

 **Tiotto, Ettore** 6:39  
So we'll we'll feel what you're saying is like as I initially implementation say some time and just generate an FMLA loop and then we need to think about what is the fast path is gonna be.

 **Philippe Tillet** 6:47  
Yeah.

 **Tiotto, Ettore** 6:52  
Umm yeah, I agree.  
As I I I think at the end there will be a need to to a efficient German implementation on the CPU as well.  
But I have three that is not gonna be.

 **Philippe Tillet** 7:01  
Yes.

 **Tiotto, Ettore** 7:03  
It doesn't have to be that the the very first goal, the I think that.

 **Philippe Tillet** 7:08  
It's also like, yeah, it's very complicated.  
I feel like it's a big project and it's own and until you actually match and Carl, it's not super useful.  
Umm so whereas like bandwidth bound kernels would be like very useful very quickly I think.

 **Video-TX-AN4-C105-MTR-6** 7:26  
Don't feel quick.  
Question then is it OK if there is a?  
I don't know compiler config or parameters that we say OK lower these two and KL calls.

 **Philippe Tillet** 7:39  
But I don't understand how we can do that, right?  
We we won't.  
We don't wanna do NKL inside of Triton inner loop.

 **Video-TX-AN4-C105-MTR-6** 7:46  
And maybe some render ankle or arm or whatever they will say.  
OK.  
We'll we'll combine that and there was a key that we lower it to the library call.

 **Bert** 8:01  
If you take like the sorry, Phil, I'll, I'll let you answer.

 **Philippe Tillet** 8:05  
Well, yeah, I think in this case I'm not sure what the benefit would be over just like calling MKL for the whole Mathml.

 **Video-TX-AN4-C105-MTR-6** 8:15  
The the camera was not changed.  
He was still called TL DOT and is the same kernel the people.

 **Philippe Tillet** 8:23  
But but, but he won't be as fast as you.  
Won't be as fast as an MKL anyway that.

 **Video-TX-AN4-C105-MTR-6** 8:27  
No, no, that instead of generating a.  
The compiler parameter says lower the deal door to until call.

 **Li, Jian Hui** 8:35  
So there's a the abstraction tried to defined is you first load to to tensor and that tensor is mapping actually to who register kind of thing and then you do TO dot so so if you use library then everything you go to memory again and.  
And.  
It's, I mean, the Triton language already break down into he allowed and thought, right.

 **Video-TX-AN4-C105-MTR-6** 8:59  
Not necessarily.

 **Philippe Tillet** 9:05  
I yeah, I think, yeah, I'm wondering it for Eric was suggesting was more like calling bliss kind of like micro kernels.

 **Video-TX-AN4-C105-MTR-6** 9:06  
Ohh.

 **Philippe Tillet** 9:13  
But I I feel like even in that case you wouldn't be end up being competitive for entire gym with the with like an option, but more.

 **Video-TX-AN4-C105-MTR-6** 9:20  
OK, OK.  
I think we can like defer this to the future.  
Like if we have better solution in the future which is performant and more important and more general, we can do that right?  
I guess there is no disconnect here, right?  
Feel better.

 **Tiotto, Ettore** 9:34  
So I I guess the main concern is that we don't want to carry a bunch of user vendor library inside the you know GitHub rappel.  
Umm, because that's gonna pull that.

 **Video-TX-AN4-C105-MTR-6** 9:44  
Right.

 **Tiotto, Ettore** 9:45  
I don't know.  
Pull up the laptop or repository.  
Essentially, that's the that's your main concern for you other than you know that the that having a loop and then just doing smaller jams may not actually give you performance as a school does.  
If you can just offload everything to a library.

 **Philippe Tillet** 10:02  
Ohh that that's one concerned, but I think marginally.  
I'm trying to think of use cases where people would actually want to do that over, like just using a more optimized Mathml or flash tension for CPU.  
I feel like we we end up kind of in a worst of both worlds situation or like the kernel as a whole is not like optimal as one would want.  
And it's also like there's a lot of complexity in there that that creeps in the Triton code base as a result.

 **Tiotto, Ettore** 10:34  
OK.  
It's a fair concern.

 **Video-TX-AN4-C105-MTR-6** 10:38  
OK, let's move on.  
Now I will capture that and let me know.  
OK, the first topic I'd like to discuss is how we actually achieve her listening to how we split data between channels so well, obviously we need to utilize CPU cores and here we can.  
Ohh use the funds that we run our kernels basically.  
So we run multiple instances of kernels and it can easily be mapped to threads.  
So we can like say each kernel instance runs in its own thread.  
For example, we can do if he's opening the task.  
Ohh number of like such tasks.  
Measure depends on data.  
It can be it can exceed number.  
Of course we have, because it would provide better load balance, but for small data takes surely can be small.  
The number, of course, just to two small tasks.  
When we have overhead exceeding actual useful lot, so and within the kernel we need to tell. I seen differently.  
So, uh, the data blog that is processed by kernel needs to be dialed and this style is defined by duration.  
And how do I capabilities?  
It can be one or more of vector native vector registers.  
And here's actually quite important question of each style size which choose and which blocks sizes which choose.  
So in my opinion, uh bloked tile my block size, style size and vector size are three independent things.  
So we can say that Tau size should be equal to the vector or that efficient block size can be a tile always.  
So umm, I can show a simple example of this vector add even we have a dialogue 16 elements which is basically 1 vector and we run only task these different number of elements included in the single task and you can see that it's too small tiles.  
We just have two, five ahead for those tasks and at some point we don't have enough release and.  
So here we have actually multiple options.  
One of them is to say that.  
Parameters that are tuned by users, so block size that is provided to kernel is an efficient block size.  
It means that this inefficient amount of data that should be processed by single a thread does.  
Uh and other ways to say that both sides provided by user and tuned is an efficient dial sets and then we use outside opening be loops and use opening piece scheduling to like group those tasks into chunks and thus get efficiency through opening.  
So I believe that the first approach is more efficient because uh, in some cases we just cannot.  
Right.  
Choose journal block size.  
Uh to be efficient as a tile so as well as example is a those kernels that user reduction, for example softmax or things like that that that's the schedules have computation on tensors and they've then they have reductions and it means that 2 split into the smaller blocks a user would need to modify kernels to introduce loops or uh partial reduction.  
And if we say that user provides parameter for efficient log size or and we tuned for that then compiler inside the kernel can introduce loops and efficient outsets and so decisions here actually affects how our calculation pipeline goes.  
Which transformations we need and which that looks delicious.  
So I assume that.  
Doing it for block size is the best to achieve the maximum performance uh tuning for tiles size is probably easiest option but will not let us achieve.  
But before once in all cases, at least without users to rewriting their channels.  
So I said kind of designed document draft where I wrote a lot of Renaults in vector and the possible usage in completion pipeline and those things come from uh assumption that our kernel process.  
He cannot block and be processed by single thread.  
So I need thoughts here.

 **Bert** 16:56  
I think I'm still.  
I mean, I know we were having this argument on Slack, but I think I'm still a little bit confused about this notion of an like a ideal block size being larger than the register set, and I'm starting from like the, you know, the Triton vector ad kernel.  
If you set the blocksize there to be something that fits in registers and then you wrap that in OMP 4 loop word like, it doesn't have to be so and Pete, you can just write your own thread pool.  
You can you can saturate the memory bandwidth with that kernel, so I don't think we, yeah.

 **Video-TX-AN4-C105-MTR-6** 17:34  
Well.  
You can do it only if you use a proper, uh scheduling for opening the kernels, because for example, if you say that each iteration can be run in any thread, then basically you have a special man.  
Each task processes just one KB of memory, and if one core plus this one KB from here, then one KB from there and start jumps in memory, it basically random access, so it's not going to be efficient if you split a whole loop evenly between cores then it basically the same as having a bit a big block size within the kernel.  
So it's basically just have a vectorized loop.  
That has continuous access to memory.  
So probably the actor at case is just too simple and it doesn't matter.  
We says that general person blog or tile, it just doesn't matter.  
So I think.

 **Hongtao Yu** 18:46  
So.

 **Bert** 18:47  
OK.

 **Hongtao Yu** 18:50  
Can you go back to your previous picture like I?

 **Video-TX-AN4-C105-MTR-6** 18:57  
This one.

 **Hongtao Yu** 18:59  
The that the histogram, the one with the?  
Yeah, this one.  
So why there's Blues after the green one?  
So how?  
How many stories does those blocks as corresponding to?  
I'm wondering, like if you look at the tail right there so.

 **Video-TX-AN4-C105-MTR-6** 19:18  
Why?  
Why?  
Why do we have to tell?  
Because but for example for 1,000,000 block size we just have 4 threads utilized.

 **Hongtao Yu** 19:29  
Falls to rise and what was their city?

 **Video-TX-AN4-C105-MTR-6** 19:29  
So yeah.

 **Hongtao Yu** 19:31  
How many harvest rise there on your CPU?

 **Video-TX-AN4-C105-MTR-6** 19:34  
I guess it's like.  
I guess it was 64 facts.

 **Hongtao Yu** 19:38  
But.  
So the green bar corresponds to just four softwares rights.

 **Video-TX-AN4-C105-MTR-6** 19:49  
Bring part is just the best result.  
That could be achieved on different block sizes, but actually you can see that like between 4128 K that there is no much big so uh, the difference is just the situation might be it says that if fish size like 1000 is efficient enough maybe 4.

 **Hongtao Yu** 20:10  
I see.  
Yeah, I was wondering like a how many is the rights to the way created like in total?  
Like should we just created the same number of threads?  
Highest threats like.

 **Video-TX-AN4-C105-MTR-6** 20:34  
Well, in doctor.  
For example, sets number fuse thread to the number of physical cores.

 **Hongtao Yu** 20:44  
Umm.

 **Video-TX-AN4-C105-MTR-6** 20:45  
But I think by default opinion P probably uses a number of logical cores.

 **Hongtao Yu** 20:53  
Yes.  
So if it's, that's the right number.  
In that way, the block size is gonna be fixed right? So.

 **Video-TX-AN4-C105-MTR-6** 21:00  
Shall we use a schedule pragma so block size for task is not fixed?  
So if you don't specify it, then default opening P, uh, told to open up.  
I guess static to split between all course and the default one is on the right, so it's not the best one, probably because a load balance is not the best.

 **Hongtao Yu** 21:31  
I see.

 **Philippe Tillet** 21:34  
There's something I'm still confused about, similar to Bird I guess, which so to me seems like the reference Triton kernel should.

 **Hongtao Yu** 21:34  
There's some.

 **Philippe Tillet** 21:43  
So I think the right way to do a vector ID then on GPUs nowadays it's just with the persistent kernels where you have an outer loop and you launched a fixed number of thread blocks and it seems to me that like the CPU code you presenting, correct me if I'm wrong, but he's basically the same thing where you have like a like a block size of 16 and then just a persistent outer loop.  
That iterates between like across the entire vector with a fixed number of program ID like 4 or 8 or however many threads you wanna parallelize over.

 **Video-TX-AN4-C105-MTR-6** 22:24  
Yes, but I could instead of using the scheduled static with a number of blocks, I would like to put in this that I could instead increase the internal loop, but it's basically the same so.

 **Philippe Tillet** 22:42  
Yes, but but.  
But I mean what I mean is, is something the the user could do as well?  
I I know it's is the.  
It's a conversation that was in Slack already.  
Umm, but like to?  
To me it seems that like the this is just like using a block size of 16 with a persistent kernels and a grid size of N and if you use N threads, unless I'm like misunderstanding the.

 **Video-TX-AN4-C105-MTR-6** 23:07  
Well, but it says if you if you say that great size is the number of threads then.

 **Philippe Tillet** 23:13  
No.

 **Video-TX-AN4-C105-MTR-6** 23:16  
Uh here?  
Great size is actually depends actually depends on input size and block size right.

 **Philippe Tillet** 23:25  
Not necessarily like even on GPU's.  
Like if you have persistent kernels, you can just and I think that's the preferred way to write kernels anyway.  
It's it's too launch with a constant grid size that on GPU's with the number of number of SMS or like a multiple of the number of SMS on the on CPUs you could view it as like a multiple of the number of cores and then you would just like the user would manually write the outer loop instead of the kernel to make sure that all the data is processed with the small grid size that that's also how how like I've been writing a lot of my GPU kernels.

 **Video-TX-AN4-C105-MTR-6** 24:00  
So does it mean that it implies additional loop in the kernel?  
Reason why user.

 **Philippe Tillet** 24:07  
Yeah, in that case the the the user would have to write kernel differently, but I think this way to write the kernel differently would be beneficial for both CPU and GPU.

 **Video-TX-AN4-C105-MTR-6** 24:20  
Well, yeah.  
If we control this block size and like number of and greed as independently through different parameters that basically in terms I provided here to be independent control of tile and blog by different parameters.  
Of course, if we if kernels are written in such ways that this can be controlled by users independently.  
The Zen basically.  
Who says that user responsible for errors?  
But it gives a responsible for the optimizations.

 **Philippe Tillet** 25:00  
Yes, but but I think that's also how it should be on GPU's.

 **Video-TX-AN4-C105-MTR-6** 25:01  
Yeah.

 **Philippe Tillet** 25:03  
That that's not how.  
Like your typical GPU programs are written, but I I think on GPU's you also wanna fix lunch like a fixed number of of thread blocks and and just like have a an outer loop in each in each kernel that processes all the data.  
Because for example, if you have a reduction, you don't want to carry like a big partial state that depend on like your input size, you want to keep like a partial reductions local and update them in an outer loop.  
Umm so I I think in general.  
It's the preferable way to to do things, so it's it's reasonable to ask the user to do it.

 **Video-TX-AN4-C105-MTR-6** 25:43  
Well, actually it is.  
Action is one of cases that because we should like to say that blocks and tiles are different and uh, users don't always want to be responsible for them.  
For example, if you have a soft Max or like few thousand elements, then it implies computations on those elements, glass reduction and.  
On CPU, efficient reduction cannot be done without dialing on several thousands of elements.  
You cannot, but you don't have such registered file.

 **Philippe Tillet** 26:22  
But even on GPU you don't want to do that.  
I think on GPU's you wanna have an outer loop like on GPU's.  
If you understand like a million element, you'll want to launch like let's say like 200 thread blocks and it's Fed block is gonna have an outer loop and process.  
Umm, I don't like 50,000 elements or something and then at the end of the outer loop they're going to have like a partial reduction state and then you can either use atomics to synchronize between all the thread blocks and get your final reduction on the last one.  
Or you can launch a second kernel, but to to me seems similar to what you do like and the CPI.  
Guess what I mean is that some point on the GPU also become like register limited.  
I cannot have like a 64K block size on on GPU either.  
So so I have to find ways to to make reductions work without.

 **Video-TX-AN4-C105-MTR-6** 27:19  
Well, what?  
What CPU?  
It would be much smaller looks as it can be processed right without partial reduction and the way it's on right now.

 **Philippe Tillet** 27:26  
Yeah.

 **Video-TX-AN4-C105-MTR-6** 27:28  
If you look into tutorials and try it and if you look into kernels that I generated by inductor then suggest well have to be rewritten to be efficiently run CPU.  
He says that users are responsible for Tyler.

 **Philippe Tillet** 27:44  
And maybe the problem is that the tutorial were not very good.

 **Video-TX-AN4-C105-MTR-6** 27:50  
And the doctor also.

 **Philippe Tillet** 27:53  
Yeah, I can work on like improving the tutorials or as someone to to help with this, but well I guess what I mean is like doing like a A1A1 million elements.  
Reduction in the city with the block size of 16 K is just as complicated as doing 64 K element reduction of CPU with the block size of 128 or something.  
To me it seems like a very similar problem that you run out of registers and so you need to have an outer look at some point.

 **Li, Jian Hui** 28:30  
So actually.

 **Bert** 28:30  
You can.  
Good.

 **Li, Jian Hui** 28:32  
So yeah, I I think yeah.  
So actually there's a the the programming model you you mentioned a field that's like user fully know at the programming time how many cores is gonna to exclusively use the runtime.  
And I I think that's a good for people who want to write high efficiency code.  
Uh, and then if this is the most use case Triton CPU want to target the four, I think it's great because now we have the like a similar thing like on the CPU side and the CPU side because people need to know how many cores up front and.

 **Philippe Tillet** 29:13  
That's it.  
Sorry to interrupt, but I don't think you need to know exactly how many cars I think you you you can just launch with like a fixed grid size and tune the the grid size so you can still run auto tuning to make sure it adjusts to the amount.  
Of course that you have.

 **Li, Jian Hui** 29:31  
I see.  
So, so, so, so basically UM, the grid size can be tunable also like uh, but the the program model is like assume that was will give me a great size and then I just cut the like divide the task like that.

 **Philippe Tillet** 29:44  
Yeah.  
Yeah, I yeah, I think I I can work on improving the tutorials or maybe ask someone to to help because I'm quite busy.  
But yeah, I think it seems like tutorials could at least show this way of writing things, which I think is better anyway.

 **Li, Jian Hui** 30:03  
Yeah.  
If if it's like that, I think it should work with the OP runtime, because OP also like user needed to, uh, set.  
Like how many cores they use and then.

 **Philippe Tillet** 30:17  
No, it's very similar, yeah.

 **Li, Jian Hui** 30:18  
Yeah.

 **Bert** 30:25  
It just one comment.  
I wanted to add on like inductors use of like loops versus like persistent.  
I mean, this is he can actually generate both depending on the input tile sizes or sorry input tensor sizes and so like you know CPU obviously being like a different back end like it'll have different heuristics.  
But I think it conceptually fits within the same framework.  
You know, do you admit a loop versus do you do everything in registers like this?  
This is this is just a tunable heuristic based on the hardware.

 **Video-TX-AN4-C105-MTR-6** 31:04  
OK.  
Well, I think this implies that they actually says that I want to see you again.  
Will not be able to efficiently run like journals that are currently in tutorials like and kernels like that.  
So they will.

 **Philippe Tillet** 31:29  
Yeah, we could defy the tutorials.  
And I I think to be fair, I think it's the right future proof way to do things you know, not not just for like GPUs and CPUs but like if you think at all the custom hardware backends forth, right on at some point all of those have like a fixed number of cores and don't have a like a good hardware scheduler to dispatch work on to those.  
Umm, so it's actually a recurring concern that I've heard, heard from people who are coming and MTI order like custom accelerator is OK like it's much easier if there's like a fixed grid size and the kernel is persistent.

 **Video-TX-AN4-C105-MTR-6** 32:12  
Uh, well, OK.  
So yeah, let's move on.  
So they're required transformations.  
So ohh that you like critical transformations to get on for months on CPU and I think everyone agrees that one of them is.  
That's why we should move for memory access, because we have to get rid of indirect memory accesses.  
Uh.  
Point out contiguity and transform.  
Bless her story into continuous flows in store and probably.  
Optimize our mask and you as well, but it's probably less important.  
I don't know.  
So no, there are currently existing facilities in Tritons that allows to determine contiguous successes.  
So we can try to to like those and as a result of transformations we can use existing work pointers.  
So probably switch to memory if, uh, well, I don't see so far I couldn't see much use of memories because for example it doesn't allow us to express mask, float and stores.  
So it seems that doesn't help there, so we just cannot use memory flows and stores anyway.  
So probably try to dialect, but we slow down, must float and store separations.  
Is good enough and we should like use it.  
But please work pointers.  
Also, I guess that other begins can benefit from using GLOW points assistant of tensor of pointers for example.  
I think pintail degree back end can benefit using 2D models and prefetches.  
So the next one is Pelican fusion.  
So if we say that compile is not responsible for splitting belongs to trials and give it to users, then we basically can place that it's not required and it might simplify things significantly and reduce number of dialects you would like to use.  
Umm, but I think Theorization stays and.  
I think we should be feel explicitation done in our malaria plane, but of course we can see that we can lower to allow VM and then run limitation it's optimizer.  
But I guess we don't have warranties that God is going to be optimized.  
So I would say that which is issue in Malayalis before.  
So for memory access, well I guess I already told.  
Or since this slide, any comments here positions?

 **Tiotto, Ettore** 35:32  
From experience.  
Memory access analysis becomes difficult in the presence of of of the alias pointers.  
So do we have any guarantee in Triton that the TT pointers, if you have two T pointers passed to kernel are actually not alias to each other?  
I mean they they they point to distinct memory or does the back end it was soon that they may point to this to overlap in memory locations.

 **Video-TX-AN4-C105-MTR-6** 35:56  
I guess.  
Like, I guess lesson is very important if we are trying to perform dialing, but if it's done by users then why should we care?  
It's just indirectly access versus continuous success.

 **Philippe Tillet** 36:21  
I think it's so so far like setting the memory, the memory semantics, right?  
So for example we have this problem on Andrew Fuse and I mean it's a very good question and I don't have like a perfect answer to this.  
What I can say is today the the GPU back end setup assumes that all input tensors, all input pointers don't alias with one another.

 **Tiotto, Ettore** 36:45  
Yeah. OK.

 **Philippe Tillet** 36:45  
Umm, that makes everything you're simpler.  
Umm.  
Now someone may still write kernels for which this is not true, and this is where we get in in like, dicey territory.  
Is that like if you know how the compiler works you you can sort of like make that could work.

 **Tiotto, Ettore** 37:00  
Right, right.  
Especially if you wanna you wanna.  
I mean, yeah.  
So if, well, that's an interesting assumption that Triton makes because it's what did the user do wrong to pass two pointer that overlap that the point of memory that overlap that there is?  
If this is not specified in the language in the Triton language, then that's entirely valid.  
You use case from the user.  
The compiler's gonna crap.  
It's going to he.  
Yeah, I something need to be done.  
Probably at the language level tool to clarify the semantics of or pointers.

 **Philippe Tillet** 37:38  
Yeah, it's an ongoing discussion.  
I think that the main issue, I mean it's as you said, it's extremely complicated to do LS analysis right?

 **Tiotto, Ettore** 37:47  
Yeah.

 **Philippe Tillet** 37:48  
And and if we assume by default that things, uh can alias, then we have to set like memory fences everywhere essentially.

 **Tiotto, Ettore** 37:58  
Yeah.  
Or you need to you to check our runtime, generate the runtime code, the check that these two pointers do not overlap, but then you also need.  
All the extent of the memory that the pointer access to because if you if you don't know then how long is that you know pointer point to a memory location other pointer points to a number maybe location.  
You also need the size of of that of the two memory location to prove that they actually don't overlap in memory and you need to generate code in the compiler to do manifest that check and potentially clone your kernel and say, oh this kernel here, this assumption is true around time I can optimize it more aggressively than the other one.  
So yeah, it's all kind of complications to the to the compiler.

 **Philippe Tillet** 38:46  
Yeah.  
I think specifically the assumption that threaten Maxine.  
Is maybe more like the pointers that are written to and the pointers are red from do not alias, but you can have aliasing between multiple pointers that are red from for example.  
Umm but if, but the pointers that are returned to do not allow us with anything else, I think that's more like and this is.

 **Tiotto, Ettore** 39:09  
But if you if you are, if you only have read through pointer no problems, but as as as when you have a right then you have issues.

 **Philippe Tillet** 39:13  
Yeah, yeah.

 **Tiotto, Ettore** 39:17  
Which?

 **Philippe Tillet** 39:18  
Yeah.  
So so I think my answer is probably fine.  
If the CPU back end makes the same assumption, if that simplifies things on, I think in practice also like torsion.  
Doctor does make sure that this is true or is aware of this at least.  
Umm yeah.

 **Tiotto, Ettore** 39:37  
OK.  
So we will, I guess you know, did the assumption here is the pointer will be noted?  
Yes.  
For the time being and one thing Clash later on, we may say to a user are using it incorrectly or or then essentially punt this problem from because I'm on one happen very often.

 **Philippe Tillet** 39:56  
Yeah.  
Yeah.  
I mean, it's supreme.  
We also have for for GPU's and we should try to fix it all at once.

 **Video-TX-AN4-C105-MTR-6** 40:17  
OK.  
Words? Umm.  
So the next topic for me was telling infusion and let's keep it for now. Hello.  
And expose about dialects to use so.  
The truth here is to like, stay on Triton dialect and probably introduce some on Google.  
The relationship we need them specifically for CPU.  
Umm there is a option to use some of the same dialects, but actually I was looking into a lineage, dialect and member I have for example from the perspective of transformations like.  
Like tiling first place, so if it's out of the question then I see no use fully knowledge here, but I guess Victor doesn't can become our target dialect at least intermediately.  
This in our logging to will be Amaya right?  
So since we want to produce after code, the extra direct provides, uh, nice Victor abstractions that can use that we can use in our way.  
So instead of directly lower but try it and operations into little bit MYR, we can them into vector dialect and then use vector dialect facilities to lower it and all of them.  
Yeah.

 **Tiotto, Ettore** 42:00  
So so you you put linear algebra, linear algebra Talic there, and we heard from is there any appetite should be avoided a linear algebra dialect or should we umm you?  
Because essentially this called the awfully will be upstream to the open AI repository or some point.  
So is there any drawbacks and try to use the linear algebra toilet or should we avoid it and just try to use more lower level die little to start with?

 **Video-TX-AN4-C105-MTR-6** 42:32  
Well.  
No, Windows dialect itself is not very useful.  
It's what useful is transformations and lower rings that can be applied to them.  
You know, and if we to not apply those transformations, then there is no use for the dialect itself. So.

 **Tiotto, Ettore** 42:56  
I thought we were talking about that we should be tiling moves, so that's in the linear algebra dial to help you with that.

 **Video-TX-AN4-C105-MTR-6** 43:01  
Yes.  
Thing just based on minutes ago.  
We said that users are going to be responsible for that.

 **Tiotto, Ettore** 43:09  
Well.

 **Philippe Tillet** 43:13  
Yeah, I think we'll, I played the tutorials because I think even for GPU's you wanna tile the loops.

 **Tiotto, Ettore** 43:13  
Umm.  
OK.  
Yeah, so prevent.  
So we don't have the prototype.

 **Philippe Tillet** 43:25  
Yeah, yeah.

 **Video-TX-AN4-C105-MTR-6** 43:25  
Yeah, problem.  
I don't see why want to use, you know.

 **Ian Bearman** 43:33  
Wouldn't.  
So the user can't be tiling operations with each other, so user couldn't tile, for example a Mathml with an ad or something like that.  
We would need the compiler to do that.

 **Philippe Tillet** 43:44  
And but but the user can can write the Colonel in the way that it does all the work and the the kernel doesn't run out of registers.

 **Ian Bearman** 43:57  
I think you're assuming that the machine is symmetrical.

 **Philippe Tillet** 43:57  
I think.

 **Ian Bearman** 44:00  
We may have a case where someone needs to create a subview, for example of a tensor, and that's not possible in Triton.  
It's only possible inside the compiler.

 **Philippe Tillet** 44:09  
But it's not possible and the GPU either.

 **Ian Bearman** 44:15  
OK, it's possible.  
I don't.  
We're talking about CPU function though.

 **Philippe Tillet** 44:21  
I mean for, for, for, for Triton.  
So so I think the the goal here is just to make sure that we don't run out of registers for the block size that are right for for CPU's and and to me this seems like the very similar problem because GPUs can also run out of registers, they just do at a different scale.  
Umm and usually the the the way this surprise for GPUs is is yes, but by having like a persistent outlook in the in the kernel, so you can keep your your block size small but still do all the all the work that you need to do.

 **Li, Jian Hui** 44:59  
Of of a city?

 **Philippe Tillet** 44:59  
Not really.

 **Li, Jian Hui** 45:00  
Yeah, actually for CPU user needed to tile at least A2 level more because the size of registers are also because the sometimes you don't, you don't have tile MMA, so it's it's just like a vector vector FMA.

 **Ian Bearman** 45:16  
You might also want to tile loops for cache and things like that as well.

 **Li, Jian Hui** 45:22  
I assume that that health or cache, that's that's something the code GPU code also do, but the tiling for smaller registers that that is extra thing.  
So it's if we push all the burden to user then it's certainly user that's too much.

 **Philippe Tillet** 45:39  
Well, I'd like to see examples of like.  
Yeah, I guess we're we're close worry that cannot be reasoned as a persistent kernels and work for GPU's but wouldn't be able to be lowered well for for CPU.  
And I think that would help me understand exactly what the problem is.  
I think another consideration here is because we're not trying to optimize for Maximals yet.  
Yeah, I think that that simplifies a lot of a lot of things.

 **Video-TX-AN4-C105-MTR-6** 46:18  
OK, we have 5 minutes left and I wanted to kind of fix first one if Dean.  
So Philip, you mentioned that you would prefer.  
And like one region, Red Bull.  
But I think no opinion P would be a better solution because it was all I was, compressibility of release on different levels and compressibility these external libraries.  
So any objections here?

 **Philippe Tillet** 46:52  
I'm not completely opposed to open MP Umm.  
I just think, yeah, if it's necessary, we have to use it.  
I suspect in a lot of cases threadpool might be sufficient, but it's.  
I mean, it's pretty minor.  
I think if if the tiling is done by but by the user, I think the open MPR Open MP runtime should be like really lightweight, right?  
It's just like when we actually dispatch the the program ID they would need it I think.

 **Video-TX-AN4-C105-MTR-6** 47:25  
Yeah, most probably.  
Yeah, maybe.

 **Philippe Tillet** 47:27  
Yeah, I think it's it's, it's fine to to to use it for now and and see like later.  
Like if we can do without, it's possible we can't.  
I'm not familiar enough with it.

 **Video-TX-AN4-C105-MTR-6** 47:41  
Or ends the last one is about external dependencies.  
We would need some.  
We need a runtime.  
It's includes like my functions for which we will need a vector finance which for example the ML and provide uh we need gym at least as external kernel.  
Probably some other kernels as well, and now we have like quite hot libraries for GPU stored within Rep and I guess this is another good option for CPS.  
Of course, we would probably want to use different libraries for different vendors and storing a lot of quite quoted would be too much so.

 **Philippe Tillet** 48:32  
I think for for math functions we also have this thing with GPU's right?  
It's like NVIDIA depend on Lib device and they also have their their live libraries.  
I think it's OK.  
I think in general the the way we've addressed this is umm by letting user link against external libraries rather than than packaging them so so users can say oh I I wanna use I have like I4 declared this function with this signature and then I link against this library and this will this will resolve the dependency.  
We have a very small set of core math function like X2 umm look to these kind of things that if we can have like portable lowering for those that that would be preferable.  
But I I don't know enough like how how doable that is.

 **Video-TX-AN4-C105-MTR-6** 49:27  
Good.  
That's all from my side.  
Umm side question to both actually.  
Is it possible to ask?  
I don't know extract 810 or maybe something else?  
Open glass or something like 12.  
This portable layer between consumers and producers.

 **Bert** 49:48  
Uh, sorry, not sure I followed the question exactly.

 **Video-TX-AN4-C105-MTR-6** 49:54  
Yeah, you.  
You were following up the fields question, right?  
So there is a intermediate level library that can hide implementation but be kind of central dispatch for all malfunctions.

 **Bert** 50:11  
Ohh.  
Uh, yeah, I mean, so we we kind of have that in, in Pytorch at least for like certain vector operations like there's this AT VEC abstraction.  
But I I'm just gonna go out on a limb and say we probably don't want to depend on Lib.  
Torch it's.  
It's like a really nasty dependency.  
It makes everything like kind of complicated.

 **Video-TX-AN4-C105-MTR-6** 50:31  
Ohh definitely does the torch, but is there any appetite to extract it outside?

 **Bert** 50:39  
No, that, that's that's just a huge, huge engineering project that we just don't have that we don't have the, the people that do it, the interest.

 **Video-TX-AN4-C105-MTR-6** 50:51  
Could.

 **Bert** 50:53  
Yeah, sorry.

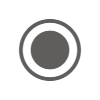
 **Video-TX-AN4-C105-MTR-6** 50:55  
No, it's good.  
Maybe we can use the another?

 **Bert** 51:00  
I mean like what we use internally for vectorized math functions is like some combination of sleeve or if it's like larger vectors, we use VML but but yeah.

 **Video-TX-AN4-C105-MTR-6** 51:13  
Sounds good. So.  
But anything else on this presentation?  
No.  
So can we.  
Can we recap the decisions?  
Yeah.  
Well, I think our decision to checked in assumptions that journal subregion where users in the search ways that we don't need additional dialing and allow optimizations, it mostly comes to vectorization both existing loops.  
And parameter tuning.  
I think that's it and see how it works.  
OK.

 **Bert** 52:05  
Cool.

 **Video-TX-AN4-C105-MTR-6** 52:07  
OK.  
We're on top of the other.  
Why don't time if there's, you know, any other question then?  
Umm.  
The guest next steps.  
Do.  
Yeah, everyone else who is interested will catch up and connect on the channel and start start implementation from prototyping.  
OK, sounds good.  
Last time you lost me talking questions.

 **Melik-Adamyan, Areg** stopped transcription