01:22:18 SECTION 2: Lets make it fast. GPUs, mixed precision, 1000ms

this point we have the gpt2 model we have some confidence that it's correctly implemented we've initialized it properly and we have a data loader that's iterating through data batches and we can train so now comes the fun part I'd like us to speed up the training by a lot so we're getting our money's worth with respect to the hardware that we are uh using here and uh we're going to speed up the training by quite a bit uh now you always want to start with what Hardware do you have what does it offer and are you fully utilizing it so in my case if we go to Nvidia SMI we can see that I have eight gpus and each one of those gpus is an a100 sxm 80 gb so this is the GPU that I have available to me in this box now when I look when I use um to spin up these kinds of Boxes by the way my favorite place to go to is Lambda Labs um they do sponsor my development and that of my projects uh but I this is my favorite place to go and this is where you can spin up one of these machines and you pay per hour and it's very very simple so I like to spin them up and then connect vsod to it and that's how I develop now when we look at the A1 100s that are available here a100 80 GB sxm is the um GPU that I have here and we have a bunch of numbers here for um how many calculations you can expect out of this GPU so when I come over here and I break in right after here so python trity so I'm breaking in right after we calculate the loit and laws and the interesting thing I'd like you to note is when I do lit. dtype this prints a torch. FL 32 so by default iny torch when you create tensors um and this is the case for all the activations and for the parameters of the network and so on by default everything is in float 32 that means that every single number activation or weight and so on is using a float representation that has 32 bits and uh that's actually quite a bit of memory and it turns out empirically that for deep learning as a computational workload this is way too much and deep learning and the training of these networks can tolerate significantly lower precisions um not all computational workflows can tolerate small Precision so for example um if we go back to to the data sheet you'll see that actually these gpus support up to fp64 and this is quite useful I understand for a lot of um scientific Computing applications and there really need this uh but we don't need that much Precision for deep learning training So currently we are here fp32 and with this code as it is right now we expect to get at at most 19.5 Tera flops of performance that means we're doing 19.5 trillion operations floating Point operations so this is floating Point multiply add most um most likely and so these are the floating Point operations uh now notice that if we are willing to go down in Precision so tf32 is a lower Precision format we're going to see in a second you can actually get an 8X Improvement here and if you're willing to go down to float 16 or B float 16 you can actually get time 16x performance all the way to 312 Tera flops you see here that Nvidia likes to site numbers that have an asterisk here this asterisk uh says with sparsity uh but we are not going to be using sparsity in R code and I don't know that this is very widely used in the industry right now so most people look at this number here uh without sparcity and you'll notice that we could have got even more here but this is int 8 and int 8 is used for inference not for training uh because int 8 has a um it basically has um uniform spacing um and uh we actually require a float so that we get a better match to the uh normal distributions that occur during training of neural networks where both activations and weights are distributed as a normal distribution and so uh floating points are really important to to match that uh representation so we're not typically using int 8 uh for training but we are using it for inference and if we bring down the Precision we can get a lot more Terra flops out of the tensor course available in the gpus we'll talk about that in a second but in addition to that if all of these numbers have fewer bits of representation it's going to be much easier to move them around and that's where we start to get into the memory bandwidth and the memory of the model so not only do we have a finite capacity of the number of bits that our GPU can store but in addition to that there's a speed with which you can access this memory um and you have a certain memory bandwidth it's a very precious resource and in fact many of the deep learning uh work workloads for training are memory bound and what that means is actually that the tensor cores that do all these extremely fast multiplications most of the time they're waiting around they're idle um because we can't feed them with data fast enough we can't load the data fast enough from memory so typical utilizations of your Hardware if you're getting 60% uh utilization you're actually doing extremely well um so half of the time in a well-tuned application your tensor cores are not doing multiplies because the data is not available so the memory bandwidth here is extremely important as well and if we come down in the Precision for all the floats all the numbers weights and activations suddenly require less memory so we can store more and we can access it faster so everything speeds up and it's amazing and now let's reap the benefits of it um and let's first look at the tensor float 32