NVIDIA H100, GeForce RTX4090
機械学習ベンチマーク報告書

文 書 番 号：	CP00021
版　　数：	第 1 版
作　成　者：	HPC システムズ株式会社
	HPC 事業部
	技術グループ
最終更新日：	2022/12/23

HPC システムズ株式会社
HPC 事業部

目次

1. 概要

本書では、弊社が実施した、NVIDIA H100 及び GeForce RTX4090 の機械学習ベンチマークについて報告します。 前世代のアーキテクチャ Ampere の時の製造プロセス 8nm から、4nm となり、飛躍的にコア数、クロックが向上し、理論性能は前世代のものと比べ、2 倍以上に伸びました。前世代と比較して実効性能を明らかにするべく、ベンチマークを行いました。

2. ベンチマーク環境

今回ベンチマークに使用したマシンは次表のとおりです。

マシン型番	HPC5000-ERMGPU8R4S
CPU	AMD EPYC 7742（2.25GHz 64core）x2
メモリ規格	256GB DDR4-3200 ECC RDIMM（16GB x16）
メモリチャンネル数	16（マシンあたり）
PCIe バス	Gen 4（x16 で 31.51GB/s）

今回使用した NVIDIA H100、NVIDA A100、GeForce RTX4090、GeForce RTX3090 のスペックは次表の通りです。

GPU 型番	NVIDIA H100-PCIE	NVIDIA A100-PCIE	GeForce RTX4090	GeForce RTX 3090
アーキテクチャ	Hopper	Ampere	Ada Lavelace	Ampere
GPU ベースクロック	990 MHz	765 MHz		
GPU Boost 時クロック	1755 MHz	1410 MHz	2520 MHz	1695 MHz
CUDA コア数	14592	6912	16384	10496
TensorCore 数	456	432	512	328
メモリ仕様	HBM2e	HBM2e	GDDR6X	GDDR6X
メモリインタフェース	5120 bit	5120 bit	384 bit	384 bit
メモリ帯域	2000 GB/sec	1935 GB/sec	1008 GB/sec	936 GB/sec
メモリ容量	80 GB	80 GB	24 GB	24 GB
最大消費電力	350 W	300 W	450 W	350 W
FP64 理論性能	48 TFLOPS	9.7 TFLOPS		
FP32 理論性能	48 TFLOPS	19.5 TFLOPS	82.6 TFLOPS	35.6 TFLOPS
FP16 理論性能	96 TFLOPS	78 TFLOPS	82.6 TFLOPS	35.6 TFLOPS
TensorCore FP64 理論性能	48 TFLOPS	19.5 TFLOPS		
TensorCore FP16 理論性能 （スパース性機能）	800 TFLOPS (1600 TFLOPS)	312 TFLOPS （624 TFLOPS）	330.3 TFLOPS (660.6 TFLOPS)	142 TFLOPS (284 TFLOPS)
TensorCore TF32 理論性能 （スパース性機能）	400 TFLOPS (800 TFLOPS)	156 TFLOPS （312 TFLOS）	82.6 TFLOPS (165.2TFLOPS)	35.6 TFLOPS (71 TFLOPS)
TensorCore FP8 理論性能 （スパース性機能）	1600 TFLOPS (3200 TFLOPS)		660 TFLOPS (1321.2 TFLOPS)	

ベンチマークデータと比較しやすいように、理論性能の比を表にしました。理論性能上は前世代の 2 倍以上の性能となります。

		H100 / A100	RTX4090 / RTX3090
FP16	通常時	2.46	2.32
	TensorCore 適応時	2.56	2.32
FP32 および TF32	通常時	2.46	2.32
	TensorCore 適応時	2.56	2.32

3. CNN ベンチマーク結果

3.1. ソフトウェア環境

Convolutional Neural Network (CNN) の学習ベンチマークには、NVIDIA GPU Cloud (以下、NGC) よりダウンロードした TensorFlow の Docker イメージを使用しました。nvcr.io/nvidia/tensorflow:22.09-tf1-py3 をフレームワークとして使用しました。ベンチマークには、同イメージに同梱の nvidia-example/cnn を用いました。

Example にある各種モデルを、精度を変更しながら、NVIDIA H100、NVIDIA A100、RTX4090、RTX3090 のそれぞれについて学習計算を行い、学習速度を測定しました。測定に用いた、それぞれのモデルの batch size を次表に示します。batch size は GPU メモリの許す限りの上限で設定しています。

表 1　NVIDIA H100、NVIDIA A100 に対して、各モデルで用いた batch size

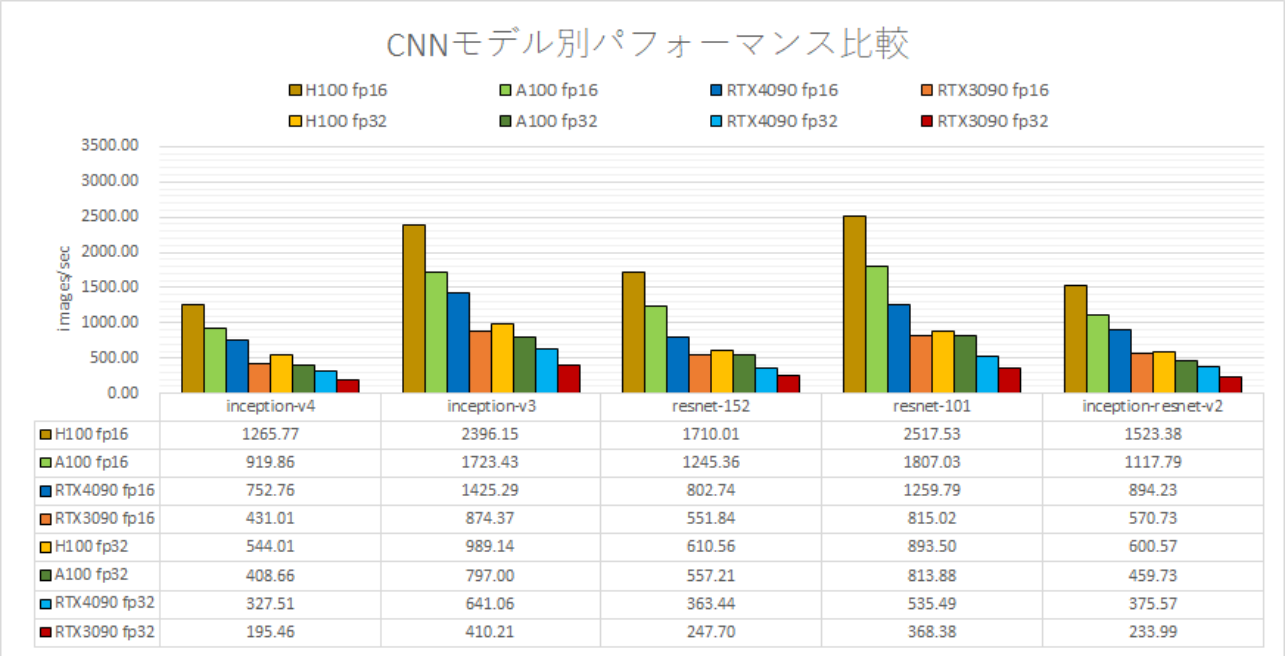
	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
fp32	400	640	350	550	425
fp16	800	1280	700	1100	850

表 2　Geforce RTX4090、Geforce RTX3090 に対して、各モデルで用いた batch size

	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
fp32	100	200	100	150	128
fp16	200	400	200	300	256

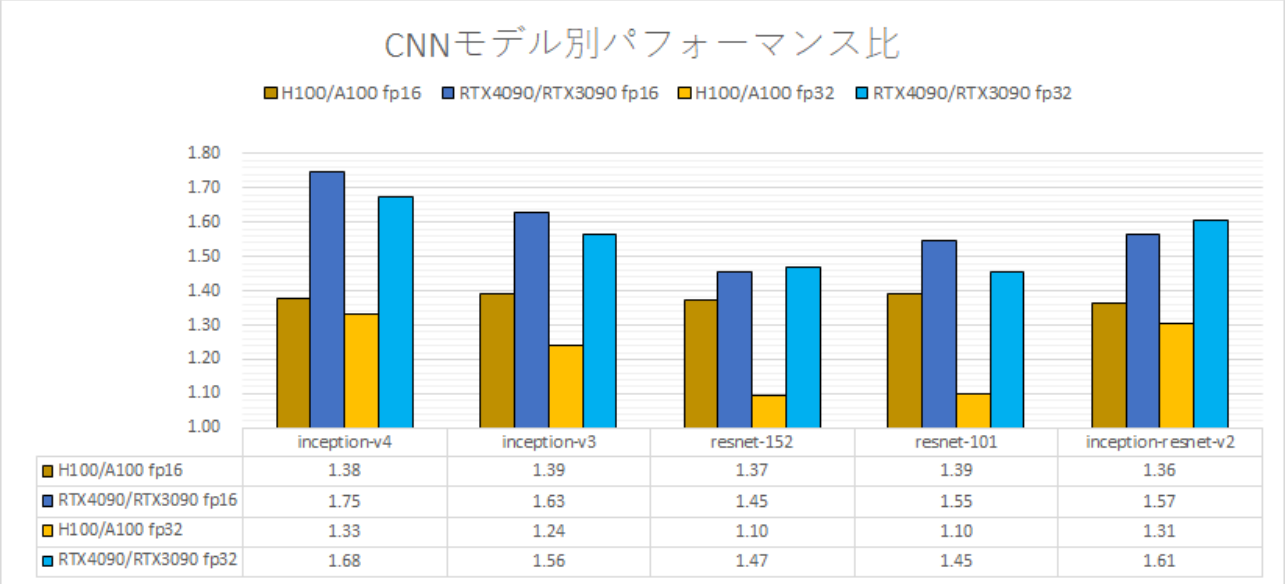
3.2. CNN 学習性能の比較

様々なモデルで、精度を変更しながら、NVIDIA H100、NVIDIA A100、RTX4090、RTX3090 について、学習速度をベンチマーク取得した結果を次図に示します。



	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
H100 fp16	1265.77	2396.15	1710.01	2517.53	1523.38
A100 fp16	919.86	1723.43	1245.36	1807.03	1117.79
RTX4090 fp16	752.76	1425.29	802.74	1259.79	894.23
RTX3090 fp16	431.01	874.37	551.84	815.02	570.73
H100 fp32	544.01	989.14	610.56	893.50	600.57
A100 fp32	408.66	797.00	557.21	813.88	459.73
RTX4090 fp32	327.51	641.06	363.44	535.49	375.57
RTX3090 fp32	195.46	410.21	247.70	368.38	233.99

H100 と A100、RTX4090 と RTX3090 を比較しやすくするためパフォーマンス比で表示すると、次図となります。



	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
H100/A100 fp16	1.38	1.39	1.37	1.39	1.36
RTX4090/RTX3090 fp16	1.75	1.63	1.45	1.55	1.57
H100/A100 fp32	1.33	1.24	1.10	1.10	1.31
RTX4090/RTX3090 fp32	1.68	1.56	1.47	1.45	1.61

RTX4090 と RTX3090 の比較では、半精度で 1.45 倍～1.75 倍、単精度で 1.45～1.68 倍となりました。H100 と A100 との比較では、半精度で 1.36～1.39 倍、単精度で 1.10 倍～1.33 倍となりました。H100 の、Resnet の単精度性能のみが大きく低いですが、同程度の学習の中、Resnet だけが低いことを鑑みると、ハードウェア起因というよりもソフトウェア起因と考える方が自然と思われます。今後の CUDA や CuDNN の更新により改善を期待したいです。

全体的には、実効性能が理論性能よりも低く留まる結果となりました。この理由は、ソフトウェア起因、または、メモリ量不足によるバッチサイズ不足であると考えています。これについては、少々込み入った説明が必要になりますため、詳しくは付録を参照ください。

4. BERT ベンチマーク結果

4.1. ソフトウェア環境

NVIDIA 社の Deep Learning Examples から、Pytorch の BERT の pretraining の速度をベンチマーク取得しました。ソフトウェア環境は NVIDIA の GitHub ページを参考にしました。ページは下記です。

https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/BERT

ベンチマークに使用した Docker イメージは、nvcr.io/nvidia/pytorch:22.09-py3 を使用しました。pretraining のパラメータは、次表のとおりとしました。

表 3 NVIDIA H100、NVIDIA A100 に対して、 BERT pretraining で指定したパラメータ

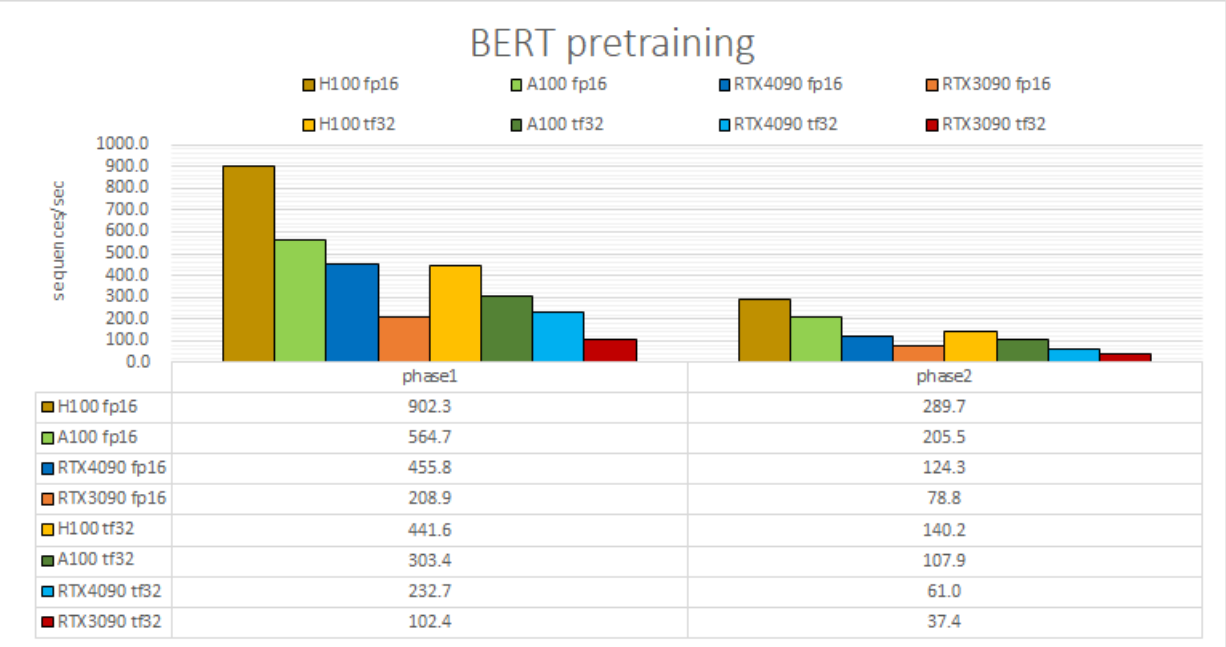
	Phase1			Phase2		
	batch size	gradient accumulation step	actual batch size	batch size	gradient accumulation step	actual Batch size
fp32	128	256	32768	16	1024	16384
fp16	256	128	32768	32	512	16384

表 4 Geforce RTX4090、Geforce RTX3090 に対して、BERT pretraining で指定したパラメータ

	Phase1			Phase2		
	sub-batch size	gradient accumulation step	Batch size	sub-batch size	gradient accumulation step	Batch size
fp32	32	1024	32768	4	4096	16384
fp16	64	512	32768	8	2048	16384

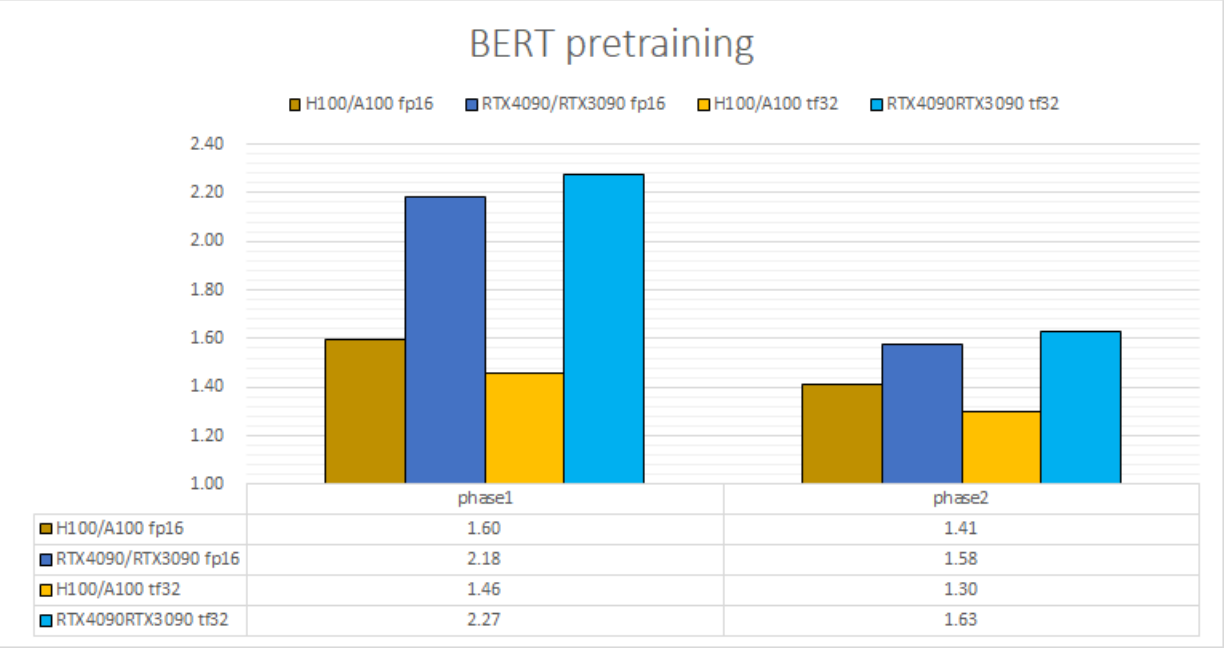
4.2. pretraining 性能の比較

BERT の pretraining を、精度を変更しながら、NVIDIA H100、NVIDIA A100、RTX4090、RTX3090 についてベンチマーク取得しました。pretraining 速度を比較したグラフを次に示します。



	phase1	phase2
H100 fp16	902.3	289.7
A100 fp16	564.7	205.5
RTX4090 fp16	455.8	124.3
RTX3090 fp16	208.9	78.8
H100 tf32	441.6	140.2
A100 tf32	303.4	107.9
RTX4090 tf32	232.7	61.0
RTX3090 tf32	102.4	37.4

H100 と A100、RTX4090 と RTX3090 を比較しやすくするためパフォーマンス比で表示したものを次図に示します。



	phase1	phase2
H100/A100 fp16	1.60	1.41
RTX4090/RTX3090 fp16	2.18	1.58
H100/A100 tf32	1.46	1.30
RTX4090RTX3090 tf32	2.27	1.63

RTX4090 と RTX3090 の速度比は phase1 の半精度においては、ほぼ理論性能通りとなりました。NVIDIA H100 と A100 の速度比は、1.3～1.6 倍、RTX4090 と RTX3090 の phase2 の速度比は、約 1.6 倍となりました。

phase2 では phase1 と比べて全体的に速度向上率が低い結果となりました。phase2 は、phase1 と比べて、インプットの文の長さが 4 倍になります。バッチサイズを 1/4 にしているので、実質的には、同じインプット量で、モデルが 4 倍の大きさになったことに近いです[1]。計算内容については phase1、phase2 は同種の処理なので、ソフトウェア起因というよりもハードウェア起因と考えられます。ハードウェアで原因となりそうなのは、メモリ帯域幅の可能性が高いです。H100 と A100、RTX4090 と RTX3090 は、メモリ帯域幅は、ほとんど向上してないためです。

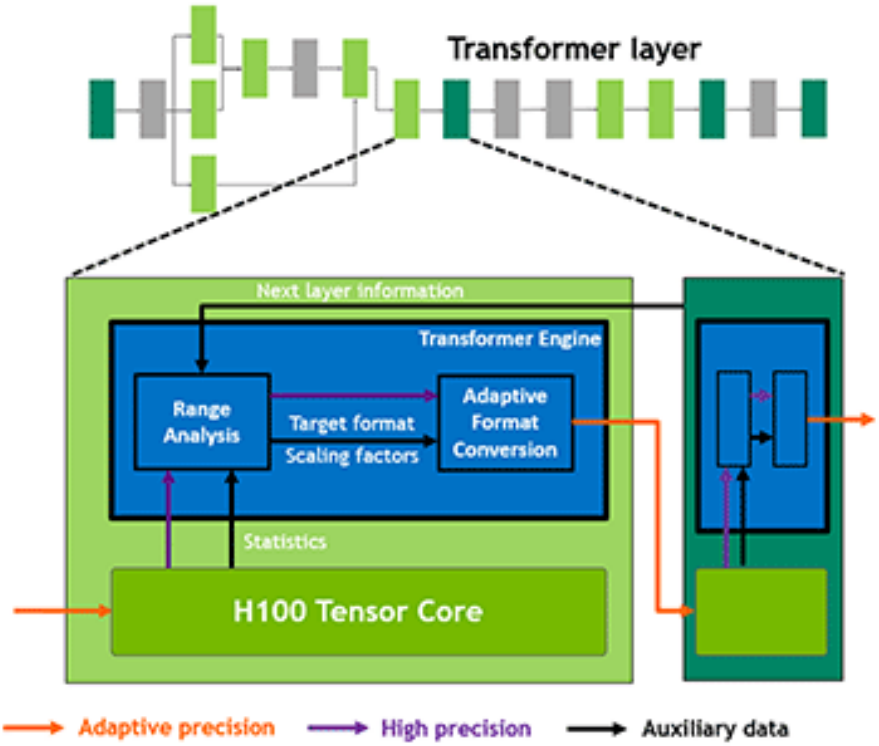
また、H100 の速度が、理論性能を大きく下回る理由としては、メモリ帯域幅の他に、ホストメモリと GPU メモリ間の帯域幅も影響しているのではないかと考えています。以前、A100 と V100 の比較を行った際、PCIe バスの Generation が Gen4 であるか Gen3 であるかが、速度に影響を与えました。A100 と H100 の理論演算性能差は 2 倍以上なので、phase2 の pretraining の速度に影響が出てもおかしくないレベルと考えています。

[1] https://arxiv.org/abs/1810.04805

5. Transformer Engine

NVIDIA H100 には Tensor Core とソフトウェアを組み合わせて Transformer Engine というソフトウェアエンジンが搭載されています。Transformer Engine は、自然言語処理の Transformer の演算を、FP16 よりも低い精度 FP8 に落として高速に計算するためのものです。従来 FP32 から FP16 のような低い精度で演算する際、低い精度の浮動小数点数で表現できる数値範囲からはみ出てしまうことが問題となっていました。この問題に対する今までの NVIDIA 社のアプローチは、学習のイタレーション毎に、特定の層に対して、loss scale という数値を乗算し、演算後、除算する、勾配の算出時に、桁溢れを起こしたら、loss scale を増やし、そのイタレーションはモデル更新をスキップするという方法でした。

今回の Transformer Engine では、演算する層の仕様を分析して、演算前に分析で導き出した数値(scaling factor)を乗算し、演算後に除算します。さすがに FP8 まで精度を落とすとイタレーション毎の処理では整合性が取れないようです。



5.1. Transformer Engine の試行

4 章で使用した BERT のプログラムに Transformer Engine を適用したベンチマーク結果を報告したかったのですが、結論から言うと速度は向上しませんでした。現在、Transformer Engine は、Pytorch のみにソフトウェアが提供されています。Transformer Engine の適用方法は、まず Transformer Engine の python モジュールをインストールし、適用したい python プログラムに、その python モジュールをインポートします。さらに、適用したい層に対して、Pytorch のクラスを、Transformer Engine と置き換える必要があります。Transformer Engine が対応しているクラスは下記のもののみです。

- Linear
- LayerNorm
- LayerNormLinear
- LayerNormMLP
- TransformerLayer

Linear、LayerNorm は、Pytorch の存在するクラスに対応しています。LayerNormLinear、LayerNormMLP は、Pytorch に無いですが、LayerNorm の後に、それぞれ Linear、MLP を適用する層ということです。TransformerLayer は、Pytorch の Transformer に対応していると思われます。

これらを実際の python プログラムに適用するのですが、現段階で下記の問題があることが判明しました。

① Pytorch の JIT コンパイルに対応していない。

Pytorch は JIT コンパイル機能により、繰り返し呼び出す部分に対して都度コンパイルが走ることを回避できます。しかし、Transformer Engine は JIT コンパイルに対応していません。

② LayerNorm の仕様が、Pytorch の古い仕様となっている。

Pytorch の LayerNorm は、Transformer や BERT が出てくる以前から存在していました。Transformer や BERT が注目を集めた頃、論文の Layer Normarization と、Pytorch の LayerNorm の仕様が異なるため、Pytorch で適応するためには、独自にクラスを作成する必要がありました。それは後に改善したのですが、Transformer Engine のマニュアルを見る限りでは、古い仕様の LayerNorm となっていて、そのまま適応することは難しいことが分かりました。

LayerNorm 以外の部分を、JIT コンパイルを適用せずにベンチマーク実行することはできましたが、散々な結果でした。以上の理由から、今回 Transformer Engine のベンチマークの報告は割愛しました。今後、ソフトウェアの改善を期待しています。

6. まとめ

NVIDIA H100、GeForce RTX4090 をそれぞれ前世代の NVIDIA A100、GeForce RTX3090 とベンチマーク比較をしました。製造プロセスが 4nm になり、理論性能が前世代の 2 倍を超えましたが、ベンチマーク結果では、BERT pretraining phase1 で RTX4090 が対 RTX3090 で 2 倍以上の性能を示す一方、CNN の Resnet 系の単精度で、H100 が対 A100 で、1.1 倍しか伸びないケースも見受けられました。速度が出ない理由としては、GPU メモリ容量および GPU メモリ帯域幅といった「足回り」が追い付かないというハードウェア起因のものも見られましたが、ソフトウェアのチューニングが追い付いてない部分も見受けられました。また、Transformer Engine のような新しい機能については使用できるまでに少し時間を要する印象を受けました。ソフトウェア部分は即座には改善できませんが、今後のアップデートで期待できると考えます。

7. 付録

CNN のベンチマーク結果を、もう少し深く考察するために一段踏み込んだ解析を行います。新しいデバイスパフォーマンスだけを知りたい方には、難しい割に得られる情報は少ないかもしれません。

7.1. 学習時間の分解

ベンチマークでは、学習速度という形式でパフォーマンスを表していますが、これは batch size を 1 イタレーションにかかる時間で割った値です。今、学習速度をv、1 イタレーションあたりの時間をt、batch size をbとすると、次のように表現できます。

$$v = \frac{b}{t}$$

ここで、1 イタレーションあたりの時間tを、batch size の 1 次の関数と近似します。

$$t = \frac{b}{v} = Fb + C \quad (F, C\text{は定数})$$

この時、$b \to \infty$では、vは、

$$v = \frac{b}{Fb + C} = \frac{1}{F + \frac{C}{b}} \to \frac{1}{F} \quad (b \to \infty\text{のとき})$$

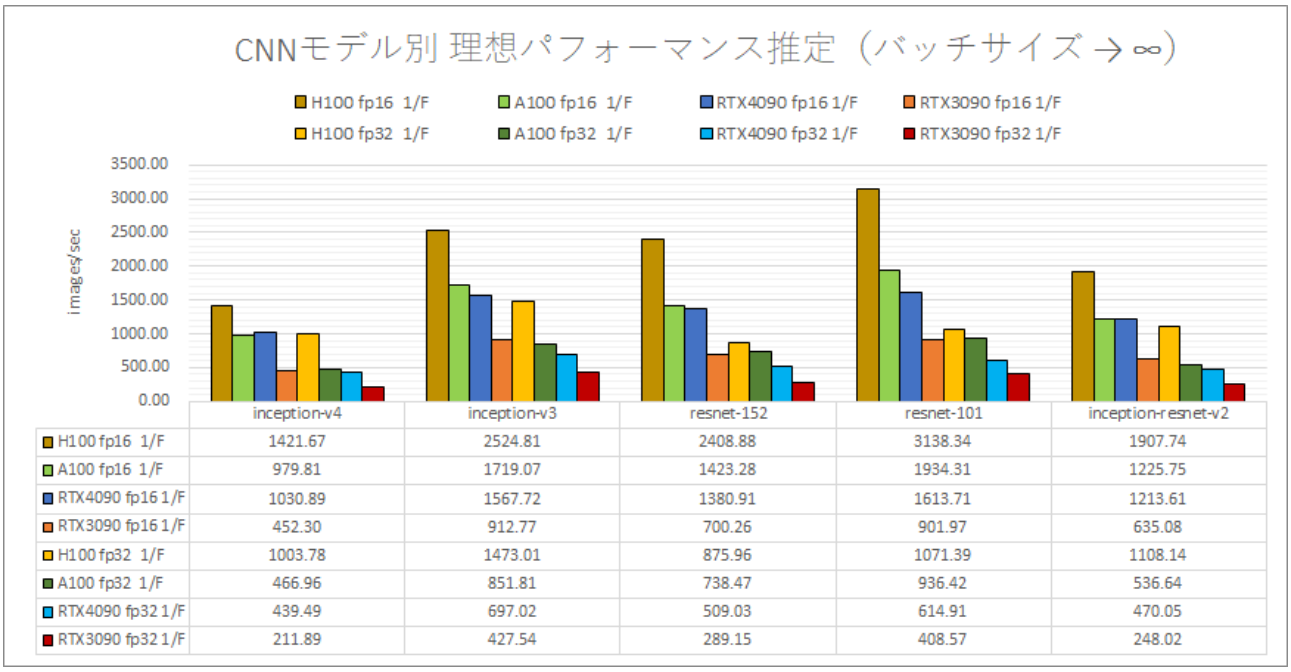
となります。

ここで定数Fは、画像あたりの処理時間で、画像処理に関わる事象の影響を受けます。大きく影響を受けるのは、画像をモデルに通す時の演算、GPU メモリ帯域幅、ホストメモリから GPU メモリへの転送帯域幅です。一方、Cは、イタレーションあたりの画像以外の処理時間で、画像とは関係ない処理が、すべて影響します。大きく影響を受けるのは、モデルの更新演算、演算の前処理で画像量に比例しないもの、通信のレイテンシ、複数 GPU での演算の場合は、GPU 間の通信時間などもCに入ります。batch size を大きくするとパフォーマンスが上がるのは、イタレーション全体の時間に対して、画像処理の時間Fbが長くなり、相対的に画像以外の処理時間Cが短くなるためです。

上記を元に、複数の batch size でパフォーマンスを計測し、一次方程式の定数F, Cを導きます。

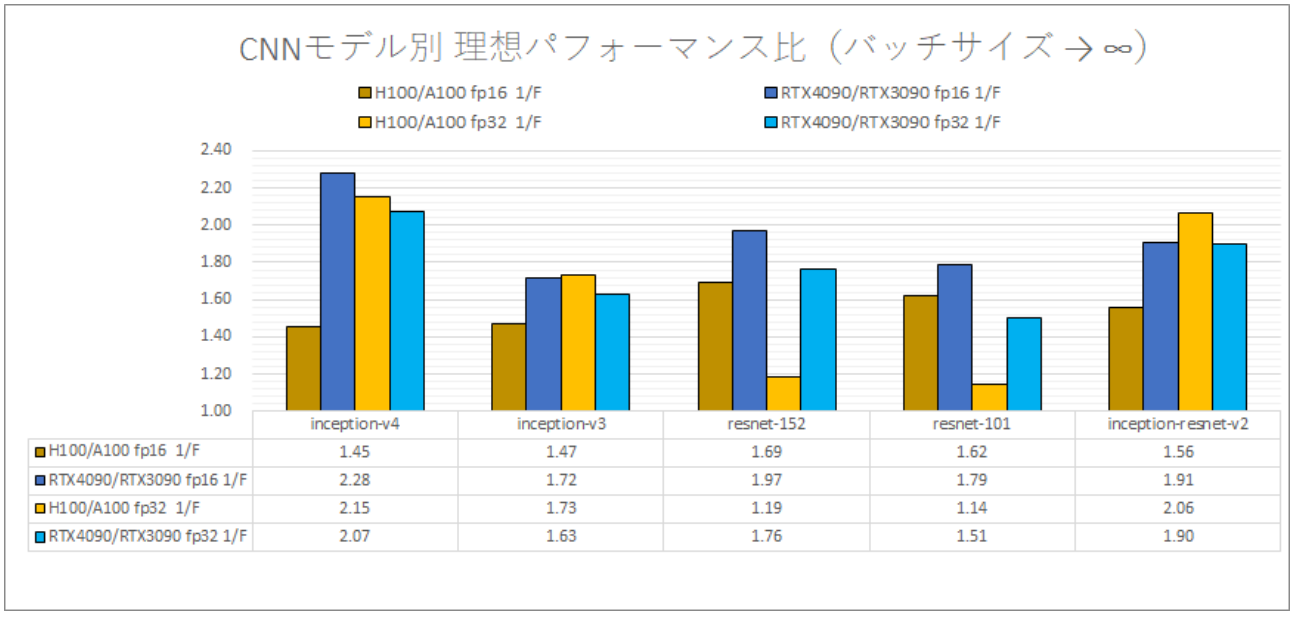
7.2. 画像処理のみの時間

前項より、画像処理のみの時間Fは、batch size を無限大にした時の、パフォーマンスの逆数となります。実際にそれらを計測した結果が下のグラフになります。



	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
H100 fp16 1/F	1421.67	2524.81	2408.88	3138.34	1907.74
A100 fp16 1/F	979.81	1719.07	1423.28	1934.31	1225.75
RTX4090 fp16 1/F	1030.89	1567.72	1380.91	1613.71	1213.61
RTX3090 fp16 1/F	452.30	912.77	700.26	901.97	635.08
H100 fp32 1/F	1003.78	1473.01	875.96	1071.39	1108.14
A100 fp32 1/F	466.96	851.81	738.47	936.42	536.64
RTX4090 fp32 1/F	439.49	697.02	509.03	614.91	470.05
RTX3090 fp32 1/F	211.89	427.54	289.15	408.57	248.02

これを実際のパフォーマンス同様に、H100 と A100、RTX4090 と RTX3090 を比較しやすいように比率にしたものを次図に示します。

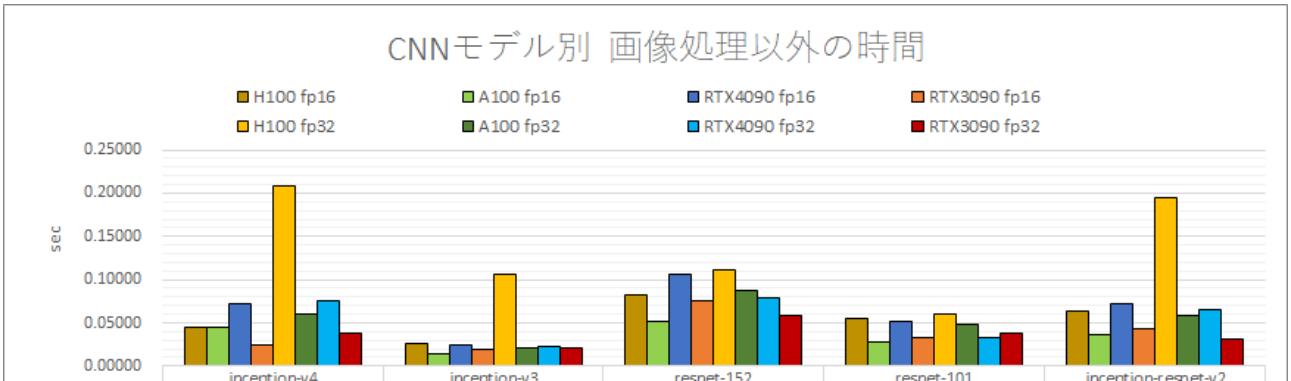


	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
H100/A100 fp16 1/F	1.45	1.47	1.69	1.62	1.56
RTX4090/RTX3090 fp16 1/F	2.28	1.72	1.97	1.79	1.91
H100/A100 fp32 1/F	2.15	1.73	1.19	1.14	2.06
RTX4090/RTX3090 fp32 1/F	2.07	1.63	1.76	1.51	1.90

batch size を無限大に設定できたと仮定した場合に到達できるパフォーマンス比を見ると、RTX4090/RTX3090 が 2 倍前後となるものが増えました。このことから、RTX4090 のメモリ容量が小さいために、大きい batch size が設定できないためにパフォーマンスが発揮しきれてないと言えます。一方、H100 の Resnet 系、FP32 のパフォーマンスは低く留まっていて、batch size を大きくしても速度向上は見込めないことが分かります。また、H100 のその他の FP32 は 2 倍近くに到達することが分かりました。これは、H100 のメモリが 80GB であっても容量不足であることを示しています。H100 の FP16 については、1.45～1.69 倍となりました。演算量が同等の FP32 でパフォーマンスが出ているのに FP16 で出ないのは、ハードウェア起因でなく、ソフトウェア起因と考えられます。

7.3. 画像処理以外の時間

次に batch size に対しての定数項Cを求めてみます。Cは画像処理以外の時間を示します。Cを実際に求めてみると、下のグラフになります。



	inception-v4	inception-v3	resnet-152	resnet-101	inception-resnet-v2
H100 fp16	0.04515	0.02670	0.08188	0.05431	0.06292
A100 fp16	0.04534	0.01498	0.05083	0.02830	0.03693
RTX4090 fp16	0.07227	0.02388	0.10533	0.05133	0.07194
RTX3090 fp16	0.02371	0.01951	0.07572	0.03274	0.04309
H100 fp32	0.20765	0.10584	0.11203	0.06050	0.19440
A100 fp32	0.05983	0.02128	0.08658	0.04815	0.05861
RTX4090 fp32	0.07608	0.02184	0.07863	0.03330	0.06518
RTX3090 fp32	0.03761	0.02033	0.05812	0.03737	0.03136

H100、FP32、inception 系で突出したものが見られます。これが何に要している時間かは不明ですが、このために H100 では大きい batch size を設定しないとパフォーマンスが出ない状態になっています。また、それ以外でも、H100、RTX4090 は前世代のものと比べて長いことが分かります。

モデルの更新時間だけを考えれば、GPU の処理速度が上がれば、更新時間は短くなるので、世代が上がるとCは小さくなります。モデルの更新以外に影響を受けるものとしては、高速化のための前処理の影響が大きいと考えています。高速化のための前処理とは、データを処理する前に、効率良く処理

するためにデータを整える処理のことを指します。高速化のための処理が足を引っ張るのは不思議ですが、高速化のための条件分岐が甘かったりすると容易に起こり得ます。ほとんどの場合は、前処理も含めてライブラリで実装されているはずなので、今後、ソフトウェアによる改善が成される可能性があります。また、単純に今まで高速化できていた処理が、ソフトウェア開発の遅れなどで、新しいバージョンで対応できていない場合も、このようなことが起きます。

7.4. 付録のまとめ

付録全体を通しての見解としては、CNN ベンチマークで速度があまり出ていない理由は、ソフトウェア開発の遅れなどによる不備で、今まで適応できていた高速演算のライブラリが、いくつか適応できてなくなっているのではと考えています。適応できないライブラリが、モデルのプログラムにより、画像と関係する部分か、そうでないかで、挙動が変わっていて、さらに、その部分がよく使用されているかどうかで、挙動の度合いが変わります。その様子を今回の解析結果が表しているのだと考えています。故に、今後のソフトウェアアップデートで理論性能付近まで性能が向上する可能性があると考えています。

8. 改版履歴

版数	作成/更新日	概要
第 1 版	2022/12/23	初稿