Vol.2019-ARC-238 No.5
2019/11/14

AIワークロードを実行する次世代計算機の運用制御に向けたAIワークロードの特徴分析

高山　沙也加[1]　白石 崇[2]　鈴木 成人[2]　山本 昌生[2]　渡辺 幸洋[2]　小口　正人[1]

概要：ディープラーニングの隆盛に伴い，これに特化したハードウェアアクセレラレータとしてGPUが普及した．汎用計算リソースであるCPUと比較すれば，GPUはまだ数が少なく貴重なリソースである．そこで，予めアプリケーション実行に必要なGPU数を予測し，無駄なくアプリケーションに割り当てを行うことで，性能劣化なくGPUの使用を最小限に抑える運用技術を検討したい．その基礎検討として，機械学習アプリケーションのベンチマークであるMLPerfを用い，その特徴分析を行う．また，アプリケーションの動作予測に必要なハードウェアログ情報を選定するため，時系列ハードウェアログを用いたリカレントニューラルネットワークによるアプリケーション分類結果を紹介する．本検討から，各アプリケーションのハードウェアログの特徴把握と，ハードウェアログを使用してアプリケーション分類が可能であることを確認した．古いGPUと新しいGPUが混在するDocker環境を想定して実行時間を見積もったところ，性能の異なるGPUを均等に割り当てた場合と比べ，GPUの違いによるジョブ性能の差が大きいベンチマークに優先的に新しいGPUを割り当てた場合では合計実行時間が8.24%減った

Characteristic Analysis of AI Workload for Operation Control of Next Generation Computer that Executes AI Workload

SAYAKA TAKAYAMA[1]　TAKASHI SHIRAISHI[2]　SHIGETO SUZUKI[2]　MASAO YAMAMOTO[2]
YUKIHIRO WATANABE[2]　MASATO OGUCHI[1]

1. はじめに

ディープラーニングの隆盛に伴い，これに特化したハードウェアアクセレラレータとしてGPUが普及した．GPUは豊富な演算コアと高速通信を可能とするメモリを所有しており，その性能は年々向上している．しかし，汎用計算リソースであるCPUと比較すれば，GPUはまだ数が少なく貴重なリソースである．

そこで，予めアプリケーション実行に必要なGPU数を予測し，無駄なくアプリケーションに割り当てを行うことで，性能劣化なくGPUの使用を最小限に抑える運用技術を検討したい．その基礎検討として，機械学習アプリケーションのベンチマークであるMLPerf[1]を用い，その特徴分析を行う．また，アプリケーションの動作予測に必要なハードウェアログ情報を選定するため，時系列ハードウェアログを用いたリカレントニューラルネットワーク(RNN)によるアプリケーション分類結果を紹介する．本検討から，各アプリケーションのハードウェアログの特徴把握と，ハードウェアログを使用してアプリケーション分類が可能であることを確認した．

2. 背景

リソース性能の違いやアプリケーションの特性を踏まえた運用手法は既に提案されている．CPUの製造過程で生じる性能や消費電力のばらつきを考慮し，特定のアプリケーションに対する電力の可変性の影響を踏まえたスケジューリングによって，実稼働クラスタで使用されているスケジューリングポリシーと比較して最大31%のジョブターンアラウンドタイムの短縮と，最大5.5%の供給電力の削減が実現されている[2]．また，HPCシステム運用におけるクラスタの使用率とジョブ待機時間の最適化を目的に，履歴ワークロードを用いたPortable Batch System（PBS）

[1] お茶の水女子大学
[2] 株式会社富士通研究所

ベースのクラスターのジョブシミュレーションを実行する方法が提案されている [3]. Facebook 社では保有するデータの大部分を機械学習パイプラインに流しており，GPU と CPU の両方を活用するトレーニングと CPU を活用するリアルタイム推論でシステムを使い分け，リソースの無駄を減らしている．特に，リアルタイム推論では入力データが大きいことから必要リソースも異なるため，機械学習に基づく AI ワークロード挙動の特徴分析の重要性が問われている [4]．Facebook 製品の機械学習を活用するタスクを簡素化することを目的とした FBLearner というツールキットが開発されており，機械学習の特徴に合わせて異なるリソース設計の CPU サーバ，GPU サーバで処理している．FBLearner は FBLearner Feature Store，FBLearner Flow，FBLearner Predictor という機械学習パイプラインのさまざまな部分に焦点を当てた 3 つのツールから構成されており，内部ジョブスケジューラを利用して，GPU と CPU の共有プール上にリソースを割り当て，ジョブをスケジュールする．

また，限られた GPU リソースの有効活用の手法として，GPU 仮想化が挙げられる．複数の GPU と CPU を単一のエラスティッククラスタに組み合わせてコンピューティングリソースの管理をサポートする仮想化レイヤとして，Bitfusion FlexDirect[5] がある．FlexDirect では GPU を任意のサイズの仮想 GPU にスライスすることで複数のワークロードを並行して実行できるように設計されており，従来の GPU ソリューションと比べて大幅な GPU リソースの削減を実現している．ローレンス・バークレー国立研究所が開発した HPC 向けの Linux コンテナである Singularity[6] では，仮想化環境上で実行される複数のアプリケーション間で GPU を共有する機能が実装されている．リソース管理にはジョブスケジューラが利用されている．

ボトルネックを分析し，リソースを配分を行う自動ワークロード管理機能や接続を分散する機能を備えた CPU やサーバは既に開発されている．CPU コアやクラスタ単位で負荷に応じて電圧と動作周波数を切り替える Dynamic Voltage and Frequncy Scaling を実行する Intel の CPU アーキテクチャ”Haswell”[7]，CPU アーキテクチャや多くの GPU を積んで AI ワークロードに特化させた NVIDIA のサーバプラットフォーム”HGX-2”[8] 等が例として挙げられる．HGX-2 は通常の IA サーバと異なり 1 サーバで最大 16 個の GPU を利用できる GPU ネックジョブ向けの専用サーバである．

本研究では，アプリケーション実行に必要な GPU 数の予測を基に GPU の使用を最小限に抑える運用技術の基礎検討として，ワークロードの特徴分析を行う．

3. 実験

AI の代表種であるアプリケーション実行時のハードウェア情報の分析を目的として MLPerf を利用して各ベンチマークの性能評価，比較を行う．性能評価には MLPerf に実装されている 9 つのベンチマークを利用する．情報取得には Zabbix[9] を用いる．Zabbix は Zabbix 社が開発したネットワーク管理ソフトウェアで，デフォルト設定の他にテンプレートを用いることによって監視対象を追加することができる．本研究では IPMI，Perf，nvidia-smi，そして Intel CPU が備える性能監視機構である PMU（Performance monitoring unit）[10] で取得できる情報を分析対象とする．各ベンチマーク毎の特徴分析のためにこれらのコマンドによって CPU や GPU，メモリ，I/O といった情報をベンチマーク実行時に取得する．情報取得は 1 分間隔で行う．各ベンチマークの測定条件を表 1 に示す．また，RNN translator のみ学習精度を 15 に変更した．今回

表 1 各ベンチマークの測定条件

Benchmark	epoch (step，iteration)	SEED	Job Time
IC	53200 (step)	1	3:01:13
SSD	11	1	3:09:12
OD	25000 (iteration)	3	2:23:26
RM	6	1	1:09:34
SA	100	1	1:22:50
RT	1	1	2:48:18
TL	17200 (step)	1	2:59:55
SR	1	1	10:53:32
RI	4	1	5:46:00

は CPU や GPU，メモリの利用量に注目して特徴分析を行った．実験環境は表 2 に示す．

表 2 実験環境

OS	ubuntsu 16.04
Server	FUJITSU Primergy RX2540 M4
CPU	Intel Xeon Skylake 2 sockets 20 cores 2.4GHz Gold 6148 150W
GPU	NVIDIA Tesla V100 16GB
Storage M2.SSD	290GB read 0.87GB/s write 1.75GB/s
Memory	192GB DDR4 2666MHz
Python	3.50
CUDA	9.2

以下に本研究で利用するソフトウェアの概要を記す．

3.1 MLPerf

MLPerf は Google，Intel，Baidu，NVIDIA および数十の企業が支援する AI ベンチマークである．AI 専用プロ

セッサの多様化に対応し，機械学習分野において異なるアーキテクチャ間でシステムパフォーマンスを測定できるベンチマークセットの構築を目的として開発された．

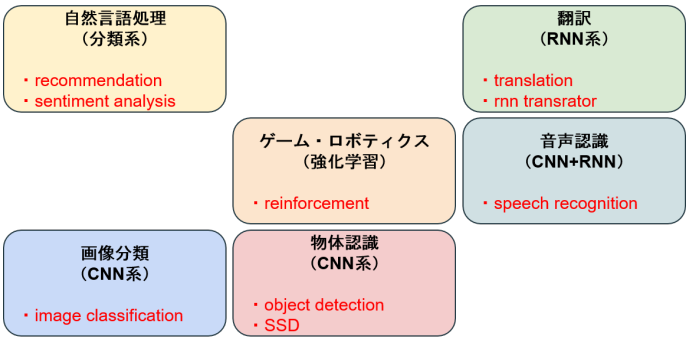

図 1 MLPerf 概要

カテゴリとしては図 1 にあるように，画像分類の image classification，物体認識の single stage detector，object detection，自然言語処理の recommendation，sentiment analysis，翻訳の rnn translator，translation，音声認識の speech recognition，強化学習の reinforcement（以降は IC，SSD，OD，RM，SA，RT，TL，SR，RI と略記する）といった 9 つのベンチマークがおよそ 6 つの分野に分けられ，TensorFlow，Pytorch，Caffe2，Paddle などのフレームワークが用いられている．本研究ではこれらのベンチマークの動作中のハードウェアログを取得し，分析する．

3.2 Zabbix

Zabbix は Zabbix 社が開発しているネットワーク管理ソフトウェアで，様々なネットワークサービス，サーバ，ネットワークハードウェアのステータスを監視・追跡できる．データ格納には PostgreSQL を用いる．本研究では図 2 にあるような環境を構築し，情報取得を行う．情報取得によるベンチマーク実行サーバへの負荷を避けるため，Zabbix エージェントと情報取得を行う Zabbix サーバを分けている．

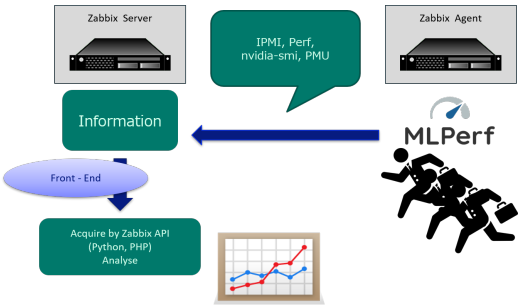

図 2 Zabbix 情報取得環境

4. 解析結果

4.1 取得データの解析

取得した時系列データがサーバ設計にどのように利用できるかの考察を行う．本項では例として IC のデータを取り上げる．

IPMI では，電力や温度といったインフラ情報の取得ができる．図 3 は CPU 温度の時系列データである．図 4 は CPU と GPU の消費電力の時系列データである．Perf では，OS が取得できる CPU 利用率，メモリ利用量が取得でき，nvidia-smi では GPU に関連する情報の取得が可能である．これらの情報はデバイスリソースの静的な最適設計に活用することができる．図 7 は DISK I/O の時系列データである．図 5 は CPU と GPU のメモリ利用率の時系列データである．図 6 は CPU と GPU の利用率の時系列データである．PMU では CPU のイベント情報から，命令数，キャッシュミス率からメモリアクセス数が取得できる．これらの情報は動的な周波数設計に活用できる．図 8 のメモリ intensive 指標の時系列データは，次式で求めた値を利用した．

$$intensive = \frac{(local + remote)}{inst} \times 100$$

ここで，local は MEM_LOAD_L3_MISS_RETIRED.LOCAL_DRAM(all) で得られる L3 キャッシュをミスして local DRAM にいった回数，remote は MEM_LOAD_L3_MISS_RETIRED.REMOTE_DRAM(all) で得られる L3 キャッシュをミスして remote DRAM にいった回数，そして inst は INST_RETIRED.ANY(all) で得られる命令数である．

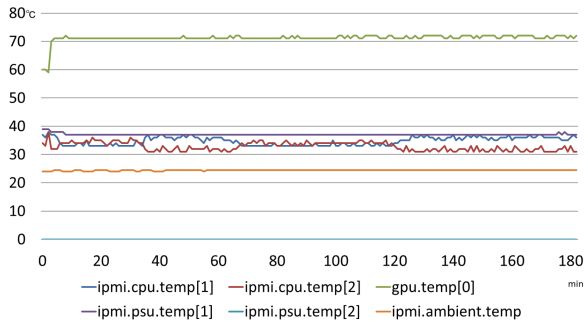

図 3 IC 温度

これらのグラフの考察を行う．まず温度と CPU 電力，メモリ intensive 指標だがベンチマーク走行開始付近を除きほぼ大きな変動は見られなかった．しかしキャッシュミスの割合が変動し各ソケットのメモリ intensive が増減すると温度と電力もほぼ同じタイミング増減しており，ソケット毎の値の変動に法則性が見られた．

CPU と GPU のメモリ利用率は，GPU メモリ利用率が開始すぐに 100%近くなっていることと，CPU メモリ利用

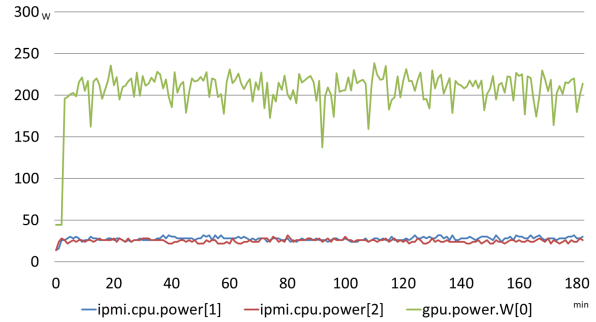

図 4　IC CPU/GPU 消費電力

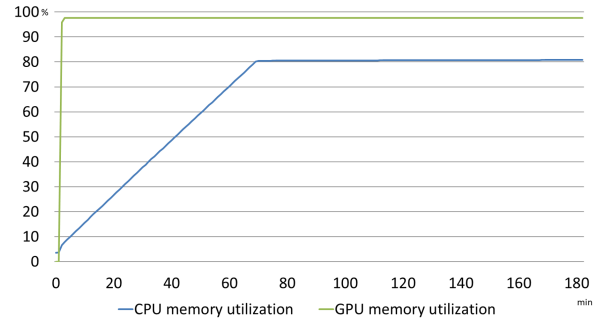

図 5　IC CPU/GPU メモリ利用率

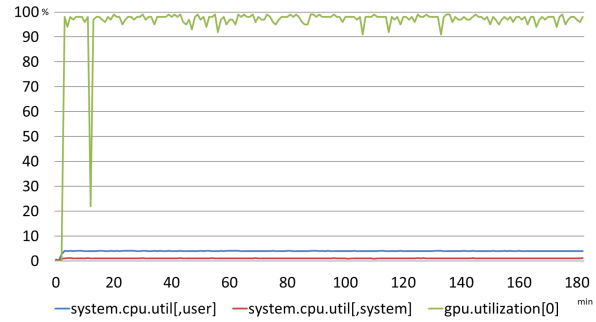

図 6　IC CPU/GPU 利用率

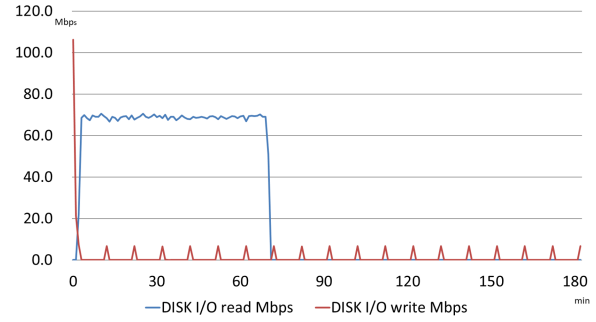

図 7　IC Disk I/O アクセス量

率が実行開始後 70 秒頃まで一次関数的に増加していたのが 80%に達してからは変化がなくなっていることが特徴として挙げられる．このベンチマークの学習用の入力データセットの総サイズは，約 140GB であるのに対し，GPU メモリは 16GB とベンチマークが要求するメモリ量と比べて著しく小さく，100%近く使われていることが分かる．CPU

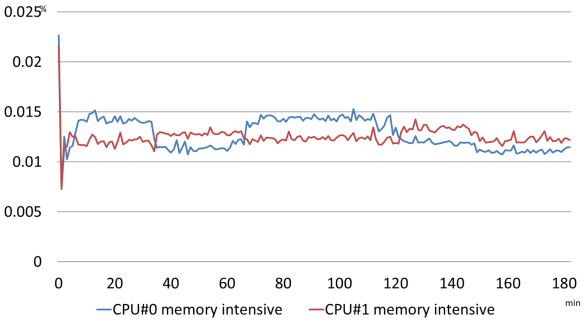

図 8　IC メモリ intensive 指標

メモリも利用率こそ GPU と比べて低いが 8 割程度利用されており，必要なデータをディスクではなく転送速度の速いメモリに格納している事が分かる．次に CPU と GPU の利用率についてだが，GPU の利用率がほぼ 100%に近いのに対して CPU 利用率は 10%を常に下回っている．

よって，IC は CPU 性能はさほど必要とされず GPU 性能が要求されるアプリケーションであり，また，メモリ容量も重要であると考えられる．

4.2　メモリ利用量

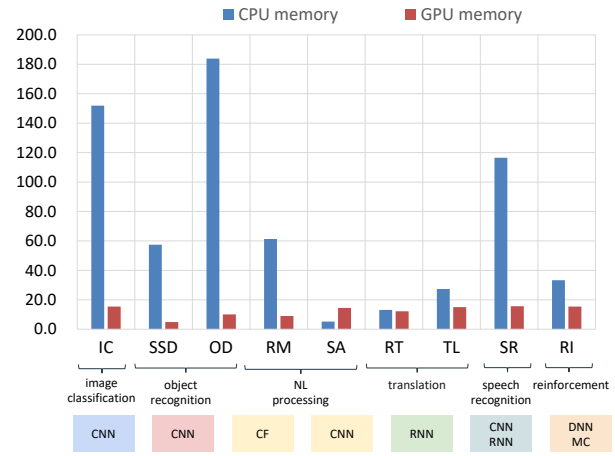

図 9　メモリ利用量

図 9 の縦軸はメモリ利用量の最大値を表している．GPU メモリ利用量と比べ，殆どのアプリケーションでは CPU メモリ利用量が大きくなっている．これはアプリケーションが要求するデータサイズに対して GPU メモリ容量が不足しており，CPU 利用率が低いアプリケーションでも CPU メモリも使う必要があるためであり，CPU と GPU 間のデータ転送が性能のボトルネックとなる可能性がある．そのため，容量不足の解決がより効率の良いアプリケーション実行に繋がると考えられる．また，画像分類系アプリケーションや翻訳系アプリケーションはどちらも GPU 利用率の高いワークロードだが，CPU メモリ利用量に大きな差がある．前者はデータ量が大きい画像データを用いて

おり，後者は比較的データ量の小さいテキストデータを用いているためである．

4.3 CPU

世代の異なるサーバで MLPerf を走行させた際のハードウェアログの比較を行う．比較に用いるサーバ性能自体は類似しているが, 利用 CPU が異なる．

まず，これまでの実験環境におけるディスクアクセス速度を測定する．表 3 は各ベンチマーク実行時のディスクアクセス速度の最大値である．実験で用いた FUJITSU Primergy RX2540 M4 の最大ディスクアクセス速度は read が 0.87GB/s，write が 1.75GB/s である．ベンチマーク実

,

表 3　ディスクアクセス速度

IC	73.92	111.32
SSD	6.75	10.96
OD	4.66	1.25
RM	2.67	29.26
SA	9.82	0.01
RT	29.50	0.06
TL	26.14	93.90
SR	7.10	8.05
RI	22.13	0.75

行時のディスクアクセス速度の最大値がサーバのディスク性能に対して余裕があり，また，ディスクアクセスの発生する時間はジョブ時間と比べて著しく短いため，ディスク性能差によるベンチマークへの影響は小さいと考えられる．

4.3.1 CPU 比較

搭載 CPU の異なるサーバを利用した際に生じるベンチマーク実行に要するジョブ時間の変化に関する考察を行う．CPU 比較実験に用意した実験環境の詳細を 4 に記す．

IC，SSD，OD，RM，SA，RT，TL，RI のベンチマークで比較を行った．

表 5 に表 4 の環境で MLPerf を実行した際に要したジョブ時間の比較結果を記す．

CPU の性能を測る意図で 1 命令の実行に要するクロック数を表す CPI の情報取得も行ったが，サーバ変更に伴う法則性のある大きな変化は見られなかった．

図 10，図 11，図 12，図 13，図 14，図 15，図 16，図 17 はそれぞれのサーバで各ベンチマークを実行した際のスレッド毎のクロック周波数の時系列データである．

SA では特定のスレッドのみクロック周波数が高い値になっており，これは，周波数が落ちると一部のスレッドのみを使うようになる仕様のためと考えられる．Skylake と Haswell の結果を比べると，最大値や細部に違いはあるがクロック周波数は時系列に沿ってある程度似た変化が起きていることが確認できる．これらの結果から，サーバや

表 4　CPU 比較用実験環境

	Environment 1	Environment 2
OS	CentOS Linux release 7.5.1804 (Core)	ubuntsu 16.04
Server	FUJITSU Primergy CX400 M1	FUJITSU Primergy RX2540 M4
CPU	Intel Xeon Haswell 2 sockets 14 cores 2.6GHz E5-2697 145W	Intel Xeon Skylake 2 sockets 20 cores 2.4GHz Gold 6148 150W
GPU GPU	NVIDIA Tesla P100 16GB	NVIDIA Tesla P100 16GB
Storage HDD	270GB read 0.21GB/s write 1.07GB/s	290GB read 0.87GB/s write 1.75GB/s
Memory	256GB DDR4 2133MHz	192GB DDR4 2666MHz
Python	3.50	3.50
CUDA	9.2	9.2

表 5　ジョブ時間の比較 - CPU

Benchmark	Haswell	Skylake	Skylake / Haswell
IC	4:36:10	4:33:04	0.99
SSD	3:56:42	3:12:18	0.81
OD	2:59:34	2:57:51	0.99
RM	1:12:00	1:09:07	0.96
SA	2:00:12	1:48:14	0.90
RT	4:06:41	4:05:28	1.00
TL	4:37:47	4:29:21	0.97
SR	-	15:07:34	-
RI	6:28:00	6:08:37	0.95

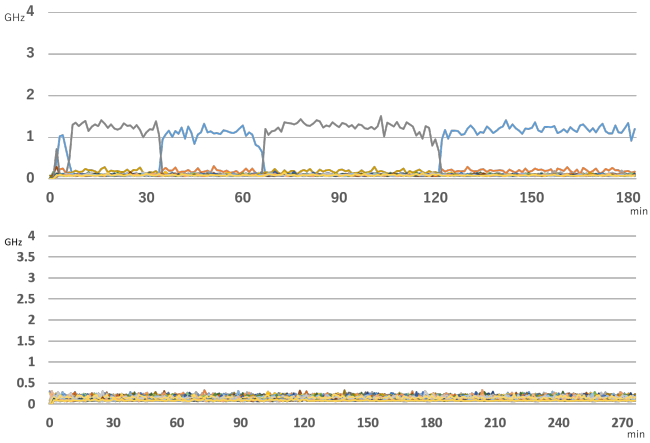

図 10　IC のクロック周波数 上: Skylake 下 : Haswell

CPU の違いが AI アプリケーションの性能に与える影響は小さいと考えられる．

4.4 GPU

各ベンチマーク実行時の GPU/CPU 比率を図 18 に示す．また，表 6 は CPU と GPU の平均利用率を表している．

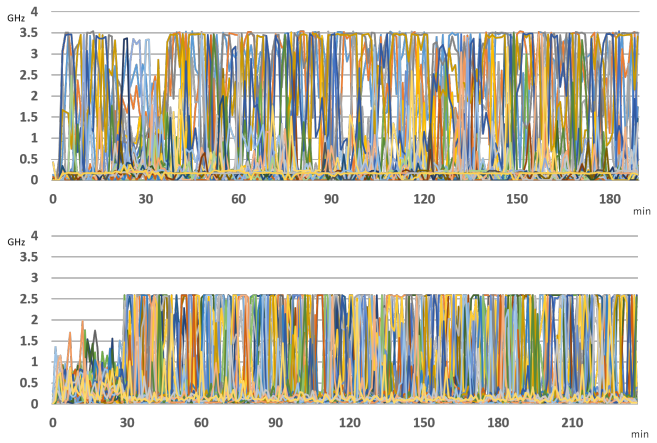

図 11　SSD のクロック周波数 上: Skylake 下：Haswell

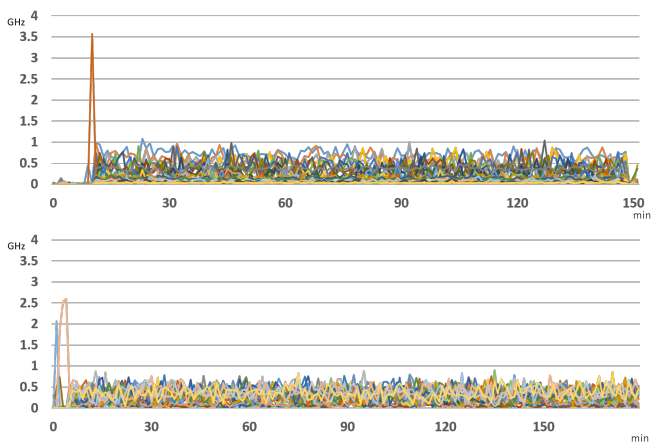

図 12　object detection のクロック周波数 上: Skylake 下：Haswell

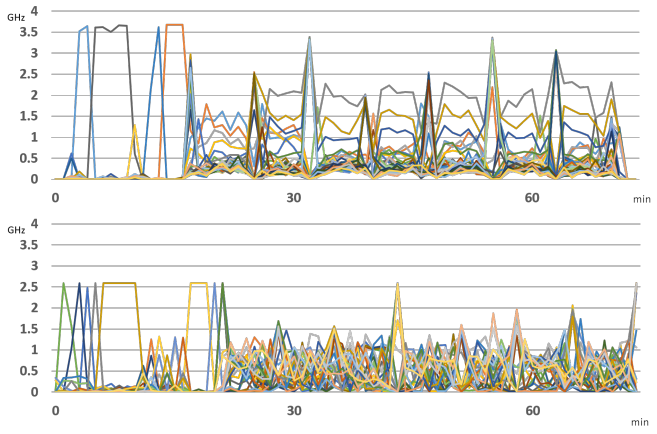

図 13　RM のクロック周波数 上: Skylake 下：Haswell

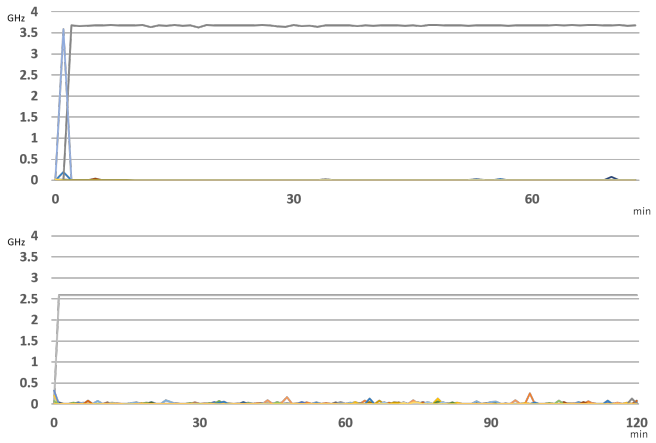

図 14　SA のクロック周波数 上: Skylake 下：Haswell

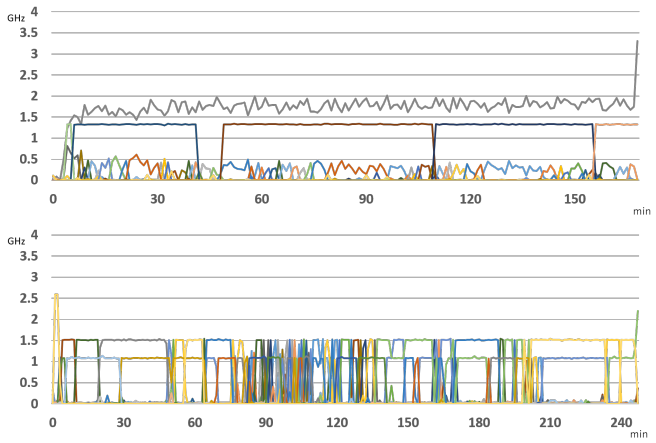

図 15　RT のクロック周波数 上: Skylake 下：Haswell

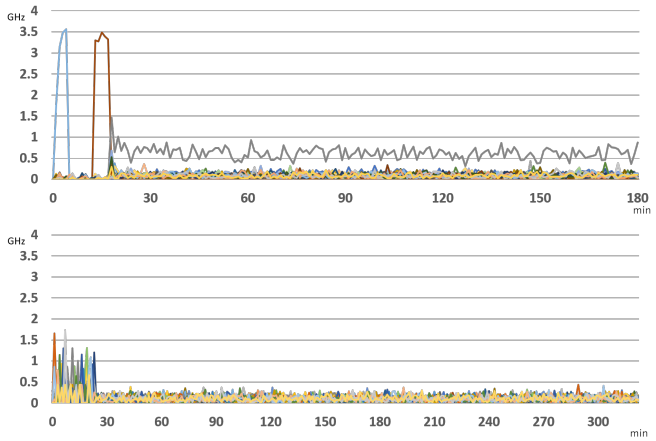

図 16　traslation のクロック周波数 上: Skylake 下：Haswell

図 18 の縦軸は 1 ソケット当たりの GPU 利用率の平均値を CPU 利用率の平均値で割ったものを表している．ここでは，本研究で利用したサーバは 2 ソケットなので CPU1 ソケット当たりの利用率に換算している．表 6 を見るとアプリケーション全体としては GPU ネックの傾向にあり，翻訳系アプリケーションでは特に GPU の必要量が大きいことが確認できる．それに対して，RM や RI のように CPU 性能も重要になると考えられるアプリケーションも見られる．また，系統の異なるアプリケーションではプロセッサの利用率に大きく違いがある．

メモリ利用量の結果も踏まえると，容量不足の問題が解決できればより効率良いアプリケーションの利用が可能になると推測できる．

4.5　GPU 比較

世代の異なる GPU で MLPerf を走行させた際のハード

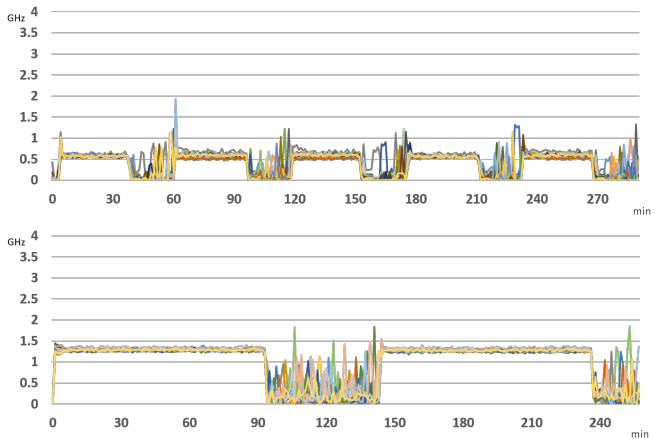

図 17 RI のクロック周波数 上: Skylake 下：Haswell

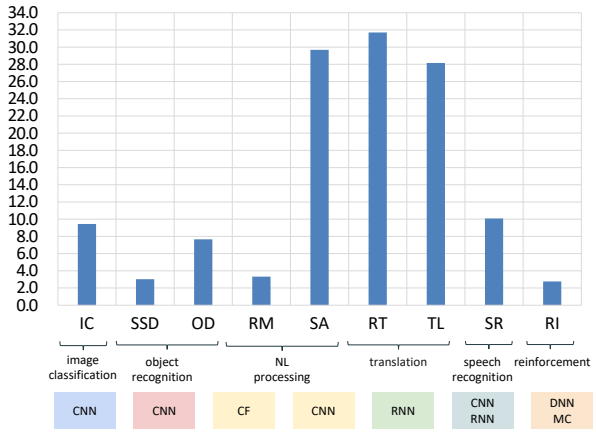

図 18 GPU/CPU 比率

表 6 CPU/GPU 平均利用率

Benchmark	CPU (%)	GPU (%)
IC	5.1	95.4
SSD	9.9	59.8
OD	4.9	74.5
RM	6.7	44.3
SA	1.4	82.0
RT	1.5	95.5
TL	1.5	83.9
SR	3.2	65.1
RI	11.4	62.7

ウェアログとの比較を行う．GPU 比較に用いる実験環境の詳細を 7 に記す．

実験に用いた GPU の V100 と P100 のスペックを表 8 に示す．

表 9 は GPU を V100 から P100 に変更し，それぞれで各ベンチマークを実行した際のジョブ時間の比較結果である．表 6 の GPU 平均利用率と比べると，GPU 平均利用率が高いベンチマークほど GPU を変更した際のジョブ時間の変化が大きくなっている．

表 7 GPU 比較用実験環境

	Environment 1	Environment 2
OS	ubuntsu 16.04	ubuntsu 16.04
Server	FUJITSU Primergy RX2540 M4	FUJITSU Primergy RX2540 M4
CPU	Intel Xeon Skylake 2 sockets 20 cores 2.4GHz Gold 6148 150W	Intel Xeon Skylake 2 sockets 20 cores 2.4GHz Gold 6148 150W
GPU	NVIDIA Tesla P100 16GB	NVIDIA Tesla V100 16GB
Storage HDD	290GB read 0.87GB/s write 1.75GB/s	290GB read 0.87GB/s write 1.75GB/s
Memory	192GB DDR4 2666MHz	192GB DDR4 2666MHz
Python	3.50	3.50
CUDA	9.2	9.2

表 8 GPU スペック

	P100	V100
Core	3584	5120
MHz	1300	1455
FP16	18.636	119.19
FP32	9.318	14.90
FP64	4.659	7.45
Memory Bandwidth	720	900

表 9 ジョブ時間の比較 - GPU

Benchmark	P100	V100	V100 / P100
IC	4:33:04	3:01:13	0.66
SSD	3:12:18	3:09:12	0.98
OD	2:57:51	2:23:26	0.81
RM	1:09:07	1:09:34	1.01
SA	1:48:14	1:22:50	0.76
RT	4:05:28	2:48:18	0.68
TL	4:29:21	2:59:55	0.67
SR	15:07:34	10:53:32	0.72
RI	6:08:37	5:46:00	0.93

これらの結果から GPU 利用率が高いジョブは GPU 性能によるジョブ性能向上効果が高く，CPU 性能によるジョブ性能の差は小さいと推測できる．

4.6 優先制御

Docker 環境下で，1 つのアプリケーションで物理マシンを占有させることを想定する．この想定では，MLPerf の 9 つのベンチマークを処理するそれぞれのマシンには GPU リソースとして各 10 つの P100 と V100 のうちいずれか 1 つが割り当てられることとする．GPU 性能の影響の大きいワークロードに優先的に性能の良い GPU を割り当てを行った場合と均等に GPU を割り当てた場合とのジョブ実行時間の比較を行う．

表 10　割り当てポリシーの違いによる総実行時間の変化

Number of Jobs	Proposed Method (h:m:s)	Evenly Assigned (h:m:s)	Reduce (%)
200	74:30:00	81:11:27	8.241
400	149:00:00	162:22:54	8.241
600	223:30:00	243:34:21	8.241
800	298:00:00	324:45:48	8.241
1000	372:30:00	405:57:15	8.241
10000	3724:42:00	4059:32:33	8.248

表 10 は比較実験の結果である．処理するジョブの数に関係なくほぼ一定の割合でジョブ性能の改善が確認された．先行研究 [11] では，提案されたアルゴリズムによってタスク完了までの全体的な実行時間が 9.47%改善されることが観察されている．また，より精度の高い情報予測によって総実行時間が 1.2% - 8%短縮されたことが示されている [12]．見積もりの結果から，本研究で提案する制御方法では 8.24%の総実行時間の短縮が見込まれ，より無駄のないリソース運用に活かせると考えられる．

5. RNN によるベンチマークの分類

未知のアプリケーションの動作予測にあたって必要になるハードウェアログ情報を選定したい．そこで，ベンチマーク実行時に得られるハードウェアログの時系列データを用いた RNN によるクラス分類を試みる．分類には各ベンチマーク走行開始から 320 秒後までの時系列データを用い，Google が開発した深層学習のライブラリである TensorFlow で実装した RNN を利用する．表 11 に TensorFlow のパラメータ条件を記す．

表 11　TensorFlow のパラメータ条件

Epoch	200
Series Length	74
Class	9
Data	288
Batch Size	32
Hidden Layer	96
Cell	BasicRNNCell

図 19 はサーバ消費電力の時系列データを用いてクラス分類した際の精度と損失である．学習を経るごとに損失は減少傾向，精度は上昇傾向にあるがはあまり高い精度は得られなかった．次に，CPU 利用率の時系列データを用いてクラス分類した結果を図 20 に示す．こちらもサーバ消費電力を用いて分類した場合と同様にあまり精度は高くない．図 21 は GPU 利用率を学習に用いた際の精度と分類である．こちらはサーバ消費電力や CPU 利用率と比べて学習による精度向上や損失減少に成功しており，分類精度が高い．

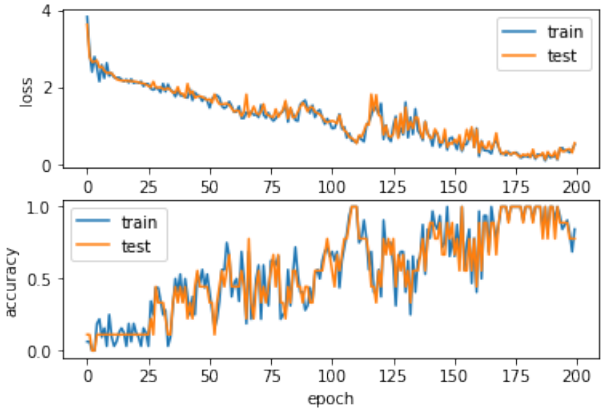

図 19　RNN によるサーバ消費電力での分類精度と損失

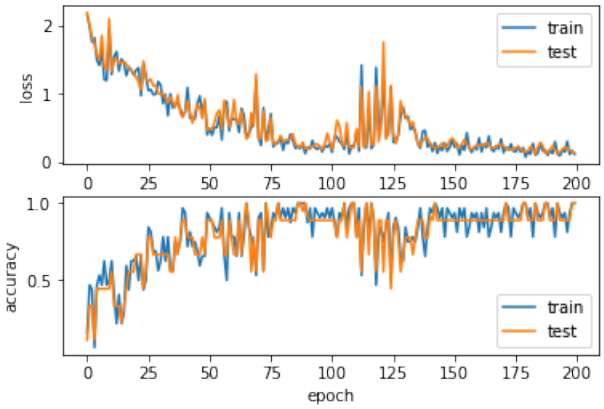

図 20　RNN による CPU 利用率での分類精度と損失

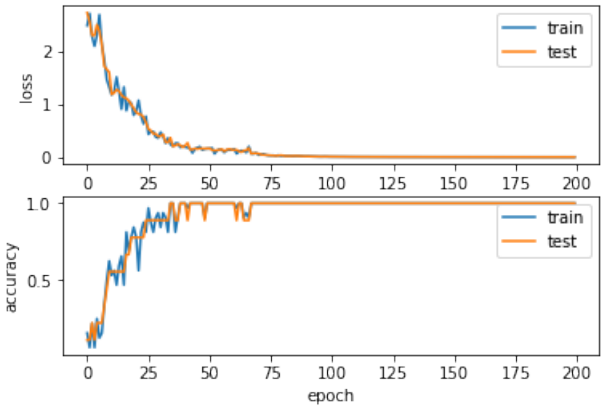

図 21　RNN による GPU 利用率での分類精度と損失

図 19，図 20，図 21 の実験で得られた各ベンチマークの正答率を表 12 に示す．

GPU 利用率を用いた際の分類の正答率が他の二つの結果と比べて全体的に高いことがわかる．また，CPU 利用率で分類した際の正答率のうち SA，RT，TL の正答率が特に低い．この三つのベンチマークは表 6 にあるように特に平均 CPU 利用率が低く，他のベンチマークと比べて値の変動が小さいためうまく分類できなかったと考えられる．

これらの結果から，AI アプリケーションのパラメータと

表 12　各ベンチマークの正答率

Benchmark	power	CPU utilization	GPU utilization
IC	0.40	0.84	0.86
SSD	0.60	0.89	0.99
OD	0.73	0.81	0.93
RM	0.69	0.91	0.99
SA	0.49	0.61	0.87
RT	0.64	0.43	0.84
TL	0.71	0.45	0.88
SR	0.80	0.97	0.94
RI	0.62	0.54	0.96

して GPU 利用率が特に重要でありアプリケーションの動作予測に有用であると考えられる．

6. まとめと今後の予定

予めアプリケーション実行に必要な GPU 数を予測し，無駄なくアプリケーションに割り当てを行うことで，性能劣化なく GPU の使用を最小限に抑える運用技術を検討したい．その基礎検討として，機械学習アプリケーションのベンチマークである MLPerf を用い，その特徴分析を行った．

サーバを変更して実験したところ，ディスクアクセス速度や CPI には大きな変化は見られなかった．

GPU を用いた AI アプリケーション群でも GPU の必要量に大きく差があるが全体的には GPU ネックの傾向があり，GPU 利用率が高いジョブは GPU 性能によるジョブ性能向上効果が高く，CPU 性能によるジョブ性能の差は小さいと推測した．

また，アプリケーションの動作予測に必要なハードウェアログ情報を選定するため，時系列ハードウェアログを用いたリカレントニューラルネットワークによるアプリケーション分類を行った．GPU 利用率のデータを用いた分類が特に精度が高く，ハードウェアログを使用したアプリケーション分類が可能であることを確認した．

古い GPU と新しい GPU が混在する Docker 環境を想定して実行時間を見積もったところ，性能の異なる GPU を均等に割り当てた場合と比べ，GPU の違いによるジョブ性能の差が大きいベンチマークに優先的に新しい GPU を割り当てた場合では合計実行時間が 8.24%減少した．

今後は，実際に GPU の運用制御を行うシステムの構築を行いたい．

謝辞

本研究の一部はお茶の水女子大学と富士通研究所との共同研究契約に基づくものである．また，本研究にご協力頂いた富士通コンピュータテクノロジーズの鈴木孝規氏に深謝する．

参考文献

[1] MLPerf v0.5. *https://mlperf.org*, Accessed: December 2018.
[2] Dimitrios Chasapis, Miquel Moretó, Martin Schulz, Barry Rountree, Mateo Valero, and Marc Casas. Power efficient job scheduling by predicting the impact of processor manufacturing variability. In *Proceedings of the ACM International Conference on Supercomputing*, pp. 296–307. ACM, 2019.
[3] Georg Zitzlsberger, Branislav Jansík, and Jan Martinovič. Job simulation for large-scale pbs based clusters with the maui scheduler. *BIG DATA ANALYTICS, DATA MINING AND COMPUTATIONAL INTELLIGENCE 2018 THEORY AND PRACTICE IN MODERN COMPUTING 2018*, p. 137.
[4] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, and et al. Applied machine learning at facebook: A datacenter infrastructure perspective. *IEEE HPCA2018*, pp. 620–629, February 2018.
[5] Bitfusion FlexDirect Virtualization Technology White Paper. *http://bitfusion.io/wp-content/uploads/2017/11/bitfusion-flexdirect-virtualization.pdf*, Accessed: September 2019.
[6] Singularity. *https://singularity.lbl.gov/*, Accessed: September 2019.
[7] Haswell. *https://ark.intel.com/ja/products/codename/42174/Haswell.html*, Accessed: December 2018.
[8] HGX-2. *https://www.nvidia.com/en-us/data-center/hgx/*, Accessed: December 2018.
[9] Zabbix. *http://www.zabbix.com*, Accessed: December 2018.
[10] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B :Order Number 253669-067US.*, pp. 19.3–19–24, 19.46–19.58, May 2018. https://software.intel.com/en-us/articles/intel-sdm.
[11] Ashish Kumar Mishra, Dharmendra K Yadav, Yogesh Kumar, and Naman Jain. Improving reliability and reducing cost of task execution on preemptible vm instances using machine learning approach. *The Journal of Supercomputing*, Vol. 75, No. 4, pp. 2149–2180, 2019.
[12] Lingyun Yang, Jennifer M Schopf, and Ian Foster. Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, p. 31. ACM, 2003.