Vol.2019-HPC-170 No.40
2019/7/26

大規模GPUクラスタにおける深層学習ワークロードの傾向把握

滝澤 真一朗[1,a)]　小川 宏高[1]　高野 了成[1]

概要：GPU，高速ネットワーク，高性能 IO 技術の発展に伴い，深層学習アプリケーションが HPC システム上で実行されるようになってきている．それに伴い，古典的な HPC アプリケーションと深層学習アプリケーションのワークロードがシステム上で共存している．しかしながら，HPC システムにおける深層学習アプリケーションのワークロードの傾向は詳細に分析されていない．その知識がないと，HPC アプリケーションと深層学習アプリケーションの混合ワークロードのスケジューリングを効率的に行うことができない．本研究では，深層学習のワークロード分析の第一歩として，AI 研究開発専用に構築された GPU クラスタで実行されたワークロードを公開する．2017 年 7 月から 2018 年 12 月の間に投入された 459,545 ジョブを含む．さらに我々はそのワークロードを 5 つの観点から分析を行い，古典的な HPC システムのワークロードとの比較を行った．

1. はじめに

深層学習の学習計算において，GPU が広く利用されている．ニューラルネットワークの複雑化や学習データセットサイズの増加に伴い，単一 GPU だけでなく，複数 GPU を搭載したサーバや，それらサーバを高性能ネットワーク・ストレージに接続した GPU クラスタも利用されている．実際に，2018 年のゴードン・ベル賞を受賞した研究では，異常気象パターンの識別に深層学習を採用し，ORNL の GPU スーパーコンピュータ Summit を用いて評価を行なっている [1]．今後，GPU を搭載したスーパーコンピュータが深層学習を用いた科学技術計算や製品開発に利用されることが期待されている．これにより，スーパーコンピュータ上で古典的な HPC アプリケーションと深層学習を用いたアプリケーションが混在することになるが，それぞれのワークロード（実行されたアプリケーションの，ジョブ投入時刻，並列度，実行時間等の情報）の違いは明らかにはなっていない．例えば，ジョブの並列度と実行時間はジョブスケジューリングポリシの設計に影響するが，それらについて，古典的な HPC アプリケーションと深層学習アプリケーションでの違いは明らかにされていない．ワークロードの特徴を把握し，ワークロードに合わせたスケジューリングポリシの設定を行うことは資源利用率，および利用者満足度（待ち時間短縮等）向上に重要である．しかしながら，スーパーコンピュータ上で実行されている一般的な並列アプリケーションのワークロード分析は広く行われているものの，そのようなシステム上での深層学習のワークロードの分析は行われていない．

深層学習ワークロードの理解の助けとなるよう，我々は GPU クラスタ上で実行された深層学習アプリケーションのワークロードを公開するとともに，その分析結果を示す．[*1] ワークロードは，我々が運用する AI 研究開発専用に構築した GPU クラスタから得られたものであり，2017 年 7 月から 2018 年 12 月の 1 年半の間に実行された 459,545 ジョブのデータを含む．深層学習の研究者や開発者がどのように計算機を利用するか把握するため，ワークロードを次の 5 つの視点で分析した．並列度，ジョブ実行時間の見積の正確性，実行時間，並列度と実行時間の相関，ジョブ到着の傾向．これらについての分析結果を，既存の並列システムのワークロード分析の結果と比較した．その結果，既存のワークロードに比較して，深層学習ワークロードでは，単一 PE（Processing Element，今回の対象は GPU）ジョブがジョブ数の大半を占めること，実行時間の見積がより不正確であること，並列度と実行時間には正の相関があること，などの知見が得られた．

2. 関連研究

HPC システムの利用傾向を把握するために，HPC シス

[1] 産総研・東工大 実社会ビッグデータ活用 オープンイノベーションラボラトリ, RWBC-OIL
[a)] shinichiro.takizawa@aist.go.jp

[*1] The workload log will be published under Creative Commons BY-SA license on *https://github.com/aistairc/aaic-workload/*.

テム上でのワークロードを分析する研究が多数行われてきた．Li らはグリッド環境における 5 種類のクラスタの分析を [2]，You らはスーパーコンピュータ Kraken の分析を [3]，Rodrigo らは NERSC の 3 システムの分析を [4]，Feng らは Tianhe-1A [5] の分析をそれぞれ行っている．ワークロード分析によく用いられているメトリクスはジョブ形状（並列度，メモリ使用量，実行時間等），時間的な特徴（ジョブ到着時刻，ジョブ到着間隔，待ち時間等），およびそれらの相関であるが，独特な視点から分析を行っている研究もある．Schlagkamp らは，スケジューリング性能が利用者のシステム利用パターンに与える影響を把握することを目的に，*Think Time* というメトリクスを導入して評価を行なっている [6]．Think Time は，ある利用者の 1 ジョブが終了した時刻と，その利用者が次に投入するジョブの投入時刻との差分である．また，Schlagkamp らは，システム利用者の満足度調査や，システム運用者が把握できない利用者間での隠れたやりとりを理解すること目的に，利用者へのアンケートベースでの利用分析を行っている [7]．Ren らは大規模商用 e-コマースシステム上での Hadoop アプリケーションの挙動把握を目的とした Hadoop ワークロードの詳細分析を行っている [8]．

これらと比較した本研究の違いは，対象ワークロードにある．本研究では，古典的な HPC システムと共通するアーキテクチャを有する GPU クラスタ上での，深層学習アプリケーションのワークロード分析を行っている．我々の分析の目的は，深層学習の研究者・開発者がどのように GPU 資源を利用するか把握することにある．分析手法とメトリクスは上記関連研究と同じものを採用しているが，大量の深層学習ワークロードに対してそのような分析を行っている研究はない．さらに我々は分析結果を関連研究で示されている結果と比較し，類似性・相違性を示している．

ワークロード分析結果の公開だけでなく，ジョブスケジューリングアルゴリズムの研究，ワークロード比較のために，ワークロードそのものを公開し，研究開発に自由に利用できるよう整備する活動もある．有名なワークロードリポジトリとして Parallel Workloads Archive (PWA) [9], [10] がある．PWA で公開されているワークロードは Standard Workload Format でフォーマットされており，30 を超えるシステムのワークロードが公開されている．他のリポジトリとしては，Grid Workloads Archive [11], [12] や，JSSPP workloads archive [13] がある．

多くのワークロードが研究目的に自由に使えるよう公開されているが，我々の知る限り，深層学習アプリケーションに特化したワークロードは公開されていない．我々が有する深層学習専用システムにおけるワークロードを公開することにより，将来の深層学習アプリケーションの研究開発向けシステムにおける，要件定義やスケジューリングポリシ設定に貢献できると考えている．

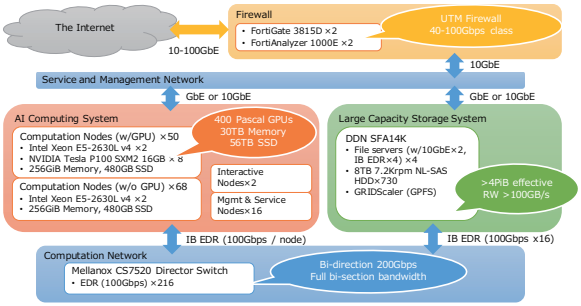


図 1: Architectural Overview of AAIC

3. 対象システム: AIST Artificial Intelligence Cloud (AAIC)

産総研が運用する，AIST Artificial Intelligence Cloud (AAIC) をワークロード公開・分析の対象とする．AAIC は AI 研究開発のための並列システムであり，ABCI [14] の前身となるシステムである．AAIC の利用者は産総研の研究者・技術者だけでなく，大学，研究機関，企業からの利用者も含まれる．AAIC は，AI のアルゴリズム，分散深層学習のような基礎研究だけでなく，ジオサイエンスやバイオサイエンス，機械翻訳などのアプリケーションも含めた，幅広い AI 研究に利用されている．しかしながら，利用者の多くは深層学習技術の発展に伴い並列システムの利用を開始したものであるため，並列システムの使い方に慣れているわけではない．

AAIC のアーキテクチャを図 1 に示す．AAIC は 50 台の GPU を搭載したノード (GPU ノード) と，68 台の GPU を搭載しないノード (CPU ノード) から構成され，それらがシングルレイルの InfiniBand EDR により接続されている．4PB の GPFS 共有ファイルシステムを有す．GPU ノード内には 8 機の NVIDIA P100 GPU があり，それらは NVLink により接続されている．

AAIC は 3 種類の計算サービスを提供しており，各サービスへの資源配分は利用率により決めている．最初のサービスはバッチジョブ実行サービスである．AAIC では GPU ノード，CPU ノードそれぞれに独立したキューを用意している．本研究での解析対象である，GPU ノード用のキュー（GPU バッチシステム）について次の段落で詳細を示す．2 つ目のサービスは，Apache Spark や Jupyter Notebook などの，対話型分散データ処理環境をオンデマンドに提供するサービスである．IBM Spectrum Conductor を用いている．最後のサービスは，OpenStack による仮想マシン提供である．このサービスは，特定のバージョンのソフトウェアしか提供できないという，最初の 2 つのサービスの欠点を補うためにあり，利用者に対して root 権限を付与した Ubuntu と CentOS の仮想マシンを提供する．

GPU バッチシステムの詳細を述べる．ジョブスケジューラには，NEC 製の NQSII を用いている．利用状況に応じ

て，スケジューラは 32〜38 台の GPU ノードを管理する．スケジューリングポリシには FCFS とバックフィルを採用している．表 1 に示す通り，それぞれ異なる最大並列度と最大実行時間を持つ 4 つのキューがある．ジョブはノード内の任意個の GPU を使うことができ，ノード内 8GPU 全て使い切るまで，複数のジョブが同時に 1 ノード上で実行されることもある．複数ノードジョブは `Normal` キューでのみ実行でき，複数ノードジョブにはノード内の全 GPU が割り当てられる．深層学習の長時間学習計算を実行できるよう，最大実行時間は長く設定されている．各ジョブにより消費された GPU 時間に対して課金を行なっている．つまり，費用は使用 GPU 数と実行時間に比例して増える．

GPU バッチシステムは深層学習の学習計算に主に用いられている．GPU バッチシステムのワークロードを解析することで，深層学習の研究者や開発者がどのように GPU クラスタを使用するか，理解する助けとなる．

4. ワークロードの詳細

AAIC の GPU バッチシステムに 2017 年 7 月 14 日から 2018 年 12 月 31 日の，1 年半ほどの期間に投入されたジョブからなるワークロードの分析を行った．総ジョブ数は 459,545，総利用者数は 106 である．各ジョブについて，表 2 の 13 項目からなる属性を収集した．

ジョブで使用されたキューの詳細を図 2 に示す．`Normal` が支配的であることがわかる．理由としては，`Normal` はデフォルトキューであること，また複数ノードジョブはこのキューでのみ許可されていることにあると考えられる．5 章の分析では全キューを対象とした．図 3 はジョブの終了状態を示す．5 章では成功ジョブのみを対象とした分析もあるが，成功ジョブは全体の 83%を締めるため，それら分析は利用者の傾向を正しく示すと考えている．

四半期ごとの投入ジョブ数の推移を図 4 に示す．AAIC の運用開始直後であるため，`17Q3` のジョブ投入数は最も小さい．また，`18Q3` 以降，ジョブ投入数は大きく下落している．これは，同じ AI 研究開発を目的とした ABCI の運用が開始したためである．図 5 にアクティブ利用者数，グループ数の推移を示す．アクティブ利用者とは，その四半期にジョブを投入した利用者であり，アクティブグループはその利用者を含むグループを意味する．`18Q2` におけるアクティブ利用者・グループ数の下落は，年度切り替わり時に AAIC の利用を停止した利用者・グループがあるためである．

5. ワークロード分析

深層学習の研究者・開発者の GPU 資源の利用状況の把握を目的に，上記のワークロード（AAIC ワークロード）の分析を，5 つの視点より行った．

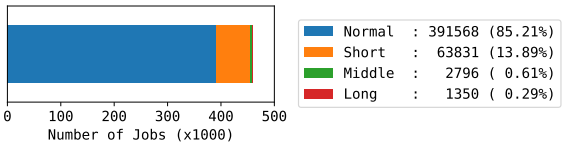


図 2: Breakdown of Used Queues

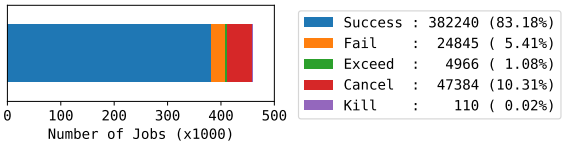


図 3: Breakdown of Termination Code

図 4: Transition of Number of Jobs

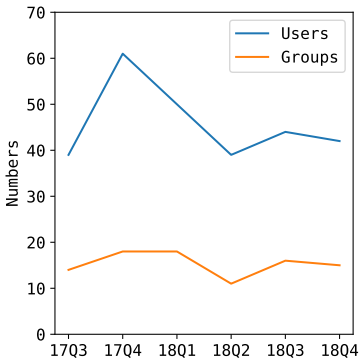


図 5: Transition of Number of Account

5.1 並列度

TensorFlow や Pytorch などの近年の深層学習フレームワークでは分散並列学習がネイティブにサポートされており，また，分散学習用汎用フレームワークとして Horovod の開発も盛んである．これらを活用した分散学習が，AI の研究開発の場でどれだけ盛んに行われているか把握するために，使用 GPU 数の分析を行った．図 6 に使用 GPU 数ごとのジョブ数を示す．テスト実行等の短時間ジョブの影響を除外するため，60 秒未満の実行時間のジョブは集計から除外した．結果，360,756 ジョブを対象としている．

図 6a より，単一 GPU ジョブがジョブ数の大半を占め，実に 96%であることがわかる．また，99%のジョブが 1 ノードで実行されている．複数 GPU ジョブの利用が少ない理由は，学習の並列度効果を得ることが難しいことに対して，

表 1: AAIC GPU Batch Queues

Queue	#Nodes/Job	#GPUs/Job	Walltime Limit (Default, hour)	Walltime Limit (Maximum, hour)
Normal	1...38	1...8	1	72
Short	1	1	1	3
Middle	1	1...8	72	168
Long	1	1...8	72	336

表 2: Attributes of Jobs in AAIC Workload Log

Attribute	Meanings
Job ID	A unique sequential number assigned to each job.
User	User who submit the job.
Group	Group where job submitter belongs. A user can belong to multiple groups.
Queue	A queue where the job submitted.
Nodes	Number of used nodes.
GPUs	Number of used GPUs.
Requested Walltime	Walltime requested by submitter.
Termcode	Job termination code: success, fail, walltime exceeded, canceled by user or killed.
Submit Timestamp	Time when job was submitted.
Start Timestamp	Time when job started running.
Finish Timestamp	Time when job ended.
Wait Time	Duration to start running after submission.
Walltime	Actual walltime of the job.

使用 GPU 数に比例して課金されることにあると考えられる．実際に，実行時間が多少長くとも，課金額に対するインセンティブがないために，1GPU のみを使用し続けている利用者がいることがアンケートよりわかっている．図 6b と 6c より，複数 GPU ジョブにおいては利用者は 2 べきの数の GPU 数を望むことがわかる．図 6d は，四半期ごとのジョブ数と割合の変化を示す．四半期ごとにジョブ数は異なるが，全体として 85%以上が単一 GPU ジョブである．

関連研究で挙げた既存研究と比較すると，AAIC ワークロードでは単一 PE（AAIC の場合は GPU）を用いたジョブが極端に多い．複数 GPU ジョブの場合，2 べきの数を好む傾向は，既存研究でも観測されている傾向である．古典的な HPC アプリケーションでは問題分割に制限があるものもあったが，深層学習計算ではフレームワークにそのような制約があるものは少ないため，スケーラビリティの評価をわかりやすく行うことが目的であると考えられる．

5.2 実行時間指定の正確性

バックフィルの動作に影響を与えるため，利用者が指定するジョブ実行時間の正確性はスケジューリングにおいて重要なパラメータである．しかしながら，多くの研究にて正確性は低いと報告されている [3], [4], [15], [16]．AAIC の利用者には AI 研究の必要性から HPC システムの利用を始めた者も多く，バッチスケジューリングシステムの利用が初めてという者もいる．彼らにとっては実行時間を指定することは新しい概念である．この分析の目的は，そのような利用者を抱えつつも，1 年半の運用を通じて実行時間の指定の正しさが向上したか確認することにある．

実行時間の指定の正確性を WRA (Walltime Request Accuracy) という指標で評価する．WRA はジョブごとに定義され，次の式により定義される 0〜1 の値を持つ指標である．

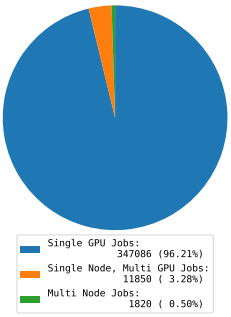
$$WRA = \frac{Actual\ Walltime}{Requested\ Walltime}$$

WRA を計算するジョブは，60 秒以上の実行時間を持つ成功ジョブ 308,555 個とした．全期間のジョブの WRA の分布を図 7a に示す．平均は 0.107 (緑三角)，標準偏差は 0.150，中央値は 0.040 (オレンジ横棒) であった．3/4 のジョブの WRA が 0.2 未満であることがわかる．四半期ごとの WRA の推移を図 7b に示す．`17Q4` より中央値，平均値の改善は見られるが，改善幅は小さい．AAIC ワークロードの WRA は，関連研究にあげた既存研究と比較しても小さい．この理由の 1 つは，36.54%のジョブが最大実行時間を指定して投入されていたことにある．

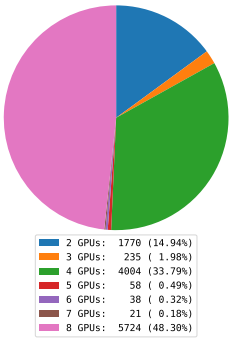
5.3 実行時間

深層学習の学習計算は，GPU を使用することで短縮はされるが，一般に長時間要するタスクであると認識されている．そのため，表 1 の通り，AAIC の初期設計時から，長時間実行可能なキューを用意している．実際の深層学習のアプリケーションやアルゴリズムの研究開発において，この認識が真であるか確認するために，ジョブの実行時間についての分析を行った．

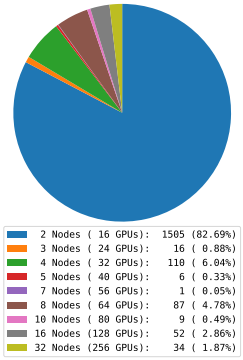
全ジョブ，および単一 GPU ジョブ，単一ノード複数



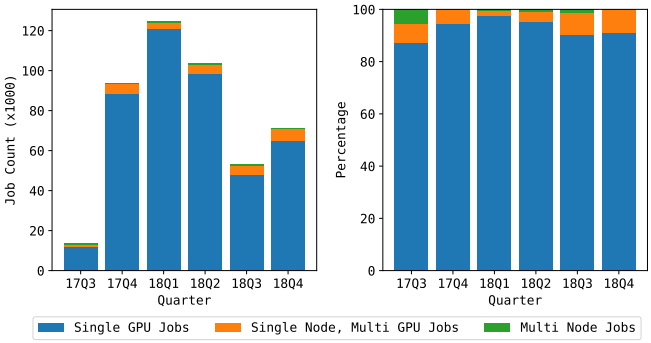
(a) Ratio of Single GPU, Single Node Multi GPU and Multi Node Jobs



(b) Breakdown of Single Node Multi GPU Jobs



(c) Breakdown of Multi Node Jobs

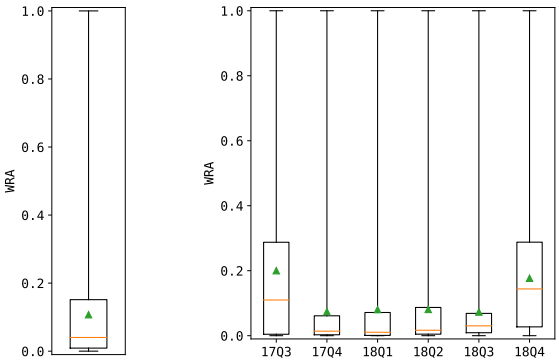


(d) Transition of Job Count and Ratio

図 6: Analyses of Degree of Parallelism based on Job Count

表 3: Walltime Statistics on AAIC and RICC Workload. Unit is Hour

System, Job type	Average	STD	Median	Min	Max
All Jobs/AAIC	2.730	8.255	0.470	0.017	332.678
Single GPU Jobs/AAIC	2.461	7.429	0.460	0.017	332.678
Single Node, Multi GPU Jobs/AAIC	14.062	22.038	5.576	0.017	276.374
Multi Node Jobs/AAIC	4.642	9.640	0.187	0.017	53.798



(a) All Period (b) Quarterly Transition

図 7: WRA Distribution

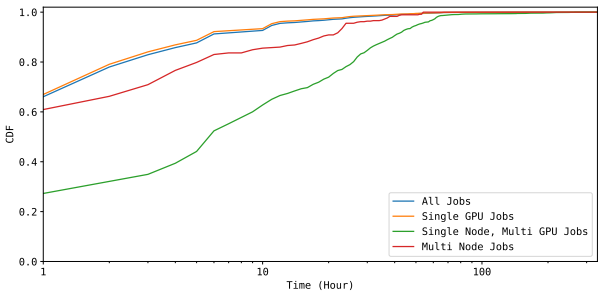


図 8: Distribution of Walltime

GPU ジョブ，複数ノードジョブの実行時間の累積分布を図 8 に示す．表 3 には実行時間の統計を示す．60 秒以上の実行時間の成功ジョブ 308,555 個を対象とした．並列度に応じて分布は異なり，単一 GPU ジョブの方が複数 GPU ジョブよりも実行時間が短い傾向にあることがわかる．例えば，単一 GPU ジョブの 84%は 3 時間以下の実行時間であるが，単一ノード複数 GPU ジョブにおいて 3 時間以下

の実行時間のジョブはわずか 35%である．24 時間以上の実行時間のジョブに関しては，単一 GPU ジョブで 2%，単一ノード複数 GPU ジョブで 22%，複数ノードジョブで 4%であった．母集団となるジョブ数がそれぞれ大きく異なること，およびアプリケーション区別の判別を行なっていないため断定はできないが，利用者が複数 GPU を利用する目的として，問題を Weak Scale させることは重要な位置付けにあると考えられる．

5.4 並列度と実行時間の相関

上記の考察をより深めるために，ジョブの並列度と実行時間の相関を評価する．正の相関が得られれば，並列度に比例して実行時間も増えるということで，Weak Scale の利用傾向を表し，負の相関が得られれば，Strong Scale の利用傾向を表すと考えられる．ここでは，スピアマンの順位相関係数と，ピアソンの積率相関係数を求めた．後者の計算では，You らの研究同様 [3]，例外値による影響を抑えるため対数変換させた値の相関を求めている．

60 秒以上の実行時間の成功ジョブ 308,555 個を対象とした．しかしながら，単一 GPU ジョブとそれ以外のジョブでジョブ数に大きな差があるため，それぞれのグループから 2000 ジョブをランダムに選択し，合計した 4000 ジョブに対して相関係数を計算した．その結果，スピアマン相関係数は 0.254，ピアソン相関係数は 0.219 であった．それぞれ，弱い正の相関を示している．Weak Scale の利用傾向があると言える．一方，You らの分析では負の相関が示されており，Strong Scale の傾向がある結果であった．

5.5 ジョブ投入

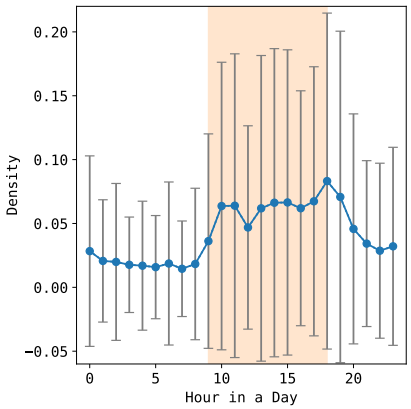
長時間のジョブ実行が可能なバッチシステムでの，利用者のジョブ投入の振る舞いを評価する．全てのジョブを対象に，You らが行なった方法と同じ方法で評価した [3]．

結果を図 9 に示す．図 9a は 1 日における 1 時間毎のジョブ投入数の推移である．1 日単位で毎時に投入されたジョブ数をカウントし，分布を計算している．図中の青点は全ての日におけるその時間の分布の平均値であり，灰色のバーは標準偏差である．オレンジのエリアは主な就業時間であり，標準偏差が大きく分散しているが，その時間帯にジョブが集中していることがわかる．図 9b は 1 週間における 1 日毎のジョブ投入数の推移である．分布は図 9a 同様に求めている．オレンジのエリアは平日であり，平日にジョブ投入が集中していることがわかる．

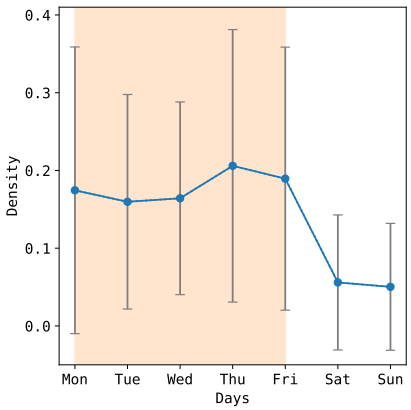
You らの研究と比較すると，平日の就業時間中にジョブ投入が集中する点は同じであるが，以下が異なる．

- You らの分析に比べ，AAIC ワークロードの方が各点における標準偏差が大きい
- You らの分析では 1 週間におけるジョブ投入は，週末に向けて明確な下落傾向が示されているが，AAIC ワークロードでは大きな変化は見られない

1 点目については，深層学習のハイパーパラメータ探索を行う際には，短時間に多量のジョブを投入する傾向が多く，投入時以外とのジョブ数のばらつきが大きいことが理由として考えられる．2 点目については，長時間ジョブはいつ投入しても結果が得られるまで待たさせるため，利用者は投入タイミングに無頓着なのかもしれない[*2]．



(a) Hourly Arrival in a Day



(b) Daily Arraival in a Week

図 9: Job Arrival Distribution

6. 考察

以上の分析結果より，深層学習向け GPU システムに対する，アーキテクチャ的要件，スケジューリング機能的要件についての考察を行う．

アーキテクチャ的要件に関しては，単一 GPU ジョブが支配的であったことから，GPU の演算性能向上，GPU メモリ性能・容量向上，ホストメモリから GPU メモリへの転送性能の向上が重要である．複数 GPU ジョブ実行時の通信性能に関しては，ジョブの大半が 16GPU 以下の利用であったため，少数 GPU・ノード間での高速通信が実現されれば，十分であると考えられる．しかしながら，複数 GPU ジョブ実行の目的にはアプリケーションの Weak Scale が

[*2] You らの分析対象システムの最大実行時間は 48 時間であり，木曜までに投入すれば大半はその週のうちに結果が得られる

あるため，通信性能だけでなく，学習データ読み込みのための IO 性能も同時に向上させる必要がある．

スケジューリング的要件に関しては，単一 GPU ジョブが支配的であったことより，NVIDIA DGX-2 のような多数 GPU を搭載したシステムにおいても，論理的にシステムを分割し，GPU 単位で利用者に提供できる仕組みが必要である．そのような機能は Slurm [17] や Univa Grid Engine [18] ですでに実現されているが，複数のジョブが 1 ノードを共有する際には，他ジョブのデータ等が閲覧できないなどの隔離構成する必要がある．また，ハイパーパラーメータ探索する際に多数のジョブが投入されうるので，特定利用者による長期間資源占有を防ぐために，利用者毎の最大ジョブ数の設定が必要になると考えられる．複数 GPU ジョブの利用を促進するために，複数 GPU を用いたジョブの課金を優遇させることは 1 つの方法である．ただ，逐次ジョブを複数まとめただけのジョブを判別し，そのジョブは優遇しない等の作り込みをしないと意味はない．

7. より正確な深層学習ワークロード取得に向けて

今後の深層学習向けシステム設計やスケジューリング・資源割り当て方針の設計に役立てることを目的に，AAIC ワークロードの分析を行なったが，次の欠点があり，結果は正確性に欠ける可能性がある．

- ジョブが本当に深層学習の学習計算である保証がない
- 複数 GPU ジョブにおいて，使用されている GPU 数に保証がない

1 つ目に関しては，産総研としてはジョブ内部の情報は取得しない方針であるため完璧な情報は取得できないが，ABCI ではこれら問題を解決すべくモニタリングを実装している．

1 点目については，ABCI ではジョブ属性として表 2 に挙げた項目以外に次の項目も記録している．

(1) 使用したノードと CPU コア，GPU ID
(2) 使用したソフトウェアモジュール (運用が Environment Modules で提供のものに限る)

(1) は次で説明する資源モニタリングと合わせ，各ジョブにおける CPU コア，GPU 利用率を取得するためにある．(2) は，ジョブが依存するソフトウェアを把握することで，深層学習ジョブであるかどうかを判別するためにある．(2) の記録の本来の目的は，運用としてサポートすべきソフトウェアを把握するためにあり，module load 時に読み込まれたソフトウェアモジュール名をジョブ属性として記録している．この情報を用いることで，例えば Python を読み込んでないジョブは深層学習でない，などの判断を行うことができる．

2 点目に関しては，ABCI 全 1088 台の計算ノードの GPU コア利用率，GPU 利用率，消費電力といった情報を 1 分間隔で Prometheus ベースの時系列データベースに蓄積している．この情報と上記 (1)，およびジョブ開始時刻・終了時刻の情報を用いることで，1 分以上の実行時間のジョブに関しては，各資源の利用率を取得できる．これにより，例えば大容量メモリを使うためにノードを占有利用したが GPU は使っていないジョブを除外できる．

8. まとめ

大規模 GPU クラスタにおける深層学習ワークロードの傾向を把握することを目的に，我々が運用する AI 研究開発向けシステム AAIC の GPU バッチシステムのワークロードの分析を行なった．そのワークロードを，並列度，ジョブ実行時間指定の正しさ，実行時間，並列度と実行時間の相関，ジョブ投入傾向の 5 つの点で評価し，既存のワークロードと比較考察を行なった．既存ワークロードとの特徴的な違いとしては，単一 PE のみを使用するジョブが支配的であり，全ジョブ数の 96%以上であること，並列度と実行時間には弱い正の相関があり，利用者は Weak Scale 目的で複数 GPU を用いている傾向があると推測されること，であった．

今後の課題としては，深層学習の学習計算のワークロードの品質を向上させることにある．7 章に挙げた通り，ABCI ではより正確にワークロードを取得できるようモニタリングシステムを実装している．十分な量のジョブ履歴が蓄積されたのちに，データの精査を行いワークロードとして整備し，同様な分析を行うとともに公開する予定である．

謝辞 この研究の一部は，NEDO 次世代人工知能・ロボット中核技術開発の一環として実施した．

参考文献

[1] Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., Mahesh, A., Matheson, M., Deslippe, J., Fatica, M., Prabhat and Houston, M.: Exascale Deep Learning for Climate Analytics, *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '18, pp. 51:1—-51:12 (2018).

[2] Li, H., Groep, D. and Wolters, L.: Workload Characteristics of a Multi-cluster Supercomputer, *Job Scheduling Strategies for Parallel Processing*, pp. 176–193 (2005).

[3] You, H. and Zhang, H.: Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer, *Job Scheduling Strategies for Parallel Processing*, pp. 253–271 (2013).

[4] Rodrigo, G. P., Östberg, P.-O., Elmroth, E., Antypas, K., Gerber, R. and Ramakrishnan, L.: Towards understanding HPC users and systems: A NERSC case study, *Journal of Parallel and Distributed Computing*, Vol. 111, pp. 206–221 (2018).

[5] Feng, J., Liu, G., Zhang, J., Zhang, Z., Yu, J. and Zhang, Z.: Workload Characterization and Evolutionary Analyses of Tianhe-1A Supercomputer, *Computational Science – ICCS 2018*, pp. 578–585 (2018).

[6] Schlagkamp, S., da Silva, R. F., Allcock, W., Deelman, E. and Schwiegelshohn, U.: Consecutive Job Submission Behavior at Mira Supercomputer, *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '16, pp. 93–96 (2016).
[7] Schlagkamp, S., da Silva, R. F., Renker, J. and Rinkenauer, G.: Analyzing Users in Parallel Computing: A User-Oriented Study, *2016 International Conference on High Performance Computing Simulation (HPCS)*, pp. 395–402 (2016).
[8] Ren, Z., Wan, J., Shi, W., Xu, X. and Zhou, M.: Workload Analysis, Implications, and Optimization on a Production Hadoop Cluster: A Case Study on Taobao, *IEEE Transactions on Services Computing*, Vol. 7, No. 2, pp. 307–321 (2014).
[9] : Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/`.
[10] Feitelson, D. G., Tsafrir, D. and Krakov, D.: Experience with using the Parallel Workloads Archive, *Journal of Parallel and Distributed Computing*, Vol. 74, No. 10, pp. 2967–2982 (2014).
[11] : The Grid Workloads Archive, `http://gwa.ewi.tudelft.nl/`.
[12] Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L. and Epema, D. H. J.: The Grid Workloads Archive, *Future Generation Computer Systems*, Vol. 24, No. 7, pp. 672–686 (2008).
[13] : JSSPP Workloads Archive, `http://jsspp.org/workload/`.
[14] : ABCI, `https://abci.ai/`.
[15] Bailey Lee, C., Schwartzman, Y., Hardy, J. and Snavely, A.: Are User Runtime Estimates Inherently Inaccurate?, *Job Scheduling Strategies for Parallel Processing*, pp. 253–263 (2005).
[16] Tsafrir, D., Etsion, Y. and Feitelson, D. G.: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 6, pp. 789–803 (2007).
[17] Yoo, A. B., Jette, M. A. and Grondona, M.: SLURM: Simple Linux Utility for Resource Management, *Job Scheduling Strategies for Parallel Processing*, pp. 44–60 (2003).
[18] : Univa Grid Engine, `http://www.univa.com/products/`.