Vol.2018-HPC-163 No.15
2018/3/1

AIクラウドでの深層学習ワークロード解析を通じたジョブスケジューリング改善に向けた考察

滝澤 真一朗[1,a)]　佐藤 仁[1]　高野 了成[1]　谷村 勇輔[1]　小川 宏高[1]

概要：深層学習ワークロードを実行する処理基盤として，GPU クラスタの利用が進んでいる．しかし，深層学習プログラムは少ない資源を長時間占有利用すること，学習時間の事前予測が困難であることなど，大規模並列処理を行う，従来の科学技術計算ワークロードとは異なる傾向を持つ．我々は深層学習ワークロード処理専用に構築された GPU クラスタ上で，5 ヶ月間に実行されたワークロードを解析した．その結果，95%を超えるジョブが Single GPU による深層学習であること，ジョブスケジューラへのジョブ投入時に指定する実行時間見積と実際の実行時間に大きな差があることが確認できた．現状では問題は顕在化しないが，今後分散深層学習が普及するにつれ，実行時間見積と実実行時間に差があるままでは資源利用率の低下が予想されるため，将来の運用課題として解決していく．

1. はじめに

近年，AI に関する研究開発が，自動運転，ロボットの自動操縦，医療，創薬などの様々な分野で盛んに行われている．特に多層構造のニューラルネットワークを用いた機械学習手法である深層学習が注目されており，画像・音声・映像認識，自然言語処理などへの応用が見込まれている．深層学習プログラムでは学習データを用いてモデルを構築し，そのモデルを用いた推論を行う．モデル構築はニューラルネットワークを用いた Iterative な処理として実装され，主計算が密行列演算であるため，計算に GPU が広く用いられている．深層学習用のネットワーク定義，GPU を用いた学習の高速化，推論処理を容易に行うため，Caffe や TensorFlow，Chainer など多くの深層学習フレームワークが提案されている．これらフレームワークは提案当初は Single GPU や Single Node のみを対象としていたが，学習の高速化やネットワークの複雑化，学習データの増大に伴い，フレームワーク自身が分散処理に対応し，分散深層学習のサポートを始めている．これに伴い，深層学習プログラムの開発者の開発評価環境はデスクサイドシステムから，並列システムへと移り変わりつつある．

分散深層学習の高速化のためには，高性能なストレージと高速なネットワークからなるクラスタ型計算機資源が必要である．そのような環境を自前で用意するのは困難であるため，ビッグデータ処理基盤であるクラウドと高性能計算によるシミュレーション基盤であるスーパーコンピュータとを融合した共有コンピューティングインフラである「AI クラウド」の構築が求められる．AI クラウドにおいて，計算資源の有効活用のためには，資源分割・時間分割して複数の利用者からの複数の計算要求を同時並行して処理することが重要である．この目的には，従来よりスーパーコンピュータで用いられていたバッチジョブスケジューリングシステムを用いることができる．しかしながら，深層学習プログラムは少数計算資源を長時間占有利用すること，学習時間の事前予測が困難であることから，従来のスーパーコンピュータのスケジューリングポリシにて効率良く資源管理できるか定かでない．システム性能は，システム上で実行されるワークロードに大きく依存するため，ワークロードに適したスケジューリングポリシの採用が重要である [1].

そこで本研究では，クラスタ型計算機システム上での深層学習ジョブのワークロードを解析し，ジョブスケジューリングによる資源利用改善に向けた手法についての考察を行う．ワークロード解析は産総研が有する AIST Artificial Intelligence Cloud（以降 AAIC）の 5 ヶ月間の利用情報を対象とした．現状の利用者のシステム利用方法を把握するという観点から解析を行い，Single GPU / Multi GPU / Multi Node 深層学習のそれぞれの利用状況，ジョブの実行時間見積の正しさ，の 2 点についての解析を行なった．その結果，現状では 95%を超えるジョブが Single GPU ジョブであること，利用者によるジョブ実行時間見積と実際の

[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

ⓒ 2018 Information Processing Society of Japan

実行時間に大きな乖離があることが確認できた．分散深層学習を普及させるには，各種フレームワークでサポートしているリモートプロセス起動手段をシステムでサポートすること，並列実行に対する課金やスケジューリングプライオリティにインセンティブを設けることが考えられる．一方，分散深層学習が普及した場合，ジョブ実行時間見積と実実行時間に乖離があると資源利用率の低下が起こり得る．この状況を防ぐためには，正しく設定することを利用者に教育するだけではなく，正しく設定するインセンティブを設けること，および，深層学習フレームワークが備えている学習状態保存・再開機能を活用することが重要となる．以上の考察を元に，今後の AAIC のスケジューリングを改善するとともに，産総研が構築中の ABCI システムのスケジューリングポリシ設定に反映させる [2].

2. AIST Artificial Intelligence Cloud

産総研では，人工知能のための研究開発，および上記の AI クラウドを実現するためのテストベッド構築を目的としたシステム AAIC を運用している [3]．AAIC の利用者には，産総研研究者を中心とした学術分野で研究を行っている研究者，産総研と共同研究を行っている民間企業の研究者・技術者がいる．

AAIC は 50 台の GPU 搭載ノードと 68 台の GPU 非搭載ノードからなる．CPU，メモリ，ネットワークインターフェースは共通しており，それぞれ Intel Xeon E5-2630Lv4 を 2 機，メモリは 256GB，InfiniBand EDR を 1 機搭載している．GPU 搭載ノードでは NVIDIA Tesla P100 を 8 機搭載している．システム管理ソフトウェアには，スパコン型の資源分割・時分割して利用者の計算要求（以降ジョブ）をスケジュール実行するバッチジョブスケジューラと，クラウド型の仮想マシンによる資源提供サービスがある．ジョブスケジューラには NQSII を用いており，仮想マシン提供サービスには OpenStack を用いている．各種深層学習フレームワークが導入されており，Environment Modules により複数のフレームワーク・バージョンを容易に切り替えて利用できることに加え，コンテナによるイメージ単位でのソフトウェアスタック配備の導入を進めている [4].

表 1 に 3 章以降で解析する，GPU 資源を提供するバッチジョブ実行サービスのキュー構成を示す．運用状況に応じてキューには合計 30〜40 ノード提供されている．標準キューは gq であり，要求実行時間に応じて他キューを用意している．AAIC は計算ノード数が多くないながらも，深層学習プログラムは 1 度の実行に長時間有するものもあるため，最大実行時間は長く取っている．キュー配下のノードのインタラクティブ利用は無効にされており，バッチ利用のみ可能である．ジョブに割り当てられた計算ノード間での SSH 等のリモートログインは許可されておらず，リモートプロセスの立ち上げには MPI を使う必要がある．使用 GPU 数，ノード数について次のルールがある．

- 8 未満の GPU を使用するジョブには，ノードを空間分割し，資源提供する．ジョブは割り当てられた GPU を占有利用できる．使用 GPU 数はジョブ投入時に利用者が指定する．オーバーコミットはしない．
- 複数ノード使用ジョブには各ノードを占有割り当てする．

3. AAIC 上での深層学習ワークロード分析

AAIC は上述の通り，人工知能のための研究開発用クラスタであり，用途もそれに限定されている．そのため，GPU を用いたジョブのほぼ全てが深層学習に関連する計算である．AAIC 上で実行された GPU ジョブのワークロードを分析することにより，ジョブスケジューラで管理された共有計算機システム上での深層学習アプリケーション利用者の利用傾向を把握することができる．

本研究では，深層学習アプリケーション利用者の資源要求，および実利用情報を元に，以下の点についての分析を行う．

分散深層学習の利用状況 Multi Node，Multi GPU を利用する分散深層学習をサポートしたフレームワークの開発が盛んに行われている．データ並列処理や InfiniBand など高速通信ネットワークを用いることにより学習の高速化が期待できる．技術の普及状況を調査分析することで，普及のために運用側がアクションをとるべきかの判断材料とする．

深層学習ワークロードをバッチジョブ管理する際の課題 深層学習アプリケーションは，大容量高速データアクセス，GPU による高スループット演算，さらには分散深層学習では高速ネットワークが必要になる等，システムアーキテクチャ要件には HPC システムと共通するものが多い．HPC システムでは利用者の計算はバッチジョブスケジューラにより管理されている．同等なシステムアーキテクチャを要求する深層学習向けシステムでも，HPC システムと同等なシステムソフトウェアによりジョブ管理することが考えられる．

バッチジョブスケジューラでは，利用者が申告したジョブの実行時間（以降，実行時間見積）を参考にジョブスケジュールする．実行時間見積の正しさがバックフィル等の資源利用率改善アルゴリズムの貢献に大きく影響するが，正しく設定することは従来の HPC アプリケーションでも困難であった．一方，深層学習アプリケーションは精度が収束するまでの Iterative 処理であること，ハイパーパラメータの設定により収束するまでの計算時間も変動するため，より設定が困難となる．現状どのように実行時間見積が設定されているか把握することで，正しく設定することを普及するために運用側がアクションをとるべきかの判断材料と

表 1: AAIC GPU バッチジョブ実行サービスのキュー構成

Queue Name	Available GPUs	Elapsed Time Limit (Default)	Elapsed Time Limit (Maximum)
gq	1-8	1 Hour	72 Hours
gq-short	1-8	1 Hour	3 Hours
gq-middle	1	72 Hours	168 Hours (7 Days)
gq-long	1-8	72 Hours	336 Hours (14 Days)

する．

3.1 解析データの用意

本研究では産総研 AAIC GPU バッチジョブ実行サービスにて，2017 年 7 月 14 日から 2017 年 12 月 31 日までに実行されたジョブを解析した．データの信頼性を上げるために，以下の条件に該当するジョブを除いた，計 55,127 個のジョブを対象とした．

- パラメータ指定が間違えているジョブ
- システム異常が原因と思われるジョブ
- 実行時間が 60 秒以下のジョブ

55,127 ジョブのうち，正常終了したジョブは 53,488，実行時間超過したジョブは 1,639 である．また，アレイジョブはアレイ数でカウントしている．ジョブを投入した利用者数は 64，うちアクティブ利用者数[*1]は 30 であった．

ジョブデータはジョブスケジューラの課金，および資源利用状況データベースから取得した．今後の研究での利用を想定し，表 2 に示す項目からなるデータベースとして作成した．データベースには SQLite を用いている．現状では GPU/GPU/メモリの資源利用については，利用者要求量しか取れておらず，使用資源量の取得については今後の課題である．また，利用されている深層学習フレームワークの種類，ジョブ内容についての情報は取得していない．

表 2: ジョブ情報データ

項目名	データ型
ジョブ ID	Integer
ジョブ種別	Integer
タスク ID	Integer
ユーザ名	Text
グループ名	Text
Priority	Integer
キュー	Text
要求ノード数	Integer
要求 CPU 数	Integer
要求 GPU 数	Integer
要求 Mem 量	Integer
実行時間見積	Integer
使用ノードリスト	Text
終了コード	Integer
投入時刻	Integer
実行時刻	Integer
終了時刻	Integer
待ち時間	Integer
実行時間	Integer

3.2 総ジョブ数に対する並列・分散深層学習ジョブ数の割合

図 1 に分散深層学習ジョブの利用状況を示す．図 1a は各ジョブタイプごとのジョブ数，および割合を示している．図に示される通り，ジョブの 95%が Singl GPU を用いた深層学習であり，並列・分散深層学習の利用は相対的に少ない．この理由として以下が考えられる．

- 研究者は深層学習アルゴリズムの基礎研究のためのハイパーパラメータを変更した計算を中心に行っており，企業研究者・技術者は既存のアルゴリズム・ソフトウェアを中心とした実アプリへの応用を中心とした研究開発を行っていると想定される．このような用途の場合，技術として新しい分散深層学習よりも，より普及している Single GPU を用いた学習が使われると考えらえる．
- ChainerMN や Distributed TensorFlow など分散深層学習フレームワークは並列化を意識して使用する必要があり，利用の敷居が高い．
- AAIC ではリモートプロセスを MPI でキックする必要があり，ChainerMN など MPI で並列実装されている分散深層学習フレームワークは親和性が高い．一方，Distributed TensorFlow など他手段でリモートプロセスを起動するフレームワークでは，MPI で起動プログラムを実装する等が必要となり敷居が高い．

図 1b は，Multi GPU ジョブにおける使用 GPU 数の割合を示している．2 べきの数の GPU 利用が多い．ただし 8GPU の結果には，ノード占有して使用するジョブも含まれており，8GPU 全て使用するジョブである保証はない．

図 1c は Multi Node ジョブにおける使用ノード数の割合を示している．多くが 2 ノードジョブである．ジョブの大部分を占める Single GPU で資源の多くが占有されているため，使用ノード数が大きくなるほど Multi Node ジョブの待ち時間が増加することが原因の 1 つと考えている．また，AAIC の各 GPU ノードは InfiniBand EDR を 1 ポート (100Gbps) しか持たないため，学習結果をノード間共有するための集団通信のスケーラビリティに乏しいことも影

[*1] 期間中，月平均 10 ジョブ以上投入している利用者をアクティブ利用者と定義した場合

ⓒ 2018 Information Processing Society of Japan

響していると考えている．

3.3 ジョブ実行時間見積の正しさの評価

ジョブの大部分を占める Single GPU ジョブのみを分析対象とする．図 2 に，キュー毎のジョブ実行時間（利用者の見積と実使用時間）とジョブ数の割合を累積ヒストグラムで示す．横軸はジョブの見積，および実際の実行時間（単位は時間）であり，縦軸は総ジョブ数に対する比率を表す．薄い青が実行時間見積，濃い青が実際の実行時間を表す．赤線は実行時間指定がない場合に適用されるデフォルト値である．なお各キュー毎のジョブ数はそれぞれ gq が 49,938，gq-short が 2,137，gq-middle が 446，gq-long が 116 である．

理想は実行時間見積と実際の実行時間がほぼ等しくなることであるが，結果が示す通り，両者には大きな乖離がある．例えば gq の場合（図 2a），およそ 50%のジョブが 1 時間以内に終了しているが，実行時間見積として 1 時間以下を指定しているジョブは 2%程度しかない．同様に，およそ 70%のジョブが 72 時間を実行時間見積として指定しているが，実際に 70 時間以上実行されたジョブは 2%程度しかない．

実行時間見積と実実行時間の乖離を，以下の評価式を用いて数値で評価する．

$$Eval\_Value = \frac{\text{実実行時間}}{\text{実行時間見積}} \quad (1)$$

実行時間見積を超えるジョブは打ち切られるため，実実行時間 <= 実行時間見積 である．この評価値は 0 から 1 までの値を取り，数値が大きいほど見積時間と実時間が近く，正しく設定されていることを示す．表 3 にキュー毎の評価値の最小値，最大値，平均，分散，中央値を示す．箱ひげ図で示した結果を図 3 に示す．オレンジのバーは中央値であり，緑の三角は平均値である．平均値，中央値を見てわかる通り，評価値は小さく，指定の正確さに欠ける事が分かる．gq-middle，gq-long では平均値・中央値共に大きくなっているが，一方で標準偏差も大きくなっており，ばらつきも大きい．実行時間見積の指定が正しくないことの理由は，本質的に深層学習プログラムの実行時間の予測が困難であり，実行時間指定が足りず打ち切られるよりは，長時間指定することを利用者が選択しているからと思われる．実際，どのキューにおいても 50%以上のジョブは最大実行時間を指定して投入されている．

文献 [5] では，スーパーコンピュータにおける科学技術計算の実行時間見積の正しさを式 1 と同じ指標で評価している．文献中には正確な平均・分散の値は記載されていないが，調査対象キューの評価値の平均は 0.3 程度である．表 3 のキューごとの平均値の平均は 0.269 であり，深層学習ワークロードの実行時間見積の正しさは科学技術計算ワークロードのそれよりも若干低い程度である．しかしながら，文献 [5] では最大実行時間が 48 時間であり，見積が外れた場合に生じる資源遊休時間は我々の環境よりも少ない．

そのほか，図 2 からは次の知見が得られる．

- gq-short のジョブのほぼ全ての実実行時間が 1 時間以下である（図 2b）．実行時間見積と実実行時間に乖離があるため，指定できる最大実行時間をより短く設定することにより，スケジューリング効率向上に貢献する可能性がある．
- gq-long にて，実実行時間が 72 時間以下のジョブが 60%ほどある（図 2d）．これらジョブは gq-moddile で実行することも可能である．これらを最大実行時間の短い gq-moddile で実行することにより，上記同様にスケジューリング効率向上に繋がる可能性がある．

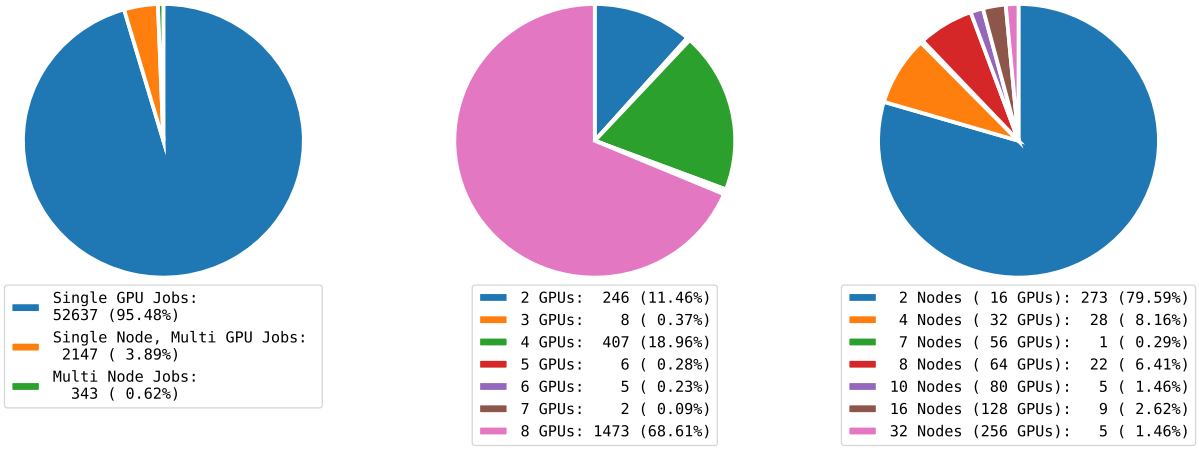
4. 考察

4.1 分散深層学習の利用普及に向けた運用アプローチ

3.2 章に示した通り，現状での AAIC における分散深層学習の利用は少ない．一方，分散深層学習を用いることで精度を保ちつつ学習時間を短縮する研究開発，事例もあるので，同技術が普及することで深層学習を応用した研究開発の加速が期待できる [6]．本章では，分散深層学習の実行を増やすために，システム運用側が取れるアプローチについての議論を行う．

分散深層学習フレームワークは，フレームワークごとにリモートプロセス起動の方法が様々に異なる．例えば，ChainerMN は MPI だが，MXNet は MPI, SSH, YARN 等様々に対応しており，TensorFlow は規定がない（SSH にて個々のホストで起動する方法が最も簡単）．AAIC のようなバッチジョブスケジューラで管理されたシステムでは，スケジューラがプロセスのトラッキングを行うために，一般にはスケジューラが指定する方法でしかリモートプロセスを起動できない．システム改修が求められうるが，SSH など多くのフレームワークサポートする，リモートプロセス起動手段を提供することが普及の第 1 歩として重要である．

スケジューリング，課金に対して優遇措置を行う方法も考えられる．例えば，Multi Node を用いたジョブはプライオリティを上げる方法，ノード時間積が等しい Single GPU ジョブよりも Multi Node ジョブの課金額を割り引く方法が考えられる．実際，西川らは FOCUS スパコンにて，後者のような課金割引を行なっている [7]．このような並列化を行うことに対するインセンティブを運用が行うことである．これにより，分散深層学習を行うジョブは優遇されるが，一方で，本来なら Single GPU のジョブとして実行されるべき多数のジョブを 1 つにまとめて，見かけ上並列実行するだけのジョブが優遇されることを防ぐのは困難である．そのようなジョブが優遇されると，スケジューリン

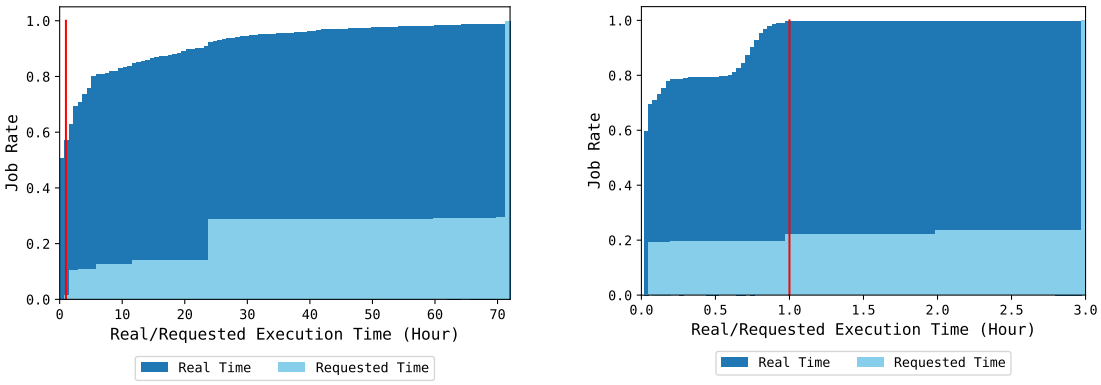


(a) Single GPU ジョブ，Multi GPU ジョブ，Multi Node ジョブの内訳

(b) Multi GPU ジョブの GPU 使用数の内訳

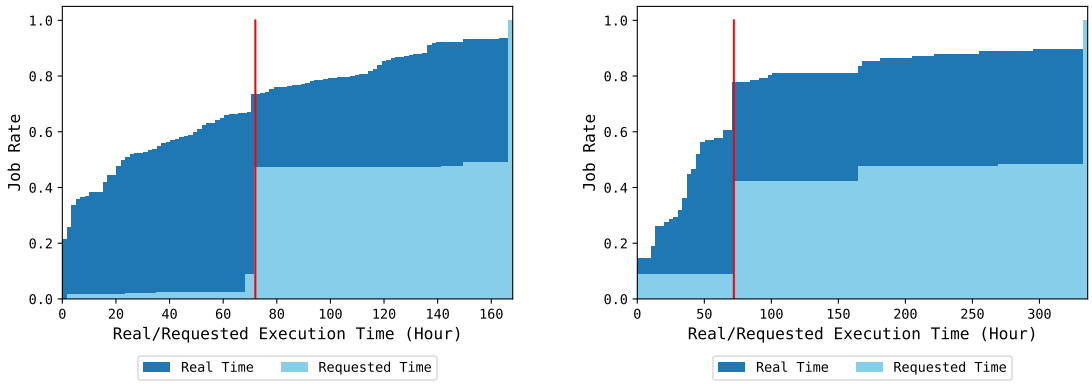
(c) Multi Node ジョブのノード使用数の内訳

図 1: 分散深層学習ジョブの利用状況



(a) gq

(b) gq-short



(c) gq-middle

(d) gq-long

図 2: ジョブ実行時間の見積と実際

グの公平性の欠如，集金額の減少へと繋がるので，インセンティブ設計は慎重に行う必要がある．

4.2 実行時間見積の正確さ向上に向けた運用アプローチ

3.3 章に示した通り，現状の AAIC での実行時間見積は正しくない．これにより資源利用効率の低下が起きていることが想像される．具体的には Single GPU ジョブと Multi GPU，Multi Node ジョブが混在した時に，図 4 に示すバックフィルが働かない状況が起きていると思われる．図中資源 3 には，ジョブ H が実際に必要とする実行時間以上の空

表 3: ジョブ実行時間見積の正しさの統計値

Queue Name	Minimum	Maximum	Mean	Std Dev.	Median
gq	0.000	1.000	0.103	0.194	0.032
gq-short	0.000	1.000	0.147	0.180	0.033
gq-middle	0.000	1.000	0.354	0.362	0.224
gq-long	0.000	1.000	0.472	0.407	0.245

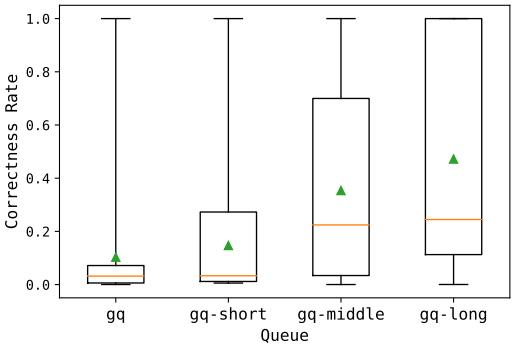


図 3: ジョブ実行時間指定の正しさ

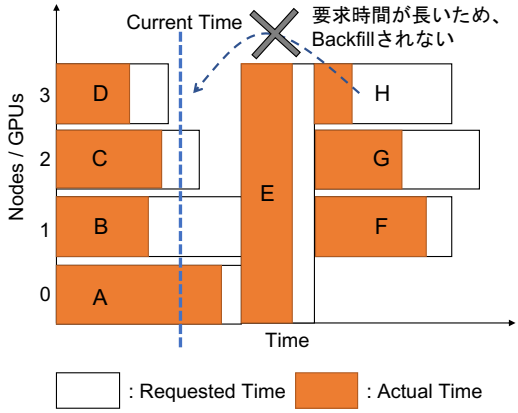


図 4: バックフィルが働かない状況

き時間があるが，利用者が指定した実行時間見積が空き時間よりも大きいためバックフィルされない．このため，利用者の実行時間見積をより正確にするための運用側からの取り組みが重要となる．

まず，実行時間見積を正しく設定することの重要さの周知が重要である．従来からの HPC システム利用者はバッチスケジューリングシステムの利用について長けているが，深層学習開発者は Single GPU を用いた開発から並列システムでの開発へと環境を変えているため，ジョブスケジューラの振る舞いについて詳しくないと想定される．実行時間見積の重要さを周知するとともに，ジョブ実行完了時には，見積時間と実使用時間との差を通知し，実行時間見積改善のフィードバックを促すことをシステム運用からのアプローチとして実現できる．

正しく指定することに対してインセンティブを設けることも考えられる．野村らは利用者の実行時間見積の指定を正確にするための方法として，課金インセンティブを設定する方法を提案している [8]．ジョブ課金パラメータの 1 つである実行時間について，一般に従来システムでは実実行時間に対してのみ課金を行っている．一方，野村らは，実実行時間と実行時間見積の両方を課金対象としており，実行時間見積が実実行時間の 3 倍以下であれば，従来よりも課金額が低くなる手法を提案している．他の方法として，正しく設定している率の高い利用者のジョブは高い優先度でスケジュールする等の方法も可能であると考えられる．

一方で，深層学習プログラムの実行時間予測が難しい，という事情もある．そこで，深層学習の状態保存・再開（Checkpoint/Restart）機能と，ジョブスケジューラの要求実行時間到達通知機能を組み合わせた方法の採用も検討する価値はある．深層学習の状態保存・再開機能は Chainer，Caffe，TensorFlow などの代表的な深層学習フレームワークにて提供されている．ジョブスケジューラの要求時間到達通知機能とは，利用者が指定した実行時間見積に達し SIGKILL シグナルがジョブに送られる前に，SIGTERM などのシグナルをスケジューラがジョブに送る機能である．ジョブではそのシグナルをフックすることで，適切な終了処理を行うことができる．この 2 つの機能を組み合わせることで，ジョブの最大実行時間を短く設定し，最大実行時間を超過したジョブは状態保存・再開機能を用いて，複数回ジョブ実行することで計算完了する．資源利用率の向上は見込めるが，利用者の対応が必要となることが本手法採用にあたっての課題である．

5. 関連研究

並列システムにおけるワークロード解析は数多くされている．Medernach は，システム利用率，および，利用者間の公平性の把握のために，グリッド環境でのワークロード解析を行ない，ジョブ到着のモデル構築を行なっている [9]．Ren らは 2000 ノードから構成される商用 Hadoop クラスタ上でのワークロード解析を行い，その結果を用いてワークロードジェネレータ，および実行頻度の高い小さいジョブ（Map/Reduce タスク数の少ないジョブ）を効率よく実行するスケジューリングアルゴリズムを提案している [10]．You らは 1.17 ペタフロップスの性能を持つスーパーコンピュータ Kraken における 1 年間のワークロード解析を行っている [5]．Kraken は学術利用専用に構築されたスーパーコンピュータとのことで，そのワークロードを解析することにより，大規模科学技術計算ワークロードの

特徴を把握することができる．これら論文では主に以下の点での解析を行っている．

ワークロード全体の特徴　キュー毎のジョブ数，利用者毎のジョブ数，ジョブのキャンセル率・成功率，実行時間見積の正しさ，など

時間的な特徴　月ごとの資源利用率，一定期間内のジョブ数の時系列遷移，ジョブ到着率，など

ジョブの特徴　待ち時間，実実行時間，並列数，メモリ使用量，など

これら既存研究と本研究の違いは，対象とするワークロードの種別にある．我々の知る限り，大規模共有計算機システム上での深層学習ワークロードを解析した研究はない．一方，本研究では上記で挙げたワークロード解析項目のうち，ワークロード全体の特徴とジョブの特徴についての一部しか示していない．今後他の項目についても情報取得し，公開することを考えている．

6. まとめと今後の展望

共有 GPU クラスタ上で，利用者がどのように深層学習プログラムを実行しているか把握することを目的に，産総研が有する AAIC の 5 ヶ月間のジョブ実行ログを対象とした解析を行なった．その結果，次の 2 点がわかった．Single GPU ジョブが全ジョブの 95%をしめること．ジョブ投入時の実行時間見積と実際の実行時間に大きな乖離があること．現状では Multi GPU / Multi Node 分散深層学習の割合が少ないため問題は顕在化していないが，今後分散深層学習の利用が増えるにつれ，バックフィルが満足に機能せず，資源利用率の低下が起きる可能性が高いこと．これらを改善すべく，システム運用側が取れるアプローチについて考察を行った．

今後の課題としては，現状の AAIC が抱える上記課題を解決すべく，開発や運用の制度設計を行っていく．また，今回は各計算ノードにおける資源（CPU, GPU, メモリなど）利用状況を取得できておらず，分散深層学習利用状況統計は一部正確さに欠ける結果となった．今後は各計算ノードに collectd などの資源監視を仕込み，詳細に資源利用状況を監視することで正確な情報を把握するとともに，これら情報を活用した，省電力スケジューリングなどの運用改善を行っていく．以上を通じて得られた知見を 2018 年度上旬に運用開始を予定している ABCI にフィードバックする．

参考文献

[1] Feitelson, D. G.: Workload Modeling for Performance Evaluation, *Performance Evaluation of Complex Systems: Techniques and Tools*, Vol. 2459, pp. 114–141 (2002).

[2] 小川宏高，松岡　聡，佐藤　仁，高野了成，滝澤真一朗，谷村勇輔，三浦信一，関口智嗣：AI 橋渡しクラウド— AI Bridging Cloud Infrastructure (ABCI) —の構想，第 160 回ハイパフォーマンスコンピューティング研究会 (2017).

[3] 産業技術総合研究所：AIST Artificial Intelligence Cloud (AAIC), `http://www.airc.aist.go.jp/info_details/computer-resources.html#section1`.

[4] 佐藤　仁，小川宏高：AI クラウドでの Linux コンテナ利用に向けた性能評価，第 162 回ハイパフォーマンスコンピューティング研究会 (2017).

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

[6] Takuya Akiba, Shuji Suzuki and Keisuke Fukuda: Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes, *CoRR*, (online), available from ⟨http://arxiv.org/abs/1711.04325⟩ (2017).

[7] 西川武志：FOCUS スーパーコンピュータシステムにおける並列課金インセンティブの効果 III，第 161 回ハイパフォーマンスコンピューティング研究会 (2017).

[8] 野村哲弘，佐々木淳，三浦信一，遠藤敏夫，松岡聡：TSUBAME2 におけるスケジュール効率化への取り組みとユーザ動向の見える化，第 150 回ハイパフォーマンスコンピューティング研究会 (2015).

[9] Medernach, E.: Workload Analysis of a Cluster in a Grid Environment, *Job Scheduling Strategies for Parallel Processing* (2005).

[10] Zujie Ren, Jian Wan, Weisong Shi, Xianghua Xu and Min Zhou: 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).

ⓒ 2018 Information Processing Society of Japan