

Kubernetes における期限制約を考慮した GPU アプリケーションの透過的移行制御機構

2531119 濱野 晃暉 成見研究室

1 はじめに

深層学習や科学技術計算に用いられる GPU ワークロードは、長時間にわたって実行される場合が多く、実行途中で強制終了すると保存されていない計算進捗が失われる。一方、複数人で共用される計算基盤では、保守や予約利用、高優先度ジョブの実行などに伴って実行中のジョブを所定の時刻までに停止し、GPU を解放しなければならない場合がある。既存のバッチスケジューラでは、中断の際にアプリケーションへ通知しジョブを終了・再開する機能が提供されている [1]。しかし、実行途中の状態を保存して再開するためには、アプリケーション側で計算進捗を保存する処理を実装する必要がある。このようなアプリケーション固有の機能実装は、利用者への負担となる。

近年、コンテナ技術が一般的になってきたことに加え、CRIU による CPU プロセス状態の保存と、GPU ドライバが提供する Checkpoint 機能を組み合わせることで、アプリケーションを改変せずに GPU を利用する計算ジョブの実行状態を保存・復元する透過的 Checkpoint/Restore が利用可能になりつつある。しかし、Checkpoint の開始から GPU が実際に再割当て可能になるまでの所要時間は、GPU メモリ使用量をはじめとして様々な要因によって変動する。

本研究では、GPU ワークロードを Pod として管理する Kubernetes 基盤を対象とし、透過的 Checkpoint/Restore に必要な時間を考慮して退避開始時刻を決定する制御機構を提案する。

2 背景

2.1 Container・Kubernetes

Container(コンテナ)は、アプリケーションと依存ライブラリを一つの実行単位として隔離して動かす仕組みである。Kubernetes はオープンソースのコンテナオーケストレーションプラットフォームであり、複数ノードから構成されるクラスターで1つ以上のコンテナを束ねた Pod を最小単位として管理し、実行先の決定や障害時の再作成を制御する。Kubernetes は Web サービス基盤や AI・機械学習基盤で広く利用されている。CNCF が 2025 年に行なった調査では、コンテナ利用者の 82% が Kubernetes を本番利用し、生成 AI モデルを運用する組織の 66% が推論処理を Kubernetes で管理している [2]。また、理化学研究所が開発

する富岳 NEXT においても、従来のジョブスケジューラと Kubernetes を併用して AI 処理を管理する方針が示されている [3]。

2.2 Checkpoint/Restore

Checkpoint/Restore(C/R) は、実行中プロセスの状態を保存し、後に同じ状態から実行を再構成する技術である。GPU アプリケーションでは、CPU 側のプロセス状態に加え、GPU メモリ、CUDA コンテキスト、ストリームなど、GPU ドライバが管理する状態も保存する必要がある。GPU アプリケーションの C/R は、表 1 に示すように、アプリケーション協調方式と、システムレベル方式に大別でき、システムレベル方式は、デバイスプロキシ方式と、ドライバ支援方式に分類できる。

従来の GPU アプリケーションの変更を必要としない透過的な GPU アプリケーションの C/R では、GPU API を仲介して状態を記録・再構成する方式が用いられてきた。一方、近年では GPU ドライバに C/R 機能が導入され、ドライバが管理する GPU 状態を直接保存する方式が利用可能になりつつあるが、GPU 対応 C/R の Kubernetes 統合に関する研究報告はまだ限られている。本研究ではドライバ支援方式を採用する。

3 目的

本研究の目的は、Kubernetes クラスター上で単一 GPU Pod を対象に、アプリケーションへ専用の C/R 処理を追加せず、指定された期限と予測所要時間に応じて Checkpoint 作成を開始し、ノードを跨いだ GPU アプリケーションの移動を制御する機構を設計・実装・評価することである。図 1 に本研究で実装する期限制約を考慮した移行の概要を示す。

4 関連研究

Stoyanov ら [4] は、CRIU と GPU ドライバの機能を連携させ、アプリケーションを変更せずに通常実行中の追加オー

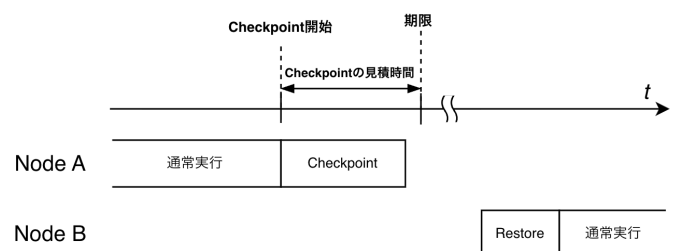


図1 本研究で実装する GPU アプリケーション移行制御の概要

表 1 Checkpoint/Restore の分類

	アプリケーション協調方式	システムレベル方式	
		デバイスプロキシ方式	ドライバ支援方式
概要 長所 短所	アプリケーションが状態を外部に保存 移植性が高い、保存対象を選択できる アプリケーションごとに実装が必要	GPU API を仲介し呼出しを記録、復元時に再実行 アプリケーションの改変不要、ベンダー非依存 性能オーバーヘッドが発生	GPU ドライバで GPU 状態を保存、復元 アプリケーションの改変不要、性能オーバーヘッド少 ベンダー依存、対応機能の制約

バーヘッドを避けながら、深層学習や HPC ワークロードの復元を実現する CRIUgpu を提案している。一方、その中心は C/R 機構であり、本研究で扱う Kubernetes による制御機構は対象としていない。

Jia ら [5] は、研究室などが所有する余剰 GPU を大学内で共有し、所有者が GPU を突然回収する環境を対象としたネットワーク・移行制御のシステムである ReclaimNet を提案している。GPU の返却期限を考慮して Checkpoint 作成の時期、移行先、通信帯域を制御している。アプリケーションが保存する学習状態を利用する方式である点や、Kubernetes の Pod 再作成との連携は中心ではない点で本研究と異なる。

5 進捗

5.1 実装

Kubernetes 開発向けライブラリ controller-runtime を用い、Go 言語で GPU アプリケーションの移行を制御するカスタムオペレーターを実装した。利用者が対象 Pod と GPU 解放期限を記述した移行要求を、Custom Resource (独自に定義した管理対象) として作成すると、移行可能性の確認、CPU・GPU 状態の保存、移行元 GPU の解放、互換ノードでの状態復元を順に実行する。状態保存には CRIU と NVIDIA ドライバの GPU C/R 機能を用いるため、アプリケーションコードやコンテナイメージに専用の保存・再開処理を追加する必要がない。

現時点では、Checkpoint にかかる時間は予測せず、事前に設定した時間に基づいて Checkpoint を開始するようになっている。今後は、GPU・CPU メモリ使用量や保存先性能などのメトリクスを取得して GPU 解放までの所要時間を推定し、実行状態に応じて Checkpoint 開始時刻を調整する機構を実装する予定である。

5.2 実 GPU 環境における基本動作の検証

表 2 に示す環境において、上記の実装の基本動作を検証した。CUDA を用いた分子シミュレーション、PyTorch を用いた学習プログラムおよび推論プログラムの 3 種を、同型 GPU を持つノード間で移行した。いずれも別ノード上で処理を再開し、計算状態と進捗が引き継がれることを確認した。表 3 に、ワークロード別の Checkpoint と Restore の所

表 2 動作検証環境

項目	内容
クラスタ構成	Control Plane 1 台, Worker 2 台
GPU ノード	各 Worker に NVIDIA RTX A4000 (16 GB)
OS・カーネル	Ubuntu 24.04.4, Linux 6.8.0 (amd64)
Kubernetes	Kubernetes 1.36.2
コンテナランタイム	containerd 2.2.1
C/R 機構	CRIU 4.2, NVIDIA GPU ドライバ 595.71.05

表 3 Checkpoint/Restore の平均所要時間と復元用イメージサイズ

ワークロード	Checkpoint[s]	Restore[s]	イメージ [MiB]
分子シミュレーション	2.002	1.215	198.4
PyTorch 学習	8.806	3.792	1016.2
PyTorch 推論	5.813	2.645	665.3

要時間、および復元用データとして共有ストレージに保存したイメージのサイズを示す。表中の値は、各ワークロードについて 3 回実施した試行の平均値である。

6 今後の予定

今後は、GPU・CPU メモリ使用量、保存先性能、ワークロード構成を変化させ、C/R の成否と GPU 解放所要時間に影響する要因を調査する、それに基づいて Checkpoint 開始時刻を決定するメカニズムを設計し、現在実装しているカスタムコントローラーへ組み込む。

評価では、C/R による進捗保持と期限内の GPU 解放を中心に、提案方式の有効性をどのような指標と実験条件で捉えるかを検討する。また、異なるワークロードや移行途中の障害などを考慮し、提案方式の適用範囲と課題について評価を行う。

参考文献

[1] SCHEDMD LLC, “Slurm Workload Manager - Job Launch Design Guide,” https://slurm.schedmd.com/job_launch.html, (アクセス日: 2026-07-28).

[2] Cloud Native Computing Foundation, “The CNCF Annual Cloud Native Survey,” 2026.

[3] 理化学研究所, 『次世代フラッグシップシステム「富岳 NEXT」基本設計技術報告書』, Version 1.1, 2026.

[4] R. Stoyanov et al., “CRIUgpu: Transparent Checkpointing of GPU-Accelerated Workloads,” arXiv:2502.16631, 2025.

[5] W. Jia et al., “ReclaimNet: Reclaim-Aware Network Protocols,” arXiv:2605.28872, 2026.