様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　６ 年　 ５ 月 １５ 日現在

機関番号：１２１０２
研究種目：若手研究
研究期間：2020～2023
課題番号：２０Ｋ１９８０７

研究課題名（和文）ＧＰＵアプリケーションに対するシステムレベルのチェックポイント技術の確立

研究課題名（英文）System-level Checkpoint Technology for GPU Applications

研究代表者
額田　彰（Akira, Nukada）

筑波大学・計算科学研究センター・教授

研究者番号：４０５４５６８８

交付決定額（研究期間全体）：（直接経費）　　3,300,000円

研究成果の概要（和文）：計算実行中の障害や実行時間制限のある共用システムで長時間要する計算を行うときには実行状態を保存するチェックポイント技術が有効である。ユーザのプログラムを修正する必要がないシステムレベルチェックポイントではＣＰＵのメモリのイメージを保存する仕組みでありＧＰＵアプリケーションに対応していない。ＧＰＵのランタイムライブラリを置き換えることによってＧＰＵ側の状態をＣＰＵメモリに保存し、また再開時にはそのデータからＧＰＵ側の状態を再構築する技術を確立した。再構築するためにはＧＰＵにアクセスするプロセスを分離することが必須で、ライブラリ関数の呼び出しをプロセス間でリレーする方式を採用した。

研究成果の学術的意義や社会的意義
現在ＧＰＵコンピューティングで主流となっているＮＶＩＤＩＡ社のＧＰＵに関して、ＣＵＤＡで開発されたアプリケーションの実行バイナリを改変することなくチェックポイントに対応するソフトウェアの確立に成功した。プロセスを分離する方式を採用していることによってライブラリ関数の呼び出しやＣＰＵとＧＰＵ間のデータ転送にオーバヘッドが生じる。主要な関数や実アプリケーションについてオーバヘッドの評価を行ったところ、ＧＰＵが十分に活用されているような状況ではオーバヘッドが非常に小さいことを確認している。

研究成果の概要（英文）：Checkpoint, which saves and restores the state of running processes, is a key technology for system failure during job executions or for executing long-running applications on systems with execution time limits. System-level checkpoint saves whole system memory image of CPU, and is not compatible with GPU applications. We introduce another GPU runtime library to replace user applications' calls to save GPU state in order to restore the GPU state on restarting. To restore the GPU state, we need to employ another dedicated process which accesses GPU, and all GPU library calls are relayed to the server process.

研究分野：GPUコンピューティング

キーワード：ＧＰＵコンピューティング　チェックポイント　ＣＵＤＡ

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

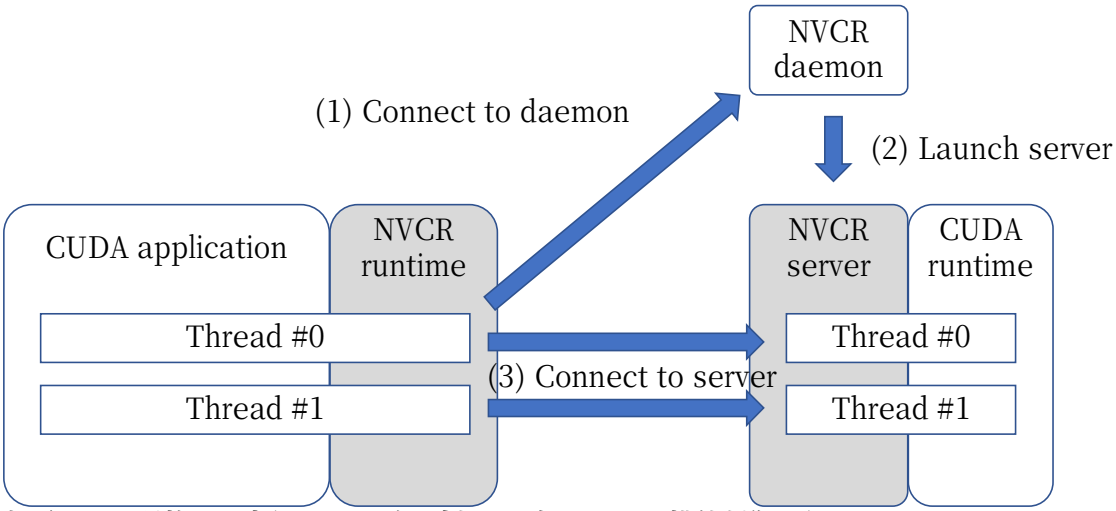
実行中のプロセスの状態を保存するチェックポイント技術は元々耐故障の目的で開発された。クラスタ計算機を用いて大規模並列計算を行うような場合に計算中に故障が発生する確率は使用するノード数倍になるため無視できないレベルとなる。実行途中で何れかの計算機に障害が発生してプロセスが強制終了された場合、計算を再び最初から実行しなければならず障害発生率によっては正常に実行が完了するまで非常に長い時間を要することになる。これに対して一定時間毎にプロセスの状態をファイルに保存することができれば障害発生時に最新の保存ファイルから実行を再開することで障害発生率が高い場合にも計算を進めることが可能となる。
チェックポイントには大きく二種類あり、アプリケーション自体にチェックポイント機能を追加実装する方法の場合、プログラムの構成によっては大きな負担となる。一方でシステムレベルチェックポイントの場合はアプリケーションの実行バイナリを改変することなくそのまま利用できるためユーザにとっての利便性は高い。このシステムレベルチェックポイントはプロセスに属する CPU メモリのイメージを全て保存する方式になっている。このため CPU メモリ上にデータを持たない GPU 側の状態を保存することができないため、GPU アプリケーションに対応できていない。

２．研究の目的

本研究の目的は GPU アプリケーションのチェックポイント、再開を可能とする技術を確立することである。過去に幾つかの成功例があったが、GPU 環境の更新、特にランタイムライブラリや API 関数の変更によって現在では機能しなくなっている。詳細なランタイムライブラリの内部実装に依存しないような、安定して長期維持可能なソフトウェア NVCR(version 2)を実現する。

３．研究の方法

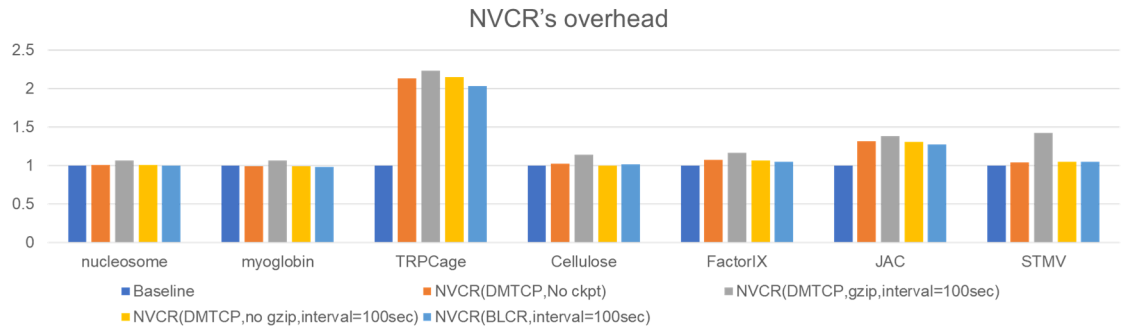
GPU アプリケーションとして特に現在主流となっている NVIDIA 社製ＧＰＵを利用した CUDA プログラムを対象とする。BLCR や DMTCP といったよく利用されているシステムレベルチェックポイントソフトウェアを GPU アプリケーションに対応できるように拡張する。GPU の状態を保存することができないため、チェックポイントを保存する時の前処理で GPU 側のデータを全て CPU 側のメモリにバックアップしてファイルイメージの一部として保存する。ファイルから実行を再開する際に GPU の状態を再構築することになり、これに必要な情報を全ての CUDA のランタイム API 関数呼び出しをモニタリングしながら回収する方式を採用する。
CUDA アプリケーションのチェックポイントにおいて一番難しい点は、ファイルからの実行再開時に GPU の状態を復元できるかどうかである。NVCR では GPU のリソース(メモリやイベント、ストリームなどの制御用データ)の確保や解放を行う CUDA API 関数の呼び出しを全てログに保存し、それを元通りの順番で再実行する（リプレイ）ことによって GPU の状態を再構築する方式を採用する。特に GPU のメモリについては同じアドレスで再確保されなければ正常に実行を再開することができない。一方でリソースに変化を与えない GPU カーネルの実行などについてはログを取ることはなく再実行することもない。
現状の CUDA のランタイムの問題として、ランタイムライブラリの初期化をもう一度実行させることができないことがある。最初に CUDA API 関数を呼んだ時点で GPU を含めて各種初期化作業が実行されるのであるが、チェックポイントイメージにはこの初期化済み状態が保存され、再開時に GPU の初期化が行われない。これを解決する唯一の手段はランタイムライブラリを使用するプロセス(NVCR server)を別途分離して、使い捨てにすることである。イメージファイルから実行を再開する時には新たなプロセスを起動することによって初期化を再度実行することができる。この分離方式にはメリットがある。一定間隔でチェックポイントを保存するような場合には保存した後に計算を続行するが、NVCR server 側の GPU リソースはそのまま使うことができる。分離せずに同じプロセスであった場合には GPU リソースを全て解放する必要があった。一方でデメリットもある。プロセスが分離されているため API 関数の呼び出しをプロセス間で転送する必要があり、また CPU と GPU 間のデータ転送も遅くなる。
NVCR のソフトウェア構成を下の図に示す。環境変数の設定によりユーザアプリケーションが CUDA のランタイムライブラリの代わりに NVCR のランタイムライブラリを読み込む。計算ノードにはあらかじめ NVCR デーモンプロセスを起動しておき、アプリケーションが最初に CUDA API 関数を呼んだ時点でデーモンに接続し、サーバプロセスを起動させる。そしてサーバプロセスへの接続を確立させ、サーバプロセスの CUDA ランタイムライブラリに API 関数を転送する。互換性や並列性のため、アプリケーションプロセス内と同様にスレッドをサーバに生成し、１：１の関係を維持する。これは互換性のためだけでなく、各スレッドが異なる GPU を使用する場合にも並列に動作するために必要な構成である。再開用に必要なログは NVCR ランライムライブラリによってアプリケーションプロセス内のメモリに保持される。GPU 関連のメモリ空間は共通化されているため、ログはプロセスに対して１つに集約する。このためログの保存が



必要な API 関数では実行からログに追加する処理までに排他制御を行っている。
チェックポイントを保存する際には GPU のメモリ上にあるデータをサーバプロセス経由でアプリケーションのメモリ内にバックアップし、サーバと切断したのちにアプリケーションプロセスのみをファイルに保存する。保存が終了した後はサーバに再接続し、実行を継続する。この場合は GPU のリソースをそのまま再利用するため高速に処理できる。ファイルから実行を再開する場合には全てのデータがメモリ上に読み込まれた後、サーバを新たに起動し、保存されたログに従って API 関数を再実行し、GPU のメモリデータをアプリケーション内のメモリから復元する。

４．研究成果

実アプリケーションでの適用例として分子動力学シミュレーションソフトウェアの AMBER18 で評価を行った。チェックポイント機能なしで通常環境で実行した場合の実行時間(Baseline)を 1.0 として、各条件での相対実行時間を示している。



入力データとして Amber Benchmark Suite に含まれるデータを使用しているが、この中では TRPCage で大きなオーバヘッドが見られる。TRPCage はこの中では一番小さい分子であり GPU カーネルの計算時間が短いため API 呼び出しをリレーしている分のオーバヘッドが相対的に大きくなっている。No ckpt はチェックポイントの保存を行わず、API 呼び出しのオーバヘッドのみの影響を受けている。次に 100 秒間隔という非常に厳しい頻度でチェックポイントの保存を行う場合であるが、この追加のオーバヘッドのほとんどはファイルの書き込み時間である。ノード内のローカル SSD にファイルを保存しているため gzip で圧縮してファイルの書き込みサイズを小さくするよりもそのまま書き込む方が早いという結果になっているが、リモートのファイルシステムに多数のプロセスが書き込みを行うような場合には圧縮する方がよい可能性が高い。ユーザレベルで動作する DMTCP に対してカーネルレベルで動作する BLCR の方がチェックポイントのイメージファイルのサイズも小さくなり、また API 関数の呼び出し時の排他制御の負荷も小さい。他の入力データに対しては十分分子が大きい場合にはオーバヘッドが非常に小さく、現実的に用いられるチェックポイント間隔であれば問題なく実用的なオーバヘッドの低さであると言える。

５．主な発表論文等

〔雑誌論文〕　計4件（うち査読付論文　4件／うち国際共著　1件／うちオープンアクセス　0件）

１．著者名 Nukada Akira、Suzuki Taichiro、Matsuoka Satoshi	４．巻 116
２．論文標題 Efficient checkpoint/Restart of CUDA applications	５．発行年 2023年
３．雑誌名 Parallel Computing	６．最初と最後の頁 103018～103018
掲載論文のDOI（デジタルオブジェクト識別子） 10.1016/j.parco.2023.103018	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Tatsugi Yuya、Nukada Akira	４．巻 -
２．論文標題 Accelerating data transfer between host and device using idle GPU	５．発行年 2022年
３．雑誌名 GPGPU '22: Proceedings of the 14th Workshop on General Purpose Processing Using GPU	６．最初と最後の頁 1-6
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3530390.3532732	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Dieguez Adrian P、Amor Margarita、Doallo Ramon、Nukada Akira、Matsuoka Satoshi	４．巻 36
２．論文標題 Efficient high-precision integer multiplication on the GPU	５．発行年 2022年
３．雑誌名 The International Journal of High Performance Computing Applications	６．最初と最後の頁 356-369
掲載論文のDOI（デジタルオブジェクト識別子） 10.1177/10943420221077964	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 該当する

１．著者名 Nukada Akira	４．巻 -
２．論文標題 Performance Optimization of Allreduce Operation for Multi-GPU Systems	５．発行年 2021年
３．雑誌名 2021 IEEE International Conference on Big Data (Big Data)	６．最初と最後の頁 3107-3112
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/BigData52589.2021.9672073	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計5件（うち招待講演　0件／うち国際学会　3件）

1．発表者名
Tatsumasa Seimi, Akira Nukada

2．発表標題
Performance Evaluation of OpenSWPC using Various GPU Programming Methods

3．学会等名
International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2024)（国際学会）

4．発表年
2024年

1．発表者名
Tatsumasa Seimi, Akira Nukada

2．発表標題
GPU Acceleration of OpenSWPC using DO CONCURRENT

3．学会等名
GPU Technology Conference 2023 Spring（国際学会）

4．発表年
2023年

1．発表者名
勢見 達将, 額田 彰

2．発表標題
DO CONCURRENT構文によるOpenSWPCのGPU化

3．学会等名
情報処理学会ハイパフォーマンスコンピューティング研究会

4．発表年
2023年

1．発表者名
Akira Nukada

2．発表標題
Hybrid Allreduce Algorithm for On-node Multi-GPU Systems using both NV-Link and PCI-Express Networks

3．学会等名
NVIDIA GTC 2022（国際学会）

4．発表年
2022年

１．発表者名 立木佑弥，額田彰
２．発表標題 遊休GPUを利用したホスト・デバイス間通信の高速化
３．学会等名 情報処理学会ハイパフォーマンスコンピューティング研究会
４．発表年 2022年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関