jh250046

CPU と GPU を同時利用する自己学習モンテカルロ法の開発

永井佑紀（東京大学情報基盤センター学際情報科学研究部門）

本研究では、CPU と GPU を同時利用する自己学習モンテカルロ法の実現に向けて、分子動力学法パッケージ PIMD の拡張を行った。独自の機械学習ポテンシャルソフトウェア AccelNet の開発を進めるとともに、汎用機械学習ポテンシャル MACE を PIMD から利用可能にするため、Fortran、C++、Python を連携させるインターフェースを開発した。特に、推論部分は C++経由で高速に実行し、学習部分は Python デーモンを用いて GPU 上で柔軟に実行する構成を採用した。さらに、ASE を経由することで多様な機械学習ポテンシャルを PIMD に導入可能とする基盤を構築した。本年度は大規模計算の本格実行には至らなかったものの、CPU-GPU 連携型自己学習シミュレーションに向けた重要なソフトウェア基盤を整備した。

1. 共同研究に関する情報

⑴ 共同利用・共同研究を実施している拠点名

東京大学 情報基盤センター

(2) 課題分野

大規模計算科学課題分野

(3) 参加研究者一覧と役割分担

永井佑紀：研究計画の立案および遂行
奥村雅彦：SLHMC の物理学的観点からの議論
住元真司、中島研吾、荒川隆：Wisteria の CPU+GPU 連携パートの技術的観点からの情報提供および計算効率化に関する議論

2. 研究の目的と意義

自己学習モンテカルロ法(Self-learning Monte Carlo, SLMC)は、マルコフ連鎖モンテカルロ法の一種であり、マルコフ連鎖において次の配位の候補を生成する際、有効模型によるマルコフ連鎖（自己学習モンテカルロ法）や分子動力学法（自己学習ハイブリッドモンテカルロ法）を用いて、候補を生成する手法である。この方法は、「機械学習を使っているにも関わらずシミュレーションの精度が元のシミュレーションによるものと等しい」という著しい特徴を持っている。この特徴のおかげで、構成する有効模型の精度に左右されずにシミュレーションを安心して遂行でき、物理量の精度を保証しつつ劇的な高速化に成功している。

この手法自体は MCMC を使う様々なシミュレーションに適用することができ、これまで、原子・分子系における機械学習分子動力学法、高エネルギー物理学分野における格子量子色力学、強相関電子系における量子モンテカルロ法など、様々な分野に適用してきた。また、機械学習分子動力学分野においては、オープンソースの分子動力学法パッケージ PIMD に SLMC 機能を追加し、多くの人が SLMC を用いたシミュレーションが可能になるように研究開発を行ってきた。そして、PIMD は、現在、大学・企業問わず様々なグループが研究開発に実際に利用している状況である。一方、SLMC は、シミュレーションの最中に有効模型を徐々に改良していく手法であるため、「オリジナルの重たいコードを実行するシミ

ュレーションパート」と「有効模型を機械学習するパート」の二つの異なる種類の計算パートが存在している。また、「有効模型を機械学習するパート」に関しては、世界中の様々な研究グループが研究開発を行っており、その際に GPU を利用することで高速に学習が実現できることが知られている。そして、第一原理分子動力学を始めとするシミュレーションパートは、マルチ CPU コアによる並列計算が効力を発揮することが知られている。しかし、二つの異なる計算パートを組み合わせて使用するコードを開発することは、計算物理学者だけでは難しく、計算科学者との協力が不可欠である。そこで、本研究では、昨年度に引き続き、CPU メイン計算機(Wisteria/BDEC-01 Odyssey 及び Miyabi-C) と、GPU メイン計算機(Wisteria/BDEC-01 及び Miyabi-G) の二つの同時使用も可能とする SLMC コードを開発することも目的とする。その際、CPU メイン機と GPU メイン機間の通信を利用して、マルチコア CPU シミュレーションと GPU による機械学習を両立させ、一つの SLMC シミュレーションとして統合されたコードを開発する。特に、最も利用者が多いと見込まれる機械学習分子動力学シミュレーションに着目し、既存の PIMD パッケージに CPU-GPU 通信を実装しつつ、GPU に対応した様々なオープンソース機械学習有効模型作成コードを連携させる。このコードの開発が実現できれば、SLMC による効率的な機械学習シミュレーションを高速に実施できるようになり、より多種多様な物質群（電池材料、熱電材料、その他高機能物性材料）の理論設計とスクリーニングが可能となり、研究開発が加速されることになる。さらに、今年度は、更新がほぼ停止している既存のソフトウェア(aenet,n2p2 等)だけではなく、最新の機械学習分野の知見を取り入れられるように、自作の GPU 対応機械学習ポテンシャルソフトウェアのオープンソースパッケージを開発し、PIMD に組み込むことを目的とする。

3. 当拠点の公募型共同研究として実施した意義

本研究は、東京大学情報基盤センターが提供する CPU 主体および GPU 主体のスーパーコンピュータ資源を同時に活用することを前提としており、単一環境では実現が困難な計算手法の開発を可能にした点に大きな意義がある。特に、Wisteria/BDEC-01 や Miyabi といった異なるアーキテクチャを持つ計算機群を連携させることで、CPU による大規模並列シミュレーションと GPU による機械学習を同時に実行し、それらを統合する新しい計算フレームワークの検討が可能となった。これは、SLMC のような異種計算の統合を必要とする手法において、本拠点のような先進的な計算環境が不可欠であることを示している。

4. 前年度までに得られた研究成果の概要

前年度においては、自己学習モンテカルロ法（SLMC）における CPU と GPU の協調利用を実現するための基盤技術の開発を行った。具体的には、分子動力学法パッケージ PIMD に対してコード解析および構造改修を実施し、従来は CPU 上で逐次的に実行されていた「シミュレーションパート」と「機械学習パート」を分離することで、異種計算資源の並列利用を可能とする設計へと拡張した。また、GPU による高速学習を実現するため、PyTorch ベースの機械学習ポテンシャル実装である aenet-PyTorch を PIMD に統合し、CPU 上のシミュレーションと GPU 上の機械学習を協調的に動作させる仕組みを構築した。これにより、従来は機械学習処理の間シミュレーションが停止していた実行形態を改良し、両者が非同期に動作する基本的な枠組みを実現した。さらに、Odyssey（CPU）と Aquarius（GPU）間での通信を利用した連携実行の検証を行

い、ファイルベースの簡易な同期機構を用いることで、異なる計算ノード間での協調的な SLMC シミュレーションが可能であることを示した。この過程において、CPU と GPU の負荷バランスが性能に大きく影響することなど、実運用上重要な知見も得られた。加えて、SLMC の有効模型の高度化として、対称性を保持した同変トランスフォーマー型モデルを開発し、少ないパラメータ数で高精度なモデル化が可能であることを示すとともに、モンテカルロ法におけるアクセプト率の向上を確認した。また、第一原理計算と機械学習ポテンシャルを組み合わせた自己学習ハイブリッドモンテカルロ法（SLHMC-MIX）を開発し、従来の第一原理 PIMD と比較して大幅な計算効率向上と高精度な構造再現を両立することに成功した。以上により、SLMC における CPU-GPU 連携の基本的枠組みの確立と、有効模型の高精度化・高速化の双方において重要な進展が得られた。

5. 今年度の研究成果の詳細

1. 独自ソフトウェア AccelNet の開発：オブジェクト指向 Fortran2008 と Julia 言語で書かれたオープンソースソフトウェア AccelNet（研究計画での仮称は BPnet）を開発し、デバッグとベンチマークを行った。既存の機械学習ポテンシャルソフトウェアである aenet では記述子と呼ばれるニューラルネットワークに入力するインプットを作成する機能がある。この記述子をどのように計算するかで精度と速度が決まる。AccelNet では同じアルゴリズムを使った場合には完全に同一（丸め誤差除く）の記述子が得られる。また、プログラムコードの効率化によって、同じネットワークを使った場合には記述子作成速度、推論速度ともに数倍以上高速となった。近年、機械学習ポテンシャルコードは大規模化しており、精度は高くなっているもの推論コストも高くなっており、MD として実行する上では低速化する方向にある。本ソフトウェア AccelNet は、シンプルなアルゴリズムと小さなモデルによって、世界最高水準の推論速度を目指している。

学習部分に関しては、Julia 言語による GPU 学習コードはすでに開発できており、動作検証を行なっている。また、同一シミュレーション上では学習と推論を同時に行う必要もあるため、Fortran2008 バージョンも作成中であるが、こちらは GPU 化が遅れている。また、別のオープンソースパッケージ n2p2 との速度比較のベンチマークを行うため Behler-Parrinello 型記述子を実装中であったが、中間評価時点で述べたように、丸め誤差を除いた完全に同一な出力が得られておらず、原因の究明に多大な時間を要してしまった。そのため、ソフトウェア全体の開発を年度内に完了することができなかった。しかしながら、高速で高精度なポテンシャルとするための条件出しを行なったことで、今後世界最高水準の推論速度のソフトウェアを作成するためにはどのようなアルゴリズムであるべきかを整理することができた。今後は、プログラムコードの効率化とともに、今後主流となる GPU 搭載計算機上での効率的な実行を可能とするアルゴリズムを開発することで、世界最高水準の推論速度を目指す。

2. PIMD+MACE 連携：元々オープンソースソフトウェア NequIP を PIMD に実装する予定であったが、さまざまな技術的な理由により別のオープンソースソフトウェアである MACE を採用することになった。MACE では、事前に訓練されたポテンシャル（汎用ポテンシャル）が用意されているため、これを初期ポテンシャルとして系に最適なオーダーメイドポテンシャルを作成できると期待できる。

異種言語間インターフェースの実装: 本研究では、Fortran（PIMD 本体）、C++（MACE）、Python（学習）という複数言語を統合する必要があった。このため、以下の 3 層構造を採

用した：

1. Fortran-C++：ISO_C_BINDING を用いた直接インターフェース

2. C++-Python：既存の Python API を利用

3. Fortran-Python：デーモン通信による間接連携

特に Fortran から直接 Python を呼び出す方式ではオーバーヘッドが大きくなるため、本研究では常駐型の Python デーモンを導入し、通信ベースで処理を委譲する方式を採用した。

1.Fortran-C++ 直接インターフェース：MACE は C++コードである LAMMPS から呼び出し可能であるが、この C++コードを参考に PIMD 用の Fortran-C++インターフェースコードを開発した。これにより、LAMMPS と同程度の速度で MACE ポテンシャルを PIMD 上で実行できるようになった。

2.,3. C++-Python および Fortran-Python: 学習に関しては、Fortran で記述された PIMD と Python ベースの機械学習コードを接続するために、デーモンプロセスを介した通信機構を構築した。この方式により、

1. Python 側で GPU を用いた学習を継続的に実行可能
2. Fortran 側からの呼び出しオーバーヘッドを削減
3. 非同期処理の実現

を達成した。Python で作成された MACE のポテンシャルは C++コード用へ変換され、そのポテンシャルファイルは C++-Fortran インターフェースを介して推論に用いることができる。

また、この実装では、推論時と学習時で異なる実行経路を採用している点も重要である。推論については、分子動力学計算の各ステップで頻繁に呼び出されるため、可能な限りオーバーヘッドを小さくする必要がある。そのため、MACE ポテンシャルを C++側で読み込み、Fortran から直接呼び出す構造とした。一方、学習については、推論と比較して呼び出し頻度は低いものの、GPU を用いた大規模な行列演算や自動微分を必要とするため、Python 側の機械学習フレームワークをそのまま利用する設計とした。

このように、推論部分は C++を経由した高速な実行を行い、学習部分は Python デーモンを経由して柔軟性を確保することで、速度と拡張性の両立を図った。機械学習ポテンシャルの分野では新しいモデルが Python パッケージとして公開されることが多いため、Python 側を完全に排除せず、必要な部分だけを高速な実行経路に接続する設計は、今後の拡張性という観点からも有用である。

なお、このデーモンは GPU 学習が可能であるため、CPU+GPU 同時使用のためのツールとして有用である。一方、開発に時間がかかってしまったため、ファイルを介しての自己学習ハイブリッドモンテカルロ法を超えるような連携手法の開発までは至らなかった。

3. PIMD への各種機械学習ポテンシャル導入：python によるデーモン起動と Fortran で書かれた PIMD を連携する方法を確立したため、python で扱える任意の機械学習ポテンシャルを PIMD に実装する方法を確立することができた。その際、python パッケージである ASE(Atomic System Environment)を経由することで、ASE に対応している機械学習ポテンシャルであればどのポテンシャルも利用することができる。この設計により、PIMD 本体を大きく改変することなく、新しい機械学習ポテンシャルを比較的容易に導入できるようになった。従来は、特定のポテンシャルを PIMD に組み込むためには、そのポテンシャルごとに個別のインターフェースを作成する必要があり、コードの保守性や拡張性に課題があった。本研究で導入した ASE 経由の実装では、PIMD 側は原子座標、原子種、セル情報などの標準的

な情報を外部プログラムに渡し、外部プログラム側でエネルギー、力、応力などを計算して返す構造となる。そのため、PIMD 側の変更を最小限に抑えつつ、ASE に対応した多様な機械学習ポテンシャルを利用できる。

この仕組みは、単に MACE を利用可能にするだけではなく、今後登場する新しい機械学習ポテンシャルを迅速に PIMD へ取り込むための共通基盤となる。機械学習分子動力学の分野では、モデルの更新速度が非常に速く、特定の実装に強く依存したコードは短期間で陳腐化する可能性がある。本研究で開発したデーモン方式および ASE 経由の連携機構は、このような分野の変化に対応するための柔軟な設計であり、PIMD を長期的に発展させる上でも重要な成果である。

現在、オープンソースソフトウェア PIMD のバージョンアップに向けて作業中であり、近日中に本プロジェクトで開発した手法が含まれたバージョンが公開予定である。

実装上の課題と得られた知見：本研究を通じて、CPU と GPU を同時利用する自己学習モンテカルロ法を実用的なソフトウェアとして構築するためには、単に GPU 対応の機械学習ポテンシャルを導入するだけでは不十分であり、シミュレーションコード全体の実行構造を見直す必要があることが明らかになった。特に、PIMD のような既存の大規模 Fortran コードでは、長年の開発の中で多数の機能が組み込まれており、外部の機械学習コードと接続する際には、入出力形式、単位系、メモリ配置、並列化方式などを一つずつ確認する必要があった。

また、機械学習ポテンシャルを分子動力学計算に組み込む際には、学習時の性能と推論時の性能を分けて考える必要がある。学習では GPU を用いた大規模な行列演算や自動微分の効率が重要である一方、推論では分子動力学の各ステップで繰り返し呼び出されるため、関数呼び出しやデータ変換のオーバーヘッドが全体の計算時間に大きく影響する。そのため、本研究では、推論については C++を経由した高速な実行経路を用い、学習については Python デーモンを用いた柔軟な実行経路を採用した。このように用途に応じて異なる経路を使い分ける設計は、今後の CPU-GPU 連携型シミュレーションにおいて重要な指針になると考えられる。

さらに、CPU 側のシミュレーションと GPU 側の学習を同時に動作させる場合、両者の計算時間のバランスが重要である。CPU 側が十分な学習データを生成できない場合、GPU 側は待機状態となり、GPU 資源を有効に利用できない。一方、GPU 側の学習が長時間を要する場合には、更新されたポテンシャルをシミュレーションに反映する頻度が下がり、自己学習の効率が低下する。このため、実際の運用では、データ生成頻度、学習頻度、ポテンシャル更新頻度を適切に調整する必要がある。本年度は大規模な本番計算には十分に至らなかったものの、これらの調整が今後の性能向上における主要な課題であることを明確にできた。

加えて、Python デーモン方式を導入したことで、ソフトウェアの保守性と拡張性に関する重要な知見も得られた。近年の機械学習ポテンシャルは Python パッケージとして提供されることが多く、モデルの更新も非常に速い。そのため、各ポテンシャルを Fortran コードに直接組み込む方式では、開発コストが高く、長期的な保守も困難になる。本研究で採用したデーモン方式では、PIMD 本体と機械学習ポテンシャル部分を疎結合に保つことができるため、PIMD 側の安定性を維持しつつ、新しい機械学習モデルを比較的容易に取り込むことができる。この点は、今後 PIMD を機械学習分子動力学の共通基盤として発展させる上で重要である。

以上のように、本年度の研究では、計算資源

を用いた大規模実行そのものよりも、CPU-GPU 連携、異種言語間接続、推論と学習の分離、デーモン方式による拡張性確保といった、今後の大規模自己学習シミュレーションに不可欠な基盤技術の整備が中心となった。これらの成果は、次年度以降に本格的な SLMC および SLHMC シミュレーションを実行するための土台となるものであり、当初計画の発展に向けた重要な段階であったと考えられる。

6. 進捗状況の自己評価と今後の展望

本年度の研究では、当初想定していた大規模な本番計算の実施よりも、基盤ソフトウェアの整備とインターフェース開発に多くの時間を要した。そのため、計算資源の本格的な利用という観点では十分ではなかった部分がある。一方で、MACE のような汎用機械学習ポテンシャルを PIMD から利用可能にし、さらに Python デーモン方式により多様なポテンシャルを導入可能にしたことは、今後の研究展開において重要な成果である。これにより、単一のポテンシャルに依存した実装ではなく、今後登場する新しい機械学習モデルを柔軟に取り込める計算基盤が整備された。

今後、開発したソフトウェア群をオープンソースとして公開することを予定している。特に、さまざまなポテンシャルに対応するために Python によるデーモン方式を開発したことで、より広い範囲で PIMD が使えるようになったのみならず、非常に急速に進展している機械学習 MD 分野における最新の知見を素早く取り入れることが可能になったことは重要である。

※7. 研究業績はウェブ入力です