jh180076-NWH

可視化用粒子データを用いた In–Situ 可視化システムの SIMD 最適化

河村拓馬（日本原子力研究開発機構）

概要
In-Situ 可視化は、スーパーコンピュータ上でシミュレーションと同時に可視化を行うことで結果データの I/O を避け、大規模シミュレーションを確実に可視化できる手法である。しかし、従来の In–Situ 可視化では対話的操作が困難であり、可視化処理性能も十分でない。また多変量可視化への対応が必要とされている。本研究ではサイズの小さな可視化用粒子データを利用して、ファイルベース制御による In-Situ 可視化システム（In–Situ PBVR）を構築し、対話的な In–Situ 可視化を実現した。In–Situ PBVR では可視化用粒子データの生成は低コストのモンテカルロ法で実行され、高いスケーラビリティがある。そして多変量可視化機能として、ユーザ指定の代数式によるボリュームデータ合成、色・不透明度合成が可能である。昨年度は最新のメニーコアアーキテクチャである KNL へ In–Situ PBVR を移植し、Oakforest-PACS を用いて約 10 万コアまでのストロングスケーリングを達成した。しかし代数式を計算するための数式処理が最適化されていなかったため KNL の SIMD 演算を利用できず、十分な性能を引き出すことができなかった。また、可視化用粒子データのファイル出力に伴う遅延が高並列時に顕在化し、スケーラビリティを劣化させていた。加えて、メニーコア向けに最適化された階層型シミュレーションへの適用が未達成であった。本年度は、SIMD 並列化可能な代数式による多変量可視化アルゴリズム、および、ファイル出力を隠蔽するタスク並列機能を開発した。また、構造格子向けに開発した最適化手法を階層型格子向けに拡張した。Oakforest-PACS、Camphor、FX100 で強スケーリング試験を実施し、数十倍の性能向上を達成した。

1. 共同研究に関する情報

(1) 共同研究を実施した拠点名

- 東京大学　情報基盤センター
- 京都大学　学術情報メディアセンター
- 名古屋大学　情報基盤センター

(2) 共同研究分野

- ☐ 超大規模数値計算系応用分野
- ☐ 超大規模データ処理系応用分野
- ☒ 超大容量ネットワーク技術分野
- ☐ 超大規模情報システム関連研究分野

(3) 参加研究者の役割分担

参加者：小山田耕二
役割：PBVR のコード開発
参加者：深沢圭一郎
役割：MHD コードの提供
参加者：山下晋
役割：JUPITER コードの提供

2. 研究の目的と意義

多くの HPC 向けアプリケーションがメニコアアーキテクチャ向けに最適化され、ペタスケールの大規模シミュレーションが日常的に実施されている。結果の解析に必要な可視化に関して、大規模な結果データを手元の PC に転送することは既に困難となっており、計算実行時に同環境を用いて可視化画像を生成する In-Situ 可視化が重視されている。本研究では In-Situ 可視化に関する以下の三つの課題を抽出し、解決を目指している。

1. データ領域構成に伴う大域的通信により、可視化処理のコストがシミュレーション処理のコストを圧迫。
2. バッチ処理投入前に視点位置、色、不透明度等の可視化パラメータを設定するため、可視化の失敗が頻発。
3. シミュレーションリアリティの増加により、多変量向けの可視化技術が必要。

本研究では、可視化用粒子データを用いた新しい In-Situ 可視化システム（In–Situ

PBVR）を構築し、最新のメニーコア環境上に最適化することで、上記課題の解決を目指している。このシステムでは、In-Situ 環境下で結果データを十分小さな可視化用粒子データに圧縮し、その粒子データをクライアントとなる PC に転送することで、バッチ処理実行時の対話的な視点位置の変更が可能である。また、対話ノードで起動されるプログラムを介してクライアント／サーバ間でファイルをやり取りし、色・不透明度関数（伝達関数）などの可視化パラメータが対話的に変更できる。In–Situ PBVR のシステム構成を図 1 に示す。粒子データの生成は領域分割形状を変更することなく並列化され、シミュレーションのスケーラビリティを阻害しない。また多変量向け可視化機能として、代数式によるボリュームデータの合成や伝達関数の合成をリアルタイムに実行できる。

昨年度は、MPI/OpenMP による並列化モデルにより In–Situ PBVR の実装を行い、KNL 上で高いスケーラビリティを示した。しかし、ユーザ指定の代数式を解釈し多変量可視化を行うアルゴリズムがボトルネックとなり、SIMD 並列化が実現できず、KNL の機能を十分に引き出すことができなかった。また、構造格子向けに開発してきた粒子生成技術を階層型格子向けに拡張し、気流のシミュレーションを行う CityLBM への適用を目指したが、未達成であった。

そこで本年度では、SIMD 並列化可能な、代数式による多変量可視化アルゴリズムを構築し、KNL および FX100 上での最適化を実施する。開発システムを CPU や KNL アーキテクチャに最適化された燃料溶融複雑系解析コード[1]および木星磁気圏 MHD コード[2]に適用し、シミュレーションの性能を劣化させること無く対話的な大規模可視化が可能なことを実証する。また、最適化した粒子生成技術を階層型格子向けに拡張し、階層型シミュレーション CityLBM に適用する。

3. 当拠点公募型共同研究として実施した意義

大規模計算の確実な可視化が約束される In-Situ 可視化は、現行あるいは次世代の計算環境に対して汎用的に動作することが求められる。本課題で実施する In-Situ 可視化システムは結合対象となる HPC アプリケーションに合わせてマルチプラットフォームでの動作を想定しており、FX100 や KNL 等の複数のアーキテクチャに対する実装が必要になる。このため、様々なプラットフォームからなる計算資源を利用可能な当拠点公募型共同研究が、本研究の推進に不可欠である。

加えて、拠点間を結ぶ広帯域なネットワークの存在も本研究にとって重要である。開発する In-Situ 可視化システムは、シミュレーションと同時に生成した可視化用粒子デー

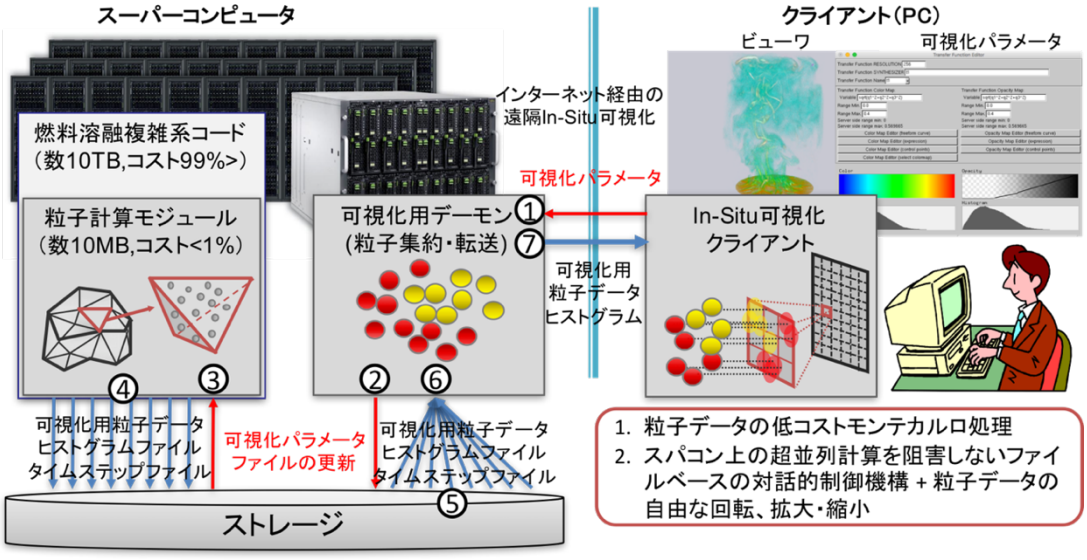

図 1　In–Situ PBVR のシステム構成

タをクライアントに転送することで、遠隔地からのリアルタイムな可視化・解析が可能になる。SINET 5 が提供する広帯域ネットワークを利用することで可視化用粒子データを高速に転送でき、可視化作業を円滑に進めることができる。

4. 前年度までに得られた研究成果の概要

本課題では前年度までに、原子力機構に設置された ICEX（Haswell）上で開発した In-Situ PBVR のシステムを JCAHPC が運用する Oakforest-PACS（KNL）上に移植した。大規模シミュレーションを阻害しないかどうかを検証するため、問題規模を変えずに物理コア数を増加させるストロングスケーリング試験と、使用メモリ量の計測を実施した。その結果、約 10 万コアまでのスケーリングと、シミュレーションの数%程度の省メモリ可視化処理を達成した。

In-Situ PBVR

開発した In-Situ PBVR は、可視化手法として、流体分野で有効性を知られているボリュームレンダリングを採用しており、可視化用粒子データを用いてボリュームレンダリングを実現するためのアルゴリズムとして、粒子ベースボリュームレンダリング（Particle-Based Volume Rendering, PBVR [3]）を利用している。PBVR では、物理値に伝達関数（色関数と不透明度関数）をマッピング、計算結果の格子中にモンテカルロ法で可視化用粒子データを生成し、それを画像面に投影することでボリュームレンダリング画像を生成する。粒子データは元データのサイズに関わらず、十分な画質を得るのに約 250MB の粒子データが必要という結果が得られており、逆に、画質を荒くすることで 3 MB 程度の少ないデータサイズで可視化が可能である。

上記の粒子生成と画像生成の処理を分割することで、遠隔地での実行が可能なシステムとして In-Situ PBVR のシステムを構築した。遠隔地で実行される大規模シミュレーションのデータを小さな可視化用粒子データに変換し、手元のクライアント PC でボリュームレンダリングにより可視化する。そして、バッチ処理されるシミュレーションの実行時における可視化を実現するために、ファイルベースによる可視化パラメータの制御機構を開発した。これらの機能は、シミュレーションに結合される粒子計算モジュール、クライアント PC 上で画像を表示するための In-Situ 可視化クライアント、そしてファイルベースの制御を実行する可視化用デーモンにより構成されている（図 1）。

In-Situ 可視化クライアント

In-Situ 可視化クライアントは、クライアント PC 上で起動され、可視化結果を表示するためのビューワ、そして多変量可視化を実現するための機能である“伝達関数合成器”（TFS）[4]から構成されている。TFS は、結果データに含まれる変量を組み合わせ新しいボリュームデータを生成するボリュームデータ合成機能と、複数の伝達関数を組み合わせる伝達関数合成機能を備えている。ユーザは TFS 上で代数式により合成関数を指定する。代数式は可視化パラメータとして粒子計算モジュールに転送され、数式処理機能によりリアルタイムにボリュームデータ・伝達関数の合成が行われる。

粒子計算モジュール

シミュレーションに結合される粒子計算モジュールは、C++で記述され、シミュレーションの領域分割に対応した MPI 並列化、そして各部分領域に対しては OpenMP による要素並列の粒子生成が実装されている。この粒子計算モジュールはライブラリとしてまとめられ、シミュレーションの各タイムステップにおける MPI プロセスから関数を一つ呼び出すだけで結合が可能なように設計されている。

粒子計算モジュールは、各タイムステップ

で生成した粒子データや計算結果のヒストグラム、そしてタイムステップ情報をファイルとしてストレージ上に出力する。粒子データファイルは、各プロセスが独立に出力するため、コストの高い同期や集団通信を必要としない。

可視化用デーモン

可視化用データは対話ノード上で、あるいは対話ジョブとして起動される。そしてストレージ上の粒子ファイル、ヒストグラムファイル、タイムステップファイルを収集し、In-Situ 可視化クライアントに送信する役割がある。図 1 に可視化用デーモンを中心とする In-Situ PBVR の動作メカニズムを示す。

1. 可視化用デーモンは、In-Situ 可視化クライアントから可視化パラメータを受信する。
2. 可視化用デーモンはストレージ上の可視化パラメータファイルを更新する。
3. 粒子計算モジュールはストレージ上の可視化パラメータファイルが更新された場合、各プロセスにブロードキャストする。
4. 粒子計算モジュールは、粒子データファイル、ヒストグラムファイル、タイムステップファイルを出力する。
5. 可視化用デーモンがタイムステップファイルの更新を検出する。
6. 可視化用デーモンは、全プロセス分の粒子データファイルが全て揃っているかどうかを調べる。
7. 粒子データを集約し、クライアントへ転送、クライアント上でレンダリングする。

上記手順 6 の処理により、粒子データファイルの集約は粒子計算モジュールと完全に非同期に実行することが可能になっている（ファイルベース制御機構）。

実験結果

In-Situ PBVR を燃料溶融複雑系解析コード JUPITER に結合し、Oakforest-PACS (OFP)の約 10 万コアまでを使用して、ストロングスケーリング試験と使用メモリ量の計測を実施した。

JUPITER は 3 次元の領域分割された構造格子上の MPI/OpenMP ハイブリッド並列モデルによって非圧縮流体モデルと Volume of Fluid 法に基づく多相多成分熱流動解析を処理する。典型的なシミュレーション時間は約 300 万タイムステップであり、本来は 1000 ステップ毎に可視化されていが、本実験では毎タイムステップに In-Situ PBVR による可視化を行なった。問題規模は 240x240x1920（約 1 億格子）に固定し、可視化用粒子は約 1000 万粒子（約 250MB）、画面解像度は 1024x1024 を使用した。本実験において、1 ノードあたり 64 コアを使用し、スレッド数を 16、プロセス数を 4 に固定した。MCDRAM をキャッシュモードで使用し、I/O システムはラウンドロビン分散の並列ファイルシステムを使用した。

ストロングスケーリング試験の結果を図 2 に示す。図 2 においてソルバと PBVR（KNL,orig.）がそれぞれ、JUPITER と粒子計算モジュールの 1 タイムステップの処理時間である。粒子計算モジュールの処理時間は JUPITER より高速で、8%-28%以内に抑えられており、約 10 万コアまでスケールした。

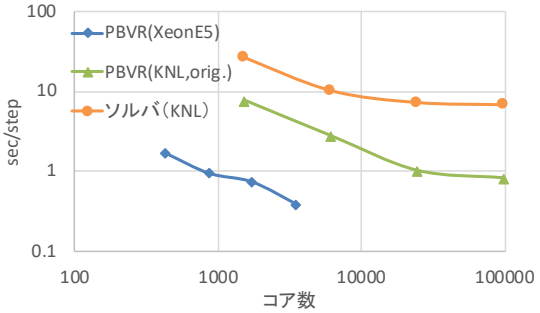

図 2　In-Situ PBVR と JUPITER の性能

表 1 の “JUPITER” と “PBVR” はそれぞれ、JUPITER と粒子計算モジュールの消費メモリ量を示している。固定した問題規模にも関わらず、袖領域と MPI バッファによって JUPITER のメモリ消費量はコア数に従って

増加している。他方、PBVR のメモリ消費量は JUPITER と比べて微増であり、各 MPI プロセスのメモリ消費量は約 3.3MB-92MB に抑えられている。

表 1　メモリ消費量

コア数	1,536	6,144	24,576	98,304
JUPITER[GB]	106.7	135.0	256.7	773.2
PBVR[GB]	8.9	9.5	11.6	20.4

問題点

図 2 では、約 10 万コアでスケーラビリティが劣化している。図 3 のコスト分布では、粒子ファイルの書き出しコストが増加していることがわかる。粒子データサイズは約 250MB とかなり小さいため、この遅延は多数ファイル（980,304 コアで 6,144 ファイル）の書き出しに伴うメタデータサーバーの遅延が原因であると推定される。また、図 2 において PBVR(XeonE5)としてプロットした Haswell を搭載した ICEX 上の計測結果と比較すると一桁近く低速である。これは、多変量可視化のための数式処理において、コストの高い文字列のパースを繰り返し、なおかつ SIMD 演算が利用できていないことが原因である。これを解決するために、将来のエクサスケール計算機への適用を見据えて、KNL の SIMD 演算を利用した高速化が必要となっている。

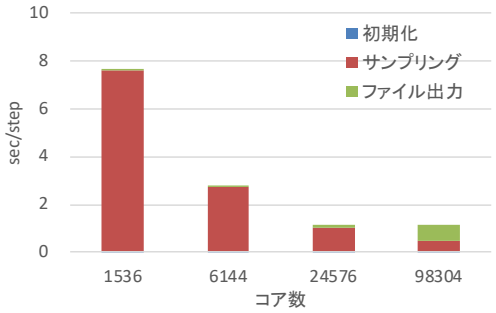

図 3　In-Situ PBVR のコスト分布

5.　今年度の研究成果の詳細

昨年度に開発した In-Situ PBVR の課題解決に向けて、粒子ファイル書き出しコストを隠蔽するためのタスク並列処理と、文字列のパースを減らし SIMD 化に適した数式処理機能を開発した。

数式処理機能（前期）

オリジナルの数式処理は、粒子計算モジュール上で、ユーザ入力の中置記法による代数式を構文解析している。中置記法は演算子の左右に被演算子が配置される数式の記述方法であり、例えば和よりも積を先に計算する等の演算子の優先順位が存在し、更に括弧によるネストによっても演算順序が変化する。そのため中置記法の処理には代数式の文字列に対する再帰的な走査が必要になる（図 4 左）。このように再帰的に繰り返される文字列処理は SIMD 化されないため KNL の性能を引き出すことができなかった。

この問題を解決するために、逆ポーランド記法(RPN)を利用して SIMD 向けの数式処理機能を開発した。RPN は被演算子の後ろに演算子が置かれる記法であり、前方から順に演算していく。RPN を計算するアルゴリズムではスタックを用意し、数式を前方から順に操作していく。そして文字が被演算子ならスタックに値をプッシュし、演算子なら値をポップし演算結果をスタックにプッシュする（図 4 右）。このアルゴリズムでは一回の走査で計算が完了するため SIMD 化に適している。本手法では、数式処理を式の変換と計算実行の部分に分割し、In-Situ 可視化クライアント上でユーザの入力した中置記法を RPN に変換し、RPN の逐次計算を粒子計算モジュールで実行する。KNL の SIMD アーキテクチャを利用するためにスタック配列を SIMD 幅（単精度 16 要素）の 2 次元配列に拡張し、16 粒子ごとに RPN を処理する。そして演算子に紐付けられた関数は SIMD 命令によって計算される。

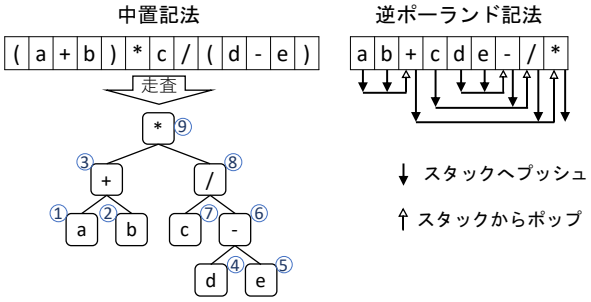

図 4　数式処理の概略

非同期ファイル出力（前期）

粒子計算モジュールの各プロセスで実行される粒子ファイルの出力を隠蔽するために、シミュレーションとファイル出力をタスク並列で実行する。スレッドベースのタスク並列の実装方法として OpenMP、pthread、Posix-AIO が候補となりうるが、本課題では C++11 以降で利用できる std::thread を選択した。std::thread はシミュレーションの OpenMP や pthread と干渉せず、データ I/O 用の C ラッパー(I/Othread)を実装することで多彩な API を設計可能にする。

I/Othread はシミュレーションコードのタイムステップループと同じ階層で定義され、粒子計算モジュール内部で利用される。粒子生成後に I/Othread 内でファイル出力関数が std::thraed に登録され、新しく生成されたスレッドでシミュレーションと独立して I/O が実行される。

OFP, Camphor での最適化コードの性能（前期）

昨年度と同様の条件で、提案手法の性能を測定した。図 5 の PBVR(KNL,opt.)に SIMD 向けの数式処理と非同期 I/O を実装した結果を、図 6 にコスト分布を示す。最適化された粒子計算モジュールはオリジナルと比べて約 13 倍から 17 倍にまで高速化された。それにより粒子計算モジュールの最大コストを 28%から 0.7%までに抑えることができた。またファイル出力コストはほとんど無視できるコストにまで減らすことができた。

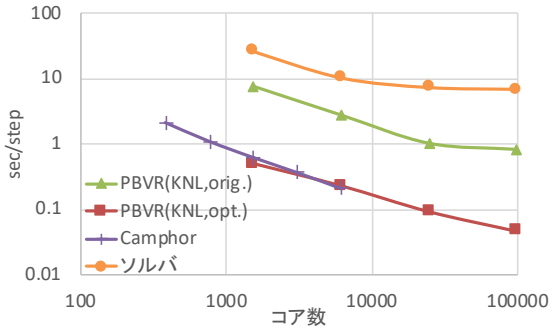

図 5　In-Situ PBVR と JUPITER の性能(OFP, Camphor)

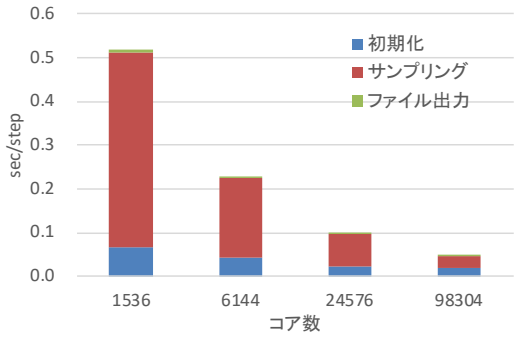

図 6　In-Situ PBVR のコスト分布(OFP)

最適化した手法を京大 Camphor に移植し、384 から 6144 コアまで増加させ、OFP と同様のストロングスケーリング試験を実施した。結果を図 5 の Camphor に示す。MPI ライブラリやネットワークの仕様の違いはあったが両機とも KNL を搭載しており、ほぼ連続する結果となっている。

FX100 での最適化コードの性能（後期）

最適化前後の PBVR を FX100 上に移植した。FX100 上では、コードのコンパイルに FUJITSU コンパイラを使用したが、コンパイラの仕様上 pthread で実装された機能が利用できなかった。そのため内部で pthread を利用している std::thread も利用できず、非同期ファイル出力が利用できなかった。そこで本研究では、通信負荷を可能な限り低減しつつファイル出力数が抑えられるノード内集約機能を実装した。

移植した PBVR を JUPITER に結合して性能測定を行った。生成粒子数と画像解像度は OFP

に合わせて 1,000 万粒子、1024^2 であり、JUPITER の格子数を約 1 億に固定したままコア数を 384 から 24576 まで変化させて強スケーリング試験を実施した。図 7 にコア数に対する処理時間を示す。最適化前は Solver より低速だった PBVR が、提案した最適化によって 100 倍以上加速された。図 8 にコスト分布を示す。非同期ファイル出力が実装された OFP の結果と比較するとファイル出力時間が顕著だが、実験に用いたコア数の範囲ではファイル出力時間は概ねスケールしていた。

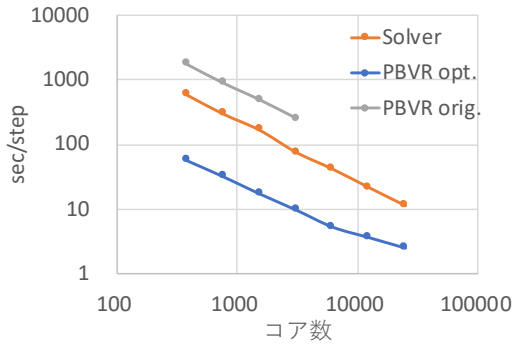

図 7　In-Situ PBVR と JUPITER の性能 (FX100)

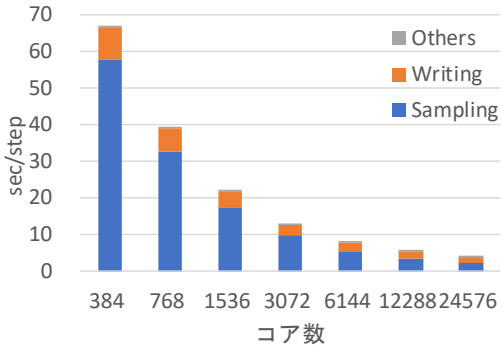

図 8　In-Situ PBVR のコスト分布 (OFP)

階層格子向け粒子生成機能の開発

最適化した In-Situ PBVR の粒子生成処理を階層型格子に拡張したこの拡張ではメニーコア計算機のメモリレイアウトに最適化された階層格子構造（Block Structured AMR）をサポートしている。このタイプの格子は、N^3 の直交格子を最小の処理領域の単位（リーフ）

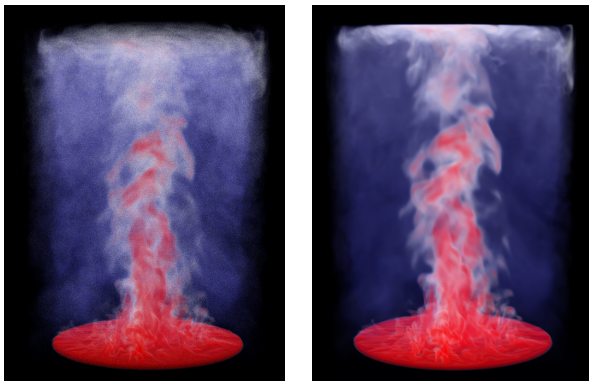

図 9　CityLBM の温度場の可視化結果
左：In-Situ PBVR、右：ParaView

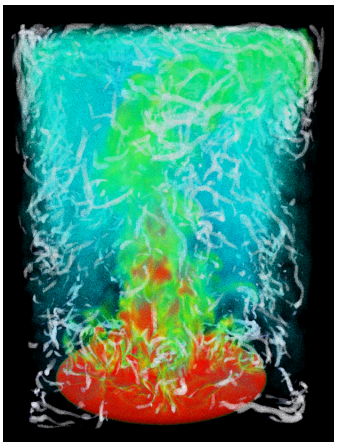

図 10　CityLBM の Q 値と温度場の合成可視化結果

と定義し、各階層でサイズの異なるリーフが接続されている。そのため Block Structured AMR は NxNxNxL(L はリーフ数)の四次元格子として定義される。

リーフ内部の処理は前期に開発した直交格子向けの SIMD 化された手法を拡張して開発した。階層レベルから格子体積を計算し生成粒子数を制御することで、階層間でも滑らかな可視化が可能なようにした。また、伝達関数で使用する微分値の計算に関しても、格子のサイズをヤコビアンに反映して計算する機能を開発した。

開発した手法を GPGPU クラスタで動作する階層型流体解析コード CityLBM による熱流動計算に適用し、温度場に対して既存の可視化アプリケーションである ParaView と

表 2　可視化性能の比較(帯域幅は ~11 [MB/sec])

	In-Situ PBVR	ParaView
データサイズ	581 MB	5144 MB
粒子生成	38 [sec/step]	
データ転送	52 [sec/step]	456 [sec/step]
可視化画像生成	7.7 [sec/step]	900 [sec/step]

の比較を行った。また代数式の機能を用いて Q 値を計算し、温度場との合成表示で可視化した。図 9 に In–Situ PBVR および ParaView で温度場を可視化した結果を示す。この図より In–Situ PBVR は従来可視化アプリと同等の結果が得られることが確認できる。また、ParaView では CityLBM を In–Situ 可視化することができず、シミュレーションデータを出力し実験を行った。表 2 には In–Situ PBVR の処理時間と、出力したデータを転送し ParaView で可視化するまでの時間を示している。この比較では In–Situ PBVR により約３０倍近く高速に可視化結果を得られている。また図 10 に示す結果に関して、可視化アプリ側で微分を含む変数の合成を行い、さらに他の変数と合成して表示する機能は In–Situ PBVR 独自のものであり、データ出力が困難になるエクサスケールシミュレーションで必要とされるポスト処理を効率的に行えるようになることが期待できる。

6. 今年度の進捗状況と今後の展望

本年度は、メニコア CPU 向けアルゴリズムの開発、OFP / Camphor / FX100 への移植と性能測定、そして階層型格子向けのコードの作成と実験を実施した。最適化によって OFP は 10 倍以上、FX100 では 100 倍もの高速化を達成できた。階層型コードでは可視化アプリケーション ParaView と比較して同等の結果を 30 倍高速に得られることを示した。

しかし MHD シミュレーションへの In-Situ PBVR の適用が未完で終わってしまった。原因としては、In-Situ PBVR が C++で開発されていたのに対して、MHD コードが FORTRAN で開発されていたため、ラッパーの作成に当初の想定を超えた作業が発生してしまったことが挙げられる。In-Situ PBVR は可視化用の多変量ボリュームデータを C 言語で定義される 2 次元配列として利用している。しかし C 言語の 2 次元配列は FORTRAN の 2 次元配列と仕様が異なり変換ができなかった。これを修正するために In-Situ PBVR の大幅な書き換えが必要になり、作業量が大幅に増加し、期間内で作業を完遂することが困難になった。

今後の展望としては MHD コードへの適用を完了し、シミュレーションのデータ同化から得られるアンサンブルデータの可視化技術を開発したい。

7. 研究成果リスト

(1) 学術論文

(2) 国際会議プロシーディングス

(3) 国際会議発表

(4) 国内会議発表

河村拓馬, 井戸村泰宏 “In-situ PBVR の SIMD 最適化”, 第 46 回可視化情報シンポジウム, 明治大学, 2018.

(5) その他（特許，プレス発表，著書等）

参考文献

[1] S. Yamashita, T. Ina, Y. Idomura, H. Yoshida, “A numerical simulation method for molten material behavior in nuclear reactors”, Nuclear Engineering and Design, vol. 332, pp. 301-312, 2017.

[2] K. Fukazawa, T. Nanri and T. Umeda, “Performance Measurements of MHD Simulation for Planetary Magnetosphere on Peta-Scale Computer FX10”, Parallel Computing: Accelerating Computational Science and Engineering (CSE), Advances in Parallel Computing 25, pp.387-394, IOS Press, 2014.

[3] Takuma Kawamura, Naohisa Sakamoto, Koji Koyamada, "A High Quality Sampling Technique for Particle-based Volume Rendering", IEEE Visualization(Poster), 2009.

[4] Takuma Kwamura, Yasuhiro Idomura, Hiroko Miyamura, Hiroshi Takemiya, “Algebraic design of multi-dimensional transfer function using transfer function synthesizer”, Journal of Visualization, vol. 20, No. 1, pp. 151-162, 2016.