対話型シミュレーションの実時間遠隔共有のための シミュレーションキャッシングフレームワーク

山本　優[1,†1,a)]　熊田　佳孝[1]　張　家超[1]　福間　慎治[1]　森　眞一郎[1,b)]

概要：我々は，遠隔地の高性能コンピュータ上でのシミュレーションにおいて，対話的な操作を可能とする対話型遠隔シミュレーションシステムの実現を目指して研究を行っている．これまでに，通信遅延に伴う対話性の低減を抑える手法としてシミュレーションキャッシングとよぶ手法を提案し，この手法を利用するためのフレームワークを開発してきた．本論文では，シミュレーションキャッシングを様々なアプリケーションに対して容易に適用可能にすると共に，複数のユーザ間でシミュレーションの実時間遠隔共有と協調作業を可能にするマルチクライアントシミュレーションキャッシングフレームワークの実装について報告する．

キーワード：対話型シミュレーション，シミュレーション結果の遠隔共有，協調作業，遅延隠蔽，一貫性制御

Simulation Caching Framework for Real-time Remote Sharing of Interactive Simulation.

Yu Yamamoto[1,†1,a)]　Yoshitaka Kumada[1]　Zhang Jiachao[1]　Shinji Fukuma[1]
Shin-ichiro Mori[1,b)]

Abstract: In order to realize interactive remote simulation steering system for HPC environment, we have to conquer the problem of network latency. For this purpose, we have been proposed a simulation model which we referred to it as “Simulation Caching”. In this paper, we propose an extended Simulation Caching Framework that can work under the multi-user environment where multiple users can share and steer the ongoing remote simulation simultaneously, to realize collaborative real-time simulation.

Keywords: Interactive simulation, Remote sharing of simulation result, Collaborative work, Latency hiding, Consistency control

1. はじめに

近年の計算機性能の急速な向上に伴い，対話的な実時間シミュレーションへの期待が高まっている．フライトシミュレーション [1] や航空管制シミュレーションのようにコンピュータ上のシミュレーション結果を操作者が直接体感し，その反応として対話的にシミュレーションをステアリング可能なシミュレーションの形態は “human-in-the-loop simulation” あるいは “interactive simulation” と呼ばれる．近年盛んに研究が行われている，災害時の緊急避難シミュレーションなどもその 1 例である．従来，このようなシミュレータ上で行われてきたシミュレーションは主に離散事象シミュレーションであったが，これをスーパーコンピュータ上の科学技術計算のシミュレーションにも拡張する試みも進められている [2], [3], [4]．しかし，実時間での対話的なステアリングまでを考慮した研究は始まったばかりである [5], [6]．これに対して我々は，遠隔地のサーバで行うシミュレーションの一部を通信遅延の少ない操作端末

[1] 福井大学大学院工学研究科 (University of Fukui)
[†1] 現在，(株) 富士通//FUJITSU Ltd.
[a)] yyu@sylph.fuis.u-fukui.ac.jp
[b)] moris@u-fukui.ac.jp

の近くで行い，2 つのシミュレーション結果を連携させて通信遅延を隠蔽する “シミュレーションキャッシング”[7] の研究を行ってきた．このようなシミュレーションキャッシング技術を応用した対話型遠隔シミュレーションが容易に実行可能となれば，スーパーコンピュータなどの高性能計算資源活用の敷居が下がり，実時間指向の幅広いユーザ層への HPC の普及と発展が期待できる．

さらに本研究では，これまでに開発したシミュレーションキャッシング技術 [8] を背景として，次世代の E-Science を推進するために計算機上の仮想実験室で行なわれる実験（実時間シミュレーション）での協調学習，教育ならびに協調作業の実技トレーニング等を可能する環境の構築を目指している．このようなシミュレーション環境はバッチ処理が主流の現在の大規模スーパーコンピュータ環境で直ちに利用可能な技術ではないが，仮想化技術の高効率化によりクラウドコンピューティング環境ではすでに大規模計算資源の対話型実時間利用が可能になりつつある背景を受け，近い将来必要となるシミュレーション実行環境の一つであると考える．本論文では，このように対話型遠隔シミュレーションシステムを複数人で利用することを目的とし，サーバで実行中のシミュレーションに対して複数のユーザが同時にステアリング可能にすると共に，シミュレーション結果を共有できるシステム（**図 1**）のフレームワーク実装を行う．ここでステアリングとは，シミュレーションの境界条件の変更やパラメータの変更，スナップショットを残した過去の時刻への巻き戻し等のシミュレーション自体への対話的な操作，さらには，シミュレーション結果の可視化パラメータの変更等を含む．例えば，複数の攪拌棒を同時に操作する流体攪拌シミュレーション [7] などでは，攪拌棒の位置情報の変更なども境界条件の変更に相当する．このようなシミュレーション環境（フレームワーク）の構築においては，フレームワークとしての利便性はもちろん，シミュレーション・キャッシングを行う際の複数クライアントの管理，各クライアント間の一貫性補償などが課題となる．

2. 関連研究

まず最初に，遠隔地にある均質/非均質な計算機資源を連携した協調計算のフレームワークに関する研究としては，協調計算のための通信ライブラリに関する研究や協調計算を行なう計算機環境に関する研究がこれまでも行なわれてきた．例えば，原子力研究所では異機種計算機間で MPI をつかった協調計算を可能にする通信ライブラリ Stampi[10] が開発されており，アーキテクチャの異なるスーパーコンピュータを連携したマルチフィジックスシミュレーションなどの研究が行われた．また最近では，大学等が所有するプライベートクラウドシステムを連携させ広域分散型のインタークラウドシステムを実現するための遠隔連携技術に

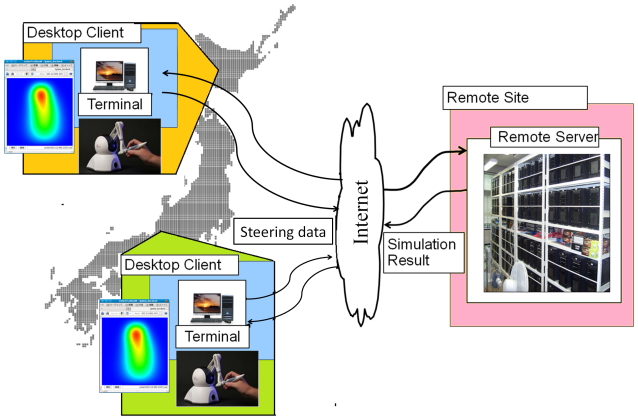


図 1 マルチクライアント対応の対話型遠隔シミュレーションシステム

関する研究 [11], [12] も行われている．しかしながら，これらの従来研究はスループット重視のアプリケーションを指向したものが大半であり，我々がターゲットとしている実時間指向のアプリケーションへの適用に関する議論は十分に行われていない．

実行中の大規模シミュレーションの遠隔モニタリングに関する関連研究としては，SIMONsystem[13]，RSVLIB[14]，VisTrace[15] などがあるが，いずれもシミュレーション自体のステアリングに関する機能は有していない．

統合的なシミュレーション・可視化環境の関連研究としては，複数のシミュレーションを統合し，シミュレーション実行，可視化，対話的な分析とステアリングをサポートする環境として World Lines[16] がある．これは境界条件の異なる複数のシミュレーションを 1 つのプラットフォーム内で制御し，過去のステアリング情報を確認・再計算が可能である．しかしながら，大規模計算機との遠隔協調シミュレーションへの応用については検討が行われていない．

スーパーコンピュータベースの遠隔シミュレーションステアリング環境の関連研究例としては COVISE[17] がある．これはシミュレーションの実行、可視化機能を統合するための分散ソフトウェア環境であり，機械設計などにおける工程（設計，モデリング, メッシュ，シミュレーション，可視化，解析）が 1 つのプラットフォームで実行可能となる．また，Web ベースのインターフェースを導入することでユビキタスなアクセスと協調作業を可能にしている．しかしながら，実時間でのシミュレーションステアリングについては考慮されていない．

シミュレーション結果の遠隔共有に関する研究としては，SAGE[18], Paraview[19], VTK[20] などの可視化ミドルウェアに関する研究やスパーコンピュータ「京」での実時間利用を想定した HIVE[21] などがあるが，複数地点への可視化処理後のシミュレーション結果の遠隔配信が主目的であり，一部で対話的な可視化パラメータの変更が可能なものもあるが基本的には単方向のストリーミングであ

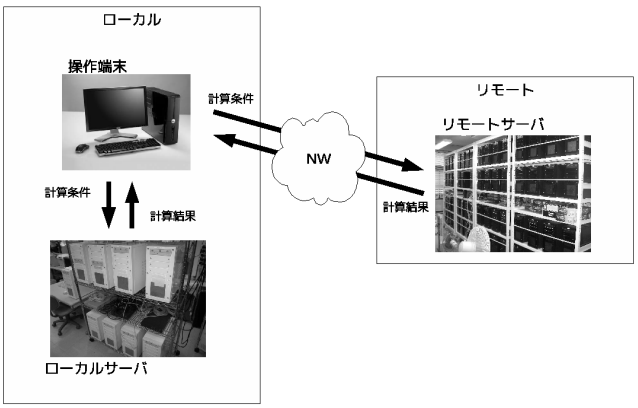


図 2 シミュレーションキャッシングを用いた対話型遠隔シミュレーションシステム

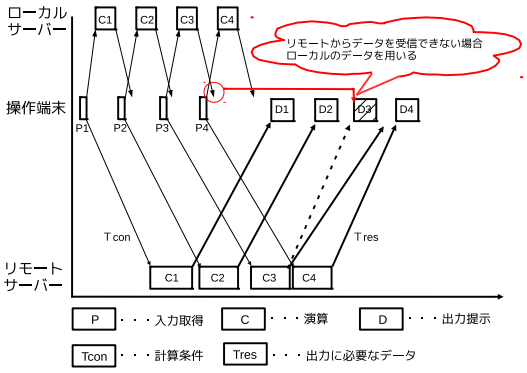


図 3 シミュレーションキャッシングの動作

る．そのためシミュレーション自体のステアリング機能は備えていない．

これらの先行研究に対して，遠隔地の大規模計算機上で実行中のシミュレーションに対し，遅延隠蔽技術を用いて複数人で同時に実時間ステアリングと可視化を可能にするシミュレーションフレームワークの提案を行うところに本研究の特徴がある．

3. 研究背景

3.1 対話型遠隔シミュレーション

手元にある計算機では実行不可能な大規模シミュレーションを，遠隔地にある高性能な計算サーバ上で実行し，ネットワークを通じて計算条件や結果を対話的に通信することで遠隔操作するシステムである．前提として，リモートサーバは対象とするシミュレーションの実行に対して十分な計算性能を持っているものとする．このシステムは，クライアント側を操作端末としてシミュレーションをステアリングし，サーバ上でステアリングされたシミュレーションの新しい条件から数値計算を行い，結果データを操作端末にフィードバックすることによって操作端末の負荷を軽減し，手元にある計算機だけでは不可能な大規模シミュレーションを実現するものである．

3.2 シミュレーションキャッシング

シミュレーションキャッシングで想定する対話型遠隔シミュレーションは，遠隔地の高性能シミュレーションサーバ（リモートサーバ）と，ユーザの操作や結果の提示を行う操作端末と，その近傍にあるパーソナルシミュレーションサーバ（ローカルサーバ）で構成されるクライアントサーバ型のシステム [7] である（**図 2**）．通信遅延が存在する環境下においても，一定間隔でのデータ出力を補償する．

3.2.1 遅延隠蔽手法

遠隔シミュレーションでは通信路の遅延変動（ジッタ）により，通信時間に変動が生じる．この問題に対して，シミュレーションキャッシングでは，リモートサーバで実行中のシミュレーションの一部をローカルサーバ上でも並列に冗長実行し操作端末に結果をストックする．ユーザのインタラクションに対してリモートサーバが即応できない場合でもストックデータで即応可能である（**図 3**）．2 つのサーバの連携により予測不可能なジッタの影響を隠蔽し実時間インタラクションを可能にする手法である．

3.2.2 サーバ間連携法

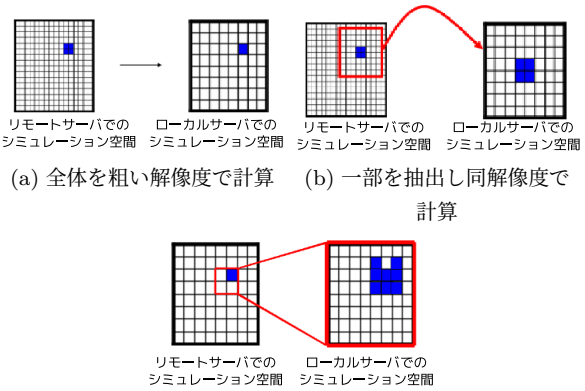
リモートサーバとローカルサーバ間の連携法は様々な形態が考えられる．例えば計算結果の提示において，出力データが精度に対して厳しい制約がある場合は多少遅延があったとしてもリモートサーバの結果を使用し，逆に応答時間に厳しい制約がある場合は多少精度が落ちたとしてもローカルサーバの結果を提示するといった方法がある．様々な連携法が考えられるがその方向性は基本的には，シミュレーション自体の時間・空間解像度の組み合わせによる連携とシミュレーションに用いる数値計算手法の組み合わせによる連携に大別される．それぞれの分類において，代表的な連携法の例を以下に示す．

(1) 時間・空間解像度の組み合わせによる連携

- リモートサーバで計算するシミュレーション領域全体を粗い解像度でシミュレーションする（**図 4**-(a)）．シミュレーション領域が 2 次元の場合，解像度が縦横 $1/N$ とすると計算量は $1/N^2$ となる．時間方向の計算量も $1/N$ となり全体として $1/N^3$ となる場合もある．
- シミュレーションする領域を限定し，リモートサーバで行っている領域の一部についてシミュレーションを行う（図 4-(b)）．縦横 $1/N$ の領域を選択しシミュレーションする場合，計算量は $1/N^2$ となる．
- シミュレーション領域の一部を高解像度でシミュレーションする（図 4-(c)）．注目領域内においてはリモートサーバよりも高精度のシミュレーションを行う．

(2) 数値計算手法の組み合わせ

- 異なる数値計算モデルを使用する．打ち切り誤差を



(a) 全体を粗い解像度で計算　(b) 一部を抽出し同解像度で計算

(c) 一部を抽出し高解像度で計算

図 4　リモート・ローカルサーバ間連携法

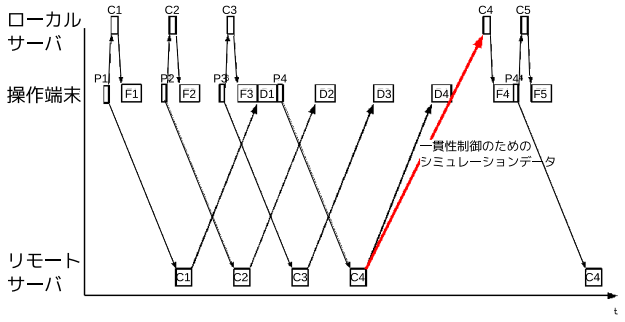


図 5　中断モデル

大きくする等，計算精度とトレードオフの関係にある計算モデルやアルゴリズムを使用する．

- 異なる数値精度を使用する．リモートでは倍精度で計算しているところを，単精度実数を用いる．

これらは適用するシミュレーションの内容および計算環境を加味した上で決定する必要がある．

3.2.3　一貫性制御

シミュレーションキャッシングを実行する時，シミュレーションが時間的な依存関係を持つ場合，シミュレーションが進むに連れて各サーバでの計算結果の誤差が蓄積する問題が発生する．そこで，シミュレーションがある程度進行した時点でシミュレーション精度の高いサーバのシミュレーション内容をもう片方のサーバに送信し反映する．これを一貫性制御と呼び，これまでに中断モデルとロールバックモデルを提案している．

中断モデルは，一貫性制御が終了するまで入力を一時中断し，データの一貫性が保証されるまでの間シミュレーションをストップさせ，一貫性制御データの反映が終わってからリスタートする（**図 5**）．ロールバックモデルはシミュレーションを継続したまま一貫性制御を見かけ上同時並列して実行し，タイムステップが一致した時点でデータを更新する．中断モデルでの一貫性制御は，シミュレーションが一時中断してしまうため，応答時間が大きい場合や頻繁に一貫性制御を行う場合不向きであるが，一貫性制御のタイミングを確実に保証する．これに対してロールバックモデルは，シミュレーションと同時進行して過去の計算条件を利用し，一貫性制御を行うためシミュレーションが中断することを避けられるが，ローカルサーバでの計算量が増え，明確にどのタイミングでデータの一貫性が保証されるか不定である．

一貫性制御は，対象とするシミュレーション内容によって送信するデータの形式（シミュレーション結果データ，圧縮したデータ，可視化画像など）や制御モデル，一貫性制御実施の頻度をカスタマイズする必要がある．

4.　マルチクライアントシミュレーションキャッシングフレームワークの概要

4.1　要件・実装方針

遅延環境の大規模シミュレーションにおいて一定間隔でのデータ出力を補償し，複数のクライアントでシミュレーションのステアリングと可視化結果の共有ができるシステム（図 1）の構築を行う．シングルクライアントのシミュレーションキャッシングフレームワーク（以降，従来フレームワーク）[9] の機能拡張を行いマルチクライアントシミュレーションキャッシングフレームワーク（以降，本フレームワーク）の実装を行う．想定するシステムのハードウェア構成は，リモートサーバとユーザ毎の操作端末・ローカルサーバを全てネットワークで結んだものである．また，近年の同一端末での並列実効性の高さを受けて，操作端末とローカルサーバを同一端末として運用する構成にも移行できるようにする．

4.1.1　機能要件

従来フレームワークは以下の機能を搭載している．

- 操作端末から各サーバへの計算条件の送受信
- 各サーバから操作端末への計算結果の送受信
- パイプライン実現のための時間管理
- 一貫性制御処理
- 操作端末における計算結果の自動選択
- 各プロセス間での同期
- スロースタートアルゴリズム（SSA）対策

本フレームワークを実現するためには次のような機能の追加が必要となる．

- 不特定多数のクライアントの管理
- 不特定多数のローカルサーバ間でシミュレーションの遠隔共有
- 複数の入出力情報の管理
- マルチクライアント対応の一貫性制御

4.1.2　実装方針

我々はフレームワークの実装を行う上でマルチクライアント対応可能な運用モデルとして，複数のユーザが同時にステアリングと可視化を行う MOMM（Multi Operator

Multi Monitor）モデルの実現を目指す．そのためには各クライアントのローカルサーバ間の一貫性補償が必要となり，従来フレームワークの一貫性制御手法だけでは実現は難しく，新たな一貫性制御手法の実装が必要となる．クライアント間の一貫性制御手法としては，共有メモリ型並列計算機におけるキャッシュメモリの一貫性制御と同様に様々な実装法が考えられるが，基本方針として各クライアントのローカルサーバ間で直接一貫性制御を行なうか，一旦リモートサーバを介して間接的にローカルサーバ間の一貫性制御を行なうかの選択が必要である．本研究では実装の容易さ，ならびに，クライアント間に発生する各種の依存性の排除を目的として，後者の間接的なクライアント間一貫性制御を実装することとした．

4.2 仕様・システム設計

C 言語を想定し，各ユーザドメイン内プロセス間通信には MPI，クライアント・サーバ間通信には Socket を使用する [22]．リモートサーバプロセスを独立して動作させ，各クライアントはドメイン I/F を経由して入出力を行う．この時，リモートサーバプロセスと各クライアントのドメイン I/F との通信は Socket ベースで実装を行い，不特定多数のクライアントが任意のタイミングでサーバに接続・退去できる構成とする．Socket ベースの通信を用いることで，クライアントの通信障害等を含む故障や通信遅延増加の影響がリモートサーバや他クライアントに波及することを防ぐことが可能である．

5. マルチクライアントシミュレーションキャッシングフレームワークの実装

5.1 一貫性制御と入力データー貫性制御

本フレームワークでは，ローカルサーバへの負荷が低く，実装が容易な中断モデルの一貫性制御を利用しシステム全体で緩い一貫性を補償する．一貫性制御は，クライアントがサーバへ接続したときとリモートサーバのマスタタイムステップが一定数経過する毎に実施する．一貫性制御実施直後であれば，各ローカルサーバは他クライアントによる影響が正確に反映されたシミュレーション結果を提示できる．しかし，次の一貫性制御のタイミングまで他クライアントの影響が反映されなくなり，リモートサーバのシミュレーション結果とのずれが大きくなる．

そこで，従来フレームワークで実装されていたシミュレーション結果の一貫性制御に加えて入力データ一貫性制御を併用する手法を新たに実装した．これは従来の一貫性制御に使っていたシミュレーション結果データよりも，相対的にデータ量が少ない入力データを利用し一貫性制御を行う手法である．これにより，ローカルサーバに全クライアントの入力データが反映されるようになり，シミュレーションの質を上げることができる．リモートサーバでは毎

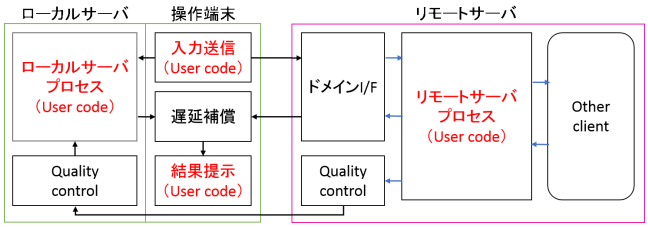


図 6 マルチクライアントシミュレーションキャッシングフレームワーク

ステップ受け取った全クライアントの入力データをまとめ，各クライアントのクオリティコントロール（QC）プロセスへ配信する．QC プロセスでは入力データをチェックして自身の入力データは除外した上で保存し，設定した間隔で貯まった入力データをローカルサーバプロセスへ送信してシミュレーションへ反映させる．入力データ一貫性制御は，基本的に通常の一貫性制御の間隔より小さく設定する．ローカルサーバは入力データを反映するため待機状態になるが，先行して動作していることから，ユーザへローカルサーバシミュレーションの結果データを提示するタイミングに間に合うようパラメータを設定することで効果的に運用できる．

5.2 新たなプロセスの機能

本フレームワークの構成を**図 6** に示す．実際に使用する際は，はじめにリモートサーバプロセスを立ち上げた後，クライアント毎に 7 つプロセスを起動しサーバへ接続しシミュレーションを実行する．この時，シミュレーションキャッシングに関するパラメータ（一貫性制御間隔など）はサーバと各クライアントで同様の数値を用いる必要がある．ローカルサーバで実行するシミュレーション本体のパラメータは，各自ローカルサーバの性能に見合った調節が可能である．リモート・ローカルの各サーバプロセスは入力データ一貫性制御を導入，リモートサーバプロセスにはマルチクライアントに対応させるため Socket を導入し複数のクライアントを管理できる枠組みを構築した．以下，本フレームワークで新たに追加したプロセスや従来フレームワークから大きく変更のあったプロセスの説明を行う．

5.2.1 サーバプロセス

リモートおよびローカルの各サーバプロセスは，入力データの受信，シミュレーションの実行，結果データの可視化（任意），結果の送信，一貫性制御，入力データ一貫性制御が主な処理である．この内，フレームワークではシミュレーション以外の処理をそれぞれ関数として提供する．リモートサーバプロセスは独立して動作するプロセスであり，Socket の Select 関数を用いてクライアントの接続や入力データを受け付けシミュレーションを実行する部分と可視化データ，一貫性制御データを配信する 2 つの部分に分かれている．**図 7**，**図 8** に処理の流れを示す．ステア

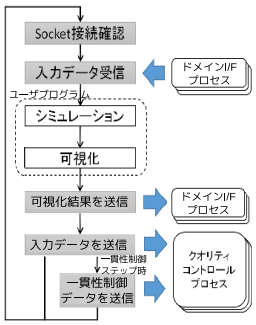


図 7 リモートサーバプロセスの動き

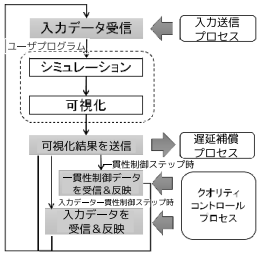


図 8 ローカルサーバプロセスの動き

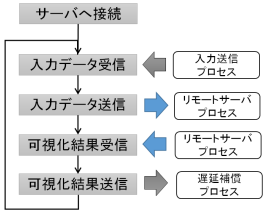


図 9 ドメイン I/F プロセスの動き

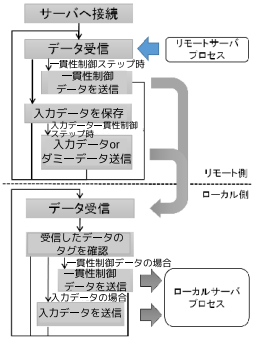


図 10 QC プロセスの動き

リング入力を受け取り，入力を反映したシミュレーションを実行後，シミュレーション結果の可視化処理を行ない，その結果を結果提示のため送出するという処理は共通であるが，リモートサーバプロセスでは，これに加えてクライアントの死活管理のための Socket 接続確認処理，入力データ一貫性制御のための入力データ配信，シミュレーション結果の一貫性制御のためのデータ送信処理が必要となる．一方，ローカルサーバには，リモートサーバから送出される他クライアントで発生した入力データをローカルシミュレーションに反映させる処理ならびに，リモートサーバからのシミュレーション結果の一貫性制御データを用いてローカルシミュレーションの再初期化を行なう処理が必要である．

5.2.2 ドメイン I/F プロセス

ドメイン I/F プロセスはサーバドメインとクライアントドメインの中継プロセスであり，これを設けることで異なる MPI 通信グループに属するサーバとクライアントの通信を可能にしている．主な役割はクライアントを接続する際にリモートサーバへの接続の確立と，入出力データの送受信である．**図 9** に処理の流れを示す．ドメイン I/F プロセスは，各ローカルサーバごとに 1 つ起動され，対応するローカルサーバから送られてきた入力データをリモートサーバに中継する処理とその逆方向の中継であるリモートサーバプロセスからの可視化結果を受け取り対応するクライアントに送出する処理で構成される．

5.2.3 クオリティコントロールプロセス

クオリティコントロールプロセスは，一貫性制御/入力データ一貫性制御の実施とスロースタートアルゴリズム（SSA）対策が主な処理である．リモートサーバからの計算結果送信の間隔と比べて，一貫性制御の間隔は大きく（例えば 100 倍）設定し，入力データ一貫性制御の間隔は一貫性制御間隔と比べて小さく（例えば 10 分の 1）設定する．入力データ一貫性制御のための頻繁な通信が SSA 対策を兼ねている．しかし，入力データ一貫性制御間隔が大きい場合やサーバへ接続しているクライアントが 1 人の場合は SSA 対策が必要となるため，その場合のみ数バイトのデータを送出する機能を設けた．**図 10** に処理の流れを示す．クオリティコントロールプロセスは，ロカールサーバ毎に 2 つのプロセスが生成され，それぞれリモートサーバとロカールサーバに分けて配置される．一貫性制御に必要なデータをリモートサーバプロセスから受け取りローカルサーバへ配信するための処理が基本であるが，SSA 対策のためのメッセージの送出のみはリモートサーバプロセスからのデータ受信とは非同期にクオリティサーバ間でのみダミー入力として送受が行なわれる．

5.3 フレームワークが提供する新たな関数の機能

5.3.1 入力データー貫性制御

```
int SC_INTEGRITYCONTROL_IN
(void* buf, //受信するバッファの先頭ポインタ
 int   count, //受信するデータの要素数
 int   datasize, //1 要素のデータサイズ [Byte]
 int   step) //タイムステップ数
```

この関数は，ローカルサーバプロセスにおいて入力データ一貫性制御を行う関数である．第 1 引数には，受信する入力データを格納する領域のアドレスを指定し，第 4 引数のタイムステップ数を見て，入力データ一貫性制御が必要な場合には，クオリティコントロールプロセスから入力データを受信する．

5.3.2 リモートサーバ内通信

```
int SC_DATASETR
(void *message, //送信するデータの先頭ポインタ
 int  count, //送信するデータの要素数
 int  datasize, //1 要素のデータサイズ [Byte]
 int  step) //タイムステップ数
int SC_DATAGETR
(void *buffer, //受信するバッファの先頭ポインタ
 int  *count, //受信したデータの要素数
 int  datasize, //1 要素のデータサイズ [Byte]
 int  step) //タイムステップ数
```

これらの関数は，指定したデータサイズのデータを送受信することができる．本フレームワークではリモートサーバプロセスにおいて，シミュレーション結果データと各ク

表 1 実験環境

	リモートサーバ	ローカルサーバ兼操作端末
CPU	IntelXeon E7-8870 2.4GHz	IntelCorei7 2700 3.4GHz
コア数	10	4
メモリ	30GB	3GB
OS	Linux 32bit	Linux 32bit
MPI	MPICH2-1.1	MPICH2-1.1

ライアントの入力データをまとめたものを配信部に送ることに使用する．受信側の関数において要素数をポインタ渡ししており，ここには受信された要素数が格納される．これによりシミュレーション結果データと入力データのどちらを受信したのかを判断したり，シミュレーションステップ毎に一定のデータサイズの結果データではなく以前のシミュレーション結果データからの差分のみを送るような場合にも対処可能である．

5.3.3 サーバステップ共有

```
int SC_STEPSYNC
（int *step）//マスタタイムステップ数の読み書き先
```

この関数は，クオリティコントールプロセスから各クライアントプロセスへ自身がリモートサーバへ接続した際のリモートサーバのマスタタイムステップを配信する関数で，クライアント側の各プロセスのはじめに 1 度だけ利用する．一貫性制御実施タイミングの同期のため必要となる．

6. 評価

福井大学の我々の研究室にローカルサーバ兼操作端末，北海道大学情報基盤センターのプロジェクトサーバにリモートサーバを設置した．なお，学内ネットワーク環境の制約のため学外に商用の VPN サーバをレンタルし，このサーバを介して福井大学ー北海道大学間でシミュレーションキャッシングを実行する環境を構築した．このときの福井大学ー北海道大学間の通信帯域は約 7[Mbps] であり，通信遅延の平均は 30[msec] であった．**表 1** に実験環境を示す．アプリケーションへのフレームワークの適用試験，入力データ一貫性制御効果の確認，マルチクライアントの影響調査のため実験を行った．

6.1 熱拡散シミュレーションへの適用

フレームワークの機能である他アプリケーションへの適応性を確認するため，熱伝導方程式ベースの差分法による熱拡散シミュレーションアプリケーションに本フレームワークを適用し，シミュレーションキャッシングを用いて遠隔シミュレーションの実験を行った．このシミュレーションでは領域内に動く熱源を用意し，それに応じて領域内の温度分布がどのようになるかをシミュレーションする．

サーバ間連携法は図 4-(a)，一貫性制御は中断モデルと入力データ一貫性制御を使用した．入力送信プロセスから各サーバプロセスに送信される計算条件は領域内の熱源座標（XY 座標），結果提示プロセスで受信するデータは，領域内の温度分布を可視化したものとした．通信データサイズは入力座標が 16[Byte]，シミュレーション結果の可視化データが 48[KB]，一貫性制御データが 128[KB] である．シミュレーション領域は二次元で，リモートサーバでは 128x128 の解像度，ローカルサーバでは，領域全体をリモートサーバの 1/2 の解像度（64x64）でシミュレーションを実行した．ローカルサーバでの計算量は時間方向も含め 1/8 である．

ステアリング部・シミュレーション本体の修正・通信関数などの新たにユーザが記述したコードは 20 行ほど（ヘッダファイルの変更等は除く）であり，容易にフレームワークの適用ができることがわかる．

6.2 性能評価

入力データ一貫性制御の効果，マルチクライアントにおけるジッタ隠蔽効果を調査するため，6.1 章で本フレームワークを適用した熱拡散シミュレーションを用いて評価実験を行った．シミュレーション単体で動作させたときの 1 ステップに必要な時間はリモートサーバ上で約 1[msec] である．パイプラインピッチは，実験環境デバイスの表示 FPS の上限，人間が違和感なく可視可能な値として 30[fps]（33[msec]）に設定した．

6.2.1 入力データー貫性制御の効果

入力データ一貫性制御の効果を確認する．一貫性制御間隔を 100 ステップに設定し測定用クライアントをサーバへ接続する．300 ステップ経過後，2 台目のクライアントを接続し測定用クライアントのローカルサーバのシミュレーション結果とリモートサーバのシミュレーション結果の各要素の絶対誤差の総和を求める．入力データ一貫性制御の有無や実施間隔の違いでシミュレーション結果の誤差にどのような影響が表れるかを測定する．**表 2** に絶対誤差の最大値と平均値，**図 11** にタイムステップ毎の絶対誤差の値を示す．

表 2，図 11 を見ると，入力データ一貫性制御によって誤差の増加が抑えられていることがわかる．また，入力データ一貫性制御なしのローカルサーバシミュレーションの最終結果の可視化画像（**図 12**-(a)）と入力データ一貫性制御ありのローカルサーバシミュレーションの最終結果の可視化画像（図 12-(b)）をリモートサーバシミュレーションの最終結果の可視化画像（図 12-(c)）と比較すると視覚的にも入力データ一貫性制御が有用に働いていることがわかる．一貫性制御および入力データ一貫性制御間隔の設定はリアルタイム性とシミュレーション精度のトレードオフになっており，実行したいシミュレーションや実行環境に合わせて適切な値を設定する必要がある．

また，一貫性制御間隔を 300 ステップに設定した結果を**表 3** に示す．誤差を比較すると，入力データ一貫性制御

表 2 絶対誤差の最大値と平均値（一貫性制御間隔 100 ステップ）

入力データ一貫性制御間隔 [step]	なし	30	2
最大絶対誤差	18594	11456	4979
平均絶対誤差	3123	2096	1229

表 3 絶対誤差の最大値と平均値（一貫性制御間隔 300 ステップ）

入力データ一貫性制御間隔 [step]	なし	30	10	2
最大絶対誤差	40202	17882	10550	7824
平均絶対誤差	7268	3905	2517	2031

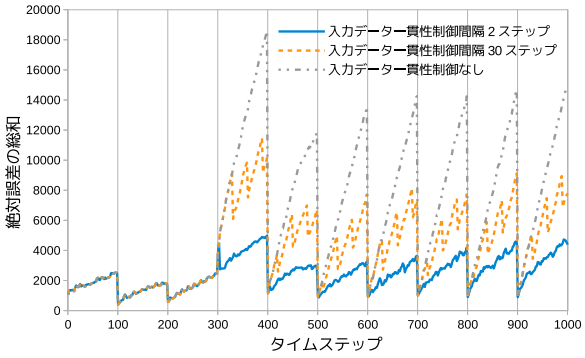
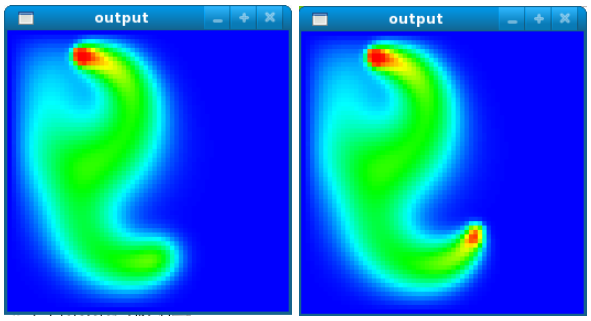
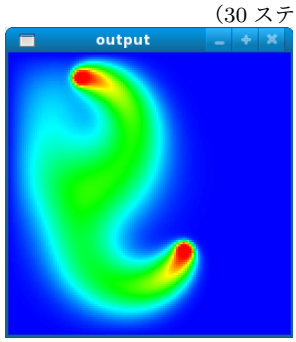


図 11 タイムステップ毎の絶対誤差（一貫性制御間隔 100 ステップ）



(a) 入力データ一貫性制御なし (b) 入力データ一貫性制御あり（30 ステップ間隔）



(c) リモートサーバ

図 12 シミュレーション最終結果の可視化画像

間隔を 10 ステップに設定したとき，表 2 の入力データ一貫性制御間隔 30 ステップのデータに近い結果を得られた．入力データ一貫性制御を導入することで，通信データ量の多い一貫性制御の間隔を伸ばすことも可能である．

6.2.2 マルチクライアントの影響

接続クライアント数を増加させた際の影響を調査する．一貫性制御間隔を 100 ステップに設定し，測定用クライアントをサーバに接続する．300 ステップ経過後に 2 台目のクライアントを接続し測定用クライアントの遅延補償プロセスにて許容遅延違反率と結果データ提示間隔を測定した．異なる 5 回の試行結果を**表 4** に 1 回目の試行の結果データ提示間隔を**図 13** に示す．比較対象として，シングルクライアント時の遅延違反率ならびに実行時間 [8] を**表 5** に，シミュレーションキャッシングなしで熱拡散シミュレーション単体を遠隔実行した場合の結果データ提示間隔を**図 14** に示す．

表 4 マルチクライアント接続時の許容遅延違反率と実行時間

試行 No.	1	2	3	4	5
許容遅延違反率 [%]	50.5	40.9	59.4	65.1	48.2
1000 ステップ実行時間 [sec]	38.2	40.6	39.6	40.6	39.2

表 5 シングルクライアント接続時の許容遅延違反率と実行時間
Table 5 Output Interval Violation Ratio and Execution Time.

試行 No.	1	2	3	4	5
許容遅延違反率 [%]	57.7	61.0	29.5	45.8	43.1
1000 ステップ実行時間 [sec]	35.5	35.8	36.2	36.5	36.0

表 4 と表 5 を比較すると，遅延違反率については試行毎に値が変動するため具体的な数値での比較はできないが，大まかな傾向を概観した限りでは特段の変化は見られない．一方で，1000 ステップ実行時間はクライアント数が増えたことで，中断モデルを用いた一貫性制御時の終了待ち時間の影響が加わり 10%程度増加する傾向が見られた．同時ステアリングを行なうクライアント数がさらに増えると実行時間も増加することが推定できるが，同時ステアリングのクライアント数を数クライアントに制限すれば実行時間の増加を一定範囲内に収めることが可能と考える．また，クライアント数を更に増やす必要がある場合には，計算精度とのトレードオフとして一貫性制御の間隔を大きくすることで，実行時間増加の影響を軽減することも可能である．

図 13 を見ると初期化と一貫性制御，入力データ一貫性制御のタイミングを除いたステップでは設定したパイプラインピッチ通り結果データを提示できている．また，2 台目クライアントが接続した直後の一貫性制御と入力データ一貫性制御において他のタイミングより大きな遅延が発生している．これは 2 台目のクライアントが接続する際のリモートサーバの負荷によるものであり，一貫性制御と入力データ一貫性制御が実施され足並みが揃うと安定した動作に戻る．

遅延時間の変動については，パイプラインピッチの設定値である 33ms 付近で小刻みに遅延時間が変動する図 14 に対して，図 13 では，一貫性制御時を除き結果データ提示間隔のゆらぎがなくジッタ隠蔽効果が発揮されていることが確認できる．

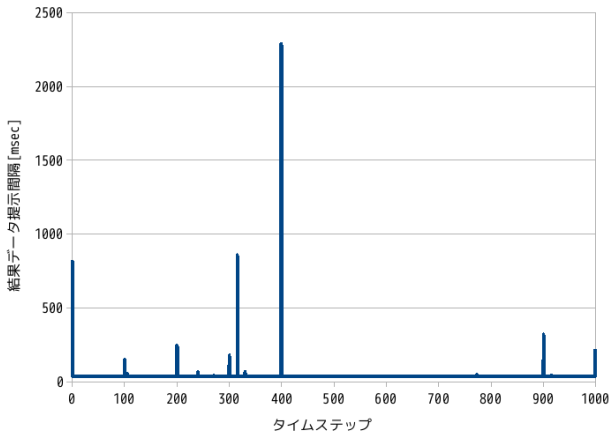


図 13 結果データ提示間隔 (違反率 50.5[%])

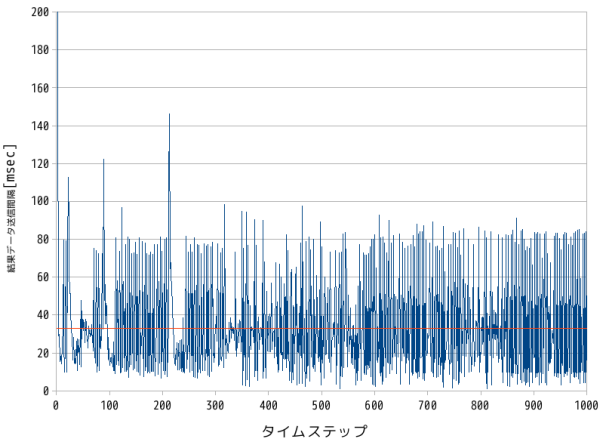


図 14 熱拡散シミュレーションの結果データ提示間隔
(シミュレーションキャッシングなし)

7. まとめ

本論文では，対話型遠隔シミュレーションシステムを複数人で同時利用することを目的とし，シミュレーションに対して複数のユーザが同時にステアリング可能にすると共に，シミュレーション結果を共有できるシステムのフレームワーク実装を行った．複数クライアントのローカルサーバ間の一貫性を補償するため，入力データ一貫性制御を実装した．その結果，シミュレーションキャッシングを複数人で実行し，同時にステアリングと可視化が可能なシステムの構築に成功した．従来の一貫性制御と合わせて運用することで，マルチクライアントにおいても各ローカルサーバ間の一貫性を補償することができ，複数のクライアントが接続した際も，クライアント接続直後に影響が見られたがシステム全体として安定した動作を確認できた．

しかし，本フレームワークを使用する際，一貫性制御や入力データ一貫性制御の間隔，許容遅延時間などを実行したいシミュレーションやサーバ間連携法に合わせてユーザが設定しなければならないため，今後は自動的にパラメータを生成する機能の追加を行いたい．さらに追加実験として，大規模シミュレーションに本フレームワークを適用した場合や，クライアント数を増加させた場合の実験を行う必要がある．

謝辞 本研究の一部は，JSPS 研究費 25280042 ならびに 25540043 の助成を得て実施した．本研究の一部は，北海道大学情報基盤センターのシステムを利用して実施した．また，数々の有力な御指導，御意見を頂いた松山幸雄技術職員をはじめ，日頃様々な面でお世話になった本学森・福間研究室の皆様に深く感謝致します．

参考文献

[1] McCarty, W.D., Sheasby, S., Amburn, P., Stytz, M.R. and Switzer, C.: A virtual cockpit for a distributed interactive simulation, IEEE Computer Graphics and Applications, Vol.14, No.1, pp.49-54(1994).
[2] Marshall, R., Kempf, J. and Dyer, S.: Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie, Proc. Symp. on Interactive 3D Graphics, pp.89-97(1990).
[3] Johnson, C., Parker, S.G., Hansen, C., Kindlmann, G.L. and Livnat, Y.: Interactive Simulation and Visualization, IEEE Computer, Vol.32, No.12, pp.59-65(1999).
[4] Zhu, Q. and Agrawal, G.: Supporting Fault-Tolerance for Time-Critical Events in Distributed Evironments, Proc. of Supercomputing, Paper Nr.32, pp.1-12(2009).
[5] Malakar, P., Natarajan, V. and Vddhiyar, S.S.: An Adaptive Framework for Simulation and Online Remote Visualization of Critical Climate Applications in Resource-constrained Environment, Proc. of Supercomputing, Paper Nr.295, pp.1-10(2010).
[6] Woodward, P.R., Porter, D.H., Greensky, J. Larson, A.J., Knox, M., Hanson, J., Ravindran, N. and Fuchs, T.: Interactive Volume Visualization of Fluid Flow Simulation Data, PARA'06 Workshop on the State-of-the-Art in Scientific and Parallel Computing(2006).
[7] 橋本健介, 手塚俊作, 森眞一郎, 富田眞治：シミュレーションキャッシングと遠隔インタラクティブ流体シミュレーションへの応用，情報処理学会論文誌:コンピューティングシステム，Vol.5，No.4，pp.76-86(2012).
[8] 山本 優, 西村祐介, 福間慎治, 森眞一郎： シミュレーションキャッシングフレームワークの実装, 情報処理学会論文誌, Vol.57, No.3, pp.823-835(2016).
[9] 西村祐介：シミュレーションキャッシングフレームワークの実装，福井大学大学院工学研究科，修士論文 (2012).
[10] 今村俊幸，小出洋，武宮博：異機種並列計算機間通信ライブラリ Stampi 利用手引書　第二版，日本原子力研究所 (2002).
[11] 棟朝雅晴： 分散クラウドシステムにおける遠隔連携技術，学際大規模情報基盤共同利用・共同研究拠点，平成 25 年度共同研究最終報告書 (2014)
[12] 實本英之，小林泰三，松本正晴，滝澤真一郎，三浦信一，中島研吾：複数拠点利用を実現するユーザ駆動型・拠点協調フレームワーク，信学技報，pp155-159(2014).
[13] 菅原章博，岸本泰明：大規模シミュレーションを中心に据えた遠隔研究システム II，J. Plasma Fusion Res., Vol.87, No.3，pp.222-229(2011).
[14] 武井利文，松本秀樹，土肥俊：リアルタイム可視化システム RVSLIB，第 10 回計算力学講演会講演論文集，pp.413(1997).

[15] 田村善昭，小笠 温滋，中島拓之，藤井孝藏：ライブラリレス・リアルタイム可視化システム，可視化情報学会誌，Vol.25，No.Suppl.1，pp.251-254(2005).
[16] Waser, J., Fuchs, R. Ribicic, H., Schindler, B., Bloschl, G. and Groller, M.E.: World lines, IEEE Transactions on Visualization and Computer Graphics, Vol.16, No.6(2010).
[17] Niebling, F., Becker, M., Kopecki, A., Stellba hydro GmbH & Co. KG, High Performance Computing Center Stuttgart(HLRS): Collaborative Steering and Post-Processing of Simulations on HPC Resource Everyone, Anytime, Anywhere, Web3D(2010).
[18] Jeong, B., Jagodic, R., Renambot, L., Singh, R., Johnson, A., Leigh, J.: Scalable Graphics Architecture for High-Resolution Displays, Proc. of IEEE Information Visualization Workshop (2005).
[19] Ahrens, J., Geveci, B., and Law, C. : ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier (2005).
[20] Schroeder, W., Martin, K., and Lorensen, B., The Visualization Toolkit (4th ed.), Kitware(2006).
[21] Nonaka, J., Ono, K., Bi, C. and Sakurai, D.: HIVE: A Visualization and Analysis Framework for Large-Scale Simulations on the K Computer, IEEE Pacific Visualization 2016, poster session handout, pp.10-11(2016).
[22] 山本優，西村祐介，福間慎治，森眞一郎：対話型遠隔シミュレーションフレームワークのマルチクライアント拡張と予備評価，情処研報，Vol.2014-HPC-147，No.15，pp.1-8(2014).