Vol.2014-ARC-213 No.15
Vol.2014-HPC-147 No.15
2014/12/9

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

山本　優[1]　西村　祐介[1]　福間　慎治[1]　森　眞一郎[1]

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. この様な対話的遠隔シミュレーションシステムでは, 通信遅延の隠蔽が課題となる. この問題に対し, シミュレーションキャッシング技術の研究を行ってきた. 本論文ではこれまでの研究を元に, シミュレーションキャッシング・フレームワークのマルチクライアント拡張を行い, 遠隔地の高性能サーバ上のシミュレーションに複数人での協調作業を可能にするフレームワークと弱い一貫性を保証するシステムを開発し, その予備評価を行った.

キーワード：インタラクティブシミュレーション, 遠隔シミュレーション, 分散協調処理, シミュレーションキャッシング, 一貫性制御, シミュレーション共有

1. はじめに

近年の計算機性能の急速な向上に伴い, インタラクティブ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一例である. 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]. しかしながら, 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, 遠隔地のスーパーコンピュータ上での科学技術計算に対して対話的な操作を可能とする実時間遠隔シミュレーションステアリング, ならびにネットワーク接続可能な場所であればどこからでも大規模シミュレーションを対話的にステアリング可能とするユビキタスなシミュレーション・インタフェースの開発を行っている.

本論文では, 遠隔地にある高性能サーバ上のシミュレーション結果を複数のユーザで共有ししつ, 対話的なシミュレーションステアリングを可能にする対話型遠隔シミュレーションフレームワークの実装を行い, その予備評価を行った結果を報告する.

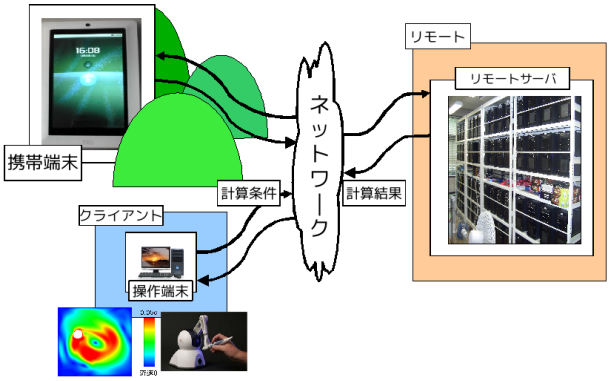

図 1　対話型遠隔シミュレーション

2. 研究背景

2.1 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 (図 1). 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

[1] 福井大学工学研究科

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というものである．この時，複数の操作端末からリモートサーバに新しい計算条件を自由なタイミングで転送するため，対象かつ双方向の対話性を各操作端末とリモートサーバは確保することが必要である．

2.2 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遠隔操作シミュレーションは，1) 遠隔地に存在する高性能シミュレーションサーバ (リモートサーバ) と，2) ユーザの操作や結果の提示を行う操作端末と，3) 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保証する．

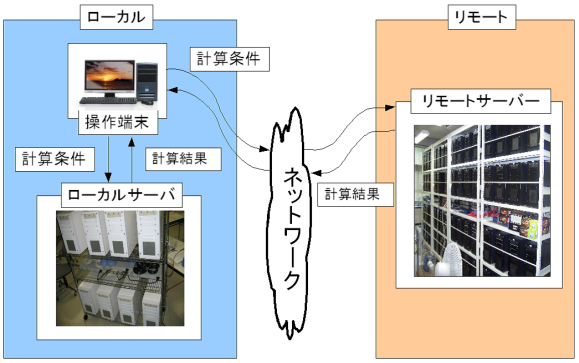

図 2 シミュレーションキャッシング

2.2.1 遅延隠蔽手法

シミュレーションにパイプライン処理を導入した上で，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することで，ユーザから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ローカルサーバが即応可能である (図 3)．２つのサーバの連携により予測不可能な通信遅延の変動による影響を隠蔽し (図 4) 実時間インタラクションを可能にする手法である．現在の高性能プロセッサに必要不可欠なキャッシュ・メモリが主記憶上のデータの一部をキャッシュしメモリアクセス遅延を隠蔽するのに対して，シミュレーションキャッシングではあたかも計算の一部をキャッシングしているかのように振る舞う．

2.2.2 リモート・ローカル間連携法

連携の方法としては，例えばシミュレーション結果の提示において出力されるデータ群の中に計算制度や応答時間の許容量に違いがある場合，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

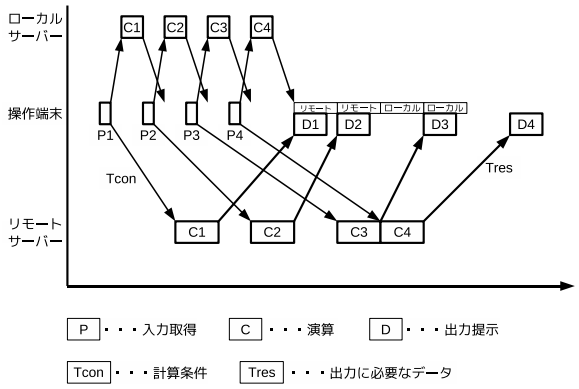

図 3 シミュレーションキャッシングの流れ

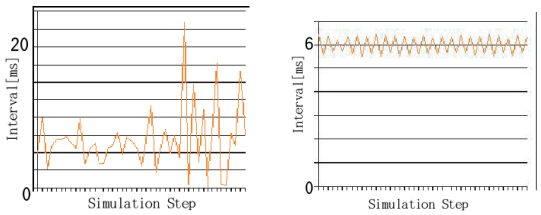

図 4 遅延変動隠蔽

を使用し，逆に応答時間に厳しい製薬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．

なお，ローカルサーバで行う「シミュレーションの一部」という概念として様々考えられる．以下に例を示す．

- 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 5)．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．場合によっては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をについてシミュレーションを行う (図 6)．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 7)．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．
- 異なる数値計算モデルを使用する．打ち切り誤差を大きくする等，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使用するハードウェア・ソフトウェアを加味した上で決定する必要がある．

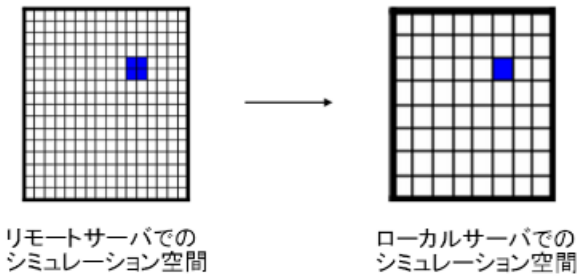

図 5 低解像度のシミュレーション

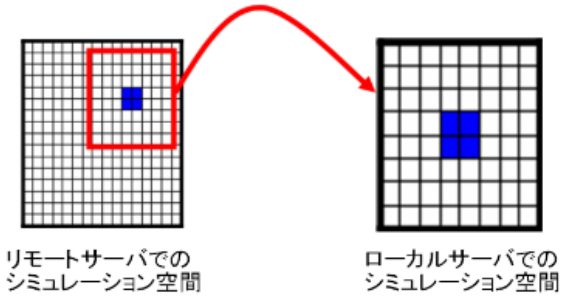

図 6 部分領域の抽出

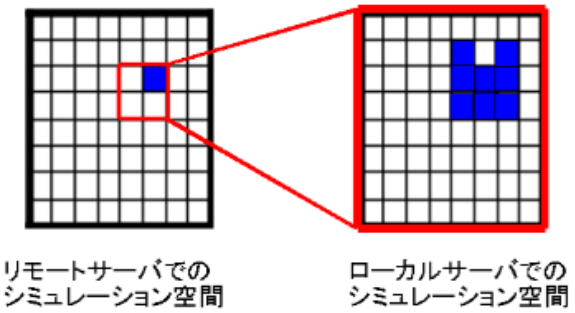

図 7 部分領域の高解像度抽出

2.2.3 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違いが大きくなるという問題が発生する. その為, シミュレーションがある程度進行した時点で各サーバの結果を一致させる処理が必要になる. これを一貫性制御と呼ぶ. 一貫性制御はシミュレーションによって実施の頻度, 制御方式などをカスタマイズする必要がある.

3. シミュレーションキャッシング・フレームワーク

我々は, シミュレーションキャッシングをユーザが容易にアプリケーションに適用可能にするフレームワークの実装 [8] を行った. これにより, ユーザは本来のシミュレーションを行う部分や入出力の部分の記述にのみ注力でき, シミュレーションキャッシングを使ったアプリケーションの開発効率を上げることができる. 必要となるネットワークプログラミングの知識も最低限で済み, 簡単にプログラミングを行うことができる. 想定するシステム構成は, 操作端末・大規模シミュレーションサーバ (リモートサーバ)・近傍の簡易シミュレーションサーバ (ローカルサーバ) の 3 つをネットワークで結んだものとする.

3.1 搭載機能・仕様

シミュレーションキャッシングを行うため, 以下のような複雑な MPI の記述やシミュレーションステップの管理, その他シミュレーションキャッシングの機能はフレームワークが関数として用意している.

- 操作端末と各サーバの計算条件・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
- 操作端末にて表示する計算結果の自動選択
- スロースタートアルゴリズム対策

スロースタートアルゴリズムとは, TCP/IP で輻輳制御に用いられるアルゴリズムで, 通信が確立した直後のネットワーク通信に対して帯域幅が制限されるというものである. 一貫性制御の時間間隔が大きくなると, 一貫性制御が行われる前にスロースタートアルゴリズムが適用され, 通信速度が低下してしまう. 特に一貫性制御のデータサイズが小さい場合問題となる.

現在の実装では, 対象とする言語は C 言語で, 各プロセス間の通信には MPI を使用する. ユーザがフレームワークの関数を用いて作成するプロセスが, 入力プロセス, 出力プロセス, ローカルシミュレーションプロセス, リモートシミュレーションプロセスであり, フレームワークが提供するプロセスとして中継プロセス, クオリティコントロールプロセスがある. 全体の構成図を図 8 に示す.

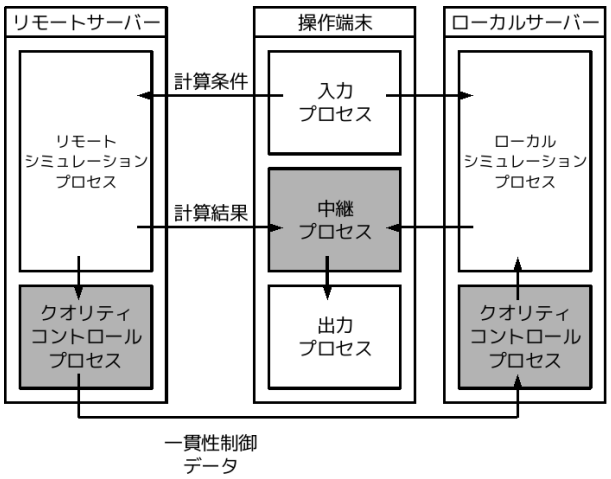

図 8 システム構成図

3.2 各プロセスの機能

3.2.1 入力プロセス

入力プロセスでは, 各シミュレーションサーバへ送信する計算条件の準備と各シミュレーションサーバへの一定間隔の送信が主な処理となる. フレームワークではこの内, 計算条件の一定間隔送信機能を関数として提供する. また, 中断モデルを用いた一貫性制御時に, 一貫性制御が終わるまで計算条件の送信を止める機能も関数として提供する. ユーザは自身の行いたいシミュレーションの計算条件をここへ記述する.

3.2.2 出力プロセス

出力プロセスでは, 各シミュレーションサーバから結果を受け取り, 出力をユーザに提示することが主な処理となる. フレームワークでは, 結果の取得機能を関数として提供する. この関数では, どちらのシミュレーションサーバからの結果を受け取ったのかという情報と受け取ったデータサイズを知ることができ, ユーザはこの情報を元にして異なる処理を行うことも可能である.

3.2.3 シミュレーションプロセス

リモート・ローカルの各シミュレーションプロセスでは, 計算条件の受信, シミュレーションの実行, 結果の送信, 一貫性制御が主な処理となる. 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. 特に一貫性制御に関しては, 使用するモデルに応じて複数の関数を提供し, ユーザは使用するモデルによって関数を選択することで一貫性制御を実現する.

3.2.4 中継プロセス

中継プロセスは, 各シミュレーションサーバからの計算結果の受信, 受信した結果のバッファリング, 一定間隔での出力プロセスへの計算結果送信がある. このプロセスが必要となる理由は, 各シミュレーションプロセスからの受信処理と出力に必要な処理を簡単に並列して実行可能になるためである. 使用するバッファの構造は, 送信元のシミュレーションサーバ識別番号 (RorL), シミュレーションステップおよびデータ格納先ポインタを持つ. データ格納領域サイズとバッファのエントリ数は, シミュレーションキャッシング用ヘッダファイルで変更できる.

3.2.5 クオリティコントロール (QC) プロセス

一貫性制御の際に一貫性制御データの送信, 通常時はスロースタートアルゴリズムによる通信遅延の増加を回避するため, 定期的に非常に小さなデータをやりとりをしている. ただし, 一貫性制御データが大きい場合や一貫性制御間隔が小さい場合, スロースタートアルゴリズムの影響は比較的に小さいものと思われるため, このプロセスを使用するか, 使用せず直接リモートシミュレーションプロセスとローカルシミュレーションプロセスで通信するかをユーザが簡単に変更できる仕様になっている.

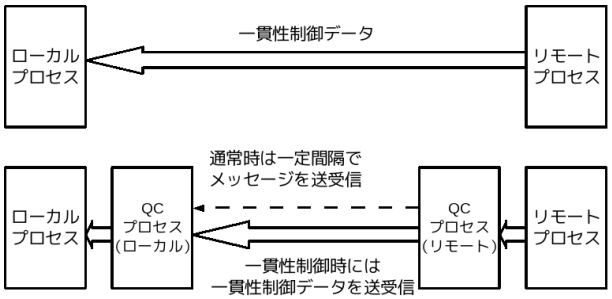

図 9 QC プロセスの処理

3.3 一貫性制御

シミュレーションキャッシング・フレームワークでは中断モデルとロールバックモデルの 2 つを用意している. 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, 一貫性制御が終わってからリスタートする (図 10). 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 (図 11). 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. よって一貫性制御モデルの決定はシミュレーション内容に大きく依存する.

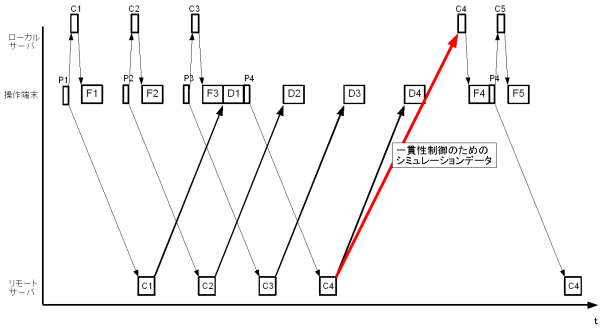

図 10 中断モデル

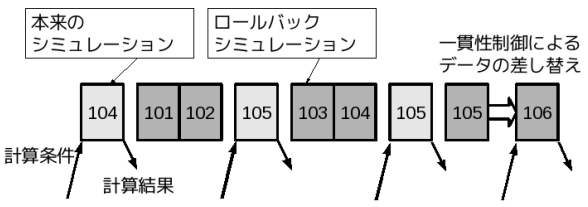

図 11 ロールバックモデル

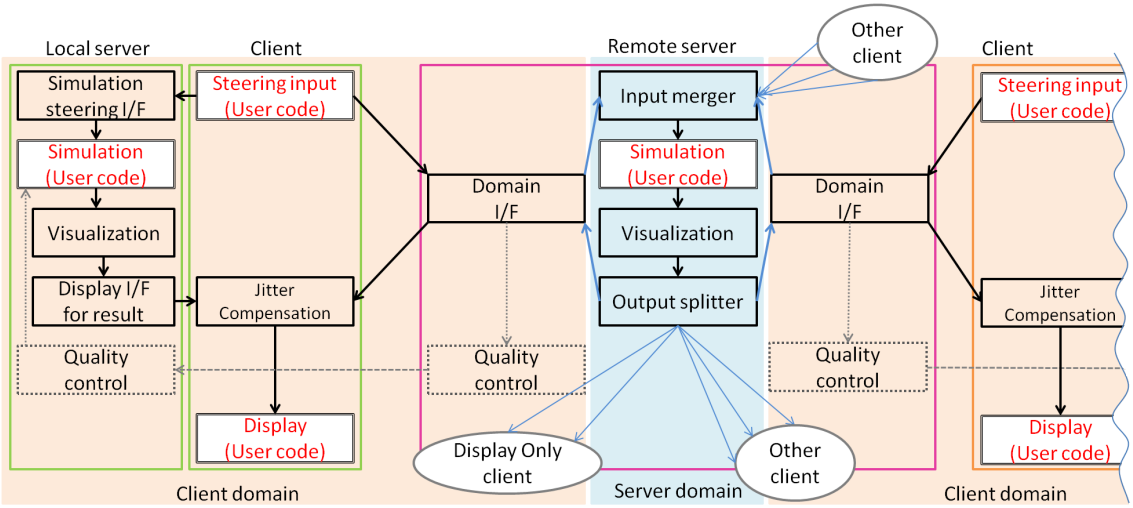

図 13 マルチクライアント拡張時の各プロセスの関係

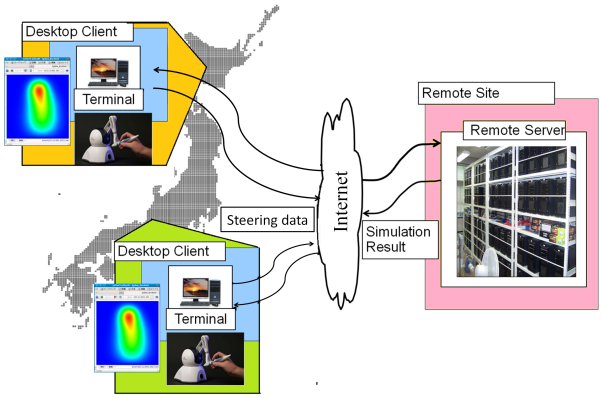

図 12 マルチクライアント拡張

4. 対話型遠隔シミュレーションシステムのマルチクライアント拡張

4.1 実装方針

大規模シミュレーションにて遅延環境下においても一定間隔でのデータ出力を保証し, 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共有ができるシステム (図 12) の構築を行う.

ここで, マルチクライアント拡張を行う上でマルチクライアント対応可能なサーバ間連携モデルがいくつか挙げられる. 一人のユーザがステアリングを行い複数のクライアントがモニタリングをする SOMM(Single Operator Multi monitor) モデルと複数のユーザが交互にステアリングを行う MOMM(multi monitor multi operator) モデル, 複数のユーザが相互干渉が十分に少ない状況で同時にステアリングを行う MOMM モデル, 複数のユーザが同時にステアリングを行う MOMM モデルなどである. 本来, MOMM モデルでは各クライアントのローカルサーバ間の一貫性制御が必要になる. しかし今回の実装では, ローカルサーバ間の厳密な一貫性制御は行っていない. そのためシステムの利用形態として「クライアントで表示する結果は自身の抽出領域のみ」という制約を与える. この条件下では, SOMM モデル, 交互 MOMM モデル, 疎干渉 MOMM モデルに対応できる. 各クライアントで表示する結果には, 他クライアントの入力による影響がすぐには反映されず一貫性制御を行うことでそれまでの入力情報がローカルサーバ上のシミュレーションに反映される.

4.1.1 機能要件・仕様

マルチクライアント拡張を実現するためには, 以下の機能を追加しなければならない.

- 不特定多数のクライアント間でのシミュレーションの遠隔共有
- 任意のタイミングでのクライアントの参加および退去
- 特定クライアントの故障, 通信回線切断の影響が全システムに波及しない枠組

これまでのシミュレーションキャッシング・フレームワークでは, シミュレーションや可視化を含む全てのプロセスが 1 つの MPI 通信グループに属していた. しかしながら, 今回のマルチクライアント拡張においては, シミュレーションプロセスや可視化プロセス等のリモートサーバ上のプロセスは複数のクライアントで共有されるプロセスであり, 特定クライアントの MPI 通信グループに属する構成をとることは妥当ではない. そこで, これらの共有プロセスを特定の MPI 通信グループから独立させたサーバドメインを設け, 各クライアントが属す MPI 通信グループにはサーバドメインとのインタフェースを担うプロセスを新たに設けることで, 複数クライアントがサーバドメインのプロセス群を共有する枠組みを構築する. この時, リモートサーバ上で稼働するクライアント毎のドメインインタフェースとサーバドメインのプロセス間通信は 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できる構成とする. これにより,

任意のタイミングでクライアントがリモートサーバ上のシミュレーションに参入および撤去が可能になるとともに, シミュレーションに参入していたクライアントの通信障害等を含む故障や通信遅延の影響がサーバプロセス群や他のクライアントに波及することを防ぐことが可能である. 図 13 にクライアントドメイン, サーバドメインのプロセス群の構成を示す.

4.2 マルチクライアント拡張機能の実装

マルチクライアント拡張する上で, シミュレーションキャッシング・フレームワークにプロセスの追加, 各プロセスの機能拡張・分割・リネームを行った. ローカルサーバ内の Simulation Steering I/F・Simulation・Visualization, リモートサーバ内の Simulation・Visualization の各プロセスは図 8 のシミュレーションプロセスの内容を分割して表したもので, 機能に大きな変化はない. Jitter Compensation(JC) プロセスは同図の中継プロセスをリネームしたもので, 機能に変化はない. 以下は新たにフレームワークに追加したプロセスである.

4.2.1 Domain I/F

サーバドメインとクライアントドメインの中継をするプロセスであり, 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やクライアントと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. QC プロセスに送る一貫性制御データやリモートサーバの進行ステップもここを経由してクライアントドメインへ送られる.

4.2.2 Input Merger

Domain I/F より中継された入力データを一括して受信するプロセスである. 複数のクライアントより同時に入力がある場合でも対応可能であり, 各クライアントの参加/退去の管理も行っている. 今回の実装では, シミュレーションをステアリングするクライアントには上限を設けた.

4.2.3 Output Splitter

リモートサーバでのシミュレーション結果を各クライアントに配信するプロセスである. 結果データと同時にシミュレーションの進行ステップ数を送信しており, JC プロセスでのデータ選択に活用している. また, このプロセスに接続することで結果データのみ受信することが可能であり, 見るだけといったクライアントにも対応できる.

4.3 一貫性制御

今回の実装では, 中断モデルを用いたローカルサーバとリモートサーバ間の一貫性制御によるシステム全体での緩い一貫性を保証している. クライアントがサーバに接続した時と, リモートサーバのシミュレーションステップが N ステップ経過するごとに各クライアントは中断モデルで一貫性制御を行っている.

マルチクライアント拡張における一貫性制御の効果を熱拡散シミュレーションを例に挙げて説明する. このアプリケーションは, 熱伝導方程式ベースの差分法により熱の拡散をシミュレーションするものである. 入力は 2 次元領域内の熱源の位置 (XY 座標), 出力はシミュレーション領域の温度を RGB に変換し OpenGL を用いて表示する. ローカルサーバでのシミュレーション解像度はリモートサーバの 1/2 で行っており, 計算量は 1/4 である. 具体的には, シミュレーション解像度をリモートにおいて 128x128, ローカルでは 64x64 としている. クライアントを 2 台接続した際の各サーバのシミュレーション内容を可視化したものと各クライアントで表示される結果を図 14,15 に示す. この場合, シミュレーションは図 5 のモデル, サーバ間連携は疎干渉 MOMM モデルにあたり, 各クライアントは決められた領域のみ入力と出力が可能である. また, 図 14 は一貫性制御直前, 図 15 は一貫性制御直後である. 図 14 のクライアント A では, リモートのシミュレーション結果データが届かずローカルサーバ A のデータを表示しており, クライアント B の入力による影響が表示されていない. しかし, 図 15 では一貫性制御の効果によりローカル A にもクライアント B の入力による影響が反映されておりリモートと差異のない結果が表示されている. しかし, 今回の実装では相手側の入力による熱源は, 時間が経過するとなくなり再びリモートサーバとの誤差が大きくなっていく. 定期的に一貫性制御を実施することが必要となる.

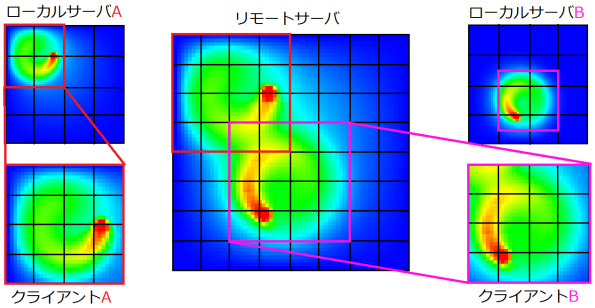

図 14 各サーバでのシミュレーション

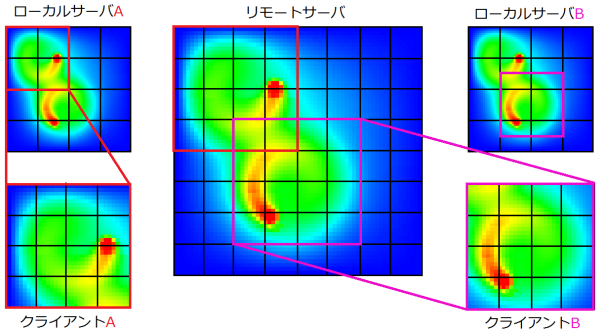

図 15 一貫性制御実施直後

5. 予備評価

VPN を用いて福井-東京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, 実験を行った. リモートサーバを東京に設置し, クライアント 2 台を共に福井に設置した. この時, ローカルサーバは各クライアントと同一端末とした. 実験環境を表 1 に示す.

表 1 実験環境

	リモートサーバ	測定用クライアント	マルチ用クライアント
CPU	Intel Core2Quad Q8400	Intel Corei7 940	Intel Core2Duo E6700
OS	Linux		
GPU	-	GeForce GTX280	GeForce GTX470
MPI	MPICH2-1.1		

5.1 遅延変動の影響

はじめに, 福井-東京間でシミュレーションキャッシングを行った場合の遅延変動の影響を調査するため, JC プロセスにおけるローカルサーバからの結果データの選択率と結果データ送信間隔を測定した. 扱ったシミュレーションは 4.3 章と同様, 最大ステップ数は 1000, 一貫性制御間隔は 100 ステップ, パイプラインピッチは 30[ms] に設定した. 異なる 6 回の試行において遅延変動の影響でローカルシミュレーションの結果が表示される割合とプログラム全体の実行時間を表 2 に, 1 回目の試行での結果データ受信間隔の時間変化と過去 10 ステップのローカル比を図 16 に示す.

表 2 より, 同じシミュレーション環境でも遅延変動の影響を受けローカル比は 0%から 50%までの幅を持つ. 図 16 からは, 設定したパイプラインピッチ通り動作していること, 遅延変動の隠蔽効果が確認できた. また, ローカルサーバの結果が選択された 12.7%の分布が固まって存在していることがわかる. これは他の測定結果でも同様であり, 遅延変動に時間的局所性のようなものがあると予想される.

表 2 結果データ選択率と実行時間

試行 No.	1	2	3	4	5	6
Local 比 [%]	12.7	0.00	48.3	10.4	36.1	18.2
実行時間 [sec]	30.36	30.19	30.53	30.19	30.31	30.21

5.2 一貫性制御間隔の影響

次に, 一貫性制御を実施する間隔を変更した場合のシミュレーション全体への影響を調査するため, JC プロセスにおける, ローカル比や実行時間, 結果データの送信間隔を測定した. シミュレーション環境等は一貫性制御間隔を除き, 5.1 章と同様である. 結果データの選択率と実行時間を表 3, 結果データの送信間隔の時間変化の例を図 17,18 に示す. ただし, 本実験でのローカル比は一貫性制御後のリモートサーバの結果データ選択率を除いて計算している. これは, 一貫性制御の直後のステップでは必ずリモートサーバの結果を表示するという制約を課しているためである.

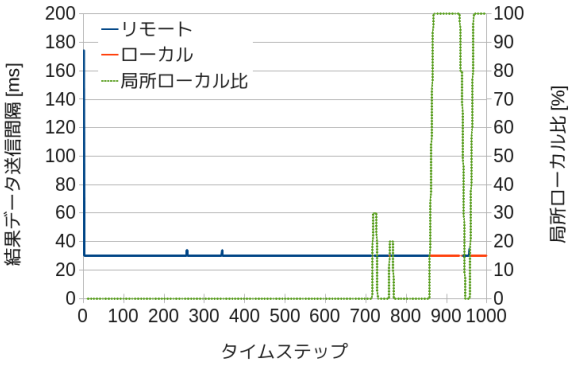

図 16 結果データ送信間隔 (試行 No.1)

図??から, 定期的にパイプラインピッチを越える遅延が発生していることがわかる. これは一貫性制御によるものであり, 遅延の発生周期は 一貫性制御間隔と等しい. 一貫性制御のタイミングで遅延ができない箇所もあり, 一貫性制御を実施した際に遅延が発生するか否か, 発生した遅延の大小等は通信路の遅延変動の影響を受ける. また今回の実験環境では, 一貫性制御間隔が 10 ステップ以下になると, 全体の実行時間や結果データの表示間隔に大きな影響を与え, 100 ステップ以下の場合, 表示結果の大半がローカルサーバの結果データになっている. これは一貫性制御間隔が短くなることで, インプットやシミュレーションプロセスが頻繁に止まり, パイプライン化の結果が十分に発揮されないためだと思われる. 逆に, 一貫性制御間隔が大きすぎるとローカルサーバとリモートサーバの差異が大きくなり, 一貫性制御後しばらく経過した時にローカルサーバの結果が表示された場合, 違和感を感じる.

表 3 一貫性制御間隔を変更した場合の結果データ選択率と実行時間

一貫性制御間隔 [step]	2	10	25	50	100	200
Local 比 [%]	99.8	96.1	86.2	74.5	12.8	0.00
実行時間 [sec]	63.53	35.89	32.47	31.41	30.17	30.13

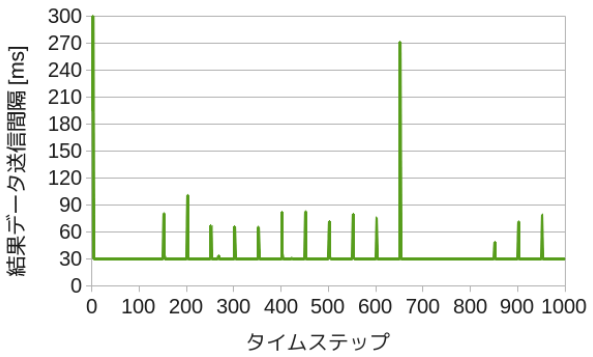

図 17 一貫性制御間隔が 50 ステップの結果データ送信間隔

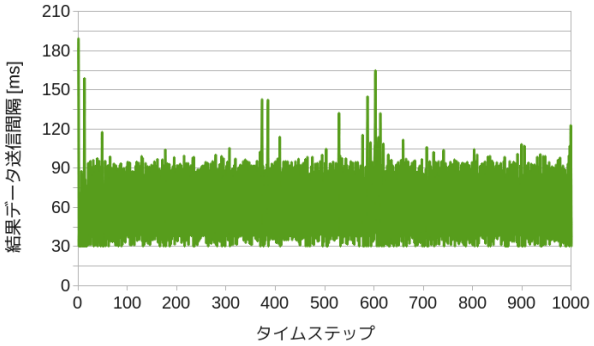

図 18 一貫性制御間隔が 2 ステップの結果データ送信間隔

5.3 接続クライアント数の影響

最後に, 接続クライアント数を増加させた場合のクライアントへの影響を調査するため, 測定用クライアントが接続し 500 ステップ程度経過した後, マルチ用クライアントを接続する. このとき測定用クライアントの JC プロセスにおけるローカル比や, 結果データの送信間隔を測定した. シミュレーション環境等は, 5.1 章と同様である. 異なる 5 回の試行において結果データの選択率と実行時間を表 4, 1 回目の試行での結果データ送信間隔時間変化を図 19 に示す.

図 19 からマルチクライアント拡張による遅延変動の隠蔽効果に影響は感じられない. しかし, 2 台目のクライアントが接続した 500 ステップ以降に局所ローカル比が増加していることがわかる. この現象は他の試行でも同様に確認された. これはクライアントが接続する際のサーバの負担によるものであり, 一貫性制御が実施されサーバと各クライアントの足並みが揃うと安定した動作に戻る. また, 表 4 の実行時間を表 2 と比較すると, わずかだが増加している. 今後, 更に接続クライアント数が増加した際に問題になる可能性がある.

表 4 測定クライアントでの結果データ選択率と実行時間

試行 No.	1	2	3	4	5
Local 比 [%]	4.7	19.9	11.5	39.1	19.8
実行時間 [sec]	31.43	32.35	32.38	33.79	32.65

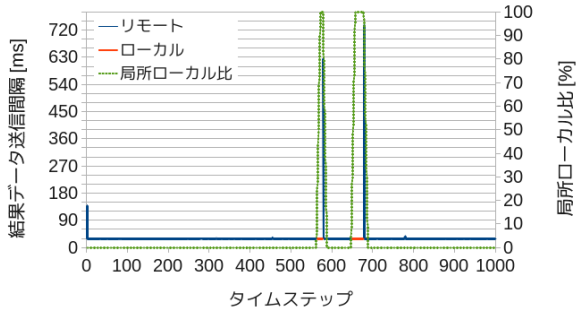

図 19 測定クライアントでの結果データ送信間隔 (試行 No.1)

6. まとめ

本稿では, 我々が開発した遠隔地にある大規模シミュレーションサーバを用いたインタラクティブシミュレーションに対する通信遅延隠蔽手法である, シミュレーションキャッシングのマルチクライアント拡張を行い, 予備評価を行った. その結果として限られたモデル上ではあるが, マルチクライアント環境でシミュレーションキャッシングを実行できるシステムを構築することに成功した. 今後は, 入力データの毎時共有法等の各ローカルサーバ間の一貫性制御手法について研究を行っていきたい.

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ました. 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に心から感謝致します. 上記の皆様方をはじめ, 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] W.D. McCarty, S. Sheasby, P. Amburn, M.R. Stytz, C. Switzer, “A virtual cockpit for a distributed interactive simulation,”IEEE Computer Graphics and Applications, Vol. 14, Issue 1, pp.49–54, 1994.

[2] R. Marshall, J. Kempf, S. Dyer, ”Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie,” Proc. Symp. on Interactive 3D Graphics, pp.89–97, 1990.

[3] C.Johnson, S.G. Parker, C. Hansen, G.L. Kindlmann, Y. Livnat, “Interactive Simulation and Visualization,” IEEE Computer, Vol. 32, No.12, pp.59-65, 1999. Dec. 1999.

[4] Q. Zhu, G. Agrawal, ”Supporting Fault-Tolerance for Time-Critical Events in Distributed Evironments”, Proc. of Supercomputing, Paper Nr. 32, pp.1-12, 2009.

[5] P. Malakar, V. Natarajan, S.S. Vddhiyar, ”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] P.R. Woodward, D.H. Porter, J. Greensky, A.J. Larson,M. Knox, J. Hanson, N. Ravindran, and T. Fuchs, “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 眞次:”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,IPSJ Symposuim Series(SACSIS ’09),Vol.2009,No5,pp.229-238, 2009.

[8] 西村 祐介, 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”, 福井大学工学研究科情報・メディア工学専攻修士論文, 2012.

[9] 山本 優, 荒川 文貴, 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システムのマルチクライアント拡張に関する検討”, 情処研報ハイパフォーマンスコンピューティング研究会, 2014-HPC-143(27),1-7, 2014