修 士 論 文

Jini技術とUNICOREを用いたリアルタイム可視化システムの構築

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

浅野喜宣

2005年3月

修 士 論 文

Jini技術とUNICOREを用いたリアルタイム可視化システムの構築

指導教官　松澤 照男 教授

審査委員主査　松澤 照男　教授
審査委員　井口 寧　助教授
審査委員　堀口 進　教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

310003 浅野喜宣

提出年月: 2005 年 2 月

概 要

科学技術の分野では，計算機の発展とともにシミュレーションの研究が盛んに行われている．数値流体力学 (Computational Fluid Dynamics CFD) の分野においても，数値シミュレーションについて注目され，流体現象を詳細にシミュレーションする要求が高まっている．しかし，要求される問題の大規模化やシミュレーションの高精度化に計算機の性能が追いついていない．

そこで，グリッドコンピューティングによる，分散環境の開発が行われている．グリッドコンピューティングとは，地理的に分散した計算資源を対象とし，それらをネットワークで動的につなぎ合わせ，物理的な距離や計算機の構成の差異を意識せずにひとつの仮想環境として利用可能にするための技術である．グリッドコンピューティングの標準化を目指し，The Global Grid Forum によって研究が行われている．一方，実現化を目指し，Globus Toolkit[†]や UNICORE[‡]等のミドルウェアが開発されている．また，ユビキタスコンピューティングの概念から PC やワークステーションなどの計算機をネットワーク上で連携させるための技術として Jini 技術[§]がある．Jini 技術は，ハードウェアやソフトウェアの実装に関係なく，ネットワーク上でのサービス実現や，そのサービスを利用するプログラム間の自発的対話のためのシンプルなインフラを提供する技術であり，JavaSpaces というオブジェクト共有空間を用いてネットワーク上に分散された計算機間のデータのやりとりを行う．Jini 技術により，任意に計算機の追加や削除が行える．

一方，計算機の高速化，数値シミュレーションの大規模化とともに，計算結果の可視化技術の重要性が増している．従来の可視化技術は，すべての計算が終了した後，結果を計算機上に保存し，利用者端末に転送してから可視化を行っていた．そのため，全ての計算が終了する前に，計算処理の途中結果を確認することが難しく，計算の進行状況を追尾したり，シミュレーションの実行自体を途中で制御することが困難であった．そのため，シミュレーションの計算結果をリアルタイムに可視化するシステムの必要性が高まっている．

本研究では，数値流体力学の分野に着目し,「計算」と「結果の可視化」の問題点である計算時間の短縮とリアルタイム可視化とを実現させるため，Jini 技術と UNICORE とを用いたリアルタイム可視化システムを構築する．構築方法は，学内の LAN に分散した WS や PC に Jini 技術と UNICORE をインストールし，UNICORE から Jini 技術の起動操作を行えるようにした．リアルタイム可視化部分は，Jini 技術上に実装され計算結果が JavaSpaces にある場合，直ちに可視化を行うようにした．本システムにおける計算から可視化までの処理の流れは，(1)UNICORE から計算ジョブの投入する.（２）投入されたジョブは JavaSpaces を介して各計算機に割り当て，計算実行される．(3) 各計算機の計算結果は JavaSpaces に送られる．(4) 送られた計算結果を直ちに可視化する．となる．

[†]http://www.globus.org
[‡]http://www.unicore.org
[§]http://www.sun.com/software/jini

また，構築したシステムで，流体計算の基本となる連立方程式のベクトル行列計算と流体計算である一次元の Burgers 方程式と流体計算である二次元の角柱周りの流れ解析とを実行し，計算時間の短縮，リアルタイム可視化の実現を検証した．本システムにより，計算時間が 7 倍に短縮できることを確認した．また，リアルタイム可視化が行え，全ての計算が終了する前に計算処理の途中結果を可視化表示させることができた．さらに，可視化結果より，計算終了を待たずに，計算を一時停止，強制終了させることも可能になった．

よって，本システムを用いることで，数値流体力学における，数値計算からリアルタイム可視化までの一連の処理は，計算時間が短縮され簡単に計算結果のリアルタイム可視化が実行でき利便性が向上し作業効率が向上した．

目 次

第 1 章　序論 **1**
1.1 背景 1
1.1.1 グリッドコンピューティング 1
1.1.2 従来のリアルタイム可視化システム 2
1.2 目的 2
1.3 本論文の構成 2

第 2 章　Jini 技術 **3**
2.1 Jini 技術 3
2.1.1 概要 3
2.1.2 Jini 基本サービス 3
2.2 JavaSpaces サービス 4
2.2.1 概要 4
2.2.2 エントリ 4
2.2.3 プロセス 4
2.3 Jini 技術の起動 5

第 3 章　UNICORE **6**
3.1 概要 6
3.2 UNICORE の仕組み 7
3.3 ジョブ実行の概要 7

第 4 章　リアルタイム可視化システム **9**
4.1 リアルタイム可視化 9
4.2 リアルタイム可視化システム構成 9
4.3 システムの起動 10
4.4 システムの認証 11
4.5 計算および可視化の実行 12
4.6 可視化の流れ 15
4.7 可視化画面 15

第 5 章　リアルタイム可視化システムの検証と評価 **16**
5.1　システムの速度向上比 16
5.1.1　同性能の計算機による性能評価 16
5.1.2　異なる性能の計算機による性能評価 19
5.2　一次元問題 (流体計算) 20
5.2.1　領域分割 21
5.2.2　計算結果 21
5.2.3　考察 26
5.3　二次元問題 (流体計算) 27
5.3.1　計算アルゴリズム 27
5.3.2　計算格子 30
5.3.3　計算結果 30
5.3.4　考察 35

第 6 章　結言 **36**
6.1　まとめ 36
6.2　今後の課題 36

謝辞 **37**

図 目 次

2.1 Lookup サービスの概要 4

3.1 UNICORE Client 6
3.2 UNICORE System 7
3.3 ジョブ実行の概要 8

4.1 リアルタイム可視化システムの構成 9
4.2 UNICORE ジョブの作成 10
4.3 システムの認証 11
4.4 実行過程 (1) 12
4.5 実行過程 (2) 13
4.6 実行過程 (3) 13
4.7 実行過程 (4) 14
4.8 実行過程 (5) 14
4.9 可視化の流れ 15

5.1 行列ベクトル積 17
5.2 行列の加算 18
5.3 行列の積 18
5.4 領域分割 21
5.5 計算結果 ($t = 1$) 22
5.6 計算結果 ($t = 1$) 22
5.7 計算結果 ($t = 1$) 23
5.8 計算結果 ($t = 1$) 23
5.9 計算結果 ($t = 100$) 24
5.10 計算結果 ($t = 100$) 24
5.11 計算結果 ($t = 100$) 25
5.12 計算結果 ($t = 100$) 25
5.13 セル番号と境界 27
5.14 スタガード格子 28
5.15 速度の計算に対するセルの検査面 29
5.16 計算格子 30

5.17 計算結果 ($t = 1$) 31
5.18 計算結果 ($t = 100$) 31
5.19 計算結果 ($t = 200$) 32
5.20 計算結果 ($t = 300$) 32
5.21 計算結果 ($t = 400$) 33
5.22 計算結果 ($t = 500$) 33
5.23 計算結果 ($t = 600$) 34
5.24 計算結果 ($t = 700$) 34

表 目 次

4.1 計算機の構成 9

5.1 計算機の性能 16
5.2 各計算機の処理能力 19
5.3 行列ベクトル積 (計算結果) 19

第1章　序論

1.1　背景

科学技術の分野では，計算機の発展とともにシミュレーションの研究が盛んに行われている．数値流体力学 (Computational Fluid Dynamics CFD) の分野においても，数値シミュレーションについて注目され，流体現象を詳細にシミュレーションする要求が高まっている．しかし，要求される問題の大規模化やシミュレーションの高精度化に計算機の性能が追いついていない．

そこで，グリッドコンピューティングによる，分散環境の開発が行われている．また，PC や WS などの計算機をネットワーク上で連携させるための技術として Jini 技術 [1] があり，Jini 技術を用いた計算環境構築が行われている [2]．Jini 技術は，ハードウェアやソフトウェアの実装に関係なく，ネットワーク上でのサービス実現や，そのサービスを利用するプログラム間の自発的対話のためのシンプルなインフラを提供する技術であり，JavaSpaces というオブジェクト共有空間を用いてネットワーク上に分散された計算機間のデータのやりとりを行う．Jini 技術により，任意に計算機の追加や削除が行える．

一方，計算機の高速化，数値シミュレーションの大規模化とともに，計算結果の可視化技術の重要性が増している．従来の可視化技術は，すべての計算が終了した後，結果を計算機上に保存し，利用者端末に転送してから可視化を行っていた．その結果，全ての計算が終了する前に，計算処理の途中結果を確認することができないため，計算の進行状況を追尾したり，シミュレーションの実行自体を途中で制御することが困難であった．そのため，シミュレーションの計算結果をリアルタイムに可視化するシステムが必要となる．

1.1.1　グリッドコンピューティング

グリッドコンピューティングは，地理的に分散した計算資源を対象とし，それらをネットワークで動的につなぎ合わせ、物理的な距離や計算機の構成の差異を意識せずにひとつの仮想環境として利用可能にするための技術である．グリッドの概念と語源は，電力供給網 (Power Grid) から由来している．電力がコンセントにプラグを挿すだけで必要な電力が得られるのと同様に，利用者端末をネットワークに接続するだけで必要な計算資源を利用できる．

グリッドコンピューティングの標準化を目指し，The Global Grid Forum によって研究が行われている．一方，実現化を目指し，Globus Toolkit[3] や UNICORE[4] 等のミドル

ウェアが開発されている．

1.1.2 従来のリアルタイム可視化システム

リアルタイム可視化システムとして，すでにいくつかの提案がなされている．例として，MIT で開発されている pV3[5] がある．このシステムでは，計算機上でグラフィカルオブジェクトを並列処理で生成し，クライアント側へ送信する．クライアント側では，OpenGL などの汎用グラフィックライブラリを用いてレンダリング処理を行う．専用のグラフィクスボードを用いることで，高速なレンダリング処理を行うことができるため，リアルタイムに可視化することが可能である．これらのシステムでは，ネットワーク上で転送し利用者端末で扱うべきデータの大きさが，シミュレーション規模とともに大きくなる欠点がある．

一方，大規模蓄積データ向けのポストプロセッサとして開発が進められているものに，ASCI(Accelerated Strategic Computing Initative) の Terascale Visualization[6] がある．このシステムは，大規模データをいかに削除してクライアント側に転送し可視化するかを重点に置いている．また，領域全体の可視化を粗い精度で行った後，ズームアップしてローカル領域を高精度で可視化することが可能である．しかし，解析結果全体を保存しなければならないため，多くの時刻ステップや様々なパラメータセットのシミュレーション結果の可視化を行う場合は効率的ではない．

1.2 目的

本研究では，数値流体力学の分野に着目し,「計算」と「結果の可視化」の問題点である計算時間の短縮とリアルタイム可視化とを実現させるため，Jini 技術と UNICORE とを用いたリアルタイム可視化システムを構築する．また，構築したシステムで，流体計算の基本となる連立方程式のベクトル行列計算と流体計算である一次元の Burgers 方程式と流体計算である二次元の角柱周りの流れ解析とを実行し，計算時間の短縮，リアルタイム可視化の実現を検証し，本システムの評価を行う．

1.3 本論文の構成

本論文では，2 章で Jini 技術を用いた分散並列計算環境について説明をし，性能評価を行う．3 章では，本研究で実装したリアルタイム可視化システムについて説明を行い，4 章でシュミレーション結果について述べて，考察を行う．5 章でまとめと今後の課題について述べる．

第2章　Jini技術

2.1　Jini技術

2.1.1　概要

Jini技術は，1999年にSun Microsystemsによって発表された．Java言語を基にした分散オブジェクト技術であり，ネットワーク上に分散されたあらゆる資源を連携させるための技術である．Jini技術では，ネットワークを経由してサービスを提供する．

Jini技術で最も重要な概念は「サービス」であり，Jini技術の中で機能を提供するものはすべて「サービス」と呼ばれる．Jini技術は，Java仮想マシンやアプリケーションをネットワークに拡張するものであり，Javaの持つ様々な特徴を持っており，ネットワーク上のどこからでも機能を呼び出すことが可能である．

Jini技術は，不安定なネットワーク上でも安定して動作するように設計されており，動的なネットワーク環境を構築することができる．

Jini技術は，以下のような機能が挙げられる．

- 利用者は，時間や場所に関係なく独自のネットワークを形成することが可能である．
- 変化に素早く適応し，システムの対故障性と冗長性に優れている．

2.1.2　Jini基本サービス

Jiniを使用するプログラムが必要とする基本的なサービス群であり，Jini技術のフレームワークを必要とする．具体的にはLookupサービス，Transaction Managerサービス，JavaSpacesサービスがある．

Lookupサービスは，Jini技術の最も重要なサービスであり，利用可能な全てのサービスの登録情報を保持する．Jini技術を利用するアプリケーションは，Lookupサービスの中にあるサービスの登録情報を探すことによって利用したいサービスを見つけ出す．全サービスは必ずLookupサービスに登録することとなる(図2.1参照)．

Transaction Managerサービスは，操作が中断した場合でも，操作が行われていない状態に戻すことが可能である．JavaSpacesサービスは，Javaオブジェクトをネットワーク環境で共有するためのサービスである．

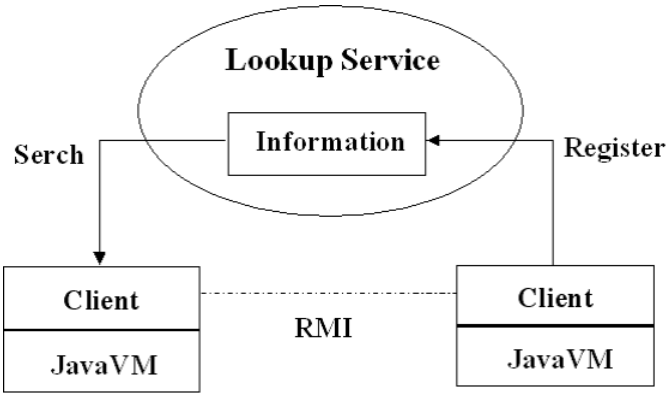


図 2.1: Lookup サービスの概要

2.2 JavaSpaces サービス

2.2.1 概要

JavaSpaces サービスは，ネットワーク上の Java のオブジェクトデータを格納あるいは交換する機能を提供する．オブジェクトデータは，Jini エントリを単位としている．JavaSpaces サービスを利用することで，エントリの共有，交換を行うことで非同期に通信を行うことができる．

JavaSpaces サービスは，基本となるエントリの「書き込み」,「読み出し」,「取り出し」，そしてエントリが JavaSpaces に「挿入されたことを知らせる」という４つの操作を提供する．

2.2.2 エントリ

各プロセス間を移動するオブジェクト単位として，Task エントリ，Result エントリが生成される．

- Task エントリ
  実際の計算プログラムと解くべき問題を保持する．また，隣り合う計算領域における境界値として用いる．

- Result エントリ
  計算によって求められた解を保持する．

2.2.3 プロセス

プロセスには，Master と Worker の二つのプロセスが存在する．

- Master
  Master は，JavaSpaces から Task エントリの配布，Result エントリの回収を行い，必要な処理を行った後，削除する．削除することにより，エントリの検索効率を向上させ，メモリの消費を抑えることができる．

- Worker
  Worker は，JavaSpaces から Task エントリを取り出し計算を行う．計算結果を Result エントリとして JavaSpaces に書き込む．

2.3 Jini 技術の起動

Jini 基本サービスを提供する計算機に認証を行い，Lookup サービス，Transaction Manager サービスの起動後，JavaSpaces サービスの起動を行う．各プロセスを提供する計算機に認証を行い，Worker プロセス，または Master プロセスを起動する．

第3章　UNICORE

3.1　概要

UNICORE(UNiform Interface to COmputing REsource) は，富士通ヨーロッパで開発されたグリッドミドルウェアである．以下に，UNICORE について説明する．

UNICORE の特徴として，Java 言語および Perl 言語で開発されているため多くのプラットフォームに対応し，広域ネットワーク上での利用を考えファイアウォールを越えた運用が容易になっている．

UNICORE の Client には GUI が用意されており，各計算機のアーキテクチャや OS の差異を隠蔽している．ユーザは Client を用いることでアーキテクチャや OS の差異を意識することなく簡単に UNICORE のサービスを利用できる．また，ユーザは，シングルサインオンにより，一度の認証で，許可されている全ての計算機を利用できる．を利用できる．

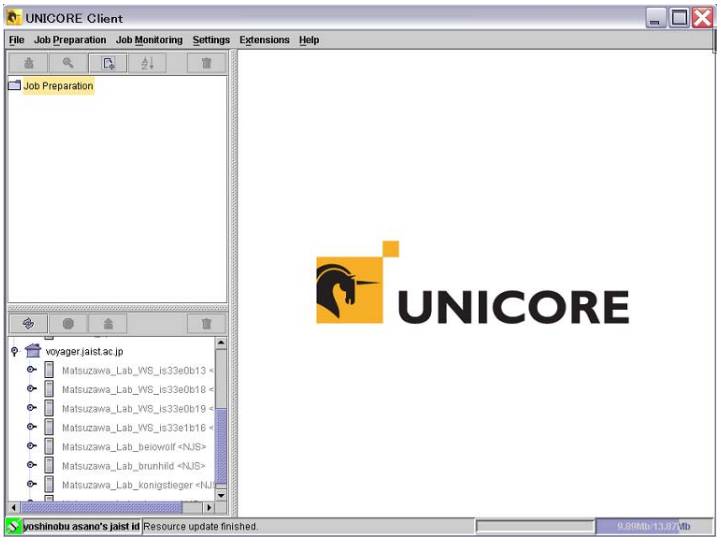


図 3.1: UNICORE Client

3.2 UNICOREの仕組み

UNICORE は，ユーザが利用する「Client」と，サーバ機能を提供する「Gateway」・「NJS」・「TSI」の 4 つのコンポーネントから成り立っている．UNICORE の仕組みを，図 3.2 に示す．

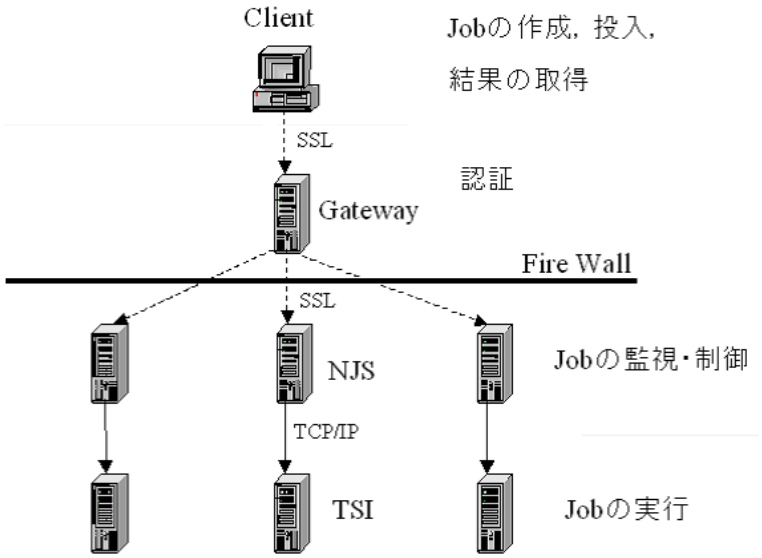


図 3.2: UNICORE System

Client は，ジョブの作成や投入，モニタリングを行う．Client と Gateway の通信は，SSL(Secure Socket Layer) プロトコルで行われ，SSL 相互認証によって通信の確立を行う．Client と NJS(Network Job Supervisor) の通信は，全て Gateway の認証を通して行われる．Client から投入されたジョブは，Gateway を通して NJS に渡される．

NJS は，投入されたジョブを解釈をして TSI(Targer System Inter face) に命令を送信し，ジョブのモニタリングや制御を行う．また，NJS は Client が持つ秘密鍵に対する公開鍵と TSI が動作している計算機のアカウントのマッピング情報である UUDB(Unicore User Data Base) を参照し，ユーザのマッピングを行う．

TSI は，NJS から受け取ったジョブを実行する．

3.3 ジョブ実行の概要

ユーザが Client の GUI を利用してワークフローを作成した後，ターゲットマシンにジョブを依頼するときの概要を図 3.3 を用いて説明する．Client で UNICORE のプログラムを実行させ，サイトリストに接続する Gateway の情報を取得する．

1. Client から指定された Gateway に処理を依頼する．

2. Gateway は Client からの依頼に対して Client の認証を行い，認証されれば Client の依頼を NJS に接続する

3. NJS は依頼されたワークフローを解析し，TSI にタスクを依頼する．このとき UNICORE ユーザとターゲットマシンのユーザマッピングを行う．

4. TSI は NJS から送られてきたタスクをターゲットマシンで処理を行わせる．

5. ターゲットマシンでの処理結果が標準出力ならば，クライアントで出力結果が得られる．

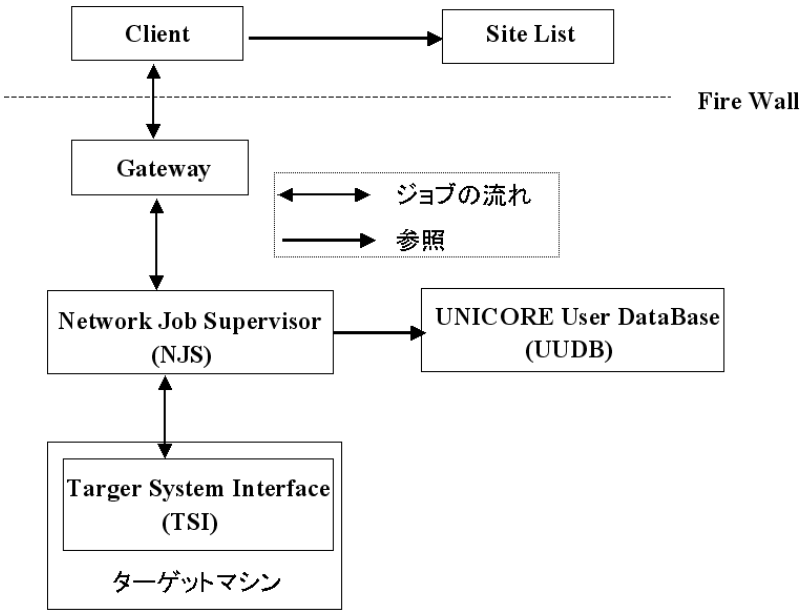


図 3.3: ジョブ実行の概要

ジョブ実行のときに使われるカレントディレクトリは，NJS の tmp ディレクトリである．ターゲットマシンで処理した結果は全て NJS の tmp ディレクトリに保存される．tmp ディレクトリであるため，ジョブ終了時には実行したファイルなどが全て削除される．ファイルを保存する場合には，ターゲットマシンまたはクライアントの適当なディレクトリに保存する．

第4章　リアルタイム可視化システム

リアルタイム可視化システムには，Jini 技術と UNICORE と Java 言語を用いて構築している．以下に，リアルタイム可視化システムについて説明する．

4.1　リアルタイム可視化

シミュレーションと同時に，あるいはシミュレーションの実行と合わせて可視化を行うリアルタイム可視化という．本システムでは計算実行途中での制御 (ステアリング) 機能の実装も含める．

4.2　リアルタイム可視化システム構成

リアルタイム可視化システムは，シミュレーション中に得られた計算結果を，可視化システムに順次転送し，リアルタイムで可視化を行うシステムであり，Master 上に実装されている．システム構成 (図 4.1)，計算機の構成 (表 4.1) に示す．クライアントモジュールは，利用者端末上で動作し，システム操作のための GUI を表示する．

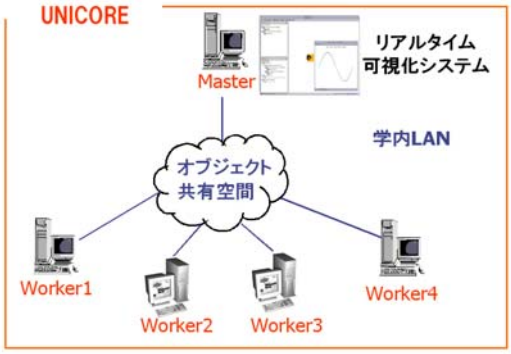


図 4.1: リアルタイム可視化システムの構成

表 4.1: 計算機の構成

	Master	Worker1,2	Worker3,4
CPU	SPARCⅢi 1.0GHz	SPARCⅢi 1.0GHz	SPARCⅡe 550MHz
MEMORY	512MB	512MB	640MB
OS	Solaris-8	Solaris-8	Solaris-9
	Master	Worker×2	Worker×2

4.3 システムの起動

Jini技術を起動するためには，Lookupサービス，Transaction Managerサービス，JavaSpacesサービスを起動した後，Workerプロセスを起動する必要があり，起動操作が繁雑になる．

このようなシステムを使用する場合，全ての計算機の認証を手動で起動する必要がある．多数の計算機を用いて計算を行う場合，逐一，計算機に起動を行うことは非常に手間であり，非効率である． また，複数のWorkerプロセスを立ち上げる場合，各計算機にそれぞれログイン操作を行ってから，Workerプロセスを起動する必要がある．その結果，計算機の台数が増えるにしたがってこのような操作が煩雑になる．

このような問題点を解決するために，UNICOREを用いて一度の認証および操作のみでWorker，Masterプロセスを立ち上げ，同時に計算途中の結果を可視化させるシステムを実装する．

Jini基本サービスの起動

Jini基本サービスであるLookupサービス，Transaction Managerサービス，JavaSpacesを順に起動する．起動方法には，ジョブスクリプト機能を用いる．全てのサービスが起動後，次のジョブへ移行する (図 4.2).

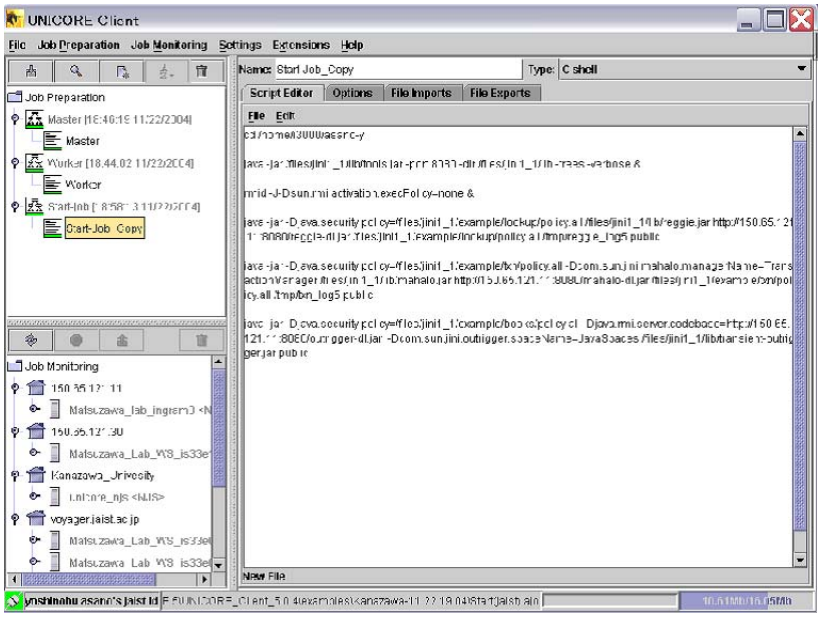


図 4.2: UNICOREジョブの作成

Worker プロセス，Master プロセスの起動

次に各計算機資源において Worker プロセスを起動する．ジョブが実行されると，Master プロセスを起動するジョブが実行される．

4.4 システムの認証

Jini 技術を利用する場合，全ての計算機の認証操作を行う必要がある．このため，多数の計算機を用いて計算を行う場合，この操作が繁雑になる．よって，全ての計算機に UNICORE を実装し，認証操作を簡易化する．利用者は，UNICORE の Gateway の認証を一度受けることで，全ての計算機の認証操作を終了することができる．よって，一度の認証操作で Jini 技術のサービスを利用できる (図 4.3)．

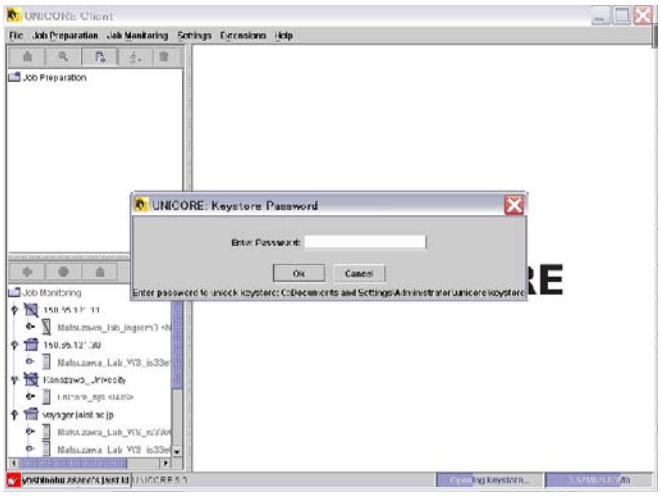


図 4.3: システムの認証

4.5 計算および可視化の実行

以下に計算と可視化が行われる過程を示す.

1. Master プロセスを起動すると，Lookup サービスを通じて Transaction Manager サービス，JavaSpaces サービスを発見する．指定された JavaSpaces に接続し，指定された個数の Task エントリを生成した後，JavaSpaces に投入する (図 4.4).

2. Worker プロセスは，Task エントリを JavaSpaces から取り出し，Task エントリ内に記述された計算プログラムによって与えられた計算領域の計算を実行する (図 4.5).

3. 計算結果より与えられた計算領域における Result エントリを生成する (図 4.6).

4. Worker プロセスが，Task エントリを JavaSpaces から取り出し，計算領域の計算を実行する (図 4.7).

5. Master プロセスは，Worker プロセスによって生成された Result エントリの結果を順次出力する (図 4.8).

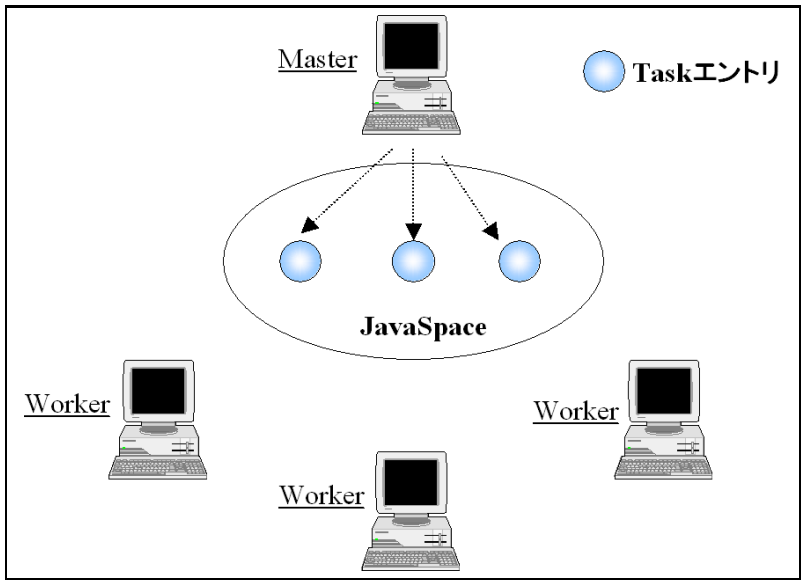


図 4.4: 実行過程 (1)

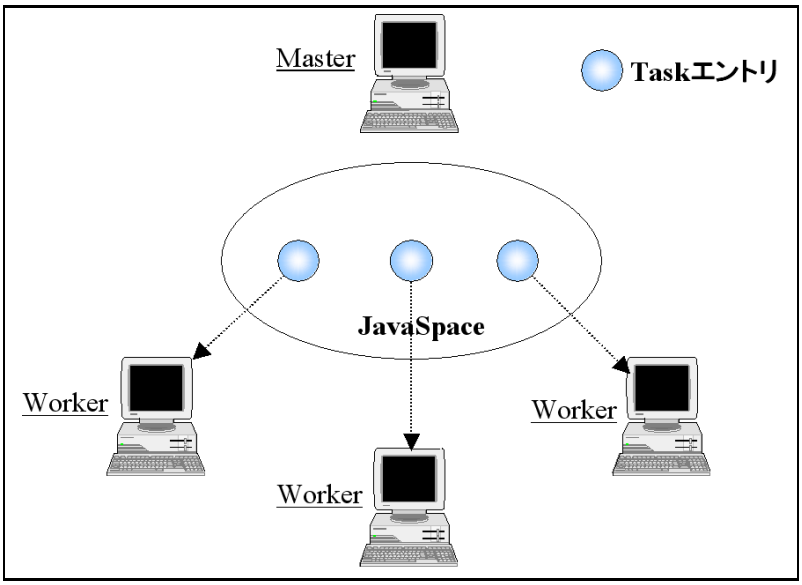


図 4.5: 実行過程 (2)

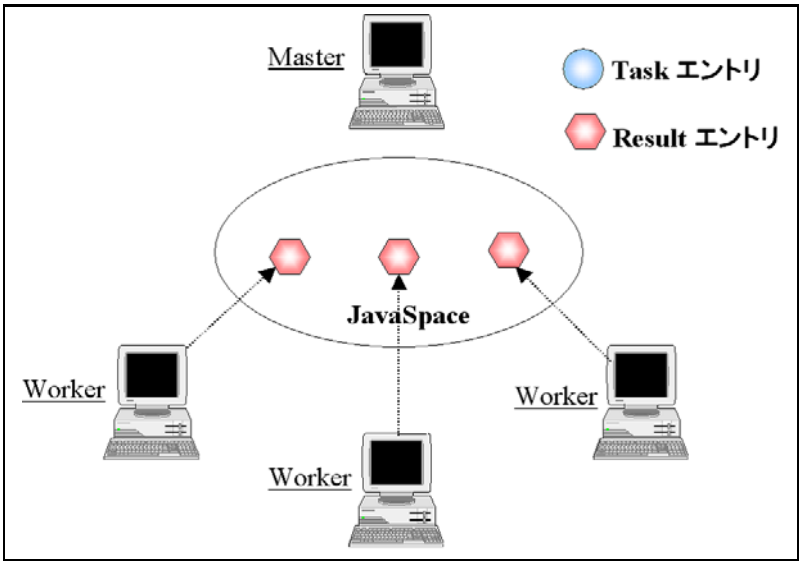


図 4.6: 実行過程 (3)

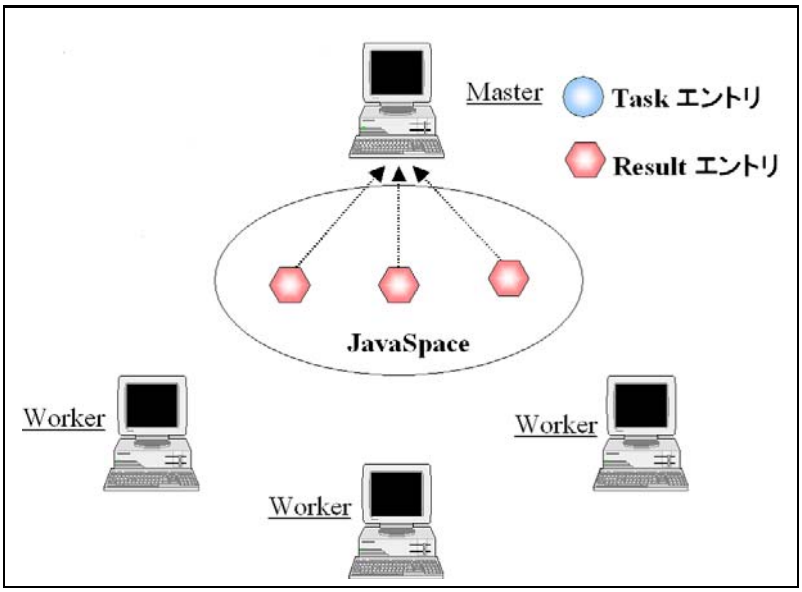


図 4.7: 実行過程 (4)

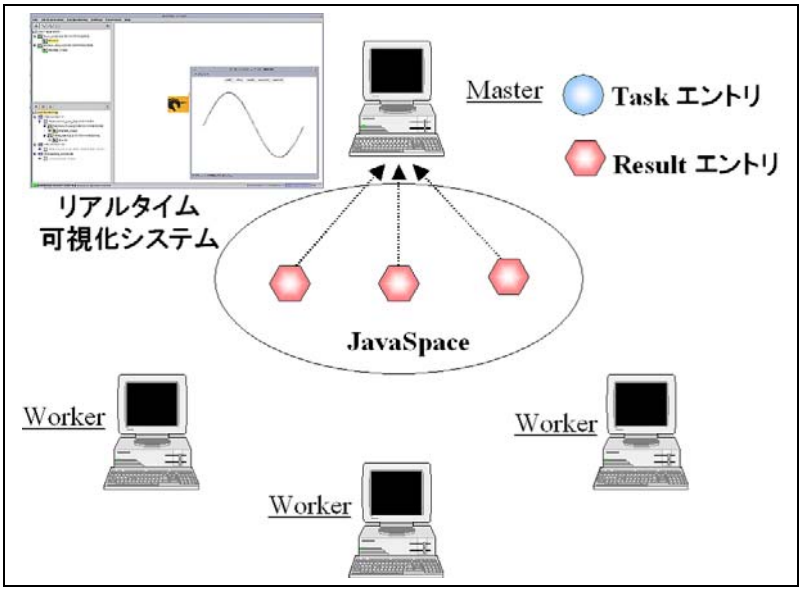


図 4.8: 実行過程 (5)

4.6　可視化の流れ

可視化の流れを図 4.9 に示す．Result エントリが JavaSpaces に存在する場合，Master が Result エントリを回収する．回収した Result エントリから計算結果を取り出し可視化処理を行う．Result エントリが JavaSpaces に存在しない場合，JavaSpaces 内の Result エントリが書き込まれるまで可視化処理が行われず，JavaSpaces への参照を繰り返す．

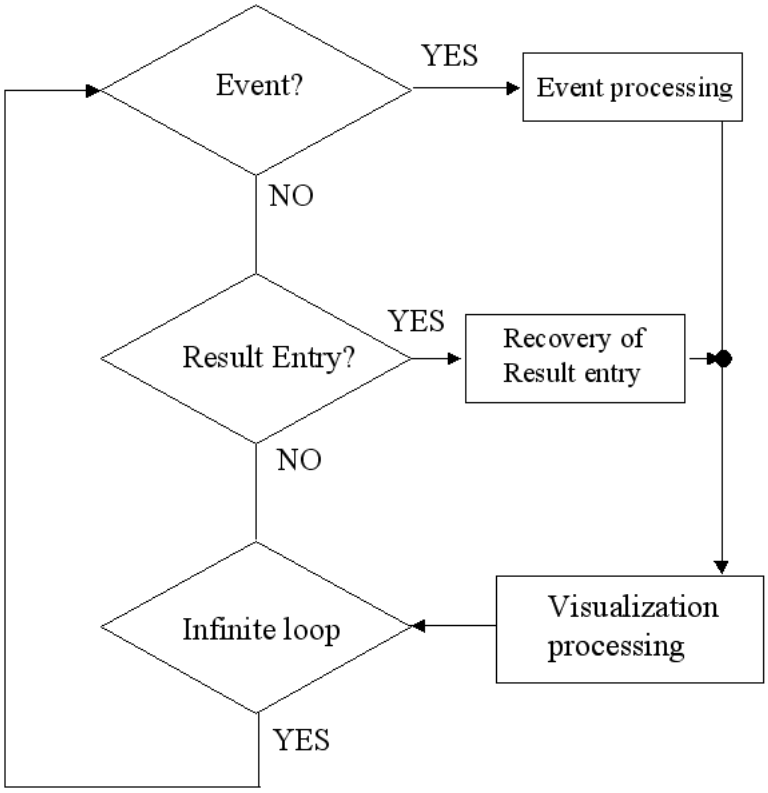


図 4.9: 可視化の流れ

4.7　可視化画面

可視化画面には,「一時停止」,「再開」,「強制終了」ボタンが実装されており，利用者が「一時停止」ボタンを押すと，表示画面が一時停止するとともに，計算全体を一時停止させることができる．また,「再開」ボタンは，一時停止した計算を再開させるボタンであり,「強制終了」ボタンは，プロセス全体を強制終了するボタンである．

第5章　リアルタイム可視化システムの検証と評価

リアルタイム可視化システムを用いて以下の検証を行う.

- 流体の運動方程式を解く際の基本の計算 (行列ベクトル積，行列の加算，行列の積) を用いて，システムの速度向上比を測定する.
- 実装したシステムを用いて，一次元の流体計算し，システムがリアルタイムで可視化を行うことを確認する.
- 一次元の流体計算を発展させ，二次元の流体計算を行う.

5.1　システムの速度向上比

まとまりのある一つの仕事を n 分割して，n 個のプロセッサで並列に実行した場合を考える．この仕事を逐次処理した場合の実行時間を T_1 とし，n 個のプロセッサで並列処理した場合の実行時間を T_n とする．速度向上比 S_p は，$S_p = T_1/T_n$ で定義される.

5.1.1　同性能の計算機による性能評価

同性能の計算機 (WS) により構成された計算環境を用いて計算を行った．この計算機に関する性能を表 5.1 に示す.

表 5.1: 計算機の性能

Machine	Sun-Blade1500
CPU	SparcIIIi(1.0GHz)
MEMORY	512MB
OS	Solaris-8

計算対象および計算手法

本実験で用いた計算は，流体の運動方程式を解く際の基本の計算 (行列ベクトル積，行列の加算，行列の積) を行った．行列の大きさは，400，800，1200 を用いて計算を行った．プログラムの実行時間を計測し，速度向上比を求める．また，計算環境では研究室内の同一フロアにある計算機を使用した．

計算結果

行列ベクトルの積の速度向上比を図 5.1，行列の加算および行列の積の速度向上比を図 5.2，図 5.3 に示す．

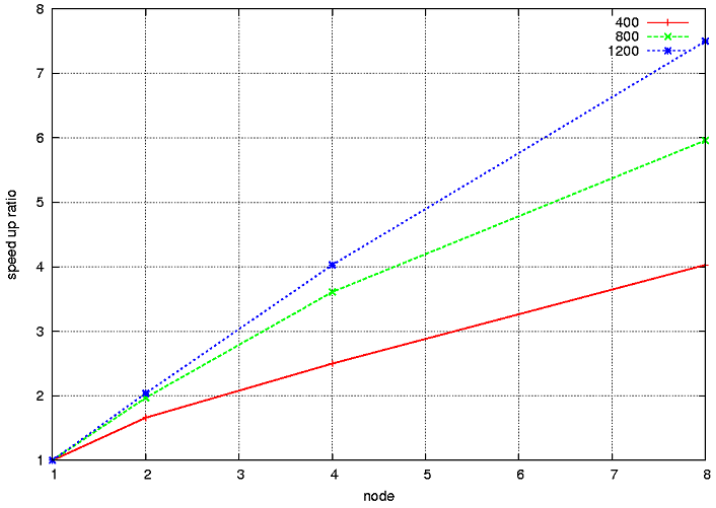


図 5.1: 行列ベクトル積

- 図 5.1〜5.2 から，Worker 数を増加することで全体の処理時間が短縮できた．
- 図 5.1 から，配列のサイズが 400 の場合は速度向上率が 4.30 倍になった．

考察

結果から，配列のサイズが 400 の場合は速度向上率が一番低かった．これは，問題サイズが小さいため，計算時間より通信時間が大きいのが原因と考えられる

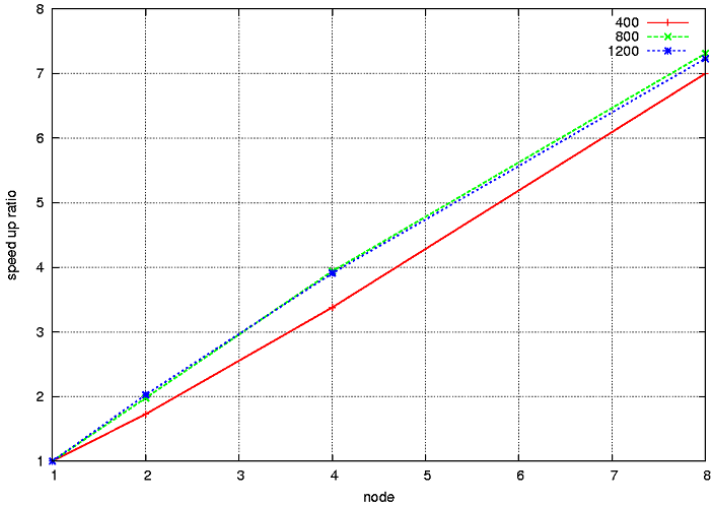


図 5.2: 行列の加算

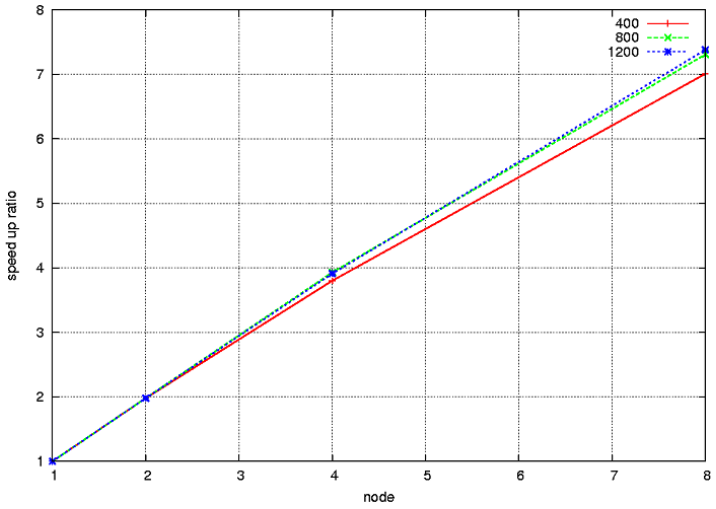


図 5.3: 行列の積

5.1.2 異なる性能の計算機による性能評価

異なる性能の計算機を用いた計算環境では，処理能力の低い計算機に合わせて全体の計算性能が低下する問題がある．そのため，各計算機の処理能力を調べる必要がある．今回用いた方法は，各計算機単体で同じ計算を行い，各計算機の処理能力を測定を行う．

各計算機の処理能力を表 5.2 に示す．

表 5.2: 各計算機の処理能力

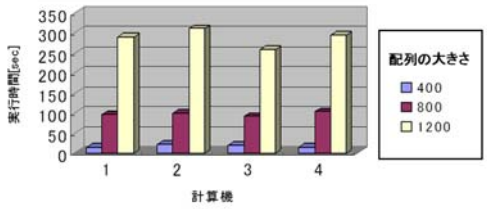
Master,Worker	処理時間 (sec)	性能比
PC1	1017	1.423
PC2	691	2.095
WS1	1448	1.000
WS2	714	2.028

計算対象および計算手法

本実験で用いた計算は，行列ベクトル積を行った．配列の大きさは，400，800，1200 を用いて計算を行った．Jini 技術を用いた分散並列環境で，プログラムの実行時間を計測し，速度向上比を求める．計算環境はプロセス，通信の負荷がある環境で測定を行う．行列ベクトルの積の計算実行時間を測定した値を表 5.3 示す．Worker に処理能力が高い計算

表 5.3: 行列ベクトル積 (計算結果)

	1	2	3	4
Master	PC1	PC2	WS1	WS2
Worker	PC2	PC1	PC1	PC1
	WS1	WS1	PC2	PC2
	WS2	WS2	WS2	WS1



機と低い計算機で計算を行った場合では，処理能力が低い計算機を計算した場合よりも，速度向上率が最大 1.835 倍に向上した．

考察

結果から，Worker に処理能力が高い計算機を割当てることで，全体の処理能力が向上すると考えれる．

5.2　一次元問題 (流体計算)

計算対象として，一次元の Burgers 方程式の数値計算を行う．Burgers 方程式は，類似の対流型非線形項と粘性拡散を有する非線形発展方程式 (5.1) であり，数値流体分野において重要な方程式と用いられる．

次式で表される一次元の Burgers 方程式の計算を行う．

$$\frac{du}{dt} + u\frac{du}{dx} = \nu\frac{d^2u}{dx^2} \tag{5.1}$$

離散化手法は，対流項に一次精度風上差分法，粘性項に二次精度中心差分法を使用する．一次精度風上差分法は，不連続な階を取り扱う場合においては，解の単調性を保持する上で重要な方法である．時間方向に Euler の前進差分法を適用すると，式 (5.1) の差分式は次式となる．

$$u_i^{n+1} = u_i^n + \Delta t\Big[- u_i^n \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} + \Big(\frac{|u_i^n|\Delta x}{2} + v\Big)\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}\Big] \tag{5.2}$$

差分式は，n 時刻での値から $n+1$ 時刻での値を計算できる陽解法 (explicit scheme) を用いる．時間方向に Euler の後退差分法を適用すれば陰解法 (implicit scheme) を用いる．その場合には，対流項に半陰的差分を施して得られる $n+1$ 時刻での解に関する反復式を各時刻において解いている．

初期条件として，$u(x,0)$=$\sin x$ を与える．

境界条件は，$u(0) = 0$ ，$u(2\pi) = 0$.

5.2.1 領域分割

計算領域は，$0 \leq x \leq 2\pi$ なるについて 4 分割する．格子点数は 100 とする．

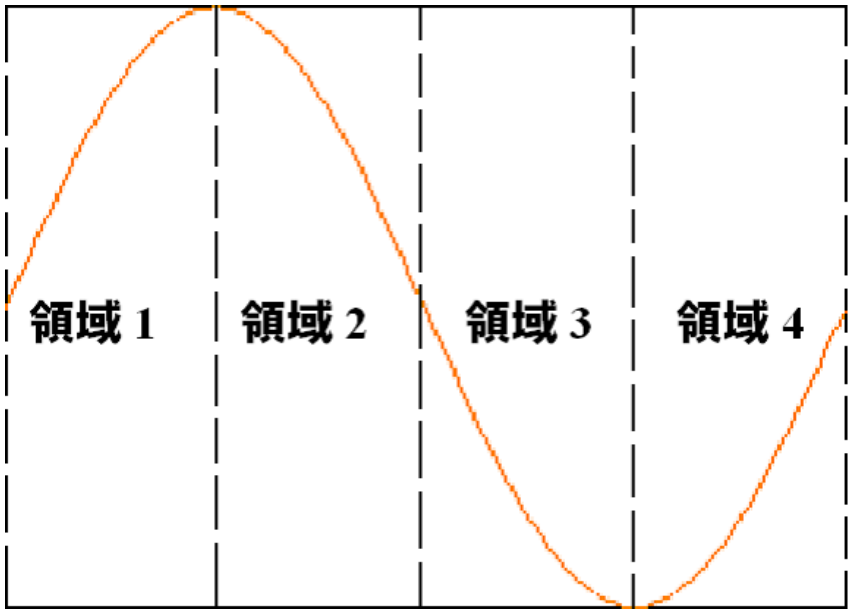


図 5.4: 領域分割

5.2.2 計算結果

リアルタイム可視化システムを用いて，一次元の Burgers 方程式計算を行った．可視化画面の「再開ボタン」を押すことで，数値シミュレーションと可視化処理の両方が実行される.「停止ボタン」が実行すると，数値シミュレーションと可視化処理の両方が一時停止を行う．計算結果は，図 5.5〜5.12 に示した．

図 5.5〜5.8 は，計算を開始した時点 $(t = 1)$ の計算結果，図 5.9〜5.12 は，計算を終了した時点 $(t = 100)$ の計算結果を表している．

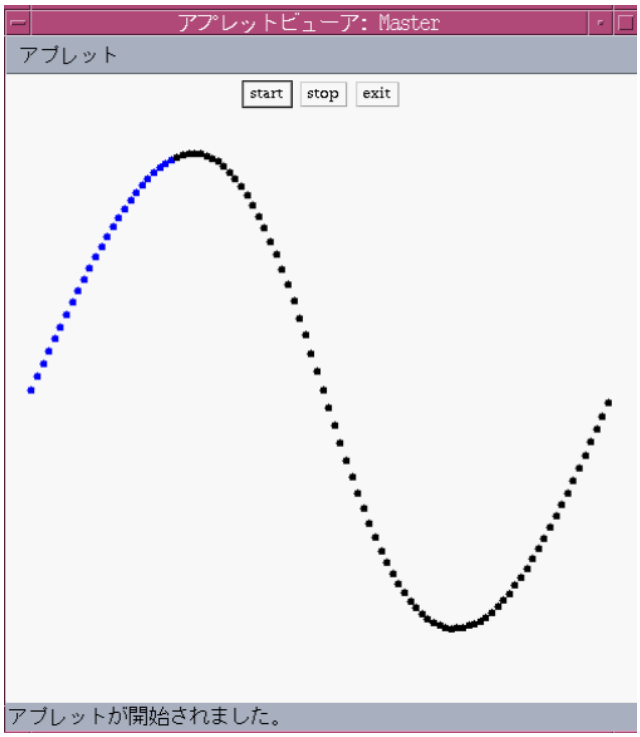


図 5.5: 計算結果 ($t = 1$)

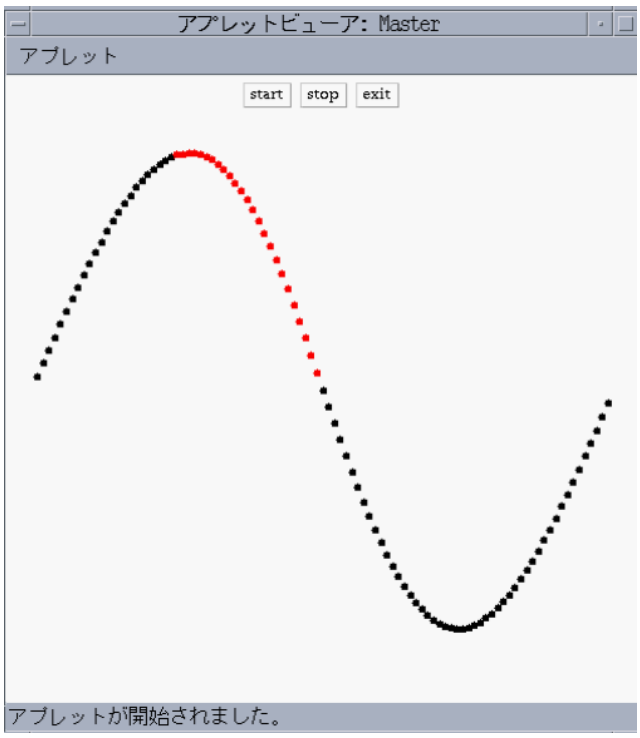


図 5.6: 計算結果 ($t = 1$)

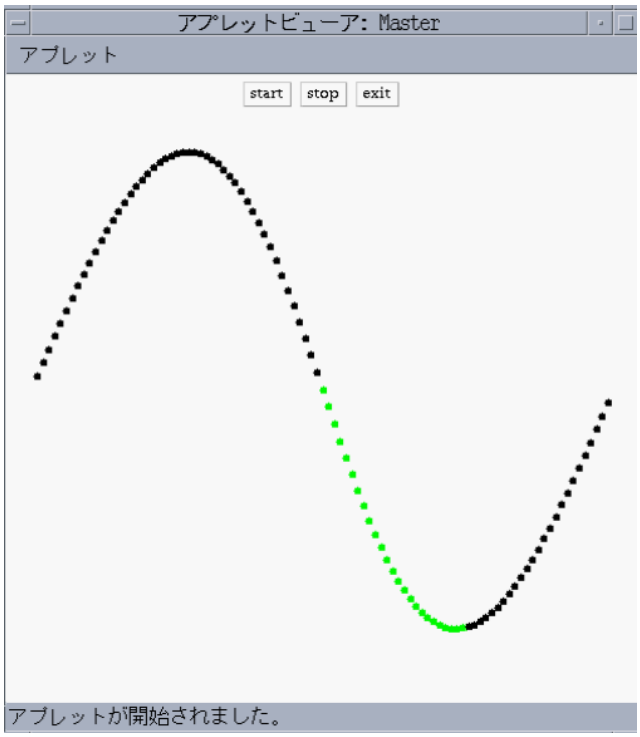


図 5.7: 計算結果 ($t = 1$)

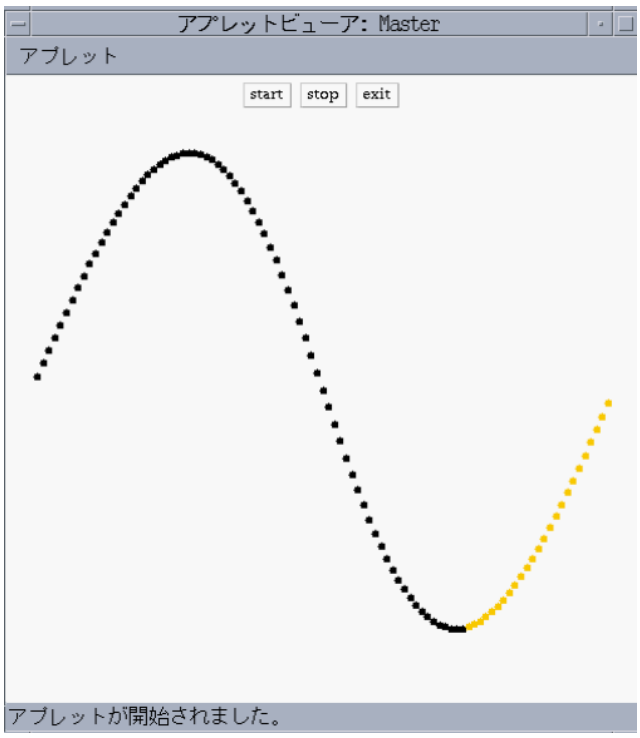


図 5.8: 計算結果 ($t = 1$)

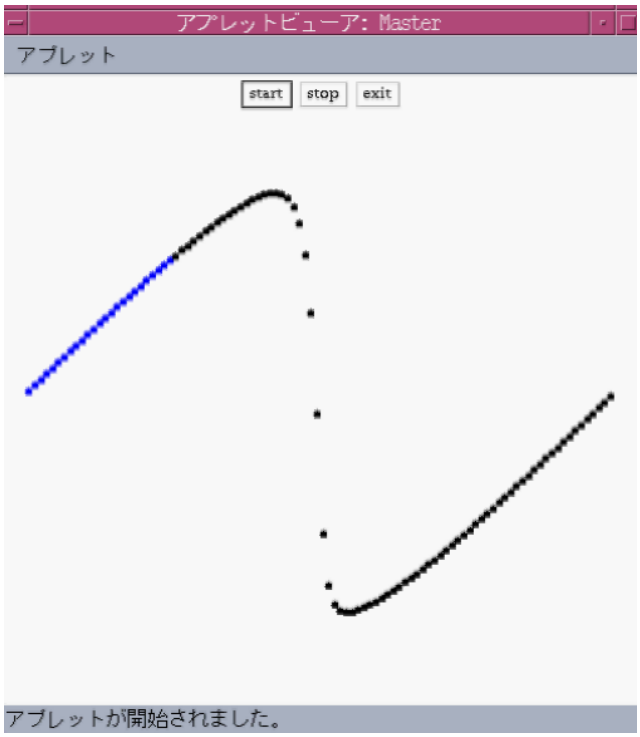


図 5.9: 計算結果 ($t = 100$)

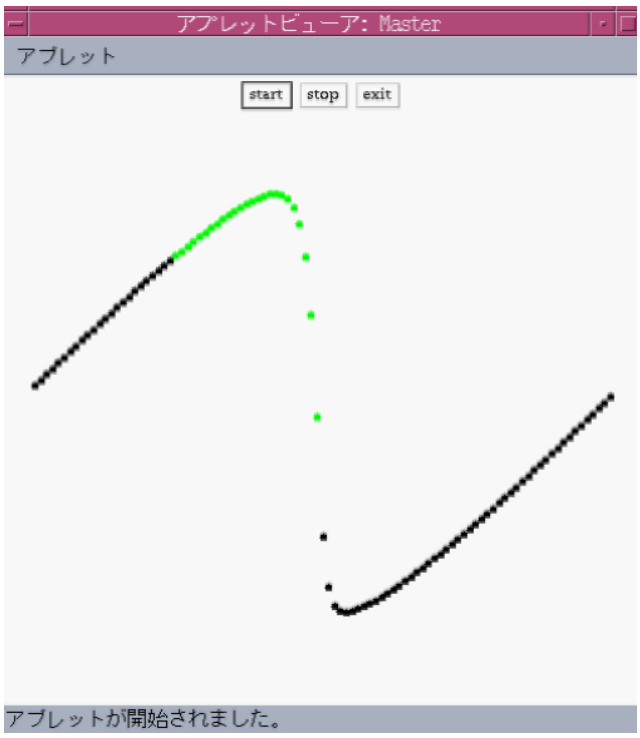


図 5.10: 計算結果 ($t = 100$)

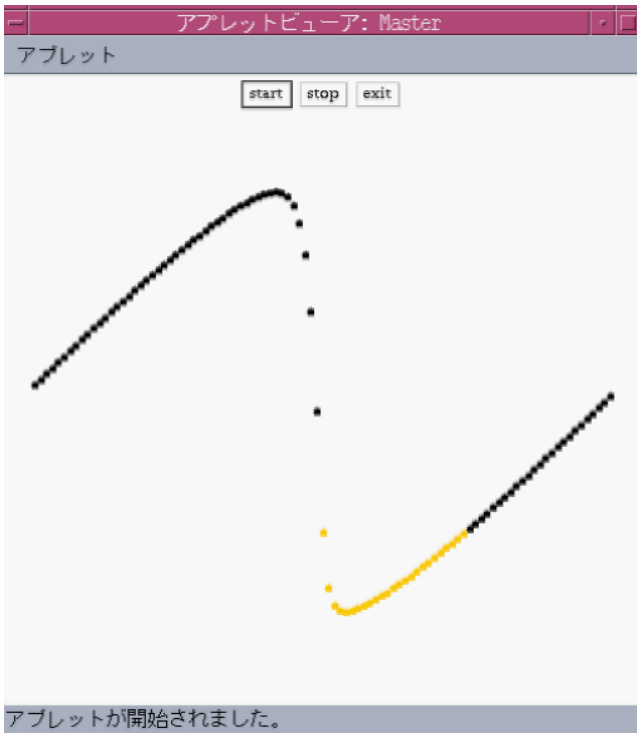


図 5.11: 計算結果 ($t = 100$)

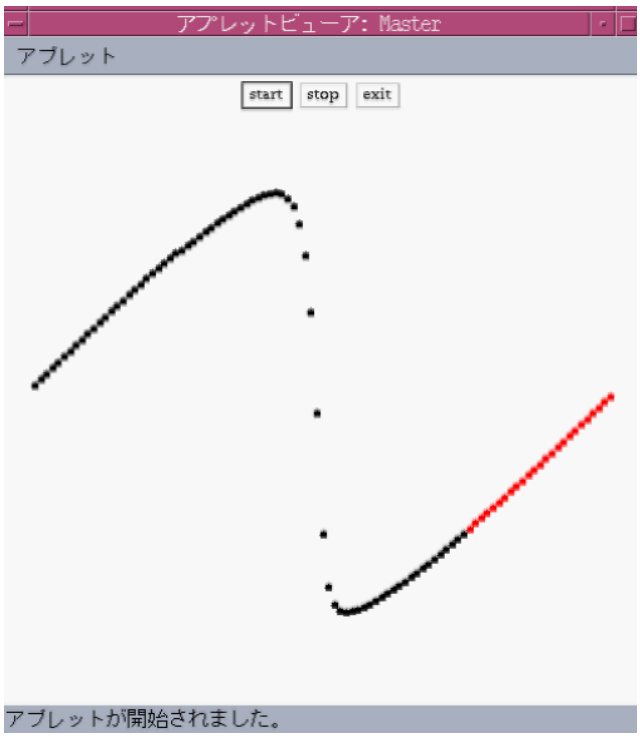


図 5.12: 計算結果 ($t = 100$)

- 図 5.5〜5.8 から，各領域ごとに各 Worker が計算した部分を色分けして可視化を行っている．
- 図 5.5〜5.12 から，可視化画面でリアルタイムで解の挙動が確認できた．
- 一次元の Burgers 方程式の計算結果に共通して，可視化画面の上部にある「一時停止」ボタンは，Worker プロセスの計算が停止し，可視化のアニメーションも一時停止した．また,「再開」ボタンを押すと，Worker プロセスの計算が再開され，可視化システムのアニメーションが再開された．各ボタンによって，計算全体の再開や一時停止の動作を正常通り行えた．

5.2.3 考察

結果から，実装したリアルタイム可視化システムを用いて，一次元の流体計算を実行することが可能であると考えられる．

また，ユーザは可視化画面から各計算機の進行状況が確認できると考えられる．

結果から，計算結果をリアルタイムで出力することで，常に利用者が計算結果を確認することができ，必要に応じて計算全体の再開や一時停止が容易に行えると考えれる．

5.3 二次元問題 (流体計算)

一次元の問題を発展させ，二次元の角柱周りの流れ解析を SOLA 法を適用して解析を行う．流体の運動を支配する方程式は，質量保存を表す連続の方程式 (5.3) および運動保存を表すナビエストークス方程式 (5.4) である．

$$\nabla \cdot v = 0 \tag{5.3}$$

$$\frac{dv}{dt} + (v \cdot \nabla)v = -\nabla p + \frac{1}{Re}\Delta v \tag{5.4}$$

x, y 方向速度を u, v とし，圧力を p とする．

$$\frac{du}{dx} + \frac{dv}{dy} = 0 \tag{5.5}$$

x 方向の運動保存則

$$\frac{du}{dt} + \frac{du^2}{dx} + \frac{duv}{dy} = -\frac{dp}{dx} + \frac{1}{Re}(\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2}) \tag{5.6}$$

y 方向の運動保存則

$$\frac{dv}{dt} + \frac{duv}{dx} + \frac{dv^2}{dy} = -\frac{dp}{dy} + \frac{1}{Re}(\frac{d^2v}{dx^2} + \frac{d^2v}{dy^2}) \tag{5.7}$$

5.3.1 計算アルゴリズム

1. 非圧縮流れでは密度は一定であり，エネルギー保存則を解かなくても速度と圧力は計算できる．

2. 流れ場を有限個の等間隔セルに分割する (図 5.13 参照).

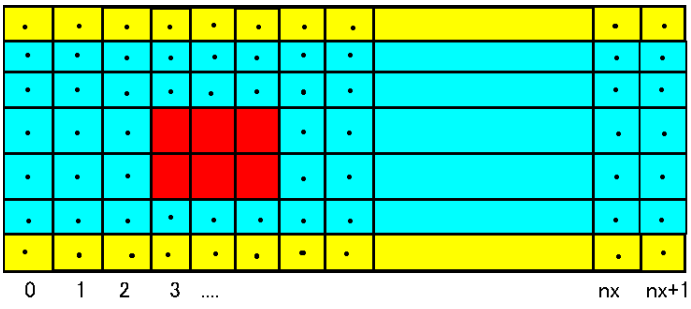


図 5.13: セル番号と境界

3. セルの中心に圧力をセルの境界に速度を定義する (図 5.14 参照).

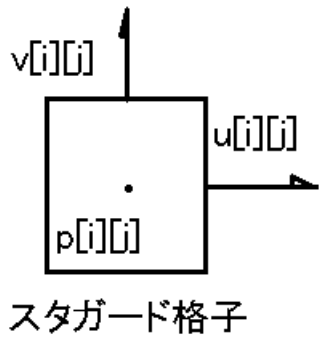


図 5.14: スタガード格子

4. 流れ場に置かれた物体形状を近似するように，流れ場と物体を区別する.

5. 初期値，境界値を設定する．初期値では，物体部分を除いて速度と圧力を一定に設定する.

6. 数値安定状態を満足する時間刻み幅 dt を求める.

7. セルの境界の圧力とセルからの流入・流出する運動量を周辺のセルの物理量から計算する．求めた運動量の流出と流入の合計から，現時点の解から dt 経過した時点の速度場を計算する．SOLA 法では，スタガード格子を使用しているため，x 方向の速度 u を計算する場合は半メッシュずれたセルを検査面として使用する (図 5.15).

運動量の式を離散化すると，$n+1$ 時刻の速度 u は時刻 n の状態から以下の式を用いて計算を行う.

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{dt(p_{i,j} - p_{i+1,j})^n}{dx} \tag{5.8}$$

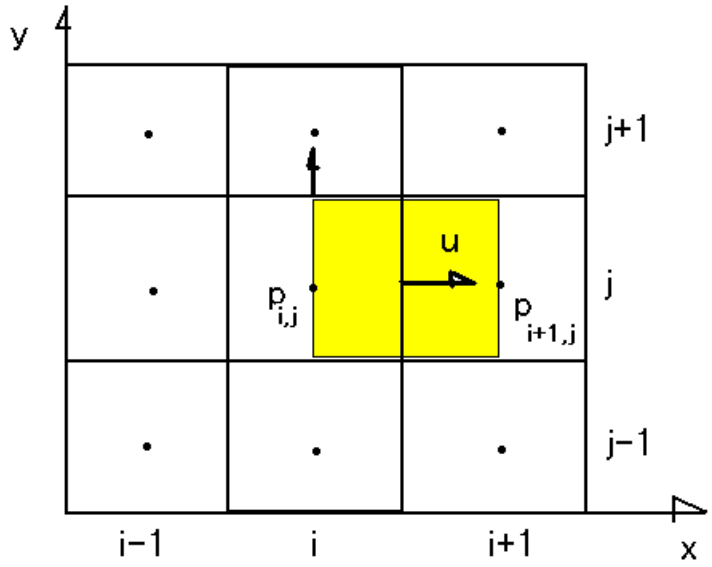


図 5.15: 速度の計算に対するセルの検査面

8. 求めた速度場は，連続条件を満足していないため，連続式から圧力の修正量を計算し，圧力変化による速度変化から連続条件を満たすように圧力を修正し，連続条件が満たされるまで計算を繰り返す．連続式を満たすように，以下の式を用いて圧力の修正を行う．

$$-\beta \cdot \delta = \frac{u_{i,j} - u_{i-1,j}}{dx} + \frac{v_{i,j} - v_{i,j-1}}{dy} \tag{5.9}$$

$$p_{i,j} = p_{i,j} + \delta * p_{i,j}, \qquad \beta = 2dt(dx^{-2} + dy^{-2}) \tag{5.10}$$

9. 連続条件が満たされた状態を次の流れ場の状態とする．

10. 流れ場の速度ベクトル等を描画して計算状況を確認する．

11. 境界値の設定，dt の計算を繰り返す．

5.3.2 計算格子

100 × 100 の計算格子を用いる．(図 5.16 参照)

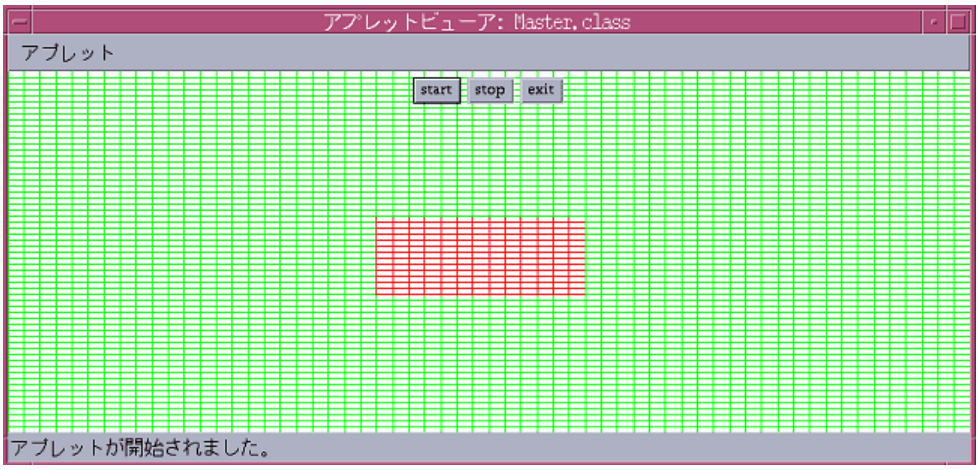


図 5.16: 計算格子

5.3.3 計算結果

角柱周りの流れ解析をスタガード格子，SOLA 法を用いて計算した流れ解析の結果を，図 5.17〜5.24 で示す．x 方向及び y 方向に 100 × 100 に分割し，流れ解析結果の図の表示は，$t = 100$ ごとに可視化表示を行った．

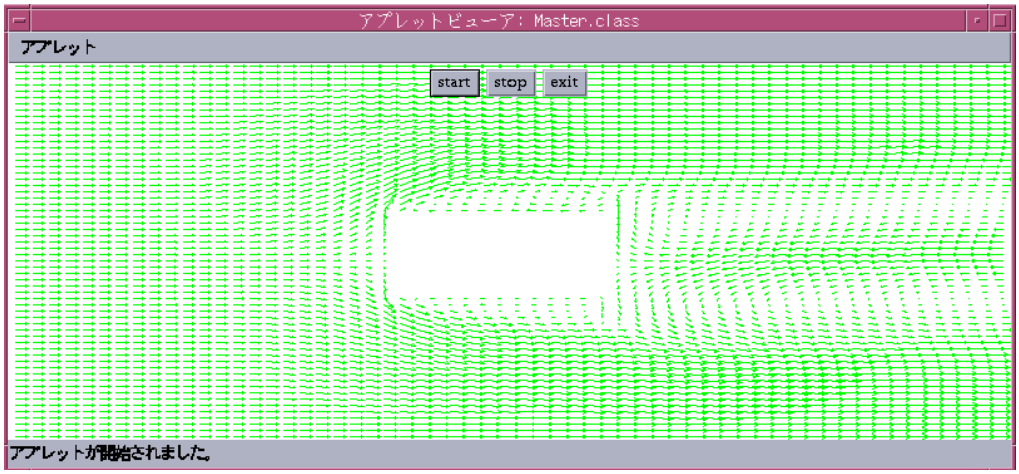


図 5.17: 計算結果 ($t = 1$)

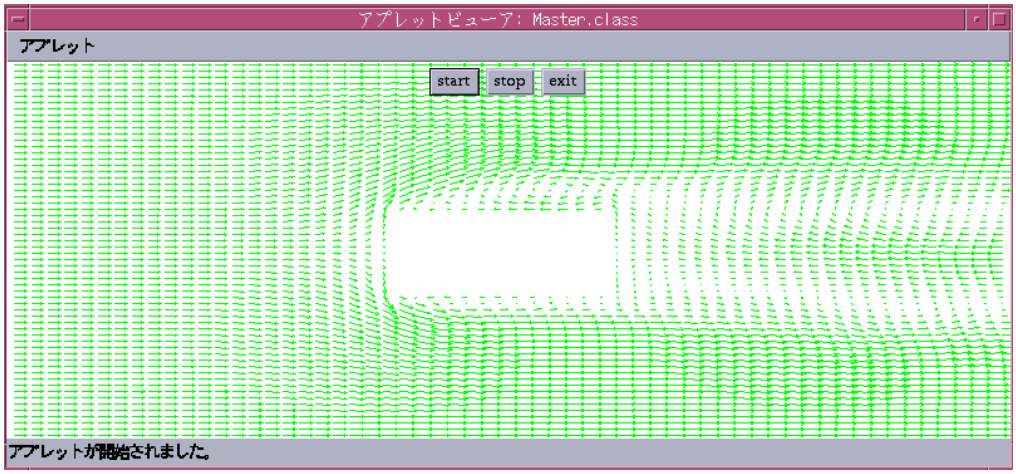


図 5.18: 計算結果 ($t = 100$)

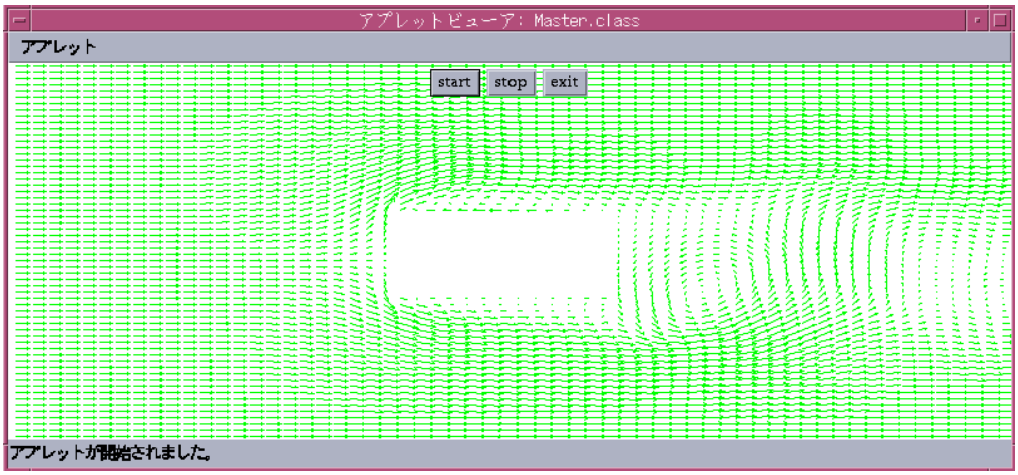


図 5.19: 計算結果 ($t = 200$)

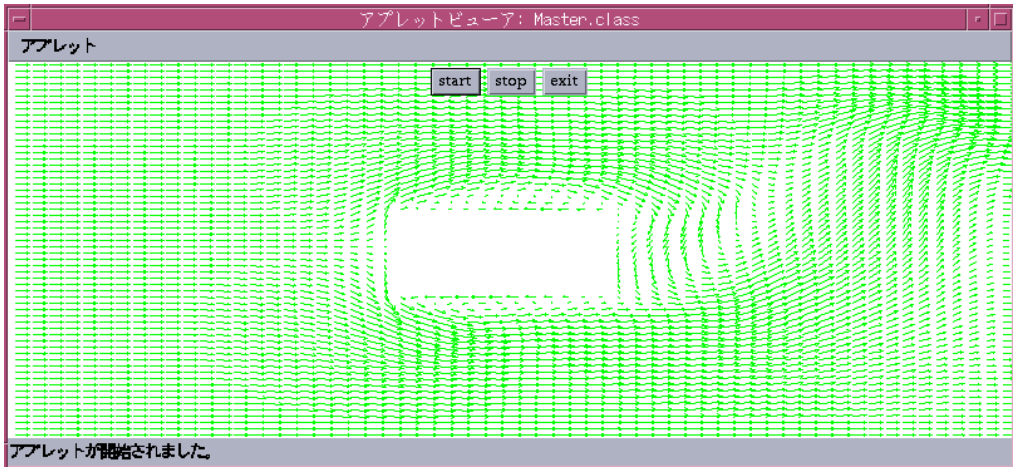


図 5.20: 計算結果 ($t = 300$)

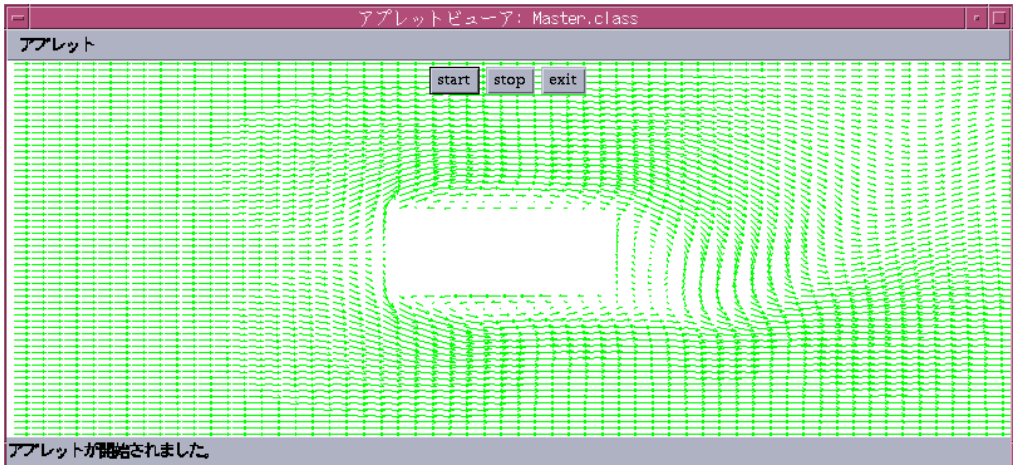


図 5.21: 計算結果 ($t = 400$)

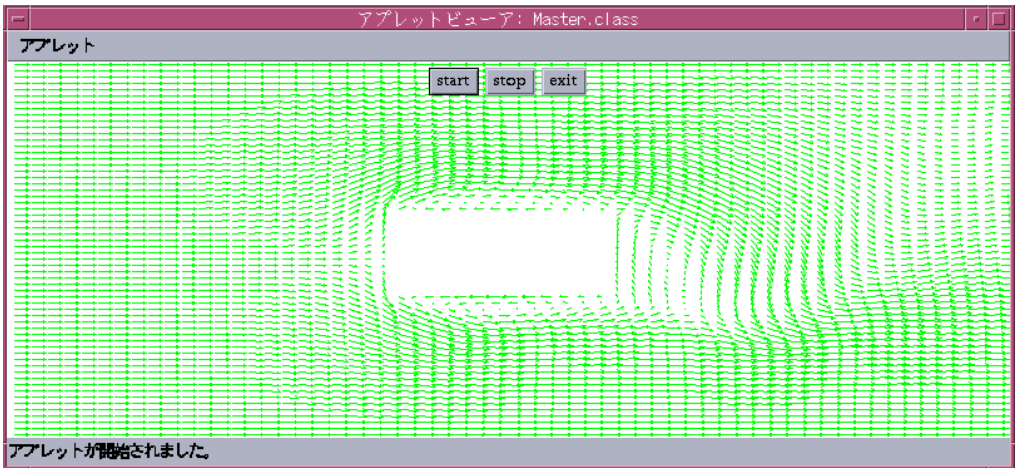


図 5.22: 計算結果 ($t = 500$)

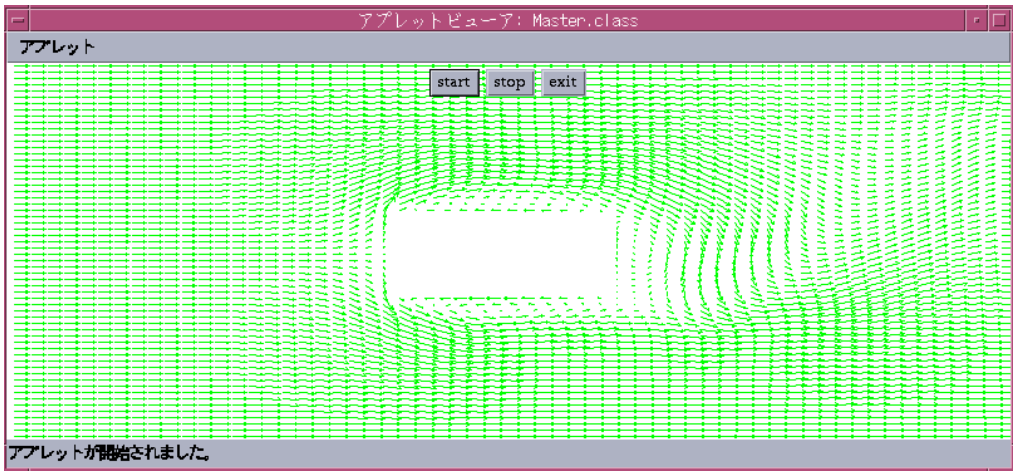


図 5.23: 計算結果 ($t = 600$)

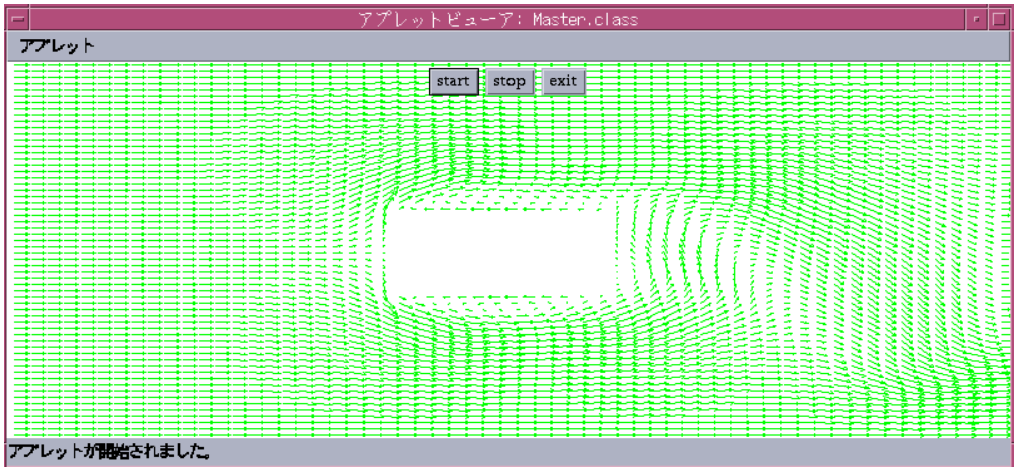


図 5.24: 計算結果 ($t = 700$)

- 図 5.17〜5.24 から，可視化画面でリアルタイムで解の挙動が確認できた．
- また，二次元の流体計算をすることができた．
- 二次元の角柱周りの流れ解析の計算結果に共通して，可視化画面の上部にある「一時停止」ボタンは，Worker プロセスの計算が停止し，可視化のアニメーションも一時停止した．また,「再開」ボタンを押すと，Worker プロセスの計算が再開され，可視化システムのアニメーションが再開された．各ボタンによって，計算全体の再開や一時停止の動作を正常通り行えた．

5.3.4 考察

結果から，実装したリアルタイム可視化システムを用いて，二次元の角柱周りの流れ解析を実行することが可能であると考えられる．

また，計算結果をリアルタイムで出力することで，常に利用者が計算結果を確認することができ，必要に応じて計算全体の再開や一時停止が容易に行えると考えれる．

第6章　結言

6.1　まとめ

利用者が，計算途中の結果を確認できるリアルタイム可視化システムの構築および実装を行った．リアルタイム可視化システムは，計算全体の一時停止および再開を行うことができる機能を実装しており，利用者は容易に計算の停止再開が行えるようになった．

計算対象に，一次元の Burgers 方程式と二次元の角柱周りの流れ解析を行い，全ての計算が終了する前に計算結果をリアルタイムで確認することができ，システムの有効性が確認できた．また，並列計算を本システムで実行させた場合も，計算した結果をリアルタイムに可視化することができた．

以上の結果より，数値解析を用いるシミュレーション結果の可視化において，本研究で実装したリアルタイム可視化システムは実用的な可視化方式の一つであると結論付ける．

6.2　今後の課題

構築および実装したシステムを，学外の機関と連携させ，広域ネットワーク上で計算，リアルタイム可視化を行い，有効性を検討する．また，流体計算の並列化を行い，本システムに適用させる．

謝辞

本論文の作成に際し，有益な助言，御指導，御鞭撻を賜わりました，松澤 照男 教授に深く感謝致します.

また，渡邉 正宏 様 (情報科学センター)，大岩 博史 様 (本学博士後期課程)，太田 理 様 (本学博士後期課程)，松澤研究室の皆さんに深く感謝いたします。

2005 年 2 月

浅野 喜宣

参考文献

[1] Jini, "http://www.sun.com/software/jini"

[2] 溝渕貴裕, "Jini 技術を用いた分散並列計算環境", 第 17 回数値流体力学講演論文集, (2003), pp.126.

[3] Globus Toolkit, "http://www.globus.org/"

[4] UNICORE, "http://www.unicore.org"

[5] pV3, "http://raphael.mit.edu/pv3/pv3.html"

[6] Terascale Visualization "http://www.llnl.gov/terascaale-viz/"

[7] 永井亨, "UNICORE 環境の構築と並列計算実験", ss 研 Grid Computing WG 成果報告書, (2003), pp.52.

[8] 城之内忠正, "実習で学ぶ Java と XML による流体シミュレーション", 日本数値流体力学会主催 CFD 講習会資料, (2002)

[9] 服部高資, "JavaSpaces を用いた分散計算環境の構築", 第 16 回数値流体力学講演論文集, (2002), pp.229

本研究に関する発表論文

[10] 浅野喜宣，大岩博史，松澤照男 ”Jini 技術とグリッドミドルウェアを用いたリアルタイム可視化システムの構築”， 第 18 回数値流体力学講演論文集，(2004)， pp.234