修 士 論 文

分散計算環境でのリアルタイム可視化に関する研究

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

松本浩之

2006 年 3 月

修 士 論 文

分散計算環境でのリアルタイム可視化に関する研究

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　党建武 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

410112 松本浩之

提出年月: 2006 年 2 月

概 要

近年の計算機の性能向上は目覚しいものがあり，数値流体力学 (Computational Fluid Dynamics:CFD) における数値流体シミュレーションは計算機の性能向上とともに飛躍的に発展した．しかし，より大規模かつ複雑な問題を解くには計算機性能はいまだ不十分であり，計算機性能にも限界がある．これを克服する技術としてグリッドコンピューティングが挙げられる．

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ネットワークで結び，論理的に足し合わされた計算資源として利用するものである．ユーザは地理的に計算資源が分散していることや通信経路，また通信に関する暗号化を意識する必要はない．グリッドコンピューティングを構成するミドルウェアとして，UNICORE (UNiform Interface to COmputing REsorces) [1]がある．また，ネットワーク間で共有したオブジェクト空間が構築できる Jini 技術[2]がある．Jini 技術は台数・処理能力ともに動的に計算資源が変化する場合においても柔軟に対応ができる．

一方，数値流体シミュレーションが大規模かつ複雑になるほど，計算結果を得るまで多くの時間がかかる．よって計算の失敗が生じたとしても利用者がそれを知るまでに多くの時間を要する．また，計算結果を把握するために任意のタイムステップでのファイルを計算機から取得し可視化するのは，ストレージ容量の問題や，ファイル出力時における数値シミュレーションへの負荷が考えられるため問題になる．これを克服する方法として，ファイルを介さず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がある．利用者はこれによって計算の間に計算が正しく行われているか確認することや計算の進行状況を知ることができる．

本研究では，３次元数値流体シミュレーションのリアルタイム可視化システムを，UNICORE と Jini 技術とを利用した分散計算環境上で構築し，システムの有効性を検証した．２次元数値流体シミュレーションに比べ，３次元数値流体シミュレーションは計算データ量が増加する．ここで，Jini 技術が提供する共有オブジェクト空間を共有メモリとして利用する．Jini 技術は Java 言語で構成される．Java 言語は言語自体マルチプラットフォームをサポートしており，異機種の計算資源によって構成された分散計算環境で動作する．グリッドミドルウェアは Java 言語と Perl 言語によって構成され，Java 言語に親和性のある UNICORE を用いた．

また可視化に関しては，シミュレーションの結果が得られる度に可視化システムへ計算結果を転送し可視化を行うことによって，リアルタイム可視化を実装した．これによって，利用者へ計算の進行状況やシミュレーションの結果をいち早く知らせることができる．この際，３次元の計算データを可視化情報として扱うと通信及び描画処理からの負荷が考えられる．よって，任意の座標軸に対して垂直に切った断面を可視化することによって，可

[1]UNICORE,”http://www.unicore.org”

[2]Jini 技術,”http://www.sun.com/software/jini”

視化に必要な計算データを減らす．さらに，可視化断面の変更・追加機能を実装し，利用者は任意の可視化断面を確認することによって１枚の可視化断面では得られなかった奥行き方向の挙動を確認できる．

構築したシステムを用いて３次元角柱流れ解析を行い，任意の座標軸の断面によるリアルタイム可視化の実現とその可視化断面の変更・追加の確認，ステアリング機能の実装の確認，可視化によるシミュレーションの実行時間への影響と高速化率の測定，異機種の分散計算環境における実行と負荷分散の考察によってシステムの有効性を検証した．

結果，各実装についてそれぞれ動作を確認できた．可視化によるシミュレーションの実行時間への影響を測定し，可視化なしに比べ圧力分布図の描画では 2.22%，速度ベクトル図の描画では 4.65%の遅延が見られた．

分散計算環境としての評価として，同機種分散計算環境における高速化率では４台での計測で最大 1.91 倍の効率が得られ，異機種分散計算環境における高速化率では８台での計測で最大 2.02 倍の効率が得られた．本システムでは JavaSpaces をタスクバッグとして構成した負荷分散を適用し，検証を行った．タスクの分割数が計算機台数と同等の場合，タスクの実行回数は計算機性能によらず一定となり，負荷分散が行われなかった．しかし計算機台数よりもタスクの分割数が多い場合，性能に応じてタスクの実行回数が変化しており，負荷分散が行われた．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

近年の計算機の性能向上は目覚しいものがあり，数値流体力学 (Computational Fluid Dynamics:CFD) における数値流体シミュレーションは計算機の性能向上とともに飛躍的に発展した．しかし，より大規模かつ複雑な問題を解くには計算機性能はいまだ不十分であり，計算機性能にも限界がある．これを克服する技術としてグリッドコンピューティングが挙げられる．

グリッドコンピューティングを構成するミドルウェアとして，globus [1] や，UNICORE (UNiform Interface to COmputing REsorces) [2] がある．globus は，グリッドコンピューティングにおける標準的なサービスを提供するための基盤アーキテクチャとして策定された OGSA(Open Grid Services Architecture) を用いたＣ言語ベースのミドルウェアである．また UNICORE は，Java 言語や Perl 言語によって構成され，計算資源の環境に依存することなく利用することができる．一方，ネットワーク間で共有したオブジェクト空間が構築できる Jini 技術 [3] があり，台数・処理能力ともに動的に計算資源が変化する場合においても柔軟に対応ができる．

従来の数値シミュレーションにおける可視化では，計算機で得られた計算結果をファイルとして保存し，利用者端末に転送してから可視化を行ってきた．数値流体シミュレーションが大規模かつ複雑になるほど，計算結果を得るまで多くの時間がかかる．よって計算の失敗が生じたとしても利用者がそれを知るまでに多くの時間を要する．また，任意のタイムステップでファイル出力するには，ストレージの問題や，ファイル出力時における解析への負荷が考えられる．よって，ファイルを介さずシミュレーションの実行と合わせて可視化を行うことにより，計算の間に計算が正しく行われているか確認することや計算の進行状況を知ることのできる，リアルタイム可視化の必要性が高まってきた．

UNICORE と Jini 技術を用いた分散計算環境の構築が行われている [4]．浅野ら [5] が構築したリアルタイム可視化システムは２次元流れの速度に関する可視化が行えるが，可視化を２次元で表示しているため３次元数値流体シミュレーションを行った際，奥行き方向の挙動を確認することができない．また浅野らは，3 章で述べる HSMAC(Highly Simplified Marker And Cell) 法の速度予測子の計算で分散計算を適用しているが，速度予測子の計算に比べ計算負荷の大きい圧力と速度の修正計算には適用していない．

1.2 目的

浅野らの２次元数値流体シミュレーションに比べ，３次元数値流体シミュレーションは計算データ量が増加する．ここで，Jini 技術が提供する共有オブジェクト空間を共有メモリとして利用する．Jini 技術は Java 言語で構成される．Java 言語は言語自体マルチプラットフォームをサポートしており，異機種の計算資源によって構成された分散計算環境で動作する．グリッドミドルウェアは Java 言語と Perl 言語によって構成され，Java 言語に親和性のある UNICORE を用いた．

また可視化に関しては，シミュレーションの結果が得られる度に可視化システムへ計算結果を転送し可視化を行うことによって，リアルタイム可視化を実装した．これによって，利用者へ計算の進行状況やシミュレーションの結果をいち早く知らせることができる．この際，３次元の計算データを可視化情報として扱うと通信及び描画処理からの負荷が考えられる．よって，任意の座標軸に対して垂直に切った断面を可視化することによって，可視化に必要な計算データを減らす．さらに，可視化断面の変更・追加機能を実装し，利用者は任意の可視化断面を確認することによって１枚の可視化断面では得られなかった奥行き方向の挙動を確認できる．

本研究では，３次元数値流体シミュレーションのリアルタイム可視化システムを，UNICORE と Jini 技術とを利用した分散計算環境上で構築した．構築したシステムを用いて３次元角柱流れ解析を行い，任意の座標軸の断面によるリアルタイム可視化の実現とその可視化断面の変更・追加の確認，ステアリング機能の実装の確認，可視化による測定時間への影響と高速化率の測定，異機種の分散計算環境における実行と負荷分散の考察によってシステムの有効性を検証した．

1.3 本論文の構成

本論文では，本システムで用いた UNICORE と Jini 技術の概要を 2 章で，本研究で用いた計算解法の概要を 3 章でそれぞれ述べる．4 章より本研究で実装した分散計算環境上で運用するシステムを説明し，5 章では本システムの有効性を確認するために実行した数値流体シミュレーションを述べ，6 章で考察を行う．7 章でまとめと今後の課題について述べる．

第2章　分散計算環境

2.1　UNICORE

UNICOREはグリッドミドルウェアのひとつであり，同様の技術としてglobusが挙げられる．グリッドミドルウェアは，異機種である計算資源を一様に管理運用でき，地理的に計算環境が分散していることや計算資源との認証，暗号化通信を意識する必要なく，利用できる全ての計算資源を統合して利用できる環境を提供する．さらにUNICOREは，以下が特徴として挙げられる．

- 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ている (マルチプラットフォーム)
- ファイアウォールを越え広域ネットワーク上での運用ができ，一度の認証だけで許可されている計算機全てを利用できる (シングルサインオン)
- スクリプト言語やフローチャートをGUI(Graphical User Interface) を用いて書き込むことでジョブを構成できる (ワークフロー)

2.1.1　UNICOREの構成

UNICOREは，利用者端末である”UNICORE Client”，クライアントの接続と認証を行う”UNICORE Grid site(Usite)”，計算機資源やデータ資源からなり計算機資源の異機種を隠蔽しジョブを実行する”Virtual site(Vsite)” の３層構造になる．また，さらにUNICOREは，”Client”，”Gateway”，”NJS(Network Job Supervisor)”，”TSI(Target System Interface)” の４つのモジュールによって構成されている (図2.1).

それぞれのモジュールに関する機能は以下の通りである．

- Client
  GUIを持った利用者端末であり，ジョブを構成しGatewayを通じてNJSへ処理の実行命令や停止，処理のモニタリングが行える
- Gateway
  Clientの認証，NJSのハード・ソフトウェア情報の集約，そしてジョブの送信の中継を行う

- NJS
  TSI が実行されるターゲットマシンのアカウントをマッピングした UUDB(UNICORE User DataBase) を参照してユーザの認可を行い，ジョブを解釈して TSI に具体的な処理を実行させる

- TSI
  ターゲットマシンに常駐し処理を NJS から受け付け，ターゲットマシンに実際の処理を行わせる

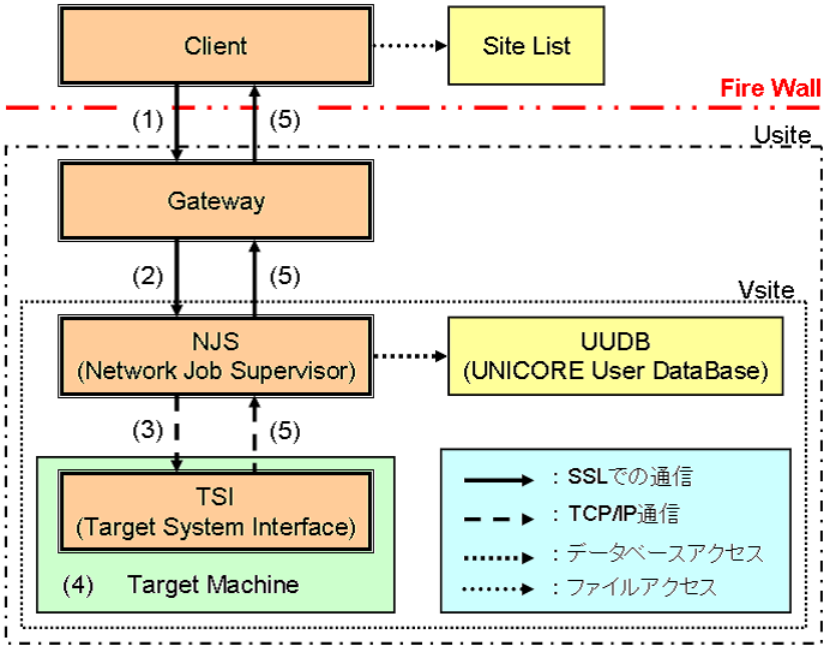


図 2.1: UNICORE の構成と処理の流れ

ターゲットマシン 1 台につき Vsite は 1 つという対応関係となり，Gateway は複数の Vsite を束ね，Client は複数の Gateway を束ねる．
FireWall に多数のポートを開ける必要がある globus に比べ，UNICORE は限られたポートを開けるだけで運用できる．また，電子鍵証明書の標準仕様とされる X.509 に基いた Client 証明書，Gateway 証明書，NJS 証明書，認証局から発行された認証局証明書を用い相互認証を行う．さらに，Client,Gateway,NJS 間の通信では SSL(Secure Socket Layer) プロトコルを用いた暗号化通信を行う．

2.1.2 ジョブ実行の概要

ユーザが UNICORE を用いて分散計算環境を構築し，ジョブを実行する過程を述べる．

1. UNICORE の起動を行う．実際にジョブを実行するターゲットマシン上で NJS 及び TSI を起動する．また，ターゲットマシンからファイアウォールを越えないネットワーク内で，Gateway を立ち上げる．

2. ユーザが利用者端末上で Client を実行し，Gateway との接続で必要となるキーストアの鍵を開けるパスワードを入力し，Client を起動する．図 2.2 は UNCORE クライアントの画面であり，左上の小エリアに作成したジョブ，左下の小エリアには接続されている Usite と Vsite が表示される．データの通信中，ジョブの実行中，ジョブの実行終了，ジョブの異常終了が，アイコンの変化によって通知される．右のエリアは作業エリアであり，ジョブの作成やジョブの結果表示が行われる．

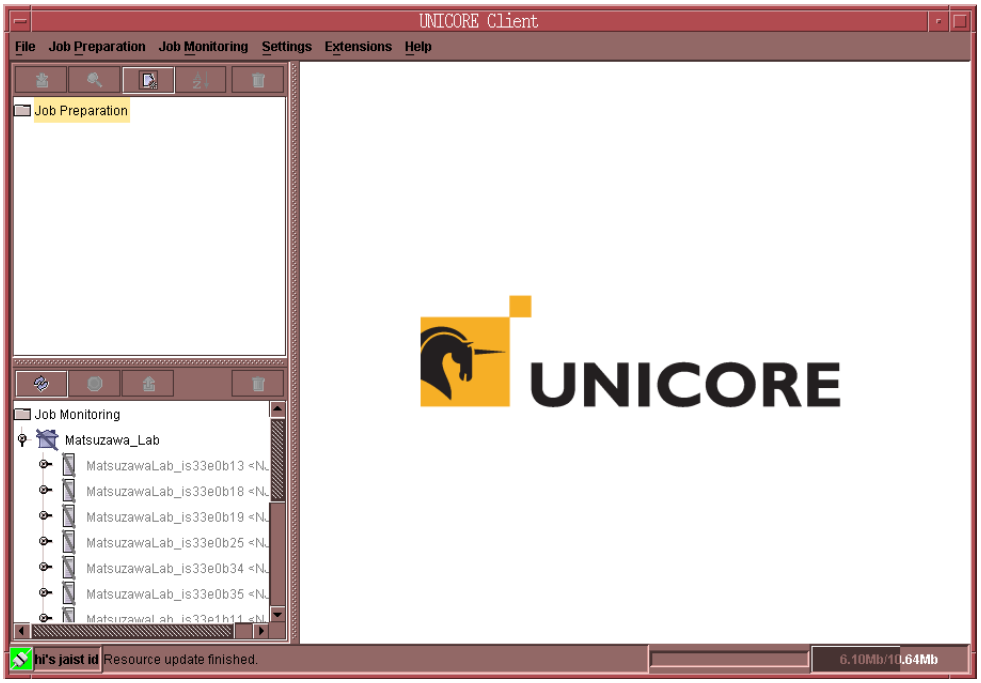


図 2.2: UNICORE クライアント

図 2.3 の作業エリア左上のエリアでジョブを実行する Usite を，作業エリア左下のエリアで Vsite を設定し，実行するジョブを作成する．UNICORE で作成できるジョブの実行内容には，シェルスクリプトの実行，実行形式のプログラムの実行，Fortran プログラムのコンパイルとリンク，NJS やターゲットマシンとのファイル操作がある．

図 2.4 は，実行するシェルスクリプトの作成を作業エリアで行っている様子である．さらに，ジョブを階層構造にするサブジョブの作成，条件分岐，ループ処理，指定した時間までジョブを停止といったジョブの操作が可能であり，図 2.3 の作業エリ

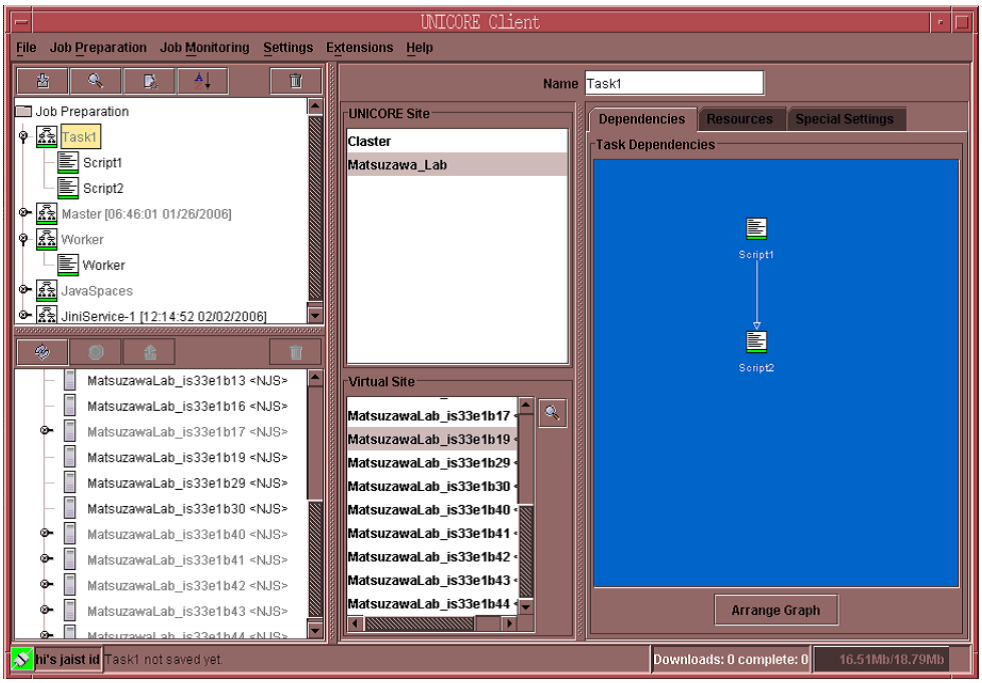


図 2.3: ワークフロー機能

ア右のエリアで表すように，複数のジョブの実行内容をフローチャートにして設定できる (ワークフロー機能).

3. Client は自身が持つ SiteList を参照し，Gateway へアクセスし，ジョブを依頼する (図 2.1(1))

4. Gateway は Client の依頼に対して認証を行い，認証された場合に依頼を NJS へ中継する (図 2.1(2))

5. NJS はユーザと TSI が実行される計算機のユーザマッピングを UUDB を用いて行い，認可された場合，依頼されたジョブを解釈し，TSI へ具体的な処理を依頼する (図 2.1(3))

6. TSI は NJS から送られてきた処理を Target Machine 上で実行する (図 2.1(4))

7. 処理結果が標準出力である場合，Client で出力結果が得られる (図 2.1(5)). 図 2.5 は出力結果が得られた様子であり，作業エリアに標準出力が表示されている.

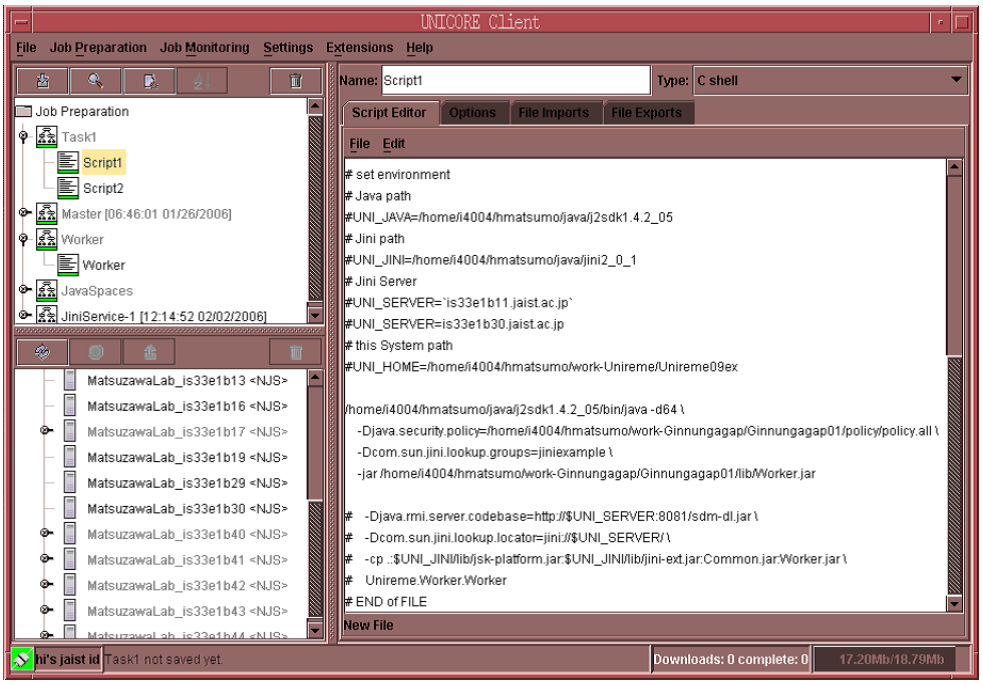


図 2.4: ジョブの実行内容の作成

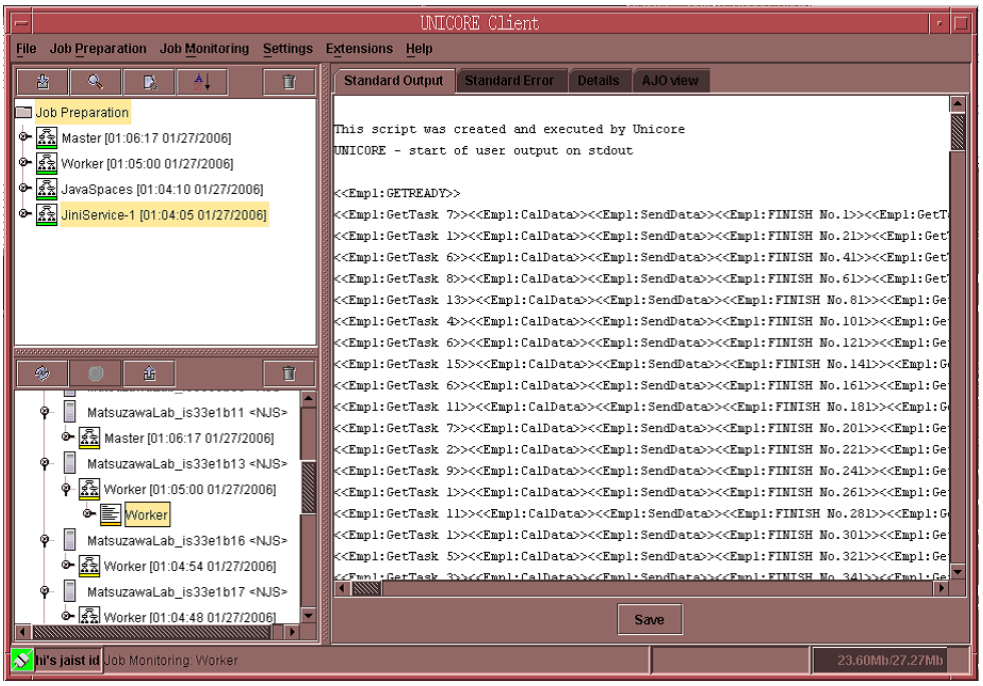


図 2.5: ジョブの結果表示

2.2 Jini 技術

Jini 技術は，1999 年に Sun Microsystems によって発表された Java 言語をベースにした分散オブジェクト技術であり，計算資源や電話，家電製品を一様なインタフェースとして利用し，相互に機能を提供しホームネットワークを構築するために提案された．しかし昨今では，異機種の計算資源を一様に扱うことのできる技術として注目され，ネットワークに接続された計算資源やプログラム，デバイスをグループ化させコミュニティを形成し，１つの動的な分散計算環境を構築する技術として用いられる．

Java 言語で記述された Java オブジェクトを通して演算能力やストレージなどの機能を提供するものは全て”サービス”と呼ばれる．Jini 技術はサービスが生成・消滅されるような動的にサービスが変化する環境下においても対応することができる．

なお，溝渕ら [4]，浅野ら [5] が用いた Jini のバージョンは 2000 年 10 月に Sun Microsystems より公開された Jini1.1 であるが，本システムでは 2005 年 3 月に公開された Jini2.0.1 を利用した．Jini2.0.1 は，Jini1.1 とは規格が異なり，SSL 通信や Kerberos 認証による機密保護モデルを導入し，Jini のプログラムに関しても最適化が行われている．

Jini 技術を利用するために必要なサービス群を，Jini 基本サービスと呼ぶ．これらは Jini 技術の中核を成すプログラム群であり，提供するサービスのインデックスとなる Lookup サービスや，共有オブジェクト空間を提供する JavaSpaces サービス，操作が中断した場合でも操作が行われていない状態へ戻す Transaction Manager サービスが挙げられる．また，サービスの利用を支援するため，新しく生成されたサービスを通知する Discovery 機能や，通信障害などで異常終了したサービスを自動的に排除する Lease 機能，あるサービスが利用された時に通知を受けることができる Event 機能が実装されている．

2.2.1 Lookup サービス

Jini 技術で最も重要なサービスであり，Lookup サービスの立ち上がったサーバを Jini サーバと呼ぶ．サービスの Proxy Code が格納され，Client は Proxy Code を通してサービスを利用することができる．ここで，Jini サーバへアクセス可能なネットワーク範囲をコミュニティと呼び，コミュニティ内では Lookup サービスが管理するサービス全てを利用することができる．また，Proxy Code は，そのサービスのインターフェイスと，サービス内容に関するその他の説明が含まれている登録情報である．図 2.6 に Lookup の過程を示す．

1. サービスを提供する Service Provider がコミュニティ内に接続されたとき，Jini 基本サービスを提供する Jini Server の Lookup Service に，Service Provider の Proxy Code が登録される (図 2.6 (1)Discovery)

2. Client がサービスを利用したい場合は，Lookup Service へ問い合わせ，Proxy Code を読み込む (図 2.6 (2)Lookup)

3. Proxy Code に書かれた接続先にアクセスを行い，サービスを利用する (図 2.6 (3)Call Service)

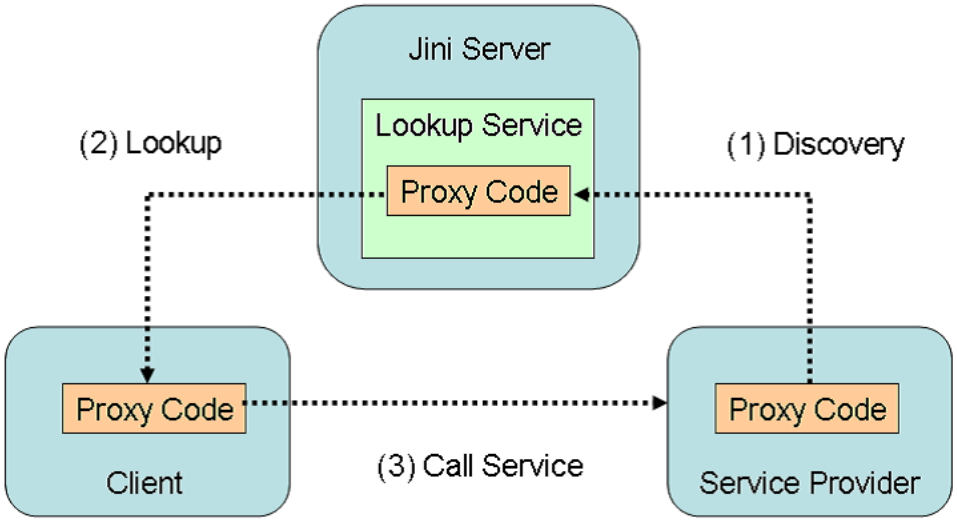


図 2.6: Lookup サービスの流れ

2.2.2 JavaSpaces サービス

分散計算環境上で Java オブジェクトデータを共有することができるサービスがあり，このサービス及びオブジェクトの共有空間を JavaSpaces と呼ぶ．JavaSpaces 内には，Java で書かれた Entry オブジェクトを保管することができ，Jini 技術を用いることによって，JavaSpaces へ Entry オブジェクトの「書き込み」，Entry オブジェクトの「読み出し」や「取り出し」を行うといった操作が可能である．

JavaSpaces サービスの流れを図 2.7 に示す．コミュニティでは，JavaSpaces サービスが立ち上がった端末と，MachineA,B が存在しており，MachineA が JavaSpaces へ Entry オブジェクトを「書き込み」，それを MachineB が「取り出す」過程を示している．

以下に JavaSpaces に対する操作について述べる．

1. MachineA が JavaSpaces へ Entry オブジェクトの「書き込み」を行った場合，MachineA のメインメモリ上の JavaSpaces へ Entry オブジェクトがコピーされ (図 2.7 (1)Copy)，JavaSpaces サービスに Entry オブジェクトの型情報が通知される (図 2.7 (2)Notice)．

2. MachineB は JavaSpaces サービスへ，読み出す Entry オブジェクトの型をテンプレートとして指定することで (図 2.7 (3)Appoint)，JavaSpaces サービスはテンプ

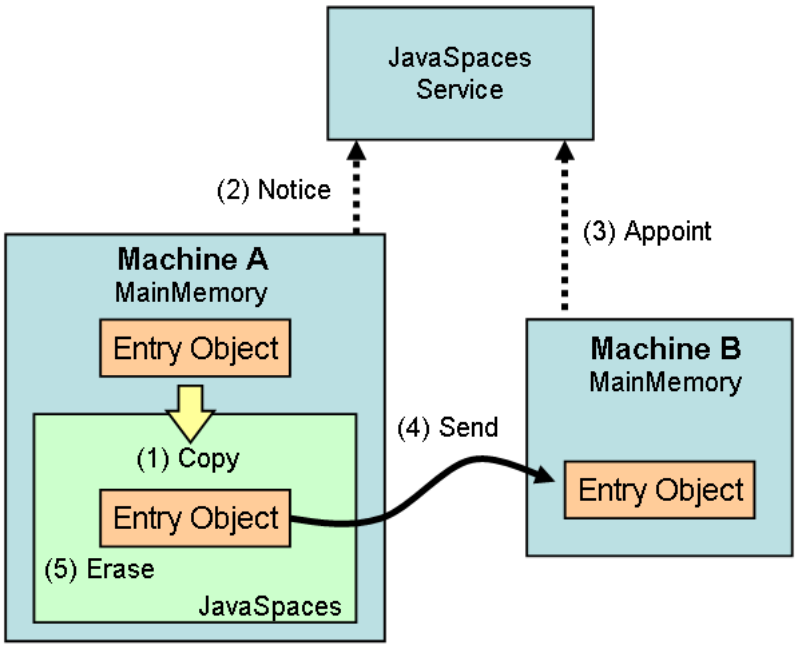


図 2.7: JavaSpaces サービスの流れ

レートにマッチングする Entry オブジェクトを探し，Entry オブジェクトが保存された MachineA を特定し，MachineA の接続先を通知する．そして，MachineB は MachineA から Entry オブジェクトの「読み出し」を行える (図 2.7 (4)Send).

3. 「取り出し」の場合,「読み出し」を行った Entry オブジェクトを JavaSpaces から削除する操作が加わる (図 2.7 (5)Erase).

JavaSpaces は，JavaSpaces サービスを介しているためどの Machine に Entry オブジェクトが置かれているかを隠蔽でき,「読み出し」や「取り出し」は JavaSpaces サービスの情報とテンプレートとのマッチングによって成り立つため目的の Entry オブジェクトが置かれる Machine を指定する必要がなく，あたかもネットワーク上に存在する共有メモリ空間のように振舞う．ここでテンプレートと JavaSpaces サービスのマッチングは以下の条件によって判別される．

- テンプレートの型が，JavaSpaces に格納された Entry オブジェクトの型と一致している．もしくはテンプレートは Entry オブジェクトのサブクラスである．
- テンプレートの変数に値が格納されている場合，JavaSpaces に格納された Entry オブジェクトの対応する変数に同値が格納されている．

図 2.8 は，後者のマッチングが適合した例を示している．$id = 2$ が格納されたテンプレートを JavaSpaces サービスへ指定し「読み出し」を行うと，JavaSpaces 内の $id = 2$ が格納された Entry オブジェクトが得られる．

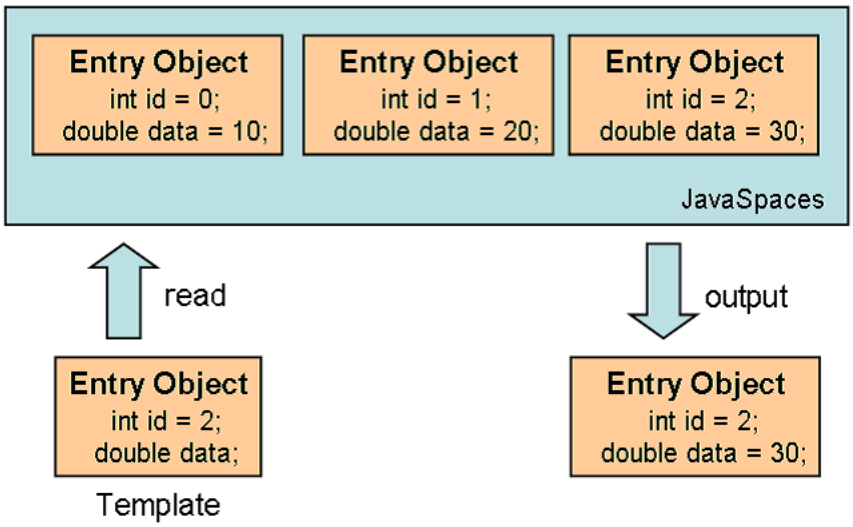


図 2.8: テンプレートとのマッチング

図 2.9 は，５台の Machine のメインメモリ上に JavaSpaces が置かれた際の概念図である．各 Machine のメインメモリに Entry オブジェクトが作成され，JavaSpaces が置かれている．破線で区切られた領域はメインメモリの領域であり，物理的に区切られているものである．点線で区切られた領域は各 Machine のメインメモリを共有した JavaSpaces である．JavaSpaces は各 Machine のメインメモリに分散して置かれる．よって物理的な隔たりを持ち，１つの Entry オブジェクトを複数の Machine で共有して持つことはできないが，論理的には各 Machine のメインメモリに置かれた JavaSpaces 統合された共有メモリ空間を構成できる．

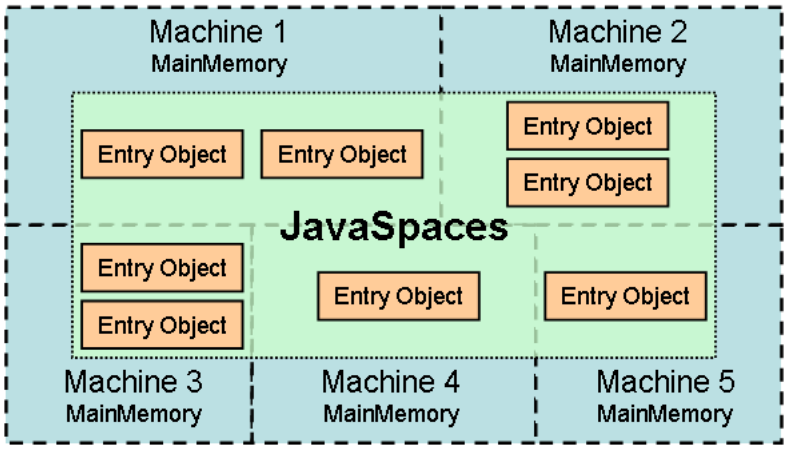


図 2.9: JavaSpaces 概念図

第3章　計算解法

本システムは非圧縮性流体を計算対象とし，計算解法として HSMAC 法を適用した．HSMAC 法は速度の予測値を用いる 2 段階解法であり，Poisson 方程式をより高速に解くため，優対角化した反復法によって速度と圧力を同時に修正しつつ時間進行させる高速解法である．以下に HSMAC 法の概要を述べる．基礎方程式は式 (3.1 - 3.4) で表される．連続の式として，

$$\frac{du}{dx} + \frac{dv}{dy} + \frac{dw}{dz} = 0 \tag{3.1}$$

x, y, z 方向の運動保存則は以下のように表される．ここで，x, y, z 方向の速度成分を u, v, w，圧力を p，時間を t，Re をレイノルズ数とする．

$$\frac{du}{dt} + \frac{du^2}{dx} + \frac{duv}{dy} + \frac{duw}{dz} = -\frac{dp}{dx} + \frac{1}{Re}(\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} + \frac{d^2u}{dz^2}) \tag{3.2}$$

$$\frac{dv}{dt} + \frac{duv}{dx} + \frac{dv^2}{dy} + \frac{dvw}{dz} = -\frac{dp}{dy} + \frac{1}{Re}(\frac{d^2v}{dx^2} + \frac{d^2v}{dy^2} + \frac{d^2v}{dz^2}) \tag{3.3}$$

$$\frac{dw}{dt} + \frac{duw}{dx} + \frac{dvw}{dy} + \frac{dw^2}{dz} = -\frac{dp}{dz} + \frac{1}{Re}(\frac{d^2w}{dx^2} + \frac{d^2w}{dy^2} + \frac{d^2w}{dz^2}) \tag{3.4}$$

運動保存則を離散化する．計算領域を x, y, z 方向について $\Delta x, \Delta y, \Delta z$ 長の大きさのスタガード格子で nx, ny, nz 個に分割し，それぞれの格子番号を i, j, k とする．スタガード格子は図 3.1 に表される構造となっており，座標軸に垂直な面の中点に速度を，格子中央に圧力を離散化して定義する．式 (3.2, 3.3, 3.4) に時間項と圧力項には前進差分 (式 (3.5,3.6))，粘性項には中心差分 (式 (3.7))，移流項にはドナーセル法 (式 (3.8)) を適用し，$t + \Delta t$ 時刻の速度 $u^{t+\Delta t}, v^{t+\Delta t}, w^{t+\Delta t}$ を求める．ただし，$\xi = x, y, z$ のとき $\phi = u, v, w$ とする．

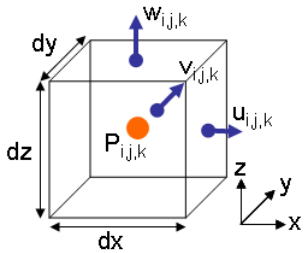


図 3.1: スタガード格子

$$\frac{d\phi}{dt} = \frac{\phi^{t+\Delta t} - \phi^t}{\Delta t} \tag{3.5}$$

$$-\frac{dp}{d\xi} = \frac{-p_{(\xi+\Delta\xi)} - p_\xi}{\Delta t} \tag{3.6}$$

$$\frac{1}{Re}(\frac{d^2\phi}{dx^2} + \frac{d^2\phi}{dy^2} + \frac{d^2\phi}{dz^2}) = \frac{1}{Re}\{\frac{1}{\Delta x^2}(\phi_{x+\Delta x} - 2\phi_x + \phi_{x-\Delta x}) + \frac{1}{\Delta y^2}(\phi_{y+\Delta y} - 2\phi_y + \phi_{y-\Delta y}) + \frac{1}{\Delta z^2}(\phi_{z+\Delta z} - 2\phi_z + \phi_{z-\Delta z})\} \tag{3.7}$$

ドナーセル法は，変化の激しい現象において安定性を維持する風上差分のひとつであり，風上の値をドナー，風下の値をアクセプターとして扱い，数値的に不安定となることを防ぐ手法である．移流項は式 (3.8) のように表される．

$$\frac{d\phi\theta}{d\xi} = \frac{\phi_e\theta_e - \phi_f\theta_f}{\Delta\xi} \tag{3.8}$$

ただし，$\theta = u, v, w$ として

$$\phi_e = (\phi_\xi + \phi_{\xi+\Delta\xi})/2, \qquad \phi_f = (\phi_{\xi-\Delta\xi} + \phi_\xi)/2 \tag{3.9}$$

$$\theta_e = \begin{cases} \theta_\xi & (\phi_e \geq 0), \\ \theta_{\xi+\Delta\xi} & (\phi_e < 0) \end{cases} \qquad \theta_f = \begin{cases} \theta_{\xi-\Delta\xi} & (\phi_f \geq 0), \\ \theta_\xi & (\phi_f < 0) \end{cases} \tag{3.10}$$

ここで求められる速度 $u^{t+\Delta t}, v^{t+\Delta t}, w^{t+\Delta t}$ は連続の式を満たしていないため，速度予測子として扱われ，続く反復計算によって連続の式を満たすまで圧力と速度を修正する．格子 (i, j, k) における発散 $D_{i,j,k}$ を求める．

$$D^{t+\Delta t}_{i,j,k} = \frac{1}{\Delta x}(u^{t+\Delta t}_{i,j,k} - u^{t+\Delta t}_{i-1,j,k}) + \frac{1}{\Delta y}(v^{t+\Delta t}_{i,j,k} - v^{t+\Delta t}_{i,j-1,k}) + \frac{1}{\Delta z}(w^{t+\Delta t}_{i,j,k} - w^{t+\Delta t}_{i,j,k-1}) \tag{3.11}$$

圧力の補正量を求める．

$$\Delta P^s_{i,j,k} = -\frac{\omega D^s_{i,j,k}}{2\Delta t(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2})} \tag{3.12}$$

ここで，ω は加速係数 $(1 \leq \omega \leq 2)$ であり，$\omega = 1.6$ が一般的に使われる．また，s は反復

回数である．続いて圧力と速度の修正計算を行う．

$$p^{s+1}_{i,j,k} = p^{s}_{i,j,k} + \Delta P^{s}_{i,j,k} \tag{3.13}$$

$$u^{s+1}_{i,j,k} = u^{s}_{i,j,k} + \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta x} \tag{3.14}$$

$$u^{s+1}_{i-1,j,k} = u^{s}_{i-1,j,k} - \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta x} \tag{3.15}$$

$$v^{s+1}_{i,j,k} = v^{s}_{i,j,k} + \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta y} \tag{3.16}$$

$$v^{s+1}_{i,j-1,k} = v^{s}_{i,j-1,k} - \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta y} \tag{3.17}$$

$$w^{s+1}_{i,j,k} = w^{s}_{i,j,k} + \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta z} \tag{3.18}$$

$$w^{s+1}_{i,j,k-1} = w^{s}_{i,j,k-1} - \frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta z} \tag{3.19}$$

以上を計算領域全ての格子に対して適用し，連続の式が満たされるまで反復計算を繰り返す．図 3.2 に HSMAC 法の解法の流れを示す．

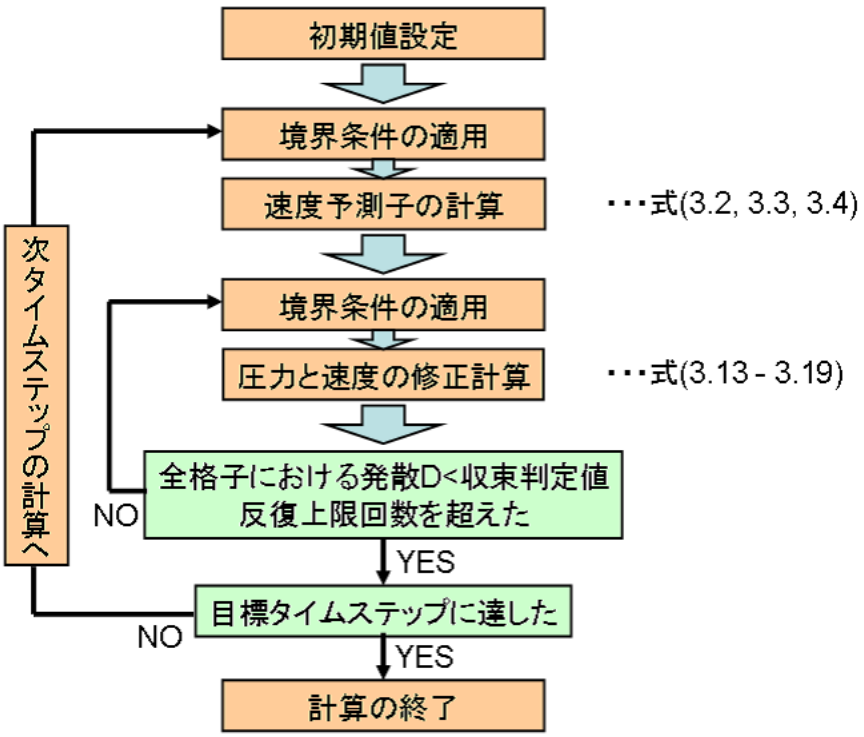


図 3.2: HSMAC 法の解法の流れ

第4章　システムの概要

本システムは，Master と複数の Worker とをネットワーク上に配置した分散計算環境上におけるリアルタイム可視化システムである．リアルタイム可視化は，シミュレーションに合わせて可視化を行うものである．本システムでは複数の Worker で計算結果が得られるたびに順次計算結果を Master へ転送しリアルタイム可視化を行う (図 4.1)．

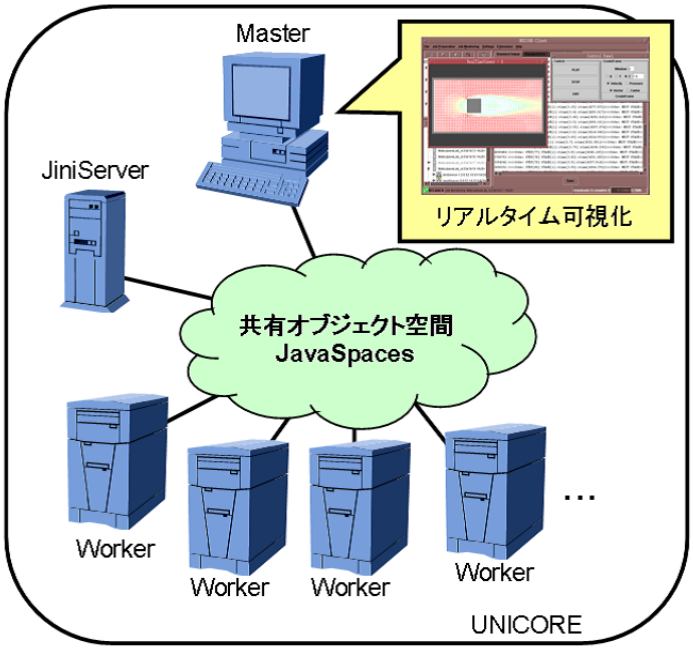


図 4.1: システム概念図

図 4.2 は本システムの階層図である．複数のターゲットマシン (図 4.2(1)) を用いて，分散計算環境を構築管理するために UNICORE を利用する (図 4.2(2))．シェルスクリプトで構成されたジョブによって JiniServer，Master，Worker を UNICORE 上で実行する．ここで，Master はシステム全体の進行を管理し，計算結果を可視化する利用者端末の役割を持つ計算機である (図 4.2(3))．Worker は Master から受け取った担当分の数値流体シミュレーションを実行し，計算データを JavaSpaces へ保管し，可視化データを Master へ返す計算機である (図 4.2(3))．JiniServer は Lookup サービスと JavaSpaces サービスが立ち上がった，コミュニティを構成し Jini 技術を利用するためのサーバである (図 4.2(4))．

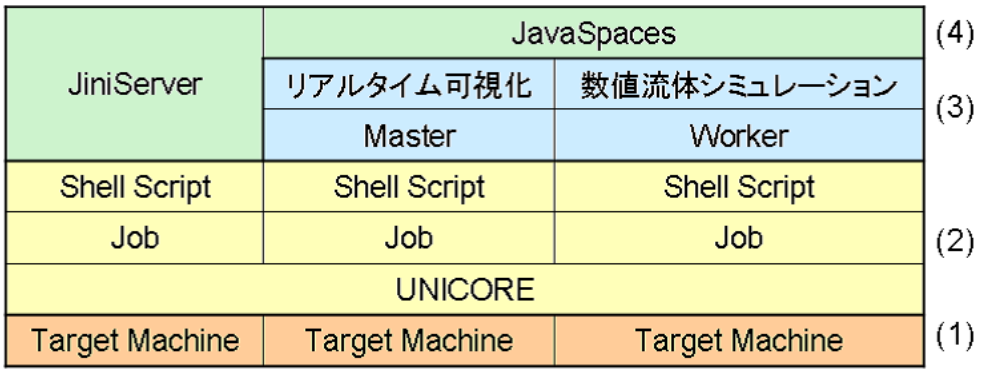


図 4.2: システム階層図

本システムの特徴は以下に挙げられる．

- JavaSpaces へ書き込みを効果的に行うための計算データの分割
- 速度予測子の計算および圧力と速度の反復計算についてループ分割
- JavaSpaces へ置かれる共有オブジェクトである Entry クラス
- JavaSpaces をタスクバッグとして利用した負荷分散
- シミュレーションに合わせて可視化を行うリアルタイム可視化
- 計算と可視化を一時停止・再開できるステアリング機能

以下の節でこれら特徴について述べたあと，JavaSpaces での Entry クラスの流れをまじえたシステム全体の流れを説明する．

4.1 計算データの分割

３次元数値流体シミュレーションでは，多くの計算データを格納するメモリ空間が必要である．ここで Jini 技術を用い，各 Worker のメインメモリを統合した共有オブジェクト空間 JavaSpaces を構築し，ここに計算データを格納する．しかし，JavaSpaces は物理的には各 Worker ごとにメモリ空間が区切られているため，全計算データをひとかたまりに JavaSpaces へ格納しても，その実体は Master もしくはある Worker 単体のメインメモリに置かれることになり，全計算データが格納しきれない場合がある．これを避けるため，計算データを複数に分け，各 Worker のメインメモリに分散して格納する．この時，各 Worker の JavaSpaces に格納される計算データの量は，計算データの数と Worker の台数が等しくない限り，Worker の計算処理能力に比例する．計算データの分割方法は，ループ分割の

大きさごとに区切り，ループ分割計算の際，必要な領域の計算データのみを JavaSpaces から取り出せるようにする．

4.2 ループ分割

式 (3.2, 3.3, 3.4) による速度予測子の計算と，式 (3.13-3.19) による圧力と速度の修正計算を，i に関してそれぞれループ分割を行い，各 Worker へ計算処理を分担する．ここで計算領域は，ループ分割数と等しい N 個に分割され，さらにそれは計算データ本体と計算データ境界値に細かく分割され，JavaSpaces に格納される．図 4.3 は，$i = 0$〜L までの計算データを N 個に分割した様子を示している．ここで分割長 $M = L/N$ である．

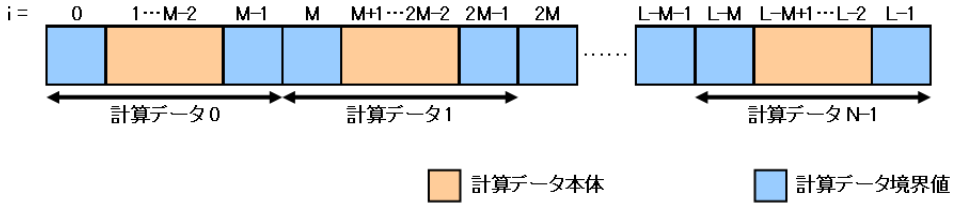


図 4.3: 計算データの構成

速度予測子のループ分割計算の模式図を図 4.4 に，圧力と速度の修正計算のループ分割の模式図を図 4.5 に示す．Worker は JavaSpaces から計算データを取り出し，速度予測子の計算もしくは圧力と速度の修正計算をし，計算結果を新しい計算データとして JavaSpaces に書き込む．

Worker が計算を行う前に，境界条件の適用を行う必要がある．このとき，流入面と流出面を除いた位置 i を計算する際は $i-1, i+1$ の値が必要となるため，任意の計算データ Q を計算するには，その計算データに隣接する計算データ $Q-1, Q+1$ の計算データ境界値 $i = QM-1, (Q+1)M+1$ が必要となる．ここで，計算領域の末端である $i = 0, L-1$ を除いた計算データ境界値 $i = QM-1, (Q+1)M+1$ は，隣接する計算データを計算する際にも必要となるため，JavaSpaces から計算データ境界値の取り出しは 2 度行わなければならない (図 4.4 (1))．このため，計算データ境界値にフラグを設け，フラグが立てられていない計算データ境界値を JavaSpaces から取り出した際は，計算データ境界値にフラグを立て，再び計算データ境界値を JavaSpaces へ書き込み直す．これによって，計算データ境界値を JavaSpaces から 2 度取り出すことができる．

圧力と速度の修正計算の際，一反復計算のうちに，i に関して式 (3.15) で第一の修正計算，式 (3.14) で第二の修正計算が行われる．しかし i に関して，計算データごとにループが分割されているため，計算データにまたがって二度の修正計算を行わなければならない．ここで，第二の修正計算に関して，修正する前の値を第一の修正計算した値として参照し，反復計算を繰り返すことによって，その影響を緩和する．

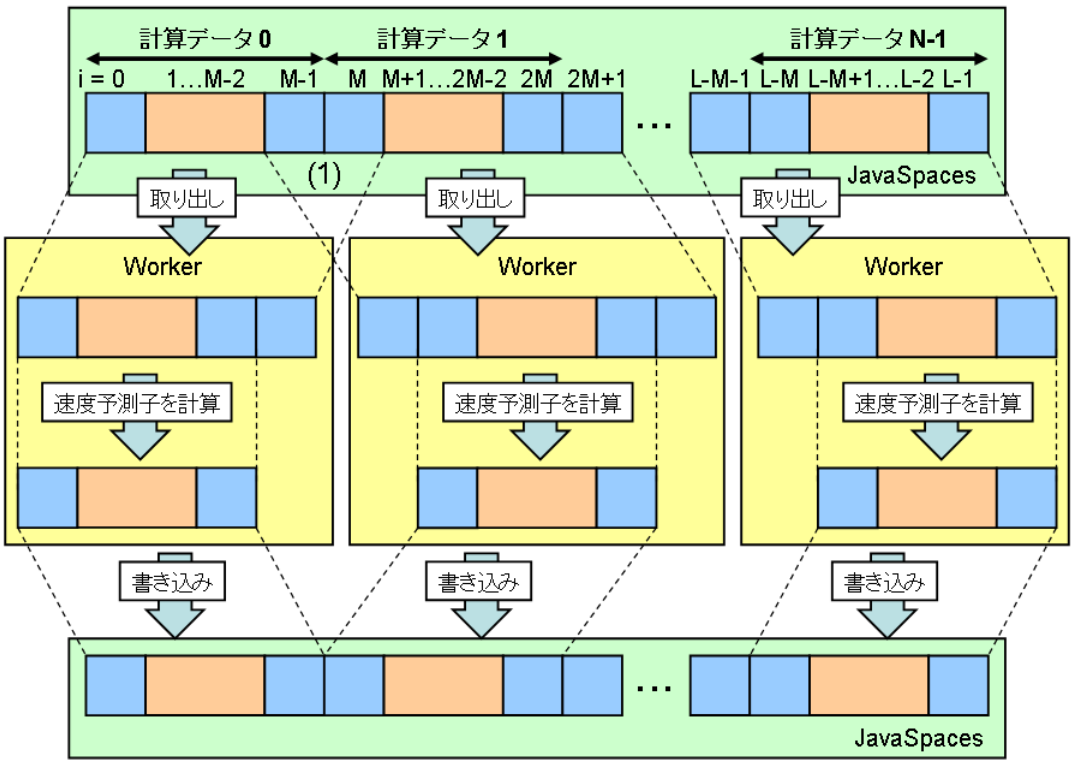


図 4.4: 速度予測子のループ分割計算

$i = QM$ ごとに修正計算の重複箇所を設け，計算データ $Q-1$ では $i = (Q-1)M$ から $i = QM-1$ までの修正計算を行い，さらに $i = M$ について第一の修正計算を行う．このとき $i = M$ に関する修正計算は，式 (4.1) によって修正量のみを記録する．(図 4.5 a) 計算結果を JavaSpaces へ再び計算データとして書き込む．修正量の値は，式 (3.14) より以下のように与えられる．

$$u^{s+1}_{i-1,j,k} = -\frac{\Delta t \Delta P^{s}_{i,j,k}}{\Delta x} \tag{4.1}$$

計算データ Q では，$i = QM$ について第二の修正計算を，$i = QM+1$ から $i = (Q+1)M-1$ までの修正計算をそれぞれ行い，JavaSpaces へ計算結果を書き込む (図 4.5 b)．JavaSpaces には同じ $i = QM$ に関する計算データ境界値が，図 4.5 の a と b のように二つ存在することになる (図 4.5 a,b)．次反復計算もしくは速度予測子の計算が行われる際，計算を行う前に $i = QM$ に関する二つの計算データ境界値を JavaSpaces から取り出し，二つの値を足し合わせることによって，$i = QM$ に関する修正計算が完了する (図 4.5 (2))．

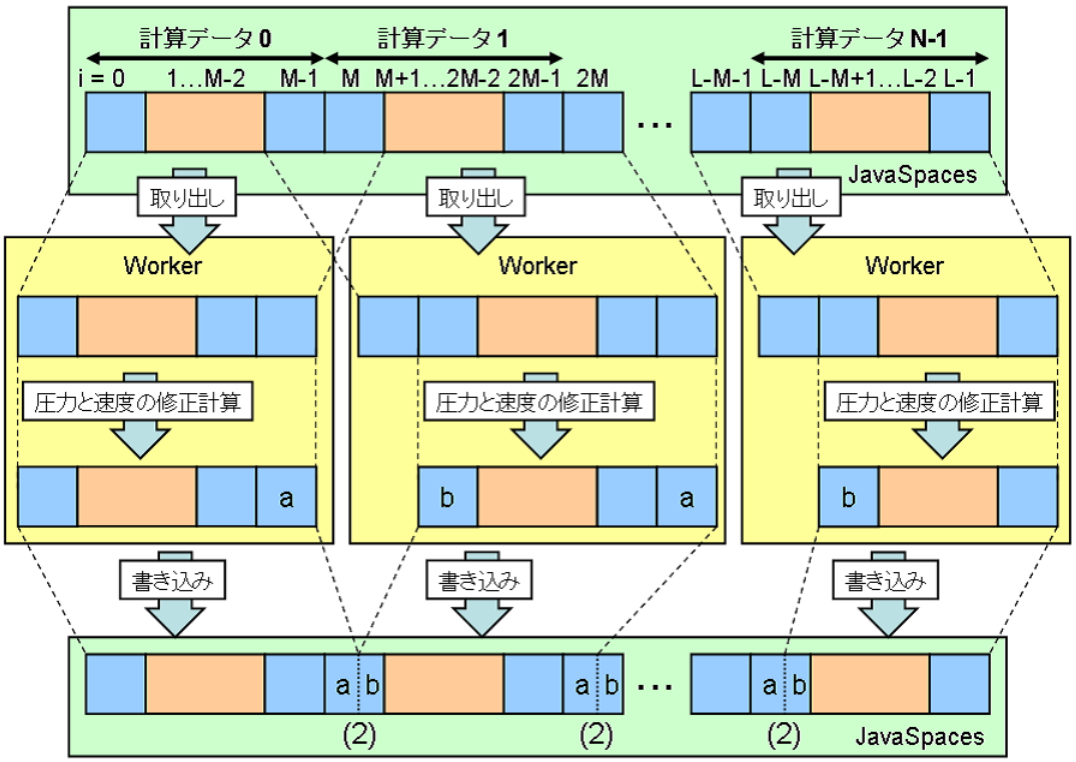


図 4.5: 圧力と速度の修正計算のループ分割

4.3 負荷分散

負荷分散は，ネットワークに接続された複数の計算機でタスクを実行する際，計算機性能に応じて負荷を均衡化することによって，それぞれの計算機のプロセッサ利用率を高めることを目的とした技術である．

タスクを実行する前に各計算機の処理能力を測定し，これをもとに計算機に割り振るタスクの大きさを設定し，すべての計算機に均等な負荷が与える手法を静的負荷分散という．タスクの実行中に計算機台数の変更や，計算機に別な負荷が与えられて処理能力が変化した場合に対処することができない．

一方，動的負荷分散がある．これは，タスクを計算機台数に比べ細かな粒度に分割し，計算機にタスクを一つずつ実行させる手法である．計算機が高速であるほど，結果的にタスクの実行回数は多く，計算機負荷が与えられる．タスクの粒度が細かいほどタスクのアクセスが頻繁になり，通信時間の影響が大きくなるため，低速な計算機環境には不向きであるが，タスクの実行中に計算機台数や処理能力に変化が生じた場合においても適切に負荷が分散される．

本研究では，計算資源の構成を変更する場合や通信障害が発生した場合において柔軟に対応をする Jini 技術の特徴に適した後者を採用し，JavaSpaces をタスクバッグとして計算機台数に比べ粒度の細かいタスクを配置し，異機種の計算環境の場合には処理能力に応じて実行するタスクの量が調整されることによって動的負荷分散を行う．

4.4 Entry クラス

本システムでは，Entry クラスを拡張した以下のオブジェクトを利用する．これらオブジェクトを，JavaSpaces を介して，Master と Worker，もしくは Worker と Worker の通信を行う．

- 計算データオブジェクト
  Fig.4.3 における計算データ本体が格納される．

- 境界値オブジェクト
  Fig.4.3 における計算データ境界値が格納される．

- タスクオブジェクト
  Master から Worker へ計算開始を知らせるオブジェクトである．計算データの番号が格納され，速度予測子の計算もしくは修正計算のどちらを行うかといった計算命令も記述される．これによって，Master はどの計算データについて速度予測子もしくは修正計算のどちらを計算開始させるかを Worker へ命令できる．さらに可視化に関する情報が格納される．

- 計算終了オブジェクト

Worker から Master へ計算終了を知らせるオブジェクトである．計算データの番号が格納され，計算データの計算終了を通知する．

- 可視化オブジェクト
  Worker から Master へ送られる，可視化断面における計算データが格納されたオブジェクトである．

4.5 リアルタイム可視化

本研究におけるリアルタイム可視化は，シミュレーションに合わせて可視化を行うものである．タイムステップ毎もしくは圧力と速度の修正計算によって数値流体シミュレーションの計算結果が得られる度に，可視化システムへ転送し可視化を行うことによって，リアルタイム可視化を実装した．これによってユーザは，数値流体シミュレーションが正しく行われているか確認することや，計算の進行状況やシミュレーションの結果をいち早く知ることができる．しかしながら，リアルタイム可視化による数値流体シミュレーションへの負荷や計算の遅延といった影響が及ぶことのないことが重要である．

Java 言語では，GUI(Graphical User Interface) 開発のための API(Application Program Interface) セットとして Swing が用意されており，Swing を用いることによって，プラットフォームに依存しない描画処理が可能である．Swing は描画処理専用のスレッドが可視化処理を請け負うため，可視化処理と続く処理内容がマルチスレッドで実行され，負荷の大きい描画処理が生じても，可視化システムの処理に遅延が生じることはない (図 4.6(a))．また，描画処理専用のスレッドは，汎用性や拡張の簡易化のためにシングルスレッド設計となっており，描画処理専用のスレッドを複数生成することや，複数の描画命令を同時に受け付けることはできない (図 4.6(b))．Java 言語かつ Swing を用いて可視化を行うことによって，可視化に時間がかかる場合においても，次の可視化要求が割り込みを行い可視化システムに障害が起こることなく，可視化システムを運用することができる．

Swing によって，描画処理とそれに続く処理がマルチスレッドで実行される．しかしどちらも単一の計算機によって動作するため大きな描画負荷が与えられた場合，マルチスレッドで実行される描画処理に続く処理にも影響を与える可能性がある．図 4.7 は 3 反復の修正計算までのタイムステップ計算の流れと可視化のタイミングを示している．図 4.7(1) はシミュレーションと可視化を逐次処理したものであり，可視化に大きな描画負荷が与えられたとき，システム全体の遅延となる．本システムでは，Worker がシミュレーションを行っている間に，Master は前タイムステップもしくは前反復計算によって得られた結果を可視化することで，シミュレーションと可視化を同時に行い描画負荷からの影響を最小限にした (図 4.7(2))．

２次元計算よりも多くの計算データを扱うため，可視化の際に 3 次元計算データすべてを転送させると，通信の過負荷や可視化クライアントのメモリオーバフロー，描画処理の負荷の増加によって，シミュレーションのリアルタイム性が得られなくなる．そこで，３

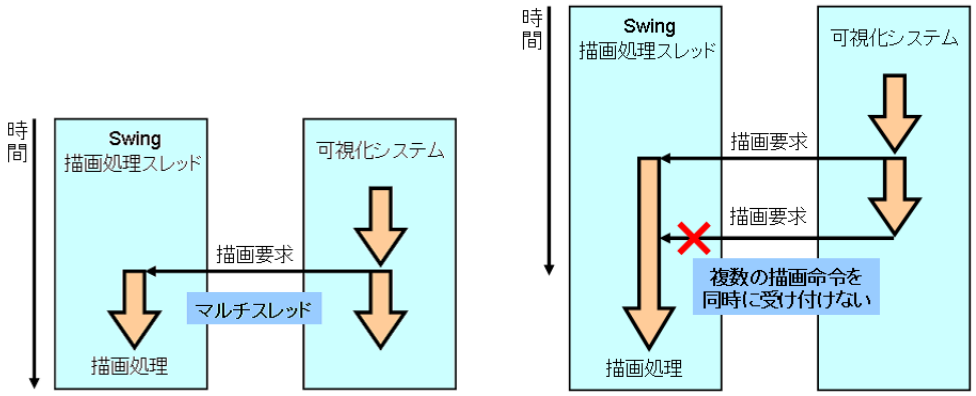


図 4.6: Swing の描画処理

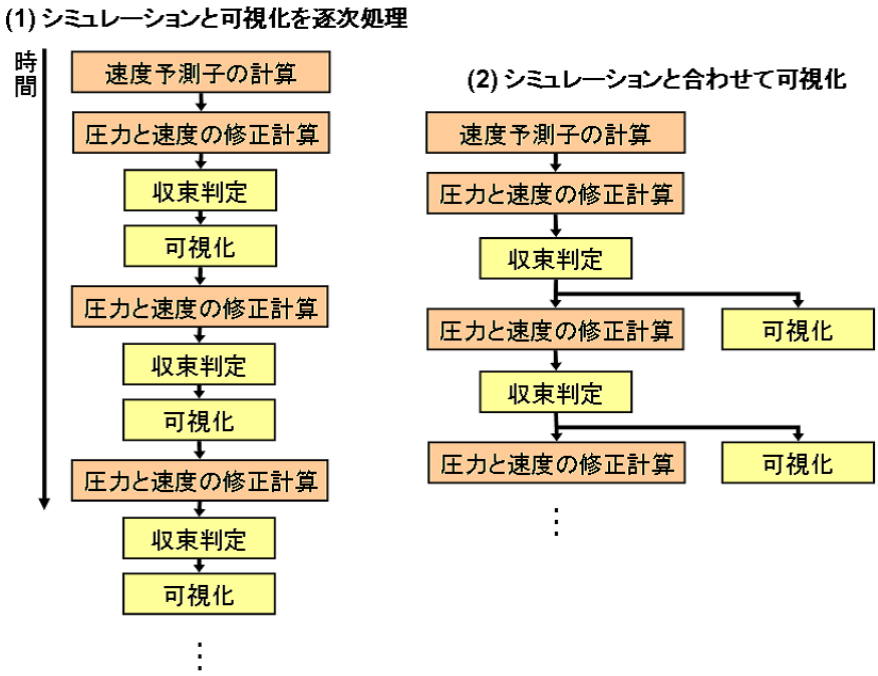


図 4.7: 可視化のタイミング

次元データを立体的に表示させるのではなく，任意の座標軸に対して水平方向に切った断面を可視化することによって，可視化に必要な計算データを減らし，ネットワーク負荷，描画処理の負荷を軽減させる．また，可視化のタイミングに関しても，これらの負荷が影響しない程度のタイムステップ毎もしくは修正計算毎に行うものとする．

可視化機能としては以下を可視化の条件として設定できる．ただし，圧力はスカラー値であり，ベクトルとして表現できないため，圧力ベクトル図としては可視化できない．

- 可視化の断面となる座標軸 (x,y,z)
- 指定された座標軸における断面位置
- 速度もしくは圧力を選択できる可視化内容
- ベクトル図もしくは分布図を選択できる可視化方法

タスクオブジェクトには，上記の可視化の条件が格納される．可視化断面の変更・追加を行うために可視化の条件は配列として格納される．Worker はタスクオブジェクトを読み込み可視化断面を特定し，計算データから抽出を行い，設定された可視化断面の数だけ可視化オブジェクトへ格納し，JavaSpaces を介して Master へデータを送る．Master は，可視化オブジェクトに記述された領域番号の順に可視化断面の計算データを配列に格納し，可視化断面のデータを構築する．計算領域すべての可視化オブジェクトが得られ，可視化断面のデータが完成したとき可視化を行う．

タイムステップ毎もしくは反復計算毎に可視化は同期を取ることになるが，可視化直前に可視化画面の追加や削除が行われた場合，同期がとれていない可視化画面が表れることになり，誤った情報を可視化する場合や Worker から可視化のデータが渡されていないのに可視化を強行しシステムがダウンする場合がある．可視化直前に以下のチェックを行うことで，これら不具合を防ぐ．

- 可視化オブジェクトは全計算領域分すべて得られ，可視化させる計算データには不足部分は存在しないか
- タスクオブジェクトで記述した可視化の条件と，Worker から得られた可視化の情報は適合しているか
- 可視化画面が存在するか

4.6 ステアリング操作

リアルタイム可視化に関する補助的な機能としてステアリング機能は存在する．ステアリング操作は，計算と可視化を一時停止・再開できる機能であり，シミュレーションにおける物理量の変化が顕著である場合，計算を一時停止させ計算結果を詳細に確認すること

ができる．また計算対象が小さいが計算領域が広い場合，全体を可視化すると計算対象のまわりの挙動を把握するのが困難となる．よって計算対象付近の挙動を詳細に確認するため，可視化画面の拡大・縮小・移動の操作を実装した．

ステアリング操作による計算と可視化の一時停止は，タイムステップ毎もしくは修正計算毎のタイミングで行われる．Master がタスクオブジェクトを JavaSpaces へ書き込まないことで，Worker はタスク待ちの状態となり，計算と可視化が一時停止される．また，タスクオブジェクトの書き込みを許すことによって，タスクが実行され，計算が再開される．一時停止中における計算データは，すべて JavaSpaces に保管されることになり，これを取り出すことによって，計算データの損失が行われることなく計算を再開できる．

可視化画面の拡大・縮小・移動の操作は Master に実装されており，可視化ウィンドウにキーボード入力に関するイベントを付加しキーボードからの入力を認識することで，ベクトル図のベクトル長やコンター図の解像度，可視化位置を変更する．

4.7　システムの流れ

Jini 技術より提供されるサービスを利用するには，Jini サービスの起動を行わなくてはならない．さらに Master と複数の Worker を立ち上げなければならない．これらの起動は，全ての計算機の認証操作を行う必要があり，多数の計算機を用いて計算を行う場合煩雑となる．ここで，全ての計算機に UNICORE を実装することによって一度の認証操作で全ての計算機を利用することができ，システムの起動が簡易化できる．本システムでは，煩雑となる JiniServer，Master，Worker の起動をシェルスクリプトに記述し，これをジョブとして Vsite へ投入することによって UNICORE と Jini 技術を連携させた．以下に，UNICORE と Jini 技術を用いた本システムの実行の過程を示す．

1. UNICORE の起動
   本システムを構成する JiniServer，Master，Worker の計算機を Vsite として UNICORE を起動する (2.2 節 UNICORE の起動).

2. JiniServer の起動
   UNICORE Client から，JiniServer をジョブとして Vsite へ投入する．ここで，JiniServer は Jini 技術を利用するために必要な Lookup サービスと JavaSpaces サービスを起動する．

3. Worker の起動
   UNICORE Client から，Worker をジョブとして Vsite へ投入する．Worker は Lookup サービスを通じて JavaSpaces サービスを発見し，JavaSpaces へタスクオブジェクトが書き込まれるまで待機する．

4. Master の起動
   UNICORE Client から，Master をジョブとして Vsite へ投入する．Master は Lookup

サービスを通じて JavaSpaces サービスを発見し，ユーザからの計算開始のステアリング操作を待つ (図 4.8).

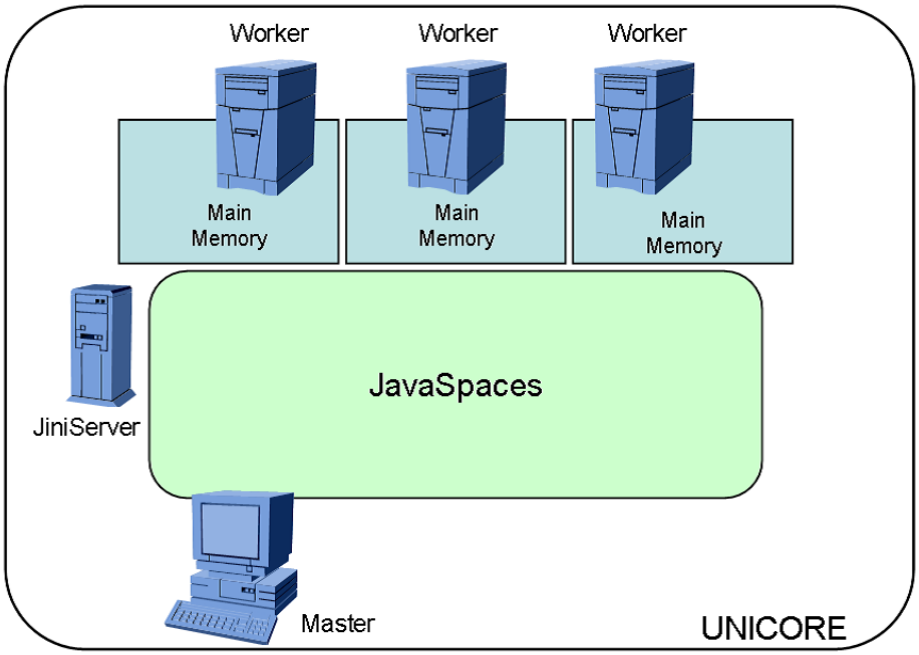


図 4.8: システムの実行 (1)

5. 計算開始命令
   ユーザから計算開始のステアリング操作を受け取った時，領域番号が振られたタスクオブジェクトを JavaSpaces へ書き込む．このとき，Master は初回に限り計算データを生成，初期化し，計算データオブジェクトと境界値オブジェクトとして JavaSpaces へ書き込む (図 4.9).

6. タスクオブジェクトの受け取り
   Worker は JavaSpaces からタスクオブジェクトを一つ取り出す．ここで取り出されるタスクオブジェクトは一意に決まっておらず，ランダムである．タスクオブジェクトに振られた領域番号を読み，同じ領域番号を持つ計算データを JavaSpaces から取り出す (図 4.10)．修正計算後の JavaSpaces では，修正計算が未完了の二つの境界値オブジェクトが存在する (図 4.5 a,b)．この場合，それらを JavaSpaces から取り出し，この二つの値を足し合わせ，修正計算を終了させる (図 4.5 (2)).

7. 境界値オブジェクトの再分配
   計算データを計算するには隣り合う計算データ境界値が必要となる．計算領域端を除く計算データ境界値が格納された境界値オブジェクトを JavaSpaces へ書き込み，隣接する計算データの境界値オブジェクトを JavaSpaces から取り出す (図 4.11).

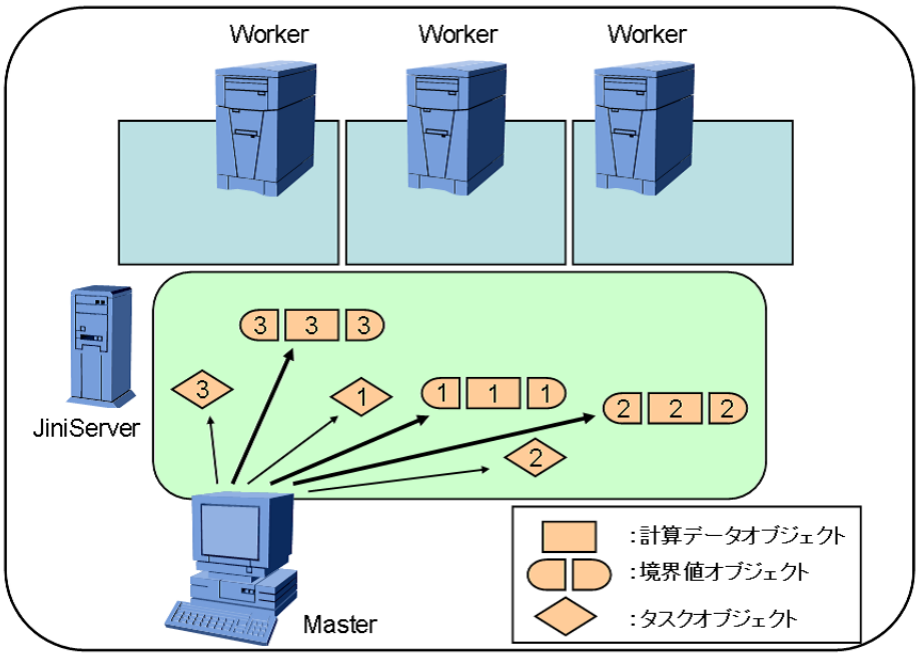


図 4.9: システムの実行 (2)

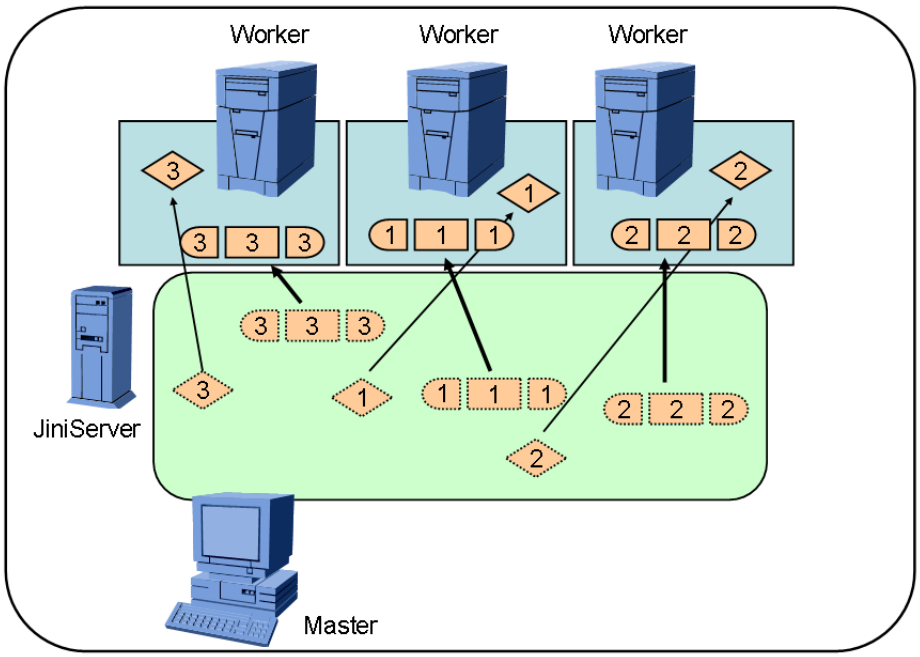


図 4.10: システムの実行 (3)

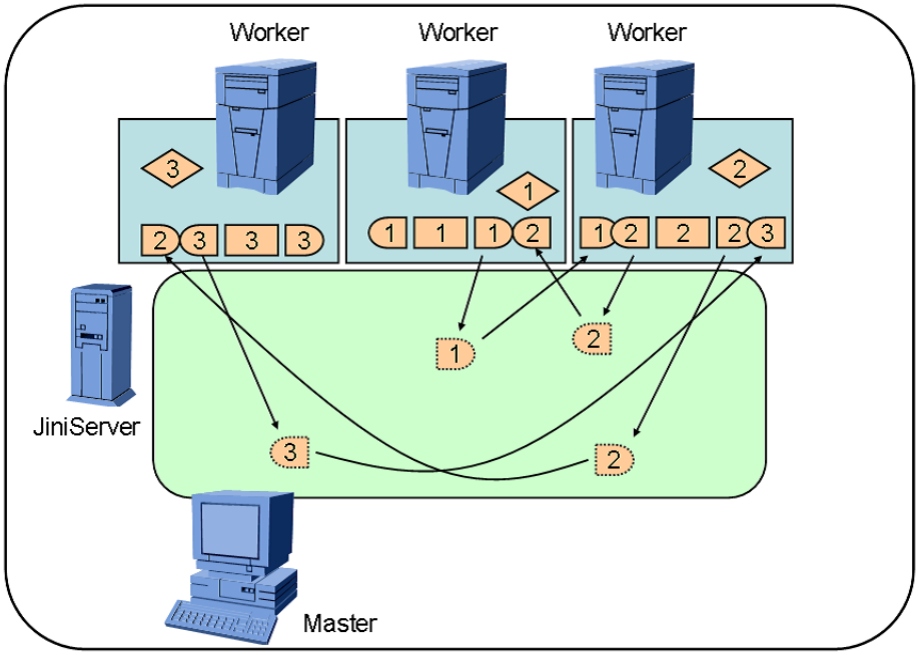


図 4.11: システムの実行 (4)

8. 可視化データの収集と可視化
   修正計算後であった場合，手順 6 で，修正計算を完了させたのち，可視化を行う．Worker はタスクオブジェクトの可視化に関する情報を読み，可視化に必要な計算データを抽出し，可視化オブジェクトとして JavaSpaces へ書き込む．Master はこれらを JavaSpaces から取り出し，可視化データを統合し可視化を行う (図 4.12).

9. 数値流体シミュレーション
   タスクオブジェクトに記述された計算命令によって，格納された計算データに対して速度予測子の計算 (図 4.4) もしくは圧力と速度の修正計算 (図 4.5) を行う．得られた計算結果は，再び計算データに格納され，JavaSpaces へ書き込まれる (図 4.13). ただし，圧力と速度の修正計算を行った場合，$i = QM$ ごとに修正計算が未完了の境界値オブジェクトが存在するが，手順 6 で境界値オブジェクトを再び回収したときに修正計算を完了する (図 4.5 (2)).

10. 計算の完了
    Worker はタスクオブジェクトで依頼された計算を完了すると計算を終了した領域番号を記述した計算終了オブジェクトを JavaSpaces に書き込み，Master はそれを取り出す (図 4.14). 圧力と速度の修正計算の場合は計算完了オブジェクト内に発散 D の値を格納し，Master は収束判定を行う．

11. 判定

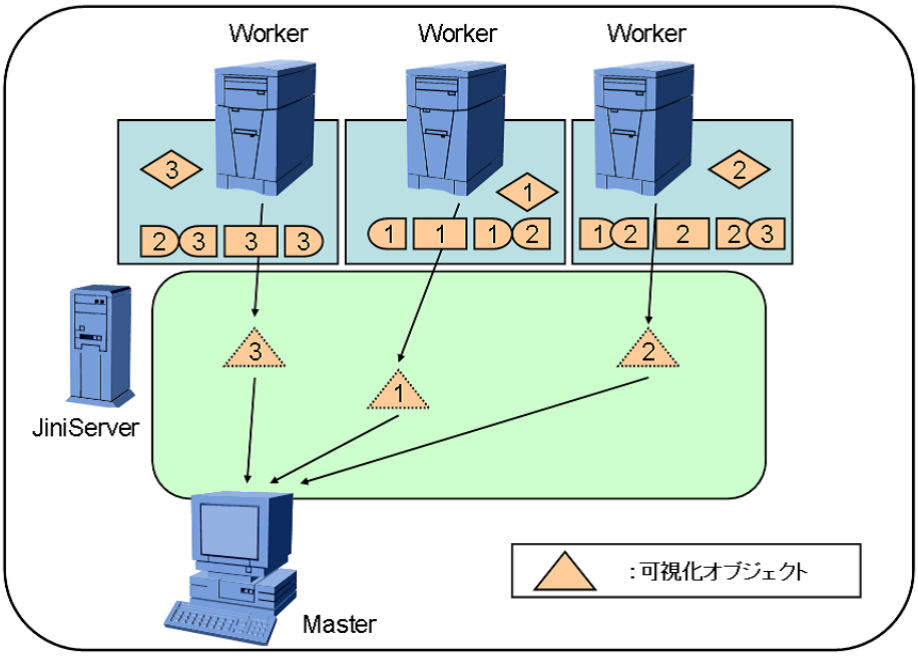


図 4.12: システムの実行 (5)

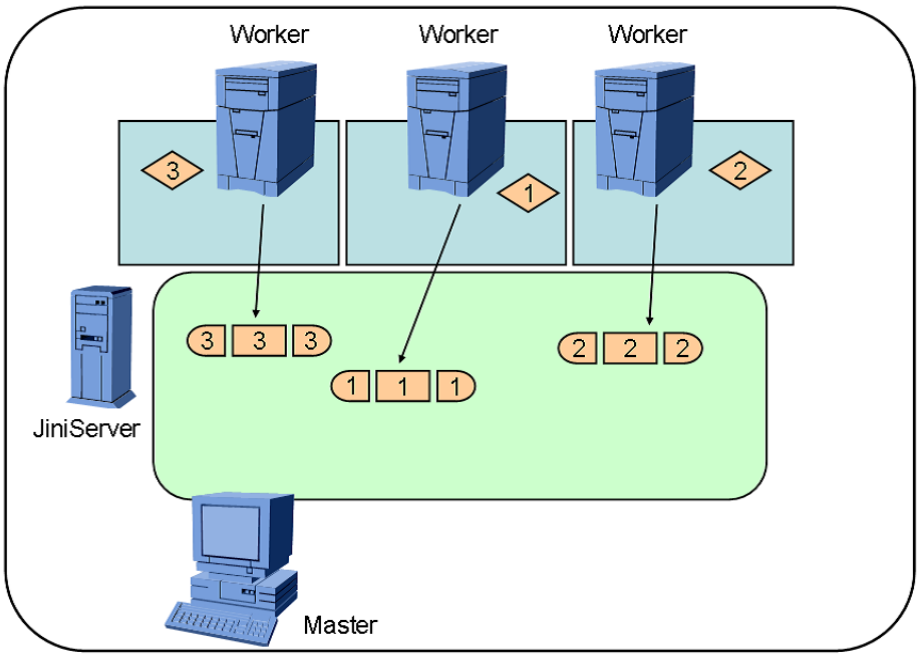


図 4.13: システムの実行 (6)

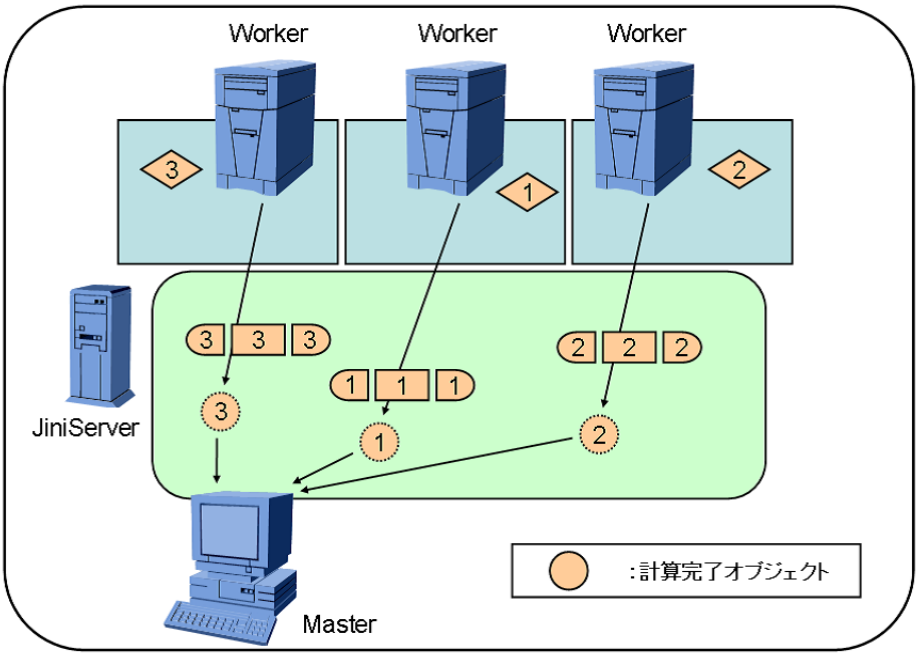


図 4.14: システムの実行 (7)

目標タイムステップに達した場合はここで計算を終了する．引き続き計算を行う場合は手順 5 へ戻る．手順 5 へ戻る前での JavaSpaces の中身は，図 4.15 のように表される．ステアリング機能による一時停止はここで行われ，再開の際は手順 5 へ戻る．

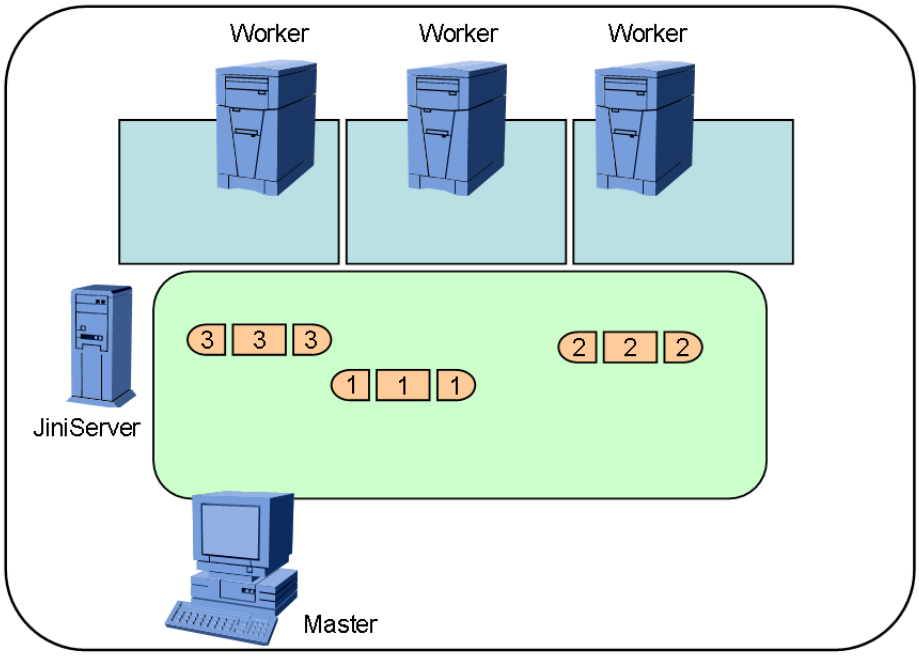


図 4.15: システムの実行 (8)

第5章　数値流体シミュレーションの適用

数値流体シミュレーションの適用として３次元角柱流れ解析を行い，任意の座標軸の断面によるリアルタイム可視化の実現とその可視化断面の変更及び追加の確認，可視化による測定時間への影響と並列化効率の測定，異機種分散計算環境におけるシミュレーションの実行と負荷分散の考察によってシステムの有効性を検討する．

5.1　計算条件

３次元角柱流れに関する計算条件を表 5.1 に，計算領域を図 5.1 に示す．

表 5.1: ３次元角柱流れ解析に関する計算条件

レイノルズ数	1000
領域サイズ	8.0 ×4.0 × 4.0
格子数	64 ×32 × 32
角柱サイズ	1.0 ×1.0 × 3.0
初期条件	角柱を除いて領域全体に速度 $u = 1.0$ を与える
流入条件	x=0 における流入面に一様速度 $u = 1.0$ を与える
境界条件	角柱壁と z=0 における領域底面を non-slip 条件 その他の領域境界を slip 条件を与える

5.2　計算環境

本システムは，表 5.2 に示した計算機性能の計算機で構成されており，JiniServer には WorkStation1，Master には WorkStation2，Worker には WorkStation3 と同様の計算機性能を持つ計算機を配置し，同機種分散計算機環境を構築した．なお，それぞれの計算機は 100Mbps の学内 LAN によって接続されている．

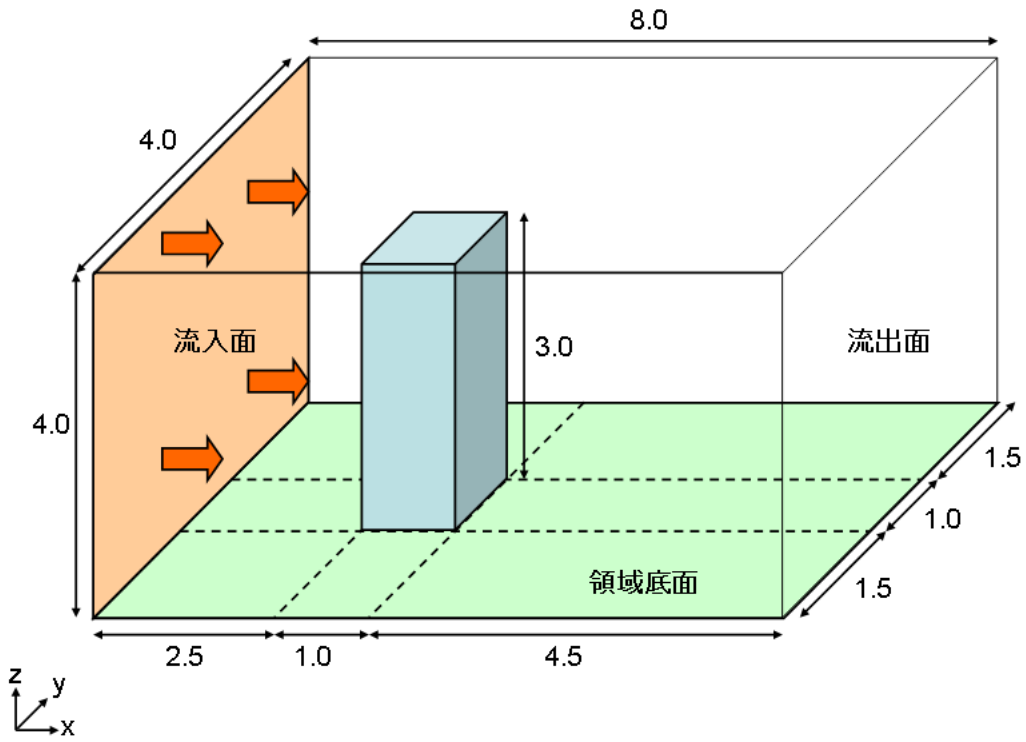


図 5.1: 計算領域

表 5.2: 同機種分散計算環境を構築する計算機の計算機性能

	WorkStation1	WorkStation2	WorkStation3
CPU	SPARC IIIi 1.0GHz	SPARC IIe 550MHz	SPARC IIIi 1.0GHz
MEMORY	512MB	640MB	512MB
OS	Solaris-9	Solaris-9	Solaris-8
	JiniServer	Master	Worker

5.3　結果

システム運用中の様子を図 5.2 に示す．後ろで起動しているウィンドウは UNICORE であり，UNICORE 上で本システムを起動する．画面右の小ウィンドウはステアリング命令と可視化断面の設定を行う操作パネルである．画面左は可視化ウィンドウであり，可視化している計算結果は，$t = 10.0$ における $z = 1.0$ の断面について速度ベクトルを描画したものであり，計算中の様子を示している．

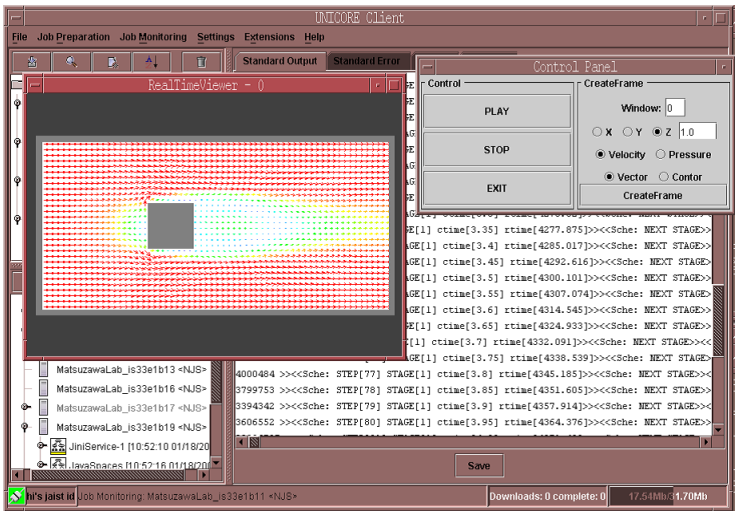


図 5.2: システム運用中の様子

図 5.3 は可視化内容と可視化方法を変更して可視化したものである．画面左上の可視化ウィンドウは速度をベクトル図として可視化したものであり，画面左下の可視化ウィンドウは速度を分布図として可視化したものであり，画面右下の可視化ウィンドウは圧力を分布図として可視化したものである．なお，いずれの可視化ウィンドウも，$t = 0.0$ における $z = 1.0$ の断面について計算中のものである．

図 5.4 は，$t = 0.0$ における $z = 1.0$ の断面に関して速度ベクトル図を可視化したものであり，角柱まわりの現象がより詳細に把握できるよう可視化画面を拡大・移動したものである．

5.3.1　流れ解析の結果

３次元角柱まわりの流れ解析の結果を図 5.5-5.9 に示す．

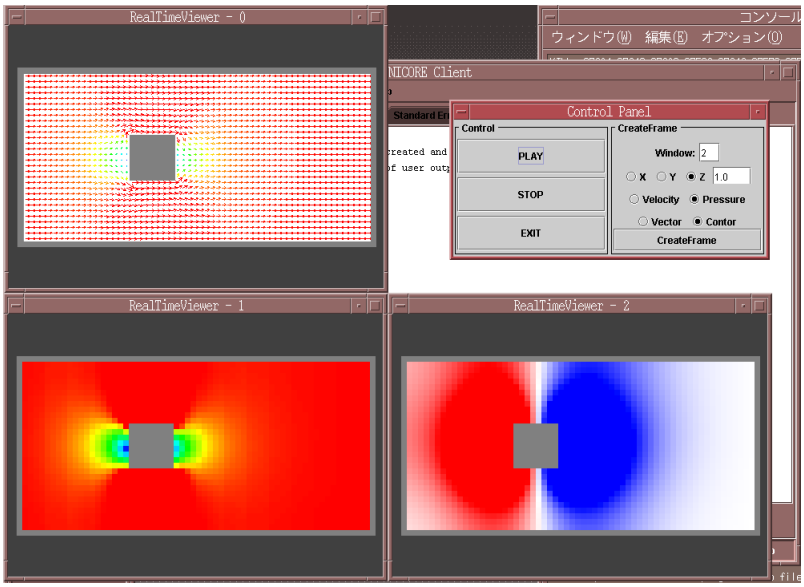


図 5.3: 可視化の表現

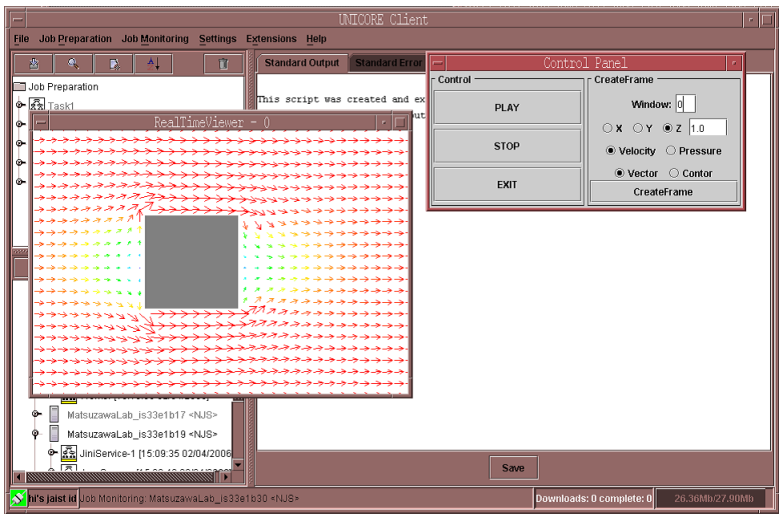
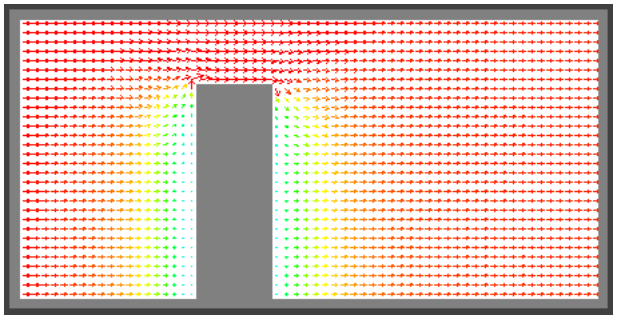
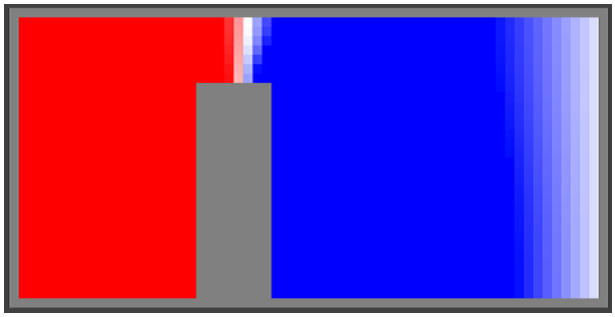


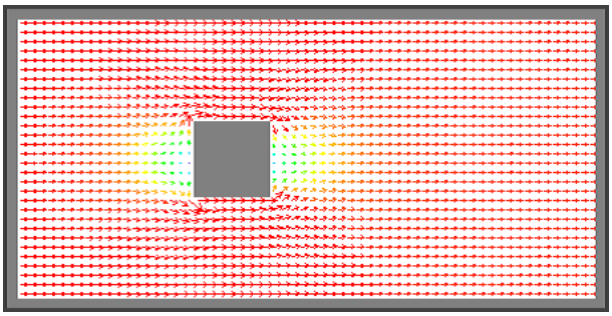
図 5.4: 可視化画面の拡大・移動



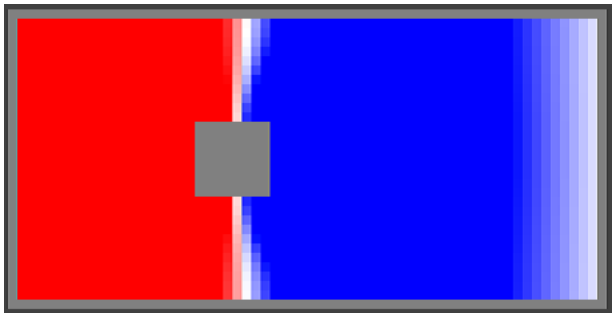
(a) y=2.0 断面における速度ベクトル図



(b) y=2.0 断面における圧力コンター図

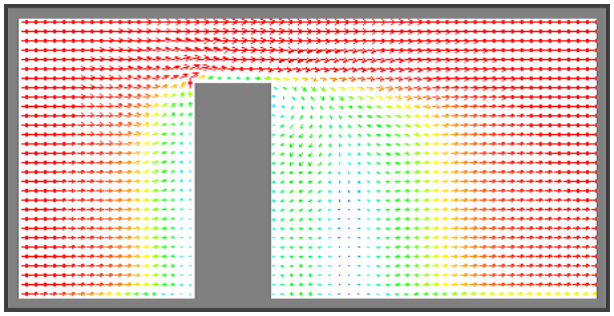


(c) z=1.0 断面における速度ベクトル図

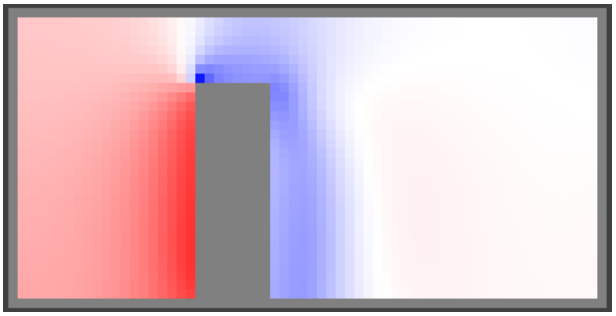


(d) z=1.0 断面における圧力コンター図

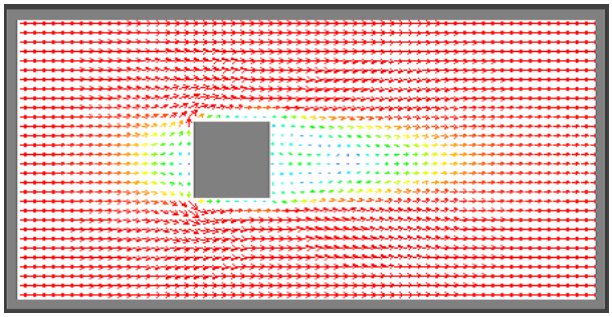
図 5.5: 計算結果 (t=0.0)



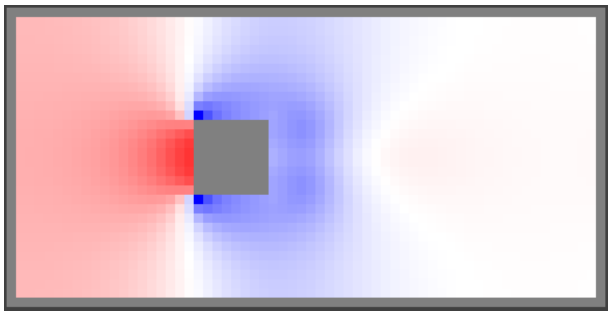
(a) y=2.0 断面における速度ベクトル図



(b) y=2.0 断面における圧力分布図

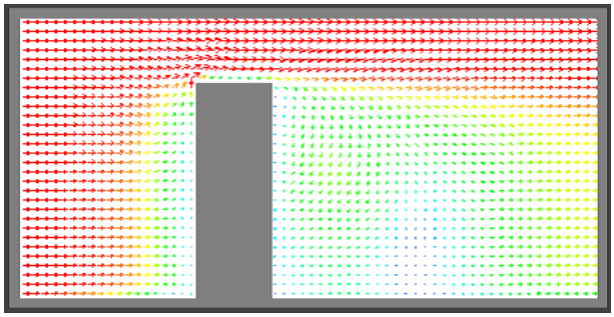


(c) z=1.0 断面における速度ベクトル図

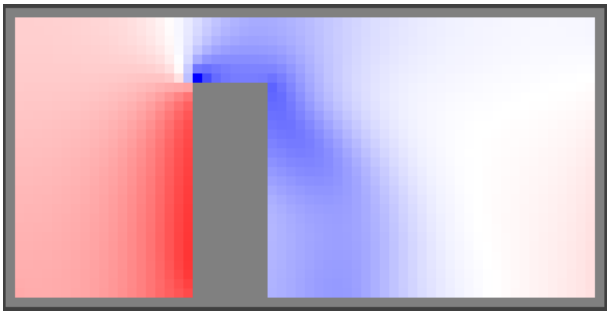


(d) z=1.0 断面における圧力分布図

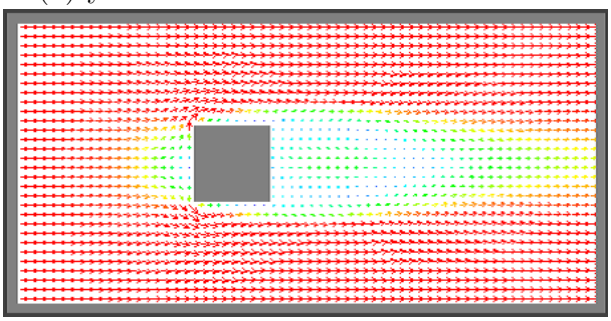
図 5.6: 計算結果 (t=2.5)



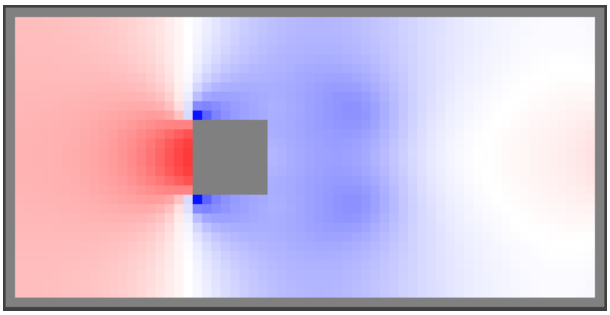
(a) y=2.0 断面における速度ベクトル図



(b) y=2.0 断面における圧力分布図

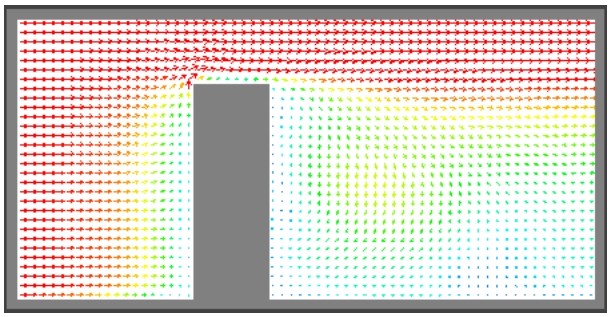


(c) z=1.0 断面における速度ベクトル図

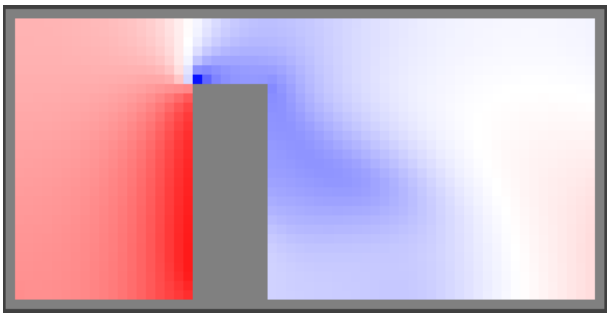


(d) z=1.0 断面における圧力分布図

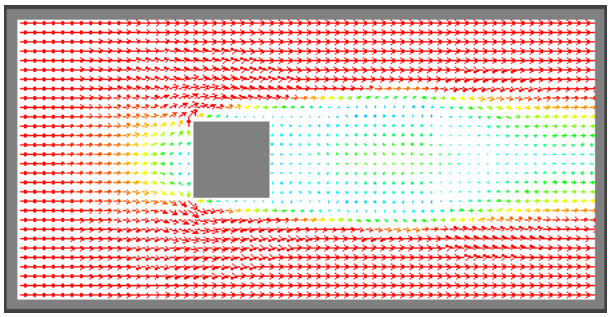
図 5.7: 計算結果 (t=5.0)



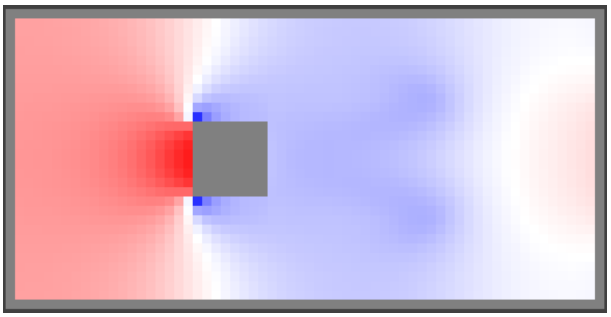
(a) y=2.0 断面における速度ベクトル図



(b) y=2.0 断面における圧力分布図

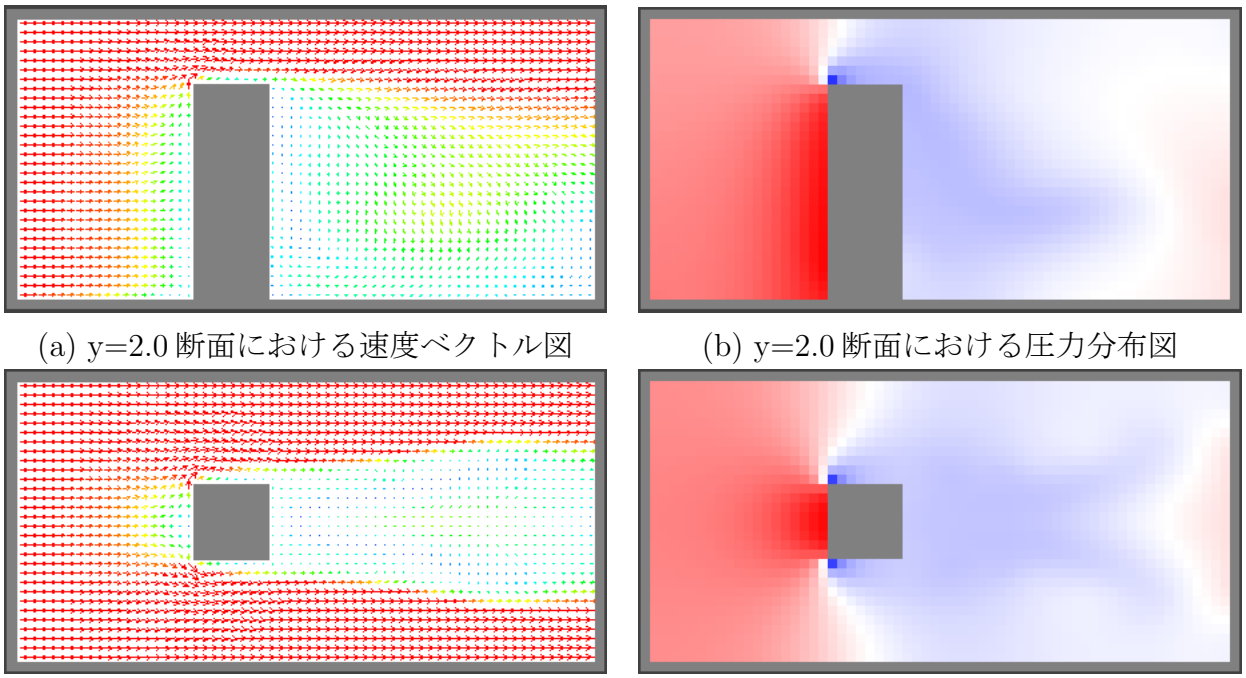


(c) z=1.0 断面における速度ベクトル図



(d) z=1.0 断面における圧力分布図

図 5.8: 計算結果 (t=7.5)



(a) y=2.0 断面における速度ベクトル図　(b) y=2.0 断面における圧力分布図

(c) z=1.0 断面における速度ベクトル図　(d) z=1.0 断面における圧力分布図

図 5.9: 計算結果 (t=10.0)

5.3.2　シミュレーションの実行時間

シミュレーションの開始から $t = 10.0$ のシミュレーションが終わるまでに所要した時間をシミュレーションの実行時間，ループ分割数及びタスクの分割数は 16 とし，本節では実行時間の測定結果を示す．

可視化処理による実行時間への影響を調べた．可視化画面なしの場合でシミュレーションに費やした時間と，可視化断面 $z = 1.0$ における圧力分布図の描画の場合，可視化断面 $z = 1.0$ における速度ベクトル図の描画の場合とで実行時間を比較した．

図 5.10 より，圧力分布図の描画は可視化画面なしに比べ 2.22%の測定時間の遅延が見られ，速度ベクトル図の描画は可視化画面なしに比べ 4.65%の実行時間の遅延が見られた．以下の測定では，遅延が見られるがシミュレーションの進行状況を確認できる $z = 1.0$ における速度ベクトル図の描画を実行しての測定を行う．

同機種分散計算環境における高速化率を求め，図 5.11 に示す．G 台の Worker での実行時間を T_G，基準となる Worker 1 台での実行時間を T_1 としたときの，高速化率 S_G は以下のように表す．

$$S_G = T_1/T_G \tag{5.1}$$

2 台の Worker では 1.474 の高速化率，4 台の Worker では 1.910 の高速化率が得られた．

表 5.3 で表される計算機性能の Worker と，表 5.2 で表される JiniServer と Master によって構成された異機種分散計算環境を構築した．各 Worker の処理能力を測定し，処理能力

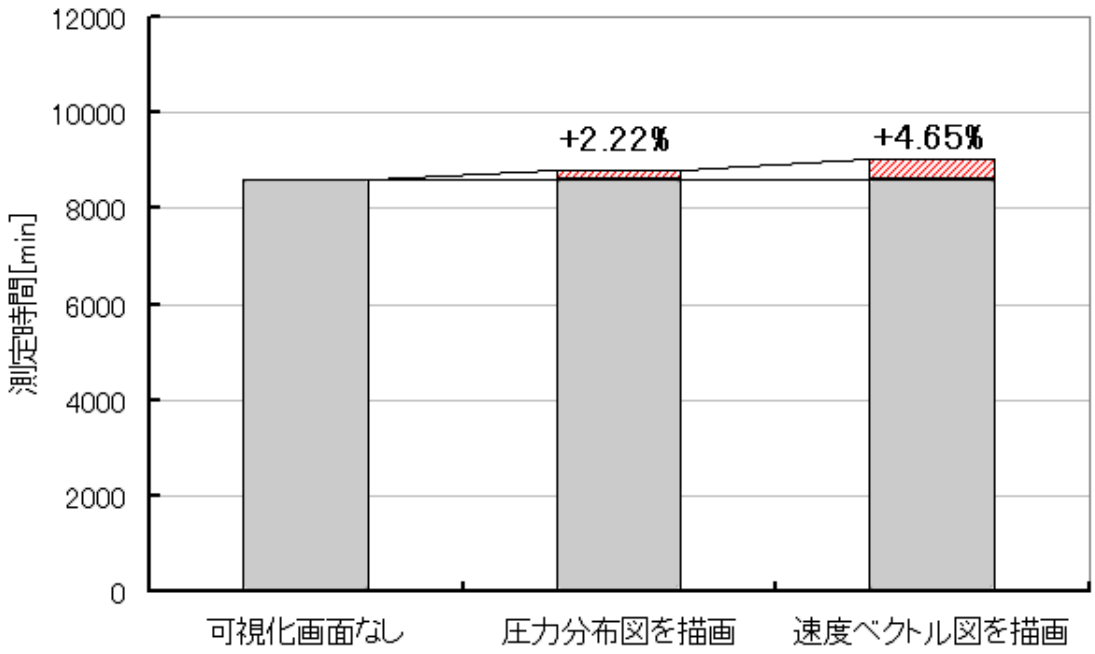


図 5.10: 可視化による遅延

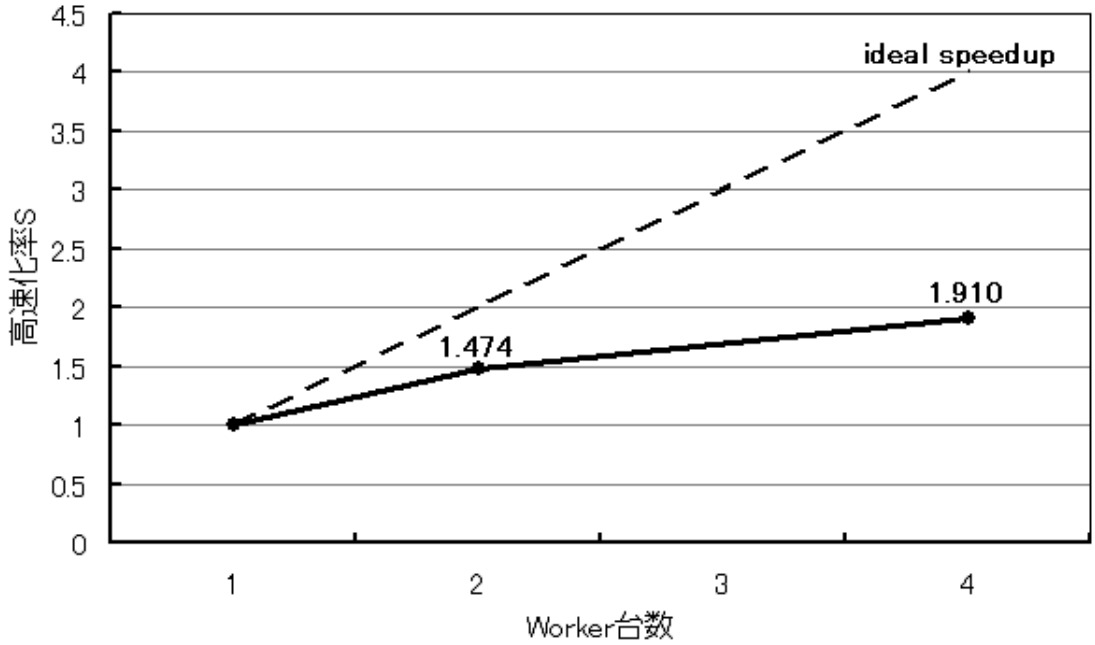


図 5.11: 同機種分散計算環境における高速化率

に応じたタスクの分配がなされ負荷分散が行われたかを調べる．

表 5.3: 異機種分散計算環境を構築した計算機構成

	WorkStation1	WorkStation2	WorkStation3	WorkStation4
CPU	SPARC IIIi 1.0GHz	SPARC IIe 550MHz	SPARC IIIi 1.0GHz	SPARC IIIi 1.0GHz
MEMORY	512MB	640MB	512MB	1024MB
OS	Solaris-8	Solaris-9	Solaris-9	Solaris-9

基準となる Worker を WorkStation1 とし，この実行時間を T_1 として式 5.1 より，表 5.4 に表され Worker の構成による異機種分散計算環境における高速化率を求め，図 5.12 に示す．

表 5.4: 異機種分散計算環境における Worker の構成

台数	利用した Worker	WorkStation1 との高速化率
1	WorkStation1	1.0
2	WorkStation1,2	1.23
4	WorkStation1,2,3,4	1.81
8	WorkStation1×4 台, $2, 3 \times 2$ 台, 4	2.02

各 Worker 単体でシミュレーションを行い，実行時間を Worker の処理能力とし，WorkStation1 の処理能力を基準に，それぞれの Worker の性能比 $SpecRatio$ を以下の式で表し，表 5.5 および図 5.13 に示した．

$$SpecRatio = T_{WorkStation1}/T_{WorkStation} \tag{5.2}$$

$T_{WorkStation}$ は対象となる Worker の実行時間，$T_{WorkStation1}$ は WorkStation1 の実行時間とする．なお，実行時間には通信負荷による遅延も含まれる．

WorkStation1,2,3,4 で構成される 4 台の Worker でシミュレーションを行い，各 Worker におけるタスクの実行回数を計測し，WorkStation1 の実行回数を基準に，それぞれの Worker のタスク実行回数比 $TaskCountRatio$ を以下の式で表し，表 5.5 および図 5.13 に示した．

$$TaskCountRatio = (S_{WorkStation1}/S_{WorkStation})^{-1} \tag{5.3}$$

$S_{WorkStation}$ は対象となる Worker のタスク実行回数，$S_{WorkStation1}$ は WorkStation1 のタスク実行回数とする．処理能力が高いほど実行時間は短くなるのに対し，処理能力が高い

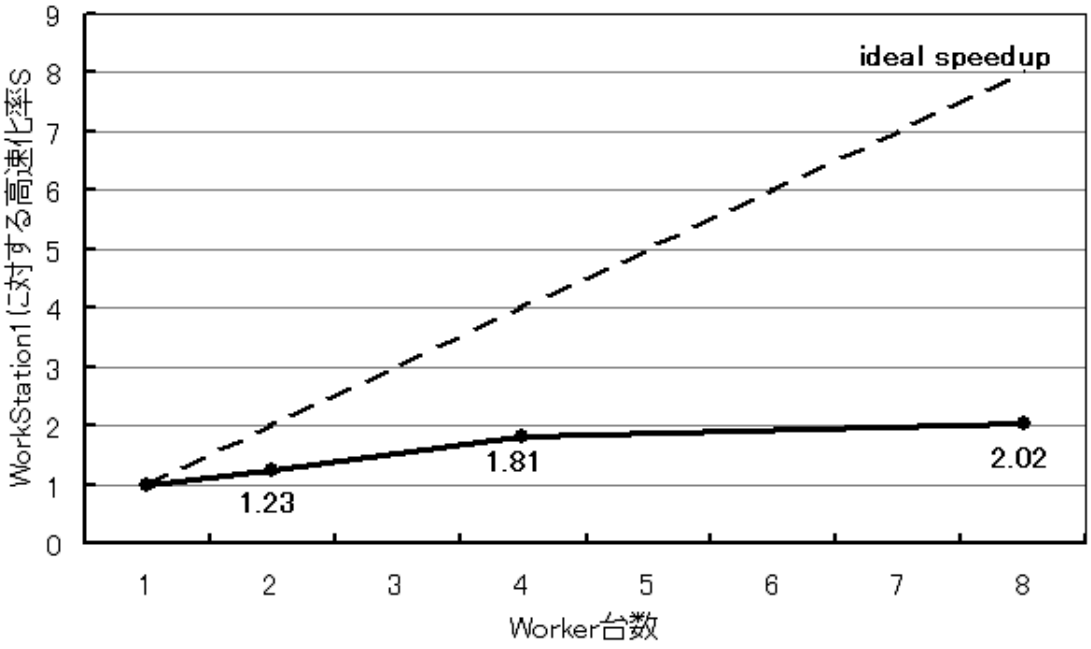


図 5.12: 異機種分散計算環境における高速化率

ほどタスクの実行回数は増えるので，性能比と比較するためにタスクの実行回数比は逆数を取った．ここで，タスク実行回数比１はタスクの分割数を１６，タスク実行回数比２はタスクの分割数を８，タスク実行回数比３はタスクの分割数を４と設定したものである．

表 5.5: 各 WorkStation における性能比とタスク実行回数比

	WorkStation1	WorkStation2	WorkStation3	WorkStation4
性能比	1.00	0.55	1.01	0.96
タスク実行回数比１	1.00	0.75	0.99	0.93
タスク実行回数比２	1.00	0.74	0.99	0.99
タスク実行回数比３	1.00	1.00	1.00	1.00

タスクの分割数を変化させた際の WorkStation1,2,3,4 で構成される４台の Worker でシミュレーションを行った実行時間を図 5.14 に示す．

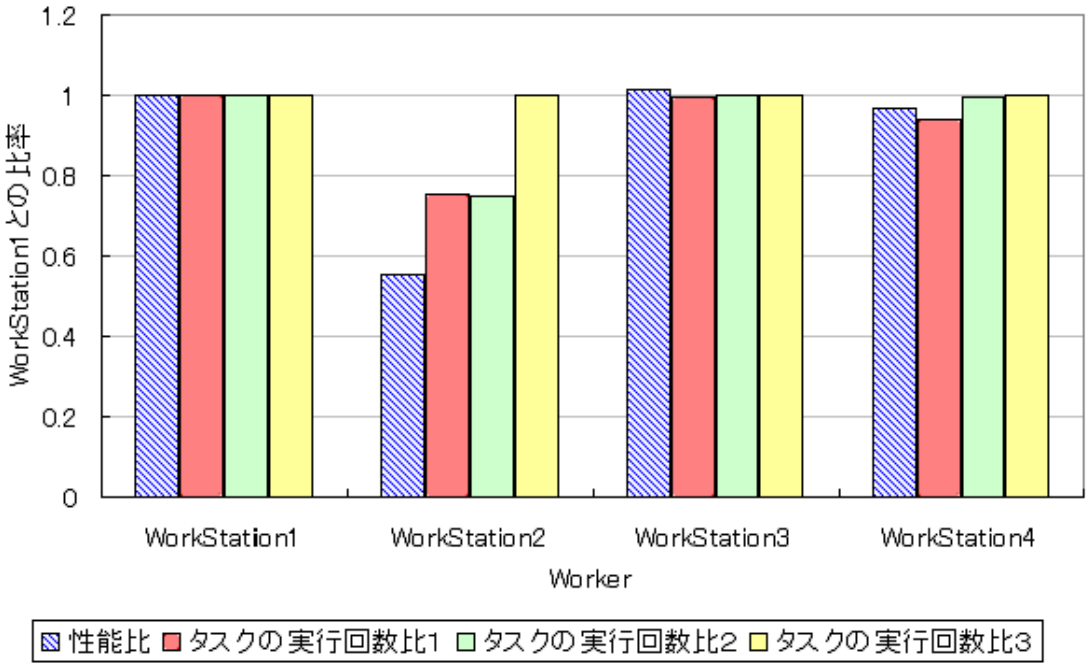


図 5.13: 各 WorkStation における性能比とタスクの実行回数比

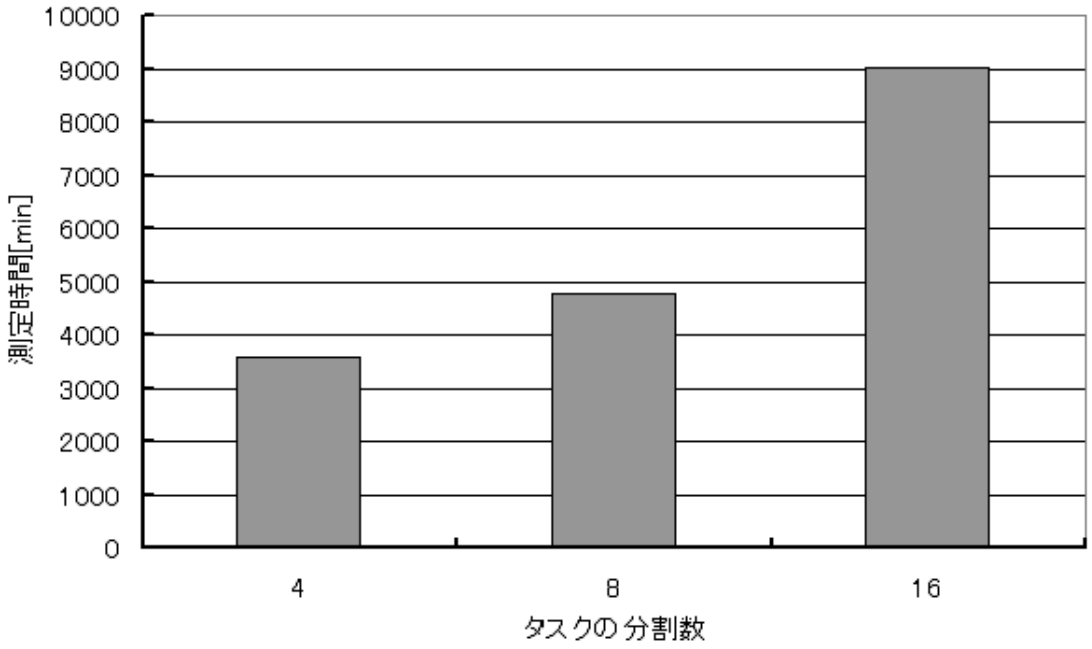


図 5.14: タスクの分割数の変化と実行時間

第6章　考察

本研究では，３次元数値流体シミュレーションのリアルタイム可視化システムを，UNICORE と Jini 技術とを利用した分散計算環境上で構築し，5.3 節より本システムは以下のことが考察された．

6.1　数値流体シミュレーション

３次元数値流体シミュレーション機能の確認として，５章で３次元角柱ながれ解析を行い，図 5.5-5.9 より，$y = 2.0$ 断面と $z = 1.0$ 断面における速度ベクトル図と圧力分布図を計算結果として示した．$y = 2.0$ 断面からは角柱を回り込み，大きな渦が発生された様子が見らた．また，$z = 1.0$ 断面からは角柱後方に双子渦が発生し成長する様子が見られたが，周期的に発生するカルマン渦の発生は確認することができなかった．これは角柱周辺の計算格子の不足や計算解法の精度不足，もしくはより長い時間経過が必要だったことが考えられる．

6.2　リアルタイム可視化とステアリング機能

本システムでは，JavaSpaces を介して Worker から設定した可視化断面に関する計算データを集め，タイムステップ毎もしくは反復計算毎に可視化を行い，計算が正しく行われているかを確認できるリアルタイム可視化を実装した．また，シミュレーション結果を詳細に確認するため，計算と可視化を一時停止・再開するステアリング機能や，可視化画面の拡大・移動の機能，可視化断面の変更・追加の機能を実装した．

図 5.2 より，操作パネルでの”PLAY”ボタンにより計算が実行され，”STOP”ボタンにより描画が静止し，反復計算またはタイムステップにおいて計算が停止した．さらに停止中に”PLAY”のボタンを押すことにより計算が再開された．よってステアリング機能が正しく機能したことが確認できた．

図 5.2 における操作パネルの右側にある可視化画面の設定により，任意の可視化ウィンドウ番号を指定し可視化する断面の対象となる座標と，可視化する断面の座標値を入力，速度または圧力の可視化内容と，ベクトルまたは分布の可視化方法を指定し，ボタンを押すと可視化ウィンドウが表示された．図 5.3 では速度ベクトル図，速度分布図，圧力分布図の３種類の可視化の表現ができ，任意の可視化内容と可視化方法で可視化できることが

確認された．また図 5.5-5.9 より $y = 2.0$ 断面及び $z = 1.0$ 断面を可視化でき，任意の可視化断面となる座標軸と断面位置ができることが確認された．

さらに，すでに表示されている可視化ウィンドウ番号を指定しボタンを押すと，指定された番号のウィンドウが閉じた．このように，可視化ウィンドウを一度閉じ，可視化の条件を再設定し，ウィンドウを開くことで可視化断面を変更することができた．また，可視化ウィンドウの数は任意の枚数開くことができた．可視化断面の変更，追加は計算中においても変更することができた．

任意の可視化ウィンドウにフォーカスを当て，”PageUp”キーを入力すると可視化画面が拡大し，”PageDown”キーを入力すると可視化画面が縮小した．矢印キーを押すことによって，可視化位置が移動した．これにより図 5.4 では，可視化画面を拡大・移動することによって，角柱まわりの流れを強調して確認することができた．

図 5.5-5.9 より，シミュレーションに合わせて計算結果の可視化を行い，ユーザが常に計算結果を確認できるシステムを構築することができ，可視化断面を変更もしくは追加することによって，ユーザは任意の奥行き方向の挙動を把握することができた．

6.3 分散計算環境

大規模計算になるほど，計算結果をファイルに出力し利用者端末へ転送し可視化するには，ストレージ容量の問題やファイル出力時における解析への負荷が考えられてきたが，リアルタイム可視化はシミュレーションに合わせて可視化に必要な計算結果のみを利用者端末へデータとして転送し可視化を行い，データでの転送を行うためストレージ容量の問題は克服された．しかし，ファイル出力は行わないが可視化に関して解析への負荷が与えられる可能性がある．図 5.10 より，可視化によって解析への負荷が若干ながらも与えられたことがわかった．圧力分布図の描画には Worker から圧力 p の値を Master へ転送させているが，速度ベクトル図の描画には速度 u, v, w の値を Master へ転送させているためにネットワーク負荷が高く，圧力分布図の描画に比べ速度ベクトル図の描画のほうが実行時間の遅延が生じると考えられる．

同機種分散計算環境における高速化率は，図 5.11 で表されるように，Worker 台数 4 台で最大 1.9 倍の高速化率が得られた．

- 計算データが分割されているために JavaSpaces へのアクセス頻度が多い
- Jini 技術や JavaSpaces の通信負荷が大きかった
- JavaSpaces から取り出した計算データオブジェクトや境界値オブジェクトからの変数の取得の処理からの負荷
- 計算終了時における新しい計算データオブジェクトや境界値オブジェクトの生成の負荷

これらが ideal speed up よりも低い高速化率であった原因として考えられる．

異機種の計算機で構成された分散計算環境において，JavaSpaces をタスクバッグにした負荷分散を調べるため，図 5.13 で示されるように性能比とタスクの実行回数比を比較した．タスクの分割数 = 計算機台数としたタスクの分割数 4 では，計算機の性能によらずどれも同じ数のタスクが実行され，負荷分散はされなかったように思われる．タスクの分割数 8 や 16 では，WorkStation2 のような性能の劣った計算機に対しタスクの実行回数が減少したため，負荷分散が行われたと思われる．

また，図 5.13 でのタスクの分割数 8 や 16 において，WorkStation2 に関して性能に対して多めにタスクが実行されたように見受けられる．これは 1 度のループ分割においてタスクの数が計算機台数よりも多い場合性能に関係なく計算機すべてに最低 1 回はタスクが与えられるため，WorkStation2 のような性能の低い計算機にもある一定の回数タスクが実行されるのではないかと考えられる．これを克服するには，よりタスクの粒度を細かくし性能のよい計算機がより多くタスクを実行することによって低性能の計算機に与えられる一定のタスクの量を無視できることになるが，タスクの粒度を細かくすることによって JavaSpaces への通信回数が増えることになり，通信速度の低いネットワーク環境下では通信処理による遅延が大きくなる．

タスクの分割数が増えると，タスクや計算データの取得の回数が増えるためにネットワークアクセスが頻繁になり，通信負荷の影響がより顕著に現れる．図 5.14 で示されるように，タスクの分割数を増やしたところ，それだけ JavaSpaces へのアクセス回数が増え，実行時間の増加が見られた．通信負荷の影響を緩和するには，より高速なネットワークを利用する．もしくは計算規模を大きくし 1 タスクあたりの計算負荷を重くすることによって，実行時間に占める計算負荷が通信負荷よりも増加し，通信負荷の影響が緩和できる．

異機種分散計算環境における高速化率は，図 5.12 で表されるように，Worker 台数 8 台で最大 2.02 の高速化率が得られた．高速化率が低い原因は，低性能の計算機に負荷分散しきれなかったことや，同機種分散計算環境における高速化率の原因と同じものが考えられる．

第7章　結言

7.1　まとめ

UNICORE と Jini 技術によって構成された分散計算環境上で動作する，３次元数値シミュレーション機能，任意の座標軸の断面によるリアルタイム可視化機能，可視化断面の変更・追加機能，ステアリング機能が実装されたシステムを構築した．

本システムの検証として，３次元角柱流れ解析を行い，任意の座標軸の断面によるリアルタイム可視化の実現と可視化断面の変更・追加，速度もしくは圧力といった可視化内容とベクトル図もしくは分布図といった可視化方法の変更，可視化画面の拡大・縮小・移動機能，ステアリング機能による計算と可視化を一時停止・再開をそれぞれ確認した．可視化による測定時間への影響を測定し，可視化なしに比べ圧力分布図の描画では 2.22%，速度ベクトル図の描画では 4.65%の遅延が見られた．

分散計算環境としての評価として，同機種分散計算環境における高速化率では４台での計測で最大 1.91 倍の効率が得られ，異機種分散計算環境における高速化率では８台での計測で最大 2.02 倍の効率が得られた．本システムでは JavaSpaces をタスクバッグとして構成した負荷分散を適用し，負荷分散の検証を行った．タスクの分割数が計算機台数と同等の場合，タスクの実行回数は計算機性能によらず一定となり，負荷分散が行われなかった結果が得られた．しかし計算機台数よりもタスクの分割数が多い場合，性能に応じてタスクの実行回数が変化しており，負荷分散が行われていたと考えられた．タスクの分割数を変化させた測定結果では，分割数が少ない結果が最も実行時間が短かった．これは計算による負荷よりも通信負荷が大きく影響したためと考えられ，タスク１つあたりの計算負荷に対して通信負荷が無視できるような大規模な計算を行い，測定を行う必要があると思われる．

分散計算環境としての評価では，より効率的なタスクの分割による負荷分散や，計算領域の拡大による影響がこれからの課題になった．しかし可視化としての評価では，利用者に分散計算環境での３次元数値シミュレーションのリアルタイム可視化システムとして，可視化面では３次元本システムは有効なものであると結論付けられる．

7.2　今後の課題

- 汎用可視化ツールによる一定時間間隔毎のファイル可視化機能
  汎用可視化ツールによる一定時間間隔毎のファイル可視化機能を実装する．一定時

間毎に計算結果を汎用可視化ツールの入力フォーマットに則ってファイルに書き出し，汎用可視化ツールがファイルを読み込み可視化する．これによって，全計算領域が求まらないと表示することができない流線図を描画することができる．利用者はリアルタイム可視化では得られなかった，より詳細な計算結果を一定時間毎に得ることができる．

- 可視化環境の改善
  リアルタイム可視化におけるベクトル長とコンター図のスケールが計算中に大きく変動する場合のために再設定できる機能や，コンターのスペクトルの表示機能，ステアリングによる停止中において可視化断面を変更した場合に再描画する機能，任意のタイムステップ毎に可視化をログとして保管し過去の可視化断面との比較が容易にできる機能を実装する．

  速度ベクトル図の描画に関して，可視化する座標軸によらず Worker は Master へ速度 u, v, w の値を転送していたが，可視化する際は断面情報のみ必要であるので Worker は可視化する座標軸に応じて速度の値を選んで転送することで通信負荷が軽減される．また，速度コンター図の描画に関しても同様に，Worker は Master へ速度 u, v, w の値を転送していたので，Worker にて u, v, w からコンター図描画に必要な絶対値を求め，これを転送することで通信負荷が軽減される．さらにより大きな規模の計算になり可視化断面の情報を転送するにも負荷がかかる場合において，可視化断面の情報を可視化に影響がない程度に間引き処理を加える必要もある．

- 高速化率及び負荷分散
  計算初期での反復計算は多くの回数反復することになるが，反復のたびに収束判定を行い反復計算のループ分割処理に同期をかけるのはリスクを伴う．よって，収束判定を任意のタイミングで行い，ループ分割処理に極力同期をかけることなく計算を行うべきである．

  以上のことが本システムでの機能の拡張と今後の課題として挙げられる．

謝辞

本研究を行うにあたり，松澤 照男 教授には，御指導，御鞭撻を賜り，深く感謝致します.

また，分散計算環境構築やループ分割処理に関して有益な御意見，御助言，御協力をいただきました大岩 博史 様 (本学博士後期課程)，太田 理 様 (本学博士後期課程)，数値流体シミュレーションに関して貴重な意見をくださいました渡邉 正宏 様 (情報科学センター)，野口 和博 様 (本学博士前期課程)，その他多くの御協力をくださいました中山 敏男 様 (本学博士後期課程)，松澤研究室の皆様に心から感謝いたします.

最後に，励ましや支えとなってくれた家族や友人知人の皆様，本当にありがとうございました.

2006 年 2 月
松本 浩之

参考文献

[1] globus toolkit, "http://www.globus.org/"

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

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

[4] 溝渕貴裕, 石井克哉, 宮本亮, 橋本敦, "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, pp.126, 2003.

[5] 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, pp.234, 2004.

[6] 大岩博史，松澤照男，"異機種並列計算における数値流体計算の負荷分散", 第 16 回数値流体力学講演論文集，pp.227, 2002.

本研究に関する発表論文

[7] 大岩博史，浅野喜宣，松本浩之，松澤照男，”広域ネットワーク上での数値流体計算におけるリアルタイム可視化システムの構築” 日本機械学会 2005 年度年次大会講演論文集 Vol1, pp.57.

[8] Hiroshi Oiwa, Yoshinobu Asano, Hiroyuki Matsumoto, Teruo Matsuzawa, ”Development of the real-time visualization system on network” PDCAT’2005 (December 5-8, 2005, Dalian, China) pp.784-786.

[9] 松本浩之，大岩博史，松澤照男，”分散計算環境上での 3 次元数値流体シミュレーションのリアルタイム可視化システム”, 第 19 回数値流体力学講演論文集, pp.144, 2005.