JAERI-Data/Code
2003-010

分散コンピューティング・システム整備

― コントロールサーバの開発 ―

2003年8月

磯貝　健太郎

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは、日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは、日本原子力研究所研究情報部研究情報課（〒319-1195 茨城県那珂郡東海村）あて、お申し越し下さい。なお、このほかに財団法人原子力弘済会資料センター（〒319-1195 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布を行っております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 〒319-1195, Japan.

編集兼発行　日本原子力研究所

JAERI－Data/Code 2003－010

分散コンピューティング・システム整備

－コントロールサーバの開発－

日本原子力研究所計算科学技術推進センター
磯貝　健太郎

（2003年5月2日受理）

今日、様々な研究分野において大型計算機を利用した大規模シミュレーション（計算）が行われているが、研究者にとって無駄な労力を課せられている場合がある。それは、計算機の大型化・高性能化により、取り扱うことのできるデータが多種大量となってきたため、その入出力データの管理が充分なものではなくなっていることや、計算しようとしている入力パラメータ・環境設定が、実は同じ分野を研究している他の研究者が既に計算済であるにもかかわらず、情報を共有できていない、もしくは計算結果を紛失したなど、再度長時間を要する計算を実行しなければならず、研究者が本来の研究活動とは無縁なところで時間を費やしていることである。このように、入出力データを一元管理し、かつ、過去のデータを自由に、労することなく参照できるシステムの必要性が生じてきた。また、超並列計算機や可視化計算機、データベースサーバなど複数のサーバを連携して大規模シミュレーションを遂行できるよう、ITBL 利用推進室では、研究者にとって研究活動を促進するためのシステムの構築を目指し、大規模シミュレーションを支援する分散コンピューティング・システムの開発・整備を行ってきた。

本報告書では、分散コンピューティング・システムの紹介及びその中で特にコントロールサーバの解説を行う。

日本原子力研究所（関西駐在）：〒619-0215　京都府相楽郡木津町梅美台8-1

JAERI－Data/Code 2003－010

Distributed Computing System

- Development of a Control Server -

Kentaro ISOGAI

Center for Promotion of Computational Science and Engineering
(Kansai Site)
Japan Atomic Energy Research Institute
Kizu-cho, Souraku-gun, Kyoto

(Received May 2, 2003)

In these days, a large scale simulation by supercomputers has been prevalent in various fields of research and development. In such computations, however, there may be an excessively needless amount of work for researchers. This is due to such various facts such as that due to the development of high performance supercomputers, too much data or too large a variety of data have become possible to treat, or the management of input or output data has become inadequate or improper in circumstances such as setting the environment of input parameters, missing computational results, and information processing which are common experiences among researchers.

Therefore, we have been engaged in the construction of an integrated management system of input & output data in supporting a distributed computational system for large scale simulations for a variety of researchers as a common utility.

In this article, the introduction of the distributed computing system, especially the explanation of the control server in it is described.

Keywords: Distributed Computing System, Control Server, Large Scale Simulation, Integrated Management

目　　次

Contents

1. はじめに

ナノテクやバイオ、光量子などの研究分野において、一般にシミュレーション計算を実行するときには、入力パラメータの設定、計算実行、可視化・解析、データベース登録等の処理が必要となるが、近年、超並列計算機の性能向上は目覚しく、取り扱うことのできるデータ量、計算量が大規模なものとなり、実行するシミュレーション計算も大規模なものとなってきた。

また、このような大型計算機を個人で所有することができるはずもなく、共用計算機として存在することとなり、多くの研究者が研究活動の一環としてその大型計算機上でシミュレーション計算を実行することとなる。

各研究者が多種大量のデータを管理することも困難となってくるが、シミュレーション計算を実行する際、過去に他の研究者が実行したデータを自動的に参照し、同じ入力パラメータを用いて実行した結果があるときに、実際に長時間を要する計算を実行する前にそれを提示することができるシステムは、研究者にとって無駄な時間を省けるとともに研究活動を促進するものとなる。

また、運用管理側から見ると、すべての研究者にその計算資源を供給するためにいくつかの制約を課さざるを得ないが、その制約のひとつとして計算機（CPU）の占有時間に対する制限が設けられることとなる。

そこで各研究者は、実際には長時間を要するシミュレーション計算を分割して、複数回実行する必要が出てくるが、スムーズにシミュレーション計算を実行していくためには、ひとつの計算が終了したことを常時確認し、次の計算を実行していかなければならない。

このような計算終了の確認・次計算実行や、シミュレーション計算を実行している最中に出力される結果の可視化やデータベースへの登録など、機械的作業に対して、複数サーバ・システムを用いた分散処理により各処理を適宜適当なサーバへ割り振り、全体効率の向上を図ることは、非常に有益である。

そこで、これまで日本原子力研究所　計算科学技術推進センター[1]　ITBL利用推進室[2]では、研究者にとって研究活動を促進するためのシステムの構築を目指し、大規模シミュレーションを支援する分散コンピューティング・システムの開発・整備を行ってきた。

本開発・整備は関西研究所　光量子科学研究センター[3]　光量子シミュレーション研究グループと共同で実施されていることもあり、同グループにて開発された超並列プラズマ粒子（P-cube：Progressive Parallel Plasma）コードをシミュレーション・コードの対象としている。

本報告書では、この分散コンピューティング・システムについて解説する。

2. 分散コンピューティング・システム

2.1 背景

関西研究所　光量子科学研究センター　光量子シミュレーション研究グループでは、超並列計算機上で動作する超並列プラズマ粒子（P-cube：Progressive Parallel Plasma）コードを用いて、時空間的に大規模かつ高精度なシミュレーション計算を実行し、光量子実験研究を先導している。

しかしながら、シミュレーション計算を実行するに当たり大量のデータや限られた計算資源の効率的な取り扱いについて、以下のような問題が生じていた。

・多様なパラメータを多数入力する必要がある
・設定したパラメータと対応する出力結果の整理、管理が非効率的である
・多種、大量に出力されるデータの整理が必要である
・入力パラメータが結果に及ぼす影響、物理現象の分析（計算結果の解析）等に計算科学に関する専門知識（可視化手法、データフォーマットなど）が必要である
・シミュレーション計算の繰り返し実行が必要である

そこで、上記問題を解決するために、大規模シミュレーションの入力パラメータや出力結果を統合管理し、円滑な研究活動を支援する大規模シミュレーション支援システムを同グループにて開発してきた。

ITBL利用推進室では、ITBL構想[4]で提唱されている複数サイトを連携させて利用できるシステムやTME（Task Mapping Editor）、RIS（Resource Information Service）といったITBL基盤ソフトとの統合も視野に入れ、スクリプト言語（Perl、csh 等）で開発されてきた上記大規模シミュレーション・システムを、Java言語を利用した汎用性・拡張性のあるシステムに再構築すべく、開発・整備を行ってきた。

2.2 概要

本システムは、コントロールサーバを中心にWebサーバ、計算サーバ（超並列計算機）、データベースサーバ、可視化グラフィックサーバ、データサーバ、バックアップ装置の連携により実現する。

本システムの構成を図 2-1 に示す。

なお、各サーバおよび装置はLANで接続されている。

本システムで使用するデータを、大きく入力情報と出力結果に分類する。入力情報は、シミュレーションへの入力パラメータと実行環境の設定で、出力結果は、シミュレーションの出力データ（Snap データ、Image データ）と可視化データ（出力データを可視化した画像ファイル）で構成される。

各サーバの機能について以下に列記する。

・コントロールサーバ
　- 機能分散システムの制御
　- 超並列計算機の計算コード実行

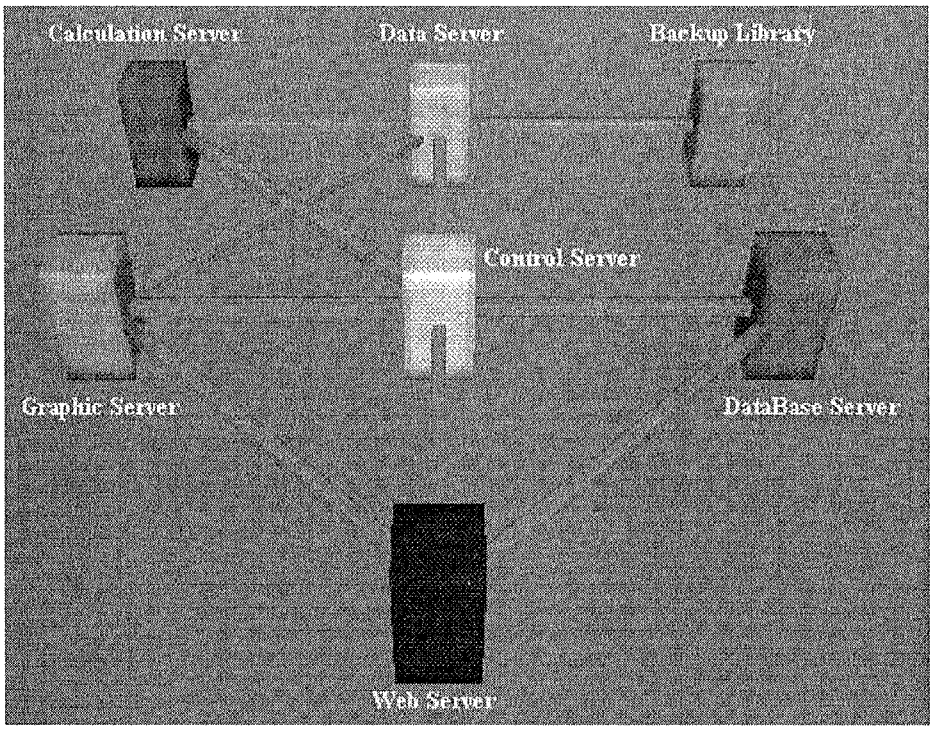

図 2-1 分散コンピューティング・システム

- グラフィックサーバの可視化実行
- データベースサーバへの出力結果登録
- バックアップ装置へのバックアップ・リストア実行

・Web サーバ
- データベースサーバへの入力パラメータ登録・検索
- コントロールサーバからシミュレーション計算の進捗取得
- シミュレーション可視化結果の表示

・超並列計算機（計算サーバ）
- 計算コード（大規模シミュレーション計算）の実行

・データベースサーバ
- 入力パラメータおよび出力結果を統合したデータベースを管理

・可視化グラフィックサーバ
- 出力データの可視（グラフ、アニメーション）化および格納
- p3plot,AVS/Express やフリーウェアなどのソフトウェアを搭載

・データサーバ
- 出力データの一時格納

・バックアップ装置
- 出力データのバックアップ・リストア

2.3 コントロールサーバ

シミュレーション計算の実行は、全てコントロールサーバが処理要求を受け取って、その計算の動作を制御する（図 2-2）。今回は本システム運用について核となるコントロールサーバの開発を行った。

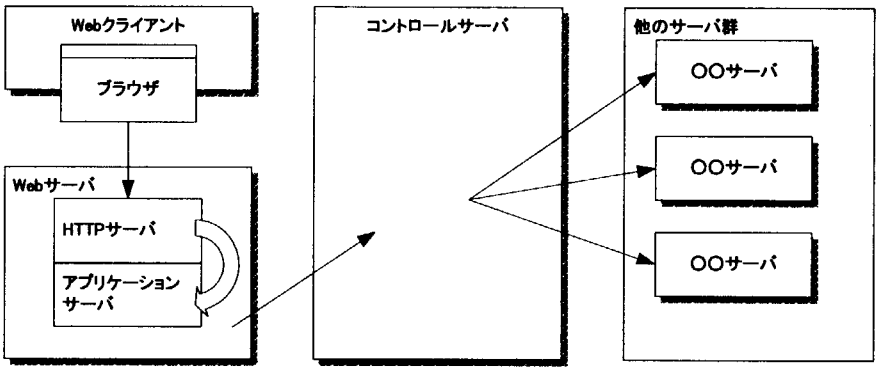

図 2-2 処理の流れ

コントロールサーバは、デーモンプロセスを形成し Web サーバからの処理要求を待ち受ける。

デーモンプロセスが Web サーバから処理要求を受信すると、処理要求に従いジョブスレッドの生成、または割り込み処理を実行する。

コントロールサーバの内部構成を図 2-3 に示す。

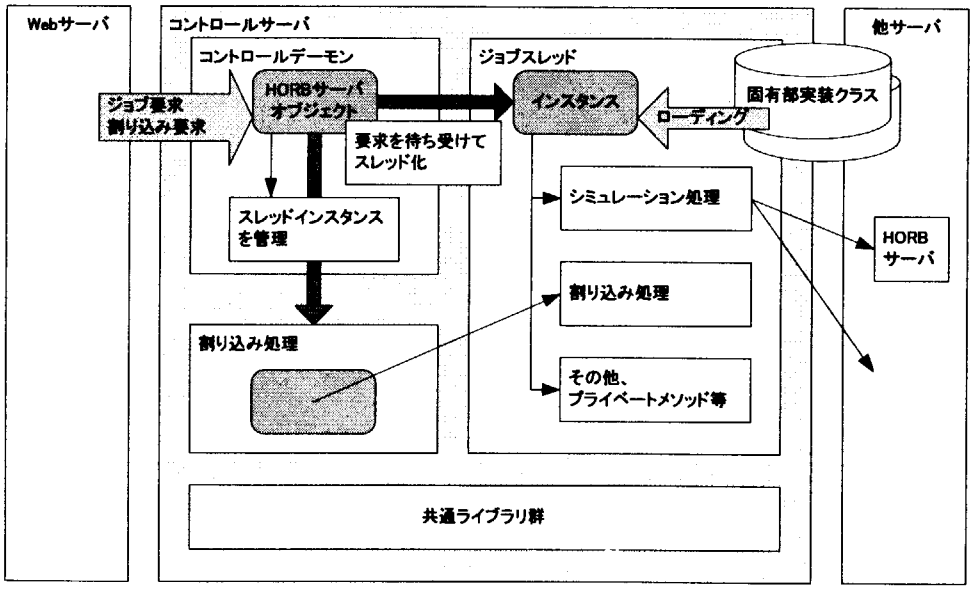

図 2-3 コントロールサーバ内部構成

2.4 処理フロー

分散コンピューティング・システムは、コントロールサーバを中心とした各種サーバ群から成り立つ。

ここでは、利用者から処理要求が行われた際に、各サーバ間での動作を段階ごとにまとめる。

(1) シミュレーション計算前１（動作環境設定）
(2) シミュレーション計算前２（シミュレーション入力パラメータ設定）
(3) シミュレーション計算開始
(4) シミュレーション計算実行中
(5) シミュレーション計算終了

2.4.1 シミュレーション計算前１（動作環境設定）

シミュレーション計算を実行する前に行う作業として、その計算実行のための環境を整えることが必要である。

ここでは、コントロールサーバがシミュレーション計算を実行する動作環境の設定を保持し、シミュレーション計算実行時に各サーバに配布する場合について説明する。

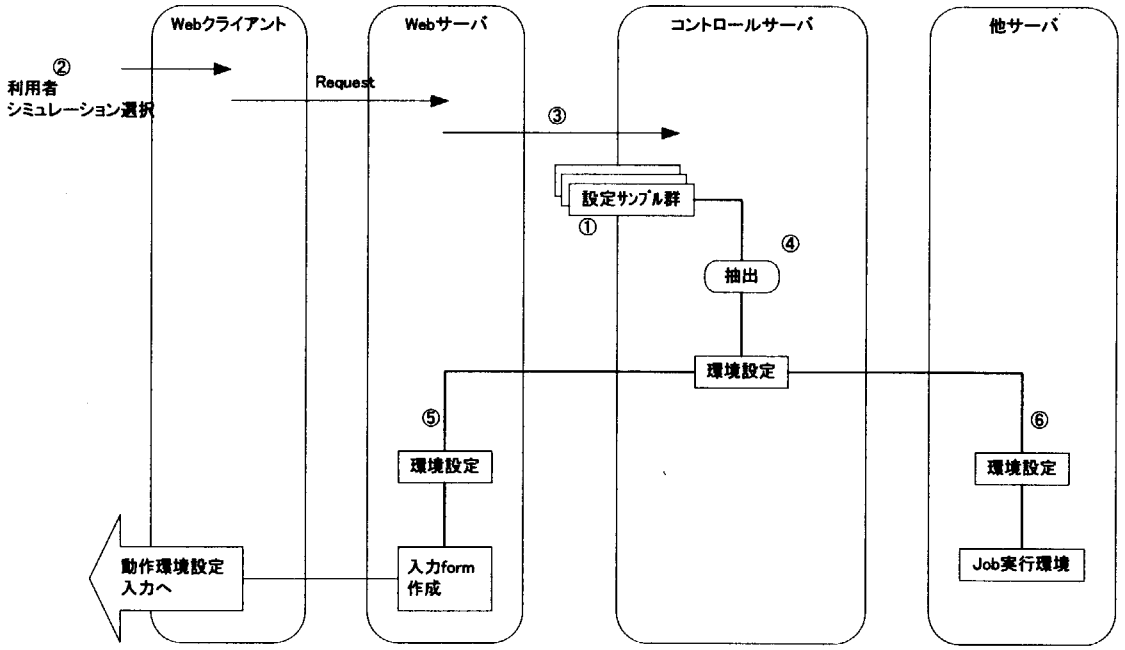

図 2-4 動作環境設定

①コントロールサーバが各シミュレーション・コードに対応する実行環境の設定サンプル群を管理する。

②利用者がシミュレーションを選択する。

③Web サーバ経由でコントロールサーバが要求シミュレーションを受信する。

④コントロールサーバの管理機能が該当シミュレーション・コードの環境設定を抽出する。

⑤Web サーバが入力 form を作成して Web クライアント画面に表示し、利用者は抽出した環境設定を編集する。

⑥シミュレーション計算を実行する際には、環境設定をシミュレーション計算に使用するサーバ（超並列計算機など）に配布する。

2.4.2 シミュレーション計算前2（シミュレーション入力パラメータ設定）

入力パラメータ設定フェーズは、シミュレーション計算を実行するための入力情報となる入力パラメータの設定を行う。

設定の方法としては3種類ある。

(1) 利用者が手で入力する。

(2) DB サーバから過去に実行した入力パラメータを取得する。

(3) Excel やエディタなど、外部で生成した CSV（Comma Separated Value）形式または XML（eXtensible Markup Language）ファイルを読み込む。

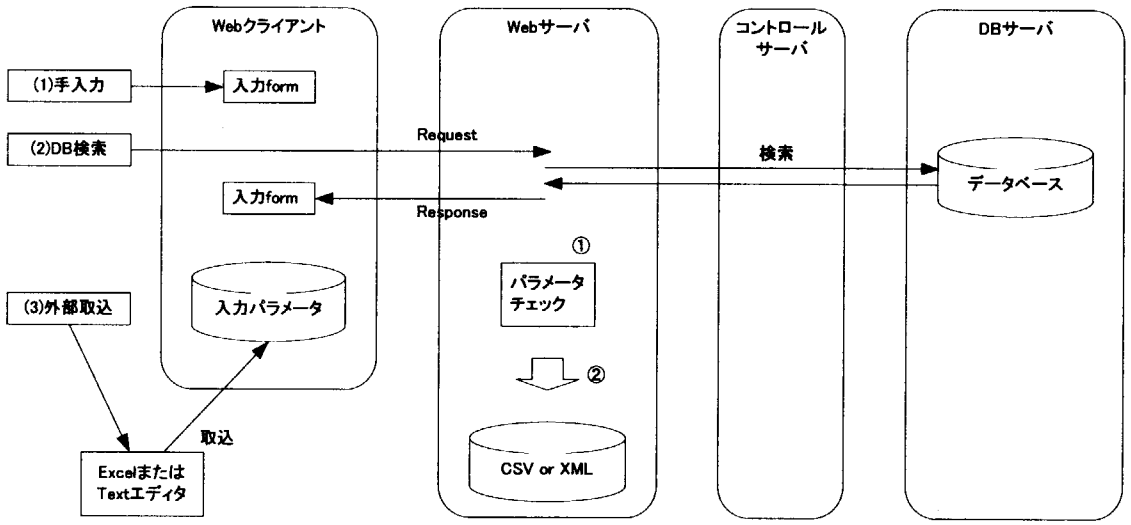

図 2-5 シミュレーション入力パラメータ設定

入力パラメータ設定の流れ

①Web サーバが入力パラメータのチェックを行う（範囲チェック、妥当性）。

②入力項目を CSV or XML 形式の文書に変換する。

2.4.3 シミュレーション計算開始

シミュレーション計算は、Web クライアントからの要求をトリガとし、Web サーバを経由してコントロールサーバからの要求発行により開始される。

ここでは、コントロールサーバが各サーバ上に存在する、もしくは配置するべきファイルの移動を制御する。

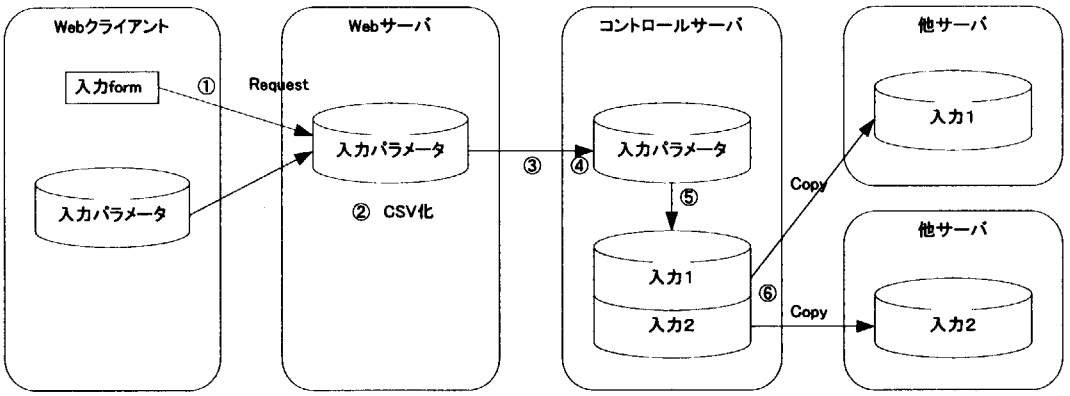

図 2-6 シミュレーション計算開始

①Web クライアントよりシミュレーション計算開始指示
②Web サーバは入力パラメータを CSV データ化
③Web サーバからコントロールサーバに対してシミュレーション計算の開始要求
④コントロールサーバが Web サーバから CSV データをダウンロード
⑤CSV データから各サーバ用に入力パラメータのファイルを生成
⑥コントロールサーバが各サーバに入力パラメータのファイルを転送

2.4.4 シミュレーション計算実行中

シミュレーション計算実行中の処理フローとして想定されるトリガは、以下の２種類ある。

（1） ジョブ実行状況監視
（2） ジョブ中断
（3） ジョブ停止

2.4.4.1 ジョブ実行状況監視

ジョブの実行状況の監視は、シミュレーション計算の処理中ステータスをHTML化してWebクライアントがジョブの進行状況を目視できるようにする機能である。コントロールサーバはジョブごとに各サーバの実行状況を監視し、状態の変化とともにジョブステータスファイルを生成する。

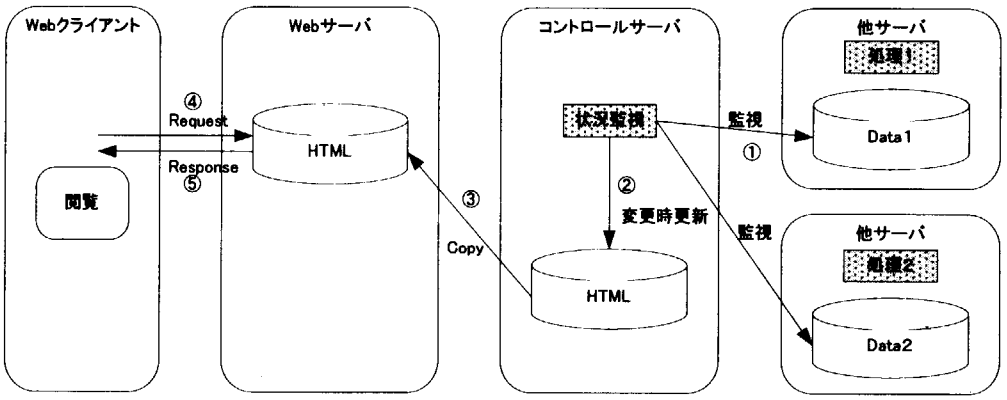

図 2-7　ジョブ実行状況監視

①コントロールサーバがジョブの進行状況を監視する。
②進行に変化があった場合、その進行状況を HTML 化する。
③HTML を Web サーバにアップロードする。
④Web クライアントよりジョブ進行状況の表示を要求。
⑤Web サーバは、現在保持している該当ジョブの HTML を Web クライアントに表示する。

2.4.4.2　ジョブ中断

コントロールサーバは、Web クライアントより依頼されたシミュレーションに対して、何らかの要因により処理を中断したい場合のインタフェースを提供する。

処理中断時はその時点で作成された結果ファイル、中間ファイルを削除することが必要となる。

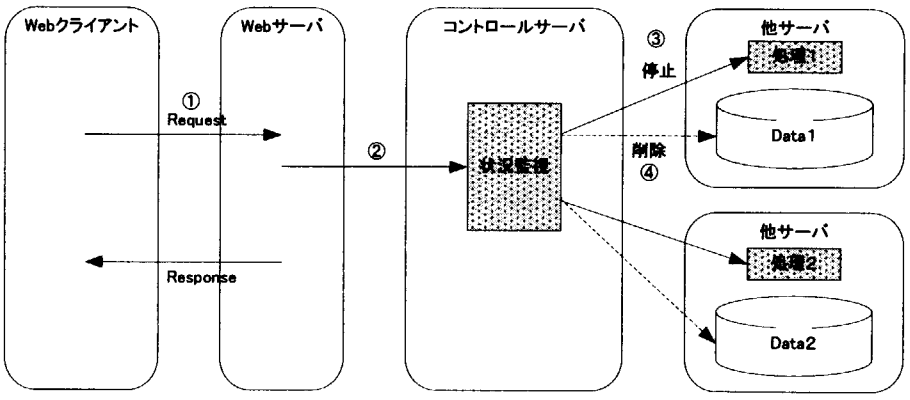

図 2-8　ジョブ中断

①Web クライアントがジョブのキャンセルを要求する。
②Web サーバがコントロールサーバにジョブのキャンセルを要求する。
③コントロールサーバはジョブの進行状況をチェックして稼動中処理をキャンセルする。
④コントロールサーバはジョブ中に生成された結果ファイル、中間ファイルを削除する。

2.4.4.3 ジョブ停止

コントロールサーバは、Web クライアントより依頼されたシミュレーションに対して、シミュレーション中の処理を停止するインタフェースを提供する。

処理停止時はその時点で作成された結果ファイルを回収して、停止するまでのジョブの結果をクライアントに表示することができる。

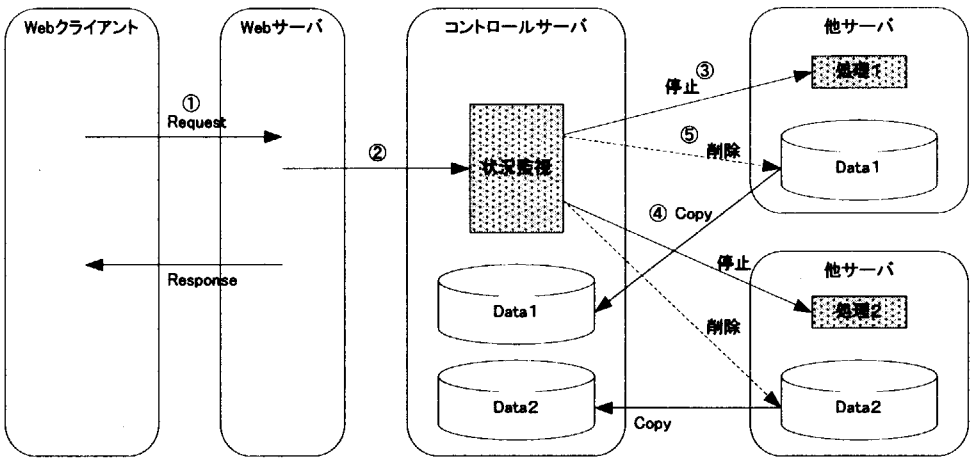

図 2-9 ジョブ停止

①Web クライアントがジョブのストップを要求する。

②Web サーバがコントロールサーバにジョブのストップを要求する。

③コントロールサーバはジョブの進行状況をチェックして稼動中処理をストップする。

④コントロールサーバはその時点にジョブ中で生成された結果ファイルを回収する。

⑤コントロールサーバはジョブ中に生成された結果ファイル、中間ファイルを削除する。

2.4.5 シミュレーション計算終了

2.4.5.1 ジョブ正常終了

コントロールサーバは、シミュレーション計算の監視時に予定されたジョブの内容が完了した場合に後処理を行う。

後処理は、各サーバ上で作成された結果ファイルを回収し、中間ファイル、作業用ディレクトリなどを削除する。

①コントロールサーバは、ジョブ完了後各サーバ上の結果ファイルを回収する。

②同様に各サーバ上の中間ファイル、回収後の結果ファイル、作業用ディレクトリを削除する。

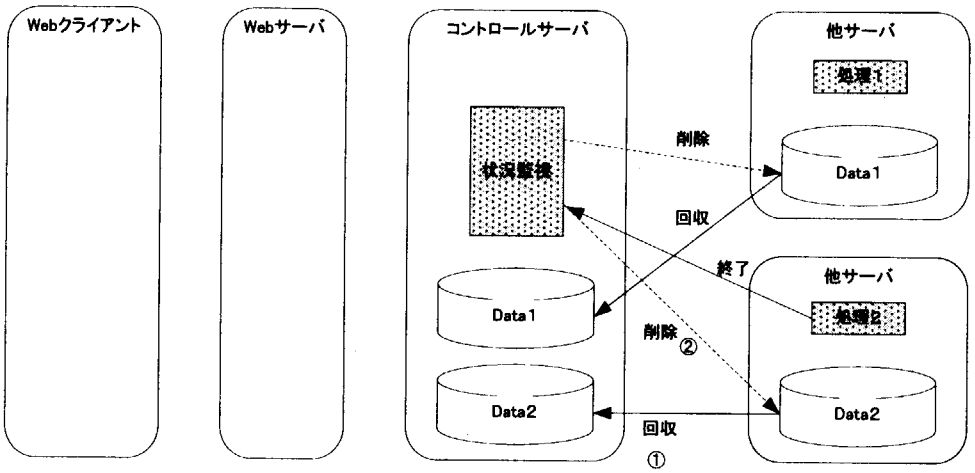

図 2-10　シミュレーション計算終了

2.4.5.2 異常終了

コントロールサーバの稼動中に異常を検出した場合の動作についてまとめる。

稼動中の異常発生については、異常の発生箇所、重要度によって処理が異なる。

以下に分類する。

(1) コントロールサーバのエンジン処理部で続行不可能な致命的なエラーを検出した場合

(2) コントロールサーバのエンジン処理部で続行可能なエラーを検出した場合

(3) コントロールサーバ内でジョブ処理中にエラーを検出した場合

(4) 他サーバ上でジョブ処理中にエラーを検出した場合

(1) コントロールサーバのエンジン処理部で続行不可能な致命的なエラーを検出した場合

コントロールサーバが続行不可能な致命的なエラーを検出した場合は、コントロールサーバのデーモンプロセスを終了する。

また、その時点で受け付け済みのジョブ要求については以下のように振舞う。

①実行待ちジョブ　：　コントロールサーバで待機する実行待ちジョブ
コントロールサーバ内で待機する全ジョブをキャンセルする。

②実行中ジョブ　：　コントロールサーバ内の全実行中処理について中断する。
ただし、ジョブ中に他のサーバへの処理要求を行ったものに関しては、突き放しでの処理とするため継続する。

③実行済みジョブ　：　特に処理しない

コントロールサーバは、ジョブのスケジューリングや監視中などの制御処理中に続行不可能なエラーを検知した場合、ログファイルに異常情報を出力し、自身のプロセスを終了する。

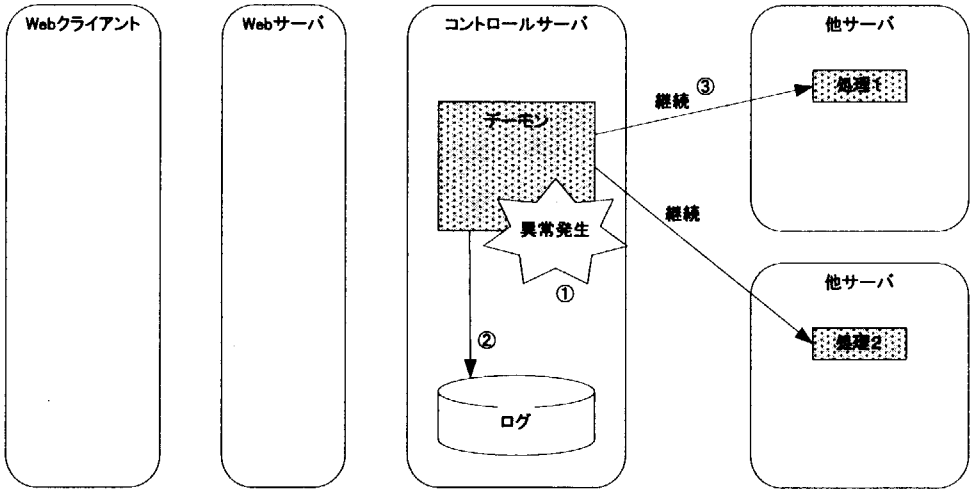

図 2-11　異常時処理(1)

①自身のプロセス内で異常が発生する。
②コントロールサーバが異常情報をログファイルに書き込む。
③コントロールサーバは他サーバで実行中の処理は継続する。
④自身のプロセスを終了する。

(2) コントロールサーバのエンジン処理部で続行可能なエラーを検出した場合
続行可能なエラーの場合、ワーニングエラーとして処理を続行する。
その際、ログファイルにエラー情報を出力する。

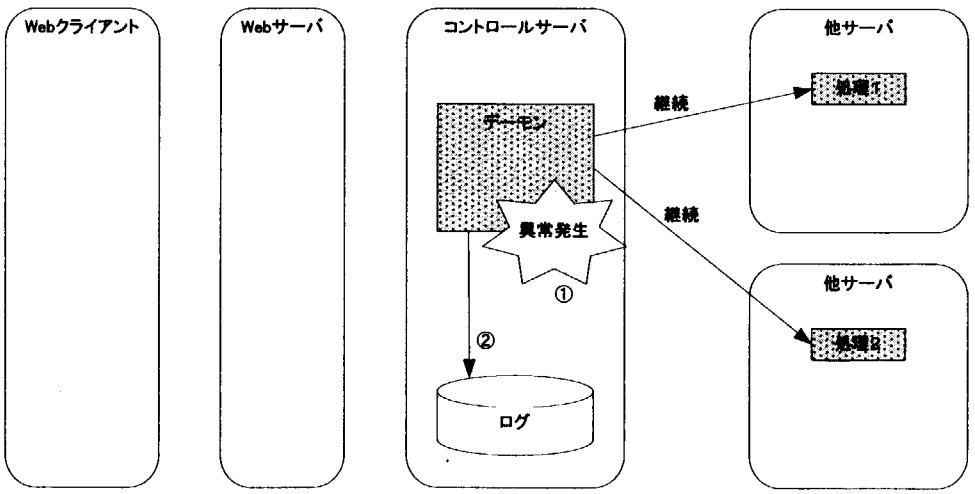

図 2-12　異常時処理(2)

①自身のプロセス内で異常が発生する。
②コントロールサーバが異常情報をログファイルに書き込む。

(3) コントロールサーバ内でジョブ処理中にエラーを検出した場合

コントロールサーバのジョブスケジューリング処理中にエラーを検出した場合、ジョブを中断する。

コントロールサーバは当該ジョブを異常終了ジョブとしてスレッドを終了する。

その際、他サーバで実行中の処理に対しては処理を継続する。

エラー情報はログに出力する。

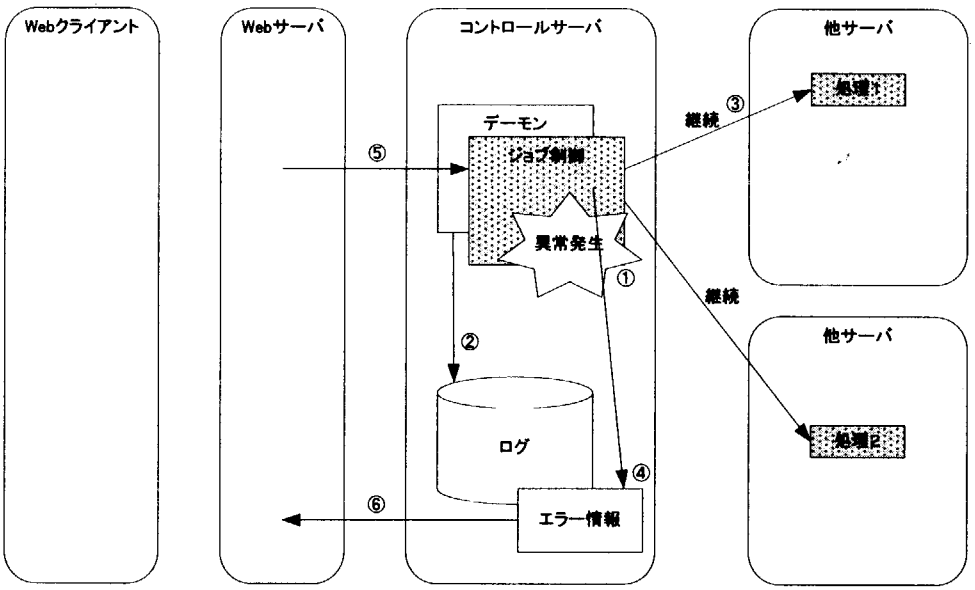

図 2-13　異常時処理(3)

①コントロールサーバのジョブ処理内で異常を検知する。

②ジョブ処理はログにエラー情報を出力する。

③他サーバで処理中の処理がある場合は、処理を継続する。

④コントロールサーバはエラー情報を内部で保持する。

⑤Web サーバよりジョブ状況確認を行う。

⑥コントロールサーバがエラー情報を Web サーバに通知する。

(4) 他サーバ上でジョブ処理中にエラーを検出した場合

他サーバでのジョブ実行中にエラーが発生した場合、他サーバ上に Java アプリケーション(HORB サーバ)が存在するかどうかでエラーの検知方法が変わる。

Java アプリケーションが存在する場合、Java アプリケーションが自サーバ内のジョブに対する異常を検知し、コントロールサーバに通知する。

また、Java アプリケーションが存在しない場合、コントロールサーバのジョブ監視機能が他サーバ上での処理異常を検知する。

ただし、この段階での異常検知は、個々のシミュレーションによって対応が異なる。

当該ジョブが異常を検知した後、どう振舞うかはシミュレーションによって異なる。

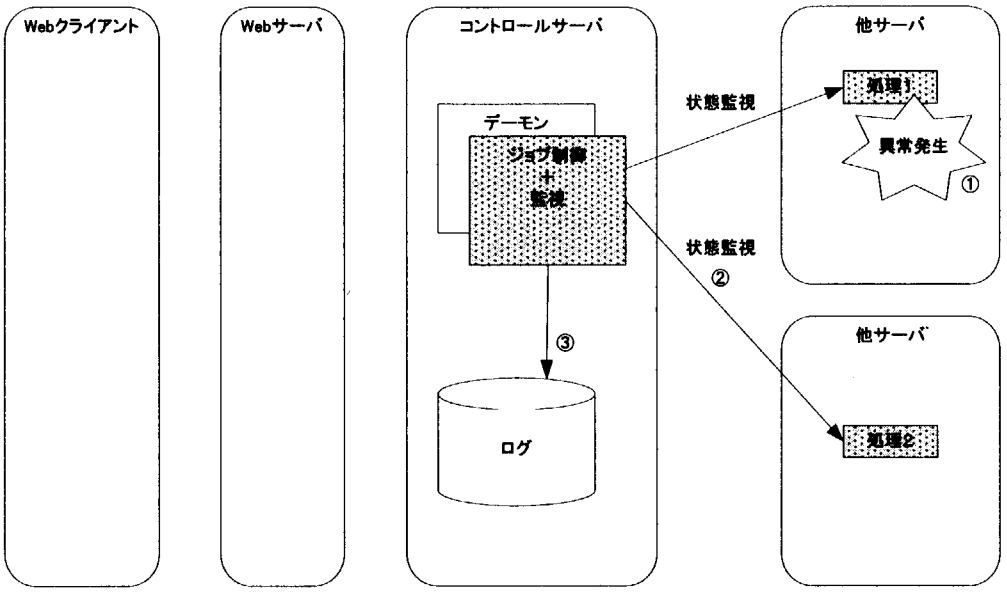

図 2-14　異常時処理(4)

① 他サーバ処理中に異常が発生
② 他サーバの監視アプリケーションが異常を検知する。
　または、監視アプリケーションが存在しない場合、コントロールサーバの監視処理が異常を検知する。
③ コントロールサーバのジョブ監視処理がエラー情報をログに出力する。

3. 利用について

3.1 ディレクトリ構造

本システムは、システムルートから下記の構造を構成して稼動する。

表 3-1 各ディレクトリとその役割

ディレクトリ名	役 割	備 考
Bin	起動／停止用バッチファイル、およびシェル	
Classes	システムのクラスファイルを配置	クラスファイル
Conf	システムの環境定義ファイルを配置	XML ファイル
Job	固有部クラスを配置	クラスファイル
Lib	システムが使用する jar (Java-archive) ファイルを配置	jar ファイル
Logs	システムが出力するログファイルの生成場所	
Service	サービスクラスを配置	クラスファイル

(1)bin ディレクトリ

bin ディレクトリは、本システムの起動や停止などの実行ファイルを配置する。

Linux/Windows 動作環境のどのオペレーティングシステムでも稼動できるように、バッチファイルとシェルファイルを用意する。

- startup.bat (Windows 動作環境用起動バッチファイル)
- startup.sh (Linux 動作環境用起動シェルファイル)
- stop.bat (Windows 動作環境用停止バッチファイル)
- stop.sh (Linux 動作環境用停止シェルファイル)

(2)classes ディレクトリ

classes ディレクトリは、lib ディレクトリに存在する jar ファイルより上位にローディングしたいクラスを配置する。

本システムのクラス群は lib ディレクトリに jar 形式で提供されるが、本システム上のあるクラスをオーバーライドしたい場合や、追加したいユーティリティクラスを生成した場合に当ディレクトリに配置する。

当ディレクトリは、JavaVM (Java Virtual Machine) 起動時に lib ディレクトリ下の jar 形式ファイルより上位にクラスパスを設定するため、当ディレクトリに配置したクラスを優先してロードされることになる。

したがって本システムインストール直後では、当ディレクトリにはクラスは存在しない。

(3)conf ディレクトリ

conf ディレクトリは、コントロールサーバに対する定義ファイル以外にサービス機能が稼動するための定義ファイルなど本システムが稼動するための定義ファイルを配

置する。

本システムインストール直後では、以下のファイルが存在する。

・コントロールサーバ定義ファイル(controlserver.xml)

・サーバ監視定義ファイル(servermanager.xml)

これらのファイルはXMLで構成されている。

定義ファイルの内容については、「3.6 定義ファイル」を参照のこと。

(4)jobディレクトリ

jobディレクトリは、利用者が実行したいシミュレーションの動作を実装したクラスを配置する。

当ディレクトリに配置するクラスは、必ずSimulatorImplクラスを継承するクラスであることが必要である。

SimulatorImplクラスを継承しないクラスは、たとえ当クラスに配置されたとしても固有部クラスとして実行されることはない。

クラスの実装については、「3.5 固有部クラス」を参照のこと。

(5)libディレクトリ

libディレクトリは、jar形式で作成されたファイルを配置する。

当ディレクトリに配置するjar形式ファイルは、本システムが稼動するために必要なクラス群である。

当ディレクトリで有効なファイルは、拡張子が「.jar」であることが必要である。

したがって、クラス単体を当ディレクトリに配置してもシステムとしてJavaVMがローディングを行うことはない。

(6)logsディレクトリ

logsディレクトリは、本システムが出力するログファイルが生成される。

ログについては、「3.7 ログ」を参照のこと。

(7)serviceディレクトリ

serviceディレクトリは、本機能に追加したいサービス機能を実装したクラスを配置する。

当ディレクトリに配置するクラスは、必ずBaseServiceクラスを継承するクラスであることが必要である。

BaseServiceクラスを継承しないクラスは、たとえ当ディレクトリに配置されたとしてもサービスクラスとして実行されることはない。

クラスの実装については、「3.4 サービスクラス」を参照のこと。

3.2 起動方法

本システムは、bin ディレクトリ配下に起動用のバッチファイル、およびシェルファイルを配置し、オペレーションシステムごとにその起動方法は異なる。

以下にそれぞれの起動方法について記述する。

※以下の"[Return]"は、リターンキー押下を示す。

(1)Linux 動作環境

Linux に関しては、以下のコマンドを実行することで起動を行う。

```
% startup.sh [Return]
```

(2)Windows 動作環境

Windows に関しては、以下のコマンドを実行することで起動を行う。

```
> startup [Return]
（または、> startup.bat [Return]）
```

3.3 停止方法

本システムは、bin ディレクトリ配下に停止用のバッチファイル、およびシェルファイルを配置し、オペレーションシステムごとにその停止方法は異なる。

以下にそれぞれの停止方法について記述する。

(1)Linux 動作環境

Linux に関しては、以下のコマンドを実行することで停止する。

```
% stop.sh [Return]
```

(2)Windows 動作環境

Windows に関しては、以下のコマンドを実行することで停止する。

```
> stop [Return]
（または、> stop.bat [Return]）
```

3.4 サービスクラス

サービスクラスとは、コントロールサーバがWebサーバから処理要求を受け付ける機能のほかに独自でアドオン的に動作する機能を提供するインタフェースである。

3.4.1 実装方法

サービスクラスは、コントロールサーバ起動時に動的に読み込まれ稼動する。
そのために、BaseServiceクラスを継承するクラスである必要がある。

パッケージは、"jp.go.jaeri.apr.service"パッケージとする。

```
public class サービスクラス名 extends BaseService {
    //実装
}
```

サービスクラスは、以下のメソッドを実装する必要がある。

(1)initメソッド

initメソッドは、サービスクラスの初期処理を行うクラスである。
例えば、定義情報の読込などサービスを行うために必要な準備処理を実装する。

```
public void init() {
    //初期処理実装
}
```

(2)invokeメソッド

invokeメソッドは、サービス機能自身を実行する処理を実装する。

```
public void invoke() {
    //サービス処理実装
}
```

(3)cleanupメソッド

cleanupメソッドは、サービスクラスの終了処理を行うクラスである。
例えば、サービスクラス内で使用した資源の解放など、後処理を実装する。

```
public void cleanup() {
    //終了処理実装
}
```

(4)fatalError メソッド

fatalError メソッドは、サービスクラス内の異常処理を行うクラスである。
サービスクラス実行時に発生した異常に対して行う処理を実装する。

```
public void fatalError() {
    //異常処理実装
}
```

上記のメソッドは、BaseService クラス内の抽象メソッドであるため、たとえ実装する処理がない場合でもサービスクラス内で実装する必要がある。

3.4.2 クラスの配置

サービスクラスは、本システムで任意の場所に配置する必要がある。

本システムでは、コントロールサーバ起動時に動的にサービスクラスを読み込み、サービスを開始する。

そのため、サービス機能を実装したサービスクラスは、クラスファイルを"service"ディレクトリ下に配置する。

3.5 固有部クラス

固有部クラスとは、利用者がある特定のシミュレーションを実行するために、シミュレーション処理を実装したクラスである。

3.5.1 実装方法

固有部クラスは、コントロールサーバがシミュレーション実行要求を受け付けた際に動的に読み込まれる。

そのために、SimulatorImpl クラスを継承するクラスである必要がある。

パッケージは、"jp.go.jaeri.apr.simulation"パッケージとする。

```
public class 固有部クラス名 extends SimulatorImpl {
    //実装
}
```

固有部クラスは、以下のメソッドを実装する必要がある。

(1)init メソッド

init メソッドは、固有部クラスの初期処理を行うクラスである。
例えば、入力情報の読込などシミュレーションを行うために必要な準備処理

を実装する。

```
public JobParam init(String jobId) throws SimulatorException {
    //初期処理実装
}
```

(2)validate メソッド

validate メソッドは、シミュレーションのための確認処理を行う。

例えば、シミュレーションを行う超並列計算機への資源の配置などの処理を行う。

```
public JobParam invoke(JobParam param) throws SimulatorException {
    //確認処理実装
}
```

(3)invoke メソッド

invoke メソッドは、シミュレーション自身を実行する処理を実装する。

```
public JobParam invoke(JobParam param) throws SimulatorException {
    //シミュレーション処理実装
}
```

(4)cleanup メソッド

cleanup メソッドは、固有部クラスの終了処理を行うクラスである。

例えば、固有部クラス内で使用した資源の解放など、後処理を実装する。

```
public void cleanup(JobParam param) throws SimulatorException {
    //終了処理実装
}
```

(5)cancel メソッド

cancel メソッドは、固有部クラス内で実装したシミュレーション処理に対して割り込みで処理を中断するメソッドである。

```
public void cancel(String jobId) throws SimulatorException {
    //キャンセル処理実装
}
```

(6)stat メソッド

stat メソッドは、固有部クラス内で実装したシミュレーション処理の実行状況を通知するクラスである。

```
public Result stat(String jobId) throws SimulatorException {
```

```
        //ステータス処理実装
    }
```

3.5.2 クラスの配置

固有部クラスは、本システムで任意の場所に配置する必要がある。

本システムでは、コントロールサーバがシミュレーション実行要求を受け付けた際に動的に固有部クラスを読み込み、シミュレーションを実行する。

そのため、シミュレーション処理を実装した固有部クラスは、クラスファイルを"job"ディレクトリ下に配置する。

3.6 定義ファイル

3.6.1 コントロールサーバ定義ファイル

(1)ファイル名

"controlserver.xml"

(2)書式

指定書式には、以下の２パターンがある。

1) 開始。終了タグ指定

<タグ名　パラメータ名＝値　・・・　＞・・・</タグ名>

2) 属性指定

<タグ名　パラメータ名＝値　/>

(3)定義情報

コントロールサーバ定義ファイルの主なタグ内容は以下の通りである。

（凡例）タグの省略

○：省略可能　×：省略不可　－：対象外

（凡例）複数指定

○：複数指定可　×：複数指定不可　－：対象外

表 3-2　コントロールサーバ定義ファイルのタグとその説明

タグ名	タグの説明	省略	複数指定
ControlServer	コントロールサーバ定義ファイルの開始と終了を定義する	×	×
Server	コントロールサーバが HORB サーバとして起動する情報を定義する	×	×
Job	コントロールサーバがジョブを制御するときの情報を定義する	×	×

(4)タグ階層

各タグは階層化されており、下位階層のタグは、上位階層の開始タグと終了タグの中に記述する。

コントロールサーバ定義ファイルのタグ階層を以下に示す。

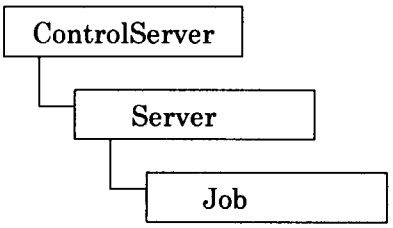

図 3-1 コントロールサーバ定義ファイルのタグ階層図

各タグの属性値について説明する。

1)Server タグ

Server タグは、コントロールサーバが HORB サーバとして動作するための情報を保持する。

表 3-3 Server タグの属性とその説明

属性	説明	属性の省略
name	HORB サーバとして起動したときのリモートオブジェクト名を定義する	○："daemon"
Port	HORB サーバとして起動したときのポート番号を定義する	○：8080
logLevel	コントロールサーバが出力するログの出力レベル “error”：通常レベル。エラー時に出力する “debug”：デバッグレベル	○："error"

2)Job タグ

Job タグは、コントロールサーバがシミュレーション実行要求を受け付けた際に、そのジョブを制御する。

表 3-4 Job タグとその説明

属性	説明	属性の省略
Interval	計算コード実行中に、その進行状況を監視するための間隔 単位は、秒	○：5
connectionTimeout	ジョブの実行中にコントロールサーバから他サーバへ接続した際のタイムアウト時間 単位は、ミリ秒	○：60

3.6.2 サーバ監視定義ファイル

(1)ファイル名

"servermanager.xml"

(2)書式

指定書式には、以下の２パターンがある。

1) 開始。終了タグ指定

<タグ名　パラメータ名＝値　・・・　＞・・・</タグ名>

2) 属性指定

<タグ名　パラメータ名＝値　/>

(3)定義情報

コントロールサーバ定義ファイルの主なタグ内容は以下の通りである。

（凡例）タグの省略

○：省略可能　×：省略不可　－：対象外

（凡例）複数指定

○：複数指定可　×：複数指定不可　－：対象外

表 3-5　サーバ監視定義ファイルのタグとその説明

タグ名	タグの説明	省略	複数指定
ServerManager	サーバ監視定義ファイルの開始と終了を定義する	×	×
Manager	サーバ監視サービスがサーバを監視する間隔を定義する	○	×
Server	サーバ監視サービスが監視するサーバの種類を定義する	×	○
Host	サーバ監視サービスが監視するサーバについての情報を定義する	×	×

(4)タグ階層

各タグは階層化されており、下位階層のタグは、上位階層の開始タグと終了タグの中に記述する。サーバ監視定義ファイルのタグ階層を以下に示す。

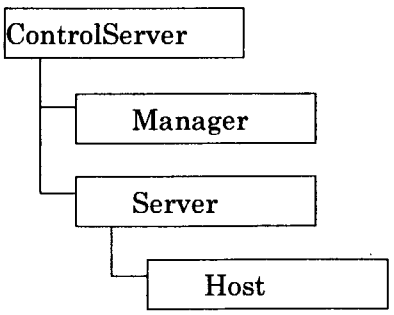

図 3-2　サーバ監視定義ファイルのタグ階層図

各タグの属性値について説明する。

1)Manager タグ

Manager タグは、サーバ監視サービスが他のサーバを監視する間隔を保持する。

表 3-6 Manager タグの属性とその説明

属性	説明	属性の省略
Interval	サーバ監視サービスが監視すべきサーバに対してハートビートを行う間隔を定義する 単位は秒	○：60

2)Server タグ

Server タグは、サーバ監視サービスが監視するサーバの種類を保持する。

表 3-7 Server タグの属性とその説明

属性	説明	属性の省略
Type	サーバ監視サービスが監視するサーバの種類を定義する	×

3)Host タグ

Host タグは、サーバ監視サービスが監視の対象とするホスト情報を保持する。

表 3-8 Host タグの属性とその説明

属性	説明	属性の省略
Name	サーバ監視サービスが監視の対象とするサーバのホスト名を定義する	×
Port	サーバ監視サービスが監視の対象とするサーバのポート番号を定義する	×
Instance	サーバ監視サービスが監視の対象とするサーバのインスタンス名を定義する 対象サーバが HORB サーバとして起動する場合に必要となる	○："" 設定がない場合は HORB サーバでない。

3.7 ログ

コントロールサーバは、設定によりログを出力する。

3.7.1 ログファイル名

コントロールサーバが出力するログファイルは、２種類ある。

(1)コントロールサーバログファイル

コントロールサーバのデーモンが出力するログファイルである。

control.YYYY-MM-DD.log
① ② ③

①："control"固定
②：YYYY-MM-DD ⇒ 西暦-月-日
③："log"固定

(2)ジョブログファイル

ジョブ単位に出力するログファイルである。

ジョブ名.YYYY-MM-DD.log
① ② ③

①：ジョブ ID
②：YYYY-MM-DD ⇒ 西暦-月-日
③："log"固定

3.7.2 出力フォーマット

ログは、以下のフォーマットで出力する。

YYYY-MM-DD hh:mm:ss メッセージ
① ②

①YYYY-MM-DD hh:mm:ss ⇒ 年月日 時分秒
②メッセージ ⇒ メッセージ本文

※出力イメージ

```
2002-06-28 11:45:21 ControlDaemon.init start.
2002-06-28 11:45:21 read controlserver.xml ----------
2002-06-28 11:45:21 serverName=daemon
2002-06-28 11:45:21 serverPort=8080
2002-06-28 11:45:21 jobWatchInterval=5
2002-06-28 11:45:21 connectionTimeout=60000
2002-06-28 11:45:21 logLevel=1
2002-06-28 11:45:21 read controlserver.xml ----------
```

3.7.3 ログの実装

ログの出力は、サービスクラスや固有部クラスの実装時に行う。コントロールサーバは４種類の出力方法を提供する。

(1) public void log(String msg);
"error"レベルでログを出力するためのインタフェース。

(2) public void log(String msg, Throwable th);
異常発生時、異常情報を出力するためのインタフェース。

(3) public void log(String msg, int level);
ログレベルを指定してログを出力するためのインタフェース。
指定したレベルがコントロールサーバ定義ファイルに定義したログレベルより低い場合は出力されない。

3.8 インタフェース

コントロールサーバは、HORB サーバとして起動し、HORB サーバが提供するリモートオブジェクトを介して Web サーバからの要求を受け付ける。

実際には、リモートオブジェクトに対するメソッド呼び出しという形で各種要求を行うことができる。

以下に、Web サーバからの要求の呼び出し方法について記述する。

(1)ジョブ実行要求
ジョブの実行要求を行う。

public synchronized void request(String jobId, String[ ] jobClass);

○引数
・jobId　　String　ジョブ ID
　　システム内でユニークな文字列であること
・jobClass　String[ ]　ジョブ固有部実装クラスのクラス名群
　　フルパッケージであること

○復帰値
なし

(2)ジョブ停止要求
実行中ジョブの停止要求を行う。既に停止しているジョブ、存在しないジョブを指定した場合、なにも行われない。

public synchronized void cancel(String jobId);

○引数

・jobId　　String　停止対象ジョブ ID

○復帰値

なし

(3)ジョブステータス確認要求

実行中ジョブの稼動状況確認を行う。

public synchronized Result jobStatus(String jobId);

○引数

・jobId　　String　確認対象ジョブ ID

○復帰値

・Result クラスのインスタンス

(4)ジョブ異常情報取得要求

実行ジョブの異常結果情報を取得する。

public synchronized Exception getError(String jobId);

○引数

・jobId　　String　ジョブ ID

○復帰値

・Exception クラスのインスタンス

(5)コントロールサーバ停止

HORB サーバとして起動するコントロールサーバを停止する。

public synchronized void stop();

○引数

なし

○復帰値

なし

4. おわりに

今回の開発では P-cube をシミュレーション・コードの対象としていたが、今後、本システムに対応するシミュレーション・コードを追加していく予定である。

しかし、シミュレーション・コードを追加するたびにサービス形態・インタフェースを再構築するということは開発効率の悪化を招くため、その分、利用者への公開時期が遅れることとなる。

そのため、本報告書で紹介したように、今回、分散コンピューティング・システムの核となるコントロールサーバに的を絞り、汎用性・拡張性のあるシステムに再構築すべく、開発・整備作業を行った。

本システムは未だ開発途上であり、今後の展開について述べる。

今後は対応シミュレーション・コードの拡充を図り、研究者にとって研究活動の促進となるシステムおよびサービス形態について開発・整備していく。

既に ITBL 基盤ソフトとして TME や RIS などが開発されているが、機能によっては、シミュレーション・コードは Stampi を用いて開発される必要があるなど制約・制限がある。

また、分散コンピューティング・システムはイベントドリブンタイプであり、一方向の流れ（flow 型）で処理を行う TME とは異なるが、今後、TME がイベントドリブンタイプになるときには、分散コンピューティング・システムは TME から呼び出される単なるプログラム群になると思われるため、現在、その共通性のある TME の呼び出しプログラム群を蓄積することも念頭に入れて整備を行っていく。

また、超並列計算機だけでなく、セミナー室などの PC 群も計算資源として利用できるよう検討するとともに、研究会等の場において研究者が意見を交換し合い、その場に居ながらリアルタイムでデータの可視化・解析を行えるような環境の整備についても模索していく予定である。

ITBL 基盤ソフトとの統合の後、ITBL 環境にて利用できるシステムとなれば、多くの研究者にとって、その研究活動が促進されることになると期待している。

謝辞

本報告書の執筆の機会を与えて下さったITBL利用推進室の相川室長、ITBL利用推進室の皆様に感謝いたします。また、分散コンピューティング・システムの構築において、ご協力・ご助言を頂きました(財)高度情報科学技術研究機構[5]、(株)富士通ソーシアルサイエンスラボラトリ[6]、(株)神戸デジタル・ラボ[7]、光量子科学研究センターの皆様ならびにITBL利用推進室の皆様に感謝いたします。

参考文献

1）計算科学技術推進センター　http://www2.tokai.jaeri.go.jp/ccse/index-j.html
2）ITBL利用推進室　http://wwwitblpg.apr.jaeri.go.jp/itblpg/index.html
3）関西研究所　光量子科学研究センター　http://wwwapr.apr.jaeri.go.jp/
4）ITBLプロジェクト　http://www.itbl.jp/
5）(財)高度情報科学技術研究機構　http://www.rist.or.jp
6）(株)富士通ソーシアルサイエンスラボラトリ　http://www.ssl.fujitsu.com
7）(株)神戸デジタル・ラボ　http://www.kdl.co.jp

国際単位系 (SI) と換算表

表 1　SI 基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表 2　SI と併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表 5　SI 接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^{9}	ギガ	G
10^{6}	メガ	M
10^{3}	キロ	k
10^{2}	ヘクト	h
10^{1}	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

表 3　固有の名称をもつ SI 組立単位

量	名称	記号	他の SI 単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	m·kg/s^2
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表 4　SI と共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

（注）

1. 表 1 − 5 は「国際単位系」第 5 版，国際度量衡局 1985 年刊行による。ただし，1 eV および 1 u の値は CODATA の 1986 年推奨値によった。
2. 表 4 には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. bar は，JIS では流体の圧力を表わす場合に限り表 2 のカテゴリーに分類されている。
4. EC 閣僚理事会指令では bar，barn および「血圧の単位」mmHg を表 2 のカテゴリーに入れている。

換　算　表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062 × 10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10^{-4}	1.35951 × 10^{-3}	1.31579 × 10^{-3}	1	1.93368 × 10^{-2}
	6.89476 × 10^{-3}	7.03070 × 10^{-2}	6.80460 × 10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778 × 10^{-7}	0.238889	9.47813 × 10^{-4}	0.737562	6.24150 × 10^{18}
	9.80665	1	2.72407 × 10^{-6}	2.34270	9.29487 × 10^{-3}	7.23301	6.12082 × 10^{19}
	3.6 × 10^6	3.67098 × 10^5	1	8.59999 × 10^5	3412.13	2.65522 × 10^6	2.24694 × 10^{25}
	4.18605	0.426858	1.16279 × 10^{-6}	1	3.96759 × 10^{-3}	3.08747	2.61272 × 10^{19}
	1055.06	107.586	2.93072 × 10^{-4}	252.042	1	778.172	6.58515 × 10^{21}
	1.35582	0.138255	3.76616 × 10^{-7}	0.323890	1.28506 × 10^{-3}	1	8.46233 × 10^{18}
	1.60218 × 10^{-19}	1.63377 × 10^{-20}	4.45050 × 10^{-26}	3.82743 × 10^{-20}	1.51857 × 10^{-22}	1.18171 × 10^{-19}	1

1 cal = 4.18605 J (計量法)

= 4.184 J (熱化学)

= 4.1855 J (15 °C)

= 4.1868 J (国際蒸気表)

仕事率　1 PS (仏馬力)

= 75 kgf·m/s

= 735.499 W

放射能	Bq	Ci
	1	2.70270 × 10^{-11}
	3.7 × 10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58 × 10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

(86 年 12 月 26 日現在)

分散コンピューティング・システム整備 ―コントロールサーバの開発―