JAERI-Data/Code
2000-007

異機種並列計算機間通信ライブラリ Stampi 設計書

2000年3月

今村俊幸・武宮　博・小出　洋

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 2000－007

異機種並列計算機間通信ライブラリ Stampi 設計書

日本原子力研究所計算科学技術推進センター
今村 俊幸・武宮 博・小出 洋

(2000 年 1 月 24 日 受理)

計算科学技術推進センターでは，異なる並列計算機やワークステーションを任意に組み合わせ計算を行うために，MPI に基づいた異機種並列計算機間通信ライブラリ Stampi を開発した．現在，Stampi システムは Stampi 並びに Stampi/Java という形で構成されており，複合並列計算機 COMPACS のみならず WWW ブラウザ上で動作する Applet との通信も可能になっている．本報告書は，開発した Stampi システムの設計方針並びに詳細仕様等をとりまとめたものである．

日本原子力研究所: 〒 153-0061 東京都目黒区中目黒 2-2-54

JAERI－Data/Code 2000－007

The Specification of Stampi, A Message Passing Library for Distributed Parallel Computing

Toshiyuki IMAMURA, Hiroshi TAKEMIYA and Hiroshi KOIDE

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received January 24, 2000)

At CCSE, Center for Promotion of Computational Science and Engineering, a new message passing library for heterogeneous and distributed parallel computing has been developed, and it is called as Stampi. Stampi enables us to communicate between any combination of parallel computers as well as workstations. Currently, a Stampi system is constructed from Stampi library and Stampi/Java. It provides functions to connect a Stampi application with not only those on COMPACS, COMplex Parallel Computer System, but also applets which work on WWW browsers. This report summarizes the specifications of Stampi and details the development of its system.

Keywords : Specifications, Distributed Parallel Computing, Communication Library, MPI, Dynamic Process Creation, External Communication.

目次

はじめに 1

1 概要 2
1.1 システムの概要 2
1.1.1 Stampi の概要 2
1.1.2 Stampi/Java の概要 3
1.2 本報告書の目的 3

2 動作環境 4
2.1 計算機環境 4
2.2 利用者プログラム 6

3 機器構成 7
3.1 計算機環境 7
3.2 システム構成 7

4 機能 9
4.1 機能一覧 9
4.2 Stampi プログラム起動機能 10
4.3 異機種間通信路確立機能 11
4.3.1 Slave プロセス生成機能 11
4.3.2 通信路情報設定機能 14
4.3.3 プロセス情報参照機能 15
4.3.4 Server ポートオープンクローズ機能 16
4.3.5 ポート接続受け付け機能 17
4.3.6 ポート接続機能 19
4.3.7 ポート切断機能 20
4.4 異機種間通信機能 21
4.5 同一機種内通信機能 23
4.6 バッチ連携機能 24
4.7 異機種間通信中継機能 25
4.8 Stampi/Java 機能 26
4.8.1 Stampi/Java 起動機能 26
4.8.2 Stampi/Java 関数機能 27
4.8.3 Stampi/Java 終了機能 27

5 インタフェース 28
5.1 ハードウェアインタフェース 28
5.2 ソフトウェアインタフェース 28
5.3 MPI-2 関数 30
5.3.1 MPI_Comm_spawn 31
5.3.2 MPI_Comm_get_parent 32
5.3.3 MPI_Open_port 32

5.3.4 MPI_Close_port . . . 33
5.3.5 MPI_Comm_accept . . . 34
5.3.6 MPI_Comm_connect . . . 35
5.3.7 MPI_Comm_disconnect . . . 36
5.3.8 MPI_Info_create . . . 36
5.3.9 MPI_Info_set . . . 37
5.4 異機種間利用可能 MPI-1 関数 . . . 38
5.4.1 MPI_Send . . . 39
5.4.2 MPI_Recv . . . 40
5.4.3 MPI_Get_count . . . 41
5.4.4 MPI_Isend . . . 42
5.4.5 MPI_Irecv . . . 43
5.4.6 MPI_Wait . . . 44
5.4.7 MPI_Probe . . . 45
5.4.8 MPI_Iprobe . . . 46
5.4.9 MPI_Comm_remote_size . . . 46
5.4.10 MPI_Intercomm_merge . . . 47
5.4.11 MPI_Barrier . . . 48
5.4.12 MPI_Bcast . . . 49
5.4.13 MPI_Gather . . . 50
5.4.14 MPI_Gatherv . . . 51
5.4.15 MPI_Scatter . . . 52
5.4.16 MPI_Scatterv . . . 53
5.4.17 MPI_Reduce . . . 54
5.4.18 MPI_Allreduce . . . 55
5.4.19 MPI_Init . . . 56
5.4.20 MPI_Finalize . . . 56
5.4.21 MPI_Abort . . . 57
5.5 同一機種内利用 MPI-1 関数 . . . 58
5.5.1 MPI_Comm_size . . . 59
5.5.2 MPI_Comm_rank . . . 59
5.5.3 MPI_Comm_dup . . . 60
5.5.4 MPI_Comm_free . . . 60
5.5.5 MPI_Comm_split . . . 61
5.5.6 MPI_Sendrecv . . . 62
5.5.7 MPI_Type_extent . . . 63
5.5.8 MPI_Type_vector . . . 63
5.5.9 MPI_Type_hvector . . . 64
5.5.10 MPI_Type_commit . . . 64
5.5.11 MPI_Cart_create . . . 65
5.5.12 MPI_Cart_shift . . . 66
5.5.13 MPI_Cart_coords . . . 67
5.5.14 MPI_Wtime . . . 67
5.6 Stampi/Java インタフェース . . . 68

5.6.1 Stampi/Java クラス 68
5.6.2 Stampi クラス 68
5.6.3 MPI_Comm クラス 69
5.6.4 MPI_Intracomm クラス 69
5.6.5 MPI_Intercomm クラス 69
5.6.6 MPI_Datatype クラス 73
5.6.7 MPI_Info クラス 73
5.6.8 MPI_Status クラス 73
5.7 Stampi プロトコル仕様 74
5.7.1 接続要求パケット 75
5.7.2 送信開始要求 76
5.7.3 受信開始要求 77
5.7.4 データ転送パケット 78
5.7.5 バリア同期パケット 78
5.7.6 SPAWN 開始パケット 79
5.7.7 SPAWN info パケット 80
5.7.8 ACK パケット 80
5.7.9 NACK パケット 81
5.8 Stampi/Java プロトコル仕様 81
5.8.1 Send プロトコル 82
5.8.2 Recv プロトコル 83
5.8.3 Info_create プロトコル 84
5.8.4 Info_set プロトコル 85
5.8.5 Spawn プロトコル 86

6 処理方式 87
6.1 Stampi 起動コマンド 87
6.2 通信制御プログラム 88
6.3 NQS 起動コマンド 89
6.4 MPI 関数処理方式 90
6.4.1 MPI_COMM_SPAWN 90
6.4.2 MPI_COMM_GET_PARENT 91
6.4.3 MPI_OPEN_PORT 91
6.4.4 MPI_CLOSE_PORT 92
6.4.5 MPI_COMM_ACCEPT 92
6.4.6 MPI_COMM_CONNECT 93
6.4.7 MPI_COMM_DISCONNECT 93
6.4.8 MPI_SEND 94
6.4.9 MPI_RECV 94
6.4.10 MPI_GET_COUNT 95
6.4.11 MPI_ISEND 95
6.4.12 MPI_IRECV 96
6.4.13 MPI_WAIT 96
6.4.14 MPI_BARRIER 97

6.4.15 MPI_INIT 97
6.4.16 MPI_FINALIZE 98
6.4.17 MPI_ABORT 98
6.4.18 MPI_COMM_SIZE 99
6.4.19 MPI_COMM_RANK 99
6.4.20 MPI_COMM_REMOTE_SIZE 100
6.4.21 MPI_COMM_DUP 100
6.4.22 MPI_COMM_FREE 101
6.4.23 MPI_COMM_SPLIT 101
6.4.24 MPI_SENDRECV 102
6.4.25 MPI_TYPE_EXTENT 102
6.4.26 MPI_TYPE_VECTOR 103
6.4.27 MPI_TYPE_HVECTOR 103
6.4.28 MPI_TYPE_COMMIT 104
6.4.29 MPI_BCAST 104
6.4.30 MPI_REDUCE 105
6.4.31 MPI_ALLREDUCE 105
6.4.32 MPI_CART_CREATE 106
6.4.33 MPI_CART_SHIFT 106
6.4.34 MPI_CART_COORDS 107
6.4.35 MPI_GATHER 107
6.4.36 MPI_GATHERV 108
6.4.37 MPI_WTIME 108
6.5 Stampi/Java通信中継モジュール 109
6.6 Stampi/Javaライブラリ 109

7 データ及びファイル 110
7.1 内部データ 110
7.1.1 ポート管理テーブル 110
7.1.2 コミュニケータ管理テーブル 111
7.1.3 非同期通信要求ハンドル管理テーブル 112
7.2 MPI_COMM_SPAWNに引き渡すデータ 113
7.3 サブプロセス起動ファイル 113

おわりに 114

謝辞 114

参考文献 114

Contents

Introduction 1

1 Overviews 2
1.1 Stampi Systems 2
1.1.1 Stampi Library 2
1.1.2 Stampi/Java 3
1.2 Objectives of This Report 3

2 Target Platforms 4
2.1 Resources and Environment 4
2.2 User's Program 6

3 System Configurations 7
3.1 Hardware Requirement of Stampi 7
3.2 Software Components of Stampi 7

4 Functions of Stampi System 9
4.1 Table of Functions 9
4.2 Startup of Stampi Programs 10
4.3 Establishment of Communication Path between Machines 11
4.3.1 Creation of Slave Processes 11
4.3.2 Specification of Communication Path 14
4.3.3 Reference of Process Information 15
4.3.4 Opening or Closing Server Port 16
4.3.5 Accepting Port Connections 17
4.3.6 Connecting Port 19
4.3.7 Closing Port 20
4.4 Communication between Processes on Different Machines 21
4.5 Communication between Processes on the Same Machine 23
4.6 Correlating with Batch Systems 24
4.7 Relaying Messages between Machines 25
4.8 Stampi/Java 26
4.8.1 Startup Stampi/Java 26
4.8.2 Procedure of Stampi/Java 27
4.8.3 Closing of Stampi/Java 27

5 Interfaces 28
5.1 Hardware Interfaces 28
5.2 Software Interfaces 28
5.3 MPI-2 Functions 30
5.3.1 `MPI_Comm_spawn` 31
5.3.2 `MPI_Comm_get_parent` 32
5.3.3 `MPI_Open_port` 32

5.3.4 MPI_Close_port . . . 33
5.3.5 MPI_Comm_accept . . . 34
5.3.6 MPI_Comm_connect . . . 35
5.3.7 MPI_Comm_disconnect . . . 36
5.3.8 MPI_Info_create . . . 36
5.3.9 MPI_Info_set . . . 37
5.4 MPI-1 Functions, Available on the Inter-machine Communication . . . 38
5.4.1 MPI_Send . . . 39
5.4.2 MPI_Recv . . . 40
5.4.3 MPI_Get_count . . . 41
5.4.4 MPI_Isend . . . 42
5.4.5 MPI_Irecv . . . 43
5.4.6 MPI_Wait . . . 44
5.4.7 MPI_Probe . . . 45
5.4.8 MPI_Iprobe . . . 46
5.4.9 MPI_Comm_remote_size . . . 46
5.4.10 MPI_Intercomm_merge . . . 47
5.4.11 MPI_Barrier . . . 48
5.4.12 MPI_Bcast . . . 49
5.4.13 MPI_Gather . . . 50
5.4.14 MPI_Gatherv . . . 51
5.4.15 MPI_Scatter . . . 52
5.4.16 MPI_Scatterv . . . 53
5.4.17 MPI_Reduce . . . 54
5.4.18 MPI_Allreduce . . . 55
5.4.19 MPI_Init . . . 56
5.4.20 MPI_Finalize . . . 56
5.4.21 MPI_Abort . . . 57
5.5 MPI-1 Functions, Available on the Internal-machine Communication . . . 58
5.5.1 MPI_Comm_size . . . 59
5.5.2 MPI_Comm_rank . . . 59
5.5.3 MPI_Comm_dup . . . 60
5.5.4 MPI_Comm_free . . . 60
5.5.5 MPI_Comm_split . . . 61
5.5.6 MPI_Sendrecv . . . 62
5.5.7 MPI_Type_extent . . . 63
5.5.8 MPI_Type_vector . . . 63
5.5.9 MPI_Type_hvector . . . 64
5.5.10 MPI_Type_commit . . . 64
5.5.11 MPI_Cart_create . . . 65
5.5.12 MPI_Cart_shift . . . 66
5.5.13 MPI_Cart_coords . . . 67
5.5.14 MPI_Wtime . . . 67
5.6 Interfaces of Stampi/Java . . . 68

5.6.1 Stampi/Java Class . . . 68
5.6.2 Stampi Class . . . 68
5.6.3 MPI_Comm Class . . . 69
5.6.4 MPI_Intracomm Class . . . 69
5.6.5 MPI_Intercomm Class . . . 69
5.6.6 MPI_Datatype Class . . . 73
5.6.7 MPI_Info Class . . . 73
5.6.8 MPI_Status Class . . . 73
5.7 Specifications of the Stampi Protocols . . . 74
5.7.1 Connection Request Packet . . . 75
5.7.2 Request of Start-Of-Send . . . 76
5.7.3 Request of Start-Of-Recv . . . 77
5.7.4 Data Communication Packet . . . 78
5.7.5 Barrier Synchronization Packet . . . 78
5.7.6 SPAWN start Packet . . . 79
5.7.7 SPAWN info Packet . . . 80
5.7.8 ACK Packet . . . 80
5.7.9 NACK Packet . . . 81
5.8 Specifications of the Stampi/Java Protocols . . . 81
5.8.1 Send Protocol . . . 82
5.8.2 Recv Protocol . . . 83
5.8.3 Info_create Protocol . . . 84
5.8.4 Info_set Protocol . . . 85
5.8.5 Spawn Protocol . . . 86

6 Details of Processing . . . 87
6.1 Stampi Startup Command . . . 87
6.2 Communication Control Program . . . 88
6.3 NQS Startup Command . . . 89
6.4 Processing of MPI Functions . . . 90
6.4.1 MPI_COMM_SPAWN . . . 90
6.4.2 MPI_COMM_GET_PARENT . . . 91
6.4.3 MPI_OPEN_PORT . . . 91
6.4.4 MPI_CLOSE_PORT . . . 92
6.4.5 MPI_COMM_ACCEPT . . . 92
6.4.6 MPI_COMM_CONNECT . . . 93
6.4.7 MPI_COMM_DISCONNECT . . . 93
6.4.8 MPI_SEND . . . 94
6.4.9 MPI_RECV . . . 94
6.4.10 MPI_GET_COUNT . . . 95
6.4.11 MPI_ISEND . . . 95
6.4.12 MPI_IRECV . . . 96
6.4.13 MPI_WAIT . . . 96
6.4.14 MPI_BARRIER . . . 97

6.4.15 MPI_INIT . . . 97
6.4.16 MPI_FINALIZE . . . 98
6.4.17 MPI_ABORT . . . 98
6.4.18 MPI_COMM_SIZE . . . 99
6.4.19 MPI_COMM_RANK . . . 99
6.4.20 MPI_COMM_REMOTE_SIZE . . . 100
6.4.21 MPI_COMM_DUP . . . 100
6.4.22 MPI_COMM_FREE . . . 101
6.4.23 MPI_COMM_SPLIT . . . 101
6.4.24 MPI_SENDRECV . . . 102
6.4.25 MPI_TYPE_EXTENT . . . 102
6.4.26 MPI_TYPE_VECTOR . . . 103
6.4.27 MPI_TYPE_HVECTOR . . . 103
6.4.28 MPI_TYPE_COMMIT . . . 104
6.4.29 MPI_BCAST . . . 104
6.4.30 MPI_REDUCE . . . 105
6.4.31 MPI_ALLREDUCE . . . 105
6.4.32 MPI_CART_CREATE . . . 106
6.4.33 MPI_CART_SHIFT . . . 106
6.4.34 MPI_CART_COORDS . . . 107
6.4.35 MPI_GATHER . . . 107
6.4.36 MPI_GATHERV . . . 108
6.4.37 MPI_WTIME . . . 108
6.5 Relay Module of Stampi/Java Communication . . . 109
6.6 Stampi/Java Library . . . 109

7 Data and Files . . . 110
7.1 Specifications of Internal Data . . . 110
7.1.1 Management Table of Port . . . 110
7.1.2 Management Table of Communicator . . . 111
7.1.3 Management Table of Asynchronous Request Handle . . . 112
7.2 Configuration Data Handled in MPI_COMM_SPAWN . . . 113
7.3 Configuration File for Start up of Subprocesses . . . 113

Summary . . . 114

Acknowledgment . . . 114

References . . . 114

はじめに

計算機ならびにネットワーク技術の発展に伴い, ベクトル並列計算機, スカラ並列計算機等種々のアーキテクチャの並列計算機を同時に利用し高速で大規模な科学技術計算が可能となってきている. 原研においても計算科学技術推進センターに 5 台の並列計算機, 東海, 那珂, 関西研究所にそれぞれ高性能の並列計算機を計 10 数台保有しており, それら並列機は計算物理分野を中心として所内外の研究者によって利用されている.

従来, これら並列計算機は単体としての利用がなされてきたわけであるが, 各種並列計算機にはアーキテクチャに応じて有効かつ効率的に働く問題対象があることが過去の研究例から指摘されている (あえて文献を引用することもないであろう). また, 高速ネットワークの普及に従い, これら複数の並列計算機資源 (ここでは単に計算機だけではなく周辺の機器資源やデータベース等の情報資源も含めて) を連携させることで効率的な計算を実現する利点があるともいわれている.

上で述べた様な複数の異なる性質を持った計算機を結合させ効率的な計算を行うためには, 各並列計算機間で共通に利用可能となる通信基盤 (ライブラリ) が必要となる. 本報告書では, 計算科学技術推進センターが開発した異機種並列計算機間通信ライブラリ Stampi に関してその設計方針並びに詳細仕様をまとめている.

以下に本報告書の概要について記述する. Stampi システムの概要について第 1 章で, Stampi システム動作に必要な計算機環境を第 2 章で, 第 3 章では Stampi のシステム構成を述べている. さらに, 第 4 では Stampi が提供する機能一覧について, 第 5, 第 6 章では Stampi が提供する関数群について, それぞれそのインターフェイス仕様と実装処理方式をまとめている. 第 7 章では内部処理に用いるデータ仕様を述べている.

1 概要

1.1 システムの概要

1.1.1 Stampiの概要

異機種並列計算機間通信ライブラリ (以下, Stampiと称す) は, MPI (Message Passing Interface) [1, 2] を用いて異なる計算機で稼働する利用者プログラム間でMPI通信を行う機能を提供する.

Stampiの構成を, 図1.1に示す.

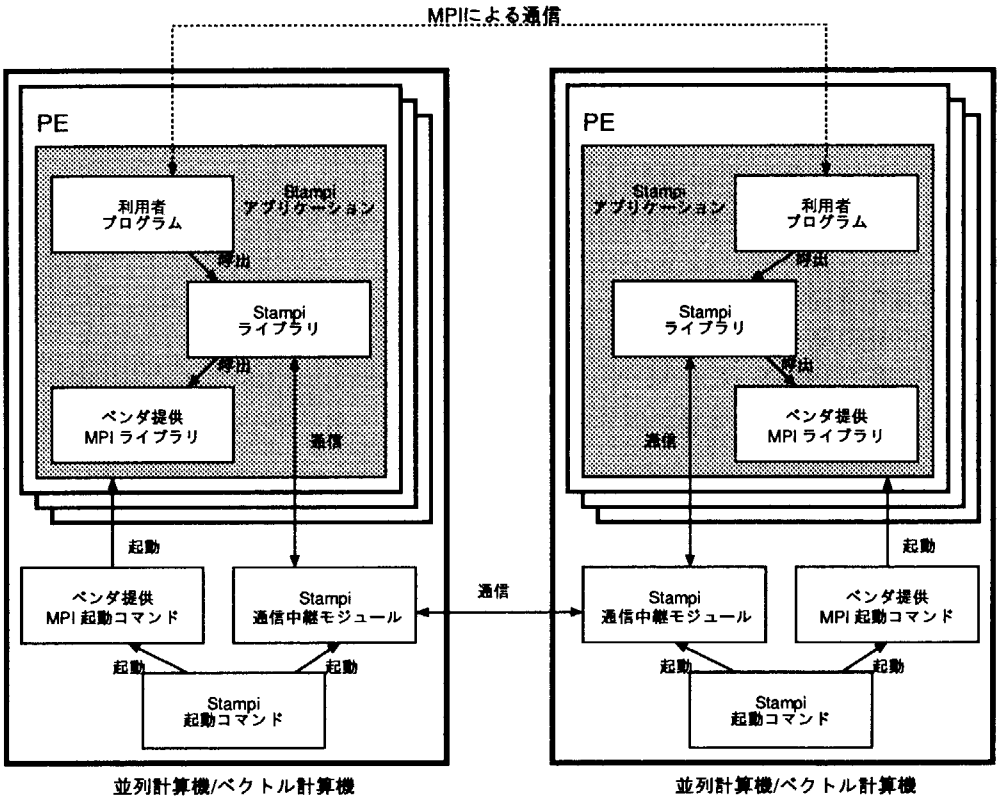

図 1.1: Stampiのシステム構成

利用者は, C言語またはFortran言語で記述した利用者プログラムをコンパイルし, 本システムが提供するStampiライブラリとリンクすることで, 実行形式の利用者プログラムを作成する (Stampiライブラリと結合した実行形式の利用者プログラムをStampiアプリケーションと呼ぶ). 利用者プログラムは, プログラムを実行するそれぞれの計算機に用意しておく.

その後, 本システムが提供するStampi起動コマンドを使用して利用者プログラムの実行を行うことにより, 複数の計算機間でMPI通信を用いた計算が可能になる.

尚, Stampiでは, 複数の計算機で稼働する利用者プログラム間でMPIによる通信路を確立する手段として, Master/Slave型の接続 (一方の計算機の利用者プログラム (Master) から, 他方の計算機の利用者プログラム (Slave) を起動する方式) と, Client/Server型の接続 (個別の計算機で独立して起動した利用者プログラム間で通信路を確立する方式) をサポートする.

1.1.2 Stampi/Java の概要

Stampi/Java は, Stampi の機能を WWW ブラウザ上で稼働する Java アプレット [5] に拡張するためのコンポーネントである.

Stampi/Java の構成を図 1.2に示す.

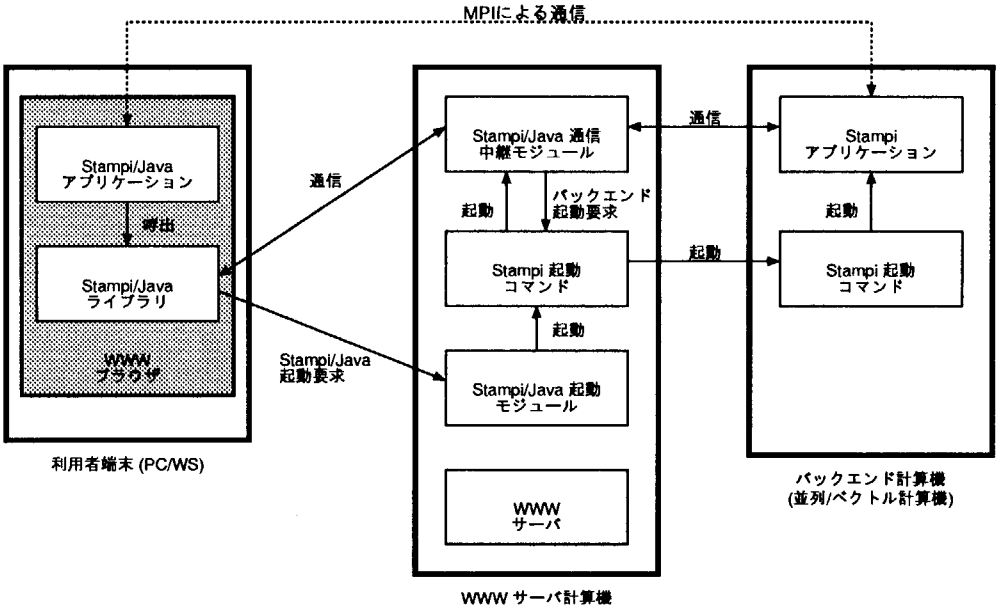

図 1.2: Stampi/Java のシステム構成

Stampi/Java では, WWW ブラウザで稼働する利用者作成の Java アプレット (Stampi/Java アプリケーション) と, 並列計算機・ベクトル計算機で稼働する利用者プログラム (Stampi アプリケーション) との間での MPI 通信をサポートする.

Stampi/Java アプリケーションは, 本システムが提供する Stampi/Java ライブラリを使用するように記述された利用者作成の Java プログラムであり, WWW サーバ計算機の所定のディレクトリに格納しておくことによって, WWW ブラウザに (Stampi/Java ライブラリと共に) ローディングされて実行することができる.

一般に Java アプレットは Java の制約により, WWW ブラウザ計算機としか通信することができない. このため Stampi/Java では, Stampi/Java 通信中継モジュールを WWW サーバ計算機で実行し, WWW ブラウザで稼働する Stampi/Java アプリケーションとバックエンド計算機で稼働する Stampi アプリケーションとの通信を中継することによって, 任意の並列計算機・ベクトル計算機との通信を可能にする. Stampi/Java 通信中継モジュールは, Stampi/Java ライブラリの初期化時にシステムが自動的に起動するため, 利用者はその存在を意識する必要はない.

1.2 本報告書の目的

本報告書では、 Stampi システムを構成する以下の項目を設計する.

(1) Stampi 本体の設計

(2) Stampi/Java の設計

2 動作環境

2.1 計算機環境

Stampiは, 以下に示す計算機環境で利用できる.

このうち, 動作環境13, 14, 15, 16の各機種では, Apacheなどの WWW サーバと組み合わせることにより, Stampi/Javaのサーバとなることができる.

各機種への実装に関しては, 各社MPIマニュアルを利用した [4, 6, 7, 8, 9, 10, 11].

- 動作環境 1:

ハードウェア	原研殿開発小型並列計算機, および, 日立製作所 SR2201
OS	HI-UX/MPP
Fortran	HI-UX/MPP 最適化 Fortran90
C	HI-UX/MPP 最適化 C
MPI	HI-UX/MPP MPI

- 動作環境 2:

ハードウェア	富士通株式会社 VPP300
OS	UXP/V
Fortran	UXP/V Fortran90/VP
C	UXP/V C/VP
MPI	UXP/V MPI

- 動作環境 3:

ハードウェア	日本電気株式会社 SX-4
OS	SUPER-UX
Fortran	SUPER-UX Fortran90/SX
C	SUPER-UX C/SX
MPI	SUPER-UX MPI/SX

- 動作環境 4:

ハードウェア	Cray Research, Inc. T94/4128
OS	UNICOS
Fortran	CF90
C	Cray Standard C
MPI	Message Passing Toolkit

- 動作環境 5:

ハードウェア	IBM CORP SP2
OS	AIX
Fortran	XL Fortran for AIX
C	C for AIX
MPI	IBM Parallel environment for AIX

- 動作環境 6:

ハードウェア	SGI Onyx
OS	IRIX
Fortran	MIPSpro Fortran90
C	MPISpro C
MPI	MPICH

- 動作環境 7:

ハードウェア	富士通 AP3000
OS	Solaris
Fortran	富士通 Fortran90
C	富士通 C
MPI	富士通 MPI

- 動作環境 8:

ハードウェア	富士通 VP2400
OS	UXP/M
Fortran	UXP/M Fortran77
C	UXP/M C
MPI	MPICH

- 動作環境 9:

ハードウェア	Intel Paragon
OS	OSF/1
Fortran	Intel Paragon Fortran77
C	Intel Paragon C
MPI	Intel Paragon MPI

- 動作環境 10:

ハードウェア	IBM SP2
OS	AIX
Fortran	XL Fortran for AIX
C	C for AIX
MPI	MPICH

- 動作環境 11:

ハードウェア	DEC Alpha
OS	DEC OSF/1
Fortran	DEC OSF/1 Fortran90
C	DEC OSF/1 C
MPI	MPICH

- 動作環境 12:

ハードウェア	DEC Alpha
OS	DEC OSF/1
Fortran	DEC OSF/1 Fortran90
C	DEC OSF/1 C
MPI	DEC MPI

- 動作環境 13:

ハードウェア	HP PA ワークステーション・サーバ
OS	HPUX
Fortran	HP Fortran
C	HP C
MPI	MPICH

- 動作環境 14:

ハードウェア	SUN Sparc ワークステーション・サーバ
OS	Solaris
Fortran	SUN Fortran90
C	SUN C
MPI	MPICH

- 動作環境 15:

ハードウェア	PC
OS	FreeBSD
Fortran	GNU Fortran
C	GNU C
MPI	MPICH

- 動作環境 16:

ハードウェア	PC
OS	Linux
Fortran	GNU Fortran
C	GNU C
MPI	MPICH

2.2 利用者プログラム

Stampi は以下のプログラム言語のいずれか, または, 以下のプログラム言語を組み合わせて記述された利用者プログラムから利用できる.

(1) Fortran (JIS X3001–1994 準拠)

(2) C (JIS X3010–1993 準拠)

また, Stampi/Java を使用する場合には, Java 言語で記述した利用者プログラムも利用できる.

3 機器構成

3.1 計算機環境

2.1 前提条件参照.

3.2 システム構成

Stampi 及び Stampi/Java システムの全体構成を図 3.1に示す.

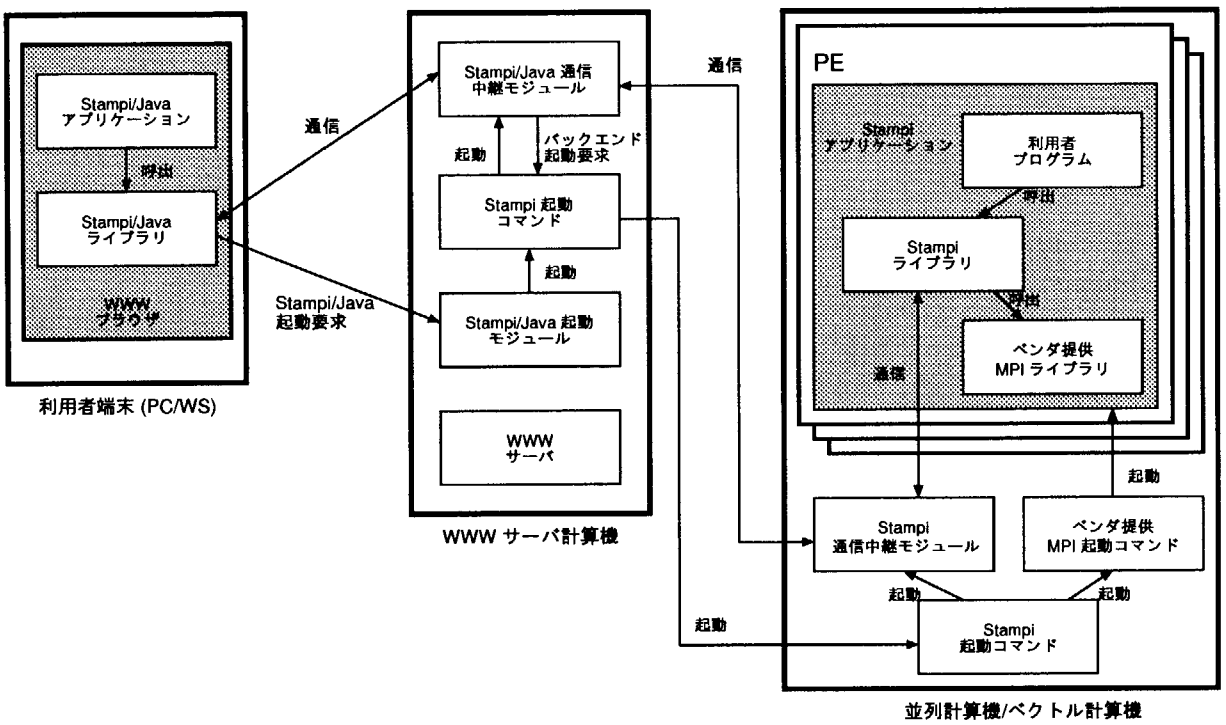

図 3.1: システム構成

本システムは, その計算機の通信性能を最大限に引き出すことを目的として, 同一計算機内の通信に対しては各計算機ベンダが提供する MPI を利用する.

(1) Stampi 起動コマンド
異機種並列計算機間での MPI 起動の制御を行う. Stampi 通信中継モジュールを起動し, 通信準備を行うとともに, ベンダ提供 MPI 起動コマンドを起動して利用者プログラムの実行を開始する. また, Client/Server 型通信でのサーバポートの管理や, Master/Slave 型通信での Slave 側プロセスの起動処理も行う.

(2) Stampi 通信中継モジュール
異機種並列計算機間での通信手順の調整, 及び, 通信の中継を行う. Stampi 通信中継モジュールは, 異機種並列計算機間の通信が必要となった場合に起動される. 異る計算機間の MPI プロセス間で直接通信できる場合には, Stampi 通信中継モジュールを経由せずに, MPI プロセス間で直接通信することもできる. この場合, Stampi 通信中継モジュールは起動されない.

(3) ベンダ提供 MPI 起動コマンド (ベンダ提供)
MPI を使用した利用者プログラムを起動する.

(4) 利用者プログラム, Stampi アプリケーション (利用者開発)
MPI を使用した利用者プログラムである. また, Stampi ライブラリを結合した利用者プログラムを特に Stampi アプリケーションと呼ぶ.

(5) Stampi ライブラリ
利用者プログラムから呼び出され, 通信処理を行う. 具体的には, 通信相手が自計算機か他計算機かを調べ, 自計算機の場合にはベンダ提供 MPI ライブラリを呼び出し, 他計算機の場合には他計算機との通信を行う. また, 利用者プログラムの呼び出しに応じて, 異なる計算機への MPI プロセスの生成や, Client/Server 型通信の Server ポートの作成や Client の接続を行う.

(6) ベンダ提供 MPI ライブラリ (ベンダ提供)
単一計算機内での MPI 通信を行う. 各計算機が備えるプロセッサ間通信の機構を用いて, 効率的に MPI 通信を行う.

(7) Stampi/Java アプリケーション (利用者開発)
Stampi/Java ライブラリを用いた Java プログラム (Java Applet) である.

(8) Stampi/Java ライブラリ
Java プログラムと並列計算機との MPI 通信を実現する Java クラスライブラリ.

(9) WWW ブラウザ (ベンダ提供)
Stampi/Java アプリケーションと, Stampi/Java ライブラリの実行環境である.

(10) WWW サーバ (ベンダ提供)
WWW ブラウザに対して Stampi/Java アプリケーションと Stampi/Java ライブラリの実体である Java class ファイルを転送する.

(11) Stampi/Java 起動モジュール
WWW サーバ計算機上で Stampi プロセスを起動する.

(12) Stampi/Java 通信中継モジュール
Stampi/Java アプリケーションと Stampi アプリケーションとの MPI 通信を中継する.

4 機能

4.1 機能一覧

システムの機能を表 4.1に示す.

表 4.1: 機能一覧

#	機能	概要
1	Stampi プログラム起動機能	Stampi を使用する利用者プログラムを起動する機能. Stampi 実行環境を整え, 利用者プログラムを起動する.
2	異機種間通信確立機能	Master/Slave 型, または, Client/Server 型の通信を行うために, 異なる 2 MPI プロセス間で通信路を確立する機能.
3	異機種間通信機能	異機種間通信路確立機能によって確立した通信路を用いて 2 MPI プロセス間の通信を行う機能. 実際の通信機能の他, 通信に付随する各種機能 (コミュニケータ操作など) を含む.
4	同一機種内通信機能	同一機種内の MPI 通信を実現する機能. 実際の通信機能の他, 同一機種内での通信に付随する各種機能 (コミュニケータ操作など) を含む.
5	バッチ連携機能	`MPI_Comm_spawn` で生成するプロセスを NQS のバッチキューに投入する機能. バッチの状態を監視し, TSS– バッチ間, バッチ – バッチ間の Stampi 連携を実現する.
6	異機種間通信中継機能	VPP や SP の各ノードプロセッサと他の計算機との間の通信を中継する機能.
7	Stampi/Java 機能	WWW ブラウザ上で稼働する Java アプレットで Stampi を利用した通信を実現する機能

4.2 Stampi プログラム起動機能

(1) 機能概要

Stampi プログラム起動機能は, Stampi ライブラリを使用して作成した利用者プログラムを起動する機能である. 利用者プログラムの呼び出しの他, 呼び出しの前処理として, 通信制御プログラムの起動や通信ポートの設定を行い, 利用者プログラムから MPI プロセスを起動したり, 他計算機と通信するための準備を行う.

また, 利用者プログラムを呼び出した後は, 利用者プログラム終了の待ち合わせと後処理を行う.

(2) 起動条件

Stampi 起動コマンドを実行した際, 起動される.

(3) インタフェース

Stampi プログラム起動機能のインタフェースを表 4.2に示す.

表 4.2: Stampi プログラム起動機能のインタフェース

#	入出力情報	内容	属性
1	利用者プログラム	利用者プログラムの名称とパラメタ	入力
2	ノード数	起動するプログラムで使用するノード数	入力
3	Master 通信ポート	Slave の場合, Master プロセスと通信するための Master 側通信ポート情報	入力
4	Master ノード数	Slave の場合, Master プロセスグループのノード数	入力

(4) 実現方法

Stampi プログラム起動機能は, 以下の処理を行うことにより実現する.

(a) 通信制御プログラム起動
通信制御プログラムを起動し, 異機種間通信を行う準備をする. また, 通信制御プログラムで設定した通信ポートの情報を受け取り, 利用者プログラムにポート情報を引き渡せるようにする.

(b) 利用者プログラム起動
ベンダ提供の MPI 起動コマンド経由で利用者プログラムを起動する. 起動に際しては, 通信制御プログラムから受け取った通信ポート情報を環境変数に設定し, 利用者プログラムでポートの情報を参照できるようにする.

(c) 利用者プログラム終了待ち合わせ
`wait` システムコールを使用して, 利用者プログラムの終了を待ち合わせる.

(d) 通信制御プログラム終了待ち合わせ
`wait` システムコールを使用して, 通信制御プログラムの終了を待ち合わせる.

4.3 異機種間通信路確立機能

異機種間通信路確立機能は, 表 4.3のサブ機能から構成される.

表 4.3: 異機種間通信路確立機能のサブ機能

#	機能	概要
1	Slaveプロセス生成機能	MPI_Comm_spawn 関数によって, Master/Slave 型通信の Slave となる MPI プロセスを生成する.
2	通信路情報設定機能	MPI_Info_create と MPI_Info_set 関数によって, 異機種間通信で使用する通信路に関する情報を設定する.
3	プロセス情報参照機能	MPI_Comm_get_parent 関数によって, Master/Slave 型通信のプロセス生成元コミュニケータを参照できる. また, MPI_Comm_remote_size 関数によって, 通信相手のコミュニケータのサイズを参照できる.
4	Server ポート オープンクローズ機能	MPI_Open_port 関数によって, Server のポートを開く. また, MPI_Close_port 関数によって, Server のポートを閉じる.
5	ポート接続受け付け機能	Server 側の MPI_Open_port で開いたポートに対する MPI_Comm_accept 関数によって, Client の MPI_Comm_connect 関数からの接続要求を受け付ける機能.
6	ポート接続機能	Client の MPI_Comm_connect 関数によって, Client 側から Server 側のポートに接続する.
7	ポート切断機能	MPI_Comm_disconnect 関数によって, MPI_Comm_accept または MPI_Comm_con nect で接続した通信路を切断する.

4.3.1 Slaveプロセス生成機能

(1) 概要

Master/Salve 型通信において, Slave となる新しい MPI プロセスを生成する.

(2) 起動条件

MPI_Comm_spawn 関数の呼び出しによって起動される.

(3) インタフェース

Slave プロセス生成機能のインタフェースを表 4.4に示す.

表 4.4: Slave プロセス生成機能のインタフェース

#	入出力情報	内容	属性
1	利用者プログラム	起動する MPI プロセスで実行する利用者プログラムとパラメタ	入力
2	ノード数	生成する MPI プロセスで実行するノード数	入力
3	Master コミュニケータ	プロセス生成元のコミュニケータ	入力
4	プロセス生成情報	生成するサブプロセスに関する情報．サブプロセス生成情報設定機能によって作成する情報	入力
5	Slave コミュニケータ	生成した新しい MPI プロセスのコミュニケータ	出力

(4) 実現方法

Slave プロセス生成機能は, 以下の処理を行うことにより実現する.

(a) `MPI_Comm_spawn` の処理

(i) MPI プロセス生成要求
`MPI_Comm_spawn` の root プロセスから Stampi 起動コマンドに対して MPI プロセス生成要求を行う．プロセス生成の結果 (プロセス生成の成否と生成したプロセス数, 通信制御プログラムのポート情報) はコミュニケータ内のプロセスにブロードキャストする.

(ii) ランク通知
`MPI_Comm_spawn` 実行元コミュニケータ内での各プロセスのランクを通信制御プログラムに通知し, ランクを指定した通信ができるようにする.

(iii) コミュニケータ内同期
`MPI_Comm_spawn` 実行元コミュニケータ内の全プロセスで `MPI_Comm_spawn` を開始するのを待ち合わせる.

(iv) コミュニケータの作成とリターン情報の設定
プロセスの生成に成功した場合, 新しいコミュニケータ (Slave コミュニケータ) を作成し, 生成したプロセスと通信できるようにする．また, プロセス生成結果を設定する.

(b) Stampi 起動コマンドの処理

(i) MPI プロセス起動処理 (Master 側)
Slave 側計算機に, rsh を用いて Slave 計算機の Stampi 起動コマンドを起動する.

(ii) 通信制御プログラム起動判定
Master 側 Stampi 起動コマンドと salve 側 Stampi 起動コマンドの間で通信制御プログラムを起動する側を調整し, 通信制御プログラムを起動する．起動した通信制御プログラムのポート情報を入手し, 相手の Stampi 起動コマンドに通知する.

(iii) 利用者プログラムへの通信制御プログラムポート情報の通知
Master 側 Stampi 起動コマンドの場合, 通信制御プログラムのポート情報をプロセス起動要求元 (`MPI_Comm_spawn`) に通知する．Slave 側 Stampi 起動コマンドの場合, 通信制御プログラムのポート情報を環境変数に設定して利用者プログラムを起動する.

(c) 通信制御プログラムの処理

(i) ランク管理
プロセス生成元, 生成先, 双方について, 各プロセスとランクとの対応をとり, ランクを指定した通信ができるようにする.

(ii) 通信プログラム間接続
Master 側と Slave 側の双方で通信制御プログラムを起動する場合, Master 側通信制御プログラムから Slave 側通信制御プログラムに接続して通信制御プログラム同士で通信できるようにする.

(d) MPI_Init の処理

(i) ランク通知
Slave 側の場合, 通信制御プログラムに対して, MPI_COMM_WORLD コミュニケータ内でのランクを通知し, ランクを指定した通信ができるようにする.

4.3.2 通信路情報設定機能

(1) 概要

MPI_Info_create 関数と, MPI_Info_set 関数によって, 通信路に関する関する情報を設定し, MPI_Comm_spawn 関数や MPI_Comm_accept 関数, MPI_Comm_connect 関数および MPI_Open_port 関数に引き渡せるようにする.

(2) 起動条件

MPI_Info_create 関数, MPI_Info_set 関数の呼び出しによって起動される.

(3) インタフェース

通信路情報設定機能のインタフェースを表 4.5に示す.

表 4.5: 通信路情報設定機能のインタフェース

#	入出力情報	内容	属性
1	情報ハンドル	一連の情報を管理するためのハンドル	入出力
2	キー	情報の内容を識別するための識別子	入力
3	値	キーに対応する値	入力

(4) 実現方法

通信路情報設定機能は, 以下の処理を行うことにより実現する.

(a) MPI_Info_create の処理

(i) 情報ハンドル作成
新しい情報ハンドルを作成し, 返却する.

(b) MPI_Info_set の処理

(i) 情報の設定
キー・値の組を情報ハンドルに登録する. 既にキーが登録されている場合は, そのキーに対応する値を更新する.

4.3.3 プロセス情報参照機能

(1) 概要

MPI_Comm_get_parent 関数によって, プロセス生成元コミュニケータを参照できる.
また, MPI_Comm_remote_size 関数によって, 生成元 (または生成先) コミュニケータのサイズを参照できる.

(2) 起動条件

MPI_Comm_get_parent 関数, MPI_Comm_remote_size 関数の呼び出しによって起動される.

(3) インタフェース

プロセス情報参照機能のインタフェースを表 4.6に示す.

表 4.6: プロセス情報参照機能のインタフェース

#	入出力情報	内容	属性
1	親コミュニケータ	MPI_Comm_get_parent で得られる自プロセスの生成元コミュニケータ. 生成元コミュニケータが存在しない場合 (親プロセスの場合), MPI_COMM_NULL となる.	出力
2	相手コミュニケータ	MPI_Comm_remote_size の入力である. サイズを調べる対象となるコミュニケータを指定する.	入力
3	サイズ	MPI_Comm_remote_size の出力である. 相手コミュニケータのサイズ	出力

(4) 実現方法

プロセス情報参照機能は, 以下の処理により実現する.

(a) MPI_Init の処理

(i) 親コミュニケータ作成
子プロセスの場合, 親コミュニケータを作成し, コモン領域に格納する.
また, MPI_Comm_remote_size でサイズを参照できるように, 作成したコミュニケータに親コミュニケータのサイズを格納する. 親プロセスの場合には, コモン領域に MPI_COMM_NULL を格納する.

(b) MPI_Comm_get_parent の処理

(i) 親コミュニケータの返却
コモン領域に格納されている親コミュニケータを返却する. 本領域には, 親コミュニケータの場合, MPI_Init において MPI_COMM_NULL が格納されている.

(c) MPI_Comm_remote_size の処理

(i) コミュニケータサイズの返却
指定されたコミュニケータが Stampi で作成したコミュニケータ (MPI_Comm_spawn で作成した子プロセスのコミュニケータと, 子プロセスの MPI_Init で作成する親プロセスのコミュニケータがある) の場合, コミュニケータに格納されているサイズを返却する.

(ii) MPI 互換処理
指定されたコミュニケータが Stampi で作成したコミュニケータでない場合, ベンダ提供の MPI_Comm_remote_size を呼び出す.

4.3.4 Server ポートオープンクローズ機能

(1) 概要

Client/Server 型通信において, Server が接続の待ち合わせに用いるポートをオープン (MPI_Open_port 関数), クローズ (MPI_Close_port 関数) する.

(2) 起動条件

MPI_Open_port 関数, MPI_Close_port 関数が呼び出しによって起動される.

(3) インタフェース

Server ポートオープンクローズ機能のインタフェースを表 4.7に示す.

表 4.7: Server ポートオープンクローズ機能のインタフェース

#	入出力情報	内容	属性
1	情報ハンドル	MPI_Open_port でオープンするポートに関する情報.	入力
2	ポート名	MPI_Open_port でオープンしたポートの名称 (出力). または, MPI_Close_port でクローズするポートの名称 (入力).	入出力

(4) 実現方法

Server ポートオープンクローズ機能は, 以下の処理を行うことにより実現する.

(a) MPI_Open_port の処理

(i) Server ポートオープン
Stampi 起動コマンドに接続し, Server 用のポートオープン要求を行う. Stampi 起動コマンドから, オープンしたポートの名称を取得する.

(b) MPI_Close_port の処理

(i) Server ポートクローズ
Stampi 起動コマンドに接続し, Server 用ポートのクローズ要求を行う.

(c) Stampi 起動コマンドの処理

(i) Server 用ポートオープン要求
MPI_Open_port の要求に対して, ポートの割り当てを行い, ポートの名称を応答する. 割り当てたポートは connect キューと共にポート管理テーブル (Stampi 起動コマンドの内部テーブル) によって管理する.

(ii) Server 用ポートクローズ要求
MPI_Close_port の要求に対して, ポートの削除を行う. ポート管理テーブルに connect キューがある場合, connect の失敗を connect 要求元に応答する.

4.3.5 ポート接続受け付け機能

(1) 概要

Client/Server 型通信において, Client からの接続受け付けを行う.

(2) 起動条件

MPI_Comm_accept 関数の呼び出しによって起動される.

(3) インタフェース

ポート接続受け付け機能のインタフェースを表 4.8に示す.

表 4.8: ポート接続受け付け機能のインタフェース

#	入出力情報	内容	属性
1	ポート名	接続受け付けの対象となる (MPI_Open_port でオープンした) ポートの名称.	入力
2	情報ハンドル	MPI_Comm_accept での接続受け付けに関する情報.	入力
3	Server コミュニケータ	Server 側 MPI プロセスのコミュニケータ.	入力
4	Client コミュニケータ	Client 側 MPI プロセスのコミュニケータ.	出力

(4) 実現方法

ポート接続受け付け機能は, 以下の処理を行うことにより実現する.

(a) MPI_Comm_accept の処理

(i) ポート接続受け入れ要求
MPI_Comm_accept の root プロセスから Stampi 起動コマンドに対して, ポート接続受け入れ要求を行う. ポート接続の結果 (接続の成否と Client 側プロセス数, 通信制御プログラムのポート情報) は Server コミュニケータ内の全プロセスにブロードキャストする.

(ii) ランク通知
MPI_Conn_accept 実行元コミュニケータ内での各プロセスのランクを通信制御プログラムに通知し, ランクを指定した通信ができるようにする.

(iii) コミュニケータ内同期
MPI_Comm_accept 実行元コミュニケータ内の全プロセスでMPI_Comm_accept を開始するのを待ち合わせる.

(iv) Client コミュニケータ作成とリターン情報の設定
Client との接続に成功した場合, 新しいコミュニケータを作成し, Client と通信できるようにする. また, 接続の成否を設定する.

(b) Stampi 起動コマンドの処理

(i) 接続受け入れ処理
Client からの MPI_Comm_connect 要求が既にあれば (Stampi 起動コマンド内部テーブルの

キューとして実現), キューから 1 つ取り出して処理する. なければ, MPI_Comm_connect 要求が発生するのを待つ.

(ii) 通信制御プログラム起動判定
Server 側 Stampi 起動コマンドと Client 側 Stampi 起動コマンドの間で通信制御プログラムを起動する側を調整し, 通信制御プログラムを起動する. 起動した通信制御プログラムのポート情報を入手し, 相手の Stampi 起動コマンドに通知する.

(iii) 利用者プログラムへの通信制御プログラムポート情報の通知
通信制御プログラムのポート情報をプロセス起動要求元 (MPI_Comm_accept) に通知する.

(c) 通信制御プログラムの処理

(i) ランク管理
Server 側, Client 側, 双方について, 各プロセスとランクとの対応をとり, ランクを指定した通信ができるようにする.

(ii) 通信プログラム間接続
Server 側と Client 側の双方で通信制御プログラムを起動する場合, Server 側通信制御プログラムから Client 側通信制御プログラムに接続して通信制御プログラム同士で通信できるようにする.

4.3.6 ポート接続機能

(1) 概要

Client/Server 型通信において, Client から Server への接続を行う.

(2) 起動条件

MPI_Comm_connect 関数の呼び出しによって起動される.

(3) インタフェース

ポート接続受け付け機能のインタフェースを表 4.9に示す.

表 4.9: ポート接続機能のインタフェース

#	入出力情報	内容	属性
1	ポート名	接続先のなる (Server 側の MPI_Open_port でオープンした) ポートの名称.	入力
2	情報ハンドル	MPI_Comm_connect での接続に関する情報.	入力
3	Client コミュニケータ	Client 側 MPI プロセスのコミュニケータ.	入力
4	Server コミュニケータ	Server 側 MPI プロセスのコミュニケータ.	出力

(4) 実現方法

ポート接続機能は, 以下の処理を行うことにより実現する.

(a) MPI_Comm_connect の処理

(i) ポート接続要求
MPI_Comm_connect の root プロセスから Stampi 起動コマンドに対して, ポート接続要求を行う. ポート接続の結果 (接続の成否と Server 側プロセス数, 通信制御プログラムのポート情報) は Client コミュニケータ内の全プロセスにブロードキャストする.

(ii) ランク通知
MPI_Conn_connect 実行元コミュニケータ内での各プロセスのランクを通信制御プログラムに通知し, ランクを指定した通信ができるようにする.

(iii) コミュニケータ内同期
MPI_Comm_connect 実行元コミュニケータ内の全プロセスでMPI_Comm_connect を開始するのを待ち合わせる.

(iv) Server コミュニケータ作成とリターン情報の設定
Server との接続に成功した場合, 新しいコミュニケータを作成し, Server と通信できるようにする. また, 接続の成否を設定する.

(b) Stampi 起動コマンドの処理

(i) 接続処理
Server からの MPI_Comm_accept 要求が待ち状態になっている場合は, MPI_Comm_accept の処理を行う. 待ち状態になっていない場合は, キューに追加する.

(ii) 通信制御プログラム起動判定
Server 側 Stampi 起動コマンドと Client 側 Stampi 起動コマンドの間で通信制御プログラムを起動する側を調整し, 通信制御プログラムを起動する. 起動した通信制御プログラムのポート情報を入手し, 相手の Stampi 起動コマンドに通知する.

(iii) 利用者プログラムへの通信制御プログラムポート情報の通知
通信制御プログラムのポート情報をプロセス起動要求元 (MPI_Comm_connect) に通知する.

(c) 通信制御プログラムの処理
ポート接続受け付け機能と同様の処理を行う. 詳細は, ポート接続受け付け機能参照.

4.3.7 ポート切断機能

(1) 概要

Client/Server 型通信において, 接続したポートの切断を行う.

(2) 起動条件

MPI_Comm_disconnect 関数の呼び出しによって起動される.

(3) インタフェース

ポート切断受け付け機能のインタフェースを表 4.10に示す.

表 4.10: ポート切断機能のインタフェース

#	入出力情報	内容	属性
1	コミュニケータ	MPI_Comm_accept またはMPI_Comm_connect で作成したコミュニケータ. 切断の対象である.	入力

(4) 実現方法

ポート切断機能は, 以下の処理を行うことにより実現する.

(a) MPI_Comm_disconnect の処理

(i) ポート切断処理
MPI_Comm_accept または MPI_Comm_connect で接続を確立したポートを閉じる. コミュニケータも解放する.

(b) 通信制御プログラムの処理

(c) ポート切断処理
Server 側と Client 側双方のポートの切断を受けて, 処理を終了する.

4.4 異機種間通信機能

(1) 概要

異機種間の通信を実現する機能. 実際の通信の他, 通信に付随する各種機能を含む.

(2) 起動条件

MPI 関数において, 異機種間の通信を行う際に呼び出される.

(3) インタフェース

異機種間通信機能のインタフェースを表 4.11に示す.

表 4.11: 異機種間通信のインタフェース

#	入出力情報	内容	属性
1	コミュニケータ	通信対象を表すコミュニケータ.	入出力
2	非同期入出力ハンドル	非同期通信を表すハンドル.	入出力

(4) 実現方法

異機種間通信機能は, 以下の処理を行うことにより実現する.

(a) ブロッキング通信関数
コミュニケータが同一機種内コミュニケータか異機種間コミュニケータかを判定し, 異機種間コミュニケータの場合には, 通信制御プログラム経由で相手計算機と通信を行う. このとき, 通信するデータのサイズに応じて, 以下のプロトコルを使い分ける.

(i) short データ転送プロトコル
比較的転送するデータサイズが小さい場合に用いる (サイズは設定ファイルおよびオプションで変更可能). 1 パケットで全データを送り, 受信側は ACK を返さないという特徴を持つ.

(ii) eager データ転送プロトコル
short データ転送プロトコルより大きなデータサイズの転送で, 且つ, 1 パケットの最大長を下回るサイズのデータ転送に用いる. short プロトコル同様, 1 パケットで全データを送信するが, 受信側から ACK または NACK の応答がある. NACK が返った場合は, large データ転送プロトコルでデータを再送する.

(iii) large データ転送プロトコル
eager データ転送プロトコルで NACK が返った場合と, データサイズが大きく 1 パケットで送信が終らない場合に用いる. 送信要求と受信要求が揃ってから, 双方に ACK を返した後, 別パケットでデータの転送を行うという特徴を持つ.

(b) ノンブロッキング入出力関数
`MPI_Isend`, `MPI_Irecv` 関数では, コミュニケータが同一機種内コミュニケータか異機種間コミュニケータかを判定し, 異機種間コミュニケータの場合には, 通信制御プログラム経由で相手計算機と通信を行うとともに, Stampi のノンブロッキング通信ハンドルを作成し, `MPI_Wait` を行う準備をする.
`MPI_Wait` 関数では, 受け取ったノンブロッキング通信ハンドルが同一計算機内通信のものか異機

種間通信のものかを判定し，異機種間通信のものならば，通信制御プログラムと通信して，ノンブロッキング通信終了処理を行う．
ノンブロッキング通信関数もブロッキング通信関数と同様のプロトコルを使用する．

(c) バリア同期関数
コミュニケータが異機種間コミュニケータである場合，以下の処理を行う．

(i) 対応する同一機種内コミュニケータを使ってバリア同期を取る．

(ii) 同一機種内コミュニケータのランクが 0 のプロセスが，通信制御プログラム経由で，異機種間コミュニケータのランク 0 のプロセスに同期パケットを送信する．

(iii) 同一機種内コミュニケータのランク 0 プロセスで，異機種間コミュニケータのランク 0 プロセスから同期パケットの受信を待つ．

(iv) 再び，同一機種内コミュニケータを使ってバリア同期を取る．

4.5 同一機種内通信機能

(1) 概要

同一機種内の MPI 通信を実現する機能. 実際の通信機能の他, 同一機種内での通信に付随する各種機能 (コミュニケータ操作など) を含む.

同一計算機内の通信は, ベンダ提供 MPI 関数を使用して行う. Stampi では, 同一機種内の通信か他計算機との通信かを判定し, 同一機種内の通信の場合には, ベンダ提供 MPI 関数を使用する通信に必要な情報を揃えて, MPI 関数を呼び出す処理を行う.

(2) 起動条件

MPI 関数において, 同一計算機内のコミュニケータを使用した通信を行う際に呼び出される.

(3) インタフェース

同一機種内通信機能のインタフェースを表 4.12に示す.

表 4.12: 同一機種内通信のインタフェース

#	入出力情報	内容	属性
1	コミュニケータ	通信対象を表すコミュニケータ.	入出力
2	非同期入出力ハンドル	非同期通信を表すハンドル.	入出力

(4) 実現方法

同一機種内通信機能は, 以下の処理を行うことにより実現する.

(a) コミュニケータ作成関数
ベンダ提供の MPI 関数を実行した後, 得られたコミュニケータ (ベンダ提供 MPI 関数の形式) を Stampi の同一機種内コミュニケータに変換する.

(b) コミュニケータ参照関数
Stampi 形式のコミュニケータが, 同一機種内コミュニケータか異機種間コミュニケータかを判定し, 同一機種内コミュニケータの場合はベンダ提供 MPI 関数形式のコミュニケータに変換し, ベンダ提供の MPI 関数を呼び出す. 異機種間コミュニケータの場合は, 異機種間通信機能が対応している関数の場合には異機種間通信機能を起動し, そうでない場合はエラーとする.

(c) 非同期入出力関数
コミュニケータが異機種間コミュニケータか同一機種内コミュニケータかを判定し, それぞれの非同期入出力動作を行う. 同一機種内通信の場合には, ベンダ提供 MPI 関数が作成した非同期ハンドルを Stampi の非同期ハンドルに変換する. また, 異機種間通信の場合には, Stampi の非同期ハンドルを新たに作成する. `MPI_Wait` 関数では, Stampi の非同期ハンドルを参照し, 同一機種内通信の場合には, ベンダ提供 MPI 関数の非同期通信ハンドルに変換してベンダ提供の `MPI_Wait` 関数を呼び出す.

4.6 バッチ連携機能

(1) 概要

MPI_Comm_spawn で生成するプロセスを NQS のバッチキューに投入する機能．バッチの状態を監視し，TSS– バッチ間，バッチ – バッチ間の異機種 MPI 連携を実現する．

(2) 起動条件

MPI_Comm_spawn で NQS バッチキューの指定 (nqsq キー) が指定された場合，起動される．

(3) インタフェース

バッチ連携機能のインタフェースを表 4.13に示す．

表 4.13: バッチ連携機能のインタフェース

#	入出力情報	内容	属性
1	キュー名称	NQS ジョブ投入先キュー名称	入力
2	パーティション名	NQS ジョブで使用する SR2201 のパーテション	入力
3	プロセッサ数	NQS ジョブで使用するプロセッサの数	入力
4	プロセス数	MPI プロセス数	入力

(4) 実現方法

バッチ連携機能は，以下の処理を行うことで実現する．

(a) 通信制御プログラム
MPI プロセス生成 (MPI_Comm_spawn) の際に，NQS バッチキューの指定があるか調べ，指定がある場合には，リモート計算機で Stampi 起動コマンドの代わりに NQS 起動コマンドを起動する．この場合，Stampi 起動コマンドは，NQS 起動コマンドによって登録したバッチジョブから起動される．

(b) NQS 起動コマンド
NQS 起動コマンドは以下のの処理を行う．

(i) Stampi 起動コマンドと接続する通信ポートを作成する．本ポートでは，Stampi 起動コマンドの標準出力と標準エラー出力を受け取る．

(ii) NQS のジョブキューに投入するバッチファイル (shell script) を作成する．ファイルは，Stampi 起動コマンドの実行を含む．

(iii) NQS のジョブ投入コマンド (qsub) を起動し，バッチファイルをジョブキューに投入する．

(iv) (a) で作成したポートに対する接続を待ちながら，定期的に投入したジョブが終了していないかを qstat コマンドを用いて調べる．もし，ジョブが終了してキューから消えていた場合は，NQS 起動コマンドの実行を終了する．

(v) (a) で作成したポートに接続があった場合，ポートの接続が切れるまでポートから入力したデータを標準出力に転送する．ポートの接続が切れたら処理を終了する．

(c) Stampi 起動コマンド
NQS 起動コマンド経由で起動された場合，NQS 起動コマンドのポートに接続して，標準出力と標準エラー出力をこのポートに振り向ける．

(5) 備考

本機能は, NQS を備えている機種でのみ実装する.

4.7 異機種間通信中継機能

(1) 概要

各ノードプロセッサと他の計算機との通信を中継する機能.

(2) 起動条件

Stampi 起動コマンドが起動された際に呼び出される.

(3) インタフェース

異機種間通信中継機能のインタフェースを表 4.14に示す.

表 4.14: 異機種間通信中継機能のインタフェース

#	入出力情報	内容	属性
1	Master 通信ポート	Master 側計算機の通信制御プログラムと接続するための通信ポート情報.	入力
2	通信ポート	通信制御プログラムの通信ポート情報.	出力

(4) 実現方法

異機種間通信中継機能は, 以下の処理を行うことで実現する.

(a) Stampi 起動コマンド
Stampi 起動コマンドが起動したら, 通信制御プログラムを起動した後, 通信制御プログラムから通信ポート情報を受け取り, 利用者プログラムと通信制御プログラムの間で通信できるように準備する.

(b) 通信制御プログラム
通信制御プログラムでは, 以下の処理を行う.

(i) 自通信制御プログラムの通信ポートを開き, Stampi 起動コマンドにポート情報を通知する. また, このポートへの接続を待ち合わせる.

(ii) Slave 側通信制御プログラムの場合には, Master 側通信制御プログラムに接続し, Master - Slave 間の通信路を確立する. 接続した Slave 側のポートをポートテーブルに登録する.

(iii) (i) で作成したポートに接続があった場合, `accept` システムコールで新しいポートを得る. ポートに接続があると, 引続き, 同一計算機内外からの接続かの情報や, 同一計算機内の接続の場合, MPI のランクの情報が伝えられるので, これらの情報をまとめてポートテーブルに登録する. その後, このポートに対する着信を待つ.

(iv) (ii) で接続, または, (iii) で取得したポートに着信があった場合, 同一計算機内からの通信は他計算機へ, また, 他計算機からの通信は同一計算機内の該当するランクのプロセスに転送する.

(c) `MPI_Init` 関数
通信制御プログラムに接続し, 自プロセスのランクを通知する.

4.8 Stampi/Java 機能

Stampi/Java 機能は, 表 4.15のサブ機能から構成される.

表 4.15: Stampi/Java 機能のサブ機能

#	機能	概要
1	Stampi/Java 起動機能	Stampi/Java を起動し, Java 言語で Stampi を利用する環境を設定する.
2	Stampi/Java 関数機能	Stampi 関数を Java 言語で使用可能とする Java 言語インタフェースを実現する.
3	Stampi/Java 終了機能	Stampi/Java の使用を終了する.

4.8.1 Stampi/Java 起動機能

(1) 概要

Stampi/Java を起動し, Java 言語で Stampi を利用する環境を設定する.

(2) 起動条件

`Stampi` クラスの `Init` メソッド (`MPI_Init` 相当) が呼び出されたときに起動する.

(3) インタフェース

Stampi/Java 起動機能のインタフェースを表 4.16に示す.

表 4.16: Stampi/Java 起動機能のインタフェース

#	入出力情報	内容	属性
1	`SCEWindow`	WWW サーバとの通信に使用する.	入力
2	ソケット	Stampi/Java で使用する通信ポート.	出力

(4) 実現方法

Stampi/Java 起動機能は, 以下の処理を行うことで実現する.

(a) WWW サーバで稼働している STA サーバと接続し, Stampi/Java 起動要求を行い, Stampi/Java 通信中継モジュールを起動する.

(b) Stampi/Java の通信で使用するソケットを生成するし, 起動した Stampi/Java 通信中継モジュールに接続する.

(c) Stampi/Java 通信中継モジュールでは, `MPI_Init` を行い, Stampi の初期化をする.

4.8.2 Stampi/Java 関数機能

(1) 概要

Stampi 関数を Java 言語で使用可能とする Java 言語インタフェースを実現する.

(2) 起動条件

Stampi/Java が提供する Stampi 関数 (メソッド) の呼び出しにより起動する.

(3) インタフェース

インタフェースは各 Stampi 関数 (メソッド) に従う.

(4) 実現方法

Stampi/Java 関数機能は, 以下の処理を行うことにより実現する.

(a) Java の Stampi 関数で, 入力データを基に Stampi/Java パケットを作成する.

(b) Stampi/Java パケットを Stampi/Java 通信中継モジュールに送信する.

(c) Stampi/Java 通信中継モジュールは, Stampi/Java パケットに対応する Stampi 関数を実行する.

(d) Stampi 関数の実行結果を基に応答用の Stampi/Java パケットを作成し, 要求元に送信する.

(e) Java の Stampi 関数では, 結果を受信し, 関数呼び元に返却する.

4.8.3 Stampi/Java 終了機能

(1) 概要

Stampi/Java の終了処理を行う.

(2) 起動条件

`Stampi` クラスの `Finalize` メソッドを呼び出した際に起動される.

(3) インタフェース

Stampi/Java 終了機能のインタフェースは特に無い.

(4) 実現方法

Stampi/Java 終了機能は, 以下の処理を行うことにより実現する.

(a) `Stampi` クラスの `Init` メソッドで作成したソケットをクローズする.

(b) Stampi/Java 通信中継モジュールは, ソケットの EOF を検知して処理を終了する.

5 インタフェース

5.1 ハードウェアインタフェース

特別なハードウェアインタフェースはない.

5.2 ソフトウェアインタフェース

MPI 関数が利用者プログラムとインタフェースを持つ.
インタフェースの詳細は, MPI 1.0 (一部 MPI 2.0) に従う.
MPI 関数の一覧を表 5.1に示す.

表 5.1: MPI 関数一覧

#	関数	説明
1	MPI_Comm_spawn	Master/Slave 型通信の Slave プロセスを生成する.
2	MPI_Comm_get_parent	親プロセスのコミュニケータを返却する. 親プロセスがない場合, MPI_COMM_NULL を返却する.
3	MPI_Open_port	Client/Server 型通信の Server ポートをオープンする.
4	MPI_Close_port	MPI_Open_port でオープンした Server ポートをクローズする.
5	MPI_Comm_accept	MPI_Open_port でオープンした Server ポートに Client からの接続を受け付ける.
6	MPI_Comm_connect	Client/Server 型の通信で Client から Server のポートに接続する.
7	MPI_Comm_disconnect	MPI_Comm_accept または MPI_Comm_connect で確立した接続を切断する.
8	MPI_Info_create	MPI_Info 型のデータを生成する.
9	MPI_Info_set	MPI_Info 型のデータに情報を設定する.
10	MPI_Send	ブロッキング通信によって, 他プロセスにデータを送信する.
11	MPI_Recv	ブロッキング通信によって, 他プロセスからデータを受信する.
12	MPI_Get_count	MPI_Recv や MPI_Irecv で受信したデータ数を返却する.
13	MPI_Isend	ノンブロッキング通信によって, 他プロセスにデータを送信する.
14	MPI_Irecv	ノンブロッキング通信によって, 他プロセスからデータを受信する.
15	MPI_Probe	受信メッセージの確認を行う. 受信メッセージが届くまでブロックする.
16	MPI_Iprobe	受信メッセージの確認を行う. 受信メッセージが届いていない場合でもブロックしない.
17	MPI_Comm_remote_size	指定された親プロセスグループ (または子プロセスグループ) のプロセス数を返却する.
18	MPI_Intercomm_merge	グループ間コミュニケータの両端のプロセスグループを合成する.
19	MPI_Barrier	コミュニケータ内の全プロセスでバリア同期を行う.
20	MPI_Bcast	コミュニケータ内の 1 つのプロセスから他の全プロセスに対してデータを送信 (または受信) する.
21	MPI_Gather	コミュニケータ内の全プロセスから特定のプロセスに同一サイズのデータを収集し, ランクの順に格納する.

#	関数	説明
22	MPI_Gatherv	コミュニケータ内の全プロセスから特定のプロセスにデータを収集する．データサイズと格納場所をランク毎に指定可能．
23	MPI_Scatter	特定のプロセスにランク順に格納されている同一サイズのデータをコミュニケータ内の全プロセスに配送する．
24	MPI_Scatterv	特定のプロセスに格納されているデータをコミュニケータ内の全プロセスに配送する．データサイズと格納場所をランク毎に指定可能．
25	MPI_Reduce	コミュニケータ内の全プロセスで大域的なリダクション操作（総和，最大値，論理積など）を行い，結果を特定のプロセスに返却する．
26	MPI_Allreduce	コミュニケータ内の全プロセスで大域的なリダクション操作を行い，結果を全プロセスに返却する．
27	MPI_Init	MPI 初期化関数．
28	MPI_Finalize	MPI 終了関数．
29	MPI_Abort	コミュニケータ内のプロセスの実行を中断する．
30	MPI_Comm_size	コミュニケータ内のプロセス数を返却する．
31	MPI_Comm_rank	コミュニケータ内での自プロセスのランクを返却する．
32	MPI_Comm_dup	コミュニケータの複製を作成する．
33	MPI_Comm_free	コミュニケータを解放する．
34	MPI_Comm_split	コミュニケータ内のプロセスをサブグループに分割することによって，新しいコミュニケータを定義する．
35	MPI_Sendrecv	コミュニケータ内の他プロセスとの間で送受信動作（送信兼受信）を行う．
36	MPI_Type_extent	データ型の大きさ（バイト数）を求める．
37	MPI_Type_vector	等間隔に並んだデータ型のコピーから，新しいデータ型を作成する．データの間隔は，データの数で表す．
38	MPI_Type_hvector	等間隔に並んだデータ型のコピーから，新しいデータ型を作成する．データの間隔は，バイト数で表す．
39	MPI_Type_commit	データ型の記憶操作を行う．通信バッファの記述（通信バッファの内容ではない）を記憶する．
40	MPI_Cart_create	コミュニケータにカルテシアン（直積構造のトポロジ）情報を付加した新しいコミュニケータを返却する．
41	MPI_Cart_shift	カルテシアン構造のトポロジにおいて，座標方向へのデータシフト動作を支援する．具体的には，MPI_Sendrecv 関数に指定するランク（通信相手のコミュニケータ内のランク）を求める．
42	MPI_Cart_coords	コミュニケータ内のプロセスのランクを座標に変換する．
43	MPI_Wtime	ある時刻からの経過時間を返却する．ある時刻は，利用者プログラムの開始から変更されない．返却する時刻は，MPI_Wtime を呼び出したノードにローカルである．

5.3 MPI-2 関数

STAMPI で C 言語から使用可能な MPI-2 関数を表 5.2に示す.
STAMPI では, これらの関数を異機種間通信, および, 同一機種内通信の両方に使用できる.

表 5.2: MPI-2 関数一覧

#	関数	説明
1	MPI_Comm_spawn	Master/Slave 型通信の Slave プロセスを生成する.
2	MPI_Comm_get_parent	親プロセスのコミュニケータを返却する. 親プロセスがない場合, MPI_COMM_NULL を返却する.
3	MPI_Open_port	Client/Server 型通信の Server ポートをオープンする.
4	MPI_Close_port	MPI_Open_port でオープンした Server ポートをクローズする.
5	MPI_Comm_accept	MPI_Open_port でオープンした Server ポートに Client からの接続を受け付ける.
6	MPI_Comm_connect	Client/Server 型の通信で Client から Server のポートに接続する.
7	MPI_Comm_disconnect	MPI_Comm_accept または MPI_Comm_connect で確立した接続を切断する.
8	MPI_Info_create	MPI_Info 型のデータを生成する.
9	MPI_Info_set	MPI_Info 型のデータに情報を設定する.

5.3.1 MPI_Comm_spawn

(1) 概要

Master/Server 型通信において, Slave となる MPI プロセスの生成を行う. 具体的には, Slave STAMPI 起動コマンドを起動して Slave 側利用者プログラムと通信する準備を行う.

(2) インタフェース

```
int MPI_Comm_spawn(char *command, char *argv[], int maxprocs, MPI_Info info,
                   int root, MPI_Comm comm, MPI_Comm *intercomm,
                   int array_of_errcodes[])
```

(3) 説明

MPI_Comm_spawn は集団操作である.

command は, 生成するプログラムの名前を格納する.

argv は, プログラムに渡す引数を格納する文字型配列である. *argv* を省略する場合, MPI_ARGV_NULL を指定する.

本関数は, *maxprocs* の数だけプロセス生成する. プロセスの生成に失敗した場合, エラーを返却する.

info は, このプロセスを実行するホストや手段などの指示情報をシステムに伝える. *info* を省略する場合, MPI_INFO_NULL を指定する.

comm は集団操作のコミュニケータで, Master のコミュニケータとなる. *command*, *argv*, *maxprocs*, *info* は, *comm* 中のランクが *root* と等しいプロセス (ルートプロセス) でだけ意味を持つ.

intercomm は Slave との間のグループ間コミュニケータを返却する.

array_of_errcodes は *maxprocs* の数だけの配列を持ち, 生成されたプロセスの数分 MPI_SUCCESS が返却される. *array_of_errcodes* にMPI_ERRCODES_IGNORE を指定した場合, エラー値は返却しない.

(4) 引数

入力	*command*	文字型	生成するプログラムの名前 (root でのみ意味を持つ)
入力	*argv*	文字型配列	*command* への引数 (root でのみ意味を持つ)
入力	*maxprocs*	整数型	起動プロセスの最大数 (root でのみ意味を持つ)
入力	*info*	ハンドル	プロセス起動に関する情報 (root でのみ意味を持つ)
入力	*root*	整数型	前の引数を認知しているプロセスについてのランク
入力	*comm*	ハンドル	集団操作のコミュニケータ (Master のコミュニケータ)
出力	*intercomm*	ハンドル	Slave との間のグループ間コミュニケータ
出力	*array_of_errcodes*	整数型配列	プロセス毎の返却コード

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.3.2 MPI_Comm_get_parent

(1) 概要

親プロセスのコミュニケータを返却する．親プロセスがない場合 (master 側の場合), MPI_COMM_NULL を返却する．

(2) インタフェース

```
int MPI_Comm_get_parent(MPI_Comm *parent)
```

(3) 説明

プロセスが Slave プロセスの場合 (MPI_Comm_spawn で始められた), Master プロセスとの間のグループ間コミュニケータを返却する．そうでない場合には MPI_COMM_NULL を返す．

(4) 引数

出力	*parent*	ハンドル	Master との間のグループ間コミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する．関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する．

5.3.3 MPI_Open_port

(1) 概要

Client/Server 型通信において, Server 側のポートをオープンする．オープンしたポートに対して, Client からの接続を受け付けることができるようになる．

(2) インタフェース

```
int MPI_Open_port(MPI_Info info, char *port_name)
```

(3) 説明

Server 用にポートをオープンし, ポートの名称を *port_name* に設定する．*info* の指定によって, 利用者が特定のポートを指示することができる．*info* に MPI_INFO_NULL が指定されている場合, オープンするポートはシステムが割り当てる．

(4) 引数

入力	*info*	ハンドル	オープンするポートの情報
出力	*port_name*	文字型	オープンしたポートの名称

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する．関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する．

5.3.4 MPI_Close_port

(1) 概要

MPI_Open_port でオープンしたサーバ用のポートをクローズする．クローズした際にまだ受け付けられていない Client からの接続要求は，全てエラーとなる．

(2) インタフェース

```
int MPI_Close_port(char *port_name)
```

(3) 説明

port_name で指定されたポート (MPI_Open_port で開いた Server 用ポート) をクローズする．

(4) 引数

入力	*port_name*	文字型	クローズするポートの名称

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.3.5 MPI_Comm_accept

(1) 概要

Client/Server 型の通信において, Server 側で Client からの接続の受け付けを行う.

(2) インタフェース

```
int MPI_Comm_accept(char *port_name, MPI_Info info, int root, MPI_Comm comm,
                    MPI_Comm *intercomm)
```

(3) 説明

Client からの MPI_Comm_connect による接続の受け付けを行う.

MPI_Comm_accept は集団操作関数である.

port_name は, 接続に使用するサーバのポート名である. *info* は接続に関する情報を指定する. *root* は root プロセスのランクを指定する. *comm* は集団操作のコミュニケータで, Server のコミュニケータとなる. *intercomm* は Client との間のグループ間コミュニケータを返却する.

(4) 引数

入力	*port_name*	文字型	接続を受け付けるポートの名称 (root プロセスのみ意味を持つ)
入力	*info*	ハンドル	接続に関する情報 (root プロセスのみ意味を持つ)
入力	*root*	整数型	root プロセスのランク
入力	*comm*	ハンドル	集団操作のコミュニケータ (Server のコミュニケータ)
出力	*intercomm*	ハンドル	Client との間のグループ間コミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.3.6 MPI_Comm_connect

(1) 概要

Client/Server 型の通信において, Client 側で Server が用意しているポートに接続を行う.

(2) インタフェース

```
int MPI_Comm_connect(char *port_name, MPI_Info info, int root, MPI_Comm comm,
                     MPI_Comm *intercomm)
```

(3) 説明

Client 側から Server が用意しているポートに接続する.

MPI_Comm_connect は集団操作関数である.

port_name は, 接続に使用するサーバのポート名である. *info* は接続に関する情報を指定する. *root* は root プロセスのランクを指定する. *comm* は集団操作のコミュニケータで, Client のコミュニケータとなる. *intercomm* は Server との間のグループ間コミュニケータを返却する.

(4) 引数

入力	*port_name*	文字型	接続先のポートの名称 (root プロセスのみ意味を持つ)
入力	*info*	ハンドル	接続に関する情報 (root プロセスのみ意味を持つ)
入力	*root*	整数型	root プロセスのランク
入力	*comm*	ハンドル	集団操作のコミュニケータ (Client のコミュニケータ)
出力	*intercomm*	ハンドル	Server との間のグループ間コミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.3.7 MPI_Comm_disconnect

(1) 概要

MPI_Comm_accept または MPI_Comm_connect で確立した接続を切断する.

(2) インタフェース

```
int MPI_Comm_disconnect(MPI_Comm *intercomm)
```

(3) 説明

intercomm に指定されたコミュニケータによる接続を切断する.

intercomm には, MPI_Comm_accept または MPI_Comm_connect の出力のコミュニケータ (*intercomm*) を指定する.

intercomm による接続を切断するとともに, *intercomm* 自身を解放する.

(4) 引数

入出力	*intercomm*	ハンドル	接続を切断するコミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.3.8 MPI_Info_create

(1) 概要

MPI_Info 型で表される設定情報の集合を生成する. MPI_Info_set や, MPI_Comm_spawn , Client/Server 型通信の関数と組み合わせて使用することによって, プロセスの生成や接続を制御できる.

(2) インタフェース

```
int MPI_Info_create(MPI_Info *info)
```

(3) 説明

info に生成した設定情報の集合を格納する.

(4) 引数

出力	*info*	ハンドル	生成された設定情報の集合

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.3.9 MPI_Info_set

(1) 概要

MPI_Info_create で生成した MPI_Info 型の設定情報の集合に情報を登録する.

(2) インタフェース

```
int MPI_Info_set(MPI_Info info, char *key, char *value)
```

(3) 説明

MPI_Info_create で生成した MPI_Info 型の設定情報の集合に, (*key*, *value*) の組みから成る情報を登録する.

key および, *value* は文字型で, 任意の文字列を指定することができる.

(4) 引数

入出力	*info*	ハンドル	設定情報の集合
入力	*key*	文字型	登録する情報のキー
入力	*value*	文字型	登録する *key* に対する情報の値

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.4 異機種間利用可能 MPI-1 関数

STAMPI で C 言語から使用できる MPI-1 関数のうち, 異機種間通信で使用できるものを表 5.3に示す.

STAMPI では, これらの関数を異機種間通信, および, 同一機種内通信の両方に使用できる.

表 5.3: 異機種間利用可能 MPI-1 関数一覧

#	関数	説明
1	MPI_Send	ブロッキング通信によって, 他プロセスにデータを送信する.
2	MPI_Recv	ブロッキング通信によって, 他プロセスからデータを受信する.
3	MPI_Get_count	MPI_Recv や MPI_Irecv で受信したデータ数を返却する.
4	MPI_Isend	ノンブロッキング通信によって, 他プロセスにデータを送信する.
5	MPI_Irecv	ノンブロッキング通信によって, 他プロセスからデータを受信する.
6	MPI_Probe	受信メッセージの確認を行う. 受信メッセージが届くまでブロックする.
7	MPI_Iprobe	受信メッセージの確認を行う. 受信メッセージが届いていない場合でもブロックしない.
8	MPI_Comm_remote_size	指定された親プロセスグループ (または子プロセスグループ) のプロセス数を返却する.
9	MPI_Intercomm_merge	グループ間コミュニケータの両端のプロセスグループを合成する.
10	MPI_Barrier	コミュニケータ内の全プロセスでバリア同期を行う.
11	MPI_Bcast	コミュニケータ内の 1 つのプロセスから他の全プロセスに対してデータを送信 (または受信) する.
12	MPI_Gather	コミュニケータ内の全プロセスから特定のプロセスに同一サイズのデータを収集し, ランクの順に格納する.
13	MPI_Gatherv	コミュニケータ内の全プロセスから特定のプロセスにデータを収集する. データサイズと格納場所をランク毎に指定可能.
14	MPI_Scatter	特定のプロセスにランク順に格納されている同一サイズのデータをコミュニケータ内の全プロセスに配送する.
15	MPI_Scatterv	特定のプロセスに格納されているデータをコミュニケータ内の全プロセスに配送する. データサイズと格納場所をランク毎に指定可能.
16	MPI_Reduce	コミュニケータ内の全プロセスで大域的なリダクション操作 (総和, 最大値, 論理積など) を行い, 結果を特定のプロセスに返却する.
17	MPI_Allreduce	コミュニケータ内の全プロセスで大域的なリダクション操作を行い, 結果を全プロセスに返却する.
18	MPI_Init	MPI 初期化関数.
19	MPI_Finalize	MPI 終了関数.
20	MPI_Abort	コミュニケータ内のプロセスの実行を中断する.

5.4.1 MPI_Send

(1) 概要

コミュニケータ内の他プロセスにデータを送信する．送信はブロッキング通信によって行う．

(2) インタフェース

```
int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)
```

(3) 説明

comm で指定されるコミュニケータのランク *dest* に対して *buf* の内容を送信する．送信するメッセージは，*datatype* で示される型の連続した *count* 個の要素である．

メッセージタグ (*tag*) は受信側でメッセージの識別に使用される．

(4) 引数

入力	*buf*	任意	送信バッファの先頭アドレス
入力	*count*	非負整数型	送信バッファ内の要素数
入力	*datatype*	ハンドル	送信バッファの各要素のデータ型
入力	*dest*	整数型	送信先のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.2 MPI_Recv

(1) 概要

コミュニケータ内の他プロセスからデータを受信する．受信はブロッキング通信によって行う．

(2) インタフェース

```
int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source,
             int tag, MPI_Comm comm, MPI_Status *status)
```

(3) 説明

comm で指定されるコミュニケータのランク *source* から送信された *tag* が一致するメッセージを受信し，*buf* に格納する．受信するメッセージは，*datatype* で示される型の連続した最大 *count* 個の要素である．受信するメッセージの長さは，受信バッファの長さ以下でなければならない．

受信した結果情報は，*status* に格納される．

source に MPI_ANY_SOURCE を指定した場合，任意の送信プロセスからのメッセージを受信する．また，*tag* に MPI_ANY_TAG を指定した場合，任意のメッセージタグと一致する．

(4) 引数

出力	*buf*	任意の型	受信バッファの先頭アドレス
入力	*count*	整数型	受信バッファ内の要素数
入力	*datatype*	ハンドル	受信バッファの各要素のデータ型
入力	*source*	整数型	送信元のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ
出力	*status*	ステータス	ステータスオブジェクト

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.3 MPI_Get_count

(1) 概要

MPI_Recv や MPI_Irecv で受信したデータ数を返却する.

(2) インタフェース

```
int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype, int *count)
```

(3) 説明

MPI_Recv や MPI_Irecv で受信した要素の数を *count* に返す. *datatype* 引数は, *status* 変数をセットした受信関数に渡されたデータ型と一致しなければならない.

(4) 引数

入力	*status*	ステータス	受信関数の返却ステータス
入力	*datatype*	ハンドル	受信バッファの各エントリのデータ型
出力	*count*	整数型	受信されたエントリの数

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.4.4 MPI_Isend

(1) 概要

コミュニケータ内の他プロセスにデータを送信する．送信はノンブロッキング通信によって行う．このため，MPI_Isend 終了後，MPI_Wait を行う必要がある．

(2) インタフェース

```
int MPI_Isend(void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)
```

(3) 説明

comm で指定されるコミュニケータのランク *dest* に対して *buf* の内容を送信する．送信するメッセージは，*datatype* で示される型の連続した *count* 個の要素である．

メッセージタグ (*tag*) は受信側でメッセージの識別に使用される．

ノンブロッキング送信呼び出しは，システムが送信バッファからデータをコピーすることを開始してもよいということを意味する．送信側は，ノンブロッキング送信操作が呼ばれたあとは，送信が完了するまで送信バッファのどの部分にアクセスすることも許されない．

(4) 引数

入力	*buf*	任意の型	送信バッファの先頭アドレス
入力	*count*	非負整数型	送信バッファ内の要素数
入力	*datatype*	ハンドル	送信バッファの各要素のデータ型
入力	*dest*	整数型	送信先のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ
出力	*request*	ハンドル	通信要求

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.5 MPI_Irecv

(1) 概要

コミュニケータ内の他プロセスからデータを受信する．受信はノンブロッキング通信によって行う．このため，MPI_Irecv 終了後，MPI_Wait を行う必要がある．

(2) インタフェース

```
int MPI_Irecv(void* buf, int count, MPI_Datatype datatype, int source,
              int tag, MPI_Comm comm, MPI_Request *request)
```

(3) 説明

comm で指定されるコミュニケータのランク *source* から送信された *tag* が一致するメッセージを受信し，*buf* に格納する．受信するメッセージは，*datatype* で示される型の連続した最大 *count* 個の要素である．受信するメッセージの長さは，受信バッファの長さ以下でなければならない．

source に MPI_ANY_SOURCE を指定した場合，任意の送信プロセスからのメッセージを受信する．また，*tag* に MPI_ANY_TAG を指定した場合，任意のメッセージタグと一致する．

ノンブロッキング受信の呼出しは，システムが受信バッファにデータを書き込むことを開始してもよいということを意味する．受信側は，ノンブロッキング受信操作が呼ばれたあとは，受信が完了するまで受信バッファのどの部分にアクセスすることも許されない．

(4) 引数

入力	*buf*	任意の型	受信バッファの先頭アドレス
入力	*count*	整数型	受信バッファの中の要素数
入力	*datatype*	ハンドル	各受信バッファの要素のデータ型
入力	*source*	整数型	送信元のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ
出力	*request*	ハンドル	通信要求

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.6 MPI_Wait

(1) 概要

ノンブロッキング通信の終了待ち合わせを行う．MPI_Isend や MPI_Irecv を行った後に呼び出す．

(2) インタフェース

```
int MPI_Wait(MPI_Request *request, MPI_Status *status)
```

(3) 説明

MPI_Wait の呼出しは *request* によって識別される操作が完了したときに戻る．この要求に対応する通信要求 (*request*) がノンブロッキング送信やノンブロッキング受信の呼び出しによって作られた場合には，MPI_Wait を呼び出すことによって *request* は解放され，要求ハンドルは MPI_REQUEST_NULL に設定される．

完了した操作についての情報を *status* に返す．

(4) 引数

入出力	*request*	ハンドル	通信要求
出力	*status*	ステータス	ステータスオブジェクト

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.7 MPI_Probe

(1) 概要

受信メッセージの確認を行う．受信メッセージが届くまでブロックする．

(2) インタフェース

```
int MPI_Probe(int source, int tag , MPI_Comm comm, MPI_Status *status)
```

(3) 説明

comm で指定されるコミュニケータのランク *source* から送信された *tag* が一致するメッセージの内容を確認する．該当するメッセージが届いていない場合，受信メッセージが届くまでブロックする．

受信メッセージの確認結果が *status* に格納される．

source に MPI_ANY_SOURCE を指定した場合，任意の送信プロセスからのメッセージを受信する．また，*tag* に MPI_ANY_TAG を指定した場合，任意のメッセージタグと一致する．

(4) 引数

入力	*source*	整数型	送信元のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ
出力	*status*	ステータス	ステータスオブジェクト

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.8 MPI_Iprobe

(1) 概要

受信メッセージの確認を行う．受信メッセージが届いていない場合でもブロックしない．

(2) インタフェース

```
int MPI_Iprobe( int source, int tag , MPI_Comm comm, int *flag, MPI_Status *status)
```

(3) 説明

comm で指定されるコミュニケータのランク *source* から送信された *tag* が一致するメッセージの内容を確認する．受信メッセージが届いていない場合でもブロックしない．該当するメッセージが届いている場合，*flag* は真 (1) となり，受信メッセージの確認結果が *status* に格納される．該当するメッセージが届いていない場合，*flag* は偽 (0) となる．

source に MPI_ANY_SOURCE を指定した場合，任意の送信プロセスからのメッセージを受信する．また，*tag* に MPI_ANY_TAG を指定した場合，任意のメッセージタグと一致する．

(4) 引数

入力	*source*	整数型	送信元のランク
入力	*tag*	整数型	メッセージタグ
入力	*comm*	ハンドル	コミュニケータ
出力	*flag*	論理型	受信確認結果フラグ
出力	*status*	ステータス	ステータスオブジェクト

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.9 MPI_Comm_remote_size

(1) 概要

指定された親プロセスグループ（または子プロセスグループ）のプロセス数を返却する．

(2) インタフェース

```
int MPI_Comm_remote_size(MPI_Comm comm, int *size)
```

(3) 説明

指定された親プロセスグループ（または子プロセスグループ）に属するプロセスの個数を *size* に返す．

(4) 引数

入力	*comm*	ハンドル	グループ間コミュニケータ
出力	*size*	整数型	*comm* 中の他グループに属するプロセスの個数

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.10 MPI_Intercomm_merge

(1) 概要

グループ間コミュニケータの両端のプロセスグループを合成する.

(2) インタフェース

```
int MPI_Intercomm_merge(MPI_Comm intercomm, int high, MPI_Comm *newintracomm)
```

(3) 説明

グループ間コミュニケータ (*intercomm*) の両端のプロセスグループを合成して一つのプロセスグループを生成し, 生成したプロセスグループのグループ内コミュニケータ (*newintracomm*) を返却する.

生成したグループ内コミュニケータ内での各プロセスのランクは, *high* によって決まる. *high* は真 (非0) または偽 (0) を指定する. 全てのプロセスは, *high* にグループ内で同一の値を指定しなければならない. 一方のグループが *high* に真を指定し, 他方のグループが偽を指定する場合, 生成したグループ内コミュニケータのランクは, *high* に偽を指定したグループのプロセスのランクが前 (ランク 0 側) になるように決定される. 両方のプロセスグループが *high* に同じ値を指定した場合は, 生成したプロセスグループ内のランクはシステムが決定する[1].

(4) 引数

入力	*intercomm*	ハンドル	合成対象のグループ間コミュニケータ
入力	*high*	論理型	ランク付け順序の指示
出力	*newintracomm*	ハンドル	合成の結果生成されたグループ内コミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

[1] Stampi では, MPI_Comm_spawn または MPI_Comm_accept を実行したグループ側が前 (ランク 0 側) になるようにランクの割り当てを行う.

5.4.11 MPI_Barrier

(1) 概要

コミュニケータ内の全プロセスでバリア同期を行う.

(2) インタフェース

```
int MPI_Barrier(MPI_Comm comm)
```

(3) 説明

MPI_Barrier は全グループ・メンバが呼び出すまで呼び出し側をブロックする. この手続き呼び出しは, 全グループメンバが MPI_Barrier を呼び出した後でなければ戻らない.

comm にグループ間コミュニケータを指定した場合, グループ間コミュニケータの両端のプロセスグループの全プロセスでバリア同期を行う.

(4) 引数

入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合, MPI_SUCCESS を返却する. 関数の実行に失敗した場合, MPI_SUCCESS 以外の値を返却する.

5.4.12 MPI_Bcast

(1) 概要

コミュニケータ内の1つのプロセスから他の全プロセスに対してデータを送信（または受信）する．

(2) インタフェース

```
int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root,
              MPI_Comm comm)
```

(3) 説明

comm にグループ内コミュニケータを指定した場合，ランクが *root* であるプロセスから，そのプロセスを含むグループ内の全プロセスへメッセージを送信（または受信）する． *comm* および *root* として同じ引数を使って全グループメンバにより呼び出される． 戻った時点で，*root* の通信バッファの内容が全プロセスへコピーされている．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループのルートプロセスのランクを指定する． このとき，ルートプロセスの *buffer* の内容が全リモートプロセスの *buffer* にコピーされる． *root* が
MPI_PROC_NULL のプロセスは通信に参加しない．

(4) 引数

入出力	*buffer*	任意の型	バッファの開始アドレス
入力	*count*	整数型	バッファ内の要素の数
入力	*datatype*	ハンドル	バッファのデータ型
入力	*root*	整数型	ブロードキャスト・ルートのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.13 MPI_Gather

(1) 概要

コミュニケータ内の全プロセスから特定のプロセスに同一サイズのデータを収集し，ランク順に格納する．

(2) インタフェース

```
int MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype sendtype,
               void *recvbuf, int recvcount, MPI_Datatype recvtype,
               int root, MPI_Comm comm)
```

(3) 説明

comm がグループ内コミュニケータの場合，各プロセス（ルートプロセスを含む）は，送信バッファ（*sendbuf*）の内容を *sendcount* のサイズ分だけルートプロセスへ送信する． ルートプロセスはメッセージを *recvcount* のサイズ分だけ受信し，ランクの順に受信バッファ（*recvbuf*）に格納する．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループの root プロセスのランクを指定する． このとき，リモートプロセスの送信バッファ（*sendbuf*）の内容が *sendcount* のサイズ分だけルートプロセスに転送され，リモートプロセスのランク順に受信バッファ（*recvbuf*）に格納される．*root* が MPI_PROC_NULL のプロセスは通信に参加しない．

送信側の *sendcount* の値とルートプロセスの *recvcount* の値は一致しなければならない．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの開始アドレス
入力	*sendcount*	整数型	送信バッファの要素の数
入力	*sendtype*	ハンドル	送信バッファの要素のデータ型
出力	*recvbuf*	任意の型	受信バッファのアドレス（ルートでのみ意味を持つ）
入力	*recvcount*	選択型	受信バッファの要素の数（ルートでのみ意味を持つ）
入力	*recvtype*	ハンドル	受信バッファの要素のデータ型（ルートでのみ意味を持つ）
入力	*root*	整数型	受信プロセスのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.14 MPI_Gatherv

(1) 概要

コミュニケータ内の全プロセスから特定のプロセスにデータを収集する．データサイズと格納場所をランク毎に指定できる．

(2) インタフェース

```
int MPI_Gatherv(void *sendbuf, int sendcount, MPI_Datatype sendtype,
                void *recvbuf, int *recvcounts, int *displs,
                MPI_Datatype recvtype, int root, MPI_Comm comm)
```

(3) 説明

comm がグループ内コミュニケータの場合，各プロセス（ルートプロセスを含む）は，送信バッファ（*sendbuf*）の内容を *sendcount* で指定されたサイズ分だけルートプロセスへ送信する．ルートプロセスは各ランクから送信されたメッセージを *recvcounts* で指定されたサイズ分だけ受信し，受信バッファ（*recvbuf*）の *displs* の位置に格納する．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループの root プロセスのランクを指定する．このとき，リモートプロセスの送信バッファ（*sendbuf*）の内容が *sendcount* のサイズ分だけルートプロセスに転送され，受信バッファ（*recvbuf*）の *displs* の位置に格納される．*root* が MPI_PROC_NULL のプロセスは通信に参加しない．

recvcounts と *displs* はそれぞれ整数ベクトルで，各要素は送信プロセスから受信するデータのサイズと（*recvbuf* 中での）位置を表す．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの開始アドレス
入力	*sendcount*	整数型	送信バッファの要素の数
入力	*sendtype*	ハンドル	送信バッファの要素のデータ型
出力	*recvbuf*	任意の型	受信バッファのアドレス（ルートでのみ意味を持つ）
入力	*recvcounts*	整数型	整数配列．プロセスｉから受信する要素の数をｉ番目の要素に指定（ルートでのみ意味を持つ）
入力	*displs*	整数型	整数配列．プロセスｉから受信するデータを格納する recvbuf からの相対位置をｉ番目の要素に指定（ルートでのみ意味を持つ）
入力	*recvtype*	ハンドル	受信バッファの要素のデータ型（ルートでのみ意味を持つ）
入力	*root*	整数型	データを受信するプロセスのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する．関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.15 MPI_Scatter

(1) 概要

特定のプロセスにランク順に格納されている同一サイズのデータをコミュニケータ内の全プロセスに配送する．

(2) インタフェース

```
int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype sendtype,
                void *recvbuf, int recvcount, MPI_Datatype recvtype,
                int root, MPI_Comm comm)
```

(3) 説明

comm がグループ内コミュニケータの場合，ルートプロセスは各プロセス毎に送信バッファ（*sendbuf*）を *sendcount* のサイズ分だけ送信する． 各プロセス（ルートプロセスを含む）は，受信バッファ（*recvbuf*）にルートプロセスから受信したメッセージを *sendcount* のサイズ分だけ格納する．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループの root プロセスのランクを指定する． このとき，ルートプロセスの送信バッファ（*sendbuf*）の内容がリモートプロセスのランク順に *sendcount* のサイズ分だけリモートプロセスに転送され，リモートプロセスの受信バッファ(*recvbuf*)に格納される． *root* が MPI_PROC_NULL のプロセスは通信に参加しない．

ルートプロセス側の *sendcount* の値と受信プロセスの *recvcount* の値は一致しなければならない．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの開始アドレス（ルートでのみ意味を持つ）
入力	*sendcount*	整数型	送信バッファの要素の数（ルートでのみ意味を持つ）
入力	*sendtype*	ハンドル	送信バッファの要素のデータ型（ルートでのみ意味を持つ）
出力	*recvbuf*	任意の型	受信バッファのアドレス
入力	*recvcount*	選択型	受信バッファの要素の数
入力	*recvtype*	ハンドル	受信バッファの要素のデータ型
入力	*root*	整数型	受信プロセスのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.16 MPI_Scatterv

(1) 概要

特定のプロセスに格納されているデータをコミュニケータ内の全プロセスに配送する． データサイズと格納場所をランク毎に指定できる．

(2) インタフェース

```
int MPI_Scatterv(void *sendbuf, int *sendcount, int *displs,
                 MPI_Datatype sendtype, void *recvbuf, int recvcounts,
                 MPI_Datatype recvtype, int root, MPI_Comm comm)
```

(3) 説明

comm がグループ内コミュニケータの場合，ルートプロセスは送信バッファ（*sendbuf*）の *displs* の位置にあるデータを *sendcounts* 分だけ各プロセスに送信する． 各プロセス（ルートプロセスを含む）は，ルートプロセスから受信したメッセージを受信バッファ（*recvbuf*）に *recvcount* で指定されたサイズ分だけ格納する．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループの root プロセスのランクを指定する． このとき，ルートプロセスの送信バッファ（*sendbuf*）の *displs* で指定された位置の内容が *sendcounts* のサイズ分だけリモートプロセスに転送され，リモートプロセスの受信バッファ（*recvbuf*）に格納される． *root* が MPI_PROC_NULL のプロセスは通信に参加しない．

sendcounts と *displs* はそれぞれ整数ベクトルで，各要素は受信プロセスに送信するデータのサイズと（*sendbuf* 中での）位置を表す．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの開始アドレス（ルートでのみ意味を持つ）
入力	*sendcounts*	整数型	整数配列． プロセス i に送信する要素の数を i 番目の要素に指定（ルートでのみ意味を持つ）
入力	*displs*	整数型	整数配列． プロセス i に送信するデータが格納されている sendbuf からの相対位置を i 番目の要素に指定（ルートでのみ意味を持つ）
入力	*sendtype*	ハンドル	送信バッファの要素のデータ型（ルートでのみ意味を持つ）
出力	*recvbuf*	任意の型	受信バッファのアドレス
入力	*recvcount*	整数型	受信バッファの要素の数
入力	*recvtype*	ハンドル	受信バッファの要素のデータ型
入力	*root*	整数型	データを送信するプロセスのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.17 MPI_Reduce

(1) 概要

コミュニケータ内の全プロセスで大域的なリダクション操作（総和，最大値，論理積など）を行い，結果を特定のプロセスに返却する．

(2) インタフェース

```
int MPI_Reduce(void *sendbuf, void *recvbuf, int count,
               MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
```

(3) 説明

各プロセスの *sendbuf* の内容に対して演算（*op*）を実施した結果をルートプロセスプロセスの *recvbuf* に格納する． *sendbuf* と *recvbuf* は同じ型の要素を同じ個数だけ持つ．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの一方のプロセスグループでは *root* にMPI_ROOT（ルートプロセス）または MPI_PROC_NULL（ルートプロセス以外）を指定し，他方のプロセスグループでは *root* にリモートプロセスグループの root プロセスのランクを指定する． このとき，ルートプロセスの *recvbuf* に各リモートプロセスの *sendbuf* に *op* を実施した結果が格納される． *root* が MPI_PROC_NULL のプロセスは通信に参加しない．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの先頭アドレス
出力	*recvbuf*	任意の型	受信バッファの先頭アドレス（ルートでのみ意味を持つ）
入力	*count*	整数型	送信バッファ中の要素数
入力	*datatype*	ハンドル	送信バッファの要素のデータ型
入力	*op*	ハンドル	演算
入力	*root*	整数型	ルートプロセスのランク
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.18 MPI_Allreduce

(1) 概要

コミュニケータ内の全プロセスで大域的なリダクション操作（総和，最大値，論理積など）を行い，結果を全プロセスに返却する．

(2) インタフェース

```
int MPI_Allreduce(void *sendbuf, void *recvbuf, int count,
                  MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

(3) 説明

各プロセスの *sendbuf* の内容に対して演算（*op*）を実施した結果を各プロセスプロセスの *recvbuf* に格納する． *sendbuf* と *recvbuf* は同じ型の要素を同じ個数だけ持つ．

comm にグループ間コミュニケータを指定した場合，グループ間コミュニケータの双方のプロセスグループで個別に *sendbuf* に *op* を行った結果をリモートプロセスグループの *recvbuf* に格納する．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの先頭アドレス
出力	*recvbuf*	任意の型	受信バッファの先頭アドレス
入力	*count*	整数型	送信バッファ中の要素数
入力	*datatype*	ハンドル	送信バッファの要素のデータ型
入力	*op*	ハンドル	演算
入力	*comm*	ハンドル	コミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.19 MPI_Init

(1) 概要

MPI 初期化関数． STAMPI の初期化を行う．

(2) インタフェース

```
int MPI_Init(int *argc, char ***argv)
```

(3) 説明

MPI プログラムはすべて，MPI_Init の呼び出しを含まなければならない． この関数は他の MPI 関数を呼び出す前に呼ばなければならない．

(4) 引数

入出力	*argc*	整数型	main 関数の第 1 引数 (argc)
入出力	*argv*	文字型配列	main 関数の第 2 引数 (argv)

C 言語の場合は，main 関数の引数によって渡される *argc* と *argv* を受け取る．

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.20 MPI_Finalize

(1) 概要

MPI 終了関数． STAMPI の終了処理を行う．

(2) インタフェース

```
int MPI_Finalize(void)
```

(3) 説明

この関数はすべての MPI 状態を終了させる． この関数がいったん呼び出された後は，どんな MPI 関数も呼び出すことはできない． したがって，利用者はこの関数を呼び出す前にそのプロセスと関連するすべての保留中の通信を完了するようにしなければならない．

(4) 引数

なし．

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.4.21 MPI_Abort

(1) 概要

コミュニケータ内のプロセスの実行を中断する．

(2) インタフェース

int MPI_Abort(MPI_Comm *comm*, int *errorcode*)

(3) 説明

このルーチンは，コミュニケータ comm のグループ内のすべてのタスクを中断しようとする．

(4) 引数

入力	*comm*	ハンドル	中断するタスクのコミュニケータ
入力	*errorcode*	整数型	呼び出し環境へ返すエラー・コード

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5 同一機種内利用 MPI-1 関数

STAMPI で C 言語から使用できる MPI-1 関数のうち，同一機種内通信でのみ使用できるものを表 5.4に示す．

STAMPI では，これらの関数を同一機種内通信に限り使用することができる． これらの関数を異機種間通信で使用した場合はエラーとなる．

表 5.4: 同一機種内利用 MPI-1 関数一覧

#	関数	説明
1	MPI_Comm_size	コミュニケータ内のプロセス数を返却する．
2	MPI_Comm_rank	コミュニケータ内での自プロセスのランクを返却する．
3	MPI_Comm_dup	コミュニケータの複製を作成する．
4	MPI_Comm_free	コミュニケータを解放する．
5	MPI_Comm_split	コミュニケータ内のプロセスをサブグループに分割することによって，新しいコミュニケータを定義する．
6	MPI_Sendrecv	コミュニケータ内の他プロセスとの間で送受信動作（送信兼受信）を行う．
7	MPI_Type_extent	データ型の大きさ（バイト数）を求める．
8	MPI_Type_vector	等間隔に並んだデータ型のコピーから，新しいデータ型を作成する．データの間隔は，データの数で表す．
9	MPI_Type_hvector	等間隔に並んだデータ型のコピーから，新しいデータ型を作成する．データの間隔は，バイト数で表す．
10	MPI_Type_commit	データ型の記憶操作を行う． 通信バッファの記述（通信バッファの内容ではない）を記憶する．
11	MPI_Cart_create	コミュニケータにカルテシアン（直積構造のトポロジ）情報を付加した新しいコミュニケータを返却する．
12	MPI_Cart_shift	カルテシアン構造のトポロジにおいて，座標方向へのデータシフト動作を支援する． 具体的には，MPI_Sendrecv 関数に指定するランク（通信相手のコミュニケータ内のランク）を求める．
13	MPI_Cart_coords	コミュニケータ内のプロセスのランクを座標に変換する．
14	MPI_Wtime	ある時刻からの経過時間を返却する． ある時刻は，利用者プログラムの開始から変更されない． 返却する時刻は，MPI_Wtime を呼び出したノードにローカルである．

5.5.1 MPI_Comm_size

(1) 概要

コミュニケータ内のプロセス数を返却する．

(2) インタフェース

```
int MPI_Comm_size(MPI_Comm comm, int *size)
```

(3) 説明

コミュニケータに関わるのプロセスの個数を *size* に返す．

(4) 引数

入力	*comm*	ハンドル	コミュニケータ
出力	*size*	整数型	*comm* のグループ中のプロセス数

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.2 MPI_Comm_rank

(1) 概要

コミュニケータ内での自プロセスのランクを返却する．

(2) インタフェース

```
int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

(3) 説明

特定のコミュニケータ・グループの中の自プロセスのランクを *rank* に返す．

(4) 引数

入力	*comm*	ハンドル	コミュニケータ
出力	*rank*	整数型	グループ *comm* における呼び出しプロセスのランク

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.3 MPI_Comm_dup

(1) 概要

コミュニケータの複製を作成する．

(2) インタフェース

```
int MPI_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm)
```

(3) 説明

この関数は，既存のコミュニケータ *comm* を複製し，新しいコミュニケータを *newcomm* に返す．

(4) 引数

入力	*comm*	ハンドル	コミュニケータ
出力	*newcomm*	ハンドル	*comm* の複製

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.4 MPI_Comm_free

(1) 概要

コミュニケータを解放する．

(2) インタフェース

```
int MPI_Comm_free(MPI_Comm *comm)
```

(3) 説明

この関数は，コミュニケータ *comm* を解放する． *comm* には MPI_COMM_NULL が設定される．

(4) 引数

入出力	*comm*	ハンドル	解放されるコミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.5 MPI_Comm_split

(1) 概要

コミュニケータ内のプロセスをサブグループに分割することによって，新しいコミュニケータを定義する．

(2) インタフェース

```
int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)
```

(3) 説明

この関数は，*comm* に付随するグループを *color* の値に対応してそれぞれ重ならないサブグループに分割する． 各サブグループは同一 *color* のプロセスを全て含む． 各サブグループ内で，プロセスは *key* の値の順序でランクが付けられ，同一の *key* のプロセスは旧グループのランク順に従ってランクが決められる． 各サブグループに対応した新しいコミュニケータが生成され，*newcomm* に返される．

(4) 引数

入力	*comm*	ハンドル	コミュニケータ
入力	*color*	非負整数型	サブセット割当の制御
入力	*key*	整数型	ランク割当の制御
出力	*newcomm*	ハンドル	新しいコミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.6 MPI_Sendrecv

(1) 概要

コミュニケータ内の他プロセスとの間で送受信動作（送信兼受信）を行う．

(2) インタフェース

```
int MPI_Sendrecv(void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 int dest, int sendtag, void *recvbuf, int recvcount,
                 MPI_Datatype recvtype, int source, int recvtag,
                 MPI_Comm comm, MPI_Status *status)
```

(3) 説明

送受信操作は，特定の送信先に対するメッセージの送信動作と，他のプロセスから別のメッセージを受信する動作を１つに組み合わせたものである． この関数では，送信と受信の両方で同じコミュニケータが使われているが，タグは異なったものとすることが望ましい． ただし，送信バッファと受信バッファは独立したものであり，異なった長さとデータ型を持つことができる．

(4) 引数

入力	*sendbuf*	任意の型	送信バッファの先頭アドレス
入力	*sendcount*	整数型	送信バッファ内の要素数
入力	*sendtype*	ハンドル	送信バッファ内の要素の型
入力	*dest*	整数型	送信先のランク
入力	*sendtag*	整数型	送信タグ
出力	*recvbuf*	任意の型	受信バッファの先頭アドレス
入力	*recvcount*	整数型	受信バッファ内の要素数
入力	*recvtype*	ハンドル	受信バッファ内の要素の型
入力	*source*	整数型	送信元のランク
入力	*recvtag*	整数型	受信タグ
入力	*comm*	ハンドル	コミュニケータ
出力	*status*	ステータス	ステータスオブジェクト

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.7 MPI_Type_extent

(1) 概要

データ型の大きさ（バイト数）を求める．

(2) インタフェース

```
int MPI_Type_extent(MPI_Datatype datatype, int *extent)
```

(3) 説明

この関数は，*datatype* のデータ型で作成されるメッセージデータの合計サイズをバイト数で *extent* に返す．

(4) 引数

入力	*datatype*	ハンドル	データ型
出力	*extent*	整数型	データ型の範囲

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.8 MPI_Type_vector

(1) 概要

等間隔に並んだデータ型のコピーから，新しいデータ型を作成する． データの間隔は，データの数で表す．

(2) インタフェース

```
int MPI_Type_vector(int count, int blocklength, int stride,
                    MPI_Datatype oldtype, MPI_Datatype *newtype)
```

(3) 説明

等間隔に並んだデータ型のコピーから新しいデータ型を作成する． 個々のブロックは同じ数の元のデータ型の連なりからなる． データの間隔は元のデータ型の範囲の倍数となる．（データの数）

(4) 引数

入力	*count*	非負整数型	繰り返し回数
入力	*blocklength*	非負整数型	個々のブロックの要素数
入力	*stride*	整数型	個々のブロックの先頭の間隔の要素数
入力	*oldtype*	ハンドル	旧データ型
出力	*newtype*	ハンドル	新データ型

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.9 MPI_Type_hvector

(1) 概要

等間隔に並んだデータ型のコピーから，新しいデータ型を作成する． データの間隔は，バイト数で表す．

(2) インタフェース

```
int MPI_Type_hvector(int count, int blocklength, MPI_Aint stride,
                     MPI_Datatype oldtype, MPI_Datatype *newtype)
```

(3) 説明

等間隔に並んだデータ型のコピーから新しいデータ型を作成する． 個々のブロックは同じ数の元のデータ型の連なりからなる． データの間隔は元のデータ型の範囲の倍数となる． （バイト数）

(4) 引数

入力	*count*	非負整数型	繰り返し回数
入力	*blocklength*	非負整数型	個々のブロックの要素数
入力	*stride*	整数型	個々のブロックの先頭の間隔のバイト数
入力	*oldtype*	ハンドル	旧データ型
出力	*newtype*	ハンドル	新データ型

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.10 MPI_Type_commit

(1) 概要

データ型の記憶操作を行う． 通信バッファの記述（通信バッファの内容ではない）を記憶する．

(2) インタフェース

```
int MPI_Type_commit(MPI_Datatype *datatype)
```

(3) 説明

記憶操作は，通信バッファの内容ではなく，通信バッファの記述を記憶する．

(4) 引数

入出力	*datatype*	ハンドル	記憶するデータ型

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.11 MPI_Cart_create

(1) 概要

コミュニケータにカルテシアン（直積構造のトポロジ）情報を付加した新しいコミュニケータを返却する．

(2) インタフェース

```
int MPI_Cart_create(MPI_Comm comm_old, int ndims, int *dims, int *periods,
                    int reorder, MPI_Comm *comm_cart)
```

(3) 説明

この関数は，カルテシアン・トポロジー情報を付加した新しいコミュニケータのハンドルを *comm_cat* に返す． *reorder* が `false` であれば，新規グループの中の各プロセスのランクは旧グループのそれと同じである． そうでない場合は，プロセスを並べ替える場合がある．

(4) 引数

入力	*comm_old*	ハンドル	入力コミュニケータ
入力	*ndims*	整数型	カルテシアン格子の次元数
入力	*dims*	整数型配列	各次元に対しそのプロセス数を指定したサイズ *ndims* の整数型配列
入力	*periods*	整数型配列	各次元に対しその格子が周期的（`true`）か否か（`false`）を指定したサイズ *ndims* の整数型配列
入力	*reorder*	論理型	ランク付けを変更してよい（`true`）か否か（`false`）
出力	*comm_cat*	ハンドル	新しく生成させたカルテシアン・トポロジーを持つコミュニケータ

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.12 MPI_Cart_shift

(1) 概要

カルテシアン構造のトポロジにおいて，座標方向へのデータシフト動作を支援する． 具体的には MPI_Sendrecv 関数に指定するランク（通信相手のコミュニケータ内のランク）を求める．

(2) インタフェース

```
int MPI_Cart_shift(MPI_Comm comm, int direction, int  disp, int *rank_source,
                   int *rank_dest)
```

(3) 説明

指定された座標方向におけるカルテシアン・グループが周期的か否かに応じて，循環シフトかまたはエンドオフ・シフトの識別子を提供する． エンドオフ・シフトの場合，値 MPI_PROC_NULL を *rank_source* または *rank_disp* に返すことがある． これは，シフトの送信元または送信先が範囲外であることを示している．

(4) 引数

入力	*comm*	ハンドル	カルテシアン構造を持つコミュニケータ
入力	*direction*	整数型	シフトを行う座標の次元
入力	*disp*	整数型	変位（> 0: 上方へのシフト，< 0: 下方へのシフト）
出力	*rank_source*	整数型	送信元プロセスのランク
出力	*rank_disp*	整数型	送信先プロセスのランク

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.13 MPI_Cart_coords

(1) 概要

コミュニケータ内のプロセスのランクを座標に変換する．

(2) インタフェース

```
int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int *coords)
```

(3) 説明

この関数は，逆のマッピングであるランクから座標への変換を行う．

(4) 引数

入力	*comm*	ハンドル	カルテシアン構造を持つコミュニケータ
入力	*rank*	整数型	グループ *comm* の中でのプロセスのランク
入力	*maxdims*	整数型	呼び出し側プログラムのベクトル *coords* のサイズ
出力	*coords*	整数型配列	指定したプロセスのカルテシアン座標を格納する（サイズ *maxdims* の）整数型配列

(5) 返却値

関数の実行に成功した場合，MPI_SUCCESS を返却する． 関数の実行に失敗した場合，MPI_SUCCESS 以外の値を返却する．

5.5.14 MPI_Wtime

(1) 概要

ある時刻からの経過時刻を返却する． ある時刻は，利用者プログラムの開始から変更されない． 返却する時刻は，MPI_Wtime を呼び出したノードにローカルである．

(2) インタフェース

```
double MPI_Wtime(void)
```

(3) 説明

ある時刻からの経過時刻を表す秒数を浮動小数点で返す． 秒数の単位を変更したい場合には，ユーザーが自分で変更を行わなくてはならない． また，返される時刻は，この関数を呼び出したノードにローカルである．

(4) 引数

なし．

(5) 返却値

ある時刻からの経過時刻を返却する．

5.6 Stampi/Java インタフェース

5.6.1 Stampi/Java クラス

Stampi/Java では、表 5.5のクラスを提供する。

表 5.5: Stampi/Java 提供クラス

#	クラス	説明
1	Stampi	Stampi/Java 全体を表す
2	MPI_Comm	コミュニケータ
3	MPI_Intracomm	内部コミュニケータ
4	MPI_Intercomm	相互コミュニケータ
5	MPI_Datatype	データ型
6	MPI_Info	各種情報
7	MPI_Staus	通信情報

5.6.2 Stampi クラス

Stampi クラス定義を以下に示す (public 部分のみ)。

```
public class Stampi extends Object
{
    public MPI_Comm         COMM_NULL;   // MPI_COMM_NULL
    public MPI_Intracomm    COMM_WORLD;  // MPI_COMM_WORLD
    public MPI_Intracomm    COMM_SELF;   // MPI_COMM_SELF
    public MPI_Datatype     CHAR;        // MPI_CHAR
    public MPI_Datatype     SHORT;       // MPI_SHORT
    public MPI_Datatype     INT;         // MPI_INT
    public MPI_Datatype     FLOAT;       // MPI_FLOAT
    public MPI_Datatype     DOUBLE;      // MPI_DOUBLE
    public MPI_Datatype     BYTE;        // MPI_BYTE
    public MPI_Info         INFO_NULL;   // MPI_INFO_NULL
    public string[]         ARGV_NULL;   // MPI_ARGV_NULL
    public int              ANY_SOURCE;  // MPI_ANY_SOURCE
    public int              ANY_TAG;     // MPI_ANY_TAG
    public Stampi(SceWindow)             // Stampi クラスコンストラクタ
    public void Init()                   // MPI_Init
            throws IOException,
                   UnknownHostException,
                   SceException,
                   OutOfBufferException
    public void Finalize()               // MPI_Finalize
            throws IOException
}
```

5.6.3 MPI_Comm クラス

MPI_Comm クラスのクラス定義を以下に示す (publuc 部分のみ)。

```
public class MPI_Comm
{
    public int Get_size()  // MPI_Comm_size, 返却値: size
    public int Get_rank()  // MPI_Comm_rank, 返却値: rank
}
```

5.6.4 MPI_Intracomm クラス

MPI_Intracomm クラスのクラス定義を以下に示す (public 部分のみ)。

```
public class MPI_Intracomm extends MPI_Comm
{
    public MPI_Intercomm Spawn(  // MPI_Comm_spawn, 返却値: MPI_Intercomm
         String   command,       //     コマンド
         String[] args,          //     コマンド引数
         int      maxprocs,      //     起動する MPI プロセス数
         MPI_Info info,          //     MPI_Info
         int      root)          //     root プロセス
            throws IOException
}
```

5.6.5 MPI_Intercomm クラス

MPI_Intercomm クラスのクラス定義を以下に示す (public 部分のみ)。

```
public class MPI_Intercomm extends MPI_Comm
{
    public void Send(            // MPI_Send (scaler char 用)
        char buf,                //     送信バッファ
        int dest,                //     受信 rank
        int tag)                 //     送信 tag
            throws IOException
    public void Send(            // MPI_Send (scaler short 用)
        short buf,               //     送信バッファ
        int dest,                //     受信 rank
        int tag)                 //     送信 tag
            throws IOException
    public void Send(            // MPI_Send (scaler int 用)
        int buf,                 //     送信バッファ
        int dest,                //     受信 rank
        int tag)                 //     送信 tag
            throws IOException
    public void Send(            // MPI_Send (scaler float 用)
```

```
    float buf,              //      送信バッファ
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (scaler double 用)
    double buf,             //      送信バッファ
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (scaler byte 用)
    byte buf,               //      送信バッファ
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (配列 char 用)
    char[] buf,             //      送信バッファ
    int count,              //      送信要素数
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (配列 short 用)
    short[] buf,            //      送信バッファ
    int count,              //      送信要素数
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (配列 int 用)
    int[] buf,              //      送信バッファ
    int count,              //      送信要素数
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (配列 float 用)
    float[] buf,            //      送信バッファ
    int count,              //      送信要素数
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
public void Send(           // MPI_Send (配列 double 用)
    double[] buf,           //      送信バッファ
    int count,              //      送信要素数
    int dest,               //      受信 rank
    int tag)                //      送信 tag
        throws IOException
```

```
public void Send(            // MPI_Send (配列 byte 用)
    byte[] buf,              //     送信バッファ
    int count,               //     送信要素数
    int dest,                //     受信 rank
    int tag)                 //     送信 tag
        throws IOException
public void Send(            // MPI_Send (string 用)
    string buf,              //     送信バッファ
    int dest,                //     受信 rank
    int tag)                 //     送信 tag
        throws IOException
public char Recv_char(       // MPI_recv (scaler char 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
    MPI_Status status)       //     受信状態
        throws IOException,
               OutOfBufferException
public short Recv_short(     // MPI_recv (scaler short 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
    MPI_Status status)       //     受信状態
        throws IOException,
               OutOfBufferException
public int Recv_int(         // MPI_recv (scaler int 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
    MPI_Status status)       //     受信状態
        throws IOException,
               OutOfBufferException
public float Recv_float(     // MPI_recv (scaler float 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
    MPI_Status status)       //     受信状態
        throws IOException,
               OutOfBufferException
public double Recv_double(   // MPI_recv (scaler double 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
    MPI_Status status)       //     受信状態
        throws IOException,
               OutOfBufferException
public byte Recv_byte(       // MPI_recv (scaler byte 用)
    int source,              //     送信 rank
    int tag,                 //     送信 tag
```

```
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 char 用)
    char[] buf,               //      受信バッファ
    int count,                //      受信要素数
    int source,               //      送信 rank
    int tag,                  //      送信 tag
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 short 用)
    short[] buf,              //      受信バッファ
    int count,                //      受信要素数
    int source,               //      送信 rank
    int tag,                  //      送信 tag
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 int 用)
    int[] buf,                //      受信バッファ
    int count,                //      受信要素数
    int source,               //      送信 rank
    int tag,                  //      送信 tag
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 float 用)
    float[] buf,              //      受信バッファ
    int count,                //      受信要素数
    int source,               //      送信 rank
    int tag,                  //      送信 tag
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 double 用)
    double[] buf,             //      受信バッファ
    int count,                //      受信要素数
    int source,               //      送信 rank
    int tag,                  //      送信 tag
    MPI_Status status)        //      受信状態
        throws IOException,
               OutOfBufferException
public void Recv(             // MPI_recv (配列 byte 用)
```

```
        byte[] buf,             //      受信バッファ
        int count,              //      受信要素数
        int source,             //      送信 rank
        int tag,                //      送信 tag
        MPI_Status status)      //      受信状態
            throws IOException,
                   OutOfBufferException
    public string Recv_string(  // MPI_recv (string 用)
        int source,             //      送信 rank
        int tag,                //      送信 tag
        MPI_Status status)      //      受信状態
            throws IOException,
                   OutOfBufferException
    public int Get_remote_size() // MPI_Remote_size, 返却値: remote size
}
```

5.6.6 MPI_Datatype クラス

MPI_Datatype クラスのクラス定義を以下に示す (public 部分のみ)。

```
public class MPI_Datatype
{
}
```

5.6.7 MPI_Info クラス

MPI_Info クラスのクラス定義を以下に示す。

```
public class MPI_Info
{
    public MPI_Info(Stampi)     // MPI_Info クラスコンストラクタ
            throws IOException
    public void Set(            // MPI_Info_set
        string key,             //      key
        string val)             //      value
            throws IOException
}
```

5.6.8 MPI_Status クラス

MPI_Status クラスのクラス定義を以下に示す (public 部分のみ)。

```
public class MPI_Status
{
    public MPI_Status()         // MPI_Status クラスコンストラクタ
    public int Get_count(       // MPI_Get_count, 返却値: 受信要素数
        MPI_Datatype)           //      受信データ型
```

```
    public int Get_source()       // MPI_SOURCE
    public int Get_tag()          // MPI_TAG
    public int Get_error()        // MPI_ERROR
}
```

5.7 Stampi プロトコル仕様

Stampi では異機種間の通信に表 5.6に示すパケットから構成されるプロトコルを用いる.

表 5.6: パケット一覧

#	パケット	説明
1	接続要求	接続時, ランク情報などを通信制御プログラムに通知する
2	送信開始要求	送信の開始を要求する
3	受信開始要求	受信の開始を要求する
4	データ転送	送信要求, 受信要求に従い, データの転送を行う
5	バリア同期	異機種間でバリア同期をとる
6	SPAWN 開始	新しいプロセスの生成を開始する
7	SPAWN info	プロセス生成に必要な情報を通信制御プログラムに通知する
8	SPAWN 実行開始	新しいプロセスの実行開始を要求する
9	ACK	汎用応答 (成功)
10	NACK	汎用応答 (失敗)

5.7.1 接続要求パケット

(1) 概要

接続要求パケットは, 通信制御プログラムに接続したポートの種類を通信制御プログラムに対して通知する.

ポートの種類としては, MPI プロセス (同一機種内から接続) と, 他の通信制御プログラム (他機種からの接続) の 2 種類がある.

(2) 処理条件

接続要求パケットは, 以下の条件で使用される.

(a) 利用者プログラムで, MPI_Init を実行した際に使用される.

(b) 利用者プログラムで, MPI_Comm_spawn を行い, 子プロセスグループ用の通信制御プログラムが起動した際に使用される.

(3) 内容

接続要求パケットの内容を表 5.7に示す.

表 5.7: 接続要求パケットの内容

#	項目	内容
1	種別	パケットの種類. 接続要求パケットを表す.
2	送信元ランク	接続したプロセスのランク. 他通信制御プログラムからの接続の場合は無効.
3	プロセスグループサイズ	接続したプロセスが属すプロセスグループのサイズ. 他通信制御プログラムからの接続の場合は 0 となる.

5.7.2 送信開始要求

(1) 概要

送信要求パケットは, 通信制御プログラムに対して, 送信開始を要求する.

通信制御プログラムは, 送信開始要求パケットと, 受信開始要求パケットの対応をとって, 2 プロセス間での通信を許可する.

(2) 処理条件

送信開始要求は, 以下の条件で使用される.

(a) MPI_Send MPI_Isend に異機種間コミュニケータを指定した場合.

(3) 内容

送信開始要求パケットの内容を表 5.8に示す.

表 5.8: 送信要求パケットの内容

#	項目	内容
1	種別	パケットの種類. 送信要求パケットを表す.
2	送信元ランク	データ送信元プロセスのランク.
3	受信先ランク	データ受信先プロセスのランク.
4	タグ	送信タグ.
5	エントリ数	送信するデータの数.
6	エントリサイズ	送信するデータの１エントリのバイト数.
7	データサイズ	エントリサイズ × エントリ数.

5.7.3 受信開始要求

(1) 概要

受信要求パケットは, 通信制御プログラムに対して, 受信開始を要求する.

通信制御プログラムは, 送信開始要求パケットと, 受信開始要求パケットの対応をとって, 2 プロセス間での通信を許可する.

(2) 処理条件

受信開始要求は, 以下の条件で使用される.

(a) MPI_Recv, MPI_Irecv に異機種間コミュニケータを指定した場合.

(3) 内容

受信開始要求パケットの内容を表 5.9に示す.

表 5.9: 受信要求パケットの内容

#	項目	内容
1	種別	パケットの種類. 受信要求パケットを表す.
2	送信元ランク	データ送信元プロセスのランク.
3	受信先ランク	データ受信先プロセスのランク.
4	タグ	送信タグ.
5	エントリ数	受信するデータの数.
6	エントリサイズ	受信するデータの 1 エントリのバイト数.
7	データサイズ	エントリサイズ × エントリ数.

5.7.4 データ転送パケット

(1) 概要

データ転送パケットは, 通信制御プログラム経由で相手プロセスにデータを転送する.

(2) 処理条件

送信開始要求は, 以下の条件で使用される.

(a) 送信要求パケットを発行し, 受信要求パケットを受け取った際に使用される.

(3) 内容

データ転送パケットの内容を表 5.10に示す.

表 5.10: データ転送パケットの内容

#	項目	内容
1	種別	パケットの種類. データ転送パケットを表す.
2	送信元ランク	データ送信元プロセスのランク.
3	受信先ランク	データ受信先プロセスのランク.
4	シーケンス	何番目のパケットかを表すシーケンス.
5	トータルパケット数	1回のデータ転送で使用するパケット数.
6	データサイズ	このパケットに含まれるデータのサイズ.
7	データ	データの実体.

5.7.5 バリア同期パケット

(1) 概要

バリア同期パケットは, 異機種間でのバリア同期を要求する.

異機種間のバリア同期は, 2つのプロセスグループを代表する各々1つのプロセス間での単純同期 (本パケットを使用して実現) と, 2つのプロセスグループの各々のバリア同期を組み合わせて実現する.

(2) 処理条件

バリア同期要求は, 以下の条件で使用される.

(a) `MPI_Barrier` に異機種間コミュニケータを指定した場合.

(3) 内容

バリア同期パケットの内容を表 5.11に示す.

表 5.11: バリア同期パケットの内容

#	項目	内容
1	種別	パケットの種類. バリア同期パケットを表す.
2	送信元ランク	パケット送信元プロセスのランク.
3	受信先ランク	パケット受信先プロセスのランク.

5.7.6 SPAWN開始パケット

(1) 概要

SPAWN開始パケットは, 新しいMPIプロセス生成の開始を要求する.

MPIプロセス生成は, SPAWN開始パケット, SPAWN infoパケット, SPAWN実行開始パケットの3種類のパケットから構成される. このうち, SPAWN infoパケットとSPAWN実行開始パケットはrootプロセスでのみ発行する. SPAWN開始要求パケットは`MPI_Comm_spawn`を行う各プロセスから発行される.

(2) 処理条件

SPAWN開始パケットは, 以下の条件で使用される.

(a) `MPI_Comm_spawn`が呼び出された場合.

(3) 内容

SPAWN開始パケットの内容を表5.12に示す.

表 5.12: SPAWN開始パケットの内容

#	項目	内容
1	種別	パケットの種類. SPAWN開始パケットを表す.
2	送信元ランク1	パケット送信元プロセスの`MPI_Comm_world`内のランク.
3	送信元ランク2	パケット送信元プロセスの`MPI_Comm_spawn`実行コミュニケータ内でのランク.
4	コミュニケータサイズ	`MPI_Comm_spawn`を実行したコミュニケータのサイズ.
5	ルートプロセス	ルートプロセスか否かの情報.
6	生成プロセス数	生成するMPIプロセスのプロセス数. ルートプロセスのみ有効.

5.7.7 SPAWN info パケット

(1) 概要

SPAWN info パケットは, プロセスの生成に必要な情報を通信制御プログラムに通知する.
通知する情報は, MPI_Info に格納されている情報である.
SPAWN info パケットは, MPI_Comm_spawn のルートプロセスから, 情報の数分だけ発行される.

(2) 処理条件

SPAWN info パケットは, 以下の条件で使用される.

(a) MPI_Comm_spawn のルートプロセスにおいて, SPAWN 開始パケットを発行した後に使用される.

(3) 内容

SPAWN info パケットの内容を表 5.13に示す.

表 5.13: SPAWN info パケットの内容

#	項目	内容
1	種別	パケットの種類. SPAWN info パケットを表す.
2	情報	MPI_Info に格納されている情報.

5.7.8 ACK パケット

(1) 概要

ACK パケットは, 汎用の処理成功通知である.
ACK パケットは, 他のパケットに対する処理の応答として使用される.

(2) 処理条件

ACK パケットは, 以下の条件で使用される.

(a) 何らかの処理を行い, 処理成功を通知するとき. 続いて発行される.

(3) 内容

ACK パケットの内容を表 5.14に示す.

表 5.14: ACK パケットの内容

#	項目	内容
1	種別	パケットの種類. ACK パケットを表す.
2	送信元ランク	ACK パケット送信元プロセスを表す.
3	受信先ランク	ACK パケット受信先プロセスを表す.

5.7.9 NACK パケット

(1) 概要

NACK パケットは, 汎用の処理失敗通知である.
NACK パケットは, 他のパケットに対する処理の応答として使用される.

(2) 処理条件

NACK パケットは, 以下の条件で使用される.

(a) 何らかの処理を行い, 処理失敗を通知するとき. 続いて発行される.

(3) 内容

NACK パケットの内容を表 5.15に示す.

表 5.15: NACK パケットの内容

#	項目	内容
1	種別	パケットの種類. NACK パケットを表す.
2	送信元ランク	NACK パケット送信元プロセスを表す.
3	受信先ランク	NACK パケット受信先プロセスを表す.

5.8 Stampi/Java プロトコル仕様

Stampi/Java では, Java アプレットと Stampi/Java 通信中継モジュールの間の通信に表 5.16に示すパケットから構成されるプロトコルを用いる. 尚, Stampi/Java 通信中継モジュールとバックエンド計算機との間では Stampi プロトコルを使用した通信を行う.

表 5.16: Stampi/Java プロトコルのパケット一覧

#	パケット	説明
1	Send	MPI_Send によるデータ送信要求
2	Recv	MPI_Recv によるデータ受信要求
3	Info_create	MPI_Info_create による Info オブジェクト作成要求
4	Info_set	MPI_Info_set による Info オブジェクト設定要求
5	Spawn	MPI_Comm_spawn による子プロセス生成要求
6	ACK	汎用応答 (成功)
7	NACK	汎用応答 (失敗)

5.8.1 Send プロトコル

(1) 概要

Send プロトコルは MPI_Send に対応したプロトコルであり, Send パケットとして表現される. Send パケットは, 応答として ACK パケット (成功) または NACK パケット (失敗) が返される.

(2) 処理条件

Send プロトコルは MPI_Intercomm クラスの Send メソッドを呼び出した際に使用される.

(3) 内容

Send パケットの内容を表 5.17に示す. また, Send パケットの応答を表 5.18に示す.

表 5.17: Send パケットの内容

#	項目	内容
1	type	パケットの種類. Send パケットを表す.
2	size	送信するデータのバイト数.
3	handle	コミュニケータ (ハンドル).
4	remote	受信先のランク.
5	tag	通信タグ.
6	transsize	Send では size と同じ値となる.
7	data	送信するデータの実体

表 5.18: Send パケットの応答

#	項目	内容
1	type	パケットの種類. ACK (成功) または NACK(失敗).
2	size	常に 0.
3	handle	常に 0.
4	remote	常に 0.
5	tag	常に 0.
6	transsize	常に 0.

5.8.2 Recv プロトコル

(1) 概要

Recv プロトコルは MPI_Recv に対応したプロトコルであり, Recv パケットとして表現される. Recv パケットは, 応答として ACK パケット (成功) または NACK パケット (失敗) が返される.

(2) 処理条件

Recv プロトコルは MPI_Intercomm クラスの Recv メソッドを呼び出した際に使用される.

(3) 内容

Recv パケットの内容を表 5.19に示す. また, Recv パケットの応答を表 5.20に示す.

表 5.19: Recv パケットの内容

#	項目	内容
1	type	パケットの種類. Recv パケットを表す.
2	size	Recv では常に 0.
3	handle	コミュニケータ (ハンドル).
4	remote	送信元のランク (または −1: MPI_ANY_SOURCE).
5	tag	通信タグ.
6	transsize	受信バッファのバイト数.

表 5.20: Recv パケットの応答

#	項目	内容
1	type	パケットの種類. ACK (成功) または NACK(失敗).
2	size	受信したデータのサイズ (ACK のとき. NACK では常に 0).
3	handle	常に 0.
4	remote	送信元ランク.
5	tag	送信タグ.
6	transsize	size と同じ値となる.
7	data	受信したデータの実体

5.8.3 Info_create プロトコル

(1) 概要

Info_create プロトコルは MPI_Info_create に対応したプロトコルであり, Info_create パケットとして表現される. Info_create パケットは, 応答として ACK パケット (成功) または NACK パケット (失敗) が返される.

(2) 処理条件

Info_create プロトコルは MPI_Info クラスを生成した際に使用される.

(3) 内容

Info_create パケットの内容を表 5.21に示す. また, Info_create パケットの応答を表 5.22に示す.

表 5.21: Info_create パケットの内容

#	項目	内容
1	type	パケットの種類. Info_create パケットを表す.
2	size	常に 0.
3	handle	常に 0.
4	remote	常に 0.
5	tag	常に 0.
6	transsize	常に 0.

表 5.22: Info_create パケットの応答

#	項目	内容
1	type	パケットの種類. ACK (成功) または NACK(失敗).
2	size	常に 0.
3	handle	Info オブジェクト (ハンドル).
4	remote	常に 0.
5	tag	常に 0.
6	transsize	常に 0.

5.8.4 Info_set プロトコル

(1) 概要

Info_set プロトコルは MPI_Info_set に対応したプロトコルであり, Info_set パケットとして表現される. Info_set パケットは, 応答として ACK パケット (成功) または NACK パケット (失敗) が返される.

(2) 処理条件

Info_set プロトコルは MPI_Info クラスの Set メソッドを呼び出した際に使用される.

(3) 内容

Info_set パケットの内容を表 5.23に示す. また, Info_set パケットの応答を表 5.24に示す.

表 5.23: Info_set パケットの内容

#	項目	内容
1	type	パケットの種類. Info_set パケットを表す.
2	size	data のバイト数.
3	handle	Info オブジェクト (ハンドル).
4	remote	常に 0.
5	tag	常に 0.
6	transsize	常に 0.
7	data	"*key*:*value*" 形式の文字列.

表 5.24: Info_set パケットの応答

#	項目	内容
1	type	パケットの種類. ACK (成功) または NACK(失敗).
2	size	常に 0.
3	handle	常に 0.
4	remote	常に 0.
5	tag	常に 0.
6	transsize	常に 0.

5.8.5 Spawn プロトコル

(1) 概要

Spawn プロトコルは MPI_Comm_spawn に対応したプロトコルであり, Spawn パケットとして表現される. Spawn パケットは, 応答として ACK パケット (成功) または NACK パケット (失敗) が返される.

(2) 処理条件

Spawn プロトコルは MPI_Intracomm クラスの Spawn メソッドを呼び出した際に使用される.

(3) 内容

Spawn パケットの内容を表 5.25に示す. また, Spawn パケットの応答を表 5.26に示す.

表 5.25: Spawn パケットの内容

#	項目	内容
1	type	パケットの種類. Info_set パケットを表す.
2	size	data のバイト数.
3	handle	Info オブジェクト (ハンドル).
4	remote	起動する MPI プロセス数.
5	tag	常に 0.
6	transsize	常に 0.
7	data	"*command arg...*" 形式の文字列.

表 5.26: Spawn パケットの応答

#	項目	内容
1	type	パケットの種類. ACK (成功) または NACK(失敗).
2	size	常に 0.
3	handle	コミュニケータ (ハンドル).
4	remote	起動した MPI プロセス数.
5	tag	常に 0.
6	transsize	常に 0.

6 処理方式

6.1 Stampi起動コマンド

(1) 概要

Stampi起動コマンドは, 異機種間でのMPI起動の制御を行う.

(2) 処理条件

Stampi起動コマンドは, 以下の条件で起動される.

(a) 利用者が, Stampi起動コマンドを起動したとき起動される.

(b) 利用者プログラムで `MPI_Comm_spawn` を実行したとき, 他計算機の通信制御プログラムから起動される. (但し, バッチ起動の場合は, NQS起動コマンドが作成するスクリプト経由での起動となる)

(3) 処理内容

Stampi起動コマンドは, 以下の処理を行う.

(a) NQS起動コマンドから起動された場合, 標準出力と標準エラー出力をNQS起動コマンドの通信ポートに振り向ける.

(b) 通信制御プログラムを起動する. slave側の場合には, 通信制御プログラムにmaster側通信制御プログラムと接続するためのポート情報を引き渡す.

(c) 起動した通信制御プログラムからポート情報を受け取る.

(d) ベンダ提供のMPI起動コマンドを起動し, 利用者プログラムの実行を開始する. このとき, 通信制御プログラムから受け取ったポート情報を利用者プログラムに引き渡す.

(e) 利用者プログラムの終了を待ち合わせる.

(f) 通信制御プログラムの終了を待ち合わせる.

6.2 通信制御プログラム

(1) 概要

異機種間ので通信の調整 (ネゴシエーション) と通信の中継を行う. また, master 側通信制御プログラムは, 利用者プログラムの要求 (MPI_Comm_spawn) に基づき, slave 側 Stampi 起動コマンドを起動する.

(2) 処理条件

通信制御プログラムは, Stampi 起動コマンドが起動した際, 無条件に起動される.

(3) 処理内容

Stampi 起動コマンドは, 以下の処理を行う.

(a) slave 側の場合 (Stampi 起動コマンドから master 側ポート情報を受け取っている), master 側通信制御プログラムに接続する.

(b) 通信ポートを作成し, ポート情報を Stampi 起動コマンドに通知する.

(c) 通信ポートからの接続, 及び, 接続済みポートからの通信を待ち, 接続か通信があった場合, 以下を行う.

(i) 通信ポートからの接続に対しては, 新しいポートを作成 (accept) し, ポート管理テーブルに登録する.

(ii) 接続済みのポートで, 初めての通信の場合, 接続パケットが送付されるので, 内容を確認し, ポート管理テーブルを更新する.

(iii) その他の場合, パケットの内容に応じて以下を行う.

- SPAWN 系パケットの場合は, 他計算機で MPI プロセスを起動する.
- 送受信系パケットの場合は, 対応する送受信パケットの到着を待って, 通信相手のプロセスのパケットを送付する.
- その他のパケットの場合, 通信相手にパケットを送付する.
- パケットを受信せず, EOF を検出した場合は, そのポートを閉じる.

尚, 通信相手とは, 他計算機の通信制御プログラムからの通信の場合は計算機内の MPI プロセスであり, 計算機内の MPI プロセスからの通信の場合は他計算機の通信制御プログラムである.

(iv) 全ての MPI プロセスのポートが閉じたら, 処理を終了する.

6.3 NQS起動コマンド

(1) 概要

NQS起動コマンドは, MPI_Comm_spawn で生成するプロセスをNQSのバッチ上で実行するように制御する.

(2) 起動条件

MPI_Comm_spawn の実行の際に, NQSバッチキューの指定がある場合, 通信制御プログラムから起動される.

(3) 処理内容

NQS起動コマンドは以下の処理を行う.

(a) Stampi起動コマンドと接続する通信ポートを作成する. 本ポートでは, Stampi起動コマンドの標準出力と標準エラー出力を受け取る.

(b) NQSのジョブキューに投入するバッチファイル (shell script) を作成する. ファイルは, Stampi起動コマンドの実行を含む.

(c) NQSのジョブ投入コマンド (qsub) を起動し, バッチファイルをジョブキューに投入する.

(d) (a) で作成したポートに対する接続を待ちながら, 定期的に投入したジョブが終了していないかをqstat コマンドを用いて調べる. もし, ジョブが終了してキューから消えていた場合は, NQS起動コマンドの実行を終了する.

(e) (a) で作成したポートに接続があった場合, ポートの接続が切れるまでポートから入力したデータを標準出力に転送する. ポートの接続が切れたら処理を終了する.

6.4 MPI関数処理方式

6.4.1 MPI_COMM_SPAWN

(1) 概要

MPIプロセスの生成を行う．具体的には，slave Stampi起動コマンドを起動してslave側利用者プログラムと通信する準備を行う．

(2) 処理条件

利用者プログラムから`MPI_Comm_spawn`関数の呼び出しがあった際に起動される．

(3) 処理内容

`MPI_Comm_spawn`関数は，以下の処理を行う．

(a) rootプロセスの場合，以下の処理を行う．

(i) Stampi起動コマンドに接続する．

(ii) SPAWN開始パケット，SPAWN infoパケット，SPAWN実行パケットの順に送付し，子プロセスの生成を要求する．

(iii) 子プロセス生成の成否と中継プログラムのポート情報を受け取る．

(b) rootプロセスから他のプロセスに子プロセス生成の成否と中継プログラムのポート情報をブロードキャストする．

(c) 子プロセスの生成に成功した場合，以下の処理を行う

(i) 中継プログラムに接続し，ランク情報を通知する．

(ii) 同期を取る．

(iii) 相互コミュニケータを作成し，返却する．

6.4.2 MPI_COMM_GET_PARENT

(1) 概要

親プロセスのコミュニケータを返却する. 親プロセスがない場合 (Master 側の場合), MPI_COMM_NULL を返却する.

(2) 処理条件

利用者プログラムから MPI_Comm_get_parent 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_get_parent 関数は, 以下の処理を行う.

(a) コモン領域に格納されている親コミュニケータを返却する.

6.4.3 MPI_OPEN_PORT

(1) 概要

Client/Server 型通信の Server ポートをオープンする.

(2) 処理条件

利用者プログラムから MPI_Open_port 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Open_port 関数は, 以下の処理を行う.

(a) Stampi 起動コマンドに接続し, ポート生成パケットを送付し, ポートの割り当てを受ける.

(b) Stampi 起動コマンドからポートの名称を取得する.

(c) ポート名を設定し, 呼び元に戻る.

6.4.4 MPI_CLOSE_PORT

(1) 概要

MPI_Open_port 関数でオープンした Server ポートをクローズする.

(2) 処理条件

利用者プログラムから MPI_Close_port 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Close_port 関数は, 以下の処理を行う.

(a) Stampi 起動コマンドに接続する.

(b) ポートクローズパケットを送付してポートをクローズする.

6.4.5 MPI_COMM_ACCEPT

(1) 概要

Client/Server 型の通信において, Server 側で Client からの接続を受け付けを行う.

(2) 処理条件

利用者プログラムから MPI_Comm_accept 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_accept 関数は, 以下の処理を行う.

(a) root プロセスの場合, 以下の処理を行う.

(i) Stampi 起動コマンドに接続する.

(ii) accept パケットを送付し, Client からの接続受け付けを要求する.

(iii) 中継プログラムのポート情報を受け取る.

(b) root プロセスから他のプロセスに中継プログラムのポート情報をブロードキャストする.

(c) 中継プログラムに接続し, ランク情報を通知する.

(d) 同期を取る.

(e) 相互コミュニケータを作成し, 返却する.

6.4.6 MPI_COMM_CONNECT

(1) 概要

Client/Server型の通信において, Clinetから Serverのポートに接続を行う.

(2) 処理条件

利用者プログラムから MPI_Comm_connect 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_connect 関数は, 以下の処理を行う.

(a) root プロセスの場合, 以下の処理を行う.

(i) Stampi起動コマンドに接続する.

(ii) connectパケットを送付し, Serverへの接続を要求する.

(iii) 中継プログラムのポート情報を受け取る.

(b) rootプロセスから他のプロセスに中継プログラムのポート情報をブロードキャストする.

(c) 中継プログラムに接続し, ランク情報を通知する.

(d) 同期を取る.

(e) 相互コミュニケータを作成し, 返却する.

6.4.7 MPI_COMM_DISCONNECT

(1) 概要

Client/Server型の通信において, MPI_Comm_accept または MPI_Comm_connect で確立した接続を切断する.

(2) 処理条件

利用者プログラムから MPI_Comm_disconnect 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_disconnect 関数は, 以下の処理を行う.

(a) 中継プログラムとの接続を切断する.

(b) 相互コミュニケータを解放する.

6.4.8 MPI_SEND

(1) 概要

コミュニケータ内の他プロセスにデータを送信する．送信はブロッキング通信によって行う．

(2) 処理条件

利用者プログラムから MPI_Send 関数の呼び出しがあった際に起動される．

(3) 処理内容

MPI_Send 関数は，以下の処理を行う．

(a) コミュニケータが同一機種内の場合，ベンダ提供の PMPI_SEND を呼び出し，処理を終了する．

(b) 送信要求パケットを作成し，通信制御プログラムに送付する．

(c) 通信制御プログラムから NACK パケットを受け取った場合，エラーとする．

(d) 受信要求パケットを受け取った場合，受信サイズと送信サイズを比較し，小さいサイズでデータ転送パケットを作成し，データを送信する．データがバッファサイズを超える場合，パケットを複数に分割して送付する．

(e) 通信相手プロセスから応答を受信する．ACK パケットを受信した場合，送信を成功とし，NACK パケットを受信した場合には，エラーとする．

6.4.9 MPI_RECV

(1) 概要

コミュニケータ内の他プロセスからデータを受信する．受信はブロッキング通信によって行う．

(2) 処理条件

利用者プログラムから MPI_Recv 関数の呼び出しがあった際に起動される．

(3) 処理内容

MPI_Recv 関数は，以下の処理を行う．

(a) コミュニケータが同一機種内の場合，ベンダ提供の PMPI_RECV を呼び出し，処理を終了する．

(b) 受信要求パケットを作成し，通信制御プログラムに送付する．

(c) 通信制御プログラムから NACK パケットを受け取った場合，エラーとする．

(d) 送信要求パケットを受け取った場合，受信サイズと送信サイズを比較し，小さいサイズでデータを受信する．データサイズが大きく，バッファに入りきらない場合は，パケットを分割して受信する．

(e) 受信に成功した場合，ACK パケットを通信相手のプロセスに送付する．何らかの事情で受信に失敗した場合は，NACK パケットを送付する．

6.4.10 MPI_GET_COUNT

(1) 概要

MPI_Recv や MPI_Irecv で受信したデータ数を返却する.

(2) 処理条件

利用者プログラムから MPI_Get_count 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Get_count 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_GET_COUNT を呼び出す.

6.4.11 MPI_ISEND

(1) 概要

コミュニケータ内の他プロセスにデータを送信する. 送信はノンブロッキング通信によって行う. このため, MPI_Isend 終了後, MPI_Wait を行う必要がある.

(2) 処理条件

利用者プログラムから MPI_Isend 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Isend 関数は, 以下の処理を行う.

(a) コミュニケータが同一機種内の場合, ベンダ提供の PMPI_ISEND を呼び出す. その後, 返却された通信要求ハンドルを Stampi の非同期通信ハンドルに変換して, 呼び元に返し, 関数を終了する.

(b) Stampi の非同期通信ハンドルを作成する.

6.4.12 MPI_IRECV

(1) 概要

コミュニケータ内の他プロセスからデータを受信する．受信はノンブロッキング通信によって行う．このため，MPI_Irecv 終了後，MPI_Wait を行う必要がある．

(2) 処理条件

利用者プログラムから MPI_Irecv 関数の呼び出しがあった際に起動される．

(3) 処理内容

MPI_Irecv 関数は，以下の処理を行う．

(a) コミュニケータが同一機種内の場合，ベンダ提供の PMPI_IRECV を呼び出す．その後，返却された通信要求ハンドルを Stampi の非同期通信ハンドルに変換して，呼び元に返し，関数を終了する．

(b) Stampi の非同期通信ハンドルを作成する．

6.4.13 MPI_WAIT

(1) 概要

ノンブロッキング通信の終了待ち合わせを行う．MPI_Isend や MPI_Irecv を行った後に呼び出す．

(2) 処理条件

利用者プログラムから MPI_Wait 関数の呼び出しがあった際に起動される．

(3) 処理内容

MPI_Wait 関数は，以下の処理を行う．

(a) 対応する非同期入出力関数 (MPI_Isend, MPI_Irecv) のコミュニケータが同一機種内の場合，通信要求ハンドルをベンダ提供の通信要求ハンドルに変換した後，ベンダ提供の PMPI_WAIT を呼び出すし，関数を終了する．

(b) Stampi の非同期通信ハンドルから，通信情報を取り出し，MPI_Send かMPI_Recv を実行する．

6.4.14 MPI_BARRIER

(1) 概要

コミュニケータ内の全プロセスでバリア同期を行う.

(2) 処理条件

利用者プログラムから MPI_Barrier 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Barrier 関数は, 以下の処理を行う.

(a) コミュニケータが同一機種内のものである場合, ベンダ提供の PMPI_BARRIER を呼び出し, 関数を終了する.

(b) コミュニケータに対応する内部コミュニケータ (MPI_Comm_spawn を実行したプロセスの場合は, MPI_Comm_spawn を実行したときのコミュニケータ, そうでない場合は, MPI_Comm_world) でバリア同期をとる.

(c) (b) のコミュニケータでランク 0 のプロセス同士で, バリア同期パケットを交換する.

(d) 再び, (b) のバリア同期を行う.

6.4.15 MPI_INIT

(1) 概要

MPI 初期化関数. Stampi の初期化を行う.

(2) 処理条件

利用者プログラムから MPI_Init 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Init 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_INIT を呼び出す.

(b) 通信制御プログラムに接続し, 接続パケットを送付する.

6.4.16 MPI_FINALIZE

(1) 概要

MPI 終了関数. Stampi の終了処理を行う.

(2) 処理条件

利用者プログラムから MPI_Finalize 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Finalize 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_FINALIZE を呼び出す.

6.4.17 MPI_ABORT

(1) 概要

コミュニケータ内のプロセスの実行を中断する.

(2) 処理条件

利用者プログラムから MPI_Abort 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Abort 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_ABORT を呼び出す.

6.4.18 MPI_COMM_SIZE

(1) 概要

コミュニケータ内のプロセス数を返却する.

(2) 処理条件

利用者プログラムから MPI_Comm_size 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_size 関数は, 以下の処理を行う.

(a) コミュニケータが同一機種内の場合, ベンダ提供の PMPI_COMM_SIZE を呼び出し, 関数を終了する.

(b) コミュニケータ管理テーブル内のサイズを返却する.

6.4.19 MPI_COMM_RANK

(1) 概要

コミュニケータ内での自プロセスのランクを返却する.

(2) 処理条件

利用者プログラムから MPI_Comm_rank 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_rank 関数は, 以下の処理を行う.

(a) コミュニケータが同一機種内の場合, ベンダ提供の PMPI_COMM_RANK を呼び出し, 関数を終了する.

(b) コミュニケータ管理テーブル内のランクを返却する.

6.4.20 MPI_COMM_REMOTE_SIZE

(1) 概要

指定された親プロセスグループ (または子プロセスグループ) のプロセス数を返却する.

(2) 処理条件

利用者プログラムから MPI_Comm_remote_size 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_remote_size 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用コミュニケータでない場合, ベンダ提供の PMPI_COMM_REMOTE_SIZE を呼び出し, 関数を終了する.

(b) コミュニケータ管理テーブル内のサイズを返却する.

6.4.21 MPI_COMM_DUP

(1) 概要

指定されたコミュニケータの複製を作成し, 返却する.

(2) 処理条件

利用者プログラムから MPI_Comm_dup 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_dup 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用のコミュニケータの場合, エラーとする.

(b) ベンダ提供の PMPI_COMM_DUP を呼び出す.

6.4.22 MPI_COMM_FREE

(1) 概要

指定されたコミュニケータを開放する.

(2) 処理条件

利用者プログラムから MPI_Comm_free 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_free 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用のコミュニケータの場合, エラーとする.

(b) ベンダ提供の PMPI_COMM_FREE を呼び出す.

6.4.23 MPI_COMM_SPLIT

(1) 概要

コミュニケータ内のプロセスをサブグループに分割することによって, 新しいコミュニケータを定義する.

(2) 処理条件

利用者プログラムから MPI_Comm_split 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Comm_split 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_COMM_SPLIT を呼び出す.

6.4.24 MPI_SENDRECV

(1) 概要

コミュニケータ内の他プロセスとの間で送受信動作 (送信兼受信) を行う.

(2) 処理条件

利用者プログラムから MPI_Sendrecv 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Sendrecv 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_SENDRECV を呼び出す.

6.4.25 MPI_TYPE_EXTENT

(1) 概要

データ型の大きさ (バイト数) を求める.

(2) 処理条件

利用者プログラムから MPI_Type_extent 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Type_extent 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_TYPE_EXTENT を呼び出す.

6.4.26 MPI_TYPE_VECTOR

(1) 概要

等間隔に並んだデータ型のコピーから, 新しいデータ型を作成する. データの間隔は, データの数で表す.

(2) 処理条件

利用者プログラムから MPI_Type_vector 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Type_vector 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_TYPE_VECTOR を呼び出す.

6.4.27 MPI_TYPE_HVECTOR

(1) 概要

等間隔に並んだデータ型のコピーから, 新しいデータ型を作成する. データの間隔は, バイト数で表す.

(2) 処理条件

利用者プログラムから MPI_Type_hvector 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Type_hvector 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_TYPE_HVECTOR を呼び出す.

6.4.28 MPI_TYPE_COMMIT

(1) 概要

データ型の記憶操作を行う. 通信バッファの記述 (通信バッファの内容ではない) を記憶する.

(2) 処理条件

利用者プログラムから MPI_Type_commit 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Type_commit 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_TYPE_COMMIT を呼び出す.

6.4.29 MPI_BCAST

(1) 概要

コミュニケータ内の 1 つのプロセスから他の全プロセスに対してデータを送信 (または受信) する.

(2) 処理条件

利用者プログラムから MPI_Bcast 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Bcast 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用コミュニケータでない場合, ベンダ提供の PMPI_BCAST を呼び出す.

(b) root プロセスから対になるグループ内の root プロセスに対してデータを転送し, 対になるグループ側でベンダ提供の PMPI_BCAST を呼び出す.

6.4.30 MPI_REDUCE

(1) 概要

コミュニケータ内の全プロセスで大域的なリダクション操作 (総和, 最大値, 論理積など) を行い, 結果を特定のプロセスに返却する.

(2) 処理条件

利用者プログラムから MPI_Reduce 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Reduce 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用コミュニケータでない場合, ベンダ提供の PMPI_REDUCE を呼び出す.

(b) 対になるグループ内でPMPI_REDUCE を行い, root に対してデータを転送する.

6.4.31 MPI_ALLREDUCE

(1) 概要

コミュニケータ内の全プロセスで大域的なリダクション操作 (総和, 最大値, 論理積など) を行い, 結果を全プロセスに返却する.

(2) 処理条件

利用者プログラムから MPI_Allreduce 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Allreduce 関数は, 以下の処理を行う.

(a) コミュニケータが異機種間通信用コミュニケータでない場合, ベンダ提供の PMPI_ALLREDUCE を呼び出す.

(b) 対になるグループ内でPMPI_REDUCE を行い, root に対してデータを転送しかつグループ内でPMPI_ALLREDUCE を呼び出す.

6.4.32 MPI_CART_CREATE

(1) 概要

コミュニケータにカルテシアン (直積構造のトポロジ) 情報を付加した新しいコミュニケータを返却する.

(2) 処理条件

利用者プログラムから MPI_Cart_create 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Cart_create 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_CART_CREATE を呼び出す.

6.4.33 MPI_CART_SHIFT

(1) 概要

カルテシアン構造のトポロジにおいて, 座標方向へのデータシフト動作を支援する. 具体的には MPI_Sendrecv 関数に指定するランク (通信相手のコミュニケータ内のランク) を求める.

(2) 処理条件

利用者プログラムから MPI_Cart_shift 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Cart_shift 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_CART_SHIFT を呼び出す.

6.4.34 MPI_CART_COORDS

(1) 概要

コミュニケータ内のプロセスのランクを座標に変換する.

(2) 処理条件

利用者プログラムから MPI_Cart_coord 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Cart_coord 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_CART_COORD を呼び出す.

6.4.35 MPI_GATHER

(1) 概要

コミュニケータ内の全プロセスから１つのプロセスにスカラデータを収集し, ランクの順にデータを格納する.

(2) 処理条件

利用者プログラムから MPI_Gather 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Gather 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_GATHER を呼び出す.

6.4.36 MPI_GATHERV

(1) 概要

コミュニケータ内の全プロセスから1つのプロセスにベクトルデータを収集し, ランクの順にデータを格納する.

(2) 処理条件

利用者プログラムから MPI_Gatherv 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Gatherv 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_GATHERV を呼び出す.

6.4.37 MPI_WTIME

(1) 概要

ある時刻からの経過時刻を返却する. ある時刻は, 利用者プログラムの開始から変更されない. 返却する時刻は, MPI_Wtime を呼び出したノードにローカルである.

(2) 処理条件

利用者プログラムから MPI_Wtime 関数の呼び出しがあった際に起動される.

(3) 処理内容

MPI_Wtime 関数は, 以下の処理を行う.

(a) ベンダ提供の PMPI_WTIME を呼び出す.

6.5 Stampi/Java 通信中継モジュール

(1) 概要

Stampi/Java 通信中継モジュールは, Java アプレットとして WWW ブラウザ上で稼働する Stampi/Java アプリケーションと, 並列計算機などのバックエンド計算機で稼働する Stampi アプリケーションとの通信を中継することによって, 本来 WWW サーバ計算機としか通信できないという制約を持つ Java アプレットと任意のバックエンド計算機との間で Stampi 通信を実現する.

(2) 起動条件

Stampi/Java アプリケーションで `Stampi` クラスの `Init` メソッドを呼び出した際に起動される.

(3) 処理内容

Stampi/Java 通信中継モジュールは以下の処理を行う.

(a) 起動時, `MPI_Init` を行い, Stampi の初期化をする.

(b) Stampi/Java アプリケーションから Stampi/Java プロトコルが送信されるのを待ち合わせる.

(c) Stampi/Java プロトコルに従った処理を行う.

(d) EOF になるまで, (b) (c) を繰り返す.

(e) EOF になった場合, `MPI_Finalize` を行い, 処理を終了する.

6.6 Stampi/Java ライブラリ

(1) 概要

Stampi/Java ライブラリは, Java アプレットに Java 言語による MPI 関数インタフェースを提供する. MPI 関数の機能は, ほぼ Stampi/Java 通信中継モジュールの側で実現されており, Stampi/Java ライブラリは, Stampi/Java 通信中継モジュールへの関数パラメータの提供と, 呼び元への結果の返却を行う.

(2) 起動条件

Stampi/Java ライブラリが提供する MPI 関数を Java アプレットが呼び出した際に起動される.

(3) 処理内容

Stampi/Java ライブラリは以下の処理を行う.

(a) 関数パラメータを元に Stampi/Java プロトコルのパケットを組み立てる.

(b) (a) で組み立てたパケットを Stampi/Java 通信中継モジュールに送信する.

(c) Stampi/Java 通信中継モジュールからの応答を待つ.

(d) (c) の応答を元に返却値を作成し, 返却する.

7　データ及びファイル

7.1　内部データ

7.1.1　ポート管理テーブル

(1)　概要

通信制御プログラム内で使用する通信ポートを管理する.

(2)　処理条件

通信制御プログラムにネットワーク経由での接続があった場合に作成し, 通信の待ち合わせに使用する.

(3)　内容

ポート管理テーブルの内容を表 7.1に示す.

表 7.1: ポート管理テーブルの内容

#	項目	内容
1	ポート FD	ポートに接続されているファイルディスクリプタ (ソケット).
2	種別	接続待ちポート, MPI プロセス, 外部通信制御プログラムの種別.
3	ランク	MPI プロセスの場合, ランク値.

7.1.2 コミュニケータ管理テーブル

(1) 概要

Stampi 関数内で, コミュニケータを表す. 同一計算機内通信のコミュニケータか異機種間コミュニケータかの判別と, 異機種間コミュニケータの場合, 通信に必要な各種情報を保持する.

Stampi 関数内で, ベンダ提供のコミュニケータと置き換えて使用する.

(2) 処理条件

コミュニケータ生成関数 (MPI_Comm_spawn, MPI_Comm_split) で作成し, 各 Stampi 関数で使用する.

(3) 内容

コミュニケータ管理テーブルの内容を表 7.2に示す.

表 7.2: コミュニケータ管理テーブルの概要

#	項目	内容
1	種別	同一計算機内コミュニケータか異機種間コミュニケータかを表す.
2	コミュニケータ	同一計算機内コミュニケータの場合, ベンダ提供 MPI 関数のコミュニケータ.
3	ポート FD	異機種間通信をするための通信ポートのファイルディスクリプタ (ソケット).
4	内部コミュニケータ	異機種間コミュニケータにおいて, 対応する内部コミュニケータ.
5	サイズ	異機種間コミュニケータにおいて, コミュニケータのサイズ

7.1.3 非同期通信要求ハンドル管理テーブル

(1) 概要

非同期通信の情報を管理する.

(2) 処理条件

非同期通信関数 MPI_Isend, MPI_Irecv で作成し, MPI_Wait で参照, 削除する.

(3) 内容

非同期通信要求ハンドル管理テーブルの内容を表 7.3に示す.

表 7.3: 非同期通信要求管理テーブルの内容

#	項目	内容
1	コミュニケータ種別	内部コミュニケータによる通信か, 異機種間コミュニケータによる通信かの種別
2	通信種別	送信か受信かの種別
3	コミュニケータ	通信に使用するコミュニケータ
4	バッファ	バッファアドレス
5	型	データの型
6	要素数	転送するデータの要素数
7	送信元ランク	送信元プロセスのランク
8	受信先ランク	受信先プロセスのランク
9	タグ	通信タグ

7.2 MPI_COMM_SPAWN に引き渡すデータ

MPI_Comm_spawn の *info* に引き渡すことができるデータには表 7.4のものがある.

表 7.4: MPI_COMM_SPAWN に引き渡すことができるデータ

#	キー	説明
1	host	プロセス生成先ホスト
2	user	プロセスを実行するユーザ
3	wdir	プロセスを実行するディレクトリ
4	path	コマンドサーチパス
5	part	プロセスを実行するパーティション (SR2201 のみ)
6	nqsq	NQS のキュー (指定しない場合, TSS で実行)
7	node	NQS で使用するプロセッサノード数 (省略時はプロセス数)
8	file	サブプロセス起動ファイル

7.3 サブプロセス起動ファイル

MPI_Comm_spawn の *info* に file キーのデータとしてサブプロセス起動ファイルを引き渡すことができる. サブプロセス起動ファイルは, 行頭から,

key: *value*

という行の集まりで表現される. *key* は, データを識別するキーで, *value* はキーに対応する値である.

もし, MPI_Comm_spawn の *info* と, サブプロセス起動ファイルで同一の情報が定義されている場合, サブプロセス起動ファイルの定義が優先される.

サブプロセス起動ファイルに記述できるデータには, 表 7.5のものがある.

表 7.5: サブプロセス起動ファイルに記述できるデータ

#	キー	説明
1	host	プロセス生成先ホスト
2	user	プロセスを実行するユーザ
3	wdir	プロセスを実行するディレクトリ
4	path	コマンドサーチパス
5	part	プロセスを実行するパーティション (SR2201 のみ)
6	nqsq	NQS のキュー (指定しない場合, TSS で実行)
7	node	NQS で使用するプロセッサノード数 (省略時はプロセス数)
8	-cmd	起動する利用者プログラムとパラメタ (MPI_Comm_spawn の指定を置き換える)
9	-np	起動する MPI プロセス数 (MPI_Comm_spawn の指定を置き換える)

おわりに

本報告書では異機種並列計算機間通信ライブラリ Stampi に関する，システム構成，各機能詳細，インターフェイス，実現手法等々についての詳細設計仕様をまとめた．Stampi は MPI2 による異機種並列計算機間通信を初めて実現したライブラリであり，更なる機能拡張と性能向上がなされていく予定である．今後，本ライブラリを用いたアプリケーションの成果が多く出されることを著者らは希望します．

謝辞

本報告書を取りまとめるにあたり，貴重な機会を与えてくださいました計算科学技術推進センター長秋元正幸氏，並列処理基本システム開発グループリーダー平山俊雄氏に深謝致します．また，本ライブラリ開発にあたり，甚大なる御支援を賜りました早稲田大学理工学部笠原博徳教授に謝意を表します．利用者の立場から数々の貴重な意見を頂いた並列計算法開発グループ木村俊哉副主任研究員，並びに並列処理基本システム開発グループ各員に感謝致します．

参考文献

[1] Message Passing Interface Forum : *MPI: Message passing interface standard* (1996). `http://www.mpi-forum.org/`

[2] Message Passing Interface Forum : *MPI-2: Extensions to the message-passing interface* (1997). `http://www.mpi-forum.org/`

[3] 今村, 小出, 武宮： 異機種並列計算機間通信ライブラリ:Stampi－利用手引書, JAERI Data/Code 98-034 (1998).

[4] Gropp W. and Lusk E. : *User's Guide for mpich, a Portable Implementation of MPI* (1998). `http://www-c.mcs.anl.gov/mpi/mpich/`

[5] サン・マイクロシステムズ, `http://java.sun.com/`

[6] 日立製作所: *HI-UX/MPP MPI·PVM*使用の手引き, 日立ソフトウェアマニュアル 6A20-3-026-10 (1998).

[7] 富士通株式会社: *UXP/V MPI*使用手引書 *V11*用, 富士通マニュアル J2U5-0271-01 (1998).

[8] 日本 IBM: *Parallel Environment for AIX: MPI Programming and Subroutine Reference*, Ver. 2, Rel. 1, IBM manual GC23-3894-00 (1995).

[9] CRAY Research Inc.: *Message Passing Toolkit: Release Overview*, CRAY manual RO-5290 1.1 (1996).

[10] CRAY Research Inc.: *Message Passing Toolkit: MPI Programmer's Manual*, CRAY manual SR-2197 1.1 (1996).

[11] 日本電機株式会社: *SUPER-UX MPI/SX*利用の手引き, 日本電気マニュアル G1AF09-1 (1996).

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	$m\cdot kg/s^2$
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

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

（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)

異機種並列計算機間通信ライブラリ Stampi 設計書