マルチメディア通信と分散処理 111－34
コンピュータセキュリティ 20－34
（２００３．２．２８）

大規模分散システムに向けた高信頼化機構の設計

村山和宏† 落合真一†

近年，電話の交換機システムやレーダのデータ処理システムといった社会インフラを支えるシステムなどにHPCクラスタを採用する動きがある．ミッションクリティカルなシステムでHPCクラスタを動作させるためには，システム異常時もシステム性能を維持しつつ連続運転が実現できるようなソフトウェア機構を組み込むことが必要である．我々は大規模HPCクラスタの高信頼化を実現するため，システム故障を検出し，故障伝播の防止および故障からの自動復旧を可能とする高信頼化機構を開発した．実装した高信頼化機構は，大規模クラスタの統合管理，プロセッサの動的構成変更実現のほか，並列処理ミドルウェア：MPIとの連携により，アプリケーションの自動縮退運転，自動機能回復運転をサポートする．

Design of high-available middleware for large-scale distributed systems

Kazuhiro Murayama † and Shinichi Ochiai†

Recently, high performance computing (HPC) clusters can be applied to mission-critical systems, such as telecommunication systems and radar information processing systems. To use HPC clusters for those systems, it is required to adopt the software mechanism that can support high-performance and non-stop operation in case of failure. For this reason, we have been developing the high-availability middleware which has two distinctive features about fault tolerant management; (1) automatic recovery from failure, (2) dynamic system configuration on MPI. In this paper, we describe the design of our middleware, MPI/SP.

1. はじめに

クラスタは以下の３種類に分類することができる．

- フェイルオーバー型クラスタ：銀行のオンラインシステム，電力システムなどの社会インフラを支えるシステムに使用．システムを多重化し，主系と待機系で同期を取り合い，主系故障時には待機系により運用を継続することにより，24時間365日の連続運転を実現する．
- 負荷分散クラスタ：webサーバなどに使用．ロードバランサが複数台のサーバに均等にジョブを振り分けることにより，外部アクセスによる高負荷を回避する．
- HPCクラスタ：科学技術計算に使用．多数のプロセッサを使用することにより高い演算性能を実現する．

近年，大規模プロセッサによるHPCクラスタを電話の交換機システム，レーダのデータ処理システムなどの社会インフラを支えるシステムに採用する動きがある．ミッションクリティカルなシステムでHPCクラスタを動作させるためには，システム多重化技術などのフェイルオーバー型クラスタの技術を融合させる必要がある．

我々は，センサ信号処理システムに大規模HPCクラスタを適用するため，故障検出，故障時における他への伝播防止を実現し，プロセッサ故障からの自動復旧をサポートする高信頼化ミドルウェアの設計を行なった．さらに，業界標準の並列処理ミドルウェアであるMPI[4]（Message Passing Interface）と連携することによりプロセッサ構成変更を隠蔽し，無停止連続運用が可能な並列処理アプリケーション構築の容易化を実現した．以下，高信頼化機構の設計内容，およびMPIとの連携方法について述べる．

† 三菱電機（株）情報技術総合研究所
IT R&D Center, Mitsubishi Electric Corp

2. 背　景

2.1 ターゲットシステムの特徴

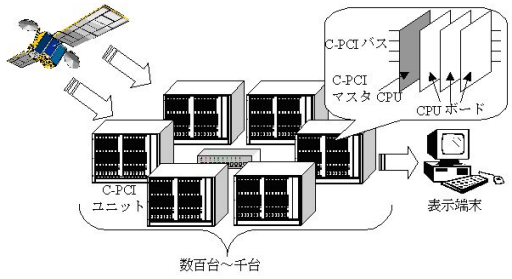

図1 ターゲットシステムの構成

図1は，開発ターゲットであるセンサ信号処理システムの概要を示したものである．本システムは以下の特徴を持つ．

CPU ボードクラスタ： 本システムは，N枚のCPUボードを持つCompact PCI（C-PCI）ユニットをネットワークで接続したボードクラスタ構成をとる．各CPUボードはネットワークインタフェースを持ち，これにより並列処理用のネットワークを構築する．

大規模プロセッサ構成： 本システムでは，膨大な量のデータを交換し，リアルタイム処理を行なう．そのため，数百台～千台程度のプロセッサでシステムを構築する．

プロセッサの M+N 重化構成： 本システムでは，プロセッサ故障に備えて数百台〜千台あるプロセッサの一部を待機系とする冗長化構成を採用している．故障時には待機系プロセッサに処理を切り替えることによって，システム性能を維持したまま運用を継続できる．

多重化ネットワーク構成： 本システム内では膨大な量のデータが交換される．そこで，ネットワークの負荷を分散させるため，プロセッサ間を複数の通信路で接続してシステムを構築している．また，通信路故障時は通信路を切り替えることによって継続運用が可能である．

2.1.1 ターゲットシステムの要求

ターゲットシステムは，システムの一部に故障が発生した場合においてもアプリケーションが停止することなく 24 時間 365 日の無人連続運転を行なうことが求められている．

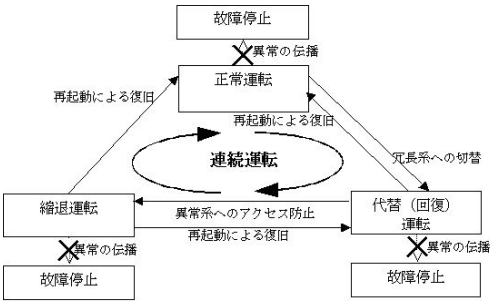

図 2 連続運転実現のために

本要求を実現するには，システムの一部に異常が発生した場合には待機系に自動的に切り替え（代替運転），待機系も全て故障した場合には性能を低下させて運転を継続する（縮退運転）とともに，自動的に故障箇所を復旧させて運転を行なう（回復運転）ことが必要となる（**図 2**）．

2.2 課　　題

2.2.1 関連研究：SPF

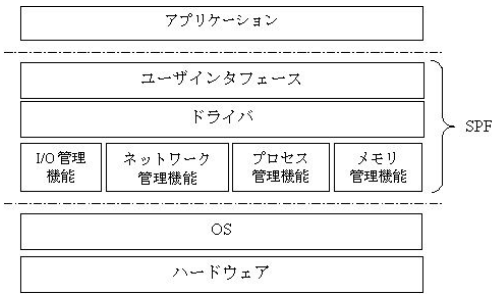

図 3 SPF のソフトウェア構成

本研究に関連し，大規模クラスタの高信頼化を実現する研究としては文献 3) などがある．3) では，並列分散環境において，故障を検出し，故障から復旧した場合にはアプリケーションの状態を異常が発生する直前に戻して処理を継続することが可能なソフトウェアアーキテクチャ：SPF（Software-Implemented, Parallel-System Fault Tolerant）を設計している．SPF は I/O 管理，ネットワーク管理，プロセス管理，メモリ管理のサブモジュールを持ち，それぞれネットワーク異常，プロセス異常，メモリデータの管理などを行なっている（**図 3**）．

2.2.2 高信頼並列処理ミドルウェア：MPI/SP

SPF 同様，我々も大規模クラスタ上での高信頼並列処理を実現するための研究開発を行なっており，これまでに大規模クラスタ上での CPU 間通信を実現する MPI/SP[1)]，および高信頼化機構 HAM[2)] を設計している． MPI/SP と HAM は連携可能であり，この二つを組み合わせることにより高信頼なプロセッサ間通信を実現することができる．

MPI/SP と HAM を組み合わせた場合のソフトウェア構造を**図 4** に示す．

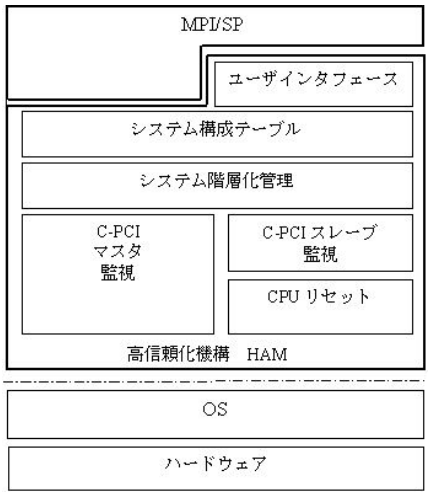

図 4 MPI/SP，HAM を連携した構成

HAM， MPI/SP の特徴は以下の通りである．

- プロセッサの故障監視，自動復旧：

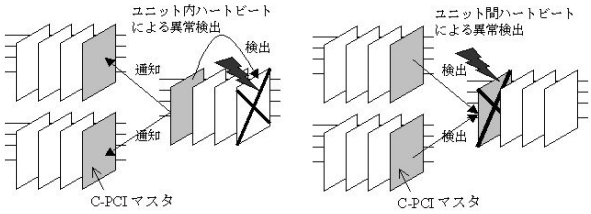

図 5 構成管理機構 HAM による故障検出

HAM では，各 CPU 間でハートビートを行なうことによって CPU の故障監視を行なう（**図 5**）．各プロセッサは，ハートビートの有無により CPU 故障を検出し，故障時には CPU ボードの自動再起動を行なってシステムの長期にわたる性能低下を防止する．

- 階層型クラスタによるシステム構成管理：
HAM では， C-PCI マスタ CPU の中から 1 台「システムマスタ」を選択する．「システムマスタ」が C-PCI マスタの故障検出を行い， C-PCI マスタが同一 C-PCI ユニット内のスレーブ CPU の故障検出を行なう階層型クラスタ管理により，全 CPU の故障検出を可能とす

る．また，異常検出時には階層構造を利用して CPU 情報を送信することにより全プロセッサでのシステム構成情報の共有を実現する（**図 6**）．

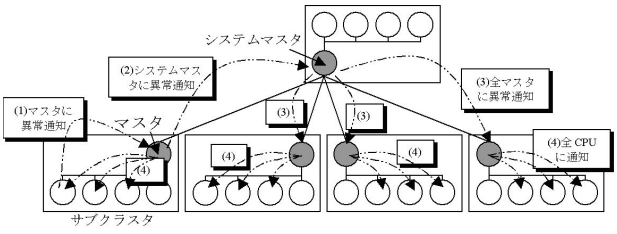

図 6　階層型クラスタと異常情報の伝達

- MPI インタフェースでの自動縮退，回復運転実現：MPI/SP では，異常プロセッサを並列処理から切り離して処理を継続（縮退運転）し，復旧したプロセッサと通信コネクションを設立して並列処理に参入する（回復運転）機能を MPI インタフェースに拡張している．

2.2.3　本研究の課題

しかし，ターゲットシステムの要求を実現するためには，以下の課題を解決する必要がある．

- 最適な異常対処：現状の HAM では，CPU 異常しか検出できず，異常時には CPU ボードの再起動を行なうため，CPU 以外の障害発生時には適切な障害復旧が行なわれず，性能回復が遅れる原因ともなる．
- MPI/SP での自動構成制御対応：現状の MPI/SP では待機系プロセッサの参入などの自動構成制御に対応しておらず，待機系への処理の切替はアプリケーションが行なう必要がある（**図 7**）．そのため，プログラムの記述が複雑になる．

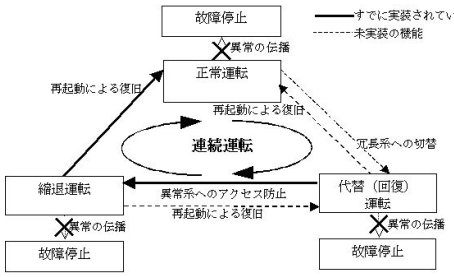

図 7　現状の MPI/SP が提供可能な機能

本研究は，HAM を拡張して詳細な障害検出や障害への適切な対処を可能とすることによってアプリケーションの無停止連続運用をサポートするとともに，MPI/SP を拡張し，HAM と MPI/SP の連携を強化することによりターゲットシステム上での高信頼アプリケーション構築を容易にすることを目的とする．

3.　高信頼化機構の設計

本研究では，2.2.3 節で示した課題を解決するために，高信頼化機構に対し以下の拡張を行なう．

- 異常対処機能の拡張：異常検出の精度を向上し，適切な異常対処を実現する．
- MPI/SP との連携：MPI/SP の機能拡張によって MPI インタフェースでの代替運転を実現する．

これらの設計内容について述べる．

3.1　異常対処機能の拡張

3.1.1　設計方針

設計方針は以下の通り：

- システムの性能保持：異常検出精度を向上させ，異常を検出した場合には異常の大小に関わらず直ちに並列処理から切り離して待機系プロセッサとの切替を行なうことにより，処理性能の低下を防ぐ．
- 異常判別・適切な対処の実現：異常プロセッサに対しては，故障箇所を判別し，適切な対処を取ることにより復旧の高速化を目指す．

3.1.2　設計内容

本研究では，異常対処機能の拡張に関し，以下の 2 点の拡張を行なう．

- 異常検出機能の拡張：検出する異常の詳細化を実現．
- 異常判別機能の拡張：異常レベル（軽故障・重故障）を判別し，適切な対処を実現．

異常検出機能の拡張：

システムを構成する要素としては，CPU· メモリなどの CPU 周辺機器，ネットワーク機器，OS，アプリケーションがある．これらのどれか一つに異常が発生しても処理を継続することはできない．

従来の HAM の異常検出機能は，CPU 監視タスクが C-PCI バスを経由して他のプロセッサ上の CPU 監視タスクにアクセスし，C-PCI バス，CPU 周辺機器，OS，CPU 監視タスクが正常に動作していることを確認する（**図 8**）．したがって，並列アプリケーション，ネットワークの障害は検出できない．

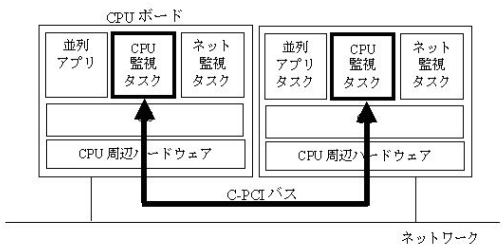

図 8　HAM による異常検出

これを解決するために，ネットワーク監視タスク，アプリケーション間でもメッセージを交換する（**図 9**）．

これにより，アプリケーション，ネットワークの検出が可能となり，CPU 監視タスクのメッセージ交換と合わせ，システムの構成要素全ての異常を検出することができ，システムの信頼性向上が実現できる．

異常を検出した場合，HAM によってすべてのプロセッサに異常発生が伝えられるため，正常に動作しているプロセッサは故障プロセッサへのアクセスを防止することができ

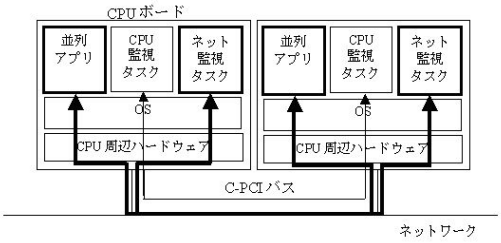

図 9 異常検出機能拡張

る．また，システムマスタが代替プロセッサを並列処理に参入させることにより，システム性能を低下させることなく処理を継続させることができる．

異常判別機能の拡張：

発見した故障プロセッサの復旧を高速化するため，これらの異常を以下のように分類する．

- 一時的異常：メッセージ到着の遅れを異常と判断したことによるもの（正常動作可能）
- 軽故障：CPU の一時的な異常，アプリケーション異常（再起動によって対処可能）
- 重故障：CPU 故障，ネットワーク故障（人手による対処が必要）．

一時的異常や軽故障の場合は直ちに復旧できるが，重故障の場合は復旧に時間がかかる．そこで，一部の通信路が故障した場合に限り待機系として運用を継続し，故障が頻発した場合のシステム性能低下を防止する．

異常時の対処方法は以下の通り：

- 一時的異常：自動復旧
- 軽故障：再起動
- 重故障：
  - CPU，全通信路の故障：
    プロセッサを停止し，ハードウェア交換を行なう．
  - 一部通信路の故障：
    待機系として運用を継続し，プロセッサに故障が発生した場合には並列処理に参入する．

3.1.3 実装内容

本研究では，3.1.2節に示した異常検出機能，異常対処機能を以下のようにして実装している．

故障検出機能の拡張：

- アプリケーション異常検出方法：
  図 10 に異常検出を行なうタスクの構成を示す．

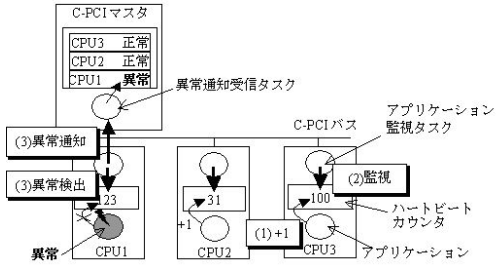

図 10 HAM によるアプリケーション異常検出

アプリケーション異常の検出手順は以下の通り：

(1) 一定周期でハートビートする関数をアプリケーションプログラムに埋め込む

(2) 各 CPU 上に，ハートビートを監視するタスクを起動し，アプリケーションの動作を監視する．

(3) 異常を検出したら，ハートビート監視タスクは C-PCI マスタ上にある「異常通知受信タスク」に向け，異常検出を通知する．C-PCI マスタは全プロセッサに向けてプロセッサ異常が発生したことを通知する．

- ネットワーク異常の検出：
  C-PCI マスタプロセッサ上にあるネットワーク監視タスクが各スレーブプロセッサに向けてハートビートなどを実行する．ハートビートに失敗した場合は全プロセッサに向けてプロセッサ異常が発生したことを通知する．

異常判別機能の拡張：

検出した異常は，C-PCI マスタプロセッサ上にある「システム構成テーブル」（**表 1**）に保存する．その際，重故障／軽故障の判別を可能にするため，システム構成テーブルの結果欄には以下のように記入する．

- 軽故障（CPU，アプリケーションの異常）：異常
- 重故障
  - 全通信路の異常：故障
  - 一部の通信路の異常：正常

表 1 システム構成テーブル

ID	状態					
No.	CPU	通信路 1	…	通信路 N	アプリ	結果
0	正常	正常	…	正常	正常	正常
1	正常	異常	…	正常	正常	正常
2	正常	異常	…	異常	正常	故障
3	異常	異常	…	正常	異常	異常

高信頼化機構は，結果欄に異常と示されているプロセッサに対し，以下の対処を行なう．

- CPU 異常：
  CPU 再起動を行ない，復旧すればシステム構成テーブルの CPU 欄および結果欄に正常と示す．再起動しても復旧しなければ重故障とし，システム構成テーブルの結果欄に故障と示す．
- アプリケーション異常：
  アプリケーションの再起動を行ない，復旧すればシステム構成テーブルアプリケーション欄および結果欄に正常と示す．再起動しても復旧しなければ，システム構成テーブルの CPU 欄に異常と示す．その後は CPU 異常（軽故障）として対処する．

3.2 MPI/SP との連携

MPI/SP を拡張し，高信頼化機構と連携させることにより，アプリケーション構築の容易化を目指す．今回開発する項目は，以下の 3 項目である（**図 11**）．

- 正常運転→代替運転への切替
- 縮退運転→回復運転への切替
- 代替（回復）運転→正常運転への切替

以下，これらの処理方法について述べる．

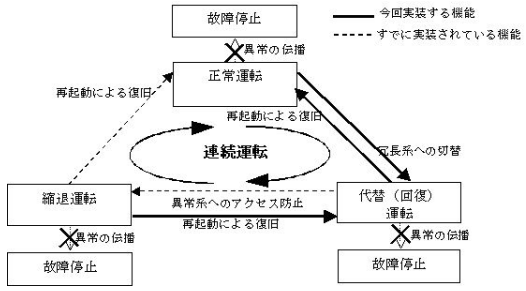

図 11　MPI/SP 拡張開発項目

3.2.1　設 計 方 針

本研究では，MPI 内部で自動的にプロセッサの切替や回復処理を行なうことにより，アプリケーション上で故障時の処理を記述することなく，高信頼なソフトウェア構築を実現することを目標とする．そのために，MPI が持つ以下の特徴を利用する．

- プロセッサ構成の抽象化：MPI は，プロセッサを「ランク」と呼ばれる番号で管理し，プロセッサの違いを隠蔽している．本機能を使用し，アプリケーションに影響を与えずにプロセッサの交換を実現する．
- ネットワーク構成の隠蔽：MPI は，メッセージの種別は「タグ」と呼ばれる番号でのみ管理し，ネットワークの経路を意識することなくメッセージ交換が可能である．本特徴により，アプリケーションに影響を与えずに使用するネットワークの切り替えを実現する．

3.2.2　設 計 内 容

(1)　正常運転→代替運転の切替：
プロセッサの代替運転は，正常に動作している待機プロセッサを故障したプロセッサのランクに割り当てることにより実現する．また，ネットワークの切替は，MPI 内部でシステム構成テーブルを参照し，正常に動作している通信路を選択することにより実現する．

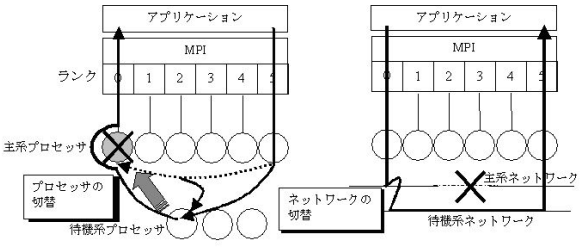

図 12　MPI による CPU ボード・ネットワーク代替運転

この設計により，アプリケーションからは該当ランクは常に動作しているように見える．したがって，アプリケーションはプロセッサの切替を意識することなく処理を継続することが可能である．
また，MPI ではユーザが通信路を選択する手段をもたないので，MPI 内部で自動的に通信路を切り替えてもアプリケーションに影響を与えることはない．受信側は，すべての通信路のメッセージ到着を監視することにより，代替通信路を用いてメッセージを送信しても受信することが可能である．受信したメッセージが意図したものかどうかは，メッセージと共に送信した「タグ」にて判別する．

(2)　縮退運転→回復運転の切替：待機系プロセッサが復旧した場合，(1) と同様の手法で復旧する．主系プロセッサが復旧した場合には，復旧したプロセッサをそのまま使用する（従来の高信頼化機構に実装済）．

(3)　回復運転→正常運転の切替：(2) の手法で復旧する．

3.2.3　実 装 内 容

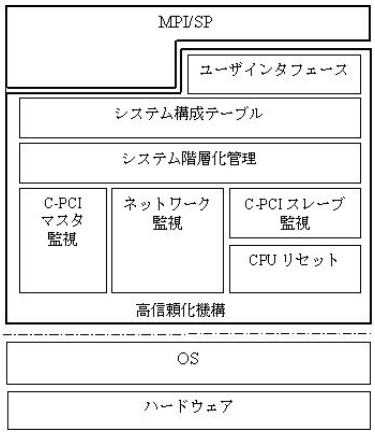

図 13　MPI/SP と高信頼化機構を連携させたソフトウェア構成

図 13 に本研究で MPI/SP と高信頼化機構を連携させた場合のソフトウェア構成を示す．高信頼化機構を MPI で隠蔽することにより，MPI インタフェースのみでの高信頼アプリケーション構築を実現している．

以下，MPI/SP と高信頼化機構の連携による代替運転，性能回復運転への切替方法を示す．

正常運転→代替運転の切替：

正常運転→代替運転の切替手順は以下の通り：

(1)　MPI/SP は，ランクとプロセッサの対応付けを行なう「ランクテーブル」と待機系プロセッサを示す「待機プロセッサリスト」を持つ．MPI アプリケーション起動時，MPI 内部で，正常に動作しているプロセッサをシステム構成テーブルより選択し，「ランクテーブル」とシステム構成テーブルの間でリンクを張る．起動時に使用しないプロセッサは「待機プロセッサリスト」に連結する（**図 14**）．

(2)　並列処理に使用しているプロセッサに故障が発生した場合，MPI/SP は「待機プロセッサリスト」から正常動作しているプロセッサを選択し「ランクテーブル」とシステム構成テーブルの間でリンクを確立する．故障プロセッサは「待機プロセッサリスト」の末尾に挿入する（**図 15**）．

縮退運転→回復運転，回復運転→正常運転への切替：

切替は以下のようにして行なう（**図 16**）．

(1)　システム構成テーブルを参照して，復旧したプロセッ

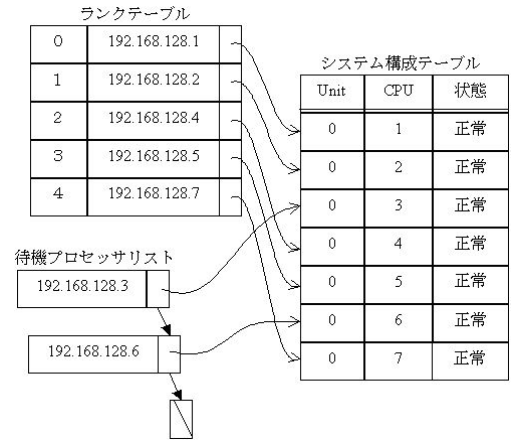

図 14 MPI/SP 内部でのプロセッサ管理

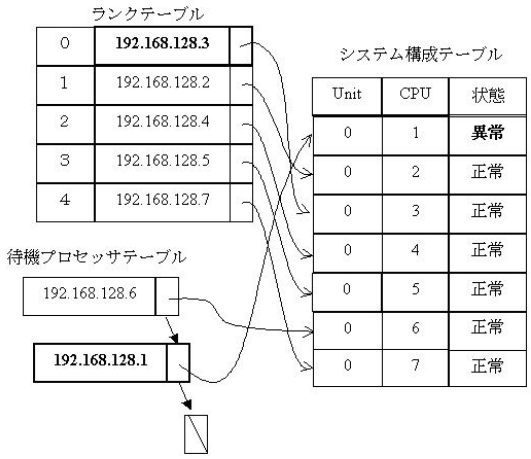

図 15 MPI/SP 内部でのプロセッサ交換による代替運転

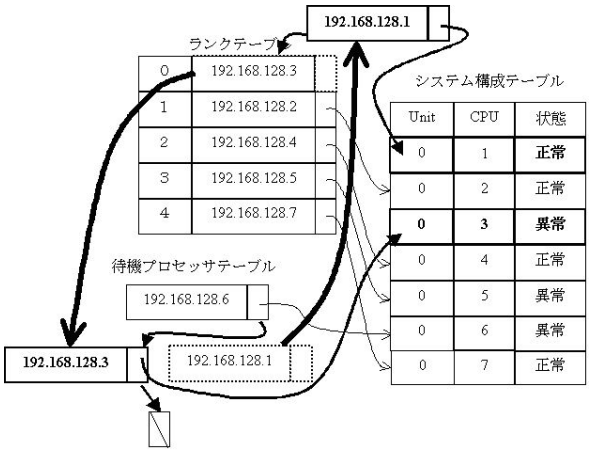

図 16 MPI/SP による性能回復運転

サを検出する．

(2) ランクテーブルとシステム構成テーブルを参照し，故障プロセッサが割り当てられているランクのうち最小の番号を検出する．

(3) (2) で発見した故障プロセッサをランクテーブルから除き，待機プロセッサリストに連結する．

(4) 待機プロセッサリストから復旧プロセッサを切り離し，ランクテーブルの該当エントリとリンクを張る．

3.3 MPI/SP の制約

本開発によって MPI インタフェース内部で動的なプロセッサ構成変更を可能にしたことに伴い，本実装の MPI インタフェースでは動的なグループ作成に対応できないという制約が発生した．

従来の MPI は，クラスタの形状や処理単位などにより，デフォルトのグループ：`MPI_COMM_WORLD` のほかに，運用中に新しく通信グループを作成してグループ通信を行なうことができる．しかし，本開発による MPI では，プロセッサ異常時には動的なプロセッサ構成変更を可能としており，構成変更時に故障したプロセッサからグループ情報などを引き継ぐことができないため，代替プロセッサは動的に作成されたグループに参入できない．

MPI/SP ではグループ通信関数が警告を返すことによりプロセッサの構成変更を知ることができるので，ターゲットシステムでは，警告を受けた場合にはアプリケーション上でグループを明示的に再構成することにより動的グループ作成に対処している．

動的に作成したグループへの代替プロセッサの自動再参入を実現する方法としては，定期的にスナップショットを取得し，プロセッサ切替時に代替系に通知する方法が考えられる．今後，代替プロセッサが故障プロセッサの処理環境を引き継ぐことができるよう改良を加えていく．

4. おわりに

本研究では，システム異常を検出し，異常を全プロセッサに通知可能な高信頼ミドルウェアを設計した．そして，MPI と高信頼ミドルウェアを連携させ，MPI 内部でプロセッサやネットワークの切り替えを実現することによりシステム構成動的変更を隠蔽し，24 時間 365 日連続運転可能なアプリケーション作成の容易化を実現した．

今後，本ミドルウェアを実システムに適用し，信頼性などに関する評価を行なう．

参 考 文 献

1) 村山　和宏，落合　真一，山口　義一：MPI/SP におけるクラスタ統合方式の設計，マルチメディア通信と分散処理ワークショップ論文集，pp85–90（2001）．

2) 落合 真一，村山 和宏，山口 義一：MPI/SP における異常処理機構の設計，第 63 回情報処理学会全国大会講演論文集（1），pp33–34（2001）

3) Gavin D.Holland，Dhiraj K.Pradhan：*A Software Implemented Fault-Tolerance Layer for Reliable Computing on Massively Parallel Computers and Distributed Computing Systems*，Department of cpmputer Science Texas A&M University（1995）．

4) MPI：*http://www.mpi-forum.org/*