B-028

超並列システムにおける ディペンダブルな処理環境についての評価実験
Evaluating the dependable system of mass parallel computing system

柳瀬 龍郎† 田村 信介 高津 麻季子††
Tatsurou Yanase Sinsuke Tamura Makiko Takatsu

1. 研究背景

グリッドやクラスタなどの大規模処理環境においては科学技術計算など長時間の実行が前提となる計算オリエンテッドなアプリケーションお実行が中心となる．むろん超高性能な計算サーバが自由に使用可能な環境を有するユーザーの場合，チェックポイント&リスタートなどの手段があり「別に問題ないが？」と考えられるであろう事は想像に難くない．しかしそうではなくて，コモディティな PC がローカルネットワークで結合された状態にあって、ユーザーが「なんとか並列処理などの手段を利用して，高性能でかつ，信頼性の高い，計算処理環境として使いたい」と考える状況が多々起きているのも事実である．しかし，構成要素の PC はある程度信頼性が期待できたとしても，「並列処理に参加」する全体の PC を１つのシステムとみれば信頼性が低下することはやむを得ない．そこで，理想的には，多数の PC がネットワークで結合された環境に合って，これを利用したいと考えた場合には；

1)長時間の実行でも，処理全体としては異常終了せず，正常に実行結果を得たい

2)通常の PC の利用者なら，だれでも簡単に利用できる

3)メッセージ通信などの情報交換手段は陽にアプリケーションコードに書かなくても良い

などの条件を満たす事が理想的と考えている．

上述のように，大規模な分散システムにおいてはアプリケーションの実行中，一部の構成要素の故障によるシステム全体への影響の波及により，結果的に処理が完了しない，ということになりかねない．このことについて既存の分散/並列処理の研究[1]等においては，言及されることはほとんどなかった．

本研究では，複数の PC をネットワーク管理 PC 群として用いることによって高い信頼性を持つ並列分散システムを構築し，実際に並列処理を行った場合についての評価実験を行ったので，結果について報告する．

2. 管理システムのアーキテクチャ

つぎに管理 PC 群による，高信頼性システムの維持方式について述べる．

2.1 システムの構成

複数の PC を管理プロセッサとして用い，これらを図１に示すように論理的に結合する．また分散・並列システムに参加する全体の PC はこれらの管理 PC 群に、それぞれ分担監視される．物理的な結合については，その方法がなんであろうとかまわないことは当然である．

アプリケーションは，システムに参加する全体の PC により分散・並列処理される．管理の仕事を行う PC は独立した別プロセスを起動することにより，アプリケーションの分担処理にも参加出来るので，信頼性の獲得のための冗長な PC の用意は必要ない．プロセス・スレッドの幾つかはアプリケーションの実行に直接参加しないことにはなる．

また，情報の交換が行われるのはすべて管理 PC 相互間あるいは，管理 PC と一般 PC 間においてのみ行われ，一般 PC 相互では一切行われない．

2.2 ジョブの分配

アプリケーションの実行はいずれかの PC に投入されることによって開始される．この処理の途中において並列処理が必要になると，幾つもの細分化され，並列実行可能な部分ジョブが生成されるが，開始された PC 以外でこのシステムに属する他の PC に，細分されたジョブが移行され並列実行される．またそれらの処理結果の値はもとの PC に返却される．

先に，陽にメッセージなどによる通信は行われないとしたが，情報の交換はすべてはいくつかの引数を持つ関数の評価依頼とその評価値の返却のみが行われるだけである．LISP や Java 等では欠かせないいわゆる"評価環境"（クロージャ）のコピーや移行は行われない．すべては引数のみによる評価を前提としている．ただし返値のアドレスに関しては管理される．

平行処理されるべきジョブの移行—関数の評価依頼—は，すべて管理 PC にまかされる．管理 PC は相互のネットワークを使っていずれかの管理 PC に評価を依頼する．依頼を受けた管理 PC は自分の管理下にあるいずれかの PC にジョブの処理を依頼する．その結果，得られた評価値も管理 PC のネットワークを経由して元の PC に返却される．

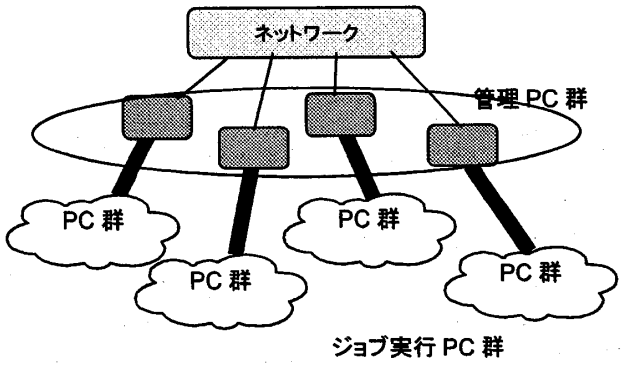

図１． 分散・並列論理ネットワーク

3. ネットワークの維持

管理 PC の仕事は，システムのネットワーク監視および維持と，評価依頼されたジョブ（関数）の分配と結果値の

†福井大学工学部情報・メディア工学専攻
††(株)ネスティ

元のPCへの返却である．

3.1 ネットワークの監視と維持

大規模なPCグリッドやクラスタシステムの場合，システム全体としての信頼性は個々の計算リソースの信頼性と比較して低下することは避けられない．そのような場合でもアプリケーションの実行システムとしての高信頼性を維持するには，システムの監視による異常の検出と，これに動的に対応し，アプリケーションを実行完了できるようにシステムを再構成する能力の維持が必要である．これらの役割を管理PCが担う．

管理PCは自分の管理するPCと常時(TCPソケット等を用いてた)通信路を保持し，この通信路が何らかの理由で異常を示せば，そのPCに依頼したジョブは全面的に破棄し、最初の依頼した時に用意した“ジョブの依頼控え”を用いて別のPCにジョブの処理を依頼する．

管理PC自身の異常は管理PCのネットワークによって他の管理PCによって検出される．この場合，異常が起こった管理PCの管理情報は，他の管理PCにコピーが置かれていることを利用して，故障PCが管理していたPC群の管理を引き継ぐ．管理されていたPC群は一時的に放置されることになるが，しばらく待てば代理の管理PCからのアクセスがあるので，しばらくはそのまま待つことにより，全体としてのシステムに再び参加することができる．このようにして故障した場合でも，これを取り除き，システムを動的に再構成することによって，アプリケーションの実行が継続され，最後まで実行が行われる可能性を格段に高める．また，取り除かれたPCが復旧する場合は新規参加としてシステムに再び参加する．

3.2 ジョブの分配と結果の返却

システムの動的再構成により，ジョブの処理とその依頼の関係が変化するので，最初にこれらの記録を用意し，再構成が発生したときに，これを更新し維持管理しなければならない．管理PCによるネットワークの監視と，再構成のためのこれらの情報の維持更新とそれを用いた再構成動作処理の仕事が高信頼化のためのオーバーヘッドとなる．

これらの操作と情報維持はアプリケーションとは全く独立した動作であり，従って並列処理を行う必要がないアプリケーションでは作動の必要もない．アプリケーションにおいて，並列処理に関する仕事の依頼が発生した時点で，すべてが準備されなければならない．従って並列処理開始の時点で最も大きなオーバーヘッドが予想されるが，これはアプリケーションの実行時間とのトレードオフの問題である．

3.3 PCの復旧と新規参加

故障や異常から復帰したPCや，新しく追加されたPCがシステムに参加する場合，管理PCとして参加するか，それ以外の一般PCとして参加するかは，最初に選択しなければならない．一般PCとして参加する場合，適当な管理PCに自分が一般PCであることを伝え，管理PCの参加リストに登録されることで，システムに参加したと見なされる．この時点以降は他の一般PCと同様な扱いを受ける．また，管理PCとして新規参加する場合，一般PCと異なりシステムの再構成が行われる．このとき，一般PCの管理の引き受けも行わなくてはならない．また，それに伴う管理情報の更新も行われる．

4. 評価実験

管理システムはアプリケーションライブラリとしてCで実装した．実験環境はLinuxOS，CPU 3GHz，1GBメモリを持つPCが64台，1GHzのetherネットで結合された均一型のクラスタである．実行アプリケーションは80個の独立/並列実行可能なジョブを次々と生成し，最後に全てのこれらのジョブの結果を収集し処理を施して標準出力にプリントアウトする単純なものとした．実験では参加PCの台数を16，32および64台と変化させ，単純にPCの故障を模倣したジョブの打ち切りを行った場合の正常にアプリケーションが終了する時間を計測した．表1に結果を示す．

表1 アプリケーションの処理時間（秒）

PC台数	故障なし	管理PC故障	処理PC故障
１６	２３９	—	２７８
３２	１５６	—	１５７
６４	１２５	１６２	１２５

表1は独立ジョブとして並列実行する関数の1つの実行時間を20秒に固定し，システムのPC台数を変化させたときの実行時間を示す．表内の“—”はアプリケーションが正常終了しなかった場合を示す．台数を64にした場合，管理PCの故障が発生してもデータの引き継ぎが行われ，アプリケーションの実行終了が確認されたが，その他の場合では管理PCの故障が発生した場合，アプリケーションの正常完了が確認できなかった．

5．まとめ

大規模な分散/並列環境において，高信頼性を有するシステムの提案を行い，そのミドルウエアとしての設計と実装，そして簡単な評価実験を行った．

システム維持のための情報を，管理PC相互間でバックアップを行い，故障に対する動的再構成を行う．この方法により計算環境要素の故障によるアプリケーションの中断の確率を抑制することができることを示した．

また，実験で特に印象深く感じたのは，管理PC自身の微妙なタイミングでの故障/異常についても見逃さず，確実に事象の生起を捉えることが出来るようなプログラムコーディングにおける工夫が必要であることも重要である．

参考文献

[1]Hai JIN, Xuanhua SHI, Weizhong QIANG and Dequing ZOU, DERIC:Dependable Grid Computing Framework, IEICE TRANS, INF. & SYST., VOL. E89-D, NO.2 (2006.02)

[2]松田大樹，Javaによるジョブの多重実行支援環境，福井大学大学院工学研究科情報メディア工学専攻博士前期課程平成17年度学位論文(2006)

[3]高津麻季子，高信頼性を有する並列処理システムの研究，福井大学大学院工学研究科情報・メディア工学専攻博士前期課程，平成１９年度学位論文(2008.02)