計算機アーキテクチャ 152－13
（２００３．３．１０）

PC クラスタ用ネットワーク RHiNET-2 上における動的負荷分散アルゴリズムの評価

北村　聡[†]　天野英晴[†]
渡邊幸之介[†]　大塚智宏[†]

RHiNET はオフィスの数フロアなど, ある程度の広さを持った空間内に配置された PC/WS を相互接続し, それらの余剰計算能力を利用して, 高い並列処理能力を得ることを目的としたネットワークである. このような環境では, 各 PC/WS の性能. 余剰計算能力が異なり, 並列処理全体の実行時間は最も性能の低い PC/WS に大きく左右される. そのため, 高い並列処理性能を得るためには動的負荷分散機構が不可欠である. 本稿では, 動的負荷分散アルゴリズムを提案・実アプリケーションに実装し, 評価を行った. 評価の結果, システムに負荷がかかっている場合に動的負荷分散を行うことによって, 行わない場合よりも, 最大 20% 程度高速に並列処理を完了できることが示された.

The evaluation of dynamic load balancing algorithm on RHiNET-2 network used at PC cluster

Akira Kitamura,[†] Amano Hideharu,[†] Konosuke Watanabe[†]
and Tomohiro Otsuka[†]

RHiNET is a network for high speed parallel processing environment by connecting PCs distributed on one or more floors or a building. In such a network, the computing power and load of PCs front-end job is different each other, thus, it is a heterogeneous environment, in which parallel execution time is dominant by the PC with the least performance. In order to keep performance, dynamic load balancing algorithms are proposed and implemented in a prototype RHiNET-2 with 64 nodes. As results of evaluation with a real application, a parallel process with dynamic load balancing achieves 20% performance improvement.

1. はじめに

RHiNET[1] はオフィスの数フロア, ビル内などある程度の広さを持った空間内に分散配置された PC/WS を相互接続し, その余剰計算能力を利用して分散並列処理を高速に行うことを目的として, 研究・開発されたネットワークである.

このような空間には PC/WS が数十〜数百台規模で設置されており, また, 一般に, その全計算能力を使用していることは少なく, 大きな余剰計算能力が潜在している. RHiNET は低レイテンシなユーザレベルゼロコピー転送によりこれらの余剰計算能力を利用して並列分散処理を行なう環境をサポートする.

RHiNET はクラスタ用システムソフトウェア SCore[2] が搭載されているが, 実際に余剰計算能力を効率的に使用するためには, 一般のクラスタとは異なった負荷分散機構が必要となる. 一般にオフィス等に設置される PC/WS を接続すると, それぞれのノードのハードウェア構成は異なるため, ヘテロジーニアスなシステムとなる. また, 各 PC/WS は端末としてそれぞれのユーザに使用されているため, 負荷が異なり, 余剰計算能力に差が生じる. そのため, 通常の PC クラスタで行なうように, 均等に処理を分散させると, 最も処理能力の低い PC/WS の処理が終了するまで他の PC/WS は待機状態となり, 余剰計算能力を有効利用できない. そこで, 処理の途中で動的に処理を移動させ, 負荷の均衡を図る動的負荷分散機構が必要となる.

本稿では, まず RHiNET について述べ, 続いて, 動的負荷分散の関連研究について触れる. その後, 実装した動的負荷分散アルゴリズムについて説明し, Ethernet と RHiNET-2 上での評価と結果を示す. 最後に, まとめと今後の課題について述べる.

2. RHiNET

RHiNET は SAN(System Area Network) に匹敵するバンド幅, 低レイテンシな通信性能と, LAN と同程度に自由度の高いトポロジやリンク長を兼ね備えたネットワークである. トポロジフリー, かつデッドロックフリーなルーティングを提供する専用のネットワークスイッチ, PC/WS の PCI バスに装着する専用のネットーワークインタフェース, 及び, それらを結合する数 Gbps クラスの転送容量を持つ光インタコネクションより構成される.

本稿で用いた RHiNET-2 のネットワークインタフェースは RHiNET-2/NI[3] と呼ばれ, ノード間通信を高速に処理するために, ユーザレベルゼロコピー通信を行う. そのた

† 慶應義塾大学 理工学部
Faculty of Science and Technology, Keio University

めに必要なアドレス変換等の処理は, RHiNET-2/NI 上に搭載されているネットワークインタフェースコントローラ Martini[4] で行う.

RHiNET-2 のネットワークスイッチを RHiNET-2/SW[5] と呼び, 8 組の光リンクを接続可能である.

図 1, 2 に RHiNET-2/NI と RHiNET-2/SW の外観を示す.

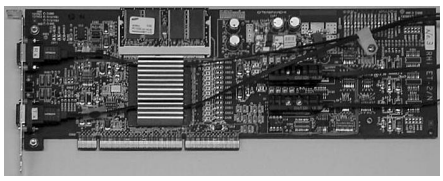

図 1 RHiNET-2/NI

図 2 RHiNET-2/SW

これらを各方向 8Gbps の帯域を持つ光インタコネクションで接続している.

RHiNET のソフトウェアレイヤは, 図 3 に示すように, まずハードウェアである RHiNET/NI 上のコアプロセッサで実行されるファームウェアが最下層に位置する.

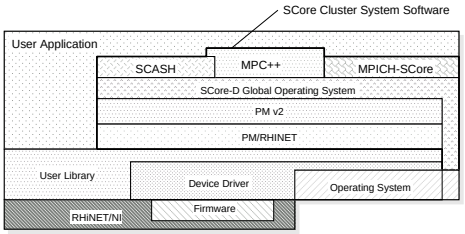

図 3 RHiNET のソフトウェアレイヤ

それより上位はホスト PC 上で実行されるソフトウェアレイヤであり, カーネルレベルで実行されるデバイスドライバと, ユーザレベルで実行されるユーザライブラリが位置している. 特に, クラスタ計算機向け高速通信ライブラリ PMv2 が移植されたことによって, SCore を利用でき, その上で MPI (MPICH-SCore) や, 分散共有メモリを提供する SCASH などが使用可能である[6]. 本稿の評価では SCore 上で MPICH-SCore を用いている.

3. 関 連 研 究

PC/WS クラスタを対象とした動的負荷分散アルゴリズムは数多く提案されており, 大きく分けると, ロードバランサと呼ばれる, 負荷分散の処理を一括して行うノードを設置する集中制御型と, ロードバランサを用いずに, 全体通信を用いて負荷分散を行う分散制御型に分けられる. Siegell のアルゴリズム[7] では, PC/WS クラスタを対象に, ロードバランサを用いて負荷分散を行う手法を採っており, 動的負荷分散のための通信に要するオーバヘッドを計算とオーバラップさせることにより隠蔽する手法を提案している.

一方, Chao-Yang らはロードバランサを用いないアルゴリズムを複数提案している. これらのアルゴリズムは, MPI などのメッセージパッシングライブラリを用いることを前提としており, 負荷分散の処理を全ノードで同期を取って行う手法と, 非同期に負荷分散の処理を行う手法である[8].

一般に, ロードバランサを用いると, 各ノードの負荷情報等の管理が容易になるが, システムを構成するノードが増加するに従って, ロードバランサ周辺のネットワークが混雑し, ボトルネックとなる. 一方, ロードバランサを用いない場合, ネットワークのトラフィックは均一化されるが, 通信量の増大を招き, また, 負荷分散の処理が複雑になる.

これら以外の手法として, 並列化コンパイラを用いて動的負荷分散を実現する手法等が提案されている[9][10].

4. 実　　装

4.1 負荷分散アルゴリズム

本稿で実装した動的負荷分散アルゴリズムは, SPMD(Single Program Multiple Data) 型, かつ並列化可能なループを持つアプリケーションを対象としている. 従って, 各ノードの処理内容は基本的には同一であり, 担当するデータ量を増減することにより, 負荷を調節する. 数ループ毎に負荷分散フェーズを設け, 負荷分散処理を行う.

実装したアルゴリズムは次の通りである.

- CLB(Central Load Balancing) : 集中制御型
- DLB(Distributed Load Balancing) : 分散制御型
- GLB(Group only Load Balancing) : グループ内限定型
- IGCLB(Inter Group CLB) : グループ間集中制御型
- IGDLB(Inter Group DLB) : グループ間分散制御型

4.1.1 CLB アルゴリズム

CLB アルゴリズムは, ロードバランサを用いた負荷分散方式である. CLB アルゴリズムの流れを図 4 に示す.

CLB アルゴリズムは, ロードバランサとなるノードを 1 つ定める. ロードバランサは他のノードから負荷情報を収集し, その負荷情報を元に, 各ノードの処理量を求める. その後, 各ノードに処理量を通知し, その処理量に従ったデータの送受信を行う.

このアルゴリズムは, ロードバランサが一括して負荷情報を持つことになるため, 負荷分散の制御が簡潔になるという利点があるが, ノード数が増加するに従って, ロードバランサ周辺のネットワークが混雑し, ボトルネックとなる. この欠点を改善する目的で実装を行ったのが, 次の DLB アルゴリズムである.

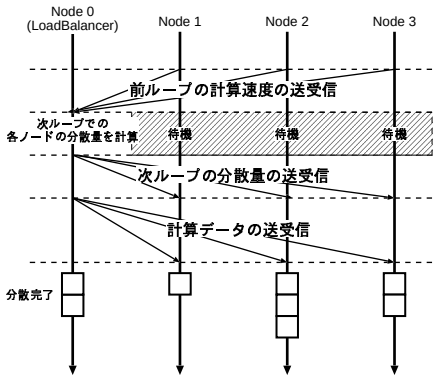

図 4 CLB アルゴリズム

4.1.2 DLB アルゴリズム

DLB アルゴリズムは, ロードバランサを用いずに, 全体通信を行うことによって, 負荷分散を実現する. DLB アルゴリズムの流れを図 5 に示す.

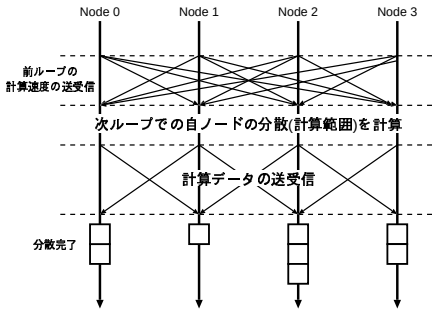

図 5 DLB アルゴリズム

このアルゴリズムでは, ロードバランサを用いないため, 負荷分散の制御が CLB アルゴリズムに比べて複雑である. また, ネットワークのトラフィックが均一化されるが, 通信量が増大する, 極端に処理の分散が偏った場合に, ある特定のノードに通信が集中するため, トラフィックが偏る可能性がある, といった欠点が存在する. そこで, トラフィックの偏りを完全に排除するために, 負荷分散の通信相手を特定したアルゴリズムが次の GLB アルゴリズムである.

4.1.3 GLB アルゴリズム

先の CLB, DLB の両アルゴリズムは, 単純にロードバランサの有無に焦点を当てたアルゴリズムであるが, GLB アルゴリズムでは, 通信相手に注目する. GLB アルゴリズムは, 数ノードで 1 グループを形成し, 負荷分散はグループ内でのみ行うアルゴリズムである. 図 6 に GLB アルゴリズムの流れを示す.

負荷分散はグループ内でのみ行われるため, ノード数が増加しても, 負荷分散に要する通信のコストは変化しない. しかし, グループを構成する全ノードの処理速度が低下すると, そのグループの処理速度がボトルネックとなる. これを防ぐためには, グループを構成するノード数を増やし, グループ内のノード全てが, 同時に処理速度の低下を起こ

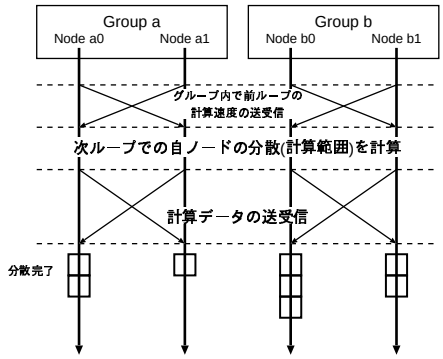

図 6 GLB アルゴリズム

す可能性を低くするか, グループ間で負荷分散を行うかのどちらかである. 後者の手法を採ったのが, IGCLB, IGDLB の両アルゴリズムである.

4.1.4 IGCLB, IGDLB アルゴリズム

図 7 に IGCLB アルゴリズム, 図 8 に IGDLB アルゴリズムの流れを示す.

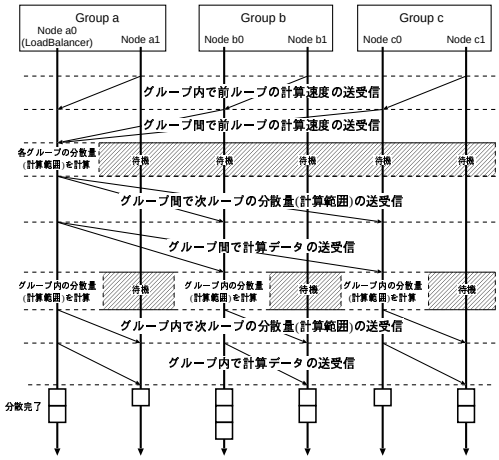

図 7 IGCLB アルゴリズム

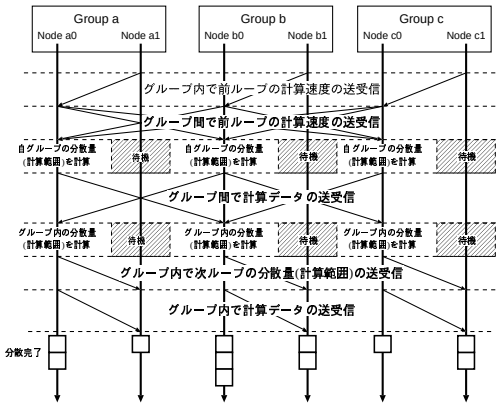

図 8 IGDLB アルゴリズム

図 7, 8 に示されるように, グループ内で 通信を行うと同時に, グループを代表するノード同士でグループ間通信

を行うことによって, 負荷分散を行う. ただし, IGCLB アルゴリズムはロードバランサを用い, IGDLB アルゴリズムはロードバランサを用いない. つまり, IGCLB アルゴリズムは GLB アルゴリズムと CLB アルゴリズムを階層的に組み合わせたもの, IGDLB アルゴリズムは GLB アルゴリズムと DLB アルゴリズムを階層的に組み合わせたものといえる.

4.2 負荷情報の評価

負荷分散を行う際に用いる負荷情報として, 式 (1) で求められる値を用いる.

$$length_i^{NEW} = INT\left(\frac{length_i^{OLD}/time_i}{\sum_j length_j^{OLD}/time_j} \times length_{all}\right) \quad (1)$$

式 (1) は, 熊谷らの手法[11] で用いられた式である. $length_i^{NEW}$ はノード i の新しい分散量, $length_i^{OLD}$ は旧分散量であり, $time_i$ は旧分散量の実行時間である.

つまり, 旧分散量の実行速度比から, 新しい分散量を求める式である.

5. 評　　価

5.1 評価環境

本来,RHiNET は机上に分散された様々な PC を接続して構成するが, このような構成は場所を必要とするため, 評価用には 64 台の実験用クラスタ (図 9) を利用した.

図 9 RHiNET-2 64 ノードクラスタ

各ノードの構成を以下に示す.

- Supermicro SuperServer6010H
- Processor : Pentium III 933MHz × 2
- Memory : PC133 SDRAM 1GBytes
- PCI : 64bit/66MHz
- OS : RedHat 7.2 (kernel 2.4.18)
- Ethernet : Intel EEPRO100 100Base-TX Fast Ethernet (Switching Hub で接続)
- SCore : 5.0.1

PCI バスには RHiNET-2/NI が装着されており, RHiNET-2/SW に接続されている. 各 RHiNET-2/SW に計算ノードを 4 台ずつ接続し, RHiNET-2/SW 同士は 4×4 の 2 次元トーラスのトポロジで接続されている. なお, 各ノードにはプロセッサが 2 基搭載されているが, アプリケーションの実行には 1 基しか用いていない.

5.2 評価条件

5.2.1 アプリケーション

アプリケーションには 8192 次元の連立一次方程式を SOR(Successive Over Relaxation) 法を用いて解くものを選択した. SOR 法は反復法の一種で, 適当に定めた近似解を徐々に真の解に収束させる手法である. 解が収束, または発散するまで同一の処理を繰り返すため, その処理部をループを回すことによって実現する. このプログラムを MPICH-SCore を用いて実装した.

本来 SOR 法は並列化が難しいが, 8192 個の一次方程式をノード数分に分割し, その分割した方程式群の中で SOR 法を用いることによって, 並列化を行った. そのため, 近似解の更新のために, 数ループごとに定期的に全体通信を, また, 解が収束したかどうかを調べるための全体通信をループ毎に行う.

本稿の実装では, SOR 法を 8 ノード以上で行うと, 解の収束までに 200 ループ以上要することが分かっているため, そこから, 負荷分散の頻度を表 1 のように設定した.

表 1 負荷分散の頻度

アルゴリズム	負荷分散の頻度
CLB	50 ループ毎に 1 回
DLB	
GLB	
IGCLB	50 ループ毎に 1 回 (ただし,GLB と IGCLB, IGDLB を交互に行う)
IGDLB	

表 1 の値は, 負荷分散の頻度を様々に変更して実行した結果, 実行時間から最も適当であると判断した値である.

なお, グループを形成するアルゴリズムでは, 2 ノードで 1 グループを形成するものとした.

IGCLB, IGDLB の両アルゴリズムは, 負荷分散のコストが他のアルゴリズムと比べて大きいため, GLB アルゴリズムと交互に用いることとした.

また, 本稿では方程式の係数等の初期値は NFS を用い共有されているものとしている. そのため, 図 4〜図 8 で示される, 計算データの送受信は行われない.

5.3 評価項目

評価は RHiNET-2 及び, Ethernet を用いて, 各アルゴリズムを 8, 16, 32, 64 ノードで行う. また, CLB, DLB の両アルゴリズム比較用として, グループを形成せず, かつ全く負荷分散を行わない noLB (no Load Balancing) と, GLB, IGCLB, IGDLB の 3 アルゴリズム比較用として, グループを形成し, かつ全く負荷分散を行わない GnoLB (Group noLB) の実行時間も測定した.

今回はノード自体に負荷をかけ, ネットワークには全く負荷をかけていない. 負荷をかけるパターンは 8 ノードを単位として, 全く負荷をかけない noload, 8 ノード中 1 ノードに負荷をかける oneload, oneload で負荷をかけたノードと同グループのノードにも負荷をかける twoload の 3 パターンである.

また, 負荷には Linux の yes コマンドを用いている.

5.4 評価結果

5.4.1 Ethernet

図 10〜図 12 に Ethernet 上での結果を, 図 13〜図 15 に RHiNET-2 上での結果を示す.

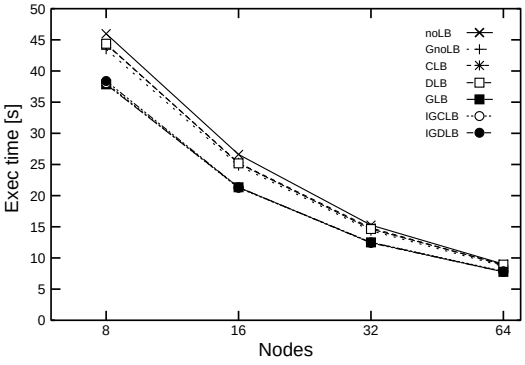

図 10 Ethernet(noload) での実行結果

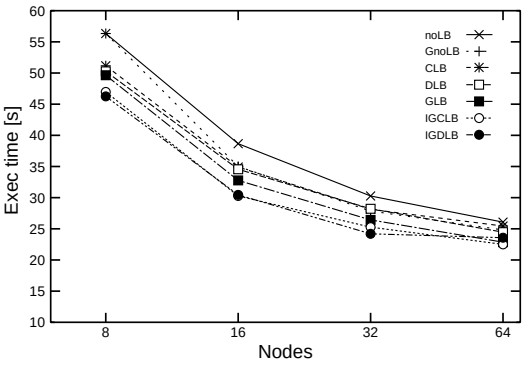

図 11 Ethernet(oneload) での実行結果

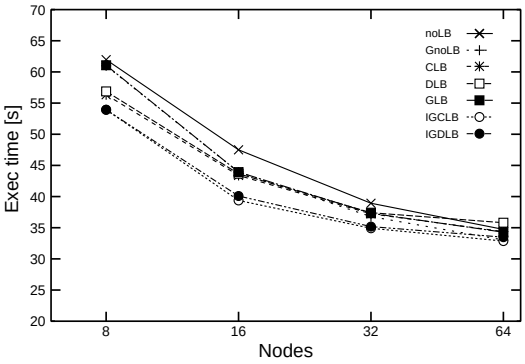

図 12 Ethernet(twoload) での実行結果

これらの図を見ると, Ethernet と RHiNET-2 に大きな差は見られない. これは, 負荷をかけていない状態 (noload) では, 計算に要する時間が, 負荷をかけた状態 (oneload, twoload) では, 全体通信に伴う待ち時間が全実行時間に大きな影響を与えるためである.

例として, CLB(noload) の実行時間の比較を行う. 表 2, 3 に CLB(noload) のノード 0 (ロードバランサ) の実行時間の内訳を示す.

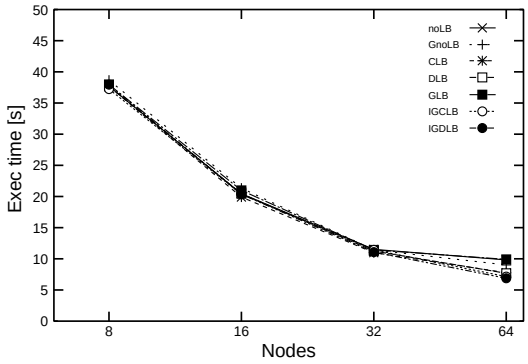

図 13 RHiNET-2(noload) での実行結果

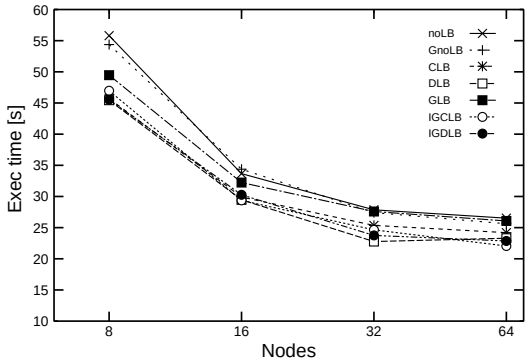

図 14 RHiNET-2(oneload) での実行結果

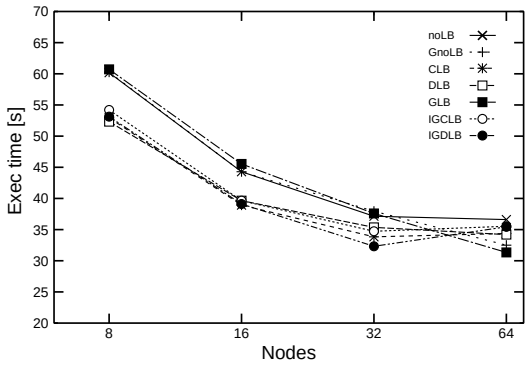

図 15 RHiNET-2(twoload) での実行結果

表 2 実行時間の内訳 (Ethernet, noload)

ノード数	総実行時間	演算時間	通信時間	その他
8	44.33	43.41	0.91	0.01
16	25.30	24.20	1.10	0.00
32	14.79	13.25	1.53	0.01
64	8.83	6.78	2.03	0.02

単位 : [秒]

8 ノードで RHiNET-2 は Ethernet の 4 倍, 16 ノードで 4.2 倍, 32 ノードで 3.2 倍の通信速度を達しているにもかかわらず, RHiNET-2 での全実行時間に占める通信時間の割合は, 8 ノードで 0.6%, 16 ノードで 1.3%, 32 ノードで

表 3 実行時間の内訳 (RHiNET-2, noload)

ノード数	総実行時間	演算時間	通信時間	その他
8	37.64	37.41	0.23	0.00
16	19.92	19.65	0.26	0.01
32	11.01	10.52	0.48	0.01
64	7.80	5.62	2.17	0.01

単位 : [秒]

4.3% であり, 実行時間に与える影響は小さい.

64 ノードでは Ethernet より通信時間に要する時間が長い. これは, RHiNET-2 が 64 ノードでは安定動作せず, プログラムに 1 箇所に通信が集中する箇所がある場合, 意図的に通信間隔を空けるための処理を挿入する必要があるためである.

また, 負荷をかけた場合の例として, CLB(oneload) のノード 0 の実行時間の内訳を, 表 4, 5 に示す. ただし, “通信時間”, “待ち時間” は “その他” に含めるものとする. これは, 全体通信時に, 負荷をかけたノードを待つ時間が発生するため, 正確に通信時間と待ち時間を切り分けることが難しいためである.

表 4 実行時間の内訳 (Ethernet, oneload)

ノード数	総実行時間	演算時間	その他
8	51.33	45.11	6.22
16	34.90	24.88	10.02
32	28.14	13.53	14.61
64	25.40	7.03	18.37

単位 : [秒]

表 5 実行時間の内訳 (RHiNET-2, oneload)

ノード数	総実行時間	演算時間	その他
8	45.68	38.57	7.11
16	29.93	20.48	9.45
32	25.32	11.18	14.14
64	24.22	5.72	18.50

単位 : [秒]

Ethernet 上での全実行時間に占める待ち時間の割合は, 8 ノードで 12.1%, 16 ノードで 28.7%, 32 ノードで 51.9%, 64 ノードで 72.3% となる. 同様に, RHiNET-2 では, 8 ノードで 15.6%, 16 ノードで 31.6%, 32 ノードで 55.8%, 64 ノードでは 76.4% に達する. これらのことから, アプリケーションによっては, 通信性能の向上が全実行時間の大幅な削減につながるとは限らないことが分かる.

個々のアルゴリズムについて触れていくと, oneload までは, どのアルゴリズムも, 負荷分散を行わない場合より実行時間が短い. twoload では GLB アルゴリズムのみ, GnoLB と同等, または若干の速度低下が見られた. これは, グループを形成するノード全てに負荷がかかっており, 負荷分散の処理を行っても処理の分散を行うことができず, 負荷分散の処理を行うだけ無駄となるためである. この点が IGCLB, IGDLB の両アルゴリズムでは改善されており, 実行時間の削減に成功している.

RHiNET-2 では 64 ノードで若干の速度低下を起こしているアルゴリズムが存在するが, これは, 先に述べたように, 安定動作のための処理が影響していると思われる.

6. まとめと課題

本稿では, 5 種類の動的負荷アルゴリズムを SOR 法のプログラムに実装し, Ethernet, 及び, RHiNET-2 上で評価を行った. 評価の結果, 負荷分散を行うことによって, システムの一部に負荷がかかった状態でも, 実行速度の低下を抑えることが可能であることが示された.

今後の課題として, 待ち時間を考慮した負荷分散方式の模索, RHiNET に適したアルゴリズムの実装, SOR 法以外のアプリケーションでの評価が挙げられる. また, 本稿では 2 ノードで 1 グループとしたが, グループを構成するノード数を変化させ, 最適なグループ構成を求めることも必要である.

参 考 文 献

1) T. Kudoh, S. Nishimura, J. Yamamoto, H.Nishi, O. Tatebe, and H. Amano. RHiNET: A network for high performance parallel processing using locally distributed computers. IWIA99, pp.69-73, 1999.
2) Y. Ishikawa, H. Tezuka, A. Hori, S.Sumimoto, T. Takahashi, F. O'Carroll, and H. Harada. RWC PC Cluster II and SCore Cluster System Software – High Performance Linux Cluster. 5th Annual Linux Expo, pp.55-62, 1999.
3) 大塚 智宏, 渡邊 幸之介, 土屋 潤一郎, 原田 浩, 山本 淳二, 西 宏章, 工藤 知宏, 天野 英晴. RHiNET ネットワークインタフェースの性能評価. 電子情報通信学会技術研究報告, CPSY2002-40-50, pp.23-28, 2002.
4) 山本 淳二, 渡邊 幸之介, 土屋 潤一郎, 原田 浩, 今城 英樹, 寺川 博昭, 西 宏章, 田邊 昇, 上嶋 利明, 工藤知宏, 天野 英晴. 高性能計算をサポートするネットワークインタフェース用コントローラチップ Martini. 情報処理学会 HPS-5-018, 2002.
5) 西 宏章, 多昌 廣治, 西村 信治, 山本 淳二, 工藤 知宏, 天野 英晴. LASN 用 8Gbps/port 8×8 One-chip スイッチ:RHiNET-2/SW. JSPP2000, pp.173–180, 2000.
6) 原田 浩, 山本 淳二, 土屋 潤一郎, 渡邊 幸之介, 天野 英晴, 工藤 知宏, 石川 裕. RHiNET の高速通信ライブラリ PMv2 による評価. 情報処理学会研究報告 2002-ARC-147(HOKKE-2002), pp.145-150, 2002.
7) Bruce S.Siegell. Automatic Generation of Parallel Programs with Dynamic Load Balcansing for a Network of Workstations. School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213 Submitted in partial fulfillment of the requirements for the degree of Doctor of, 1995.
8) Chao-Yang Gau and Mark A.Stadtherr. Parallel Interval-Newton Using Message Passing : Dyanamic Load Balcancing Strategies. Proceedings of the 2001 ACM/IEEE conference on Supercomputing, 2001.
9) 後藤 慎也, 窪田 昌史, 田中 利彦, 五島 正裕, 森 眞一郎, 中島 浩, 富田 眞治. 並列化コンパイラ TINPER による非均質計算環境向けコード生成手法. 並列処理シンポジウム JSPP'97, JSPP'97 論文集, pp.205-212, 1997.
10) 田中 慎司, 後藤 慎也, 窪田 昌史, 五島 正裕, 森 眞一郎, 富田 眞治. 非均質環境向けコンパイラ heter-TINPAR -動的負荷分散方式の改良と評価-. 情報処理学会研究報告 98-HPC-70-20(HOKKE'98), pp.115-120, 2000.
11) 熊谷 泰幸, 木下 浩三, 岸 達也, 佐々木 隆, 伊藤 聡. 自動負荷分散とその評価. 並列処理シンポジウム JSPP2001, JSPP2001 論文集, pp.75-76, 2001.