ネットワーク環境における自律的な負荷分散システム

1P－8

秋岡 明香　根山 亮　首藤 一幸　村岡 洋一

早稲田大学理工学部情報学科

1　はじめに

近年、ネットワークで接続された計算機資源を有効に活用することを目的とした、負荷分散に関する研究が盛んに行なわれているが、一般には Server-Client 型のシステムを用いることが多い。すなわち、Client の負荷情報を Server が一元管理し、Client は計算負荷 (タスク) を分散させる際、ある時点での最も負荷の小さいノードを見つけるために、常に特定の Server と情報をやりとりする。そのため、Server-Client 型のシステムには、以下のような問題点があった。

- Server の故障がシステム全体の故障に繋がる。
- Server が Client の負荷情報を更新する際にネットワークに負荷がかかる。
- Client と Server がネットワーク的に遠い場合、Client 側が負荷情報の取得に長い時間がかかり、効果的な負荷分散処理が行なえないことがある。

そのため、タスクの処理にかかる時間や移送にかかる時間によっては、時間をかけて最も負荷の小さいノードを探すよりは、短時間で比較的負荷の小さいノードを探してタスクを移送させる方が有効な場合も多い。

このように負荷が最小であるノードを探す必要がない場合、Server-Client 型のシステムのように、全てのノードの負荷情報を一元管理する必要はなく、短時間で比較的負荷の小さいノードを発見する手法が求められる。そこで本論文では、各ノードに自律的に負荷分散を行なわせる手法を提案する。

2　提案するアルゴリズム

各ノードが自律的に負荷分散を行なう場合、効率良く負荷の小さいノードを探し出す手段を持っておかなくては、効果的な負荷分散は行なえない。そこで、自律分散的にノード群を管理するアルゴリズム [1] を応用し、以下のようなトポロジを提案する。

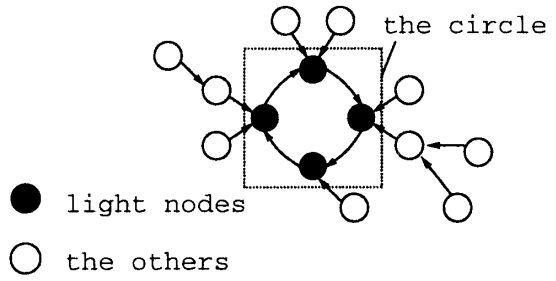

図 1: ノード管理のためのトポロジ構成

すなわち、図 1 に示すように、負荷の小さいノードで円を作り、その他のノードは負荷の小さいノードに向かって (参照先として) 枝として付く。枝の長さは規定しない。各ノードは、自分の負荷が大きくなれば自分にマークを付けて円の外へ出るように、小さくなればマークを外して円の中へ入るように、自律的に移動する。ただし、実際にノードを円の外に出す作業は、タスク移送時に行なう。

なお、負荷の大小については、2 段階の threshold (以下では upper threshold / lower threshold とする) を設定し、各ノードのキューの長さが upper threshold を上回った場合に負荷が大きい、lower threshold を下回った場合に負荷が小さいとして、タスクの移送やトポロジの変更を行なう。タスク移送のタイミングは、キューの長さが upper threshold を超えている間はタスク移送を試みることとする。移送先ノードは、自分から辿れる円の中にあるノードの中で、最も近いノードとする (以下、これを「親」と呼ぶ)。ここで、このように 2 段階の threshold を設けるのは、円外の負荷が大きいノードからタスクを 1 つ移送しただけで負荷が小さくなったと判定して円に戻すといった、頻繁なトポロジ変更を抑制するためである。

このトポロジを変更する方法は以下の通りである。

- ノードの負荷が大きくなった場合

  1. 自分にマークを付ける

Self Load Balancing System on Distributed Environment
Sayaka AKIOKA, Ryo NEYAMA, Kazuyuki SHUDO, Youichi MURAOKA
Dept. of Information and Computer Science, School of Science and Engineering, Waseda University

- ノードが実際に円の外に出る場合
  1. タスク移送時に、移送先ノードを親とする枝に付く。
  2. 自分から親までの間にあるノードに、同じ親に枝として付くように指示する。
  3. 円の中にいた際に自分を参照していたノードからの問い合わせを受けた場合には、自分の親へ参照先を変更するように指示する。
- ノードの負荷が小さくなった場合
  1. 自分のマークを外す。既に自分が実際に円の外に出ている場合には、以下の処理を行なう。
     (a) 自分の参照先を、自分の親の参照先に変更する。
     (b) 自分の親の参照先を、自分に変更するように指示する。

このように、枝の長さを可変にし、ノードの負荷が大きくなってもすぐには円の外に出さないようにしたことで、トポロジ変更に伴う通信量を削減することができる。また、タスク移送時には、自分のみでなく自分と親の間にあるノードもトポロジ変更を行なうことで、平均の枝の長さを短くでき、その結果負荷の小さいノードを探索するのにかかる時間を短縮できる。

しかし、これだけでは平均の枝の長さを確実に短くできず、枝が非常に長く延びてしまう場合も考えられる。そこで、親を探してトポロジを辿る場合、1 回に辿れるノードの数に上限を設け、親が見つからない場合でも、探索ノード数が上限に達したらそこで探索を一旦中断して、タスクの移送は行なわないことし、以下の処理を行なうことで枝を短くする (図 2)。

1. 最後に辿りついたノードの枝として付く。
2. 自分から最後に辿りついたノードまでの経路の上にある全てのノードに、最後に辿りついたノードの枝として付くように指示する。
3. トポロジ変更後も負荷が大きくタスク移送が必要であれば、再度親探索を実行する。

ここで、[2] により、問い合わせ回数を増してもタスクの平均レスポンスタイムが向上しないことが示されているので、探索するノード数に制限をつけることは、効果的に負荷分散を行なうために有効である。

また、時間の経過と共にノード状態は変化し、探索の必要性や探索結果の正確性が損われる可能性があるため、探索に時間がかからないようにするためにも、問い合わせ回数に上限を設けて打ち切ることは有効である。

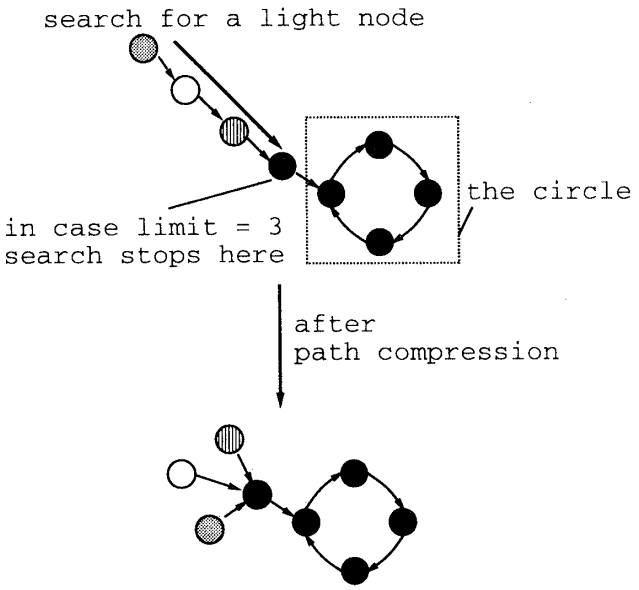

図 2: 枝を短くするための処理

3 性能評価

以上のアルゴリズムを実装し、シミュレーションによりタスクの平均応答時間を測定した結果を従来手法 [1, 2, 3] と比較したところ、良好な結果が得られた。

4 まとめ

本論文では、分散環境において、各ノードが自律的な処理を行なうことにより、負荷を分散させる手法を提案した。今後は、広域分散環境での利用のため、更なる通信量の削減を図り、情報の遅延を隠蔽する手法を導入する予定である。

参考文献

[1] "Maintaining a Dynamic Set of Processors in a Distributed System", S. Fujita and M. Yamashita, Distributed Algorithms, 10th International Workshop, WDAG'96, Lecuture Notes in Computer Science, Vol. 1151, 1996, pp.220-233.

[2] "Adaptive Load Sharing in Homogeneous Distributed Systems", Derek L. Eager, Edward D. Lazowska, John Zahorjan, IEEE Trans. on Software Eng., vol.SE-12, no 5,May 1986, pp.662-675.

[3] "Load Distributing for Locally Distributed Systems", Niranjan G. Shivaratri, Phillip Krueger, Nukesh Singhal, Ohio State University, Computer, vol.25, no.12, Dec. 1992, pp.33-44.