L-004

適応型階層構造を形成するピアツーピア Grid システム
Peer-to-Peer-based Grid with Adaptive Hierarchical Structure

佐藤　琢也† 吉田　紀彦†
Takuya Sato Norihiko Yoshida

１．はじめに

本論文は、Grid システムとピアツーピア（以降 P2P）とを融合したシステムを提案するものである。これにより、個人レベルでのインターネットを介した Grid システムの利用、またこのような Grid システムを組織することの実現を目指す。

提案するシステムは、Computing Grid の分野において Grid を用いる際に、動的にワーカとマネージャという関係が階層的に P2P ネットワーク上に構築されるシステムである。このシステムの特徴は、P2P のように身分に上下関係が無く、またネットワークのトポロジが動的に変化するシステムにおいて、どのノードも仕事を他のノードに依頼でき、仕事の依頼を受けたノード(ワーカ)も、自分の判断で子ワーカを従え、自律的に負荷分散を行なうことができ、それが階層構造を成すという点にある。

つまり、並列計算を行なう際に、問題を解いている最中に動的に負荷が変化し、負荷に偏りができてしまうものが存在するが、このような種類の問題に対しても、問題を配られた側が動的に再度負荷分散を行なうことにより、うまく資源の管理を行なうことができ、仕事の負荷に適応することができるという特徴を持つ。

従来の Grid システムでは負荷分散の管理や空いている資源の割り当てを行なう機能と、実際に計算を行なう部分が物理的に分離されており、多数のワーカマシンと少数のマネージャマシンという構成の固定的なシステムであったが、P2P という環境で Grid を構築するというシステムの特徴から、ワーカとマネージャの機能を統合することで、機能の差異の無いノード群によって形成される、適応型の階層構造を形成する Gird システムを実現した。

すなわち、Grid に参加している全てのノードが仕事の依頼を Grid システムに対して行なうことができ、自分がルートマネージャとして計算を開始し、その際に、Grid の資源からワーカを得て、その１階層分のみを管理する。また、そのワーカ群の各ノードも Grid の資源から新たにワーカを配下に付けて、自律的に負荷分散を行なうことができる。

ここで、このシステムがごく普通に家庭にある個人のマシンから構成される Grid であることから、当然各マシン間の性能が不均衡であるという問題が生ずる。このような問題にも、よりパワーの劣るマシンでは、積極的に子ワーカを作るようにすることでシステムのボトルネックにならないようにすることが可能であると考える。

２．システムの仕様

ワーカとマネージャ機能が統合されることから以下の機能を並列に協調動作するようにシステムを構築する。これら機能により、マネージャとワーカ両方の機能を併せ持つノードを実現することができる。

(1) 実際に依頼された仕事を行なう機能
汎用的な処理ができる仕組みを施し、万能な処理を行なう。すなわち、ワーカでは計算資源を貸し出すだけで、どのような仕事を行なうかを考慮する必要がない。

(2) Grid の他のワーカを動的に発見する機能
仕事でビジーの際に、空いているワーカを発見してくる。

(3) 新たな子ワーカに動的に仕事を依頼する機能
設定ポリシーに従い、どのような契機で新しい子ワーカを作るかを決定する。

(4) 子ワーカから結果を受け取る機能
仕事が終わった自分が管理している子ワーカからの、結果を受け取る役割を果たす。また、その受け取った結果を正しく処理する為の準備を行なう。

(5) 自分が担当した仕事のセグメントが全て正しく終了したか監視する機能
フォールトトレランス的要素を含み、きちんと正しく担当した仕事を終了させる必要がある。

３．システムの実装

本研究では、Grid システムを Globus Toolkit[1]の Web サービスフレームを用いて作成する。Web サービス[2]という標準化された技術を用いて、Grid を構成するノード同士がシームレスに連動できるようにした。このシステムは Java 言語で作成されている。ここで、仕様で述べた各機能の説明を行なう。

3.1 実際に依頼された仕事を行なう機能

Java のインターフェース機能を用いて、問題とそれを解く為のコードを含んだインスタンスをシリアライズして、お互いのワーカ間で転送し、決められたメソッドを呼び出すことにより汎用的な処理の実現、すなわち General Purpose Computing Grid を実現する。ここで、この仕事がカプセル化されたクラスにおいて、個々のワーカが自律的に負荷分散を行なうことができるように、問題を解くコードだけではなく、問題の分割方法もこのクラスで定義した。

3.2 Grid の他のワーカを動的に発見する機能

Pure P2P というノードすなわちワーカに他ノードを直接に発見する方法ではなく Hybrid P2P という、ある中心的なセンタが存在し、そのセンタを介して他のノードを発見し、その後はノード同士が直接通信を行なうというアプローチをとった。この様な実装によって、各ワーカの所在を管理する部分が切り放され、その登録センター間でワーカ割り当ての効率化が期待できる。

† 埼玉大学 Saitama University

3.3 新たな子ワーカに動的に仕事を依頼する機能

１つのポリシーのみに沿って動作させるというのは非現実的であると考えた。そこで、自分のマシンのパワーや資源を貸し出せる時間、仕事の作成者が想定するデフォルトで何人を付けるのが最適かという設定、これら複数のポリシーに従ってマネージャになる、またその動作の様子を決定できるように設計をしているが、現段階では一部のみしか実装できていない。

3.4 子ワーカから結果を受け取る機能

Globus Toolkit のプッシュ型のノーティフィケーション(通知)機能を用いて、子ワーカ群のマネージャが定期的に受け取りに行くという方法ではなく、仕事を終えたワーカがマネージャに通知と共に結果を送信する。

この実装方法により、ワーカは終了の通知後、すぐに暇なワーカとしてシステムに復帰することができる。またネットワーク負荷も同時に押えることができ、マネージャの負荷も軽減される。Grid の資源が枯渇した場合に、この空いたノードをすぐに再利用できるというメリットもある。

帰ってきた結果は、シリアライズされており、この情報の解凍方法を、問題を解くクラスに実装させることで、仕事のインスタンスのみで仕事を完了することができる。

3.5 自分が担当した仕事のセグメントが全て正しく終了したか監視する機能

Grid システムが階層構造を成すという特徴から、もし階層の深い方でシステムの障害により、仕事を正しく終了することができなくなってしまったワーカが発生した際に、上方の方では子ワーカの終了をいつまでも待ち続けてしまい、結果としてシステム全体の動作がストップしてしまうという問題が発生する。仕事を提起したルートマネージャに障害が起こってしまった場合は仕方がないが、一つのワーカにより全ての仕事が無駄になってしまわないような機能として、自分は誰にどのような仕事を依頼し、結果としてどうであったかをデータベースで管理し、一定時間反応の無い子ワーカが存在した際には、データベースから仕事がカプセル化されたインスタンスを取り出し、自分がその失敗したワーカに変わって仕事を行なうことによって障害から回復することができる。他にもフォールトトレランスの機能が必要であろうが、現在は検討中である。

このような実装によって、適応型の階層構造を形成する Grid システムが実現できると考える。

４．検証

作成した Grid システムの検証を行なった。多数の計算機上で実際に問題を解かせる方法がベストであろうが、今回は 1 台の計算機上で多数のワーカスレッドを起動し、性能の検証ではなく、動作の検証を行なった。また、ワーカの所在を管理するセンタも同様に１つ作成した。

用いた問題としては、ある正方形の空間があり、その空間内に無数に点があったとき、正方形をより小さい区分に分割し、それらの小正方形のなかには必ず 1 つの点しか存在しないように、再帰的に空間を分割していき、条件が満たされるまで繰り返すというものである[3]。

この問題は計算途中で動的に負荷が発生し、 その負荷が不均衡であるという性質を持つものである。 提案システム

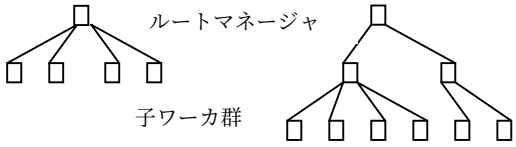

図 1. 負荷に対するシステムの適応の様子

上でこの問題を実際に投げ掛け動作の様子を追った。

結果は図１のように、負荷が高まる部分を担当したワーカにおいて、他の比較的負荷の軽い部分よりも活発に負荷分散、すなわち子ワーカを作りだし、それが階層構造を成す動作が確認された。すなわち、動的に発生した負荷に対し、必要なところに計算資源を動的に多く配分することで、その仕事の特性にうまく適応しているということが言える。

５．関連研究

ここで、この提案システムと関連のある既存の研究との差異を明白にする。階層的に負荷分散を行なうようなシステムとして Jojo[4]や Condor[5]というシステムがある。他にも階層構造を利用するシステムが幾つか存在するが、これらは、元々クラスタ等の計算の為の分散環境が構築されているシステムの連携を用いるのに対し、提案システムでは、P2P というシステム、すなわち全く計算用に環境が特化されていない状態のシステムから、階層構造を仕事の発生に伴って動的に構築していくという点が異なる。

また Jojo 等では大きな粒で Grid を作成するのに対し、提案システムでは個人の PC を多数用いて Grid を作成するという手法であり、小さな粒が集まり、その個々の粒が動的にマネージャにもワーカにもなれるという点が大きく異なる点であり、個人の PC のような最小規模のような環境からでも、うまく Grid を利用・構築することができるという利点がある。

６．まとめ

このように各ワーカが自律的に負荷の分散を行なう機能、また階層的にシステムが再構成する機能を持たせることによって、本システムの目的である、個人が利用できる汎用的な Grid システムが実現できるものと考える。

今後は、提案システムにおいて未実装部分の実装や、実装方法を検討している部分の実装を行い、より実用段階に近づけて行く。また、複数台の計算機を用いて、システムを稼動させた際に、１台の場合で仕事を行なうより、どの程度の性能の向上が得られているかも検証して行く。

参考文献

[1] http://www.globus.org

[2] http://www.w3.org/2002/ws

[3] 飯塚　肇・緑川博子　共訳，　並列プログラミング入門,丸善

[4] 中田 秀樹,松岡 聡,関口 智嗣, Java による階層型グリッド JoJo の設計と実装, 情報処理学会論文誌コンピューティングシステム Vol.44, No.SIG 11, pp.46-56, September 2003

[5] Ali Raza Butt,Rongmei Zhang, and Y.Charlie Hu. A Self-Organizing Flock of Condors. In Proceeding of ACM/IEEE SC2003: International Conference for High Performance Computing and Communications,2003
http://www.sc-conference.org/sc2003/paperpdfs/pap265.pdf

[6] 佐藤,動的階層構造を持つ GRID システム,埼玉大学卒業論文,2004