1B2-01

ノード群の協調による複数ネットワーク間における GRID 構築システムの設計と実装

LampEye: GRID Construction System by using nodes collaboration

大迫 勇哲[*1] 山崎 航[*2] 西山 裕之[*3] 溝口 文雄[*3]
Takenori Ohsako Wataru Yamazaki Hiroyuki Nishiyama Fumio Mizoguchi

[*1]東京理科大学大学院 理工学研究科
Graduate School of Science and Technology, Tokyo University of Science

[*2]東京理科大学 総合研究所 インテリジェントシステム研究部門
Intelligent-System Research Laboratory, Research Institute for Science and Technology, Tokyo University of Science

[*3]東京理科大学 理工学部
Faculty of Science and Technology, Tokyo University of Science

In this study, we designed and implemented LampEye that construct GRID environment by using nodes collaboration. LampEye provides feature of traversing NAT by using UDP hole punching and UPnP. Using LampEye, user can easily construct GRID composed of computer groups in two or more different network environments.

1. はじめに

近年の高速ネットワークインフラの普及や計算機の高性能・低価格化などにより，複数の計算機や情報機器を協調させて使用するグリッドコンピューティング (以下，グリッド) が注目されている．グリッドには，使用する機器の種類やその目的によって様々な種類があるが，本研究で対象とするグリッドは，主に個人向け計算機を使用し，分散処理環境や地理的に分散した作業者間での共同作業環境などの構築を目的としている．

グリッドは，同一ネットワーク内に存在する計算機のみで構成されるのではなく，一般的に複数の異なるネットワーク内に存在する計算機によって構成される場合が多い．具体的には，企業や研究機関，家庭などのローカルネットワーク内に存在する計算機をインターネットなどを介して統合し，グリッドを構築するような場合である．また，各ローカルネットワーク内においても，複数の内部ネットワークを階層的に構築している場合は少なくなく，これらの環境においてグリッドを構築する場合にもあてはまる．以上のような場合，問題となるのが，各計算機間の通信方法である．すなわち，ローカルネットワーク内 (あるいは内部ネットワーク内) に存在する計算機から他の異なるローカルネットワーク内に存在する計算機へデータ等を直接送信することは，通常，困難である．Globus Toolkit[2] などのグリッドを構築するためのソフトウェアは多く開発されているが，このような環境への対応は不充分である．

現在，このような環境でグリッドを構築するためには，グリッドの利用者が，(1) グリッドを構成する各計算機から接続可能な計算機を設置してデータ等を中継させる方法や，(2) NAT (Network Address Translation) などに対しポートフォワーディング等の設定を行う方法を用いることで，自ら対応しなければならない．しかしながら，(1) の方法では，中継を行う計算機に対して負荷が集中する可能性があり，スケーラビリティや耐故障性などの面で問題がある．(2) の方法でも，設定作業が面倒であったり，そもそも設定を変更する権限がないなどの問題が考えられる．

連絡先: 大迫 勇哲，東京理科大学大学院 理工学研究科 経営工学専攻，住所：千葉県野田市山崎 2641，電話番号：04-7124-1501(内線：3855)，E-Mail：j7404618@ed.noda.tus.ac.jp

本研究では，以上のような問題を解決するグリッド構築システム LapmEye の設計と実装を行った．LampEye によって構築されたグリッド環境では，異なる複数のローカルネットワーク内にそれぞれ存在する各計算機が，ほとんどの環境において，前述した (1) や (2) の対策をグリッド利用者自らが行うことなく，通信を行うことが可能になる．また，LampEye は，環境情報や他計算機からのメッセージ，設定等に基づいた動作を，各計算機に独立して行わせることにより，自律的な資源の統合やジョブの実行を実現する．

2. 設計

2.1 ノードの自律的統合

LampEye によって各計算機 (ノード) は，図 1 のようなノード管理用の組織を構築し，最初に接続される側を親，接続する側を子とする相対的な関係を持つ．また，各ノードは”ノード名. 親ノードの ID”という ID で管理される．また，上位に位置する各ノードは，ゲートウェイノードとして親・子の関係を持たずに相互に接続する．このような構造をとることにより，メッセージやデータ等のブロードキャストやユニキャスト等が，親あるいは子を知っているだけで可能になる．

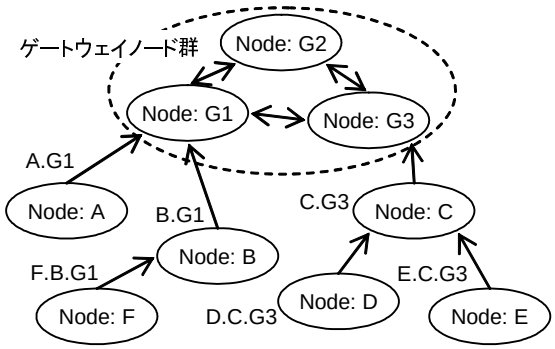

図 1: ノード組織の例

組織化されたノードの資源を利用するためには，まず，ジョブの実行条件に関するメッセージをブロードキャストし，条件を満たすノードからの返信を待つ．次に，返信があったノードの中から適切なものを選択し，2.3 の方法で接続を行い，ジョ

ブを実行する．

2.2 NAT 越え技術

異なる複数のローカルネットワーク内 (あるいは内部ネットワーク内) にそれぞれ存在する各計算機間の直接通信を実現するためには，NAT 越え (NAT の外側の計算機から内側の計算機へ通信する) が必要となる．LampEye では，NAT 越えを実現するための技術として，UPnP(Universal Plug and Play)[3] と UDP hole punching の 2 つを使用している．

UPnP UPnP に対応した情報機器に対し，XML に準拠した形式のメッセージを送信することで操作を行うことができる．この機能を利用して NAT に対しポートフォワーディングの設定を自動的に行う．しかしながら，対象となる NAT が UPnP に対応している必要がある．

UDP hole punching STUN[4] でも用いられている技術で，NAT の内側にある計算機の UDP ソケットにマッピングされる NAT 上のポートが，アドレス変換ルールを NAT が保持している間は常に同じであるという原理を利用して，外部から内部への UDP パケットの送信を可能にする．しかしながら，送信先ごとにポートを変更するタイプの NAT には適用できない．なお，約 8 割の NAT に適用できるとする調査結果もある [5]．

2.3 ノード群協調による接続ステップ

LampEye は，通信を行う通信ノードと，これらのノードの仲介を行う仲介ノードの三台一組で，NAT 越えを行う．仲介ノードは，基本的に，2 台の通信ノードに共通する上位ノードである (直接の親ノードである必要はない)．仲介ノードの主な役割は，(1) 各通信ノードの環境 (NAT タイプなど) を把握する，(2) 通信ノード間のメッセージを中継する，(3)NAT 越え通信の方法を決定する，の 3 つである．以下の流れで異なるネットワーク間での通信を実現する．

Step1 通信ノードを，仲介ノードに接続する．

Step2 仲介ノードを利用してそれぞれの NAT のタイプを判断する．UDP hole punching が適用できるのならば 2.4 のように直接通信を開始し接続ステップを終了．できないのであれば Step3 へ．

Step3 NAT の UPnP 機能の有無を確認する．UPnP 機能があればポートフォワーディングを自動設定し，接続ステップを終了．UPnP 機能がなければ Step4 へ．

Step4 直接通信をあきらめ，他のノードに通信の中継を依頼する (中継方式)．接続ステップを終了．

2.4 UDP hole punching による NAT 越え

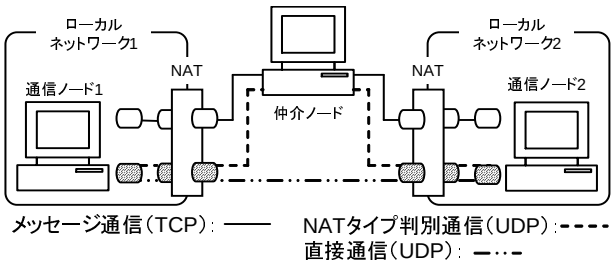

図 2: UDP hole punching による NAT 越えの概要

UDP hole punching による NAT 越えは以下のようになる．まず，通信ノードと仲介ノードをそれぞれ接続し，通信ノード間の直接通信に使用する UDP ソケットを作成する．次に，仲介ノードに対し UDP パケットを送信し，作成した UDP ソケットにマッピングされた NAT 上のポートを調べる．最後に，各通信ノードは，相手の UDP ソケットにマッピングされた NAT 上のポートに対し，UDP パケットを送信する．以後，仲介ノード等を介さない直接通信が行える．

3. 実装

LampEye は，Java 言語のみを用いて実装されている．ジョブの実行については，LampEye のパッケージを利用する方法と図 3 のような GUI を利用する方法の 2 つがある．前者の方法では，処理の依頼や結果の取得などを簡潔に記述することができ，また，オブジェクトの状態で結果を取得することも可能なため，既存の Java アプリケーションを容易に拡張できる．後者の方法では，各ノードを視覚化する機能を利用して，マウス操作で直感的にジョブの投入を行うことができる．

実際に，東京理科大学野田キャンパス内の 2 拠点と千葉県内の学外の 2 拠点にある各計算機 (計 21 台) において，LampEye を用いてグリッドを構築した．複数のアプリケーション等を用いた実験から，構築されたグリッドは，十分な台数効果を得ることができ，また，通信ノードが増加した場合でも，従来の中継方式に比べ，通信遅延を抑えることに成功した [1]．

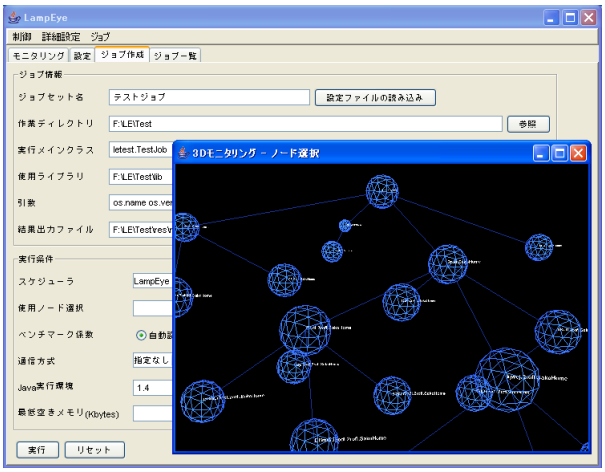

図 3: LampEye ジョブ実行 GUI およびモニタリング

4. おわりに

本研究では，異なる複数のローカルネットワーク (内部ネットワーク) 間における直接通信が可能なグリッド環境を構築するためのシステムである LampEye の設計と実装を行った．LampEye を利用することで，従来のグリッド構築ソフトウェアよりもネットワーク環境の制約を受けにくくなり，分散処理だけでなく，大量のデータ通信を必要とするデータグリッドや，テレビ会議システムなどの地理的に分散した利用者間での共同作業環境を容易に構築することが可能となる．

参考文献

[1] 大迫勇哲，山崎航，西山裕之，溝口文雄：グリッド環境における自律的な資源統合ソフトウェアの設計と実装，情報処理学会第 67 回全国大会，2005.
[2] Globus Toolkit, http://www.globus.org/
[3] Universal Plug and Play，http://www.upnp.org/
[4] Simple Traversal of UDP through NATs，RFC3489.
[5] Bryan Ford，Pyda Srisuresh，Dan Kegel：Peer-to-Peer Communication Across Network Address Translators，Proceeding of USENIX 2005，pp.179-192，2005.