1T-3

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

大迫勇哲 † 山崎航 †† 西山裕之 ‡ 溝口文雄 ‡
† 東京理科大学大学院 理工学研究科経営工学専攻 †† 東京理科大学 情報メディアセンター
‡ 東京理科大学 理工学部経営工学科

1 はじめに

本研究では，複数の環境における計算資源を統合して利用するグリッド・コンピューティング環境において，(1) 各計算機の非均質性や環境 (あるいは計算機) ごとの利用ポリシ，使用権限等を考慮したスケジューリング，(2) 異なる環境間における通信の確立，を主たる目的としたソフトウェア LampEye (以下，LE) の設計と実装を行った．

LE は，環境情報や他計算機からのメッセージ，自らの設定等，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，自律的な計算資源の統合やジョブの実行を実現する．加えて，機器の故障や各環境の管理主体が異なることなどに由来する，統合計算資源の動的な変化 (追加や離脱) にも対応する．また，通信を行う計算機がそれぞれ異なるルータ内にある場合にも，中継サーバを利用せずに直接通信を行う機能が実装されている．

2 設計

2.1 自律的な資源の統合

LE によって各計算機 (ノード) は，木構造の組織を構築する．各ノード間には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が成立し，また，各ノードは”ユニークな文字列. 親ノードの ID”という ID で管理される．このような構造をとることにより，メッセージ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．LE は，以下のステップで資源を組織化し，グリッドを構築する．

Step1 LE を起動し，複数のベンチマークプログラムの実行や，環境情報・計算機情報の収集を行う．
Step2 既知の起動済みのノード (仮親) に接続し，他ノードのリスト (ノード情報，IP，Port) を得る．
Step3 情報を得たノード (仮親も含む) の中で，(1) 接続ルールにおける条件を満たす，(2) 相手ノードが接続を拒否しない，という 2 つの条件を満たすノードのリストを作成する．
Step4 評価ルールを参照し，リストの中で最も評価値が高いものを親ノードとし，TCP で接続する．他ノードの情報はキャッシュとして保存する．
Step5 自らのノード情報及び ID を全ノードに通知する．

また，親ノードの切断や評価・接続ルールの更新があった場合には，キャッシュ等のノード情報をもとに Step2 からやり直す．ただし，子ノードが接続している場合には，Step3 において，(3) 子ノード及びその下流に接続しているノード (下流ノード) には接続しない，という条件を加える．さらに，Step5 後に，子ノード及び下流ノードの ID を書き換えさせる．

2.2 統合された計算資源の利用

LE は，統合された計算資源に対して，以下のステップでジョブを処理させる．

Step1 ジョブの依頼を行うノード (依頼ノード) を 2.1 のようにグリッドに組み込む．
Step2 依頼ノードが，処理を行うノード (処理ノード) の条件を全ノードに通知する．
Step3 条件メッセージを受信したノードは，自らの利用ポリシ及び計算機情報等を参照し，条件を満たせば許諾メッセージを依頼ノードに送信する．
Step4 依頼ノードは，許諾メッセージを返信したノードに対し，2.3 の方法を用いて他ノードを介さない，直接通信を試みる．
Step5 通信が確立できたノードに対しては，2.4 のスケジューリングを行い，ジョブを処理させる．

2.3 NAT 越え

ジョブ実行時には，通信の安定を目的に，他ノードを介さずに通信を行う．問題は，図 1 のように，通信を行うノード (cs.p と cr.p) がそれぞれ異なるルータ内に存在する場合である．LE は STUN(Simple Traversal of UDP Through NATs) プロトコルと同じ手法を用いて NAT 越え直接通信を実現する．ここで，p が親，cs.p，cr.p が子であり，cs.p から cr.p に接続する．

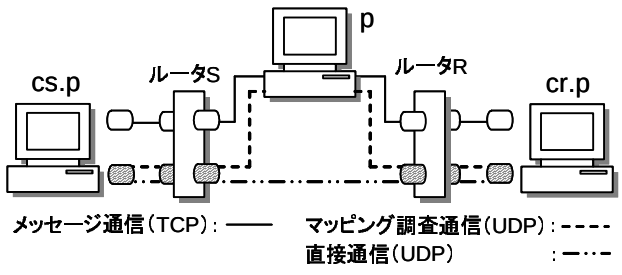

図 1: NAT 越えの概要

Step1 cs.p から p を介して cr.p に接続要求を送信する．
Step2 cs.p，cr.p は，直接通信を行うための UDP ソケット (dcs) を，メッセージ通信用とデータ通信用にそれぞれ 2 つずつ生成する．
Step3 cs.p，cr.p は，各 dcs に対してマッピングされるルータ S，R 上の IP と Port を調査するために，p に対して各 dcs から UDP パケットを送信する．
Step4 p から cs.p，cr.p に，相手の各 dcs がマッピングされた相手ルータ上の IP と Port を通知する．

Design and implementation of autonomous software integrating resource on GRID
Takenori Ohsako†, Wataru Yamazaki††, Hiroyuki Nishiyama‡, Fumio Mizoguchi‡
{†Graduate School of Science and Technology, ††Information Media Center , ‡Faculty of Science and Technology }, Tokyo University of Science

Step5 cs.p，cr.p は，通知された相手 IP と Port に対して，各 dcs を使用して UDP パケットを送信する．
Step6 互いにパケットの到達が確認できれば，各 dcs を使用した直接通信が行える．

本手法において UDP を利用するが，UDP はパケットの送信先への到達を確認する機構を持たない．LE は，各 UDP パケットのデータ域の先頭 5 バイトに”Type”と”メッセージ ID”を付加して送信し，送信先からの到達確認メッセージ (送信パケットの”メッセージ ID”) を受信するまで，一定間隔で送信を繰り返す．これにより送信先への UDP パケットの到達を確認する．

2.4 スケジューラ

過去の研究 [1] では，複数のベンチマークの結果と，異なるスペックの計算機 2 台以上の実際の処理時間から，そのジョブに対する各計算機の性能値を算出し，この値から他計算機の処理時間の予測を行う方式を設計した．具体的には，計算機 m_i における n 個のベンチマークの結果を，基準計算機の結果で割り，その値を $V_{m_ij}(j=1,2,\cdots,n)$ とするとき，この計算機の性能値 CP_{m_i} を次の式 1 ように定義する．

$$CP_{m_i} = \sum_{j=1}^{n} W_j V_{m_ij} \quad (\sum_{j=1}^{n} W_j = 1, W_j \geq 0) \quad (1)$$

ここで，計算機 m_a と m_b におけるジョブ 1 個あたりの処理時間を T_{m_a}，T_{m_b} とするとき，$\frac{T_{m_a}}{T_{m_b}} = \frac{1/CP_{m_a}}{1/CP_{m_b}}$ を満たすような W_j を確率的な手法を用いて決定する．

LE のスケジューラは，資源要求に応じてきたノード全てに対してジョブを投入し，異なるスペックの計算機 2 台以上の処理時間が計測できた時点で，W_j を決定し，各ノードの CP_{m_i} を算出する．各 CP_{m_i} から各ノードの処理時間を予測し，残りのジョブの分配数を調整する (実行中のジョブを破棄する場合もある)．また，処理時間と予測時間の誤差が大きい場合には再調整を行う．

3 実装

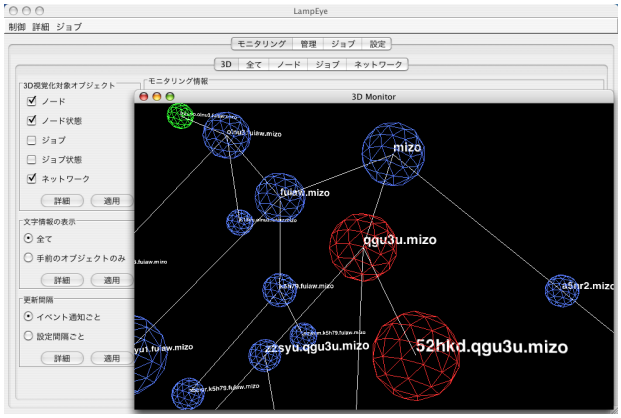

図 2: LE のジョブやノードのモニタリング機能

LE は，Java 言語のみを用いて実装されているため，Java の実行環境がインストールされている計算機であれば実行が可能である．

統合された資源へのアクセス (ジョブの実行) は，LE のパッケージを利用する方法と GUI を利用する方法の 2 つがある．前者の場合は，パッケージを import し，ResourceManager オブジェクト (rmo) を生成する．rmo に対してジョブオブジェクト (jo) を add することで処理を依頼し，get で処理後のオブジェクトを取り出せる．jo は，Job インターフェイスを実装したクラスから生成され，public Object exec() メソッド内に処理ノード上での処理を記述する．後者の場合は，実行するクラスファイルやデータファイル，実行時の引数，処理ノードの条件などを入力することで利用できる．図 2 に処理ノードの状態やネットワーク等を 3D で視覚化し，モニタリングを行っている様子を示す．

なお，処理ノード上でのジョブ実行時には，必要なクラスファイルを，ネットワークを介して依頼ノードから動的にロードできるクラスローダを実装し，使用する．

4 評価

4.1 NAT 越え直接通信の評価

図 1 の場合において，LE の通信機能を用いて通信を行う場合と，中継サーバ介して通信を行う場合の比較を行う．全ての機器は 100Mbps の LAN で接続されている．図 3 に，1 ノードあたり 10 バイトのメッセージを 1000 回送受信するのに要した時間と，同時に通信を行ったノードの数を示す．LE の通信機能の方がより高速に通信できており，また，通信ノードが増加した場合にも通信時間の増加を抑えることができている．

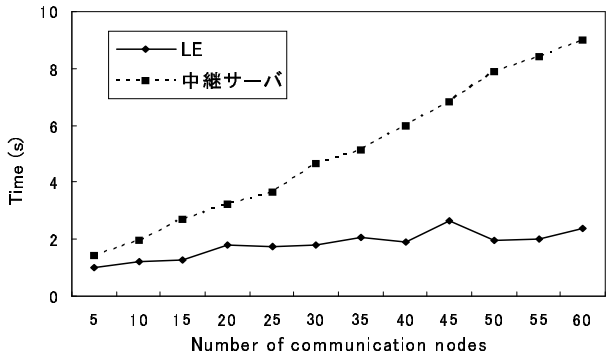

図 3: 通信時間の比較

4.2 スケジューラの評価

顔の老化シミュレータ [2] を LE 上で実行する．これは顔の老化を 1 年ごとにシミュレートするもので，ジョブ数は 40，各ジョブは均一で，使用するデータは独立している．2.4 の LE 方式との比較として，処理が一つ終わるごとにジョブを順次投入する Self-Scheduling 方式 (SS 方式) と，単一のベンチマーク値 (CaffeinMark3.0[3] の Overall) から算出した性能比に基づくジョブ分配を行う方式 (Bench 方式) の二つでも処理を行った．計算機は 6 種類，13 台を使用した．結果は，LE，SS，Bench の各方式がそれぞれ，49.87，57.42，61.86(秒) となった．処理を行う計算機の構成にもよるが，本実験の構成では，LE 方式が最も短い時間で処理が行えたことがわかる．

5 おわりに

本研究では，自律的な計算資源の統合と利用を可能にするソフトウェア LampEye の設計と実装を行った．評価実験から LE の NAT 越え通信機能やスケジューラは，他の方式に比べ優れているという結果が得られた．

参考文献

[1] 大迫勇哲，山崎航，西山裕之，溝口文雄，グリッドにおけるタスク処理履歴情報に基づく計算機資源選択方式の設計，日本ソフトウェア科学会第 21 回大会

[2] 及川雄揮，平石広典，溝口文雄，コンピュータグラフィックスによる顔の老化の表現に関する研究，情報処理学会第 66 回全国大会

[3] CaffeinMark3.0, http://www.benchmarkhq.ru/cm30/