P2P による転送遅延を考慮した静的負荷分散方式

塩谷 康夫† 太田 学‡ 片山 薫‡ 石川 博‡

†‡東京都立大学大学院工学研究科 〒192-0397 東京都八王子市南大沢 1-1

E-mail: †shioya@hikendbs.eei.metro-u.ac.jp, ‡{ohta, katayama,ishikawa}@eei.metro-u.ac.jp

あらまし 分散コンピューティングにおいて、負荷を均一に分散させると、各ノードの処理時間にばらつきが生じる。ゆえに、全体の処理時間が最も負荷処理速度の遅いノードの処理時間に依存してしまう。それゆえ、各ノードの能力に応じて負荷を分散させれば、負荷処理時間の均一化により、全体としてより効率的に処理を実行できる。その際、負荷の転送による転送遅延の影響を加味して分散すれば、より効果的である。本研究では、ノードの処理能力を示す指標としてランクを転送遅延の影響を加味して定義し、ノードのランクを隣接するノードのランクから決定する方法を示す。また、ランクに基づく負荷分散方式を提案する。

キーワード P2P ネットワーク，分散コンピューティング，負荷平衡

A static load distribution scheme considering the transmission delay in P2P

Yasuo SHIOYA† Manabu OHTA‡ Kaoru KATAYAMA‡ and Hiroshi ISHIKAWA‡

†‡Graduate School of Engineering, Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji-shi Tokyo, 192-0397

E-mail: †shioya@hikendbs.eei.metro-u.ac.jp, ‡{ohta, katayama,ishikawa}@eei.metro-u.ac.jp

Abstract In distributed computing, when a load is distributed uniformly, the processing time varies with nodes. Therefore, the whole processing time is dependent on the slowest node. If load is distributed according to the capability of each node, we can expect more efficient total processing by equalizing the processing time on each node. Furthermore, it is more effective that the transmission delay is considered when distributing the load. Considering the influence of transmission delays, this paper defines a “rank” as a metric which indicates the throughput of each node, and shows how to determine “rank” of each node from “ranks” of adjoining nodes. Moreover, this paper proposes a technique to distribute load based on the “rank” of a node.

Keyword P2P Network, Distributed Computing, Load Balance

1. はじめに

ネットワークを介して複数のコンピュータを結ぶことで、大規模な計算を高速に行う Grid computing が注目されている。P2P（peer to peer）による分散コンピューティングもその例である。

P2P による分散コンピューティングにおいて、隣接するノードに処理を行わせる時、単純に負荷を均一に分散させるよりも、各ノードの能力に応じて分散させれば、全体としてより効率的に処理を実行できると期待できる。その際、ネットワークの各ノードの転送速度を考慮して、負荷を分散させれば、より効果的に処理が行えると期待できる。そこで本研究は、P2P ネットワーク上でのデータマイニングを行うに際して、この負荷分散方式を適用する。

本研究の負荷分散方式によって分散コンピューティングを行う目的は、ネットワークを構成する各ノードのユーザが、大規模な計算を安価な設備で、より効率的に、高速に処理が行えるようにすることである。

この負荷分散方式の適用により、効率的な処理が行える例として、ネットワークの各ユーザが、データマイニングを行う場合や、大規模な計算（研究データの処理等）を処理する場合が挙げられる。

以前、我々が行った研究[1]、[2]では、各ノードでネットワークを通して処理させる際の、負荷の転送遅延の影響を考慮していない。また、各ノードにおいて転送されてきた負荷からローカルラージアイテムセットを生成する処理は、Apriori[3]で行っている。そこで、本研究は、各ノードでのローカルラージアイテムセットを生成する処理には、partition アルゴリズム[4]を用いることにより、高速に処理を行える様にした。この処理方式に併せて、転送する際に生じるネットワークの転送遅延の影響を加味し、各ノードの処理時間が均

一になるように、負荷を分散させるための基準となるノードのランク付けの方式を提案する。また、このランク付けを用いるためのネットワークの構成方法も提案する。

本稿は、2 章で関連研究について述べ、3 章では本研究で想定するネットワーク構造について述べる。4～7 章で、負荷分散の基準となるノードのランクを隣接するノードのランクから決定する方式について説明する。8 章で、本手法に基づき、データマイニングにおけるアソシエーションルール抽出実験を行った。9 章でまとめを述べる。

2. 関連研究

2.1. Grid computing

Grid computing は、ネットワークに接続された世界中のコンピュータ資源を活用して、高性能なコンピュータの代わりに、大規模な計算を高速に処理する技術である。具体的な活用の例として、SETI@home[5]がある。これは、インターネットに接続されたコンピュータを活用することで、電波望遠鏡によって受信した、宇宙からの電波の解析を行い、地球外生命体の発見を目的とするプロジェクトである。

2.2. 問い合わせ検索に用いる負荷分散

P2P を用いた大規模分散システムにおいて、問い合わせの答えとなるデータソースの発見を行う際、各ノードにおいて、その場所の検索の処理の回数の均一化をすることで、処理の高速化を行う Leonidas Galamis らの研究[6]がある。これは、DHT（distributed hash table）として、Chord を用いることで、各ノードに対して、検索の処理回数の均一化を行っている。この時検索の処理回数の均一化を実現するために、索引となる情報（カタログ情報）をキーとして、各ノードに持たせることで行っている。その結果、過負荷により問い合わせの処理ができない状態を少なくし、高速な応答を実現している。

本研究は、各ノードの負荷の処理回数の均一化を行うのではなく､各ノードの負荷の処理時間を均一化し、負荷分散による処理を高速に行うことを目的とする。

2.3. SAN（Storage Area Network）による動的負荷分散機構

SAN を適用した PC クラスタによる並列データマイニングを行う合田らの研究[7]がある。これは、SAN という専用のネットワーク上で、動的負荷分散機構を設計し、HPA（Hash Partitioned Apriori）によってアソシエーションルールの抽出を行うものである。その特徴は、構成ノードを動的に投入することにより、CPU パワーを増加させることで、CPU バウンドを抑制している。また、データが存在しているストレージデバイスからデータを分割して、未使用のストレージデバイスにコピーすることにより、並列にアクセスを行うことを可能にし、ディスク I/O の帯域を拡張することで、I/O バウンド時の性能改善を図っている。

本研究は、SAN という専用のネットワークでは無く、LAN もしくは一般的な回線を想定しており、データ転送速度が遅いという点で異なる。また、構成ノードの CPU パワーを、負荷分散を開始するノードが任意に制御することができないという点で異なる。

3. ネットワークの構成

本研究で想定したネットワーク構造は、図１に示すように、ネットワークを構成するノードの情報を管理するサーバーがあり、その下に構成ノード（peer）があるとする｡この時のサーバーのことを superpeer と表す。また、実際に負荷分散を行う場合には、ネットワークを構成している各ノードは、superpeer を利用して、負荷処理のネットワークを構成した後、処理を行う。この負荷処理のネットワークにおいて負荷処理の実行単位を subpeer と表す。

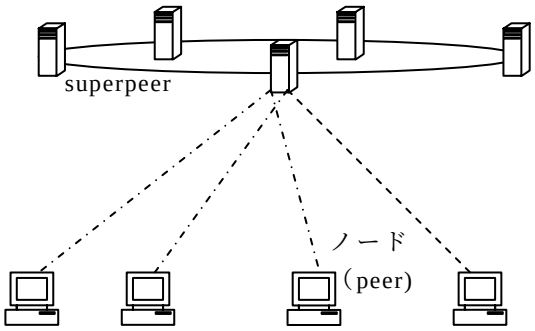


図 1　superpeer の役割

3.1. superpeer

superpeer とは、一般に P2P ネットワークにおいて、高速で信頼性があり、サーバーのような特性を持つノードが、ネットワークの管理等を行う。複数の superpeer が、図 1 に示すように、負荷を分散させて管理することで、ネットワークの安定性を増加させ、効率的に用いることができる。

本研究では、この superpeer を拡張して用いている。本研究で付加した役割は、後述する subpeer の管理、新たにネットワークに加わったノードの subpeer の作成、負荷処理のネットワーク構成時の、負荷実行能力を表すランクの計算を行うことである。

また、superpeer が全てのノードの処理能力や状態を一元的に管理した方がネットワークも高速化されるので、より効率に処理できると考えられる。しかし、本研究で superpeer にそのような機能を持たせないのは、P2P では各ノードは自由にネットワークに接続する（接続を切る）ことができ、自身の処理能力の使用制

限を行えるので一元管理するのは難しいからである。

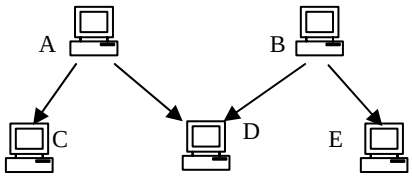
3.2. peer

peer は、subpeer と一対多の関係になっている。peer が subpeer を内包しているからである。内包する subpeer は、peer がある一定期間（T）内に異なる負荷分散によって処理した負荷のタスク数とする。これにより、peer において処理に使用されない余剰な能力を少なくするためである。

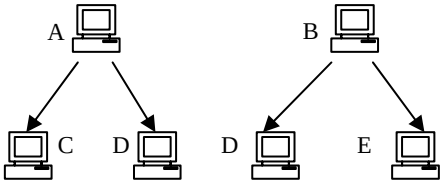
3.3. subpeer

subpeer は、負荷処理のネットワークにおいて負荷処理の実行単位を表わしている。その役割は、複数のノードが異なるタスクに関し、負荷分散による処理を行う時、その個々の処理が相互に影響しないで、独立に処理を行えるようにするという役割である。例で示すと、図 2 の（a）のようなネットワークがあり、ノード A、B から異なるタスクを与えられ、負荷処理を行うとする。この時、図 2 の（b）のような負荷処理のネットワークを構成できる。その際、subpeer の役割は、ノード D に関し、2 つの処理が独立に処理できるようにする役割である。

このことから、subpeer は元のノードの処理能力を分割した処理能力を持ち、異なるタスクごとに個々に割り当てられる。



（a）　ネットワーク



（b）　負荷処理のネットワーク

図 2　subpeer の役割

3.4. 負荷処理のネットワークの構成

本研究における負荷処理のネットワークの構成の手順を以下に示す。その例を図 3 に示す。この時、ネットワークの隣接関係は変化させずに構成することを想定する。

① 負荷処理を開始するノード（ルート）は、負荷分散の範囲（転送回数 P）を決定する。

② 隣接ノードに負荷処理を行うか問い合わせる。

③ 依頼を受けたノードは、同様にその隣接ノードに負荷処理を行うか問い合わせる。

④ ③を転送回数が P になるか、隣接ノードが無くなるまで繰り返す。

⑤ 末端のノードは、構成ノード（peer）の情報を superpeer に送る。

⑥ superpeer が、構成ノードの subpeer の情報により、負荷処理のネットワークにおける負荷実行能力を表すランクの計算を実行する。

⑦ superpeer は、ルートと中継のノードにランク情報を転送する。

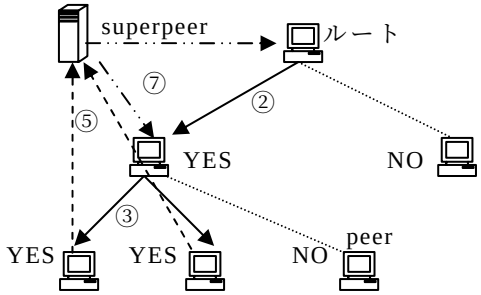


図 3　負荷処理のネットワークの構成

3.5. subpeer 情報の更新

subpeer の情報を持つノードは、ある一定期間（T）ごとに、subpeer の情報に更新があった場合に superpeer にその情報を転送する。これは、実際に処理したタスク数によって subpeer の数を定めるために、タスク数の変動で最適化をする必要があるからである。また、subpeer の数よりも多いタスクが同時にノードに到達した時は、その時点で subpeer の情報を更新し、その情報を superpeer に転送する。これにより、更なるタスクがそのノードに到達し難くし、処理効率の悪化を防ぐ。加えて、コンピュータの性能が変化した場合にも subpeer の処理能力が変化するので、更新を行う。

3.6. 処理の流れ

負荷処理のネットワークを構成した後の、処理の流れを以下で述べる。その例を図 4 に示す。

① ルートは、隣接ノードのランク情報から負荷を分割し、隣接ノードに転送する。負荷転送と同時に、partition アルゴリズムを用いることにより、割り当てられた負荷からローカルラージアイテムセット（L）の生成を行う。その際、負荷は一定の F＋ΔF 以下のデータサイズを基準に均等に分割して処理する。

② 中継のノードでも同様に、隣接ノードのランク情報から負荷の分割をし、隣接ノードに転送する。この時、ルートと同様な方式で割り当てられた負

荷の処理を行う。

③ 末端のノードに到達するまで、②を繰り返す。あるいは、負荷のデータサイズが F＋ΔF 以下に収まった時点で終了する。

④ 末端のノードで、ルートと同様な方式で割り当てられた負荷の処理を行う。

⑤ L を負荷の転送経路を逆順にルートに転送する。

⑥ 中継ノードで L を統合し、転送経路を逆順にルートに転送する。

⑦ ルートは、L を統合する。

⑧ 統合結果より、アソシエーションルールを得る。

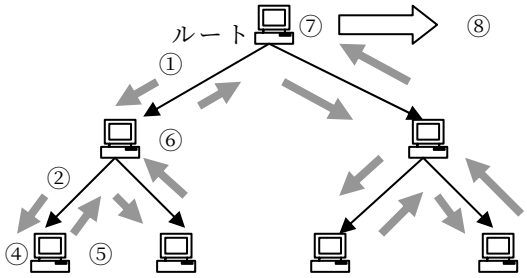


図 4　負荷分散の処理の流れ

4. 処理能力の定量化

4.1. peer の処理能力の定量化

ある負荷処理を実行する時、処理時間は、そのコンピュータの CPU の種類、速度、メモリの容量、動作クロック数、マザーボードの種類、HDD の速度等の複数の要因が互いに影響して決まる。そのため、コンピュータの処理能力を、コンピュータの構成要素から定めるのは困難である。そこで、本研究においてコンピュータの処理能力を定量化する方法を説明する。

① 処理能力を 1 と定めるベンチマーク用データ（サイズ d）の処理時間 t_0 を superpeer に設定し、基準となる処理速度 v_d[kB/s]を（1）式と定める。

$$v_d = \frac{d}{t_0} \quad \text{[kB/s]} \quad (1)$$

② ネットワークを構成するノードは、superpeer からベンチマーク用のデータを受け取る。そのデータを処理し、処理時間 t を測定する。

③ ノードのコンピュータの処理能力 a は、t_0 と t の比として（2）式で与える。

$$a = \frac{t_0}{t} \quad (2)$$

4.2. subpeer の処理能力

ある一定期間（T）に、異なる負荷分散によってノードが処理したタスク数から、求めたコンピュータの処理能力 a を分割する。これは、あるノードが子のノードが持つコンピュータの最大限使用できる処理能力を定めることを意図している。本研究で定義したコンピュータの処理能力を分割する式は、(3)式で与える。

$$sp = \frac{1}{m}a \quad (3)$$

sp は分割したコンピュータの処理能力を表し、m は T にノードが処理したタスク数を表す。また、sp が 3 章で述べた subpeer の処理能力となる。

4.3. 転送速度の定量化

データの転送速度は、データのサイズをその転送時間で割ることによって得ることができる。したがって、あるノードにおいて親のノードからのデータ転送速度 v_f[kB/s]から、（1）式で定めた v_d を用いて、そのノード間の転送速度の定量化が行える。ノード間のデータ転送能力 ta は、（4）式で与える。

$$ta = \frac{v_f}{v_d} \quad (4)$$

5. 負荷実行能力を表すランクの定義式

隣接ノードに負荷を分散させるにあたって、隣接ノードの子のノードの影響と、転送遅延の影響を考慮した負荷分散を行う必要がある。そのため、この様な影響を考慮したノードの負荷実行能力を表すランクを定義する。

負荷処理のネットワーク上のノード i を構成する subpeer の処理能力を sp_i とし、協調する隣接ノードの負荷実行能力も加味したノード i の処理能力を na_i とする。親のノード h とノード i 間のデータ転送能力を ta_{hi} とし、親のノード h からみた転送遅延を考慮したノード i の負荷実行能力を表すランクを r_{hi} とする。ノード i が指している隣接ノードの集合を S_i とおき、ノード i が負荷処理のネットワークにおいて、隣接ノードから指されているノード数を n_i とおく（だだし、ルートのノードについては、n_i=1）。ノードの処理能力 na_i の定義式を（5）式で与える。また、親のノード h から見た転送遅延を考慮したノード i の負荷実行能力を表すランク r_{hi} の定義式を（6）式で与える。

$$na_i = \frac{1}{n_i}(sp_i + \sum_{j \in S_i} r_{ij}) \quad (5)$$

$$r_{hi} = \frac{ta_{hi} \cdot na_i}{ta_{hi} + na_i} \quad (6)$$

（6）式から、ノード間の転送能力が十分に処理能力より大きければ、r_{hi}= na_i となり、ノード間の転送能力にノードの負荷実行能力が依存しない。また、逆にノー

ド間の転送能力が十分に処理能力より小さければ、r_{hi} = 0 となり、負荷実行能力は 0 となる。ゆえに、実際の状況に即した定義を表している。また、この様な（6）式でランク r_{hi} を定義する理由は後述の 7 章で述べる。

6. 負荷分割の方式

6.1. 分割の基準となる負荷のサイズの決定

分割の際の基準となる負荷のサイズ（F+ΔF）を決定する。F は分割しないで処理した方が効率的に行える負荷の最大データサイズとする。

L の生成の場合、トランザクション数（tid 数）と最小支持度（minsup）[%]で、効率的に処理が行えるデータサイズが定められる。少なくとも、tid 数×minsup/100 が 1 よりも大きくないと、アイテムセットの絞込みが全く行われないからである。そこで本研究では、F を tid 数が（7）式で与えられるデータサイズとする。

$$tid数 = 10000/\mathrm{minsup} \quad （7）$$

また、F+ΔF は分割しないで処理した時、処理時間が F の処理時間の 2 倍となるデータサイズとする。これから ΔF を定められる。

本研究の実験に用いたプログラムでは、データ量の小さな範囲において、データ量の 2 乗で処理時間が増加するのをプログラムの動作実験により確かめている。ゆえに、F+ΔF=1.414F より、ΔF を 0.414F とする。

6.2. 隣接ノードへの負荷の分割

負荷のデータ量とその処理時間は比例関係にはなっていない。しかし、本研究では次の方法で負荷のデータ量とその処理時間に比例関係を持たせている。

個々のノードに割り当てられた負荷を partition アルゴリズムによって処理する際、一定サイズ F（範囲としては F+ΔF 以下）の負荷を処理単位として均等に分割して処理することである。この時、その処理時間は F である負荷の処理時間に負荷の分割数を掛けた値となる。ゆえに、F のデータ量を基準としてデータ量と処理時間の間に比例関係を持たせられる。そのため、比例配分により負荷を分割することで、各ノードに割り当てられる負荷の処理時間の均一化が可能となる。

分割する負荷のサイズを s とおく。s が F+ΔF よりも大きければ負荷を分割し、子ノードへ転送する。この時、ノード i とその子ノード k に割り当てる負荷は、（8）式で与える。

$$\text{ノード i：} \frac{sp_i}{sp_i + \sum_{j \in S_i} r_{ij}} s \qquad \text{子ノード k：} \frac{r_{ik}}{sp_i + \sum_{j \in S_i} r_{ij}} s \quad （8）$$

6.3. ローカルラージアイテムセット(L)の生成を行う時の負荷分割数

ノードが partition アルゴリズムを用いて L の生成を行う時、分割した個々の負荷が F+ΔF 以下の範囲に収まる様に分割する。その方法は、分割する負荷のサイズを s_n とおくと、$s_n \div F$ を計算しその商 k を得た後、負荷を k 個に均等に分割する。ただし、$k(F+\Delta F) < s_n < (k+1)F$ なら、k+1 個に分割する。

7. ランク r_{hi} の定義式の導出

図 5 のような 2 つのノードからなるネットワークを考える。この時ノード 1 の持つ負荷のサイズを s_1 とおく。ノード 1、2 の subpeer の処理能力を sp_1、sp_2 とし、ノード 1 とノード 2 間のデータ転送能力を ta_{12} とおく。協調する隣接ノードの負荷実行能力も加味したノード 2 の処理能力を na_2 とし、ノード 1 からみたノード 2 の負荷実行能力を表すランクを r_{12} とする。ノード 1、2 に割り当てた負荷のサイズを x_1、x_2 とする。（8）式から、x_1、x_2 は（9）式で与えられる。

$$x_1 = \frac{sp_1}{sp_1 + r_{12}} s_1 \qquad x_2 = \frac{r_{12}}{sp_1 + r_{12}} s_1 \quad （9）$$

ノード 1、ノード 2 の負荷の処理時間を t_1、t_2 とおく。t_1 はノード 1 に割り当てた負荷の処理時間であり、t_2 はノード 1 から 2 への負荷の転送時間にノード 2 に割り当てた負荷の処理時間との和になるから、（10）式と与えられる。

$$t_1 = \frac{x_1}{sp_1 \cdot v_d} \qquad t_2 = \frac{x_2}{sp_2 \cdot v_d} + \frac{x_2}{ta_{12} \cdot v_d} \quad （10）$$

（9）、（10）式と処理時間が等しいという条件から、（11）式が導ける。

$$\frac{1}{sp_1 \cdot v_d} \frac{sp_1}{sp_1 + r_{12}} s_1 = \left(\frac{1}{sp_2 \cdot v_d} + \frac{1}{ta_{12} \cdot v_d} \right) \frac{r_{12}}{sp_1 + r_{12}} s_1 \quad （11）$$

したがって（11）式から、（12）式を得る。（5）式より $na_2 = sp_2$ が得られるので、r_{12} は、（13）式となり、ランクの定義式が導ける。この様な導出により、5 章で（6）式のように定義した。

$$1 = \left(\frac{1}{sp_2} + \frac{1}{ta_{12}} \right) r_{12} \quad （12）$$

$$r_{12} = \frac{ta_{12} \cdot na_2}{ta_{12} + na_2} \quad （13）$$

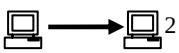


図 5　想定したネットワーク

8. 評価実験

本提案の有効性を確認するために、評価実験として、LAN によってネットワークを構成し、アソシエーションルールの抽出を行った。その際、使用する評価実験用のプログラムは、partition アルゴリズムに基づいて作成した。評価実験では、tid 数が 5M（5,000,000）、

表 1　ノードのコンピュータの処理能力 a_i、転送能力 ta_{hi}

ノード番号	1	2	3
CPU の種類と速度	pentium M 2GHz	pentium4 2.20GHz	pentium3 996MHz
メモリ容量	1GB	1GB	512MB
処理時間[s]	1.34	4.04	11.58
a_i	7.46	2.48	0.86
転送時間[s]		1.39	2.07
ta_{hi}		78.5	52.8

4	5	6	7
pentium4	celeron	pentium3	pentium3
1.5GHz	499MHz	996MHz	996MHz
512MB	64MB	256MB	192MB
7.43	21.17	9.67	11.67
1.35	0.47	1.03	0.86
9.06	10.2	10.4	5.00
12.1	10.7	10.6	22.1

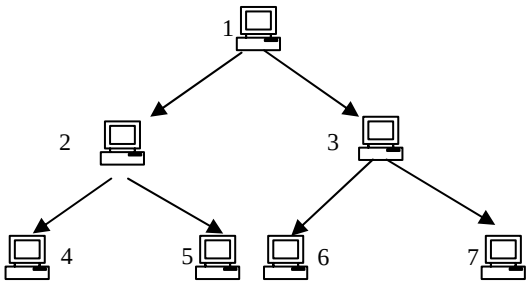


図 6　superpeer が管理しているネットワーク全体

表 2　subpeer の処理能力 sp_i、隣接ノードの負荷実行能力を加味したノード i の処理能力 na_i、親のノードからみた転送遅延を加味したノード i の負荷実行能力 r_{hi}

（a）　転送回数が 1 回の場合

ノード番号	1	2	3
subpeer の数	1	2	2
sp_i	7.46	1.24	0.43
na_i	9.11	1.24	0.43
r_{hi}		1.22	0.43

（b）　転送回数が 2 回の場合

ノード番号	1	2	3
subpeer の数	1	2	2
sp_i	7.46	1.24	0.43
na_i	10.8	2.11	1.34
r_{hi}		2.05	1.31

4	5	6	7
2	2	2	2
0.67	0.24	0.52	0.43
0.67	0.24	0.52	0.43
0.64	0.23	0.49	0.42

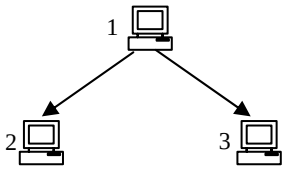


図 7　転送回数が 1 の時の負荷処理のネットワーク

10M（10,000,000）の 2 つのトランザクションデータベースを生成した。また、アイテム数は 50、平均アイテム長は 5 とした。このトランザクションデータベースそれぞれについて、最小支持度を 1%、最小確信度を 10%とした場合のマイニング処理を行った。この時、F は tid 数が 10000 であるトランザクションデータベースのデータサイズとなる。

8.1. 計算機の処理能力、転送能力の定量化

実験に使用した 7 台のコンピュータの性能を定量化するために、ベンチマーク用のデータ処理として tid 数が 10000 のトランザクションデータベースについて、L の生成を行わせた。この時、他の条件は上述の条件と同じにしている。この負荷のデータサイズは 237[kB]であり、その処理時間が 10 秒を定量化の基準とすると、基準となる処理速度 v_d は v_d=23.7[kB/s]と与えられる。

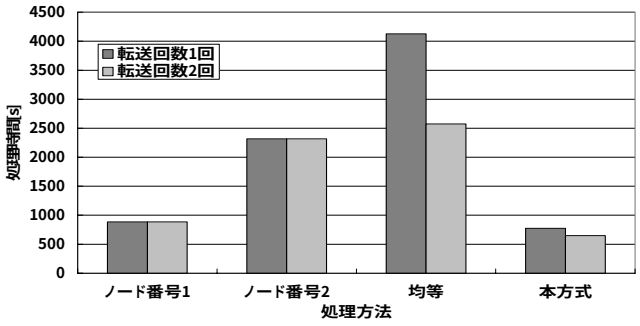
さらに、大きさ 2.59[MB]のデータの転送時間から、コンピュータの転送速度を求め、定量化を行った。この時の転送速度は、図 6 のネットワークにおいて、各ノードの親のノードからの負荷の転送速度である。

表 1 にノードのコンピュータの処理能力 a_i、転送能力 ta_{hi} の値を示す。また、表 1 に L の生成の処理時間、データの転送時間も併せて示す。ここで、表 1 のノード番号は、図 6 のノード番号と対応している。

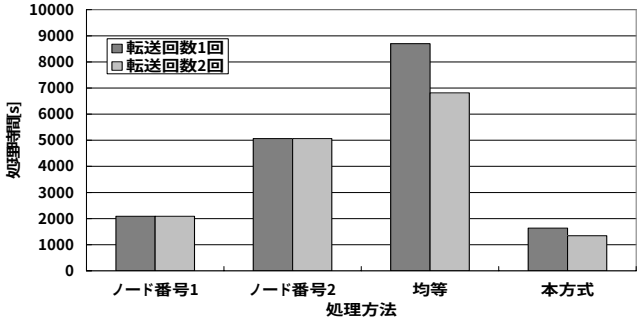
8.2. 評価実験

評価実験を行うにあたって、superpeer が管理しているネットワークの構造（ネットワークを構成するノード全体）を図 6 に示すツリー構造であるとする。評価実験として、本方式と均等に負荷を分散した場合、及び、表 1 のノード番号 1 の PC（表 1 で最も処理速度の速い PC）と表 1 のノード番号 2 の PC（表 1 で平均の能力を持つ PC）による partition アルゴリズムを用いた処理との比較を行った。その際、本方式と均等に負荷を分散した場合のそれぞれについて、転送回数が 1 回の場合（図 7 のネットワーク）と転送回数が 2 回の場合（図 6 と同等のネットワーク）で処理を行った。処理結果を示したのが図 8 で、この時の構成ノード i の subpeer の処理能力 sp_i、隣接ノードの負荷実行能力を加味したノード i の処理能力 na_i、親のノードからみた転送遅延を加味したノード i の負荷実行能力 r_{hi} と想定した subpeer の数を示したのが表 2 である。また、表 2 のノード番号は、図 6、7 のノード番号と対応している。この処理結果から、1 台の PC で処理した場合と比較して本方式の処理時間の減少率を示したのが表 3 である。

図 8 から、処理速度の最も速い PC と比較した場合



（a）　tid 数=5M



（b）　tid 数=10M

図 8　処理方式の違いによる処理時間の比較

表 3　1 台の PC と比較した時の処理時間の減少率

（a）　ノード番号 1 の PC と比較

tid 数	5M		10M	
転送回数	1	2	1	2
減少率[%]	12.2	26.7	21.8	35.8

（b）　ノード番号 2 の PC と比較

tid 数	5M		10M	
転送回数	1	2	1	2
減少率[%]	65.5	72.0	67.7	73.5

でも、本方式を用いることにより処理時間を減少させることができる。また、処理速度の遅い PC ほど本方式を用いることで処理の効率化が行える。これは、ネットワーク全体の能力を PC の能力と比較した場合に、PC の能力が小さいほど、相対的な能力の格差は大きくなるために、ネットワークを用いた時の処理効率が良くなることに因る。

表 3 から、転送回数を 1 から 2 へネットワークの規模を大きくすると、処理時間の減少率がノード番号 1 の PC と比較した場合には 14%大きくなり、ノード番号 2 の PC と比較した場合には 6%大きくなる。ゆえに、ネットワークの規模が大きくなると、処理効率が良くなることが確かめられる。この処理効率の良くなる割合は、ノードがネットワークに加わることで、ネットワーク全体の能力の増加分に依存して定まる。

図 8 から、均等に負荷を分散させると、コンピュータの処理能力が小さいノードに、過負荷が割り当てられる場合があり、結果として 1 台の PC で処理した時と比べて、処理が遅くなる場合もある。

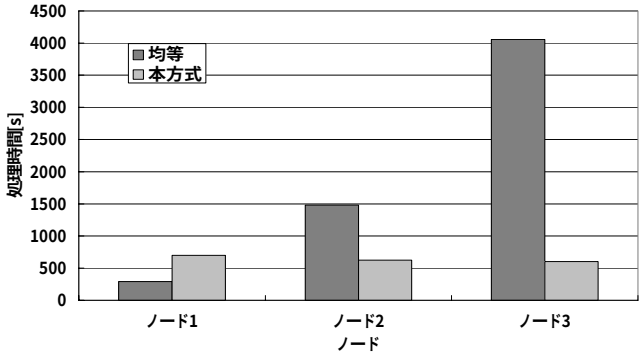
tid 数が 5M、10M のトランザクションデータベースそれぞれについて、図 6、図 7 のネットワークでの本方式と均等に負荷を分散させた場合の各ノードの負荷の処理時間を示したのが図 9、図 10 である。ただし、各ノードの実質的な負荷処理時間を比較するために、ノード 1～3 の負荷の処理時間は、子のノードの処理結果の転送待ちの時間を除いてある。また、ノード 1 の負荷の処理時間に関しては、アソシエーションルールの抽出時間も除いてある。

本方式による処理を行うと、tid 数=5M の場合、図 9 から、転送回数が 1 回の時は、各ノードの負荷の処理時間は 641.3±60 秒の範囲にあり、転送回数が 2 回の時は、各ノードの負荷の処理時間は 551.9±40 秒の範囲にある。同様に、tid 数=10M の場合、図 10 から、転送回数が 1 回の時は、各ノードの負荷の処理時間は 1281.1±200 秒の範囲にあり、転送回数が 2 回の時は、各ノードの負荷の処理時間は 1065.4±120 秒の範囲にある。このことから、本方式によって負荷を分散した場合、各ノードの負荷の処理時間が均一化されていることが確かめられる。また、深さが 2 のツリー構造のネットワークにおいても成立することから、それ以上の深さを持つツリー構造のネットワークでも、各ノードの負荷の処理時間が均一化されると期待できる。

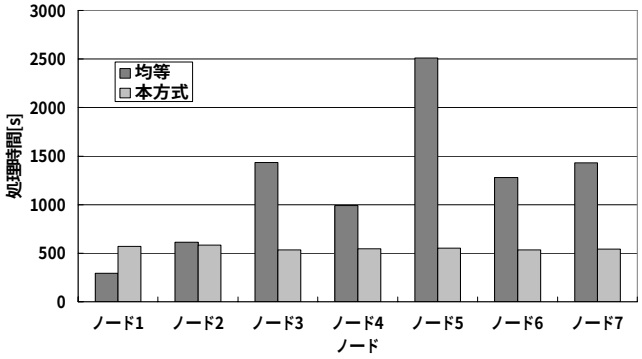
本方式を用いた時、均一化される各ノードの負荷の処理時間の誤差の原因としては 2 つ考えられる。第 1 には、各ノードでの処理単位の誤差による。本方式はネットワークの全てノードで割り当てられた負荷を一定サイズ F で分割して処理が行われた場合に理想的な均一化がされると想定している。しかし、実際には各ノード事にサイズ F+ΔF 以下の負荷に分割して処理が行われるために、処理単位の誤差による影響が、まとまった数の分割された負荷を処理することにより表れるからである。第 2 には、各ノードのコンピュータの処理能力、転送能力の定量化の際の誤差による。コンピュータの性能を正確に定量化するのは、困難であり、転送能力はネットワークの回線の状況によって変化するため、負荷分散の時、実際の状況とは異なっている場合もあるからである。

9. まとめ

本研究では、協調する隣接ノードの負荷実行能力と負荷の転送遅延を考慮して、各ノードの処理能力に応じて負荷を分散させるための、負荷実行能力のランク付けの方式を提案した。また、併せて負荷処理を行うネットワークの構成の方式を示した。

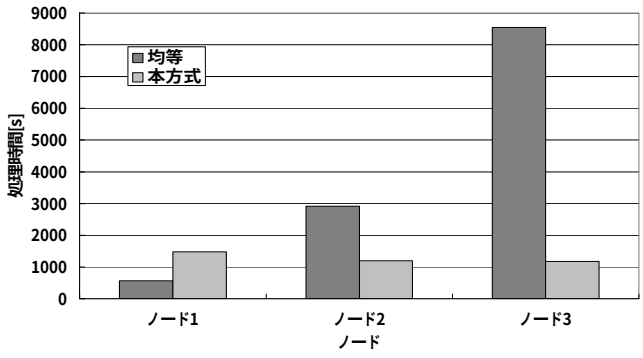


（a） 転送回数が 1 回の場合

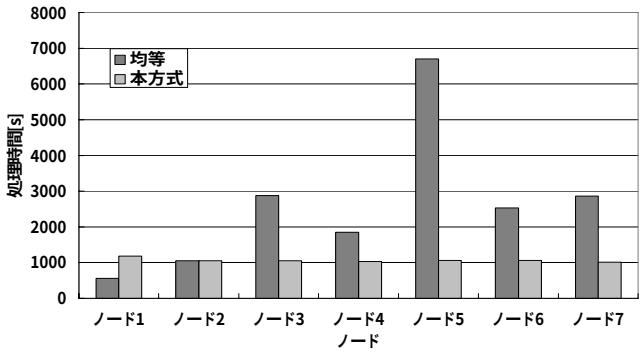


（b） 転送回数が 2 回の場合

図 9 各ノードの負荷の処理時間（tid 数=5M）



（a） 転送回数が 1 回の場合



（b） 転送回数が 2 回の場合

図 10 各ノードの負荷の処理時間（tid 数=10M）

評価実験の結果により、1 台の PC で処理するよりも、本方式に基づいて分散処理した方が処理時間を短縮することができ、効率的に処理できることが確認できた。加えて、ネットワークの規模が大きくなると、処理時間が短縮されることを確かめた。また、各ノードの負荷の処理時間を均一にできることが確認できた。

本研究の今後の課題を以下に挙げる。

- LAN よりも転送速度の遅い、一般的な回線（ADSL 等）でも本方式が成立するかを確認する。
- 負荷処理が多数発生する場合に、superpeer がランクを計算中に、ノードが別の負荷処理を請け負うことによるランクの変化を回避するための仕組みを検討する。
- 汎用性の確認のために、異なるアプリケーションにおいても本方式が有効であるかを確認する。
- 大規模ネットワークにおいて、多数のタスクを処理する場合に、全てのタスクの作業効率について検討する。
- ピアの特性（メモリが多い、画像処理が強い等）を考慮した負荷分散方式を検討する。
- ネットワークの状態や各ノードの状態が動的に変化した場合における負荷分散方式（動的な負荷分散方式）を検討する。

謝辞

本研究の一部は，文部科学省科学研究費補助金特定領域研究(2)(課題番号:16016273)による。

文 献

[1] 塩谷康夫, 太田学, 片山薫, 石川博:P2P における効率的な負荷分散方式の提案, 情報処理学会第 66 回全国大会, 2004. 3

[2] 塩谷康夫, 太田学, 片山薫, 石川博:P2P における静的負荷分散方式の提案, 電子情報通信学会, 情報処理学会, 夏のデータベースワークショップ DBWS2004, 2004. 7

[3] R.Agrawal and R.Srikant: Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Databases,Santiago, Chile, 1994.

[4] A.Sarasere, E.Omiecinsky, and S.Navathe: An efficient algorithm for mining association rules in large databases. In 21st International Conference on 1 Very Large Databases, Zurich, Switzerland, 1995.

[5] SETI@home,http://setiathome.ssl.berkeley.edu/

[6] L.Galanis Y.Wang S.R.Jeffery and D.J.DeWitt: Locating data sources in large distributed systems In 29th International Conference on Very Large Databases, Berlin, Germany, 2003.

[7] 合田和生, 田村孝之, 小口正人, 喜連川優: SAN 結合 PC クラスタ上での動的資源割り当てを用いた並列データマイニング処理, 電子情報通信学会第 13 回データ工学ワークショップ DEWS2002, 2002. 3