P2P における静的負荷分散方式の提案

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

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、負荷処理時間の最大値に全体の処理時間が依存してしまう。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と期待できる。そこで本稿では、各ノードの処理能力を示す指標としてランクを定義し、各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 Proposal of a Static Load Distribution Technique on P2P Network

Yasuo SHIOYA†　Manabu OHTA†　Kaoru KATAYAMA†　and　Hiroshi ISHIKAWA†
†Graduate School of Engineering, Tokyo Metropolitan University

Abstract　In distributed computing, when a load is distributed uniformly , the processing time varies with node, and the total processing time will be dependent on the maximum processing time on each node. Therefore distributing load according to the capability of each node can make us expect more efficient total processing by equalization of load processing time on each node. This paper defines a “rank” as a metric which indicates the throughput of each node, and shows how to determine the “rank” of each node from “ranks” of adjoining node. Moreover, this paper proposes a technique to distribute load based on a “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。我々は、P2P ネットワーク上でのデータマイニングに、この負荷分散方式を適用する。

以前、我々が行った研究[1]では、深さが 2 以上のツリー構造において、ネットワークの各ノードの負荷処理が均一な処理時間で行われなかった。そこで、本研究は、任意のツリー構造においても、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。また、負荷分割の方式についても説明する。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の抽出のシミュレーションを行った。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[2]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 [3]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、過負荷による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。

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、ネットワークの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。

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 supeer が、図 1 に示すように、負荷を分散させて管理することで、ネットワークを安定性を増加させ、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の構成の時の、負荷実行能力を表すランクの計算を行うことである。

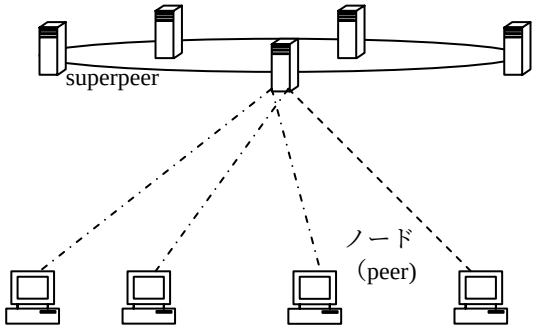


図 1　superpeer の役割

3.2. peer

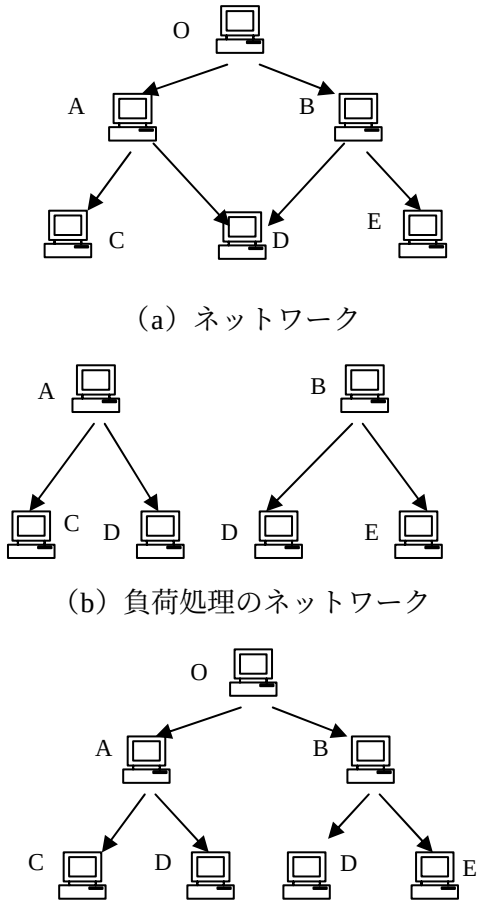
peer は、subpeer と一対多の関係になっている。peer が subpeer を内包しているからである。内包する subpeer は、superpeer が管理しているネットワークにおいて、隣接ノードから指されているノード数に依存して変化する。

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、2 つある。1 つには、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、その個々の処理の影響が相互に関与しないで、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 から異なるタスクに関し、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。もう 1 つには、同一のタスクの負荷分散による処理において、1 つのノードが複数のノードからの負荷処理を行う場合にも、その個々の処理が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)において、ノード O から、負荷処理のネットワークを構成したとする。この時、ノード O からの同一のタスクについて、ノード A、B から依頼された負荷処理をノード D が行う時、仮想的に、図 2 の(c)のような負荷処理のネットワークで処理を行うとする役割である。

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、そのノードを指している隣接ノードごとに、個々に割り当てることができる。



（a）ネットワーク

（b）負荷処理のネットワーク

(c)　仮想的な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。

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 T）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負荷処理を行うか問い合わせる。

④ ③を転送回数が T 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。

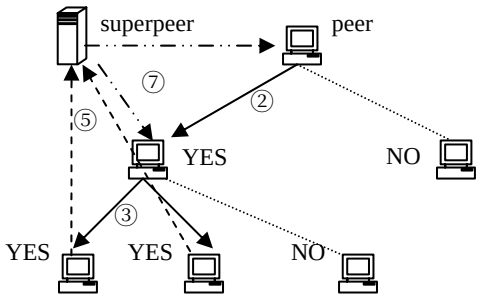


図 3 負荷処理のネットワークの構成

3.5. subpeer 情報の更新

subpeer の情報を持つノードは、subpeer の情報が更新されると、superpeer にその情報を転送する。subpeer の情報が更新されるのは、ノードの実ネットワークにおける隣接関係やコンピュータの性能が変化した場合である。

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

荷の分割をし、隣接ノードに転送する。転送後、自身に割り当てられた負荷の処理を行う。

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。転送後、自身に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。

④ 末端のノードで、割り当てた負荷の処理を行う。

⑤ 処理結果をルートに転送する。

⑥ 中継ノードで、処理結果を統合する。

⑦ ルートは、処理結果を統合する。

⑧ 統合した結果より、最終結果を得る。

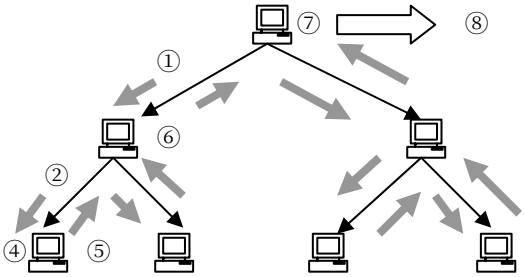


図 4　負荷分散の処理の流れ

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によって決まっている。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、本研究における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の処理時間 t_0 を superpeer に設定する。

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り、処理を行う。その処理時間 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 の比として次式のようにして与える。

$$a = \frac{t_0}{t} \quad (1)$$

4.2. subpeer の処理能力

ネットワークの隣接関係から、定めたコンピュータの処理能力 a を分割する。これは、あるノードから見た隣接ノードのコンピュータの通常最低限使用できる処理能力を定めることを意図している。すなわち、ノードが隣接ノードに負荷処理を行わせる時、少なくとも分割した処理能力によって処理が実行できるようにするためである。本研究で定義したコンピュータの処理能力の分割の式は、

$$sp = \frac{1}{n+1}a \quad (2)$$

で、与える。sp は分割したコンピュータの処理能力を表し、n は、superpeer が管理しているネットワークにおける隣接ノードから指されているノード数（隣接ノードからの負荷処理の依頼を受けるノード数）を表す。よって、(2) 式の分母は、そのノードの負荷処理の最大タスク数を意味する。また、sp が 3 章で述べた subpeer の処理能力となる。

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当たって、隣接ノードの子のノードの影響を考慮した負荷分散を行う必要がある。そのために必要な子のノードの影響を加味したノードの負荷実行能力を表すランクを定義する必要がある。

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処理能力も加味したノード i の負荷実行能力を表すランクを r_i 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 m_i とおく（ただし、ルートのノードにつては、m_i=1）。

p_iはノード i における負荷分割数とする。負荷分割による重みを $w(p_i)$とする。ノードの負荷実行を表すランクの定義式を

$$r_i = w(p_i)(sp_i + \frac{1}{m_i}\sum_{j \in S_i} r_j) \quad (3)$$

で与える。このランクは、あるノードが、隣接ノードの子のノードの影響まで含めた、通常使用するのに期待できる負荷実行能力を表している。ここで、(3) 式の $1/m_i$ は、複数の隣接ノードから転送されてきた負荷を、さらにその子のノードに分割して転送すると、個々の負荷に割り当てられる処理能力は、$1/m_i$ になること考慮した項である。また、$w(p_i)$は、各ノードに割り当てられる負荷それぞれについて、その計算コストの比を取ると、各ノードの subpeer の処理能力の比になる様にするための項である。$w(p_i)$の決定方法は、7 章で説明する。

6. 負荷分割の方式

あるデータを処理する時、その計算コストをコンピュータの処理速度で割ると処理時間が定まるとする。すると、負荷分散の際、各ノードの処理時間を均一化するには、計算コストをランクの比になるようにデータを分割すればよい。そのために、データ量と計算コストの間の関数 g を求める。逆関数 g^{-1} をランクに適用すると、計算コストでみた時にランクの比を与える、データ量の比を与える。そのため、逆関数 g^{-1} を用いて負荷を比例分割することで処理時間を均一にできる。このことから、一般式は、分割する負荷のサイズを s とおくと、ノード i とその子ノード k に割り当てる負荷は、(4)、(5) 式で与えられる。

ノード i：
$$\frac{g^{-1}(sp_i)}{g^{-1}(sp_i) + \sum_{j \in S_i} g^{-1}\left(\frac{r_j}{m_i}\right)} s \quad (4)$$

子ノード k：
$$\frac{g^{-1}\left(\frac{r_k}{m_i}\right)}{g^{-1}(sp_i) + \sum_{j \in S_i} g^{-1}\left(\frac{r_j}{m_i}\right)} s \quad (5)$$

本研究で使用したシミュレーションの範囲では、データ量の 2 乗に比例して計算コストが増加する。(4)、(5) 式より、ノード i とその子ノード k に割り当てる負荷は、(6)、(7) 式で与えられる。

ノード i：
$$\frac{\sqrt{sp_i}}{\sqrt{sp_i} + \sum_{j \in S_i} \sqrt{\frac{r_j}{m_i}}} s \quad (6)$$

子ノード k：
$$\frac{\sqrt{\frac{r_k}{m_i}}}{\sqrt{sp_i} + \sum_{j \in S_i} \sqrt{\frac{r_j}{m_i}}} s \quad (7)$$

7. 転送先の分割による影響 $w(p_i)$の決定

ネットワークの各ノードについて、負荷の処理時間を均一化するには、全負荷を各ノードの subpeer の処理能力に従って分割した場合の負荷を、各ノードに割り当てる必要がある。これを実現するためには、ある中継ノードにおいて転送されてくる負荷量が、subpeer の処理能力に従って割り当てた場合の、そのノードと子のノードに割り当てる負荷量の和になる必要がある。このことより、$w(p_i)$を決定する条件が与えられる。

簡単のために、図 5 のネットワーク構造について、$w(p_i)$を定める。定めた $w(p_i)$が図 2 の (a) のような構造にも適用できる。

図 5 のノード o、i の subpeer の処理能力を sp_o、sp_i とし,指している隣接ノードのランクを r_k ($k=1 \sim p_i-1$) とし、負荷分割の数を p_i とおく。ノード o における負荷のサイズを s とおく。この時、ノード o から負荷処理を開始したとすると、ノード i に割り当てられる負荷量は、(8) 式で与えられる。図 5 の場合には、ノー

ド i の子のノードのランクは、subpeer の処理能力に等しい。よって、subpeer の処理能力に従って分割した場合のノード i とその子のノードの負荷量の和が、（9）式で与えられる。

$$\frac{g^{-1}\left(w(p_i)(sp_i+\sum_{k=1}^{p_i-1}r_k)\right)}{g^{-1}(sp_o)+g^{-1}\left(w(p_i)(sp_i+\sum_{k=1}^{p_i-1}r_k)\right)}s \quad (8)$$

$$\frac{g^{-1}(sp_i)+\sum_{k=1}^{p_i-1}g^{-1}(r_k)}{g^{-1}(sp_o)+g^{-1}(sp_i)+\sum_{k=1}^{p_i-1}g^{-1}(r_k)}s \quad (9)$$

（8）、（9）式が等しくなることから、$w(p_i)$を満たすべき式は、

$$g^{-1}\left(w(p_i)(sp_i+\sum_{k=1}^{p_i-1}r_k)\right)=g^{-1}(sp_i)+\sum_{k=1}^{p_i-1}g^{-1}(r_k) \quad (10)$$

が、任意の p_i 、sp_i、r_k について成立することである。しかしながら、任意の p_i 、sp_i、r_k について、(10)式を満たす $w(p_i)$を定めるのは困難である。そこで今回は、$sp_i=r_k=r$ $(g^{-1}(r)=1)$ となる sp_i、r_k における $w(p_i)$を求め、任意の状態に成立すると仮定した。よって、（10）式より、（11）式で与えられる。

$$w(p_i)=\frac{g(p_i)}{p_i r} \quad (11)$$

本研究のシミュレーションの場合、データ量の 2 乗に比例して計算コストが増加するから、$w(p_i)$が満たす式は、（12）式で与えられる。

$$w(p_i)=p_i \quad (12)$$

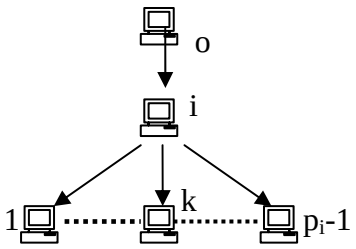


図 5 想定した負荷処理のネットワーク

8. シミュレーション

本提案の有効性を確認するために、シミュレーションとして、アソシエーションルールの抽出を行った。その際、使用するシミュレーション用のプログラムは、partition アルゴリズム[4]に基づいて作成した。シミュレーションでは、トランザクション数が 150000、アイテム数が 50、平均アイテム長が 5 のトランザクションデータベースを生成し、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。

8.1. コンピュータの処理能力の定量化

シミュレーションに使用した 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、トランザクション数が 50000 の時のローカルラージアイテムセット L の生成を行わせた。この際、他の条件は上述の条件と同じにしている。この時、処理時間が 400 秒を定量化の基準としている。表 1 にベンチマーク用データの処理時間を示し、表 2 に定量化した値を示す。ここで、表 1、2 のノード番号は、図 6、7 のノード番号と対応している。

8.2. シミュレーション

シミュレーションを行うにあたって、superpeer が管理しているネットワークの構造を図 5 に示すツリー構造であるとする。この時、ノード 1 から転送回数が 3 回の場合の負荷処理のネットワークを構成し、処理を行ったとする。その際、ネットワークの状況に変化がなかったとすると、構成される負荷処理のネットワークは図 7 のとなる。各ノードは、表 1 で、PAi(i はノード番号)は、データ分割の処理時間、Li は各ノードでの L の生成の処理時間、Ji は自分自身と子のノードで生成された L の結合時間、A はノード 1 での子のノードで生成された L の結合によるアソシエーションルールの抽出時間を表している。また、T は、A と Li の最大値に、ノード 1 から Li の最大値とな

表 1 ベンチマーク用データの処理時間

ノード番号	CPU の種類と速度、メモリの容量			処理時間[s]
1	celeron	466MHz	256M	386.7
2	pentium4	1.7GHz	768M	158.1
3	MMXpentium	233MHz	96M	266
4	pentium4	2.19GHz	1024M	84.5
5	celeron	499MHz	64M	218.4
6	pentium3	996MHz	512M	216.7
7	pentium3	996MHz	192M	361.1

表 2 ノードのコンピュータの処理能力 a_i、subpeer の処理能力 sp_i、負荷実行能力 r_i

ノード番号	1	2	3	4	5	6	7
a_i	1.03	2.53	1.50	4.73	1.83	1.85	1.11
sp_i	1.03	0.84	0.50	2.37	0.61	0.92	0.55
r_i	76.66	14.96	9.60	2.37	1.77	0.92	0.55

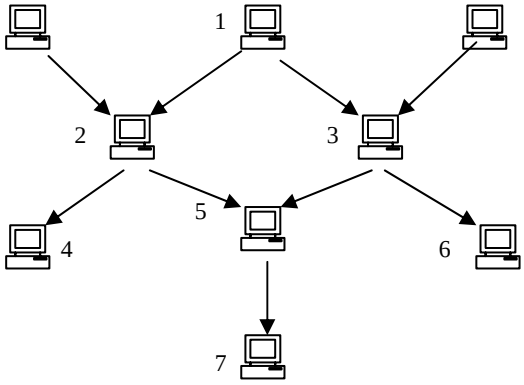


図 6　superpeer が管理しているネットワーク全体

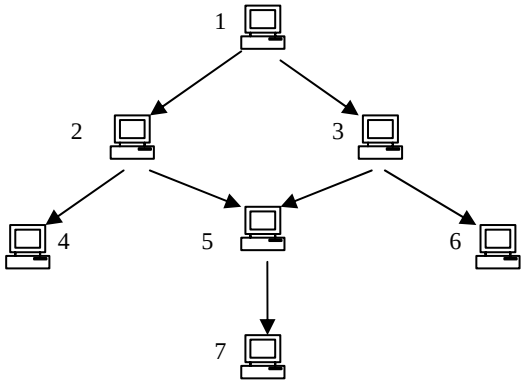


図 7　ノード 1 における負荷処理のネットワーク

るノード間で行う PAi の処理時間と、Ji の結合時間を加えた全体の処理時間を表している。ただし、PAi(j)は、ノード i におけるノード j からの負荷の分割時間、Li(j) は、ノード i におけるノード j からの負荷についての L の生成時間、Ji(j)は、ノード i とその子のノードにおいて、ノード i におけるノード j からの負荷により生成された L の結合時間を表す。

この処理結果から、ランクに基づいて分割した場合に L の生成時間がほぼ均一化されている。よって、L の生成時間の減少により、均等に分割した場合（409.1 秒）よりも、ランクに基づいて分割した場合（119.1 秒）の方が全体の処理時間が 290 秒減少する。

このことから、均等に分割して処理するよりも、ランクに基づいて分割して処理する方が、各ノードで L の生成時間が減少することにより、全体の処理時間が減少し、処理の効率化が成されることが確認できた。

また、ランクに基づいて分割した場合、各ノードの L の生成時間の平均値は、62.9 秒で、その範囲は±20 秒である。L の生成時間のば

らつきの原因は、主に、コンピュータの性能の定量化の誤差と、$w(p_i)$として、近似値を用いたことによる誤差と考えられる。

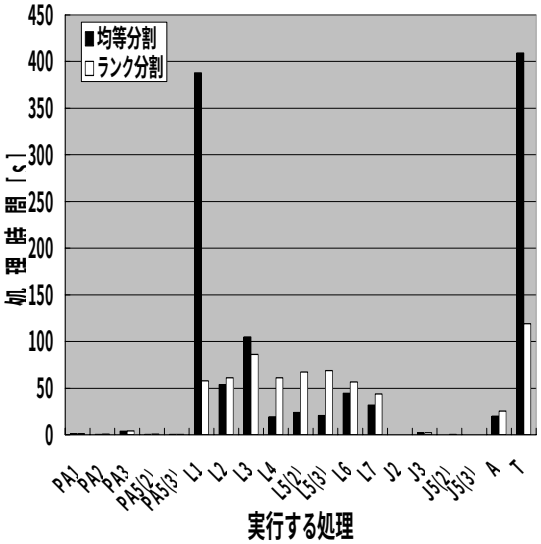


図 8 処理時間の比較

9. まとめ

本研究では、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と、負荷処理を行うネットワークの構成の方式を提案した。シミュレーションの結果により、均等に分割するよりも、ランクに基づいて分割した方が処理時間を短縮することができ、効率的に処理ができることが確認できた。また、深さが 2 以上のツリー構造の場合にも、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。

大規模な負荷になると、負荷分散の際の転送時間の影響が大きくなる。ゆえに、効率的に処理をするには、転送時間を考慮して負荷を分散させる必要がある。転送環境を考慮した負荷分散の方式の検討が今後の課題となる。また、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、すなわち、動的な負荷分散方式の検討も今後の課題となる。

謝辞

本研究の一部は, 文部科学省科学研究費特定領域研究(2)[情報学:A02](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] SETI@home,http://setiathome.ssl.berkeley.edu/

[3] 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.

[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.