社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－OS－103 (1)
2006／7／31

PeerCQにおける効率的な条件付き情報監視手法

圓山康太　田頭茂明　藤田聡

本稿では，PeerCQ に基づいた新しい P2P 型の情報監視システムを提案する．提案システムでは，AND 条件をサポートする情報監視に着目し，そのような情報監視を効率的に実現する枠組みを提供する．具体的には，AND 条件をサポートするために，PeerCQ に以下の拡張を行った．1) 与えられた AND 条件の成立を確認するエージェントとして，Conjunctive Continual Query (CCQ) を導入し，その詳細な仕様を提供した．2) 監視するデータ項目の類似性を考慮した CCQ のピアへの割り当て手法を提案した，3) 各ピアの負荷とシステム全体の負荷との間のトレードオフを制御できるように，PeerCQ において使われるユーティリティ関数を改良した．提案手法の効果をシミュレーションにより評価した．シミュレーションの結果，提案するユーティリティ関数によりシステムの性能は向上し，性能のトレードオフは関数内のパラメータを調整することにより制御できることを示した．

An Efficient Information Monitoring Technique Supporting Conjunctive Queries based on PeerCQ

Kouta Maruyama　Shigeaki Tagashira　Satoshi Fujita

In this paper, we propose a P2P information monitoring system that supports the processing of conjunctive queries. The proposed system is an extension of PeerCQ, and our main contribution in this paper is the following three points: 1) we provide a detailed specification of Conjunctive Continual Queries (CCQs) that is an agent to check the satisfaction of a given conjunction, 2) we propose a mapping of CCQs to peers by considering the similarity of the referred data items, and 3) we improve the utility function used in PeerCQ in such a way that the trade-off between the load of each peer and the overall load of the system can be explicitly controlled. The effect of the proposed method is evaluated by simulation. The result of simulation indicates that the proposed utility functions certainly improves the performance of the resultant scheme, and the trade-off can be controlled by tuning a parameter used in the utility function.

1　はじめに

近年，世界規模の情報共有システムであるインターネットにおいて，オンライン型の株式売買，ショッピング，オークションなどに代表される多くの魅力的なサービスが実現されている．このようなネットワークサービスにおける重要な技術のひとつとして，共有資源の更新情報を追跡する技術が近年高い注目を集めている．このような技術を用いることで，複数の情報ソースに含まれるデータ項目 (複数のマーケットにおける特定製品の価格や入手の可能性など) の更新を追跡し，ある与えられた条件が成立した場合に，ユーザーに通知することが可能になる．例えば，オンラインショッピングの顧客が，ある店での製品の価格が閾値より安くなった時に，自動的にメッセージを受け取ることができると非常に便利である．また，そのような機構は，スケーラビリティを確保するために，サーバーを使わずに分散的に処理されることが望まれている。

Scribe[5] は，発行／購読モデルに基づいた代表的な P2P 型のイベント通知システムである．P2P プロトコルとして Pastry[6] を採用し，各購読トピックに対して，購読者で構築されたアプリケーションレベルマルチキャスト木を通じて，イベントの通知メッセージがブロードキャストされる．トピックをピアにマップするためにはトピック識別子が使用される．また，各トピックのためのグループ通信を管理する rendezvous points を発見するために，Pastry の発見アルゴリズムが利用される．PeerCQ[1, 2] は，Scribe とは異なる P2P 型の情報監視システムである．購読トピックがより抽象的に定義されるという点で，Scribe よりも一般的である．PeerCQ においては，情報

広島大学大学院工学研究科情報工学専攻
〒 739-8527 東広島市鏡山 1-4-1
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527 Japan

監視要求の処理は第2章で述べる Continual Query (CQ) の形で実施される．PeerCQ は，WebCQ[3, 4] を P2P モデルに拡張したシステムである．WebCQ からの主な拡張点としては、"service partitioning" をあげることができる．ピアへの CQ の割り当てにおいて，この拡張を利用することにより、PeerCQ は，ヘテロな環境を考慮しながら，システム全体の負荷を抑えて各ピアの負荷の均一化を達成している．

本稿では，PeerCQ をベースにした新しい P2P 型の情報監視システムを提案する．提案システムでは，AND 条件をサポートする情報監視に着目し，そのような情報監視を効率的に実現する枠組みを提供する．AND 条件をサポートすることの重要性は，巨大なデータベースからある必要な情報を発見する状況を考えれば分かり易い．例えば，SQL のようなデータベース言語では，連言操作を主要な操作の一つとしてサポートしている．また，Google のようなキーワード検索では，目的のページを素早く発見するために，AND 条件としてキーワードの連言を入力することが一般的となっている．提案システムでは，AND 条件をサポートするために，PeerCQ に以下の拡張を行った．1) 与えられた AND 条件の成立を確認するエージェントとして，Conjunctive Continual Query (CCQ) を導入し，その詳細な仕様を提供した．2) CQ の類似性を考慮した CCQ のピアへの割り当て手法を提案した．3) 各ピアの負荷とシステム全体の負荷との間のトレードオフを制御できるように，PeerCQ において使われるユーティリティ関数を改良した．提案手法の効果はシミュレーションにより評価した．シミュレーションの結果，提案するユーティリティ関数によりシステムの性能は向上し，性能のトレードオフは関数内のパラメータを調整することにより制御できることを示した．

2 PeerCQ

2.1 概要

PeerCQ は，2003 年に Gedik と Liu らによって提案された P2P 型の情報監視システムである．PeerCQ における情報監視は，Continual Query (以下，CQ と呼ぶ) に基づいて実行される．すなわち，情報ソースに含まれるデータ項目の状態を確認するために，その情報ソースに対してメッセージを継続的に出している．CQ はユーザから要求を受けたピアによって生成される自律的なエージェントであり，CQ は適切なピアに分散的に割り当てられる．図 1 は PeerCQ の概観を簡単に表している（文献 [2] から引用）．この図では、CQ を割り当てられたピア B は，更新を検知した後に、ピア A に対して直接の通信か e-mail を利用してその事を通知している様子を示している．

PeerCQ において，CQ のピアへの割り当ては，識別子

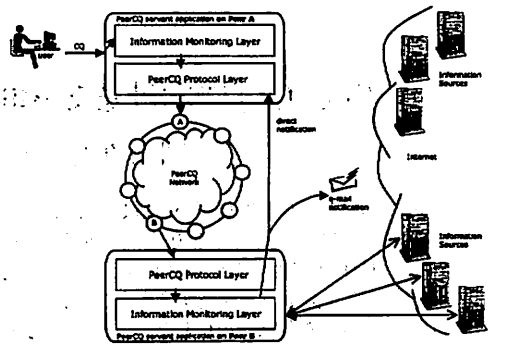


図 1: PeerCQ の概観．

(ID) である符号無し m ビット整数を利用することで行われる．各 CQ には 1 つの ID が割り当てられ，また，各ピアに対しても少なくとも 1 つの ID が与えられる．ここで，ピア間で同一の ID が割り当てられないものとし，また，ピアに与える ID の数はピアの演算能力に比例すると仮定している．これらの ID は，時計回りの昇順で円上に整列され，各 CQ はその円状の空間において，CQ の ID に最も近い ID を持つピアに割り当てられることなる (具体的な手続きは次節で述べる)．

CQ は ID に加えて，以下の情報を持つ．

1) 情報ソースに含まれる監視されるべきデータ項目

2) 要求者に通知する条件

3) そのような通知で返されるべき情報

4) CQ の終了条件

ピア p に y 個のデータ項目を参照する x 個の CQ が割り当てられていると仮定する（幾つかの CQ が同じデータ項目を参照するかもしれないので，"$x = y$" とは常にならないことに注意されたい）．$load(p)$ として示される p の負荷は以下のように定義される：

$$load(p) \stackrel{\text{def}}{=} \frac{0.25x + 1.0y}{z} \quad (1)$$

ここで，z は p に与えられた ID の数を示している．PeerCQ では出来る限りピアの負荷を減らすために，同じデータ項目を参照する CQ(同じグループの CQ ともいう) は，"類似した"ID が与えられるようになっている．これは、ID の上位 a ビットを，適切なハッシュ関数を使い，監視すべきデータ項目によって決定することで実現している．残りの $m - a$ ビットはランダムに決定される．ここで，この a は**グルーピングファクター** (GF) と呼ばれる．

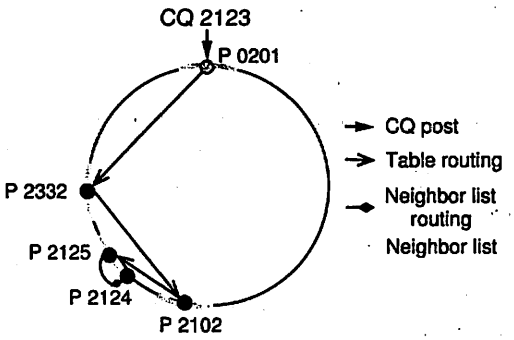


図 2: CQ の割り当て (第 1 段階).

2.2 CQ の割当

CQ のピアへの割り当ては，2 段階で実施される．まず 1 段階目では，CQ の所有者が決められ，2 段階目ではその所有者が局所的な探索を行うことで，適切な近隣のピアに CQ の実行が割り当てられる．

第 1 段階: i_q は与えられた CQ の ID とする．i_q の上位 a ビットは参照するデータ項目によって決まる．第 1 段階では，CQ は，$i_{p'} \geq i_q$ を満たす最小の $i_{p'}$ を持つピア p' に割り当てられる．すなわち，円状の ID 空間内で，CQ の ID に最も近いピアの 1 つに一時的に割り当てられる．以下では p' を CQ の所有者と呼ぶ．CQ の ID の決定方法により、同一のデータ項目を参照する CQ は，ID 空間を a 個に区切った"小さな"領域内にマップされるようになる．そのために，GF の増加は，同じデータ項目を参照する CQ が同一のピアに割り当てられる可能性を増やすことになる．このことは，式 (1) からシステム全体の負荷を減少させることにつながる．一方で，GF の増加はピアの負荷の不均衡を大きくする．負荷の不均衡をなるべく抑えつつシステム全体の負荷を下げるために，我々は，このようなトレードオフを考慮しつつパラメータ a の値を注意深く選ぶ必要がある．図 2 は第 1 段階で CQ の所有者が決まるまでの過程を表している (文献 [2] から引用)．この場合の所有者は P2124 で，CQ post は CQ の実行依頼を，Table routing はピア探索の際のショートカットを，Neighbor list は近隣ピアの探索空間 (第 2 段階で詳述する) を示している．

第 2 段階: CQ の所有者は，自身の近隣ピアに対して線形探索を行うことで，CQ の実行がより適切なピアを発見する。具体的には，探索空間のサイズはパラメータ r で決定され，円状の空間で右の近隣者 r 人と左の近隣者 r 人，そして自分自身を含む $2r+1$ 人の候補ピアの中から最も適切なピアが選択される．CQ q を割り当てるピア p の適切さは，以下に定義するユーティリティ関数を使うことで評価される．すなわち，最も評価値の大きなピアが選択される．

$$UF(p,q) \stackrel{\text{def}}{=} f(p) \times (g(p,q) + \alpha \times h(p,q)) \quad (2)$$

ここで，3 つの基本関数 f,g,h は，それぞれ以下のように定義される．関数 f は p の負荷量を示し，もし $load(p) \leq \theta$ ならば $f(p) = 1$ として，そうでなければ，$f(p) = 1 - \frac{load(p)}{\theta}$ として定義される．ここで，θ はあらかじめ決められた定数である．関数 g は，p に対する q の割り当てが，p が監視するデータ項目数を増やすかどうかを示している．p が q と同じデータ項目を既に監視しているならば 1，そうでなければ 0 として定義される．関数 h は，ネットワークコストを示し，p から q によって参照される情報ソースまでの ping 時間の逆数として定義される．

2.3 メンテナンス

A_p を，ピア p が所有者となる CQ の集合とし，B_p を，ピア p が実行する CQ の集合とする．ピアが参加する，または離脱するとき，CQ のそのピアへのマッピングは以下のように維持される．

procedure *ARRIVAL*(p)

1) 参加するピア p は，$i_{p'} > i_p$ を満たす最小の ID を持つピア p' を発見し，そのピアから $A_{p'}$ 内で $i_q \leq i_p$ を満たす CQ を引き継く．　その後，A_p を，受け取った CQ の集合に初期化し，それらの新しい所有者になる．

2) 次に，A_p に含まれる CQ に対して，ピアの割り当てを実施し，距離 r 内の全てのピアに対して，彼らが所有者となっている CQ の再割り当てを依頼する．

procedure *DEPARTURE*(p)

1) 離脱するピア p は，$i_{p'} > i_p$ を満たす最小の ID を持つピア p' を発見し，p' に A_p 内の CQ の所有者になるように依頼する．

2) もし，p に CQ の実行が割り当てられている場合，その CQ の所有者に対して，割り当て候補として p を考慮せずに，その CQ の再割り当てをするように依頼する．

3 提案手法

本章では，P2P 情報監視システムにおいて AND 条件をサポートする新しいシステムを提案する．提案手

法の基本的なアイデアは，AND 条件のクエリに対して，CCQ(Conjunctive Continual Query) と呼ばれる新しいエージェントを導入することである．以下では，CCQ の詳細を説明し，その割り当てとメンテナンスについて説明する．

3.1 CCQ

CCQ は以下の情報から構成される：1) 符号無し m ビット整数の識別子 (ID)，2)AND 条件を構成する各 CQ，3)AND 条件が満たされた時に返される情報，そして 4)CCQ の終了条件である．CCQ の概念を使い，AND 条件のクエリは以下のように実行される：

1) ユーザから AND 条件のクエリを受けとると，システムはそのクエリに相当する CCQ q^* を生成する．次に，**仲介者**と呼ばれるピアに，q^* の実行を割り当て，仲介者はその AND 条件のクエリに含まれる各 CQ の要求者になる (詳細は 3.3 節にて述べる)．

2) 仲介者は，各 CQ から受け取る通知を監視し，3.2 節で述べられる規則に従って，AND 条件の成立を監視する．

3) AND 条件が成立したことを検知すると，仲介者はその事実をユーザに対して必要な情報とともに通知する．

3.2 条件成立の基準

AND条件の成立は以下のように評価される．最初に，全てのピアはグローバルな時刻を参照することができ，CQ によって返される通知情報は，条件が成立した時刻が添えられると仮定する．2 つのパラメータ θ_1 と θ_2 を使い，AND 条件の成立を確認する手続きは以下のようになる．

1) AND 条件を構成する全ての CQ の通知情報を集めた後，仲介者は最も古い通知と最も新しい通知との間のタイムラグが，閾値 θ_1 以下であるかを確認する．もし，θ_1 以下ならば，AND 条件が成立したと認識する．そうでなければ，最も古い通知を無効にする．

2) CQ からの通知情報を受け取ると，仲介者は現在時刻と，通知情報のタイムスタンプとの間のギャップが，閾値 θ_2 以下であるかを確認する．もし，そのギャップが θ_2 を超えていたなら，受け取った通知情報を無効にする．そうでなければ，同一の CQ からの古い通知情報を，新しく受け取った通知情報に置き換える．

3) もし，AND 条件が成立していることを認識したら，仲介者はその CCQ の要求者に対して，その事実を通知する．CCQ の終了条件が満たされたなら，相当する CQ の全てにその事実を通知した後，実行を終了する．

3.3 CCQ の割り当て

CCQ の割り当ての詳細を述べる前に，AND 条件のクエリに関して，ピアの負荷を定義する．ピア p は，y 個の CQ のグループを参照する x 個の CCQ を割り当てられていると仮定する．このとき，AND 条件のクエリに関する p の負荷は，$load^*(p)$ として示し，以下のように定義する：

$$load^*(p) \stackrel{\text{def}}{=} \frac{0.25x + 1.0y}{z} \quad (3)$$

ここで，z は p に与えられた ID の数を表す．

CCQ のピアへの割り当ては，CQ の場合と同様に 2 段階で実施される．

第 1 段階: CQ の場合と同様に，CCQ の所有者を決定する．提案手法では，同じ CQ を参照する CCQ を，同一のピアに割り当てる可能性をできる限り高くするために，CCQ の ID は，CCQ を構成する各 CQ の排他的論理和として定義する．GF がシステム性能に与える影響は，第 4 章にて評価する．

第 2 段階: CCQ の所有者は，その近隣ピアに対して線形の探索を実施することで，割り当ての性能の改善を試みる．CCQ の割り当ての基本的なアイデアは，ユーティリティ関数を除いて CQ と同様である．ユーティリティ関数の詳細については次節で述べる．

3.4 ユーティリティ関数

CCQ のピアへの割り当てに利用する 2 つのユーティリティ関数を提案する．まず最初に，CQ の割り当てで利用した 3 つの基本関数を以下のように拡張する．

- ピアの負荷量を示す関数 f は，$f(p) \stackrel{\text{def}}{=} \min\{1.0, \frac{\theta}{load^*(p)}\}$ として再定義する．ここで，θ はあらかじめ決められた定数である．CQ の場合と対照的に，p の負荷が閾値を超えたときに，f の値は，負荷が増加するにつれて徐々に減っていくことに注意されたい．

- 関数 g は，p への q^* の割り当ての際，q^* によって参照される CQ 中で既に監視している CQ の数を，q^* によって参照される CQ の数で割ったものとして再定義する．

- ネットワークコストを表す関数 h は，CCQ に相当する全ての CQ に関して，p から CQ によって参照される情報ソースまでの ping 時間の逆数の和として再定義する．

これら 3 つの基本関数を利用して，以下の方法で 2 つのユーティリティ関数を提案する．最初の 1 つは式 (2) の単純な拡張であり，以下のように定義する．

$$UF_1(p, q^*) \stackrel{\text{def}}{=} f(p) \times (g(p, q^*) + \alpha \times h(p, q^*)) \quad (4)$$

ここで，αは定数である．上述の式は，$f(p)$によって支配されていることに注意されたい．すなわち，割り当てにおいて，ピアの負荷量が最も高い重みをもっている．2つ目のアイデアは，以下の方法で3つの基本関数の線形結合を行っている．

$$UF_2(p,q^*) = f(p) + \alpha \times g(p,q^*) + \beta \times h(p,q^*) \quad (5)$$

ここで，αとβは定数である．各ピアの負荷と全体の負荷との間のトレードオフは，パラメータαの調整により制御できる．すなわち，関数gは，同一のCQを参照するCCQを，できるだけ多く同一のピアに割り当てることにより冗長性を減らそうとし，関数fは利己的に個々のピアの負荷を減らそうとする，ということである．

4 評価

本章では，シミュレーションにより提案手法の性能を評価する．最初に，ピアの負荷の平均と均衡に対して，GFの影響を調べる．ここでいう，均衡とは分散を平均で割ることにより計算される測定基準である．次に，ランダムな選択の仕組みと比較することにより，提案するユーティリティ関数の有効性を示す．最後に，割り当ての段階で利用する近隣者数rの効果を調べる．

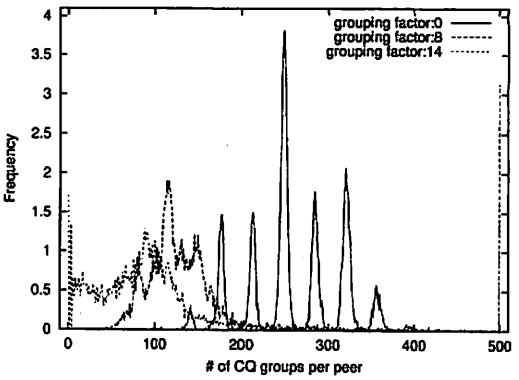
4.1 準備

シミュレーションにおいて，CQの割り当てのためのPeerCQの（円状の）オーバーレイネットワークに加えて，CCQの割り当てのためのオーバーレイネットワークを構築する．ピアの負荷の観点から，提案する手法の有効性を評価するために，ピアへのCCQの割当を実施している．以下では，システム内のピアの集合はシミュレーション中において変化しないと仮定している．そのような変化は，ピアの負荷という観点からは，手法の有効性に影響を与えない．

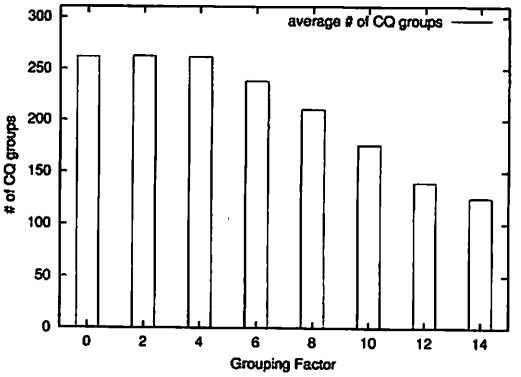
CCQに含まれるCQの数は，$i(\geq 2)$個で構成され，確率が$\left(\frac{1}{4}\right)\sum_{j=1}^{i-1}\left(\frac{3}{4}\right)^{i-1}$として与えられるような幾何分布に従っている．また，含まれるCQの最大数は10に制限されている．各CQはZipfの法則に従って，監視される対象（すなわち，ある情報ソースにおけるあるデータ項目）を確率的に選択する．すなわち，i番目に人気のデータ項目がCQによって監視される対象として選ばれる確率は以下のように与えられる．

$$P_i = \frac{1}{i^\epsilon \sum_{j=1}^{D}\left(\frac{1}{j^\epsilon}\right)} \quad (i = 1, 2, ..., D),$$

ここで，$\epsilon(\geq 0)$はZipfパラメータと呼ばれるパラメータである．以下では，特に明言されない限り1.0に固定する．ϵの増加は，確率の不均衡を促進し，$\epsilon = 0$は全てのデータ項目が同確率で選ばれる状況に相当する．



(a) CQグループ数に対するピア数の分布．



(b) 各ピアのCQグループ数の平均．

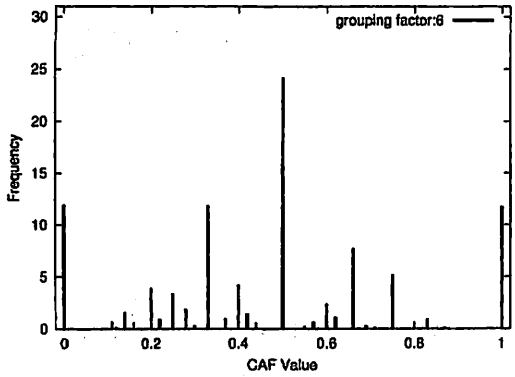
図 3: GFの影響．

本稿では，ピアと情報ソースとの間のping時間は，定数として見なすことができると仮定し，ユーティリティ関数については，以下のように仮定する．式(4)においては，いかなるpとq^*のペアについても$\alpha \times h(p,q^*) = 1.0$と仮定し，式(5)においては，$\beta = 0.0$，パラメータ$\alpha$の値は0.5，1.0，1.5として変化すると仮定する．

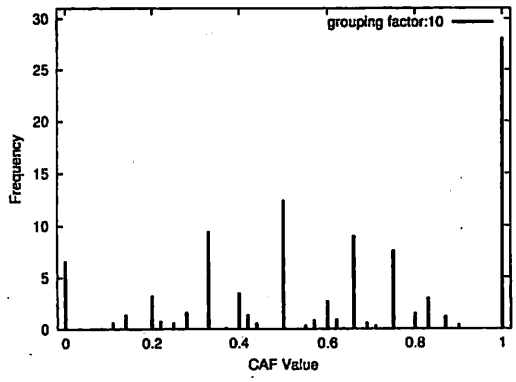
他のパラメータは以下のように設定した．1)IDの長さは64ビット，2) ピアの数は10,000，3)CCQの数は1,000,000，4) 近隣者数は5(即ち$r = 2$)，そして5) 10個のデータ項目を含む5,000個の情報ソースが存在し，データ項目の総数は50,000とした．

4.2 GFの影響

最初に，1) 各ピアに割り当てられたCQグループの数と，2) 観測された関数gの値の分布を計測することにより，提案手法の性能にGFがどのように影響するかを評価する．本節では，ユーティリティ関数としてUF_1を採用し，$\epsilon = 1.0$とした．



(a) $a = 6$.



(b) $a = 10$.

図 4: 関数 g の分布.

図 3(a) は，CQ グループ数に対するピアの数の分布を示し，図 1(b) は各ピアに割り当てられた CQ グループ数の平均を示している．図 3(a) における横軸 500 の値は，「少なくとも 500 個の CQ グループを割り当てられたピアの数」を表している．図から，ピアに割り当てられる CQ グループの数が GF が増加するにつれて減少していることが観測できる．すなわち，GF の値を大きくすると，ピアの CQ グループ数を十分に小さくできることがわかる．

図 4 は，シミュレーション時間中において，観測された関数 g の値の分布を示しており，この実験においては，GF の値を 0，6，10，14 のように変化させている．この図が示すように，関数 g の値は，GF が増加するにつれて 0 から 1 近づいており，十分に大きな GF を選択することで，関数 g は，実際に高確率で同一のピアに同じ CQ を参照する CCQ を割り当てていることを意味している．

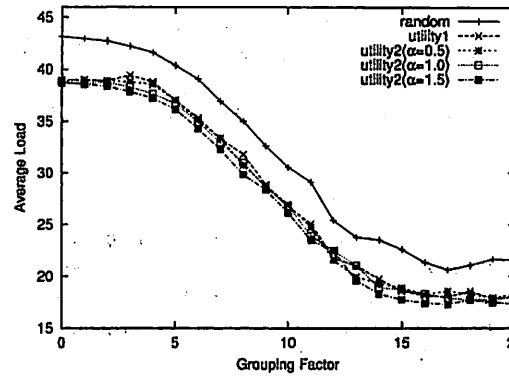


図 5: 平均の負荷による 5 つの手法の比較 ($\epsilon = 1.0$).

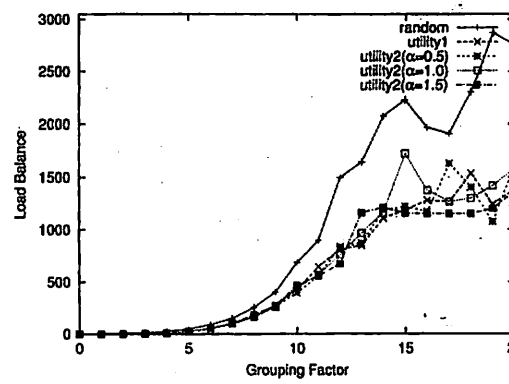


図 6: 負荷の均衡による 5 つの手法の比較 ($\epsilon = 1.0$).

4.3 ユーティリティ関数の比較

次に，2 つのユーティリティ関数の性能を比較する．この節では $\epsilon = 1.0$ の結果を示し，次の節では他の ϵ の値での結果を示す．

図 5 では，以下の 5 つの手法を用いた場合における平均の負荷を比較している．1) 近隣のピアの集合から CCQ を実行するピアをランダムに選択，2) 関数 UF_1 を使って，CCQ を実行するピアを選択，3)パラメータ $\alpha = 0.5$，1.0，1.5 を採用した関数 UF_2 を使った，CCQ を実行するピアの選択である．図からわかるように，ユーティリティ関数を使用した手法 (後半 4 つの手法) は，ランダムな手法と比べて，10%程度性能を改善しており，$\alpha = 1.5$ の UF_2 は僅かに他の手法より良いことがわかる．これは，平均の負荷を減少させるには，関数 f より，関数 g に対して，より高い重みを与える方が良いことを示している．

図 6 では負荷の均衡の観点から，5 つの手法の比較を示している．図からわかるように，後半 4 つの手法（ユーティリティ関数を使用した手法）間では，顕著な違いは

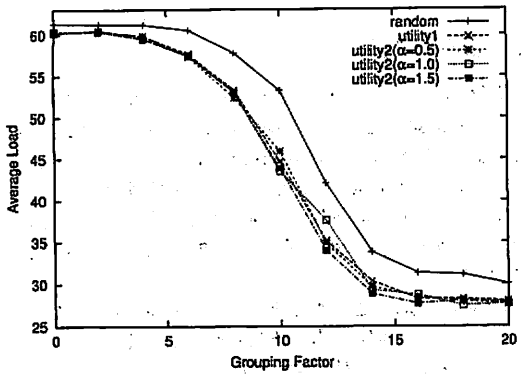


図 7: 平均の負荷による 5 つの手法の比較 ($\epsilon = 0$).

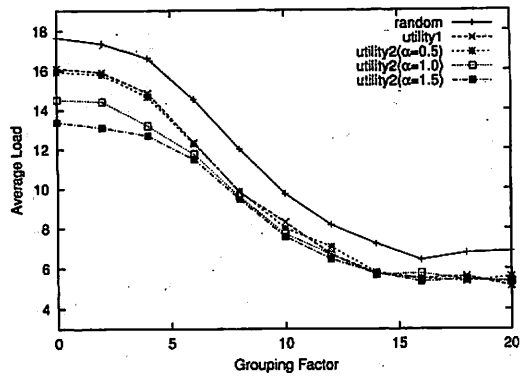


図 9: 平均の負荷による 5 つの手法の比較 ($\epsilon = 1.5$).

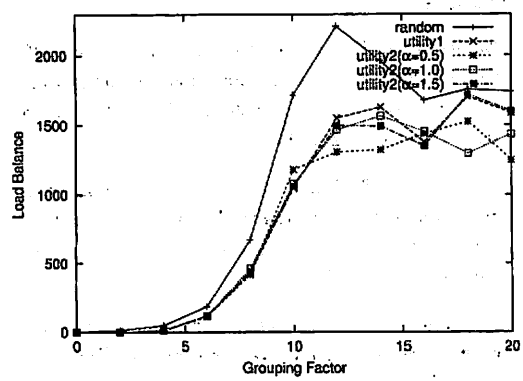


図 8: 負荷の均衡による 5 つの手法の比較 ($\epsilon = 0$).

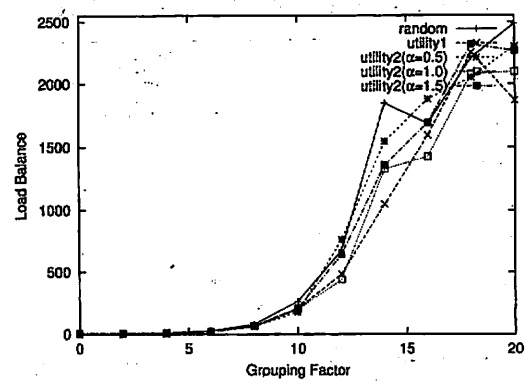


図 10: 負荷の均衡による 5 つの手法の比較 ($\epsilon = 1.5$).

見受けられないが，それらは最悪の場合でもランダムな手法と比べて，負荷の均衡を 50%程度改善している．この理由は，関数 f により明示的にピアの負荷を考慮しているためである．また，大きな GF における，負荷の均衡の増加は Zipf の法則による影響だと考えられる．すなわち，ユーティリティ関数の局所的探索により，平均化できないために，大きな不均衡が生じているのである．実際には，GF の値は，1) 負荷の均衡があまり高くなく，2) 平均の負荷が十分に小さくなるように決定されるべきなので，それぞれの手法に対して，6 から 8 の間の値が最良だと考えることができる．

4.4 Zipf パラメータの効果

次に，5 つの手法の性能に対する Zipf パラメータ ϵ の効果を調べる．

図 7 と図 8 は，$\epsilon = 0$ の場合の結果を示している．$\epsilon = 0$ は，全てのデータ項目が同じ確率で選択される状況に相当する．図 7 が示すように，平均の負荷は図 5 とほぼ同じ振る舞いを示すが，平均の負荷の絶対的な値はそれより著しく大きい．これは，$\epsilon = 1.0$ の場合では，局所性が小さくなるために参照するデータ項目数が増加したためである．一方で，図 8 が示す負荷の均衡に関しては，1)$\alpha = 0.5$ の UF_2 は僅かに他の手法よりも良く，2) 均衡の絶対的な値は図 6 より減少しているが，以前のように手法間での顕著な違いは見受けられない．すなわち，関数 g での参照データ項目の共有は，選択の一様な分布下では負荷の均衡を減少させるためにうまく働いておらず，ピアへのデータ項目の集中は発生しにくいと考えられる．

図 9 と図 10 は，$\epsilon = 1.5$ の結果を示している．図 9 が示すように，α が大きな手法は，GF が小さい時に平均の負荷の観点で最も良い性能を示している．しかしながら，GF が大きいと手法間での違いは小さくなる．これは，同じデータ項目を共有する可能性はランダムな手法でさえも十分に大きくなるためである．平均の負荷の絶対的な値は図 5 と図 7 と比較すると小さい．これは明らかに参照データ項目数が小さいためである．一方で負荷の均衡に関しては，図 10 が示すように，図 6 と図 8 での結果と

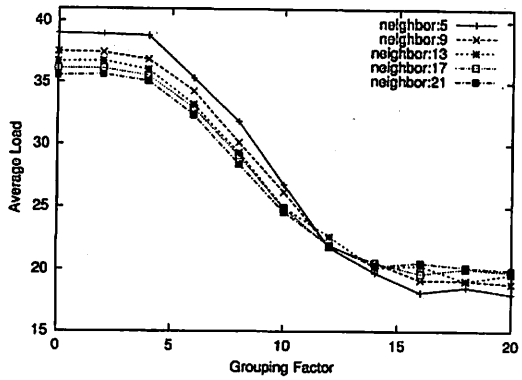


図 11: 近隣者数を変更した場合の平均の負荷の比較.

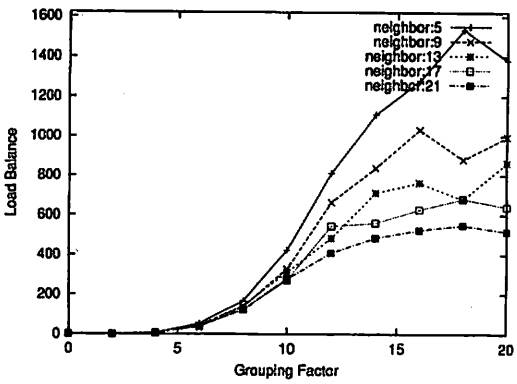


図 12: 近隣者数を変更した場合の負荷の均衡の比較.

は対照的にランダムな手法に対して，顕著な違いは見受けられない．すなわち，そのような高い集中のために起こる不均衡は単なる局所的な探索では解決できないということである．

4.5 近隣者数の影響

最後に，パラメータ r の値を 5 から 21 に増やし，全体の性能に対して近隣者数の影響を調べる (この節ではパラメータ ϵ は 1.0 に固定している)．図 11 と図 12 は，それぞれ UF_1 での負荷の平均と均衡の結果を示している．図で示すように，各サイズ間での効果の顕著な違いは見受けられないが，r の増加は，負荷の平均と均衡を減少させている．r の増加に対する平均の負荷の飽和は，負荷の均衡の減少によって説明できる．

5 おわりに

本稿では，AND 条件をサポートする P2P 情報監視システムを提案した．特に，参照するデータ項目の類似性を考慮し，ピアへの CCQ(Conjunctive Continual Query) のマッピングを提案した．個々のピアの負荷とシステム全体の負荷との間のトレードオフが明示的に制御できるように，PeerCQ において使われるユーティリティ関数を改良した．

今後の課題は，通信遅延やピアの演算能力の不均質性のような，より現実的な状況を反映した詳細にわたるシミュレーションの実施を考えている．

参考文献

[1] B. Gedik and L. Liu. PeerCQ: A Scalable and Self-Configurable Peer-to-Peer Information Monitoring System. Technical Report GIT-CC-02-32, Georgia Inst. of Technology, 2002.

[2] B. Gedik and L. Liu. PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System. in *Proc. Int'l Conf. Distributed Computing Systems*, 2003.

[3] L. Liu, C. Pu, and W. Tang. Continual Queries for Internet Scale Event-Driven Information Delivery. *IEEE Trans. Knowledge and Data Eng.*, 11(4):610-628, July/Aug, 1999.

[4] L. Liu, W. Tang, D. Buttler, and C. Pu. Information Monitoring on the Web: A Scalable Solution. World Wide Web J., 2003.

[5] A. Rowstron, A. Kermarrec, M. Castro, and P. Drushel. Scribe: The Design of a Large-Scale Event Notification Infrastructure. in *Proc. Int'l Workshop Networked Group Comm.*, 2001.

[6] A. Rowstron and P. Drushel. Pastry: Scalable, Decentralized Object Location and Routing for Largescale Peer-to-Peer Systems. in *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms*, 2001.