分散システムにおける読み書き問題に対する拡張されたコテリーの構成とその応用

芦原 評[†] 清水 謙多郎[†,††]

分散システムにおいて，性能ないし信頼性・可用性の改善のため，オブジェクトの複製が行われることがある．複製されたオブジェクトへの並行アクセスに際し一貫性を保つための方法として，従来広く用いられてきたものに，Read-One, Write-Allや，その一般化である重み付き投票がある．本論文では，一貫性を保つために必要な情報構造をより一般化した形で定義し，共コテリーと呼ぶ．これは分散排他制御に用いられるコテリーの拡張であり，読出しと書込みの2種類の操作に対して最適化されている．そして，任意の数の複製に対して共コテリーを構成する簡単な手順を与え，これが重み付き投票によるものと比べ，コストおよび可用性の両面ですぐれていることを示す．

A Construction of Coteries for Readers and Writers in Distributed Systems

HYO ASHIHARA[†] and KENTARO SHIMIZU[†,††]

Quorum consensus is a widely accepted scheme for maintaining consistency among replicated objects in distributed systems. A popular method to produce quorum sets is weighted voting, but many useful quorum sets are not represented by any vote assignment. In this paper, we present a simple method to obtain a class of non-vote-assignable quorum sets. We show that the proposed sets are superior to vote-assignable ones in many cases, especially for the system with very large number of nodes, both in communication cost and in availability.

1. はじめに

分散システムにおいて，ファイル等のオブジェクトを複製し分散配置することがある．その目的としては，(1) 近くの複製にアクセスすることによるコストの低減，(2) 単一ノード・リンクへのアクセスの集中を避けることによる並行性，性能および信頼性の向上，および (3) 自動的なバックアップによる信頼性および可用性の向上をあげることができる．ただし，これらは必ずしも同時には達成されない．オブジェクトの複製化に際しては，複製どうしの整合性を保ち，外部に対しては単一の実体と同様のセマンティクス（複製透過性）を持たせる必要があるためである．

そのためには，オブジェクトに対する並行アクセスを全システムで一意に直列化して実行すればよい．ただし，オブジェクトに対するアクセスのうち，副作用（データの変更）をともなわないものどうしの順序は交換できる．すなわち，アクセスを，副作用をともなう「書込み」と，そうでない「読出し」とに分けるとき，(a) 読出しどうしを除く，書込みと読出し，書込みと書込みのアクセスが因果律に従いつつ直列化され，(b) その順序に従い，読出しがつねに最新の書込みの結果を与えることを保証すればよい．順序は論理的なものであり，実時間やシステム内の他の事象の順序と一貫していることは，より望ましいとはいえ必須ではない．

書込みが最新の書込みの結果の上に行われることも要求される場合が多いが，書込みを，結果がその操作のみによって定まる完全な上書きとして定義することで，この性質は不要となる．以下では，特に断らない場合，この性質は仮定しない．結果がそれ以前の状態に依存するような書込み（変数のインクリメント，複数要素からなるデータの部分書換えなど）では，書込みと読出しを同時に行うと考える．

また，並行アクセスの行われる共有オブジェクトに対しては，たとえ実体が物理的に単一であっても，並行性制御を行わなければならない．複製に対しては，

† 電気通信大学情報工学科
Department of Computer Science, The University of Electro-Communications
†† 東京大学農学生命科学研究科応用生命工学専攻
Department of Biotechnology, The University of Tokyo

各複製の内的一貫性と複製相互の一貫性の双方を保証する必要がある．複製に対する並行性制御については膨大な研究が行われているが，基本的には1回ごとの読み/書きでなく，複数のアクセスを含む一連の操作を1つの不可分トランザクションとして直列化の対象とすればよい．したがって，上記と同一の機構によって実現できる[12),15)]．以下の議論ではこれらを区別しないものとする（書込み=書込みを含むトランザクションと考える）．

以上を実現する代表的な分散アルゴリズムとして2相ロックをともなう Read-One, Write-All 方式（ROWA）がある．この方式は，書込みはすべての複製に排他ロックをかけたうえですべての複製に対して行い，読出しは任意の（通常は最も近くの）複製に対し共有ロックをかけて行うものである．このうち，ロック操作が前記（a）に，実際のアクセスが（b）に相当する．アクセスが十分疎で競合のおそれがなければ，ロック操作は省略できる．このように，(a) と（b）は区別されつつも論理上は同一の問題と見なせる．

これは分散排他制御[25)]の変種でもある．すなわち，書き手と読み手は排他的であるが，読み手どうしは同時にいくつでも共存でき，書き手どうしは異なる複製上でいくつでも共存できる．また，多くの現実的なケースでは，読み手の方が書き手より多い．これらを利用して，単なる排他制御よりも高い並列性や少ないメッセージ数での一貫性を得ることができる．これを**分散読み書き問題**と呼ぶ．さまざまのオブジェクト型に関して，それぞれの独自の特徴によってさらに最適化できることもある[14)]が，読み/書きの一般性の高さは，これを特に有用なものとしている．

以下では，対象となるオブジェクトが論理的に完全結合のネットワーク上の N 個のノードに複製されている場合を考え，便宜上，ノードとそのノード上の複製を同一視する．

最も単純な読み書きの制御方式は唯一の主複製を定める（他の複製は単なるバックアップとしてのみ働く）集中方式であるが，主複製が性能および耐故障性の両面においてボトルネックとなるため，大規模な分散システムには適さない．専有権トークンを移動させることに基づく各種の方式は，トークンの位置情報管理と組み合わせた動的な集中方式と見なせる．

すでに述べた ROWA では頻度の多い読出しを安価に解決できる利点があるが，書込みの際には全ノードをロックしなければならないため，やはり規模適応性に欠ける．また，1ノードでも停止するとそのままでは動作を続けることができない．

集中方式と ROWA，および多数決方式[29)]を統一的に一般化した，重み付き投票[11)]によるクォラム合意方式では，読出しに最低 r 個，書込みに w 個の相異なるノードにアクセスし合意を得るならば，$r+w>N$ であるとき一貫性が保証される．さらに各ノードに重み（票数）を与えることでノードごとの性能や信頼性，アクセス頻度等の違いを反映させる[26)]こともできるが，ノードの条件がすべて同じ場合にはアルゴリズムも対称であることが負荷分散・危険分散の点で通常望ましい（偶奇によるたかだか1の差を除く[4)]）．逆に，重み0のノードを設けることで集中方式に近づけ，メッセージ数を減らすこともできる．ただし，一般に，制御に直接参加しないノードの存在は，本論文の議論の範囲では無視できる．この意味においては，集中方式はそもそも複製化を行わないことと等しい．

コテリー[10)]（交差シュペルナー族）はさらに一般的に，アクセスに必要なノードを集合の形で指定したものである．本来通常の分散排他制御問題に対して定義されたものであるが，読み書き問題にも適用可能である．前川の方式[20),21)]における有限射影平面はその1つであり，N 個の複製のうち約 $\sqrt{N}$ 個へのアクセスによって一貫性を保証し，静的かつすべての複製の負荷が等しいという意味において真に分散的である．しかし，当然のことながら読み書き問題の特徴（読み手の共存可能性，読み手の多さ）による優位を利用してはいない．また，きわめて限られた N に対してしか完全な形では適用できない．

多重投票[27)]，投票構造の動的変更[5),9),16),18)]，木構造による階層化[1),2)]，完全結合でないネットワークへの対応[13),17)]等の付随的な変形においてもコテリーは基本的な情報構造となる．しかし，投票によって表現できないコテリーの読み書き問題への応用については，概念的な言及のみで実際の研究は少ない．

本論文では，コテリーを分散読み書き問題に適した形に拡張した新しい構造，**共コテリー**（co-coterie）を定義し，有効な共コテリーを実際に構成する．これは，任意の N, r に対して，読出しに r 個，書込みにたかだか $\lceil N/r \rceil$ 個のノードへのアクセスのみで一貫性を保証できるというものである．

2章において，共コテリーを定義し，共コテリーを用いた読み書き制御アルゴリズムの概略とその評価基準を述べた後，3章で共コテリーの構成法を提案し，4章で従来方式との比較を行う．5章では現実の分散システムへの応用について論じる．

2. 共コテリーによる読み書き制御

2.1 定　　義

定義1 （コテリー）

集合 S があるとき，S の空でない部分集合の集合 C：

$C = \{G_i\} \subseteq 2^S$,

$G_i \subseteq S \quad (i = 0, 1, ..., |C| - 1)$

が，条件

$\forall i \neq j \quad G_i \not\subseteq G_j$ （極小性），

$\forall i, j \quad G_i \cap G_j \neq \emptyset$ （交差性）

を満たすとき，C を S 上のコテリーと呼ぶ．

詳細は文献 10) を参照．

定義2 （共コテリー）

集合 S があるとき，S の空でない部分集合の集合の対 $T = \langle RC, WC \rangle$：

$RC = \{R_i\} \subseteq 2^S$,

$WC = \{W_i\} \subseteq 2^S$,

$R_i \subseteq S \quad (i = 0, 1, ..., |RC| - 1)$,

$W_i \subseteq S \quad (i = 0, 1, ..., |WC| - 1)$

が，条件

$\forall i \neq j \quad R_i \not\subseteq R_j, \quad W_i \not\subseteq W_j$ （極小性），

$\forall i, j \quad W_i \cap R_j \neq \emptyset$ （交差性）

を満たすとき，T を S 上の共コテリーと呼ぶ．

RC, WC は，それぞれ読出しと書込みに要するノード集合に対応する．C がコテリーであるとき，かつそのときのみ，$\langle C, C \rangle$ は共コテリーである．重み付き投票によって表現される，読出しおよび書込みクォラム集合も共コテリーであるが，いかなる票配分によっても表現されない多くの共コテリーが存在する．

類似の構造については文献 3), 6), 12) などでも言及されているが，深い追求は行われていない．

上記に加えて，書込みどうしの交差

$\forall i \neq j \quad W_i \cap W_j \neq \emptyset$

が満たされるならば（すなわち WC がコテリーであるならば），書込みが最新の書込みの上に行われることも保証される．しかし，本論文では一般性を高めるため，共コテリーの定義においては読出しと書込みは対称とし，両者の違いは実現段階で表現することとした．すなわち，$\langle RC, WC \rangle$ が共コテリーであるとき，$\langle WC, RC \rangle$ も共コテリーである．

定義3 （支配性）

S 上の異なる 2 つの共コテリー $T = \langle RC, WC \rangle$ と $T' = \langle RC', WC' \rangle$ が

$\forall R'_i \in RC' \quad (\exists R_i \in RC \quad s.t. \quad R_i \subseteq R'_i)$,

$\forall W'_i \in WC' \quad (\exists W_i \in WC \quad s.t. \quad W_i \subseteq W'_i)$

なる関係にあるとき，T は T' を**支配する**という．S 上のいかなる他の共コテリーも T を支配しないとき，すなわち

$\neg(\exists R \subseteq S \; s.t. \; (\forall i \; R_i \not\subseteq R, \; \forall i \; W_i \cap R \neq \emptyset))$,

$\neg(\exists W \subseteq S \; s.t. \; (\forall i \; W_i \not\subseteq W, \; \forall i \; R_i \cap W \neq \emptyset))$

であるとき，T は**支配的**であるという．

定義4 （均等性）

S 上の共コテリー T において

$\forall i \quad |R_i| = r$,

$\forall i \quad |W_i| = w$,

$\forall e \in S \; |\{i | \; e \in R_i\}| = r^*$,

$|\{i | \; e \in W_i\}| = w^*$

（それぞれ定数）であるとき，T は**均等**であるという．

2.2 アルゴリズム

オブジェクトの複製を持つ全ノードの集合を S とし，S 上の共コテリー $T = \langle RC, WC \rangle$ を与える．ノード j における T を用いた読み書き制御アルゴリズムの例を **図1** に示す．

要求を行うクライアントノード，複製を持つサーバノードとも，ノード j は時計 TSj を持つ．TSj は実時間の経過に従って増加するとともに，タイムスタンプ TS を持つメッセージを受け取った際，$TSj := \max(TS, TSj)$ とする．また，これとは別に，ノード j の複製は，タイムスタンプ $TSRj$ を持つ．

クライアントからは W_REQUEST（書込み要求），R_REQUEST（読出し要求），RELEASE（要求終了）および YIELD（権利放棄）の 4 種，サーバからは GRANT（要求受入れ）および INQUIRE（放棄問合せ）の 2 種のメッセージが互いに送られる．各メッセージには要求の ID，要求元ノードおよびプロセスの ID，タイムスタンプ（送信時の TSj）が付されている．さらに，GRANT メッセージは複製の $TSRj$ を示すタイムスタンプを持つ．任意の 2 ノード間のメッセージの配送とその順序は保証されるとする．

各複製は受け取って終了前の書込みおよび読出し要求のキューを持つ．キューの各要素は，要求自体の内容に加えフラグ enter を含む．enter は初期値 F で，その要求に関し実際のアクセスが開始されるか，INQUIRE メッセージが送られた時にセットされ，YIELD メッセージを受け取るとリセットされる．要求は原則としてタイムスタンプの昇順に並べられる．キューの先頭にある書込み要求，またはキューの先頭から最初の書込み要求までの連続する読出し要求は，granted であるという．

読出しを行うには，いずれかの $R_i \in RC$ の全要素の合意，すなわちロックの確認を得る．同様に，書込みについては任意の $W_i \in WC$ の全要素の合意を得

・クライアント
書込み時 {
1. $W_i \in WC$ を1つ選ぶ.
2. W_i の全ノードに W_REQUEST(TSj) を送信する.
3. W_i の全ノードから GRANT(TS) を受け取るまで待つ.
4. W_i の全ノードに対し書き込む.
5. W_i の全ノードに対し RELEASE(TSj) を送信する. }

読出し時 {
1. $R_i \in RC$ を1つ選ぶ.
2. R_i の全ノードに R_REQUEST(TSj) を送信する.
3. R_i の全ノードから GRANT(TS) を受け取るまで待つ.
4. 最大の TS を持っていたノードの1つから読み出す.
5. R_i の全ノードに対し RELEASE(TSj) を送信する. }

INQUIRE を受け取ったとき {
1. 必要なすべての GRANT が揃う前なら,
・送信元ノードに YIELD を返す.
・そのノードからの GRANT をいったん取り消す.
2. GRANT が揃った後なら, そのまま実行を続ける. }

・サーバ
W_REQUEST(TS) を受け取ったとき {
1. キューが空であるなら,
・要求をキューに入れて複製を排他ロックする.
・要求元に GRANT($TSRj$) を返す.
2. キューが空でないなら,
・granted またはタイムスタンプが TS より
小さい要求すべての後に新たな要求を挿入する.
・TS より大きなタイムスタンプを持ち, granted かつ ¬ enter である要求があれば, それらの各要求元に対し INQUIRE を送る. }

R_REQUEST(TS) を受け取ったとき {
1. キューが空であるなら,
・要求をキューに入れて複製を共有ロックする.
2. キューが空でないなら,
・granted の書込みまたはタイムスタンプが TS
より小さい要求すべての後に新たな要求を挿入する.
・TS より大きなタイムスタンプを持ち, granted かつ ¬ enter である書込み要求があれば, その要求元に対し INQUIRE を送る.
3. 要求が granted ならば, 要求元に GRANT($TSRj$) を返す. }

RELEASE(TS) を受け取ったとき {
1. キューから対応する要求を取り除く.
2. 取り除いた要求が書込みであったなら, $TSRj := TS$ とする.
3. キューが空となるなら, ロックを解除する.
4. 新たな先頭の要求が書込みであるなら,
・現在のロックを解除し排他ロックする.
・新たな先頭の要求元に GRANT($TSRj$) を返す.
5. 取り除いた要求が書込み, 新たな先頭が読出しであるなら,
・排他ロックを解除し共有ロックする.
・granted となる各要求元に対し GRANT($TSRj$) を返す. }

YIELD を受け取ったとき {
1. その要求をいったんキューから除く.
2. 到着時と同じ規則によって改めてキューに入れる.
3. 取り除いた要求がキューの先頭であったなら,
・その結果新たに先頭となった要求について,
RELEASE の場合と同様の処理を行う. }

図1 共コテリーによる複製制御アルゴリズム
Fig. 1 A replica control algorithm with a co-coterie.

る. 共コテリーの交差性により, 読み手と書き手が同時に条件を満たすことはないので, 排他制御が行われる. さらに, 読み手の得た合意のうちには少なくとも1つ最新の書込みの結果を持つ複製からのものがあり, タイムスタンプによって識別できるので, 実際の読出しはそこから行えばよい. データ本体のサイズが小さいときはこれを制御メッセージと同時に送ることができる. そうでなければ別途アクセスを行う. コミット/アボート手続きの記述は省略した.

各キューにおいて, ある要求を待たせることができるのは, より小さいタイムスタンプを持つ要求か, すでに実行を開始している要求かであるので, デッドロックおよびスタベーションは発生しない. 競合の解決はここではタイムスタンプによる先着順としているが, 書込み優先・読出し優先などの他の解決法[8]も可能である.

実際の制御アルゴリズムには, この他にも無数のバリエーションが可能であるが, アルゴリズムの選択やその詳細な記述は共コテリーの選択と基本的には独立であり, 本論文の中心ではないので深くは触れない. 重要なものについて以下に簡単に述べる.

時計として, 各ノードが十分に正確な実時間時計を持つならそれを使用できる. また, Lamport の論理時計を用いることもできる. この場合, 読出しの「最新」性は実時間による順序と一致しないこともありうるが, システム内での因果に関する矛盾は生じない. 上記では両者の併用である.

書込みに際し, いずれかの R_i といずれかの W_i の合意をともに得るならば, 時計なしでもバージョン番号の管理のみで実時間における最新版の読出しと最新版への書込みが保証される. WC がコテリーであればこれは W_i からの合意だけで実現される.

要求される性能とそれを決定する要因はシステムによって異なるため, それぞれに応じた最適化が必要である. メッセージ数の最小化のためにはあらかじめ選択した W_i, R_i の要素に対して要求を送ればよいが, 耐故障性および並行性向上のため, 一定時間返答がない場合は他の $W_{i'}$, $R_{i'}$ を選び直すことができる. これは, 送信された各メッセージに対し ACK を求める (プロトコルの下位で実現されている場合が多い) ことでより効果的に行える. さらに, ネットワークのバンド幅に余裕がある場合は, 複製を持つすべてのノードに対して並行して要求メッセージを送り, 最初にいずれかの W_i, R_i の合意が揃った時点で読出しを行うことで, 操作の応答時間短縮を図ることができる.

W_i, R_i の選択はランダムに行うだけでなく, 距離

の近いノードや負荷の低いノードを多く含むものを優先して選ぶこともできる．特に，読み手/書き手が直接全複製にアクセスするのでなく，1つの複製に対して要求を送り，その複製が他の複製とのコーディネイトを行うシステムの場合，および読み手/書き手自身が複製の存在するノード上にある場合には，自分自身を含む R_i ないし W_i を選択すればメッセージ数を減らせる．

書込み時，W_i 以外の複製の更新はバックグラウンドで任意に遅延して行う（あるいは行わない）ことができる．すべての複製が最新の内容を保持することは，耐故障性の面から見て望ましいが，必ずしも必要ではない．逆に，一時的に旧バージョンを残すことで書込み途中でのクラッシュや誤操作に備えることも考えられる．アクセス終了後の解放を他の要求があるまで行わない方式，衝突の検出をアクセス後に行う楽観方式，複製の更新に代えて無効化を行う方式などを組み合わせることも可能である．また，要求ノードと複製ノード間で，データ全体を転送するのでなくコマンドと結果のみを交換する形の応用に対しても，簡単な変更で対応できる．

2.3 評価基準

与えられた共コテリー T が分散読み書き問題に適用される場合の望ましさを評価する基準を考える．

ここでは各ノードの性能および信頼性，ノード間の距離は等しいものとする．そして，冒頭に掲げた複製化の目的に従い，(1) コスト，(2) 負荷分散，(3) 耐故障性の3点から評価を行う．

T が T' を支配するとき，T' を用いて可能な読み/書きはすべて T を用いても行うことができ，かつ逆は成り立たない．よって，T はコスト・負荷・耐故障性のいずれにおいても T' よりすぐれている（均等性は崩れることがありうるが，均等性はそれ自体が直接の目的ではない）．

T が支配的でないとき，定義3により，どの R_i も含まないノード集合 R がすべての W_i と交わるか，またはどの W_i も含まないノード集合 W がすべての R_i と交わる．そのような R が存在すればこれを RC に，W が存在すれば WC にそれぞれ加え，また，R を含む R_i や W を含む W_i が存在するならばそれらを RC，WC から取り除くことで，T を支配する共コテリーを構成できる．これを繰り返すことで，支配的な共コテリーを得ることができる．以上により，T が支配的であることは，望ましく，また以下の議論において仮定してよい．

(1) コストの第一の指標は $|R_i|$ および $|W_i|$ の大きさである．制御に必要なメッセージ数，プロセッサ使用量およびディスクアクセスはいずれもこれらに比例すると考えられる．これらが小さいならば，参照される側の各複製の平均負荷も同時に小さくなる．読出し頻度が高いとき，$|R_i|$ は特に小さいことが望まれる．逆に制御メッセージに比してデータ本体が巨大な場合 $|W_i|$ が重要になる．通常 R_i と W_i の大きさはトレードオフになるのでバランスをとる必要がある．

(2) $|R_i|$，$|W_i|$ の大きさにばらつきがあるとき，単純には大きさの小さい R_i，W_i を選んでアクセスすればメッセージ数を減らせるが，それによる負荷の集中はかえって全体の性能の低下を招くことがある．一定のシステム負荷に対し，原則的にはノード間の負荷均衡がノード負荷の最大値を最小にしシステム全体の性能を最大にする．したがって T が均等またはそれに近いことを目標とする．T が均等であるなら，参照側のコストが一定になると同時に，R_i，W_i を一様に選ぶときの被参照側の負荷も等しくなり，前川の意味[21]で真に分散的なアルゴリズムが実現される．なお，r^*，w^* が一定であることは各ノードの被参照時の負荷の均衡を意味するが，r^*，w^* の大きさ自体は負荷の大きさではないことに注意する．

すべての R_i ないしすべての W_i のすべての要素の合計延べ個数は，S の全要素を重複して数えたものに等しいので，T が均等であるとき，

$$\frac{r|RC|}{r^*} = \frac{w|WC|}{w^*} = N$$

が成り立つ．1回の読出し要求によって各複製は $(r^*/|RC|) = (r/N)$ の確率で参照されるので，これを負荷の指標とできる．書込み要求についても同様である．ただしデータ本体の読出しは最新の複製の1つから行われるので，w が小さい場合最新の複製の負荷が一時的に高くなる．r，w が一定であれば N が大きいほど負荷が低くなるので，操作に要するメッセージ数は同一であっても，システム全体として並行性が高く，結果的に応答時間の短縮および最大スループットの向上が得られる．

ある R_i の r 個の各要素はそれぞれ w^* 個の W_i に含まれ，すべての W_i は R_i の少なくとも一要素を含む．したがって $|WC| \leq r \cdot w^*$，同様に $|RC| \leq w \cdot r^*$ である．よって，

$$N \leq r \cdot w$$

が得られる．等号の場合が，メッセージ数および負荷についての理想値を示す．

(3) 耐故障性に関しては，ノードの単純停止を問題とし，リンク障害や，任意故障については現時点では

考えない．ただし，ネットワーク分断は非連結となったノードの停止と同様に扱うことができる．また，停止を他ノードによって検出し，共コテリーの変更を含むさまざまの復旧措置をとることは可能であるが，ここではなんら特別の手段を講じることなしにシステムが機能する限界を検討する．さらに，アクセス途中の過渡的状態での停止に関しても共コテリーの選択でなく制御アルゴリズムの問題であるので議論から除く．

明らかに，いずれかの R_i に属する全ノードが停止することが，書込みが不可能になる十分条件である．また，T が支配的であるならば，必要条件でもある．なぜなら支配的な T においてすべての W_i が停止しているノードを含むならば，停止したノードの集合 R はすべての W_i と交わるため，いずれかの R_i を部分集合とするからである．読出しの性能および耐故障性はいずれかの R_i を揃えることが容易であることを目標とするので，書込みの耐故障性とは本質的に相反する．同様に，いずれかの W_i に相当するノードがすべて停止すれば，読出しは不可能となる．ただし，一貫性のみで最新性を要求しない「弱い読出し」を定義すれば，任意の複製で可能である．

耐故障性の指標の1つである脆弱度[4]，すなわち，最悪の場合，最小何個のノードが停止すればシステム全体で操作が不可能になるかの数は，読出しおよび書込み操作に関して，それぞれ $|W_i|$，$|R_i|$ の最小値に等しい．特に $|W_i|$ は最新データを持つ複製数でもある．これらの値は当然大きいことが望ましい．その意味で性能と耐故障性の要求は対立する．一方，均等性，r と w のバランス，多くの応用において w の方が r より大きいこと，が求められる点では共通している．特に均等性は重要である．r，w の均衡は脆弱度の均衡を，r^*，w^* の均衡はそれぞれのノードが停止した場合の損失の均衡を意味する．

最悪ケースの検討とともに，システムの平均稼働確率（可用度）[28]も重要である．この厳密な計算は一般には複雑なものになるが，素朴には，$|RC|$ が大きく $|R_i|$ が小さいことが読出しの可用度を高める．しかし，支配的な共コテリーどうしの比較においては，これはむしろ書込みの可用度の低さを意味し，読出しの可用度を高めるためには $|WC|$ が小さく $|W_i|$ が大きいことが望まれる．

各ノードが互いに独立に確率 p で停止しているとした場合に読出しが不可能になる確率（1−可用度）$Fr(p)$，書込みが不可能になる確率 $Fw(p)$ は，それぞれ，すべての R_i が少なくとも1つの停止しているノードを含む確率，すべての W_i が少なくとも1つの停止しているノードを含む確率である．T が支配的なとき，これらは，前述の読出しと書込みの相補性により，いずれかの W_i が稼働しているノードを1つも含まない確率，いずれかの R_i が稼働しているノードを1つも含まない確率にそれぞれ等しい．

また，可用度の計算は「停止しているノード」と「稼働しているノード」について対称である．すなわち，各ノードが停止している確率が p のとき，すべての R_i が少なくとも1つの停止しているノードを含む確率は，各ノードが稼働している確率が p のとき，すべての R_i が少なくとも1つの稼働しているノードを含む確率と等しい．

したがって T が支配的なとき，$Fr(p), Fw(p)$ は

$$Fr(p) = 1 - Fw(1-p)$$

なる関係を満たす．さらに，T が均等で p が十分小さいとき，$w+1$ 個，$r+1$ 個のノードが停止する確率は w 個，r 個のノードが停止する確率に比べそれぞれ無視できるので，

$$Fr(p) \simeq |WC| \cdot p^w,$$
$$Fw(p) \simeq |RC| \cdot p^r$$

である．通常はこのうち大きい方がより重要になる．

上記のさまざまな要素を統合した評価関数を作る試みもなされている[19),22),23]．その他にも，現実の分散システムは，未知のものを含む非常に多数のパラメータを持ち，評価基準も多様なものが考えられる．しかし，単純なモデルによるいくつかの基本的な評価は，多くの具体的な状況に対しても有効である．どの要素を重視するかはシステムに依存するため，単一の性能基準や最適解が定まるわけではなく，アルゴリズム同様，共コテリー自体もシステムに応じた選択が必要である．

一方，基準が与えられたとしても，N が大きいとき一般に最適のものを求めることは困難である．また N 自体も，自由に決定できる場合と与えられる場合とがある．そこで，N，r と w，$|RC|$ と $|WC|$ のバランスなど，広い範囲の要求に対し，それを満たす共コテリーを爆発的な計算量をともなわずに生成することのできる手順を与えることを目標とする．

3. 共コテリーの構成

(i) $|WC|$ 最小解

与えられた $N, r > 0$ に対して，$N \bmod r = 0$ であるとき，$w = (N/r)$ とおいて，

$$S = \{n_{ij} | 0 \le i < w, 0 \le j < r\}$$

とする．任意の番号付けにより，$i + j \cdot w$ 番目のノードを n_{ij} とすればよい．このとき，

$N = 6, r = 2, w = 3$
$S = \{ 0, 1, 2, 3, 4, 5 \}$
$WC = \{\{0,1,2\},\{3,4,5\}\}$
$RC = \{\{0,3\},\{1,3\},\{2,3\},$
$\{0,4\},\{1,4\},\{2,4\},$
$\{0,5\},\{1,5\},\{2,5\}\}$

図2 共コテリーの例
Fig. 2 An example of co-coteries.

$$W_i = \{n_{ji} | 0 \le j < w\} \quad (0 \le i < r),$$
$$R_i = \left\{ n_{jk} \middle| \left\lfloor \frac{i}{w^k} \right\rfloor \bmod w = j, \right.$$
$$\left. 0 \le j < w, 0 \le k < r \right\} \quad (0 \le i < w^r)$$

とする．すなわち，w ノードずつ r 個の W_i に分けた後，それぞれの W_i から任意のノードを1つずつ選んで和を取る．選び方は w^r 通りあるので，そのそれぞれを R_i とする．

$$|R_i| = r, \quad |W_i| = w,$$
$$|RC| = w^r, \quad |WC| = r,$$
$$r^* = w^{r-1}, \quad w^* = 1$$

である．

明らかに2.1節の条件（極小性・交差性）は満たされ，さらに支配的かつ均等である．$N = r \cdot w$ であるから，均等な共コテリーとしては与えられた N, r に対し最小の w，最小の $|WC|$ である．

上記の一部である，

$$R'_i = \{n_{ij} | 0 \le j < r\} \quad (0 \le i < w)$$

だけでも共コテリーの定義と均等性は満たされるが，支配的ではない．より自由な組合せを許すことで，可用性が増し，間接的に性能の向上・負荷均衡にも寄与する．ただし，実際のアルゴリズムにおける R_i の選択時，RC に含まれるすべてを一様に選ぶ代わりにこの $R'i$ を優先しても，各ノードの利用頻度が等しければ負荷の均等性は保たれる．また，N が大きい場合，$|RC|$ は非常に大きくなりうるが，RC 生成の規則は単純であるので全体をデータとして持つ必要はない．

図2 に，$N = 6$, $r = 2$ の場合の例を示す．

(i) において W_i に代えて任意の R_i と W_i の和をとったものは，Cheung らの解[7]（格子方式）となる．また，$r = 1$ の場合は投票方式同様 ROWA，$r = N$ の場合は Read-All, Write-One (RAWO) となる．

$N \bmod r \neq 0$ の場合であっても，同様の方法により，$w = \lfloor N/r \rfloor$ とおいて

$$|W_i| = \begin{cases} w+1 & (0 \le i < N \bmod r) \\ w & (N \bmod r \le i < r) \end{cases}$$

となるように構成でき，任意の N, r に対して $|W_i|$ のばらつきはたかだか1の差に抑えられる．均等性の崩れは最小限であり，本質的な破綻を来すことはない．以下では記述の簡略化のため $N \bmod r = 0$ とする．

(ii) $|RC|$ 最小解

(i) では，$|RC|$ に比して $|WC|$ が小さい．読出しの可用度を高めるためにはこれがふさわしいが，同じ r, w で，逆に $|WC|$ を増やして $|RC|$ を限定することも可能である．すなわち，

$$R_i = \{n_{ij} | 0 \le j < r\} \quad (0 \le i < w),$$
$$W_i = \left\{ n_{jk} \middle| \left\lfloor \frac{i}{r^j} \right\rfloor \bmod r = k, \right.$$
$$\left. 0 \le j < w, 0 \le k < r \right\} \quad (0 \le i < r^w)$$

とすれば，

$$|RC| = w, \quad |WC| = r^w,$$
$$w^* = r^{w-1}, \quad r^* = 1$$

となる．ただし $r = 1$ または $r = N$ ならば (i) と同じである．

本論文では書込みどうしの交差性を求めていないが，(ii) 型の WC から互いに交わる W_i を選択すればこの性質を実現することもできる．

(iii) 複合解

上記の解を階層的に組み合わせることで新しい共コテリーを得ることができる．その方法を以下に示す．

$r \bmod q = 0$, $w \bmod v = 0$ となる q, v があるとき，以下のように S をそれぞれ $(r/q) \times (w/v)$ の大きさを持つ $q \times v$ 個のブロックに分割する．

$$\widetilde{S} = \{m_{ij}\} \quad (0 \le i < q, 0 \le j < v),$$
$$m_{ij} = \left\{ n_{kl} \middle| 0 \le k < w, 0 \le l < r, \right.$$
$$\left. \left\lfloor \frac{k}{q} \right\rfloor = i, \left\lfloor \frac{l}{v} \right\rfloor = j \right\}.$$

各ブロック m_{ij} 上の共コテリー $T_{ij} = \langle RC_{ij}, WC_{ij} \rangle$ を，また，ブロックを単位として $\widetilde{S}$ 上の共コテリー $\widetilde{T} = \langle \widetilde{RC}, \widetilde{WC} \rangle$ を，それぞれ上記 (i), (ii) のいずれかの方法によって構成する．ただし各 T_{ij} の構成法は統一しないと均等性が保たれない．

$\widetilde{RC}$ の一要素を $\widetilde{R_i}$ とする．$\widetilde{R_i}$ の要素 m_{jk} のそれぞれについて，m_{jk} 上の RC_{jk} が対応する．$|\widetilde{R_i}| = q$ であるから，q 個の RC_{jk} がある．それぞれの RC_{jk} から要素を1つずつ選び，q 個の和を取ったものを RC

の1つの要素とする．それぞれの $\widetilde{R_i}$ について選び方が $|RC_{jk}|^q$ 通りあるので，総計 $|RC| = |\widetilde{RC}| \cdot |RC_{jk}|^q$ となる．

WC も同様にして構成でき，$|WC| = |\widetilde{WC}| \cdot |WC_{jk}|^v$ である．

したがって，

(iii-i) T_{ij}，$\widetilde{T}$ を (i) 型によって構成するとき

$$|RC_{jk}| = \left(\frac{w}{v}\right)^{\frac{r}{q}}, \qquad |\widetilde{RC}| = v^q,$$

$$|RC| = v^q \cdot \left(\frac{w}{v}\right)^r, \quad |WC| = q \cdot \left(\frac{r}{q}\right)^v.$$

また，

(iii-ii) T_{ij}，$\widetilde{T}$ を (ii) 型によって構成するとき

$$|RC_{jk}| = \frac{w}{v}, \qquad |\widetilde{RC}| = v,$$

$$|RC| = v \cdot \left(\frac{w}{v}\right)^q, \quad |WC| = q^v \cdot \left(\frac{r}{q}\right)^w$$

となる．

いずれの場合も，交差性・最小性はもちろん，均等性および支配性も保たれる．$r \cdot w = N$ より，

$$r^* = \frac{|RC|}{w}, \quad w^* = \frac{|WC|}{r}$$

である．

$$w \le v \cdot \left(\frac{w}{v}\right)^q \le v^q \cdot \left(\frac{w}{v}\right)^r \le w^r,$$

$$r^w \ge q^v \cdot \left(\frac{r}{q}\right)^w \ge q \cdot \left(\frac{r}{q}\right)^v \ge r$$

であるから，r，w を変えずに (i)(ii) の中間的な $|RC|$，$|WC|$ をとることができる．

なお，T_{ij} を (i) 型，$\widetilde{T}$ を (ii) 型によって構成した場合，結果は q によらず，(iii-i) 型において $q = 1$ とした場合と同一となる．また，T_{ij} を (ii) 型，$\widetilde{T}$ を (i) 型によって構成した場合には，(iii-ii) 型において $v = 1$ とした場合と同じになる．

図3 に，$N = 2^{10}$，$r = 2^4$，$w = 2^6$ とした場合の可能な $|RC|$ と $|WC|$ の対を示す．$|RC|$ と $|WC|$ はおおむねトレードオフになるが単純でない．

さらに，ブロックの大きさが同一でない分割も可能である．この場合均等性が崩れるが，アクセス時の R_i，W_i の選択確率の調整によって負荷を均衡させることができる．また，複合化を再帰的に繰返し階層構造とすることもできる．これらによって，十分大きな N に対してほぼ自由に $|RC|$ と $|WC|$ のバランスを変えられる．

この複合化は，一般のコテリーおよび共コテリーに適用可能である．異なる方式によるものを組み合わせることにより，それぞれの特徴を折衷することができる[24]．木構造[2]はこの特別な場合にあたる．

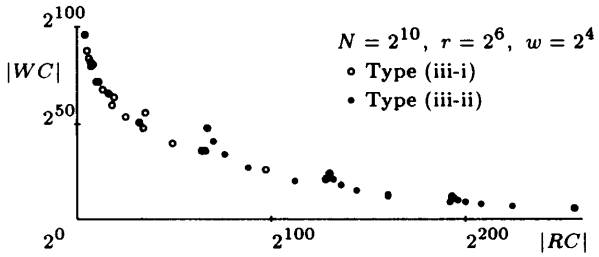

図3 $|RC|$ と $|WC|$ の関係 ($\log_2$)
Fig. 3 $|RC|$ and $|WC|$ ($\log_2$).

複合化はまた，大規模ネットワークを階層的に管理する場合にも応用できる．まず，ブロックごとに管理ノードをおき，それぞれのブロックを代表させる．ただし，管理ノードの故障への対処もブロック内部で行うなどブロックの管理自体も分散化・多重化し，外部に対しては透明化することがより望ましい．このようにしてブロック全体を1個の仮想ノードのごとく扱う．各要求ノードは複製を持つ個々のノードではなくブロックの存在のみを意識し，$\widetilde{T}$ に基づいて各ブロックに対し要求を行う．各ブロックではそれぞれの内部において T_{ij} による合意を得て要求ノードに返せば，その総和が T による合意となる．これによって，非常に多数の複製を持つシステムにおいて，各ノードが全複製の情報を管理する必要がなくなる．メッセージの伝達が段階的になる分の伝達の遅れは出るが，各ノードが扱うメッセージの量を制限できるので全体として規模適応性に寄与する．

4. 既存方式との比較・評価

2.3節での議論に基づいて，この共コテリーを評価し，既存の他方式，特に投票方式と比較する．

集中方式はメッセージ数の点では最善となる．したがって集中方式が妥当であるようなシステムは十分に存在しうるが，負荷分散・耐故障性の面で問題があることは明らかである．トークンによる方式では，負荷をある程度分散するとともに，同一複製上で連続するアクセスのコストを低減できる．しかし，トークンを持つノードの障害に対しては脆弱である．また，前川方式は読み書きの差をつけないため，任意の N および r に対応できる提案方式の方が一般には有利である．格子方式は，提案方式の特殊な場合にあたる．木方式は単独の共コテリーというよりはその組合せと動的切換えの方式であり，提案方式に対しても適用可能

である．ただし，他の動的手法と同様，障害ノードを明示的に検出する必要があり，また，根ノードに負荷が集中する欠点がある．

投票方式では，各ノードの重みが等しいとき，書込み時のノード数 w は任意の r に対し $N-r+1$ である．提案方式では $w=(N/r)$ であるから，同一の N，r で比較すると書込み時のメッセージ数，負荷および並行性においてつねに提案方式の方がすぐれている．また，書込みの可用度でも有利になる．

書込みどうしの交差を求める（書込みと読出しを同時に行う）場合，必要なノード数は $(N/r)+r-1$ となるが，この場合でも $(N/2)>r$ であれば本方式が有利であり，かつ $\lceil N/2\rceil < r$ では投票方式でも書込みどうしの交差が保証されなくなる．

また，読出しの頻度が書込みの α 倍，データそのものの転送コストが制御メッセージのコストの β 倍であるとした場合，$\rho=\alpha/(1+\beta)$ とおくと，操作の平均コストは $\alpha(r+\beta)+w(1+\beta)=(1+\beta)(\rho r+w+\rho\beta)$ にほぼ比例するので，定数成分を除いて $\rho r+w$ を性能指標とすることができる．N を固定したとき，投票方式ではこれは r の一次式になるため，$\rho\geq 1$ であればつねに $r=1$（ROWA）が最適解となり，一般の投票方式の意味を薄れさせている．$\rho\leq 1$ では $r=N$（RAWO）が，$\rho\leq 1$ かつ書込みの交差を求める場合は $r=\lceil N/2\rceil$ が最適である．これに対し提案方式においては，

$$\begin{aligned}\rho r+w&=\rho r+\frac{N}{r}\\&=\rho\left(\sqrt{r}-\sqrt{\frac{N}{\rho r}}\right)^2+2\sqrt{\rho N}\end{aligned}$$

であるので，$r=\sqrt{N/\rho}$，$w=\sqrt{\rho N}$ 付近でコストが最小化される．したがって ρ が既知であれば最適の r を選ぶことができ，与えられた ρ，N に対する最小コストは約 $2\sqrt{\rho N}$ である．ROWA では $\rho+N$，RAWO では $\rho N+1$ であるから，つねに提案方式がすぐれている（ただし $\rho\geq N$ のとき ROWA に，$\rho\leq(1/N)$ のとき RAWO に一致する）．N の増加に従って r，w は $O(\sqrt{N})$ で増加し，投票方式に対するコストの優位が広がると同時に，可用度も読出し・書込みともに向上する．これに対し ROWA では書込みの，RAWO では読出しの可用度がきわめて低く，かつコストが高くなる．

ただし，読出し頻度 ρ を仮定して r を決定したところ実際には ρ' であったとすると，ROWA とのコストの差は，$\sqrt{N}=x$ とおいて

$$\begin{aligned}&\rho'+N-\left(\rho'\sqrt{\frac{N}{\rho}}+\sqrt{\rho N}\right)\\&=x^2-\left(\frac{\rho'}{\sqrt{\rho}}+\sqrt{\rho}\right)x+\rho'\\&=\left(x-\frac{\rho'}{\sqrt{\rho}}\right)(x-\sqrt{\rho})\\&=\left(x-\frac{\rho'+\rho}{2\sqrt{\rho}}\right)^2-\left(\frac{\rho'-\rho}{2\sqrt{\rho}}\right)^2\end{aligned}$$

であるので，$\rho'>\sqrt{\rho N}$ では ROWA の方が有利になり，その差は，

$$N=\left(\frac{\rho'+\rho}{2\sqrt{\rho}}\right)^2$$

であるとき，最大値

$$\left(\frac{\rho'-\rho}{2\sqrt{\rho}}\right)^2$$

をとる．$\rho\geq 1$ の範囲では，ρ を大きめに見積もっておけば最適ではなくとも危険を少なくできる．RAWO との比較も同様．

他方，投票方式は簡潔さと自由度においてすぐれ，同一の N，r における読出しの可用度が最善であるが，r が比較的小さく個々のノードの信頼性が高いならば，本方式においても十分な可用性が保証される．すなわち，ノードの故障率がすべて $p\ll 1$ である場合，読出しが不可能になる確率 $Fr(p)$，書込みが不可能になる確率 $Fw(p)$ は，(i) 型の場合

$$\begin{aligned}Fr(p)&=1-(1-p^w)^r\\&\simeq r\cdot p^w,\\Fw(p)&=(1-(1-p)^w)^r\\&\simeq w^r\cdot p^r\end{aligned}$$

である．ただし，$N \bmod r\neq 0$ ではこれより悪い．**図 4** に $N=32$，$p=0.1$ の場合の r に対する，**図 5** に $r=4$，$p=0.1$ の場合の N に対する，**図 6** に $N=32$，$r=4$ の場合の p に対する，それぞれ $Fr(p)$ および $Fw(p)$ の変化と投票方式との比較を示した．

$Fr(p)$ は，p の減少と r の減少，および N の増加につれて減少する．逆に $Fw(p)$ は一定の r に対して N が増加すると増加するが，投票方式の場合よりゆるやかである．p が小さいなら，$Fr(p)=Fw(p)$ となる付近の r においてその双方ともに小さい値を得ることができる．r の最適値は，一般解としては得られていないが，個々の場合について数値的に求めることは容易である．さらにコストとの両立を考えれば総合的に投票方式より有利といえる．$r>(N/2)$ の領域では $Fr(p)$ が高くなるが，通常このような r をとる必要はないので，現実的な障害とはならない．たとえ

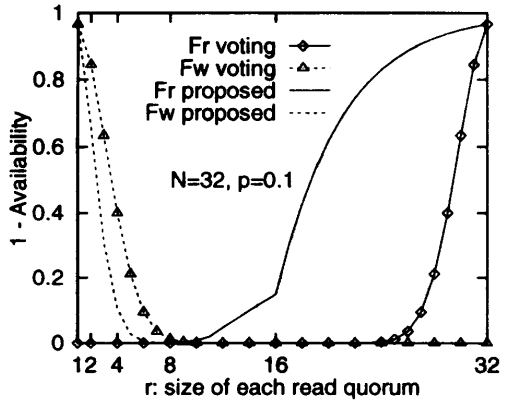

図4 r と $Fr(p), Fw(p)$ の関係
Fig. 4 $Fr(p)$ and $Fw(p)$ versus r.

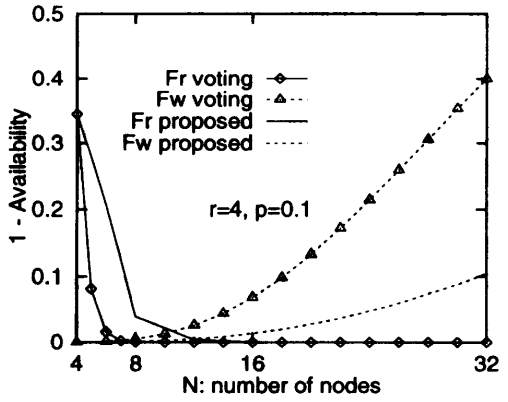

図5 N と $Fr(p), Fw(p)$ の関係
Fig. 5 $Fr(p)$ and $Fw(p)$ versus N.

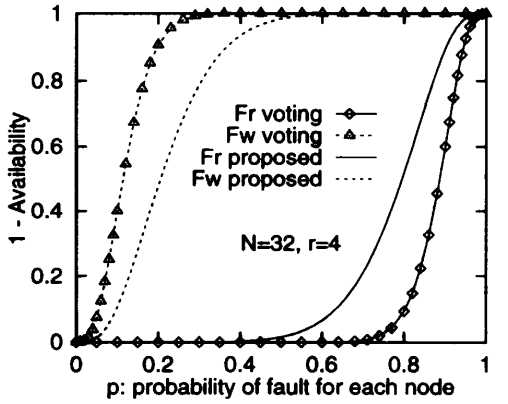

図6 p と $Fr(p), Fw(p)$ の関係
Fig. 6 $Fr(p)$ and $Fw(p)$ versus p.

ば図4の場合，一見 $r = 24$ 付近での不利が大きそうに見えるが，$r = 8$ の本方式では通常の目的には十分な $Fr(p)$ と $Fw(p)$ を得られると同時に，コスト上 $r = 24$ の投票方式に ρ にかかわらず優越する．

同様に，N，w を同一にとった場合には，投票方式より少ない r，低い $Fr(p)$ による一貫性を得られる．また，N の増加によって $Fw(p)$ が減少し，$Fr(p)$ が増加する．このとき $Fw(p)$ は投票方式より高くなるが，書込みの可用度を高めたい場合には，(ii) 型の構成をとれば，

$$\begin{aligned} Fr(p) &= (1-(1-p)^r)^w \\ &\simeq r^w \cdot p^w, \\ Fw(p) &= 1-(1-p^r)^w \\ &\simeq w \cdot p^r \end{aligned}$$

と，$Fr(p)$ が約 r^{w-1} 倍，$Fw(p)$ が約 w^{r-1} 分の1になる．さらに，ブロック化による複合構成では，

(iii-i) 型：

$$\begin{aligned} Fr(p) &= 1-\left(1-\left(1-\left(1-p^{\frac{w}{v}}\right)^{\frac{r}{q}}\right)^v\right)^q \\ &\simeq \frac{r^v}{q^{v-1}} \cdot p^w, \\ Fw(p) &= \left(1-\left(1-\left(1-(1-p)^{\frac{w}{v}}\right)^{\frac{r}{q}}\right)^v\right)^q \\ &\simeq \frac{w^r}{v^{r-q}} \cdot p^r, \end{aligned}$$

(iii-ii) 型：

$$\begin{aligned} Fr(p) &= \left(1-\left(1-\left(1-(1-p)^{\frac{r}{q}}\right)^{\frac{w}{v}}\right)^q\right)^v \\ &\simeq \frac{r^w}{q^{w-v}} \cdot p^w, \\ Fw(p) &= 1-\left(1-\left(1-\left(1-p^{\frac{r}{q}}\right)^{\frac{w}{v}}\right)^q\right)^v \\ &\simeq \frac{w^q}{v^{q-1}} \cdot p^r \end{aligned}$$

のように中間のバランスも設定できる（図3参照）．

これを利用すれば，r，w ともに投票方式と同一にとって N を自由に選ぶとき，多くの場合は $Fr(p)$，$Fw(p)$ ともに投票方式より有利にできる．たとえば $r = 2$，$w = 4$ のとき，投票方式では $N = 5$ で $|RC| = {}_5C_2 = 10$，$|WC| = {}_5C_4 = 5$ であるのに対し，(iii-i) 型の本方式では，$N = 8$，$q = 1$，$v = 2$ とすれば $|RC| = 8$，$|WC| = 4$ とできる．またこの場合，操作に要するメッセージ数は同一でも，N が大きいため本方式はより負荷が低く並行性が高くなる．

メッセージ数，可用度に代わる他の評価基準を採用した場合も，同様に最適の r/w 比を選ぶことができ，N および $|RC|/|WC|$ 比の選択とあわせて，個々のシステムの要求に沿った共コテリーを構成できる．

5. 応　　用

現実のシステムにおいては，個々のノードやリンクの性能，信頼性，利用頻度，さらにそれらの相互関係も考慮しなければならない．本論文の方式は，それらに適応して変形することができる高い一般性を持つ．

特に重要なのはネットワークトポロジである．ネットワークが均一な完全結合でない場合，ノード間には経由しなければならない中継ノード数やリンクの性能によって異なる距離が規定され，アクセスに際しある R_i を選択したときのコストは，単に $|R_i|$ でなく，R_i 内の各ノードへの距離に依存する．各ノードへのメッセージを独立に送信するならば，総コストは全ノードへの距離の総和に比例すると見ることができる．しかし，システムによっては，同一ノード・リンクを経由して送る複数のノードへのメッセージは1つにまとめることができる．その場合，ネットワークを重み付きグラフと見て，R_i と要求ノードを含む最小生成木を発見すればコストが最小になる．ただし，隣接するノード間の距離が等しければ，どの生成木でもコストは等しい．一方，R_i の各ノードへの要求が並行して送られ，かつ負荷が低く要求どうしの衝突が少なければ，操作の遅延はもっぱら R_i 内の最も遠いノードまでの距離によって決まる．読出し時にはこれらの値が最小となる R_i を選択すればよい．W_i についても同様のことがいえる．

いずれの場合も，複製を持つ全ノードを互いの距離の小さいノード群からなるクラスタに分割し，それぞれを R_i の1つにとることによって読出しの性能を高めることができる．最適の分割法の発見は一般の場合には必ずしも容易でないが，多くのシステムではトポロジは何らかの規則を持つので，それを利用した自然な分割が可能である．ネットワークが木であればあるノードの子孫，リングであれば一連の隣接ノード群などがこれにあたる．

特に有効な例は，きわめて多数のノードを含み，ブロードキャストの困難な大規模ネットワークシステムである．そのようなシステムはほとんどの場合階層的な管理構造を持ちサブネットワークに分割されるので，各サブネットワーク（たとえば1つのLAN）内のノード群を R_i に当てはめればよい．サブネットワーク内での操作は，高速・高信頼であるだけでなくブロードキャストやマルチキャストが利用可能であることも多いので，均等性をいくらか犠牲にしても十分な効果が得られる．

(i) 型の共コテリーを用いる場合，R_i の選択時はこの近傍グループのうち読み手/書き手に最も近いものを第一候補とし，グループ内に停止したノードあるいは特に高負荷のノードが存在する場合にその不足分のみを他グループに要求すれば，耐故障性や負荷均衡を損わずに通常操作の効率を上げることができる．

(ii) 型の構成をとる場合には，R_i がグループのみに限られる代わり，書込みはそれぞれのグループから任意に1ノードずつを選択して行えばよい．したがって，書込み時の要求は各グループに対して行い，グループ内でのノードの選択はそれぞれのグループの管理に任せるようにすれば，要求ノードの負担が減るだけでなくより有効な負荷分散が行える．

このようなノード群は，R_i 以外にも W_i あるいは (iii) 型の複合構成時のブロックとして利用することができる．ただし，地理的・物理的に近い（同一電源系など）ノード群は同一の原因によって一斉に停止する危険性も高いので，特に信頼性の高いノード，あるいはグループ外のノードを少数加えておくことで全体の可用性を確保する方法もある．

共コテリーによる複製管理は，ネットワークシステムだけでなく，共有メモリを持たない並列プロセッサシステム，あるいは，特にNUMA型システムなど，共有メモリがあっても，キャッシュないし固有メモリとプロセス間通信の組合せが共有メモリアクセスより安価になる場合にも応用可能である．

並列システムにおいては，非常に多数のプロセッサが協同して1つの作業を行うことがしばしばあるが，隣接するプロセッサ間の結合が比較的高速であるのに対しメモリアクセスはボトルネックとなりがちである．こうした場合，各ノード（プロセッサあるいはプロセッサ群）がそれぞれの固有メモリに共有データの複製を持つことが有効である．ネットワークシステムほど可用性が問題とはされず，少ないメッセージ数と高い並行性が求められることが多いので，r，w の小ささが特に重要である．各ノードがルータを持ち，プロセッサを介さずにメッセージを中継できるシステムでは，他ノードを経由することによる遅れは最小限にできる．ただしこの場合，一般には同一ノード経由のメッセージも別々に送らなければならないので総通信量は増える．また，共コテリーは均等であれば支配的でなくともよい．すなわち前述した近傍グループにあたる最小限の R_i，W_i のみを考えてもよい．動的結合網を持つシステムでは有利なトポロジを設定できるが，静的な結合の場合も高い規則性に従うので効率的なクラスタリングが行える．

たとえばメッシュ結合では，格子状の分割のほかに，行・列をそのまま $R_i \cdot W_i$ に利用することもできる．プロセッサ間の距離を同一とすると，各 R_i の直径，最小生成木のコストともに $r-1$，距離の総和の最大値 $r \cdot (r-1)/2$ である．

2進ハイパーキューブでは，各ノードが n 桁の2進数で表され，1ビット異なるものどうしが結合されて

いる．n 桁中任意の m 桁を選択すると，2^m ノードからなる部分ハイパーキューブが 2^{n-m} 個とれ，どのノードもそれらのうち1つかつ1つのみに属する．逆にその m 桁が一致するものどうしを集めれば，それぞれ 2^{n-m} ノードからなる 2^m 個のクラスタに分割され，どの1つも最初の分割のすべてと交わる．したがってそれぞれの分割を RC, WC にとることができ，各 R_i は直径 m，生成木長 2^m-1，距離の和 $m \cdot 2^{m-1}$ となる．各ノードは自身を含む R_i, W_i にアクセスすればよい．

複製がノードでなくプロセスに所属するものとして，物理的な共有メモリや共有オブジェクトを持たないプロセス群がデータを共有する場合に用いることもできる．

6. むすび

本論文においては，分散システム上で複製されたオブジェクトの一貫性を保つアルゴリズムに応用できる情報構造として，拡張されたコテリー（共コテリー）を定義し，そのいくつかの性質を示した．そして，任意のノード数および任意の読出し数に対し，基本的な構成法とその複合によって得られる，単純ではあるが効果的な共コテリーを与え，これが投票によって実現されるものよりも多くの点ですぐれることを示した．

今後の研究課題の1つとして，投票方式の場合と同様，不均一な分散システムに対するノードの重みの導入とさまざまの基準における最適な重み付けの方法を検討する必要がある．また，構造の動的な再構成によってさらに性能および信頼性を向上させることも考えられる．書込みどうしの交差を求める場合をはじめ，さまざまの条件，評価基準における共コテリーの改善可能性の理論的上限についてもさらに研究の余地がある．

謝辞 本研究に関し重要な示唆を与えてくれた松本眞氏（慶應大）に感謝する．

参考文献

1) Agrawal, D. and El Abbadi, A.: An Efficient and Fault-tolerant Solution for Distributed Mutual Exclusion, *ACM Trans. Comput. Syst.*, Vol.9, No.1, pp.1–20 (1991).
2) Agrawal, D. and El Abbadi, A.: The Generalized Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data, *ACM Trans. Database Syst.*, Vol.17, No.4, pp.689–717 (1992).
3) Barbara, D. and Garcia-Molina, H.: Mutual Exclusion in Partitioned Distributed Systems, *Distributed Computing*, Vol.1, pp.119–132 (1986).
4) Barbara, D. and Garcia-Molina, H.: The Vulnerability of Vote Assignments, *ACM Trans. Comput. Syst.*, Vol.4, No.3, pp.187–213 (1986).
5) Barbara, D., Garcia-Molina, H. and Spauster, A.: Increasing Availability under Mutual Exclusion Constraints with Dynamic Vote Reassignment, *ACM Trans. Comput. Syst.*, Vol.7, No.4, pp.394–426 (1989).
6) Cheung, S.Y., Ahamad, M. and Ammar, M.H.: Optimizing Vote and Quorum Assignments for Reading and Writing Replicated Data, *IEEE Trans. Knowledge and Data Eng.*, Vol.1, No.3, pp.387–397 (1989).
7) Cheung, S.Y., Ammar, M.H. and Ahamad, M.: The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data, *IEEE Trans. Knowledge and Data Eng.*, Vol.4, No.6, pp.582–592 (1992).
8) Courois, P.J., Heymans, F. and Parnas, D.L.: Concurrency Control with "Readers" and "Writers", *Comm. ACM*, Vol.14, No.10, pp.667–668 (1971).
9) El Abaddi, A. and Toueg, S.: Maintaining Availability in Partitioned Replicated Databases, *ACM Trans. Database Syst.*, Vol.14, No.2, pp.264–290 (1989).
10) Garcia-Molina, H. and Barbara, D.: How to Assign Votes in a Distributed System, *J. ACM*, Vol.32, No.4, pp.841–860 (1985).
11) Gifford, D.K.: Weighted Voting for Replicated Data, *Proc. 7th ACM Symp. Operating Systems*, pp.150–162, ACM (1979).
12) Goldman, K.J.: Quorum Consensus in Nested Transaction Systems, *ACM Trans. Database Syst.*, Vol.19, No.4, pp.537–585 (1994).
13) Gupta, A., Bruell, S.C. and Ghosh, S.: Mutual Exclusion on a Hypercube, *J. Parallel and Distributed Computing*, Vol.17, No.4, pp.327–336 (1993).
14) Herlihy, M.: A Quorum-Consensus Replication Method for Abstract Data Types, *ACM Trans. Comput. Syst.*, Vol.4, No.1, pp.32–53 (1986).
15) Herlihy, M.: Concurrency versus Availability: Atomicity Mechanisms for Replicated Data, *ACM Trans. Comput. Syst.*, Vol.5, No.3, pp.249–274 (1987).
16) Herlihy, M.: Dynamic Quorum Adjustment for Partitioned Data, *ACM Trans. Database Syst.*, Vol.12, No.2, pp.170–194 (1987).
17) Ibaraki, T., Nagamochi, H. and Kameda, T.:

Optimal Coteries for Rings and Related Networks, *Distributed Computing*, Vol.8, pp.191–201 (1995).
18) Jajodia, S. and Mutchler, D.: Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database, *ACM Trans. Database Syst.*, Vol.15, No.2, pp.230–280 (1990).
19) Kumar, A.: Cost and Availability Tradeoffs in Replicated Data Concurrency Control, *ACM Trans. Database Syst.*, Vol.18, No.1, pp.102–131 (1993).
20) Lakshman, T.V. and Ghosal, D.: Performance Evaluation of an Efficient Multiple Copy Update Algorithm, *IEEE Trans. Parallel and Distributed Systems*, Vol.5, No.2, pp.217–224 (1994).
21) Maekawa, M.: A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems, *ACM Trans. Comput. Syst.*, Vol.3, No.2, pp.145–159 (1985).
22) Menasce, D.A., Yesha, Y. and Kalpakis, K.: On a Unified Framework for the Evaluation of Distributed Quorum Attainment Protocols, *IEEE Trans.Softw.Eng.*, Vol.20, No.11, pp.868–884 (1995).
23) Rangarajan, S., Jalote, P. and Tripathi, S.K.: Capacity of Voting Systems, *IEEE Trans. Softw. Eng.*, Vol.19, No.7, pp.698–706 (1993).
24) Rangarajan, S., Setia, S. and Tripathi, S.K.: A Fault-tolerant Algorithm for Replicated Data Management, *IEEE Trans. Parallel and Distributed Systems*, Vol.6, No.12, pp.1271–1282 (1995).
25) Sanders, B.A.: The Information Structure of Distributed Mutual Exclusion Algorithms, *ACM Trans. Comput. Syst.*, Vol.5, No.3, pp.284–299 (1987).
26) Spasojevic, M. and Berman, P.: Voting as the Optimal Static Pessimistic Scheme for Managing Replicated Data, *IEEE Trans. Parallel and Distributed Systems*, Vol.5, No.1, pp.64–73 (1994).
27) Tang, J.: On Multilevel Voting, *Distributed Computing*, Vol.8, pp.39–58 (1994).
28) Tang, J. and Natarajan, N.: Obtaining Coteries that Optimize the Availability of Replicated Databases, *IEEE Trans. Knowledge and Data Eng.*, Vol.5, No.2, pp.309–321 (1993).
29) Thomas, R.H.: A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases, *ACM Trans. Database Syst.*, Vol.4, No.2, pp.180–209 (1979).

(平成8年2月5日受付)
(平成8年11月7日採録)

芦原 評（正会員）

1964年生．1987年東京大学理学部情報科学科卒業．1992年同大学大学院理学系研究科博士課程修了．博士（理学）．同年より電気通信大学助手．並列/分散システムに関する研究に従事．ACM会員．

清水謙多郎（正会員）

1957年生．1980年東京大学理学部情報科学科卒業．1985年同大学大学院理学系研究科博士課程修了．理学博士．同年東京大学大型計算機センター助手，1986年東京大学理学部助手，1991年電気通信大学助教授，本職を併任として，1996年より東京大学農学生命科学研究科助教授．オペレーティングシステム，並列/分散処理，生命情報科学の研究に従事．本学会論文誌編集委員．ACM，IEEE，電子情報通信学会，日本ソフトウェア科学会，日本シミュレーション学会，日本生物物理学会各会員．