分散リファレンスカウント法に基づいたGCアルゴリズムの最適化技法

前田宗則[†,☆] 石川裕[†]

本論文では，並列分散環境における自動メモリ管理方式であるIRCM法を提案している．IRCM法は，分散リファレンスカウント法の一方式であるIndirect Reference Counting（IRC法）を基礎とし，IRC法の短所であった，参照を含むミューテータの通信で生じるリファレンスカウント減少メッセージを削減する最適化が行われたものである．このため，IRCM法は，IRC法の主な適用範囲である分散環境のみならず，高性能な並列環境にも適用可能である．IRCM法で導入された削減技術は2つある．1つは，同一プロセッサ上のターゲット参照ごとに送信するべきリファレンスカウント減少メッセージをまとめることで，その送信回数を減らすものである．他方は，配送中のリモート参照の到着確認をリファレンスカウントによらずに行う方法を導入し，従来の到着確認法と併用することでリファレンスカウント減少メッセージの利用を減らすものである．本方式をいくつかのアプリケーションによって評価した結果，分散リファレンスカウント法で最も知られたWRC法と比べて，メッセージ数の点で同等，メッセージ長の点でより優れていることが示された．

Effective Optimizations for Indirect Reference Counting Based Distributed GC

Munenori Maeda[†,☆] and Yutaka Ishikawa[†]

This paper describes an optimized memory management scheme based on Indirect Reference Counting, or IRC, for parallel and distributed environments. Our scheme, called IRCM, enables to reduce reference count decrement messages effectively, so it is applicable to high performance parallel environments as well as distributed environments. The IRCM employs two message reduction techniques for such messages: One is to unite reference count decrement messages for the same target object references. The other is, by introducing an alternative transit reference management scheme, to reduce occasions where the increments of reference counts are necessary to manage transit references, and, thus, to reduce reference count decrement messages. Evaluations in different sample applications result in that the IRCM is fully competitive to the WRC, a major distributed reference counting scheme, because the excessive messages of the IRC comparing with the WRC are reduced more than 90% under the IRCM.

1. はじめに

並列分散環境における自動メモリ管理アルゴリズムに分散リファレンスカウント法というカテゴリがある．これは，逐次環境における従来のリファレンスカウント法に対応し，リモート参照とその参照先オブジェクトの管理を行う．

分散リファレンスカウント法は，参照の含むメッセージの通信処理に，プロセッサ間での同期を必要とせず，ミューテータの実行を長時間中断するような処理を含まないという特色を持つ．そのため，リモート参照管理によってミューテータの実行性能が大幅に劣化することがない．これまでに，このカテゴリに対して，Weighted Reference Counting[1]（WRC）法，Indirect Reference Counting[2),3)]（IRC）法，およびGenerational Reference Counting[4]（GRC）法といったアルゴリズムが提案されている．

WRC法とGRC法は，ミューテータのメッセージの送信側で，メッセージに含まれる参照に参照管理の情報を加える方式を採用している．一方，IRC法では，ミューテータのメッセージの受信側で，メッセージに含まれる参照に対する参照管理の情報を更新するための制御メッセージを必要に応じて送信元に返信する方式を採用している．

† 新情報処理開発機構
Real World Computing Partnership
☆ 現在，富士通研究所
Presently with FUJITSU LABORATORIES LTD.

このように，これらのアルゴリズムではメッセージ長の増大とメッセージ数の増加がトレードオフとなっており，並列処理のように実行性能を追求する場合には，対象とするアプリケーション（領域）やプラットフォームとなる計算機のアーキテクチャによって求められるアルゴリズムが異なってくる．

一方，性能面ではなく，実装の容易さ，保守性やポータビリティといった面に注目するとき，IRC 法は，ミューテータのメッセージに変更を加えないために，通信処理とメモリ管理機構の独立性が高く，他のアルゴリズムに優る特色を持つ．

したがって，制御メッセージの発生を十分に減少させれば，IRC 法は，他の分散リファレンスカウント法と同等以上の性能を発揮し，より実用性の高いアルゴリズムとして適用可能な範囲が広がるであろう．

我々はこれまでに文献 5) で IRC 法に対する制御メッセージ削減の一技法を与えた．本論文では，先の方法とはまったく異なる観点から設計された制御メッセージの削減技法を提案し，2 つの技法を組み合わせることで，さらに高いメッセージ数削減率を達成する．

本論文の構成は次のとおりである．まず 2 章では，本論文で用いられる用語とリモート参照に対する基本操作を定義する．次に，3 章では IRC 法で用いられる参照管理のための構造，制御メッセージ，アルゴリズムを紹介する．これは本論文で提案する IRCM 法の基礎を与えるものである．4 章では，制御メッセージ削減の鍵となる 2 つの技術を述べた後，IRC 法と対比しやすいようにデータ構造，制御メッセージ，アルゴリズムの拡張点を明確にしたうえで IRCM 法を提示する．5 章はいくつかのアプリケーションで性能を評価した結果を示す．6 章はまとめである．

2. リモート参照に対する基本操作

本章では，ガーベージコレクションアルゴリズムで使われる一般的な用語について説明した後，本論文で使用される用語を定義する．その後，リモート参照に対する 3 つの基本操作を定義する．

オブジェクトとは，メモリ空間に割り当てられたデータ構造である．オブジェクトは名前を持ち，その名前を用いて操作される．これをオブジェクトの参照（以下，参照と略記）という．たとえば，参照は，プロセッサとそのプロセッサ内の番地の組で表現される．ミューテータは計算の担い手であり，オブジェクトの生成を依頼し，オブジェクトの内容を書き換え，参照を含むメッセージをプロセッサ間で交換するといった処理を行う．ガーベージとは，ミューテータから決してアクセスされることがないオブジェクトのことをいう．一方，アクセス可能なオブジェクトを使用中のオブジェクトという．ガーベージコレクション，GC，とは，オブジェクトの集合からガーベージを決定し，それらを回収することである．GC のルートとは，各プロセッサごとに定義され，各プロセッサのある時点の計算に含まれる参照の集合である．ミューテータのレジスタや実行スタックに保持される参照の集合がこれにあたる．

以下に，本論文で使用される用語を定義する．

オーナー：オブジェクト o のオーナーとは，o が存在するプロセッサである．

ローカル参照：オブジェクト o について，o のオーナーを P とするとき，P で o の参照をローカル参照という．

リモート参照：P が o のオーナーでないとき，P で o の参照をリモート参照という．

浮遊メッセージ，浮遊参照：配送中のメッセージを浮遊メッセージと呼び，浮遊メッセージに存在する参照を浮遊参照という．

非局所的オブジェクト：オブジェクト o のリモート参照または浮遊参照が存在するとき，o を非局所的オブジェクトという．

使用中のリモート参照：プロセッサ P でオブジェクト o のリモート参照を保持する使用中のオブジェクトが存在するとき，o のリモート参照は P で使用中という．

参照を運ぶミューテータの通信を抽象化することにより，参照に対して以下のような基本操作が定義される．

リモート参照の生成：オーナーからオブジェクト o のローカル参照を含むメッセージが送信されるとき，o のリモート参照が生成されると定義する．

リモート参照の分配：オーナーでないプロセッサからオブジェクト o のリモート参照を含むメッセージが送信されるとき，o のリモート参照が分配されると定義する．

リモート参照の削除：オーナーでないプロセッサ P でオブジェクト o へのリモート参照が使用中でなくなるとき，P で o のリモート参照が削除されたと定義する．

これらの 3 つの基本操作は抽象性が高く，プログラム言語には依存しないものである．なお，第 1 および第 2 の基本操作は，ミューテータのメッセージ送信処理によって引き起こされる．第 3 の基本操作は，ミューテータがオブジェクトの内容を書き換えることで引き

起こされるが，それが確認されるのはガーベージコレクションの実行によってである．

今後の議論の仮定として，ネットワークおよびプロセッサは，信頼性があり，任意の浮遊メッセージは有限時間で相手先プロセッサに到着することとする．

3. Indirect Reference Counting

非局所的オブジェクトに対して，リモート参照の送受信関係に基づいたツリーによって表現するアルゴリズムが独立に発見され提案されている[2),3)]．本論文では，文献2)に従いIndirect Reference Counting（IRC）法と呼ぶことにする．

本章の構成は以下のとおりである．まず3.1節でリモート参照の管理単位とその内部属性について説明する．3.2節では分散環境下でそれらが形づくるツリーを，3.3節はそのツリーと属性の一貫性を保持するための制御メッセージについて説明する．3.4節は参照管理のアルゴリズムである．3.5節ではアルゴリズムの動作を例で示す．3.6節ではIRC法の安全性を保証する定理，3.7節ではローカルガーベージコレクションの手順を示す．

3.1 ORとその属性

IRC法は，非局所的オブジェクトとリモート参照を管理の対象としている．リモート参照を送受信したプロセッサには，そのプロセッサでのリモート参照の存在を表現するデータが割り付けられる．これをObject Reference，OR，と呼ぶ．プロセッサP上のオブジェクトoに対するORを$OR(o)_P$と表現する．文脈上Pが明らかな場合はOR(o)と略記する．

オブジェクトoに対するOR(o)は，プロセッサ内のリモート参照の存在を表現するデータであるので，oのリモート参照を受信していないプロセッサには置かれない．また，同一プロセッサ上に複数の異なったOR(o)が置かれることもない．そのため，各プロセッサにはたかだか1つのOR(o)が存在する．すべてのOR(o)の集合を$\mathcal{OR}(o)$と表記する．

オブジェクトoのオーナーには必ずOR(o)が置かれている．このOR(o)は，他のOR(o)と異なり，oが非局所的であることを明示するために存在する．そのため，他のOR(o)と区別して，oのObject Directory（以下OD(o)と略記）と呼ばれる．他方，OD(o)以外のOR(o)は，oのExternal Reference（以下ER(o)と略記）と呼ばれる．

OD(o)はオブジェクトoが非局所的でなくなった後に削除される．一方，プロセッサPのER(o)は，Pでリモート参照oが使用中でなくなった後にある条件のもとで削除される．このようにOR(o)は動的に生成，削除される対象である．

OD(o)とER(o)には内部属性が定義されている．以下に，それぞれで定義されている内部属性を＜ ＞で括って列挙しておく．これらの属性の意味は次節で説明する．

$$OD(o) :: < RC >$$
$$ER(o) :: < RC, Parent, Presence >$$

3.2 $\mathcal{OR}(o)$上のインバースツリー構造

$ER(o)_P$の属性Parentは，プロセッサPで$ER(o)_P$が生成されたとき，その生成を引き起こしたoの参照の送信プロセッサを保持するためのものである．属性Parentは，ER(o)の生成時に決まり，削除されるまで変わることがない．oの参照の送信プロセッサQには，$OR(o)_Q$が存在している．なぜなら，Qからoの参照を送信するためには，Qはoのオーナーであるか，または，Qでoのリモート参照が使用中でなければならないからである．定義より，前者の場合にはOD(o)が存在し，後者の場合にはER(o)が存在している．ここで，$OR(o)_Q$を$ER(o)_P$の親という．逆に，$OR(o)_P$を$ER(o)_Q$の子という．

子$ER(o)_P$のParentには参照の送信プロセッサQが保持されており，かつ，プロセッサQには$OR(o)_Q$がただ1つしか存在しないことより，子$ER(o)_P$は親$OR(o)_Q$を参照していると見なすことができる．そこで，属性Parentを親ORへの参照という．

$OR(o)_P$の属性RCは，子OR(o)の数を保持するものであり，$OR(o)_P$のリファレンスカウントという．

$ER(o)_P$の属性Presenceは，oのリモート参照がPで使用中かどうかを表す．Presenceが真（偽）であれば，oの参照はPで使用中である（でない）．属性Presenceはリモート参照の存在という．

図1に例示されているように，OR(o)に保持される親ORへの参照によって$\mathcal{OR}(o)$上にインバースツリーが構成される．これがツリーとなる理由は以下のとおりである．

- オブジェクトoへの参照を複数の異なる宛先に送信することで，OR(o)には複数の子OR(o)が存在しうること
- 子OR(o)は生成時に定まるただ1つの親OR(o)への参照しか保持しないこと

IRC法におけるリモート参照管理の概要は次のとおりである．まずプロセッサPからオブジェクトoへの参照をリモートプロセッサQに送信する．Qでその参照が受信されたら，Q上に子OR(o)が生成され

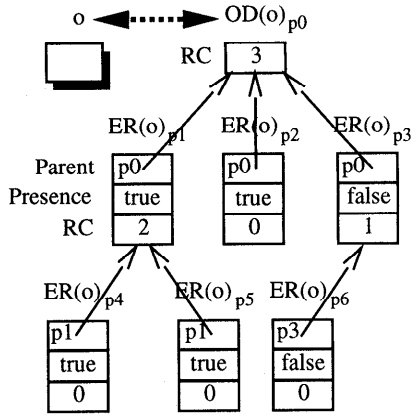

図1 $\mathcal{OR}$(o) 上のインバースツリー
Fig. 1 Inverse tree structure on $\mathcal{OR}$(o).

る．子 OR(o) が削除されたら，親にその削除が通知されることがアルゴリズムで決められており，参照を送信した回数と同数の削除通知が $OR(o)_P$ に届いたら子 OR(o) は存在しないことが保証される．送受信数の差は，リファレンスカウントを用いて監視している．$OR(o)_P$ のリファレンスカウントが 0 であり，かつ，P において o の参照がすでに使用中でなければ，P の OR(o) を削除する．IRC 法の特徴はこのような再帰的な手順にある．o のオーナーには OD(o) が置かれており，OD(o) のリファレンスカウントが 0 であれば，o の参照はオーナー以外のプロセッサ上に存在しないことが保証される．

このようにツリー構造を維持しつつ，子 OR の存在したプロセッサからの削除の通知と自プロセッサでのリモート参照の存在によって OR を削除するかどうかを決定するための手続きでは，リファレンスカウントおよびリモート参照の存在という 2 つの属性を以下のような処理のもとで適正に保持しなければならない．

（リファレンスカウント） $OR(o)_P$ のリファレンスカウントの増加は，ミューテータがリモート参照をプロセッサ P から送信するとき行われる．リファレンスカウントの減少は，他のプロセッサからリファレンスカウントの値を減少させる制御メッセージが到着することで実行される．

（リモート参照の存在） $OR(o)_P$ のリモート参照の存在は，ミューテータがリモート参照を P で受信することによって真となる．リモート参照の存在が偽になるのは，ガーベージコレクションによって，P 上に使用中の o のリモート参照が存在しないことが決定されたときである．

3.3 リファレンスカウント減少メッセージ

IRC 法では，リファレンスカウントの減少は，他のプロセッサからリファレンスカウントの値を減少させる制御メッセージを送ることによって実行される．この制御メッセージは，リファレンスカウント減少メッセージと呼ばれる．

IRC 法では，リファレンスカウント減少メッセージは以下のような 2 つの場合において送信される．

(1) $ER(o)_P$ が削除されたとき
(2) プロセッサ P から送信された o の参照を受信したプロセッサ Q に $OR(o)_Q$ がすでに存在していたとき

(1) の場合では，$ER(o)_P$ の親 OR(o) に向けてリファレンスカウント減少メッセージが送信される．他方，(2) の場合は，参照の送信元 $OR(o)_P$ と受信先 $OR(o)_Q$ が親子にならなかった状況であり，$OR(o)_P$ に向けてリファレンスカウント減少メッセージが送信される．

以下，オブジェクト o に対するリファレンスカウント減少メッセージを decr(o) と記述する．

3.4 アルゴリズム

ここではリモート参照の基本操作のもとでの OR に対する処理とリファレンスカウント減少メッセージの受信処理を定義する．以下，各処理は不可分に実行されることを仮定する．

リモート参照の生成：参照の送信側（オーナー）を P，受信側を Q とする．

送信側：P に OD(o) が存在しないならば，RC=1 として生成する．OD(o) がすでに存在するならば，RC に 1 を加算する．

受信側：Q に ER(o) が存在しないならば，ER(o) を生成し，属性を RC=0，Parent=P，Presence=true と設定する．Q にすでに ER(o) が存在するならば，Presence=true と設定し，リファレンスカウント減少メッセージ decr(o) を P に向けて送信する．

リモート参照の分配：リモート参照の送信側を P，受信側を Q とする．

送信側：P の ER(o) の RC に 1 を加算する．

受信側：Q には OD(o) が存在するか，または ER(o) が存在するか，あるいはどちらも存在しないかである．どちらも存在しないならば，ER(o) を生成し，属性を RC=0，Parent=P，Presence=true と設定する．Q にすでに OR(o) が存在するならば，ER の場合のみ Presence=true と設定し，decr(o) を P

表1 IRC法のもとでの操作例(1)における各OR属性の更新
Table 1 Updates of OR attributes in Example (1) under the IRC scheme.

事象	対象OR	事前状態	事後状態
(1.1) p0からの参照の送信	$OD(o)_{p0}$	RC=3	RC=4
(1.2) p3での参照の受信	$ER(o)_{p3}$	Presence=false	Presence=true
(1.3) p3からのdecrの送信			
(1.4) p0でのdecrの受信	$OD(o)_{p0}$	RC=4	RC=3

表2 IRC法のもとでの操作例(2)における各OR属性の更新
Table 2 Updates of OR attributes in Example (2) under the IRC scheme.

事象	対象OR	事前状態	事後状態
(2.1) p5からの参照の送信	$ER(o)_{p5}$	RC=0	RC=1
(2.2) p0での参照の受信	$OD(o)_{p0}$		
(2.3) p0からのdecrの送信			
(2.4) p5でのdecrの受信	$ER(o)_{p5}$	RC=1	RC=0

表3 IRC法のもとでの操作例(3)における各OR属性の更新
Table 3 Updates of OR attributes in Example (3) under the IRC scheme.

事象	対象OR	事前状態	事後状態
(3.1) p6からのdecrの送信			
(3.2) p6でのORの削除	$ER(o)_{p6}$	Presence=false, RC=0	削除
(3.3) p3でのdecrの受信	$ER(o)_{p3}$	RC=1	RC=0

に向けて送信する.

リモート参照の削除:プロセッサPのER(o)の属性をPresence=falseに設定する.これは3.7節で説明するGC処理によって行われる.

OR(o)の削除条件と削除処理:IRC法のもとでのプロセッサPのOR(o)の削除条件は次のとおり.

$$D_1 \equiv \begin{cases} \text{Presence} = \text{false and RC} = 0 & \\ \qquad\qquad\qquad\qquad \text{ER(o)の場合} & \\ \text{RC} = 0 \qquad\qquad\quad \text{OD(o)の場合} & \end{cases}$$

Pの ER(o) が削除条件を満たすならば,その親Qへdecr(o)を送信した後,削除する.送信と削除は不可分に行う.PのOD(o)が削除条件を満たすならば,それを削除することのみ行う.

リファレンスカウント減少メッセージの受信:decr(o)の受信プロセッサをPとする.PのOR(o)のRCから1を減算する.

3.5 ORに対する操作例

図1のもとで以下のような3つの操作を独立に適用した場合に,各OR(o)の属性がどのように変化するのかを具体的に示す.

(1) オーナーp0からプロセッサp3にリモート参照の生成を行う.

(2) プロセッサp5からオーナーp0にリモート参照の分配を行う.

(3) プロセッサp6の$ER(o)_{p6}$を削除する.

これらの操作例では,参照の送信/受信,ORの生成/削除,decrの送信/受信といったいくつかの事象が引き起こされる.これら生起される事象を生起順に並べ,事象の前後で属性が変化したORにおいて,変化の前後の値を書き並べてみる.ここで,**表1**は操作例(1),**表2**は操作例(2),**表3**は操作例(3)の場合をそれぞれ表形式でまとめたものである.

3.6 IRC法の安全性

IRC法の理論的裏付けは,Dijkstraの拡散計算の停止検出アルゴリズム[6]であるということがTelら[7]によって示されている.以下の定理3.1は,停止検出アルゴリズムの安全性の定理[6]から直接に導出されたものである[7].

定理3.1(IRCの安全性) オブジェクトoのODのリファレンスカウントRCが0であるならば,oは非局所的オブジェクトではない.

3.7 ローカルガーベージコレクション

IRC法では,各プロセッサで独立してGCを実行することが可能である.これをローカルガーベージコレクション,LGC,という.

IRC法のもとでのプロセッサPでのLGCは以下のようになる.以下,Pで任意のORから対応するオブジェクトの参照を決定する関数refの存在を仮定する.すなわち,ref(OR(o))はoの参照である.

(1) Pでリファレンスカウントが正整数であるようなすべてのODを集める.この集合を$\mathcal{OD}_P$と

する．$\mathcal{OD}_P$ から非局所的オブジェクトの参照の集合 $\mathcal{C}_P$ を求める．すなわち，$\mathcal{C}_P = \{\mathrm{ref}(e) \mid e \in \mathcal{OD}_P\}$．

(2) P 上の GC のルートと $\mathcal{C}_P$ の和集合からローカル参照のみをたどって P 上のオブジェクトの閉包を求める．閉包を求める処理は従来の GC と同様である．

(3) 上の閉包に含まれないオブジェクトはガーベージであり，これらをすべて回収する．

(4) 削除されたリモート参照を見つけ，対応する ER にリモート参照の削除操作を適用する．

定理 3.1 により，P で非局所的なすべてのオブジェクトの参照が集合 $\mathcal{C}$ に含まれていると保証される．これより，P でアクセス可能なすべてのオブジェクトは，P 上の GC のルートと $\mathcal{C}$ の和集合から求められたオブジェクトの閉包に含まれている．

4. IRC Modified（IRCM）方式

3.3 節で述べたように，IRC 法では，リファレンスカウント減少メッセージは，生成された ER が削除される場合と受信した参照に対する OR がすでに存在する場合に生成される．一方，他の分散リファレンスカウント法[1),4)]では，リファレンスカウント減少メッセージは生成された ER が削除される場合にのみ生成される．

そこで，他の分散リファレンスカウント法と同程度のリファレンスカウント減少メッセージ数を達成するためには，リモート参照の受信によって生じるリファレンスカウント減少メッセージをできるだけ削減することが必要である．

リファレンスカウント減少メッセージを削減するために IRC 法を変更するにあたっては，以下の点を考慮している．

- メッセージ中の参照に情報を付加しないこと
- OR の削除が不能になったり，大きな遅延が生じないこと

第 1 点は，IRC 法の長所をそのまま引き継ぐことである．メッセージ中の参照に情報を付加するアプローチも考えられるが，IRC 法の最適化との位置付けでこういったアプローチは除いた．第 2 点は，リファレンスカウント減少メッセージの削減や送信の抑制が，OR 自体の削除を減少させないということである．

本章では 2 つのリファレンスカウント減少メッセージの削減技法を提案する．

- リファレンスカウント減少メッセージの送信遅延と統合に基づく削減技法
- リファレンスカウントによらない浮遊参照の到着確認に基づく削減技術

第 1 のアプローチは，同一の OR(o) に対する複数回のリファレンスカウント減少メッセージを 1 つに統合することで削減するものである．

第 2 のアプローチは，オーナーに向けたリモート参照の分配操作によって生成される浮遊参照を，これまでのようなリファレンスカウントを用いた方法によらずに管理するものである．この結果，リファレンスカウント減少メッセージを送信する必要性がなくなる．

本章の構成は以下のとおりである．まず，4.1 節で第 1 のアプローチを，4.2 節で第 2 のアプローチをそれぞれ詳細に検討する．これらの検討を踏まえて，4.3 節で OR の内部属性の拡張を行う．4.4 節ではリファレンスカウント減少メッセージの拡張が行われる．4.5 節はアルゴリズムである．4.6 節では，3.5 節で与えた例題のもとで，IRCM 法のもとでの属性の更新を観察する．4.7 節では，IRCM 法で成立するインバリアントを提示し，それらから IRCM 法の安全性を導く．最後に 4.8 節で，LGC における修正点のみ指摘する．

4.1 OR の重みとリファレンスカウント減少メッセージの統合

プロセッサ P 上のオブジェクト o に対する $\mathrm{OR(o)_P}$ にとって，その親 $\mathrm{OR(o)_Q}$ 以外を宛先とするファレンスカウント減少メッセージの送信を遅延させるならば，複数の親 OR(o) を管理することと同じである．このとき，$\mathcal{OR}$(o) 上のグラフはもはやツリーではなくなる．

グラフ上にサイクルが生じれば，IRC 法では明らかに回収不能な OR(o) が生じる．サイクル構造をさけるためには，各 OR(o) がトポロジ情報を知る必要があり，その情報の獲得のために新たな制御メッセージとプロトコルが必要になる．リダクション減少メッセージを減らすために，そういった制御メッセージを導入することは受け入れがたい．

以上を鑑み，送信を遅延するリファレンスカウント減少メッセージは，単一の親 OR(o) に対するもののみとした．$\mathrm{OR(o)_P}$ の削除条件の 1 つは，子 OR(o) がすべて削除されたことであり，親 $\mathrm{OR(o)_G}$ から何度同一の参照を受信したかには依存しない．そのため，親 $\mathrm{OR(o)_G}$ に送信するべきリファレンスカウント減少メッセージを遅延させても，自 $\mathrm{OR(o)_P}$ の削除条件には影響しない．また，$\mathrm{OR(o)_P}$ から親 $\mathrm{OR(o)_G}$ に対する複数のリファレンスカウント減少メッセージは，$\mathrm{OR(o)_P}$ の削除で送信されるリファレンスカウント減少メッセージと同時にそれらが到着すると仮定しても，

親 $OR(o)_G$ の削除に影響を与えない．

ここで新しい用語と仮定を導入し，OR(o) のリファレンスカウントを再定義する．

有効な OR(o)，無効な OR(o)：OR(o) が有効であるとは，生成されてから削除されるまでの状態をいう．一方，削除された時点をもって，その OR(o) は無効であるという．

OR(o) の重み：任意の OR(o) は重みを持つ．有効な ER(o) の重みは正の整数である．他方，OD(o) と無効な ER(o) の重みは 0 である．

OR(o) のリファレンスカウント：$OR(o)_P$ のリファレンスカウントは，子 OR(o) の重みの総和に等しいかまたは大きい．これらが一致するのは，P に向かう配送中のリファレンスカウント減少メッセージが存在しない場合に限られる．

OR(o) の重みという概念を導入することで，重み n の OR(o) を削除するとき，親に n 個のリファレンスカウント減少メッセージを送信することと，一括で n 減少させる制御メッセージを送信することは同等に扱えるようになる．そこで，こういった制御メッセージを用いれば，$(n-1)$ 個のリファレンスカウント減少メッセージを削除することができる．

4.2 返戻数に基づく浮遊参照の検出

オブジェクトのマイグレーションを考慮しなければ，オブジェクト o のリモート参照の送信時のオーナーと受信時のオーナーは同一である．o のオーナーには必ず OD(o) が存在しており，OD(o) の削除はそれまでに生成されたすべての ER(o) の削除後に行われる．このような OD(o) と ER(o) の生存時間の違いに注目し，リモート参照の分配操作をその宛先によって 2 つの操作に分解して再定義する．

リモート参照の分配：オブジェクト o のリモート参照を含むメッセージが o のオーナー以外に向けて送信されるとき，o のリモート参照が分配されると定義する．

リモート参照の返戻：オブジェクト o のリモート参照を含むメッセージが o のオーナーに向けて送信されるとき，o のリモート参照が返戻されると定義する．

オブジェクト o のリモート参照を分配することで生成される o の浮遊参照や新たな OR(o) については従来どおりのツリー構造で管理することにする．他方，o のリモート参照を返戻することでは，浮遊参照の検出のみが必要であることを考慮し，浮遊参照をツリー構造によらない方法で管理する．これにより，本来ならば返戻操作で行われる参照の送信側のリファレンスカウントの増加処理と，受信側のリファレンスカウント減少メッセージの送信処理を省くことができる．

各プロセッサにおいて，オブジェクト o のリモート参照を o のオーナーに向けて送信した回数を o に対する返戻送信数と定義し，S(o) と書く．他方，o のオーナーが o の参照を受信した回数を o に対する返戻受信数と呼び，R(o) と書く．このとき，返戻された参照が配送中であれば，返戻送信数 S(o) の総和は返戻受信数 R(o) よりも大きい．また，返戻されたすべての参照がオーナーで受信されたときには，明らかに S(o) の総和と R(o) が等しくなる．

ここで，返戻送信数と返戻受信数を区別なく扱うために，o に対する返戻数という用語を定義する．

返戻数：各プロセッサにおいて，オブジェクト o の参照に対する返戻数 M(o) とは，o のオーナー以外では S(o) として定義され，オーナーでは $-R(o)$ として定義される．

返戻数を用いれば，参照の返戻操作におけるインバリアントは次のように書ける．

インバリアント 1 全プロセッサでの返戻数 M(o) の総和は 0 以上である．返戻数 M(o) の総和が 0 になるのは，返戻された o の参照がすべて受信されたときに限られる．

オブジェクト o の返戻数 M(o) は，プロセッサごとに定義される変数であるため，OR(o) の属性とすることが考えられる．ただし，OR(o) は削除されるので，返戻数の総和をつねに保存するためには，削除される OR(o) の返戻数を他の OR(o) の返戻数に加算することが必要になる．そのため，リファレンスカウント減少メッセージを用いて返戻数の情報を運ぶことにする．

リファレンスカウント減少メッセージが返戻数の一部を運ぶとき，運ばれる値を w とすると，送信側および受信側の OR の返戻数は次のように更新される．

送信側：送信プロセッサの M(o) にて，

$$M(o) = M(o) - w.$$

受信側：受信プロセッサの M(o) にて，

$$M(o) = M(o) + w.$$

以上より，無効な ER(o) の返戻数 M(o) はつねに 0 であるように保ち，無効になる前の ER(o) の返戻数は必ず有効な OR(o) の返戻数に加算するようなスキームのもとでは，OD(o) のリファレンスカウントが 0 であって，かつ，OD(o) の返戻数が 0 であらば，o に対する浮遊参照は存在しないことが保証される（4.7 節）．そこで，この結果を浮遊参照の検出に用いることができる．

4.3 拡張された OR の内部属性

これらの削減技法を導入した OR の内部属性は次のようになる．属性 RefWeight は OR の重みを表す．また，属性 MsgCtr は返戻数を表す．以下，IRC 法との違いを強調する部分は下線を引いて示す．

OD(o) :: < RC, MsgCtr >
ER(o) :: < RC, Parent, Presence, MsgCtr, RefWeight >

4.4 拡張リファレンスカウント減少メッセージ

IRCM 法のリファレンスカウント減少メッセージは，IRC 法のリファレンスカウント削除メッセージに引数を追加して拡張したものである．これまでのリファレンスカウント減少メッセージと区別するときは，これを拡張リファレンスカウント減少メッセージと呼ぶ．メッセージの形式は decr(o,m,n) である．第 2 引数 m は返戻数，第 3 引数 n は OR の重みに対応する．

4.5 アルゴリズム

IRCM 法は，IRC 法と同じく (1) リモート参照の基本操作のもとでの OR に対する処理，(2) リファレンスカウント減少メッセージ対する処理，(3) ローカルガーベージコレクション支援処理で定義されている．ここでは，(1) および (2) の処理を定義する．以下，IRC 法と同様に，各処理は不可分に実行されることを仮定する．

リモート参照の生成：参照の送信側（オーナー）を P，受信側を Q とする．

送信側：P に OD(o) が存在しないならば，RC=1，MsgCtr=0 として生成する．OD(o) がすでに存在するならば，RC に 1 を加算する．

受信側：Q に ER(o) が存在しないならば，ER(o) を生成し，属性を RC=0，Parent=P，Presence=true，MsgCtr=0，RefWeight=1 と設定する．Q にすでに ER(o) が存在するならば，Presence=true と設定し，その親が P であるならば，RefWeight を 1 増やす．親が P でなければ，リファレンスカウント減少メッセージ decr(o,0,1) を P に向けて送信する．

リモート参照の分配：リモート参照の送信側を P，受信側を Q とする．

送信側：P の ER(o) の RC に 1 を加算する．

受信側：リモート参照の生成における受信側処理と同一である．

リモート参照の返戻：リモート参照の送信側を P，受信側（オーナー）を Q とする．

送信側：P の ER(o) の MsgCtr を 1 増やす．

受信側：Q の OD(o) の MsgCtr を 1 減らす．

リモート参照の削除：プロセッサ P の ER(o) の属性を Presence=false に設定する．これは LGC によって行われる．

OR(o) の削除条件と削除処理：IRCM 法のもとでのプロセッサ P の OR(o) の削除条件とは次のとおり．

$$D_2 \equiv \begin{cases} \text{Presence} = \text{false and RC} = 0 \\ \qquad\qquad \text{ER(o) の場合} \\ \text{RC} = 0 \text{ and MsgCtr} = 0 \\ \qquad\qquad \text{OD(o) の場合} \end{cases}$$

P の ER(o) が削除条件を満たすならば，その親 Q へ decr(o,m,n) を送信，MsgCtr=0，RefWeight=0 と設定した後，削除する．送信，属性設定，削除は不可分に行う．P の OD(o) が削除条件を満たすならば，それを削除することのみ行う．なお，Q，m，n は，ER(o)_P の属性の再設定前の Parent，MsgCtr，RefWeight に格納されている．

拡張リファレンスカウント減少メッセージの受信：decr(o,m,n) の受信プロセッサを P とする．P の OR(o) の RC から n を減算し，MsgCtr に m を加算する．

4.6 OR に対する操作例

3.5 節で用いた例題を再び取り上げ，IRCM 法では OR(o) の属性がどのように設定されるのかを見てみる．なお，操作前の $\text{OR(o)}_{\text{p}_i}$，$0 \leq i < 7$，の返戻数を m_{p_i} と表す．

まず操作例 (1) は**表 4** のようになる．(1.2) で ER(o)_{p3} の重みが増加し，リファレンスカウント減少メッセージの送信が遅延されている．

続いて操作例 (2) では**表 5** のようになる．参照の返戻操作が行われ，(2.2) では返戻数の利用によってリファレンスカウント減少メッセージの生成が抑制されている．

最後に操作例 (3) では**表 6** のようになる．(3.1) で返戻数の総和を保持するため，ER(o)_{p6} が無効になる前にその返戻数が読み出され，(3.3) で有効な ER(o)_{p3} の返戻数に加算されている．

4.7 IRCM 法のインバリアントと安全性

まず定義からただちに得られるすべての OR(o) に対する不変条件を示す．OD(o) は属性 RefWeight を陽に持たないが，暗黙に RefWeight=0 が仮定されている．

表 4 IRCM 法のもとでの操作例 (1) における各 OR 属性の更新．表 1 と比較のこと
Table 4 Updates of OR attributes in Example (1) under the IRCM scheme, cf. Table 1.

事象	対象 OR	事前状態	事後状態
(1.1)	$OD(o)_{p0}$	RC=3	RC=4
(1.2)	$ER(o)_{p3}$	Presence=false, RefWeight=1	Presence=true, RefWeight=2

表 5 IRCM 法のもとでの操作例 (2) における各 OR 属性の更新．表 2 と比較のこと
Table 5 Updates of OR attributes in Example (2) under the IRCM scheme, cf. Table 2.

事象	対象 OR	事前状態	事後状態
(2.1)	$ER(o)_{p5}$	MsgCtr=m_{p5}	MsgCtr=$m_{p5}+1$
(2.2)	$OD(o)_{p0}$	MsgCtr=m_{p0}	MsgCtr=$m_{p0}-1$

表 6 IRCM 法のもとでの操作例 (3) における各 OR 属性の更新．表 3 と比較のこと
Table 6 Updates of OR attributes in Example (3) under the IRCM scheme, cf. Table 3.

事象	対象 OR	事前状態	事後状態
(3.1)	$ER(o)_{p6}$	MsgCtr=m_{p6}, RefWeight=1	MsgCtr=0, RefWeight=0
(3.2)	$ER(o)_{p6}$	Presence=false, RC=0, MsgCtr=0, RefWeight=0	削除
(3.3)	$ER(o)_{p3}$	RC=1, MsgCtr=m_{p3}	RC=0, MsgCtr=$m_{p3}+m_{p6}$

P_1： RefWeight ≥ 0
P_2： RC ≥ 0

4.5 節で述べた構成方法により，IRCM 法では，オブジェクト o の ER(o) について，RefWeight が 0 になるのは削除されるときのみであり，このとき，RC = 0 かつ Presence = false かつ MsgCtr = 0 が成り立っている．それゆえに，以下のようなインバリアントが成り立つ．

インバリアント 2 任意の ER(o) に対して，

P_3： RefWeight > 0 or (RC $= 0$ and Presence = false and MsgCtr $= 0$)

リファレンスカウント減少メッセージの送信においては，送信側の ER(o) の重み RefWeight は必ず 1 以上減少する．また，返戻数 MsgCtr も任意数の減少がある．そこで，P_1 と P_3 の不変性から，ER(o) に対する以下のようなリファレンスカウント減少メッセージの送信可能条件が導かれる．

G： RefWeight > 1 or (RefWeight $= 1$ and RC $= 0$ and Presence = false)

さらに，$\bar{G} \wedge P_1 \wedge P_2$ より，ER(o) に対する以下のようなリファレンスカウント減少メッセージの送信不能条件が求まる．

$\bar{G}_1$： RefWeight $= 0$ or (RefWeight $= 1$ and (RC > 0 or Presence = true))

オブジェクト o が局所的オブジェクトである必要条件は，すべての ER(o) のリモート参照の存在について Presence = false になることである．$\bar{G}_1$ で Presence = false とおくと，以下の $\bar{G}_2$ を得る．

$\bar{G}_2$： RefWeight $= 0$ or (RefWeight $= 1$ and RC > 0)

この結果と P_2 の不変性から次の条件が導かれる．

P_4： RC $\geq$ RefWeight

P_4 は，すべての ER(o) のリモート参照の存在について Presence = false が成り立ち，かつ，今後どの ER(o) もリファレンスカウント減少メッセージを送信しないという条件のもとで，OD(o) を含めたすべての OR(o) に成り立つ不変条件である．

リファレンスカウント減少メッセージはすべて有限時間で相手先に到着することが保証されている．そのため，リファレンスカウント減少メッセージがすべて受信された状況のもとでは，OR(o) の RC の総和と RefWeight の総和は一致する．ゆえに，P_4 より，各 OR(o) について，RC = RefWeight が成り立つ．この結果から，OD(o) について，RC = 0 が成り立つ．他方，任意の ER(o) について，$\bar{G}_2$ により，RC = RefWeight = 0，または，RC = RefWeight = 1 が成り立つ．$\mathcal{OR}(o)$ は有限で，その上のグラフはサイクルを含まないので，RC = RefWeight = 1 の ER(o) は存在しない．ゆえに，すべての ER(o) について，RC = RefWeight = 0 が成り立つ．

インバリアント 2 より，RC = RefWeight = 0 の ER(o) では MsgCtr = 0 が成立する．4.2 節で述べたように，リファレンスカウント減少メッセージで返戻数を運ぶ処理は，必ず M(o) の総和を保存する．すべ

表 7 各アプリケーションの ER 生成数
Table 7 The numbers of created ERs in the sample applications.

PE	4	8	16	32
TSP(23)	5.90×10^1	1.74×10^2	4.08×10^2	5.85×10^2
NQ(9)	1.75×10^5	2.31×10^5	2.62×10^5	2.80×10^5
BH(2048)	1.42×10^5	3.42×10^5	7.61×10^5	1.48×10^6
OSIM	3.14×10^4	3.66×10^4	4.09×10^4	4.08×10^4

表 8 各アプリケーションの N_{IRC}
Table 8 N_{IRC} in the sample applications.

PE	4	8	16	32
TSP(23)	1.36×10^2	3.50×10^2	7.82×10^2	1.10×10^3
NQ(9)	2.09×10^5	2.17×10^5	2.16×10^5	2.15×10^5
BH(2048)	3.36×10^7	4.57×10^7	5.62×10^7	6.68×10^7
OSIM	4.93×10^5	6.13×10^5	6.74×10^5	7.53×10^5

ての ER(o) の MsgCtr の総和は 0 であるので OD(o) の MsgCtr は M(o) の総和に一致する．インバリアント 1 より，M(o) の総和は 0 以上あり，かつ，0 になるのはすべての返戻された参照が受信されたときである．これより以下の定理を得る．

定理 4.1（IRCM の安全性） オブジェクト o の OD のリファレンスカウントが RC = 0，かつ，返戻数が MsgCtr = 0 であるならば，o は非局所的オブジェクトではない．

4.8 ローカルガーベージコレクションへの修正

定理 4.1 より，IRCM 法での LGC は，IRC 法での非局所的オブジェクトへの参照の集合を取り出す処理のみが修正されたものとなる．

(1) P でリファレンスカウントが正整数または返戻数が 0 でないようなすべての OD を集める．この集合を $\mathcal{OD}_P$ とする．

以下，3.7 節での処理と同一であるため省略する．

5. 性能評価

IRCM 法は，並列オブジェクト指向言語 *OCore*[8] 上に実装された．*OCore* 処理系は様々な並列分散計算機上で稼働しているが，以下の評価では，RWC PC クラスタ 2 号機[9] をプラットフォームとして選択している．RWC PC クラスタ 2 号機は，256 MByte メモリを持つ 200 MHz の PentiumPro をノードプロセッサとし，これらを Myrinet ネットワークでつないだクラスタ計算機である．レイテンシは 7.5 μsec，バンド幅は 119 MByte/sec という性能を持つ．

評価に用いたアプリケーションは，Minimum One Tree を用いて巡回セールスマン問題を解く TSP，レイヤードストリーム法を用いたクイーン配置問題 NQ，Barnes-Hut アルゴリズムによって N 体問題を解く BH，分散時刻管理方式[10] に基づく交通流シミュレータ OSIM である．NQ(n)，TSP(n)，BH(n) の n は，それぞれ，マス目の数，訪問都市数，粒子数を表す．

評価の尺度として，リファレンスカウント減少メッセージの節約率を導入する．リファレンスカウント減少メッセージの節約率とは，IRC 法におけるリモート参照の生成操作と分配操作によって生成されるリファレンスカウント減少メッセージ数を N_{IRC} とし，IRCM 法の技法を用いることで削減されたリファレンスカウント減少メッセージ数を Nd としたとき，

$$100 * Nd/N_{IRC} \quad (1)$$

で定義する．節約率が 0%であればメッセージ効率が IRC 法と同一になり，100%であれば，リモート参照の削除以外でリファレンスカウント減少メッセージは生成されなかったということを表す．節約率が 100%の状況は，他の分散リファレンスカウント法と同程度のリファレンスカウント減少メッセージが生成されていることを意味する．

表 7 は各アプリケーションで生成された ER の総数がプロセッサ台数でどのように変化するかを示している．また，**表 8** は，プロセッサ台数ごとの N_{IRC} の変化である．これらの表により，各アプリケーションにおける通信の規模が示されている．

IRCM 法が利用する 2 つの技法，(1) リファレンスカウント減少メッセージの送信遅延と統合によるメッセージ数削減と (2) 返戻数を用いた浮遊参照管理によるリファレンスカウント減少メッセージの削除は，それぞれ直交しているので，それぞれの有効性を比較するため，(2) のみサポートする場合と (1)，(2) 両方サポートする場合を計測した．また，IRCM 法がプロセッサ台数にどの程度依存するかもあわせて求めてみた．これを**表 9** に示す．2 つの技法を併用した IRCM 法では，これらすべてのアプリケーションで 90%を超えるリファレンスカウント減少メッセージを節約して

表9 リファレンスカウント減少メッセージの節約率
Table 9 Reduction percentages of the reference count decrement messages.

PE		4	8	16	32
TSP(23)	(2)	50.0	50.0	50.0	50.0
	(1,2)	100.0	100.0	100.0	100.0
NQ(9)	(2)	77.2	80.2	82.4	84.3
	(1,2)	94.4	93.6	94.1	94.5
BH(2048)	(2)	99.1	98.6	97.8	96.1
	(1,2)	99.9	99.9	99.9	99.9
OSIM	(2)	52.9	51.9	50.3	46.4
	(1,2)	99.9	99.8	99.8	99.8

表10 十分大きなヒープ空間での各アプリケーションの実行時間(秒). RWC PC クラスタ2号機, 32プロセッサ利用時. [ ] 内は, 自動メモリ管理なしの実行時間を1とした比
Table 10 Execution times (seconds) of the sample applications with large heap to suppress LGCs using 32 processors of RWC PC-Cluster II. Bracketed numbers present proportions to a comparable execution time which excludes any overhead in the automatic memory management.

	IRC	IRCM	自動メモリ管理なし
BH(2048)	74.45	56.43	50.21
	[1.483]	[1.124]	[1.000]
OSIM	9.139	7.217	6.812
	[1.342]	[1.059]	[1.000]

いる.

表10は, N_{IRC} の大きい2つのアプリケーションでの実行時間を示したものである. IRC法とIRCM法における実行では, それぞれのアプリケーションに十分なサイズのヒープ空間の割り当てることにより, LGCをまったく行わないようにした. そのため, 自動メモリ管理機構のオーバヘッドをすべて除いた場合の実行時間と比べることで, リモート参照管理のオーバヘッドが明らかになる. これらのアプリケーションでは, IRC法とIRCM法では実行時間にして30%程度が改善されている.

表11と**表12**は各アプリケーションでどういったインバースツリーが生成しているか, 木の深さと幅に注目して表にしたものである. 表12の括弧内の数字は, 1項目にまとめられた幅の範囲である. ここで注目されるのは, これらのアプリケーションに共通して, 深さ1, 幅1のツリーが多く存在することと, (全プロセッサ数 -1) に対応する31の幅が多いことである. なお, ODのRCは動的に変化するため分散環境下で正確に計測することは難しい. そこで, ODのRCが0になるまでに受信した, ルート直下のERの削除で生じたリファレンスカウント減少メッセージ数を用いて近似した.

表11 各アプリケーションのERの深さ分布(%)
Table 11 Distribution of the depth of ERs in the sample applications (%).

深さ	1	2	3–5	> 5
TSP(23)	100.0	–	–	–
NQ(9)	68.77	27.14	4.09	–
BH(2048)	99.91	0.05	0.03	–
OSIM	93.70	2.04	4.26	–

表12 各アプリケーションのODのRC分布(%). [ ] 内の数字は1と31以外のRCの最小値と最大値を示す
Table 12 Distribution of reference counts of ODs in the sample applications (%). Bracketed numbers present min-max reference counts other than cases of 1 and 31.

RC	1	31	その他
TSP(23)	99.81	0.18	–
NQ(9)	74.47	–	25.53 [2-8]
BH(2048)	71.25	27.06	1.68 [5-5]
OSIM	94.68	1.48	3.84 [2-11]

並列分散オブジェクト指向プログラミングでは, オブジェクトoのオーナーにメッセージを送信することでリモートメソッド実行を行うことが多い. このとき, メッセージには対象となるoのリモート参照が含まれることになるので, リモート参照の返戻が行われることになる. 技法(2)が有効になるのはこのような場合であり, 広く適用可能な方法であると思われる. NQとBHはこの実例となっている.

また, ブロードキャストやリダクションといった操作で参照を全プロセッサに通知する処理を繰り返し行う場合には, 各々の通知は論理的には1対1の通信であるので深さ1, 幅が (全プロセッサ数 -1) となるツリーを構成する. この結果, 技法(1)が有効になる. OSIMはこの実例となっている.

深さ1かつ幅1のツリーでは, その上での通信は必ず技法(1)か(2)で最適化可能である. 評価に用いたアプリケーションはいずれもこういったツリーが多数を占めていた. そのため, 十分な最適化の効果が得られたと思われる.

6. まとめ

本論文では, IRC法を基礎として, リモート参照の基本操作で生じるリファレンスカウント減少メッセージを削減するIRCM法を提案した. IRCM法で導入された削減技術のキーポイントは次の2つである.

(1) リファレンスカウントの減少メッセージの送信遅延と統合
(2) リモート参照の送受信関係によらない浮遊参照管理の導入

第1点目は，リモート参照を同一の宛先に n 回の送信した場合に生じる n 個のリファレンスカウント減少メッセージを1つに統合するものであった．これを実現するために，重み n を持つ OR という概念とリファレンスカウントを一括して n 減少させる拡張されたリファレンスカウント減少メッセージが導入された．

第2点目は，リモート参照の送信から宛先が参照先オブジェクトのオーナーである場合を別扱いし，従来のリモート参照の送受信関係による参照管理とは異なる枠組みで，浮遊中のリモート参照の管理を行うものであった．オーナーに向かう浮遊参照の検出には，参照の送信回数が受信回数がいつか一致することに基づく技法が導入された．

本アルゴリズムを並列オブジェクト指向言語 *OCore* の自動メモリ管理機構として実装した．これを RWC PC クラスタ2号機の上でいくつかのアプリケーションを用いて評価を行った．評価で用いられたアプリケーションに対しては，IRCM 法は，リモート参照の送信で生じるリファレンスカウント減少メッセージの90%以上を削減することができた．この結果は，IRCM 法が WRC 法と比べてメッセージ数の点で同等，メッセージ長の点でより優れていることを意味する．これにより，IRCM 法は，実用上十分な有効性を持つことが分かった．

謝辞 RWCP 並列分散システムソフトウェア研究室の皆様，および RWCP 並列分散システムパフォーマンス研究室の松田元彦氏には，日頃より様々なご助言をいただき深謝いたします．また，本論文に対し有益なコメントをいただきました査読者の皆様に感謝いたします．

参 考 文 献

1) Bevan, D.I.: *Distributed Garbage Collection Using Reference Counting*, Lecture Notes in Computer Science, Vol.259, pp.176–187, Springer-Verlag (1987).
2) Piquer, J.M.: Indirect Reference Counting: A Distributed Garbage Collection Algorithm, *Proc. PARLE '91 Conference*, Lecture Notes in Computer Science, Vol.505, pp.150–165, Springer-Verlag (1991).
3) Ichisugi, Y. and Yonezawa, A.: Distributed Garbage Collection Using Group Reference Counting, *Software Science and Engineering*, Nakata, I. and Hagiya, M. (Eds.), Vol.31, pp.212–226, World Scientific (1991).
4) Goldberg, B.: Generational Reference Counting: A Reduced-Communication Distributed Storage Reclamation Scheme, *ACM SIGPLAN Notices*, Vol.24, pp.313–321 (1989).
5) 前田宗則，清水友晴，小中裕喜，石川 裕：分散 GC アルゴリズムの実装，日本ソフトウェア科学会第 13 回大会論文集，pp.161–164，日本ソフトウェア科学会 (1996).
6) Dijkstra, E.W. and Scholten, C.S.: Termination Detection for Diffusing Computations, *Information Processing Letters*, Vol.11, No.1, pp.1–4 (1980).
7) Tel, G. and Mattern, F.: The Derivation of Distributed Termination Detection Algorithms from Garbage Collection Schemes, *ACM Trans. Programming Languages and Systems*, Vol.15, No.1, pp.1–35 (1993).
8) 小中裕喜，石川 裕，前田宗則，友清孝志，堀敦史：超並列オブジェクトベース言語 *OCore* の並列計算機上での実装，情報処理学会論文誌，Vol.36, No.7, pp.1520–1528 (1995).
9) 手塚宏史，堀 敦史，石川 裕，曽田哲之，原田 浩，古田 敦，山田 努：PC とギガビット LAN による PC クラスタの構築，情報処理学会研究報告 96-ARC-119，pp.37–42 (1996).
10) 阿部一裕，古市昌一，尾崎敦夫，田中秀俊，中島克人：空間的影響範囲を考慮した分散論理時刻管理方式，並列処理シンポジウム JSPP'97 論文集，pp.53–60，情報処理学会 (1997).

(平成 10 年 9 月 7 日受付)
(平成 11 年 3 月 5 日採録)

前田 宗則（正会員）

昭和 39 年生．平成元年大阪大学大学院基礎工学研究科物理系専攻修士課程修了．同年富士通（株）入社．平成 4～10 年技術研究組合新情報処理開発機構に出向．現在（株）富士通研究所に所属．

石川 裕（正会員）

昭和 35 年生．昭和 62 年慶應義塾大学大学院理工学部電気工学科博士課程修了．工学博士．同年電子技術総合研究所入所．昭和 63～平成元年カーネギー・メロン大学客員研究員．現在技術研究組合新情報処理開発機構に出向中．