

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4064404号
(P4064404)

(45) 発行日 平成20年3月19日 (2008. 3. 19)

(24) 登録日 平成20年1月11日 (2008. 1. 11)

(51) Int. Cl.	F I
G 0 6 F 12/00 (2006. 01)	G 0 6 F 12/00 5 9 1
G 0 6 F 9/44 (2006. 01)	G 0 6 F 9/44 5 3 0 S

請求項の数 6 (全 10 頁)

(21) 出願番号	特願2005-10862 (P2005-10862)	(73) 特許権者	390019839
(22) 出願日	平成17年1月18日 (2005. 1. 18)		三星電子株式会社
(65) 公開番号	特開2005-216298 (P2005-216298A)		S a m s u n g E l e c t r o n i c s
(43) 公開日	平成17年8月11日 (2005. 8. 11)		C o . , L t d .
審査請求日	平成17年1月18日 (2005. 1. 18)		大韓民国京畿道水原市靈通区梅灘洞 4 1 6
(31) 優先権主張番号	2004-005469	(74) 代理人	100064908
(32) 優先日	平成16年1月28日 (2004. 1. 28)		弁理士 志賀 正武
(33) 優先権主張国	韓国 (KR)	(74) 代理人	100089037
			弁理士 渡邊 隆
		(74) 代理人	100108453
			弁理士 村山 靖彦
		(74) 代理人	100110364
			弁理士 実広 信哉

最終頁に続く

(54) 【発明の名称】 適応型ガーベージコレクション方法及び前記方法を行う装置

(57) 【特許請求の範囲】

【請求項 1】

動的メモリを割り当てる方法において、

所定のプロセスの要請により第 1 メモリ領域に新しいオブジェクトのための領域が割り当てられる (a) 段階と、

前記第 1 メモリ領域に割り当てられたオブジェクトのうち前記プロセスにより使用中のオブジェクトを第 2 メモリ領域にコピーする (b) 段階と、

前記第 1 メモリ領域及び前記第 2 メモリ領域の大きさが、所定の情報によって調節される (c) 段階と

を含み、

前記情報は、前記第 1 メモリ領域にオブジェクトが割り当てられた領域の大きさと、前記第 2 メモリにコピーされたオブジェクトが割り当てられた領域の大きさととの比の履歴の平均であることを特徴とするガーベージコレクション方法。

【請求項 2】

前記プロセスは、ユーザーアプリケーションにより生成されたプロセスであることを含むことを特徴とする請求項 1 に記載のガーベージコレクション方法。

【請求項 3】

前記プロセスはシステムアプリケーションにより生成されたプロセスであることを含むことを特徴とする請求項 1 に記載のガーベージコレクション方法。

【請求項 4】

10

20

動的メモリの割り当てにおいて、所定のプロセスの要請により新しいオブジェクトを割り当てるための第1メモリ領域と、

前記第1メモリ領域に割り当てられたオブジェクトのうち前記プロセスにより使用中のオブジェクトをコピーして割り当てるための第2メモリ領域と、

前記プロセスにより活性化されて、前記メモリ領域にオブジェクトを保存するための領域を割り当て、前記第1メモリ領域及び前記第2メモリ領域の大きさを所定の情報によって調節した後、前記プロセスに制御権をわたして自身は非活性化されるガーベージコレクションモジュールと

が搭載され、

前記情報は、前記第1メモリ領域にオブジェクトが割り当てられた領域の大きさと、前記第2メモリにコピーされたオブジェクトが割り当てられた領域の大きさととの比の履歴の平均であることを特徴とするガーベージコレクション機能を提供する装置。

10

【請求項5】

前記プロセスは、ユーザーアプリケーションにより生成されたプロセスであることを含むことを特徴とする請求項4に記載のガーベージコレクション機能を提供する装置。

【請求項6】

前記プロセスは、システムアプリケーションにより生成されたプロセスであることを含むことを特徴とする請求項4に記載のガーベージコレクション機能を提供する装置。

【発明の詳細な説明】

【技術分野】

20

【0001】

本発明はガーベージコレクションに関し、より詳細にはセミスペース・コピーイング・コレクションの構造において、スペース境界線の位置が変化する適応型非対称スペース構造を利用した適応型ガーベージコレクションに関する。

【背景技術】

【0002】

ガーベージとは、プログラミングの観点からメモリ空間内でもう不要であると判断されて運用体制により回収されたデータ保存空間をいう。また、ガーベージコレクション（Garbage collection：以下、GC）とは、このようなメモリ空間を他のプログラムが効率的に再使用できるように解放することをいい、‘ゴミコレクション’ともいう。すなわち、ガーベージコレクションとは、メモリを動的に管理するための方法をいう。

30

【0003】

過去には、場合によっては使用しない保存空間を正しく返却しないプログラムは、運用体制により強制的に終了されることもあったが、一般的にはプログラムが必要とする保存空間を明示的に宣言し、その空間がもう不要となる時には、システムに返却するようにプログラムを作成しなければならなかった。

【0004】

したがって、このような作業を正しく行わなければ、正常に行われていたプログラムも原因も知らずに止まってしまうが、このような場合には、論理的なバグでもために原因を捜し出すことも非常に困難である。したがって、システム内で自動的に使用中でないメモリ領域を探して解放するGC機能が開発された。

40

前記GCに関する一般的なアルゴリズムには、セミスペース・コピーイング・コレクション方法と、世代別GC方法とがあるが、それについて簡単に説明する。

【0005】

図1は、従来のセミスペース・コピーイング・コレクション方法を示す例示図である。

前記図1に図示されたように、メモリ領域は、同じ大きさのfromスペース及びtoスペースに分割される。この時、前記fromスペースは、オブジェクトが割り当てられるメモリ領域を示し、前記toスペースは、GCにより前記fromスペースに割り当てられたオブジェクトがコピーされるメモリ領域を示す。一方、前記toスペースとfro

50

mスペースとは、その大きさが同一でなければならない。なぜなら、もし、GCが行われる時、fromスペースに存在するオブジェクトがすべて生きている（使用中の）オブジェクトである場合に、あらゆるオブジェクトがtoスペースにコピーされねばならず、また、典型的なセミスペース・コピーイング・コレクションアルゴリズムが行われるためには、ポインタ逆転のために一つのオブジェクトに対して二つのコピー本が臨時で存在しなければならないからである。これに関しては非特許文献1に開示されている。

【0006】

また、前記オブジェクトは、オブジェクト向けプログラミングにおいて特定クラスまたはそのクラスの自体方法や手順、そしてデータ変数を有しているサブクラスが実際に具現されたもの、すなわち、用例をいう。

10

1段階から3段階まで前記fromスペースにオブジェクトが割り当てられるが、3段階で前記fromスペースが満たされてそれ以上オブジェクトを割り当てられなくなる。したがって、4段階ではfromスペースに割り当てられたオブジェクトのうちもう使用中でないオブジェクト、すなわち、ガーベージを除いて生きているオブジェクトを前記toスペースにコピーする。このようにすることにより、以前にfromスペースであった領域はtoスペースに、toスペースであった領域はfromスペースに役割が変わる。したがって、5段階からは、前記1段階のようにfromスペースにオブジェクトが割り当てられる。

【0007】

一般的に、前記セミスペース・コピーイング・コレクション方法は、メモリ動的管理方法において単独で使われるよりは、他のメモリ動的管理方法に応用されるが、その例として世代別GC方法がある。

20

図2は、従来の世代別GC方法を示す例示図である。

【0008】

前記世代別GC方法は、1980年代以後に効率的なGC方法として広く使われるアルゴリズムであって、メモリヒープはオブジェクトの年齢によって二つ以上の世代領域に分割される。前記図2では、若い世代領域200と老いた世代領域250とに分けられている。また、前記若い世代領域200では、前記図1に図示されたようなセミスペース・コピーイング・コレクション方法によりGCが行われるためにfromスペース210とtoスペース220とが存在する。

30

【0009】

この時、前記オブジェクトの年齢とは、オブジェクトが一つのセミスペースから他のセミスペースにコピーされる回収をいう。したがって、例えば、あるオブジェクトが数回コピーされたとすれば、前記オブジェクトは‘老いた’といえる。

以下、従来の世代別GC方法を具体的に説明する。

若い世代領域200でのGC方法としては、セミスペース・コピーイング・コレクション方法が使われ、老いた世代領域250でのGC方法としては表示圧縮方法が使われる。

【0010】

新しいオブジェクトは、若い世代領域200であるセミスペースのうちfromスペース210に割り当てられる。前記図2に、fromスペース210に割り当てられたオブジェクト212、214、216、218を示す。

40

若い世代領域200でのGCは、fromスペース210からtoスペース220に生きているオブジェクト212、216だけコピーすることによって行われ、若い世代領域200に対するGCが数回行われる間に生きているオブジェクトは老いたオブジェクトに分類されつづける。このような老いたオブジェクト218は、老いた世代領域250に移転される。

【0011】

老いた世代領域250にそれ以上オブジェクトを移転する空間がない場合、老いた世代領域250に対するGCが行われ、一般的に若い世代領域200でのGCに比べて長時間

50

を要する。

前記図1に図示されたセミスペース・コピーイング・コレクション方法の場合には、二つのセミスペース領域のうち一つの領域だけがメモリ割り当てのために使われる。すなわち、GCはメモリ領域が満たされた時に行われるため、メモリ領域が小さいほどGCはさらに頻繁に発生するという問題点がある。

【0012】

また、前記図2に図示された世代別GC方法は、若い世代領域で主にセミスペースコピーイング方法を使用するために依然として前記のような問題点が存在し、フローティングガーベージの量が増加してしまう問題点がある。この時、前記フローティングガーベージとは、若い世代領域でのGCを通じて老いた世代領域に移動したオブジェクトが、実際にはもう使用中でないが、老いた世代領域でのGCがまだ発生しなかったためにメモリ領域にそのまま存在するということをいう。若い世代領域でのGCが頻繁に発生するほど、より多くのオブジェクトが老いた世代領域に移動するようになり、その結果、フローティングガーベージの量も増加してしまうので、全体的にメモリ効率が悪くなる。

10

【0013】

【特許文献1】特開平3-189747号公報

【特許文献2】特開2002-278828号公報

【非特許文献1】Cheneyの‘A nonrecursive list compaction algorithm, Communications of the ACM, 13(11), pp677-678, 1970’

20

【発明の開示】

【発明が解決しようとする課題】

【0014】

本発明は前記問題点を改善するために案出されたものであって、本発明は、セミスペース・コピーイング・コレクション方法を行うために必要とするスペースの大きさをそれぞれ別にすることによって、より効率的なGCが行われる方法を提供するところにその目的がある。

本発明の目的は、前記で言及した目的に限定されず、言及しなかったさらに他の目的は下の記載から当業者ならば明確に理解できる。

【課題を解決するための手段】

30

【0015】

前記目的を達成するために、動的メモリを割り当てる方法において、本発明の実施によるGC方法は、所定のプロセスの要請により第1メモリ領域に新しいオブジェクトのための領域が割り当てられる(a)段階と、前記第1メモリ領域に割り当てられたオブジェクトのうち前記プロセスにより使用中のオブジェクトを第2メモリ領域にコピーする(b)段階と、前記第1メモリ領域及び前記第2メモリ領域の大きさが所定の情報によって調節される(c)段階とを含む。この時、望ましくは、前記プロセスは、ユーザーアプリケーションにより生成されたプロセスであるか、またはシステムアプリケーションにより生成されたプロセスである。

【0016】

40

望ましくは、前記情報は、前記(a)段階及び(b)段階が行われつつ発生する履歴情報である。

望ましくは、前記情報は、前記第1メモリ領域にオブジェクトが割り当てられた領域の大きさと、前記第2メモリにコピーされたオブジェクトが割り当てられた領域の大きさととの比である。

【0017】

前記目的を達成するために、動的メモリの割り当てにおいて、本発明の実施によるガーベージコレクション機能を提供する装置には、所定のプロセスの要請により新しいオブジェクトを割り当てるための第1メモリ領域と、前記第1メモリ領域に割り当てられたオブジェクトのうち前記プロセスにより使用中のオブジェクトをコピーして割り当てるための

50

第2メモリ領域と、前記プロセスにより活性化されて、前記メモリ領域にオブジェクトを保存するための領域を割り当て、前記第1メモリ領域及び前記第2メモリ領域の大きさを所定の情報によって調節した後、前記プロセスに制御権をわたして自身は非活性化されるガーベージコレクションモジュールとが搭載された。この時、望ましくは、前記プロセスは、ユーザーアプリケーションにより生成されたプロセスであるか、またはシステムアプリケーションにより生成されたプロセスである。

【0018】

望ましくは、前記情報は、前記プロセスの要請により前記第1メモリ領域に新しいオブジェクトのための領域が割り当てられ、前記第1メモリ領域に割り当てられたオブジェクトのうち前記プロセスにより使用中のオブジェクトが前記第2メモリ領域にコピーされながら発生する履歴情報である。

10

望ましくは、前記情報は、前記第1メモリ領域にオブジェクトが割り当てられた領域の大きさと、前記第2メモリにコピーされたオブジェクトが割り当てられた領域の大きさとの比である。

【発明の効果】

【0019】

本発明の実施によって世代別GCを行うに当たって、同じ大きさのメモリを使用した場合、従来には若い世代領域に割り当てられた領域の半分だけをメモリとして割り当てることができたが、本発明を通じて半分以上、一般的には大部分の若い世代領域に割り当てられた領域をメモリとして割り当てられることによってメモリ使用効率が増加する効果がある。

20

【0020】

また、さらに大きいメモリ割り当て領域を有するために、単位時間当りガーベージコレクタが行われる回数が減少し、これに伴い従来技術の問題点であったフローティングガーベージの量も減少するようになってメモリ使用効率が向上する効果がある。

また、メモリ使用効率の向上の結果として、ガーベージコレクタの最も大きい性能上指標であるGC時間の総和もやはり従来の世代別ガーベージコレクタに比較して短縮される効果がある。

【発明を実施するための最良の形態】

【0021】

その他の実施形態の具体的な事項は、詳細な説明及び図面に含まれている。

30

本発明の利点及び特徴、そしてそれらを達成する方法は添付される図面と共に詳細に後述される実施形態を参照すれば明らかになる。しかし、本発明は以下の実施形態に限定されるものではなく多様な形態に具現でき、但し、本実施形態は本発明の開示を完全にし、当業者に本発明の範囲を完全に知らせるために提供されるものであり、本発明は特許請求の範囲により定義されるだけである。明細書全体にわたって同一参照符号は同一構成要素を示す。

【0022】

以下、添付された図面を参照して本発明の望ましい実施形態を詳細に説明する。

図3は、本発明の実施による適応型世代別GC方法のためのメモリ構造を示す例示図である。

40

前記図3に図示されたように、GCのために、ヒープのようなメモリ領域は、老いた世代領域350と若い世代領域300とに分割され、前記若い世代領域300は、スペース境界線320を中心としてfromスペース310とtoスペース330とに分けられる。この時、前記スペース境界線320は、前記二つのスペースを均等に分けることなく、自由に移動可能になっている。

【0023】

図4は、本発明の実施による若い世代領域でのGCを行う方法を示す一実施形態の処理フローチャートである。

GCを行うガーベージコレクタ400は、システム内で動作するプロセッサであって、

50

ユーザプロセス470との相互作用を通じてGC機能を行う。すなわち、前記ガーベージコレクタ400は、ユーザプロセス470からメモリ割り当てを要請されてメモリを割り当て、メモリが足りない場合、自身が管理するメモリヒープ内でもう使用中でないガーベージを除去してメモリを確保する独立的なモジュールである。また、前記ガーベージコレクタ400は、メモリの割り当て及び解除を自動的に行うために、あらゆるプロセッサに連結されてメモリ管理者として使用でき、特にJava（登録商標）をはじめとする仮想マシンにも適用できる。

【0024】

ガーベージコレクタ400は、ユーザプロセス470から新しいオブジェクトのためのメモリ割り当てを要請を受ければ動作を行い始める（S410）。それから、前記オブジェクトのための新しいメモリ空間を割り当てるためにfromスペースに余裕があるかどうかを検査する（S420）。もし、前記fromスペースに余裕がある場合には前記ユーザプロセス470にfromスペースのうち要請されたオブジェクトの大きさだけ割り当て、再びユーザプロセス470に制御権を渡す。

10

【0025】

もし、前記S420段階で、fromスペースに余裕がない場合には若い世代領域に対するGC動作が行われる（S430）。

この時、もし、t oスペースの空間が不足すれば、まだ処理され終わらずにfromスペースに残っているオブジェクトはfromスペース内で公知の表示-圧縮方法により処理される（S450）。

20

【0026】

前記ガーベージコレクタ400によりGC作業が正常に終了すれば、前記ガーベージコレクタ400は、本発明の特徴であるfromスペースとt oスペースとを分けるスペース境界線を再設定する（S460）。このために、前記ガーベージコレクタ400は所定の予測値を持ってfromスペースとt oスペース間の境界線を再設定する。

【0027】

以下では、スペース境界線を再設定する方法を具体的に説明する。

従来にはfromスペースとt oスペースとの大きさが同一であったが、本発明では、fromスペースとt oスペースとの境界が適応的に計算される。一般的な場合、fromスペースに割り当てられた大部分のオブジェクトがもう不要なガーベージに分類されて削除され、一部分だけがt oスペースにコピーされる。したがって、GCが終わる度にfromスペースがt oスペースよりさらに大きくなるようにスペース境界線を再設定する。

30

【0028】

スペース境界線が計算される方法を説明すれば次の通りである。

若い世代領域でのGCが行われた時、行われる前に割り当てられていたメモリの総和と遂行後に生き残ったオブジェクトのメモリ総和との比をR、i番目のGC時に測定された比率をR(i)と定義する。この比率に該当するR(i)が安定した確率分布を有すると仮定する場合、最近k回の平均は、

$$avg = (R(i) + R(i-1) + \dots + R(i-k+1)) / k$$

40

といえる。

【0029】

次のGC時の予想比率は、現在までの比率のk回の履歴の平均で線形近似になる。この時、線形離脱は、各R(i)比率と平均との差で定義され、この線形離脱値もGC時に履歴のひとつとして、

$$div(i) = |R(i) - avg|$$

のように別途に保存される。

【0030】

本発明では、fromスペースとt oスペースとのスペース境界線の急な移動を防止し、メモリプロファイルを安定化させた。このために最近k回のdiv(i)の値がいずれ

50

も設定限界値 MAX_D 内に入っているかどうかを確認し、これが確認された時に新しい境界線の再設定のための離脱値 div は、

$$div = \max(div, MIN_D)$$

を経て確率分布上発生しうる雑音変動を吸収できるように最小限 MIN_D の値を有するようにし、最終的には div が最大許容離脱値 MAX_D より小さい時にのみ境界線を

$$R * = \max(avg, R(i)) + div$$

のように設定する。

【0031】

本発明は、このような予測値を有して $from$ スペースと to スペースとのスペース境界線を変化させ続ける。

一方、前記図3では若い世代領域でのGC方法を示し、老いた世代領域と関連したプロセス、すなわち、数回の若い世代領域のコレクションを通じて年齢が増加したオブジェクトを老いた世代領域に移動させるプロセス及び老いた世代領域自体のGCと両世代間のメモリ参照更新などの事項は、従来の世代別GC方法を利用できる。一般的な世代別GCに関しては、JonesとLinsの‘Garbage Collection: Algorithms for Automatic Dynamic Memory Management, pp. 1-41, Wiley (1996)’に開示されている。

【0032】

本発明の実施による適応型ガーベージコレクタの性能を従来のGC方法と比較するために、サン・マイクロシステムズで標準として提供するJ2 Micro Edition Connected Device Configuration Reference Implementation (以下、‘J2ME CDC RI’という)をテストベッドとした。

【0033】

前記J2ME CDC RIでは‘マークアンドスイープ (mark and sweep)’、‘コピーイングセミスペース (copying semi-space)’、‘ジェネレーショナル (generational)’などのGCアルゴリズムが行われるが、前記3種のアルゴリズムのうち性能の最も良い‘ジェネレーショナル’、すなわち、世代別GCアルゴリズム (以下、‘RI GC’という)と本発明の実施によるアルゴリズム (以下、‘new GC’という)との性能を比較した。

【0034】

また、Java (登録商標) 仮想マシンの性能を測定するのに広く使われるSPEC jvm98ベンチマークテストを通じて前記両アルゴリズムの性能を測定したが、リファレンスPCの仕様として、リヌックスレッドホット8.0 (Linux RedHat 8.0)を運用体制とし、ペンティアム (登録商標) I1、233MHz、32MB RAMをハードウェアシステム仕様とした。これは、本発明が主にメモリリソース管理を重要視するエンベデッドシステムに有効に使われるために、前記のような仕様とした。

【0035】

表1は、前記SPEC jvm98ベンチマークに対するGCの性能を比較した結果を示している。

【表1】

ベンチマーク	メモリ	総 GC 休止時間		RI/new (%)	最大 GC 休止時間	
		RI GC	New GC		RI GC	New GC
_209_db	12M	16383.99	1988.66	823.86	424.64	115.49
_228_jack	2M	8293.63	2531.38	327.63	45.02	65.32
_213_javac	12M	39979.46	15461.86	258.56	1390.39	884.91
_202_jess	3M	1561.63	614.61	254.08	27.79	12.88
_227_mtrt	10M	1494.93	723.18	206.71	82.36	46.39

【0036】

前記表 1 で ‘メモリ’ は、ベンチマーク遂行のためのメモリの大きさを示す。

そして、‘総 GC 休止時間’ はミリ秒単位であって、GC を行うためにユーザープロセスがしばらく休止した時間を累積した値を示す。この値が小さいほど性能がさらに優秀であるということが分かる。

前記表 1 に示されたように、本発明の実施による適応型 GC 方法を使用する場合に、Java（登録商標）仮想マシンに適用されてガーベージコレクタにかかる時間がかなり短縮され、最大休止時間もまた短縮された結果を得られる。

【0037】

以上説明した本発明は、当業者により、本発明の技術的思想を離脱しない範囲内で色々な置換、変形及び変更ができるので、前述した実施形態及び添付された図面に限定されることではない。

【産業上の利用可能性】

【0038】

本発明は、適応型ガーベージコレクション方法及び前記方法を行う装置に利用されうる。

【図面の簡単な説明】

【0039】

【図 1】従来のセミスペース・コピーイング・コレクション方法を示す例示図である。

【図 2】従来の世代別 GC 方法を示す例示図である。

【図 3】本発明の実施による適応型世代別 GC 方法のためのメモリ構造を示す例示図である。

【図 4】本発明の実施による若い世代領域での GC を行う方法を示す一実施形態の処理フローチャートである。

【符号の説明】

【0040】

300…若い世代領域

310…fromスペース

320…スペース境界線

330…toスペース

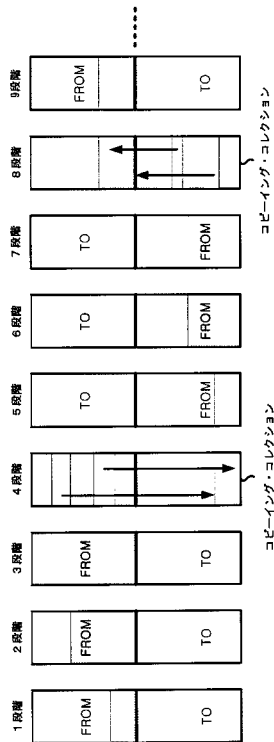
350…老いた世代領域

10

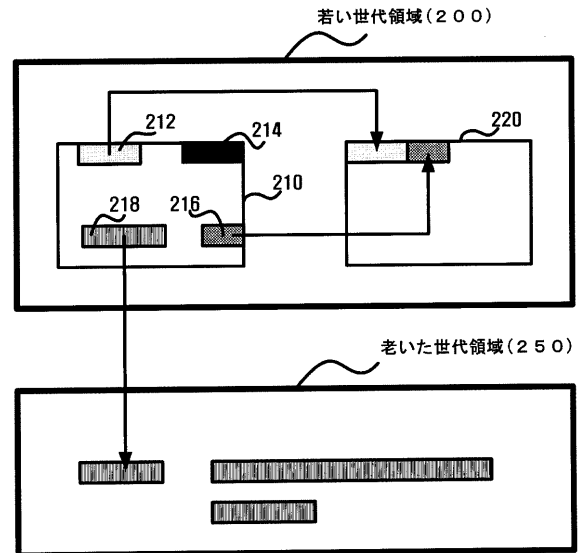
20

30

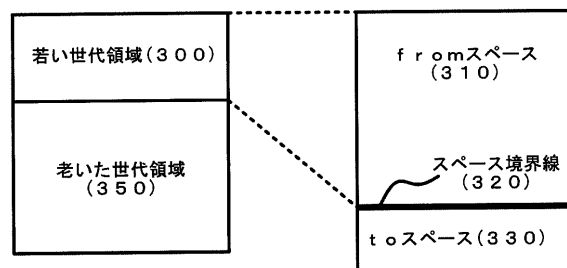
【図 1】



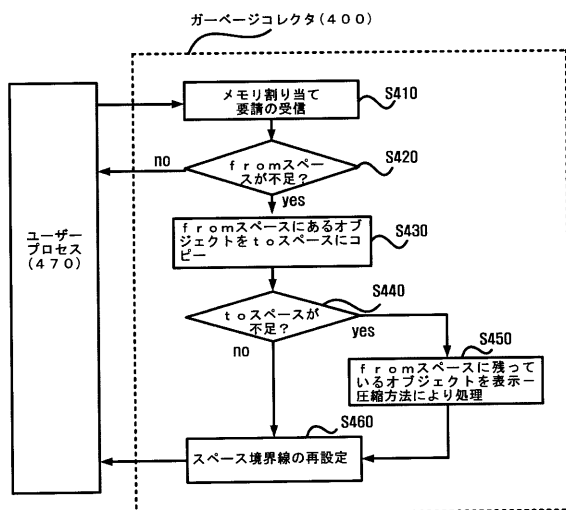
【図 2】



【図 3】



【図 4】



フロントページの続き

(72)発明者 鄭 承範

大韓民国京畿道城南市盆唐區盆唐洞 103-16 番地 204 號

(72)発明者 アレクセイ・ロマノフスキ

大韓民国京畿道城南市盆唐區亭子洞 (番地なし) ハンソルマウル住公4團地アパート 415 棟 1203 號

(72)発明者 趙 雄石

大韓民国ソウル特別市廣津區紫陽洞 600-14 番地 5 層

(72)発明者 許 得萬

大韓民国ソウル特別市江南區道谷2洞 (番地なし) 開浦韓新アパート 7 棟 101 號

(72)発明者 アレクサンダー・マルコフ

ロシア・630128・ノヴォシビルスク・デマコヴァ・ストリート・17-151

(72)発明者 ヴィタリー・ミクヒーフ

ロシア・630090・ノヴォシビルスク・テレシュコヴジュ・ストリート・10-113

審査官 田中 秀人

(56)参考文献 特開平06-309220 (JP, A)

特開平11-312117 (JP, A)

特表2003-515812 (JP, A)

(58)調査した分野(Int.Cl., DB名)

G06F 12/00 - 12/06