(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5111813号
(P5111813)

(45)発行日　平成25年1月9日(2013.1.9)　(24)登録日　平成24年10月19日(2012.10.19)

(51)Int.Cl.　F I
G06F 12/02 (2006.01)　G06F 12/02 510A
G06F 12/02 530A

請求項の数 8　(全 14 頁)

(21)出願番号　特願2006-245622(P2006-245622)
(22)出願日　平成18年9月11日(2006.9.11)
(65)公開番号　特開2008-65764(P2008-65764A)
(43)公開日　平成20年3月21日(2008.3.21)
審査請求日　平成21年9月1日(2009.9.1)

(73)特許権者　394020376
ガイアホールディングス株式会社
東京都新宿区西早稲田２－１８－１８
(74)代理人　100088683
弁理士　中村　誠
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100075672
弁理士　峰　隆司
(74)代理人　100109830
弁理士　福原　淑弘
(74)代理人　100084618
弁理士　村松　貞男
(74)代理人　100092196
弁理士　橋本　良郎

最終頁に続く

(54)【発明の名称】メモリアロケーション方法およびメモリアロケーションプログラム

(57)【特許請求の範囲】
【請求項１】
所定サイズのメモリ要求に対して、空きメモリを管理する単方向リンクリストを検索して、最初に見付かった所定サイズ以上の空きメモリを返すメモリアロケーション処理を行うメモリアロケーション方法において、
前記メモリアロケーション処理が行われるごとに、メモリ要求により要求されたサイズと、当該メモリ要求に応じて返した空きメモリの次の空きメモリにアクセスするためのアドレスを表す履歴情報とを含むノードを蓄積し、
新たなメモリ要求がなされた際に、前記各ノードに対し最大サイズを記録したノードから順にアクセスしてその記録サイズを前記新たにメモリ要求されたサイズと比較することにより、前記新たにメモリ要求されたサイズを超えない最大のサイズを記録したノードを選択し、この選択されたノードに記録されたアドレスから前記単方向リンクリストの検索を行うことを特徴とするメモリアロケーション方法。
【請求項２】
前記履歴情報は、返した空きメモリの１つ前の空きメモリのアドレスであることを特徴とする請求項１に記載のメモリアロケーション方法。
【請求項３】
前記各ノードは、メモリ要求されたサイズの降順に配列された双方向リンクリストとして管理されていることを特徴とする請求項２に記載のメモリアロケーション方法。
【請求項４】

さらにまた、前記双方向リンクリストに新たなノードを追加することによって、低サイズ側のノードに記録されたアドレスが高サイズ側のノードに記録されたアドレスよりも前記単方向リンクリスト後方を指すようになった場合には、当該高サイズ側のノードを削除することを特徴とする請求項３に記載のメモリアロケーション方法。

【請求項５】

所定サイズのメモリ要求に対して、空きメモリを管理する単方向リンクリストを検索して、最初に見付かった所定サイズ以上の空きメモリを返すメモリアロケーション処理を、プロセッサに実行させるメモリアロケーションプログラムであって、

前記メモリアロケーション処理が行われるごとに、メモリ要求により要求されたサイズと、当該メモリ要求に応じて返した空きメモリの次の空きメモリにアクセスするためのアドレスを表す履歴情報とを含むノードを蓄積する処理と、

新たなメモリ要求がなされた際に、前記各ノードに対し最大サイズを記録したノードから順にアクセスしてその記録サイズを前記新たにメモリ要求されたサイズと比較することにより、前記新たにメモリ要求されたサイズを超えない最大のサイズを記録したノードを選択し、この選択されたノードに記録されたアドレスから単方向リンクリストの検索を行う処理と

を、前記プロセッサに実行させるメモリアロケーションプログラム。

【請求項６】

前記履歴情報は、返した空きメモリの１つ前の空きメモリのアドレスであることを特徴とする請求項５に記載のメモリアロケーションプログラム。

【請求項７】

前記各ノードは、メモリ要求されたサイズの降順に配列された双方向リンクリストとして管理されていることを特徴とする請求項６に記載のメモリアロケーションプログラム。

【請求項８】

さらにまた、前記双方向リンクリストに新たなノードを追加することによって、低サイズ側のノードに記録されたアドレスが高サイズ側のノードに記録されたアドレスよりも前記単方向リンクリスト後方を指すようになった場合には、当該高サイズ側のノードを削除する処理を、前記プロセッサに実行させることを特徴とする請求項７に記載のメモリアロケーションプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、メモリアロケーション方法およびメモリアロケーションプログラムに関する。

【背景技術】

【０００２】

ＯＳ、ミドルウェア、アプリケーション実行環境には、アプリケーション等の上位プログラムに対してメモリを提供し、また提供しているメモリおよび空きメモリの管理を行うメモリマネジメントの機能を有しているものがある。その中でも、特にＪａｖａ（登録商標）アプリケーションを動かすためのＪａｖａ（Ｒ）実行環境では、メモリを提供するメモリアロケーションだけでなく、使用済みメモリを回収するガーベッジコレクション等の機能も実現されており、メモリ管理を全く意識することなくアプリケーションを作成することが可能になっている。

【０００３】

このようなメモリマネジメント機能の実装の一つに、空きメモリをリンクリストで管理し、アプリケーションから所定サイズの空きメモリが要求されると、このリンクリストを低アドレス側または高アドレス側から順に検索し、最初に見付かった所定サイズ以上の空きメモリをアプリケーションに利用させるものが知られている。しかし、このような実装では空きメモリが分散してフラグメンテーションを起こしている場合、無数の空きメモリを辿らなければならず、メモリアロケーションに時間がかかってしまうという問題があっ

た。
【０００４】
このため、メモリマネジメント機能の実装を見直してこれを高速化したいという要望は高かった。しかし、このような実装またはアルゴリズムの見直しによって、アロケートされる空きメモリが変わってしまったり、メモリ利用の形態自体が大きく変わったりすると、既存の実装では発生しなかったエラーが発生する等、アプリケーションの動作に影響を及ぼす可能性がある。例えメモリアロケーションを高速化するという目的を達成できたとしても、既存のアプリケーションの動作に影響を与えるということは、当該アプリケーションのデバッグを行わなければならないことを意味するため、その技術的な価値は必ずしも高くない。
【０００５】
メモリアロケーションを行う方法や装置については、特開２０００－１１２８１４や特開平８－２２１３１８号公報等が提案されているが、上述したように既存の実装と同じ空きメモリを高速にアロケーションする技術は知られていない。
【特許文献１】特開２０００－１１２８１４号公報
【特許文献２】特開平８－２２１３１８号公報
【発明の開示】
【発明が解決しようとする課題】
【０００６】
本発明は、空きメモリをリンクリストで管理し、アプリケーションから所定サイズの空きメモリが要求されると、このリンクリストを低アドレス側または高アドレス側から順に検索し、最初に見付かった所定サイズ以上の空きメモリをアプリケーションに利用させるメモリマネジメント機能の実装において、アロケートされる空きメモリおよびメモリ利用の形態は変えることなく、メモリアロケーションを高速化することができるメモリマネジメント方法およびメモリマネジメントプログラムを提供することを目的とする。
【課題を解決するための手段】
【０００７】
上記課題を解決するために、本発明の第１の観点によれば、所定サイズのメモリ要求に対して、空きメモリを管理する単方向リンクリストを検索して、最初に見付かった所定サイズ以上の空きメモリを返すメモリアロケーション処理を行うメモリアロケーション方法において、
前記メモリアロケーション処理が行われるごとに、メモリ要求により要求されたサイズと、当該メモリ要求に応じて返した空きメモリの次の空きメモリにアクセスするためのアドレスを表す履歴情報とを含むノードを履歴情報とを蓄積し、
新たなメモリ要求がなされた際に、前記各ノードに対し最大サイズを記録したノードから順にアクセスしてその記録サイズを前記新たにメモリ要求されたサイズと比較することにより、前記新たにメモリ要求されたサイズを超えない最大のサイズを記録したノードを選択し、この選択されたノードに記録されたアドレスから前記単方向リンクリストの検索を行うことを特徴とするメモリアロケーション方法が提供される。
【０００８】
このような構成において、前記履歴情報は、返した空きメモリの１つ前の空きメモリのアドレスとすることが好適である。また、メモリ要求されたサイズおよび返した空きメモリの１つ前の空きメモリのアドレスからなるノードは、メモリ要求されたサイズ降順の双方向リンクリストとして管理することができる。この場合には、前記双方向リンクリストに新たなノードを追加することによって、低サイズ側のノードが高サイズ側のノードよりも前記単方向リンクリスト後方を指すようになった場合には、当該高サイズ側のノードを削除することが望ましい。
【０００９】
また、本発明の第２の観点によれば、所定サイズのメモリ要求に対して、空きメモリを管理する単方向リンクリストを検索して、最初に見付かった所定サイズ以上の空きメモリ

を返すメモリアロケーション処理を、プロセッサに実行させるメモリアロケーションプログラムであって、
前記メモリアロケーション処理が行われるごとに、メモリ要求により要求されたサイズと、当該メモリ要求に応じて返した空きメモリの次の空きメモリにアクセスするためのアドレスを表す履歴情報とを含むノードを蓄積し、
新たなメモリ要求がなされた際に、前記各ノードに対し最大サイズを記録したノードから順にアクセスしてその記録サイズを前記新たにメモリ要求されたサイズと比較することにより、前記新たにメモリ要求されたサイズを超えない最大のサイズを記録したノードを選択し、この選択されたノードに記録されたアドレスから単方向リンクリストの検索を行う処理を、前記プロセッサに実行させることを特徴とするメモリアロケーションプログラムが提供される。

【００１０】

以上のような構成において、前記履歴情報は、返した空きメモリの１つ前の空きメモリのアドレスとすることができる。また、メモリ要求されたサイズおよび返した空きメモリの１つ前の空きメモリのアドレスからなるノードは、メモリ要求されたサイズ降順の双方向リンクリストとして管理することができる。このような場合には、前記双方向リンクリストに新たなノードを追加することによって、低サイズ側のノードが高サイズ側のノードよりも前記単方向リンクリスト後方を指すようになった場合には、当該高サイズ側のノードを削除することが好適である。

【発明の効果】

【００１１】

本発明のように、所定サイズのメモリ要求に対して、空きメモリを管理する単方向リンクリストを検索して、最初に見付かった所定サイズ以上の空きメモリを返すメモリアロケーションでは、サイズＸについてのメモリ要求に対してｎ番目の空きメモリが返されたとすると、ｎ－１番目以前の空きメモリにはサイズＸ以上のものは存在しないことになる。そこで本発明は、このような履歴情報を蓄積し、その後にサイズＸよりも大きいサイズのメモリ要求がなされた場合、ｎ－１番目の空きメモリからリンクリストを辿って検索を行うようにすることで、空きメモリ検索の時間を大きく短縮し、メモリアロケーションを高速化するものである。しかも、このような高速化では、アロケーションされる空きメモリの位置を変化させないので、アプリケーションに新たなエラーを発生させたり、アプリケーションの動作を変化させたりすることはなく、アプリケーションのデバッグや修正は一切必要がない。

【００１２】

上記において、論理的にはｎ＋１番目から空きメモリを検索するようにしても同様の効果を得ることができるはずだが、この場合ｎ＋１番目の空きメモリが存在しない場合の処理を入れる必要があり、この処理が全体のパフォーマンスに悪影響をおよぼす可能性がある。このため、本発明ではｎ－１番目の空きメモリから単方向リンクリストの検索を行うようにしている。

【発明を実施するための最良の形態】

【００１３】

以下、図面を参照して、本発明の実施の形態について説明する。

【００１４】

以下、添付図面を参照して、本発明の実施の形態について説明する。

【００１５】

図１は、本発明の一実施の形態に係るメモリアロケーション方法を適用したシステム１００の構成例である。図１に示されるように、このシステム１００は、主としてアプリケーション１０１と、アプリケーション１０１が使用するメモリを管理するメモリマネジメントシステム１０２と、アプリケーション１０１がメモリマネジメントシステム１０２を介して利用するメモリ１０３とから構成されている。ここで、アプリケーション１０１とメモリマネジメントシステムとは、いずれも所定のプログラムを不図示のプロセッサで実

行することにより実現されているものであるが、その機能の一部はハードウェアで実装されていても構わない。

【００１６】

メモリマネジメントシステム１０２は、アプリケーション１０１からのメモリ要求に応じてメモリ１０３のヒープメモリ１０９からメモリ割り付け（メモリアロケーション）を行うメモリアロケータ１０４と、ヒープメモリ１０９から使用済みメモリを回収して必要に応じて使用中メモリのコンパクションを行うガーベッジコレクタ１０５と、メモリアロケータ１０４が使用する双方向リンクリスト１０６とを含んでいる。このようなメモリマネジメントシステム１０２により、アプリケーション１０１はメモリ１０３からメモリ割り付けを受け、アプリケーション１０１が使用を済ませたメモリは自動的に回収される。すなわち、本実施形態では、アプリケーション１０１側ではメモリ管理を全く意識する必要がない。なお、図１のヒープ１０９で、斜線を付した部分は使用中メモリを、斜線を付していない部分は空きメモリをそれぞれ示している（以下も同様）。また、メモリ１０３にはヒープ１０９とは別にダミー１０８が設けられている。ここで、ダミー１０８はメモリ１０３上の所定アドレスにかかる領域を指したものである。

【００１７】

図２および図３は、本実施形態におけるヒープ１０９の空きメモリを管理する機構を説明するための図面である。図２に示すように、ヒープ１０９に空きメモリ２０１ａ，２０１ｂ，２０１ｃ，２０１ｄ，２０１ｅ，２０１ｆが存在する場合、これら空きメモリとダミー１０８は、図３に示すように単方向リンクリストをなしている。すなわち、ダミー１０８には最初の空きメモリのアドレスが書き込まれており、各空きメモリ２０１ａ，２０１ｂ，２０１ｃ，２０１ｄ，２０１ｅ，２０１ｆは、その先頭に自分のサイズと、次の空きメモリのアドレスが書き込まれている。このような空きメモリ上に単方向リンクリストをなす構成により、メモリアロケータ１０４は、ダミー１０８から最初の空きメモリ２０１ａにアクセスしてそのサイズを調べ、さらに次の空きメモリ２０１ｂにアクセスしてそのサイズを調べ、以下同様に空きメモリ２０１ｃ，２０１ｄ，２０１ｅ，２０１ｆへと、それぞれのサイズを調べつつ低アドレス側から順にアクセスしていくことができる。

【００１８】

従来のメモリアロケーション方法では、上記のように空きメモリを管理しつつ、アプリケーションから所定サイズのメモリ要求がなされた場合には、毎回空きメモリの単方向リンクリストを先頭から辿って、最初に検出された所定サイズ以上の空きメモリにアロケートしていた。本実施形態においても、単方向リンクリストを利用して空きメモリを検索するが、以下に説明する双方向リンクリスト１０６を併用することにより、空きメモリの検索は大幅に高速化されている。

【００１９】

すなわち、サイズｎのメモリ要求がなされた場合に、空きメモリ２０１ｄがアロケートされたとすると、より低アドレス側に存在する空きメモリ２０１ａ，２０１ｂ，２０１ｃはサイズｎ未満であることになる。したがって、次に同じサイズｎのメモリ要求がなされた場合に、空きメモリ２０１ａ，２０１ｂ，２０１ｃを対象から外して検索を行えば、空きメモリ検索のオーバーヘッドを軽減することができる。このような効果は、サイズｎよりも大きいメモリ要求がなされた場合にも、同様にして得ることができる。本実施形態は、このような知見に基づいて空きメモリの検索を高速化するものである。

【００２０】

図４は、双方向リンクリスト１０６のデータ構造を機能的に図示した図面である。この図に示した通り、双方向リンクリスト１０６は、サイズ、検索履歴、次ノードのアドレスおよび前ノードのアドレスを格納したデータ構造であるノード１０７ａ，１０７ｂ，１０７ｃ，１０７ｄがリンクされたものである。ここで、各ノードはアプリケーション１０１からのメモリ要求に対してメモリアロケータ１０４が行った検索結果の履歴を蓄積するものである。すなわち、各ノードのサイズはメモリ要求されたサイズを記録するものであり、履歴情報は当該メモリ要求に対してメモリアロケータ１０４が返した空きメモリの１個

前の空きメモリのアドレスを記録するものである。これらのノードはノード１０７ｄ，１０７ｃ，１０７ｂ，１０７ａとサイズ降順になるようにリンクされている。

【００２１】

ただし、上記にかかわらず、ノード１０７ａはサイズ＝１に固定されており、また前記の履歴情報に代えて最初の空きメモリのアドレスを記録したダミー１０８のアドレスが書き込まれている。また、双方向リンクリスト１０６外には、メモリアロケータ１０４が双方向リンクリスト１０６にアクセスするために、最大サイズにかかるノード１０７ｄのアドレスを記録したInitiator１１０が設けられており、ノード１０７ｄには前ノードのアドレスに代えてInitiator１１０のアドレスが記録されている。

【００２２】

ここで、図４には説明のためにノードの数が４個になった状態を示しているが、後述するように番兵であるノード１０７ａ以外のノードは動的に追加／削除され得るものなので、ノードの数はこれに限定されたものではない。

【００２３】

上述のように、本実施形態では、空きメモリに自己のサイズと次の空きメモリのアドレスとを埋め込んでなる単方向リンクリストと、メモリ要求に対してメモリアロケータ１０４が行った検索結果の履歴を蓄積した双方向リンクリスト１０６とによりヒープ１０９上の空きメモリを管理する。図５は、このような空きメモリ管理の態様を概略的に示したものである。

【００２４】

以下、本実施形態におけるメモリアロケーションの処理動作について、図６および図１１のフローチャートと、図７～図１０、図１２及び図１３の概略図とを参照しながら説明する。

【００２５】

まず、図６のフローチャートに沿って、メモリアロケーションがなされるまでの処理を説明する。まず、アプリケーション１０１から所定サイズのメモリ要求がなされると（ＳＴ１０１）、このメモリ要求に応じて、メモリマネジメントシステム１０２のメモリアロケータ１０４が、Initiator１１０の指すアドレスから双方向リンクリスト１０６の検索を開始する（ＳＴ１０２）。まず、最初にInitiator１１０の指す最大サイズにかかるノードにアクセスし、その状態で現在アクセスしているノードのサイズが要求サイズ以下かを判断する（ＳＴ１０３）。

【００２６】

ここで、要求サイズ以下であると判断された場合には、現在のノードに記録されている検索履歴から、空きメモリへのアクセスが行われる（ＳＴ１０４）。現在のノードの検索履歴には、前回同じサイズについて空きメモリ検索して得られたものの一つ前の空きメモリのアドレスが記録されているので、ＳＴ１０４では当該「一つ前の空きメモリ」にアクセスすることになる。一方、現在アクセスしているノードのサイズが要求サイズを超えている場合には、双方向リンクリスト１０６の次のノードへアクセスし、ＳＴ１０３の判断を繰り返す。ここで、双方向リンクリスト１０６の末尾には必ずサイズ＝１のノードが存在し、これを番兵法の番兵として利用することによって、双方向リンクリストの末端を検出する処理を省略することができる。

【００２７】

ＳＴ１０４で前記「一つ前の空きメモリ」にアクセスした後、メモリアロケータ１０４は単方向リンクリストに沿って次の空きメモリへアクセスを行い（ＳＴ１０５）、以降は要求サイズ以上の空きメモリを見出すために単方向リンクリストの検索が常法に従って行われる。すなわち、現在アクセスしている空きメモリのサイズが要求サイズ以上であるかを判断し（ＳＴ１０７）、要求サイズ以上であると判断した場合には現在の空きメモリにアロケーションを行い（ＳＴ１０９）、処理は終了する。このようにして、メモリマネジメントシステム１０２は、アプリケーション１０１のメモリ要求に対して、現在の空きメモリをアロケートして利用させる。

【００２８】
一方、ＳＴ１０７で要求サイズ未満であると判断した場合には、現在の空きメモリが単方向リンクリストの最後の空きメモリであるか否かを判断し（ＳＴ１０９）、最後の空きメモリではないと判断された場合には、次の空きメモリにアクセスし（ＳＴ１１０）、ＳＴ１０７以下の処理を繰り返す。また、ＳＴ１０８で最後の空きメモリであると判断した場合にはＯｕｔＯｆＭｅｍｏｒｙエラーを発生させて（ＳＴ１１１）処理を終了する。この場合には、アプリケーション１０１のメモリ要求に対してエラーが返されて、メモリはアロケーションされない。

【００２９】
次に、以上のような処理によりメモリアロケーションが行われた場合に、上記の処理に引き続いて、双方向リンクリスト１０６をメンテナンスする処理について図１１のフローチャートに沿って説明する。

まず、メモリ要求されたサイズにかかるノードが、双方向リンクリスト２０６中に存在するかが判断される（ＳＴ２０１）。ここで存在しないと判断された場合には、双方向リンクリスト１０６に新しいノードを追加する（ＳＴ２０２）。次いで、追加したノードに、メモリ要求されたサイズと、ＳＴ１０９でアロケートした空きメモリの一つ前の空きメモリのアドレスとを書き込む。一方、ＳＴ２０１でメモリ要求されたサイズにかかるノードが存在すると判断された場合には、メモリ要求されたものと同サイズのノードに、アロケートした空きメモリの一つ前の空きメモリのアドレスを書き込む（ＳＴ２０３）。このようなＳＴ２０１からＳＴ２０３までの処理で、双方向リンクリスト１０６に今回の検索結果が記録される。

【００３０】
ＳＴ２０３またはＳＴ２０４に引き続いて、追加したノードより大サイズのノードに、履歴情報が追加したノードよりも低アドレスのものがあるかを判断する（ＳＴ２０５）。ここで、各ノードは双方向リンクリスト１０６でサイズ降順に管理されているので、このような判断は追加したノードの一つ前のノードにアクセスして履歴情報を読み出し、現在のノードの履歴情報と比較することにより行うことができる。本実施形態ではノードを双方向リンクリストで管理しており、次ノードのみならず前ノードへのアクセスも簡易に行うことができるため、このような判断は小さいオーバーヘッドで実行することができる。また、一つ前のノードの履歴情報が現在のノードよりも低アドレスであった場合には、さらにもう一つ前のノードについて同じ処理を行うことで、追加したノードよりも低アドレスを指す大サイズのノードを全て特定することができる。

【００３１】
ＳＴ２０５で追加したノードよりも低アドレスのものがあると判断された場合、そのようなノードは双方向リンクリスト１０６から削除され（ＳＴ２０６）、削除されたノードの前後でリンクが切れないように双方向リンクリスト１０６がメンテナンスされ（ＳＴ２０７）、処理は終了する。一方、ＳＴ２０５で低アドレスのものがあると判断されなかった場合には、処理はそのまま終了される。このようなＳＴ２０５からＳＴ２０７の処理により、冗長なノードを削除することができる。

【００３２】
図７～１３は、以上のような処理を概略的に示した図面である。

図７は、双方向リンクリスト１０６に１，５，３０のノードが保持されている状態を示している。ここでは、ヒープ１０９上にサイズ１，３，５，４，９，１０，３，１０，５の空きメモリがこの順で並んでおり、サイズ１にかかるノードはダミー１０８を指しており、このダミーは最初の空きメモリのアドレスが記録されている。また、サイズ５にかかるノードには検索履歴として２番目の空きメモリのアドレスが記録されており、サイズ５にかかるノードは９番目の空きメモリが記録されている。

【００３３】
ここで、サイズ５のノードは、過去にサイズ５のメモリ要求がなされた際に追加されたものであり、２番目の空きメモリの次にはじめてサイズ５以上の空きメモリが検出された

ことを示している。したがって、２番目の空きメモリより低アドレス側には、サイズ５以上の空きメモリは存在しないことになる。サイズ３０のノードについても同様である。

【００３４】

図８は、図７に示した状態からサイズ１０のメモリ要求がなされた場合の処理を視覚的に説明するための図面である。メモリアロケータ１０４は、図に矢印を強調して示すように、まずInitiator１１０から双方向リンクリスト１０６にアクセスし、最大サイズのノードにアクセスして要求サイズとノードのサイズ（３０）を比較し、ノードのサイズが大きいため次のノード（サイズ５）にアクセスし、このノードのサイズが要求サイズより小さいため履歴情報から２番目の空きメモリにアクセスする。それ以降は、矢印で示したように空きメモリの単方向リンクリストを辿って、最初に検出されたサイズ１０の空きメモリにアロケーションを行う。

【００３５】

図９および図１０は、以上の検索結果を双方向リンクリストに蓄積する処理を同様に示した図面である。ここでは、双方向リンクリスト１０６に１０のノードが存在しないため、ノードを追加する場合を示す。まず、図１０に点線で示すようにノードの領域を確保し、次ノードであるサイズ５のアドレスと、前ノードであるサイズ３０のアドレスをそれぞれ書き込んで双方向リンクさせる。次いで、図１０に示すように追加したノードにサイズとして１０を書き込み、さらに履歴情報として今回アロケーションしたものの一つ前の空きメモリのアドレスを書き込む。最後に、空きメモリのなす単方向リンクリストについて、今回アロケーションしたものをとばすようにメンテナンスを行い、一連の処理は終了する。

【００３６】

本実施形態において、履歴情報として今回アロケーションしたものの一つ前の空きメモリのアドレスを書き込むのは、今回アロケーションした空きメモリ自体はアプリケーション１０１によって上書きされるため記録する意味が無く、次の空きメモリは存在することが保証されないためである。単方向リンクリストで管理された空きメモリの検索を常法に沿って行うためには、このような履歴情報を記録することが最も好適である。

【００３７】

図１２および１３は、図１１のＳＴ２０５～２０７で行う冗長ノードを削除する処理を同様に示した図面である。図１２は、サイズ１，５，３０，５０のノードを有する双方向リンクリスト１０６に追加されたサイズ１０のノードの履歴情報が、サイズ３０および５０のノードよりも高アドレスを指している状態を示す。ここで、サイズ１０のノードの履歴情報が指しているアドレスよりも低アドレスには、サイズ１０以上の空きメモリは検出されていないので、サイズ３０およびサイズ５０のノードが持っている履歴情報は冗長となっている。そこで、本実施形態では図１１のＳＴ２０５～２０７の処理によって、このようなノードを検出して削除する。図１３は、冗長なサイズ３０および５０のノードを削除した後における双方向リンクリスト１０６の状態を示す図面である。

【００３８】

以上説明したように、本実施形態によれば、アプリケーションからのメモリ要求に対して、単方向リンクリストで管理された空きメモリを毎回最初から検索するのではなく、以前に空きメモリ検索した結果を記録した履歴情報を利用して、空きメモリの途中から検索を行うので、空きメモリ検索のオーバーヘッドを軽減することができ、これによりメモリアロケーションを高速化することができる。実際に、空きメモリのフラグメントが激しい状態でメモリアロケーションの速度を計測するベンチマークプログラムを準備し、従来のように毎回空きメモリの単方向リンクリストを先頭から検索した場合のベンチマーク結果と、本実施形態で測定したベンチマーク結果とを比較したところ、本実施形態によりメモリアロケーションの速度は４０倍程度速くなることが確認された。

【００３９】

しかも、アロケートされる空きメモリは、毎回最初から検索した場合と全く同じものになるので、本実施形態の適用によりこれまで発生しなかったエラーが発生したり、動作が

変わったりすることはないので、従来使用していたアプリケーションを修正する必要はない。

【００４０】

ただし、ガーベッジコレクタ１０５により使用済みメモリを回収した場合、不連続な空きメモリから連続した空きメモリをなすコンパクションを行った場合には（以下ＧＣを行った場合と総称する）、双方向リンクリスト１０６のノードや履歴情報をメンテナンスする必要が生じる。このようなメンテナンスは行ってもよいが、処理コストに対して得られる効果が割に合わないと判断される場合には、ガーベッジコレクタ１０５がＧＣを行う度に双方向リンクリスト１０６からサイズ＝１の番兵以外のノードを削除してリフレッシュするようにしても構わない。これにより、ダミー１０８が指す最初の空きメモリのアドレスのみアドレス更新を行えば足りるようになるので、そのコストを相応に小さく抑えることが可能となる。

【００４１】

以上、本発明の実施の形態について説明したが、本発明はこれに限定されることなく、その趣旨を逸脱しない範囲で種々の改良・変更が可能であることは勿論である。例えば、上記ではガーベッジコレクタ１０４を備えた場合を示したが、ガーベッジコレクタ１０４は本発明に必須な要素ではないので省略しても構わない。また、図６および図７に示したフローチャートは、本発明にかかるメモリアロケーション方法およびメモリアロケーションプログラムを適用した処理の一例を示すものであり、これらに限定されるものではない。具体的には、例えば図６および図７のフローチャートに示した処理をシーケンシャルに行う必要はなく、双方の処理を同時並行的に行うようにしても構わない。

【００４２】

さらに、本明細書中に記載した「空きメモリのアドレス」は、メモリマネジメントシステム１０２の設計によって所望に取り決めることができ、メモリマネジメントシステム１０２が空きメモリを利用・管理することができれば、空きメモリの先頭アドレスであっても構わないし、空きメモリの先頭から所定ビットだけ後方のアドレスであっても構わない。

【図面の簡単な説明】

【００４３】

【図１】本発明の一実施の形態に係るメモリアロケーション方法を適用したシステムの構成例。

【図２】空きメモリを管理する機構を説明するための図面。

【図３】空きメモリを管理する機構を説明するための図面。

【図４】双方向リンクリストのデータ構造を機能的に図示した図面。

【図５】空きメモリ管理の態様を概略的に示した図面。

【図６】メモリアロケーションの処理動作を示すフローチャート。

【図７】メモリアロケーションの処理動作を示すフローチャート。

【図８】本発明の処理を概略的に示した図面。

【図９】本発明の処理を概略的に示した図面。

【図１０】本発明の処理を概略的に示した図面。

【図１１】本発明の処理を概略的に示した図面。

【図１２】本発明の処理を概略的に示した図面。

【図１３】本発明の処理を概略的に示した図面。

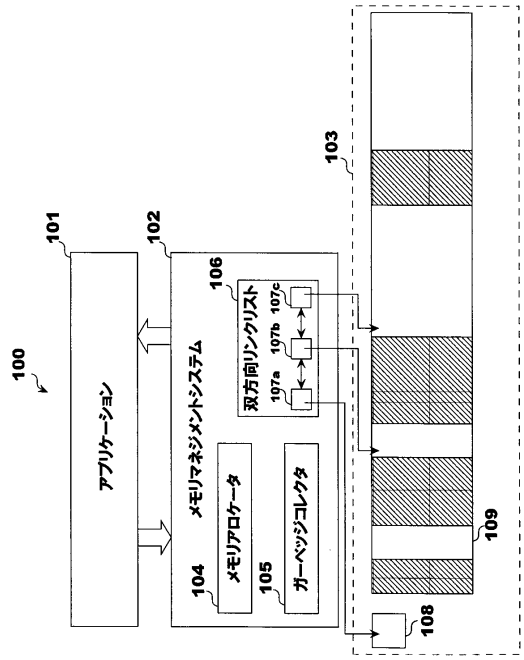
【符号の説明】

【００４４】

１００…システム、１０１…アプリケーション、１０２…メモリマネジメントシステム、１０３…メモリ、１０４…メモリアロケータ、１０５…ガーベッジコレクタ、１０６…双方向リンクリスト。

【図１】

図1

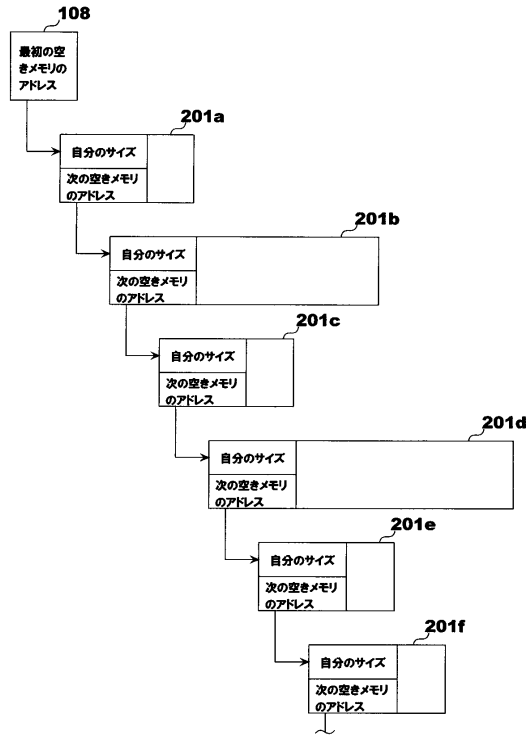


【図２】

図2

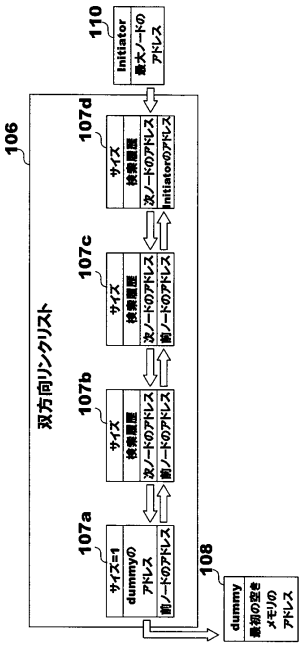
【図３】

図3



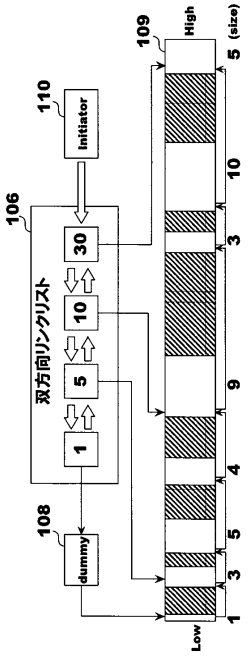
【図４】

図4



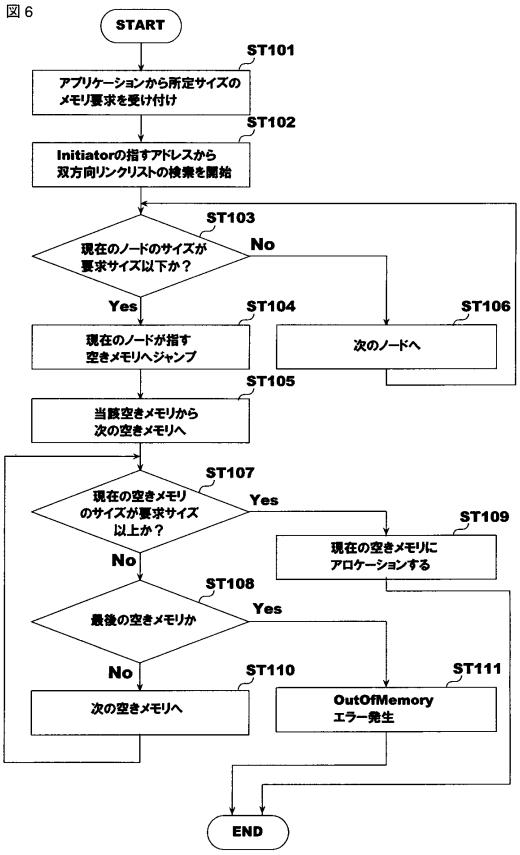
【図５】

図５



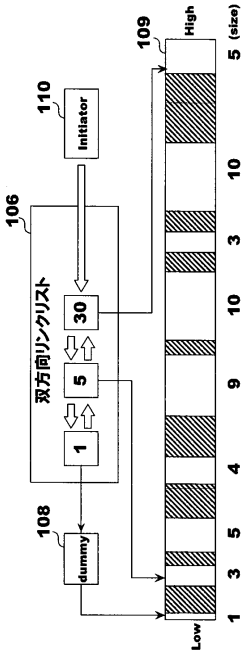
【図６】

図６



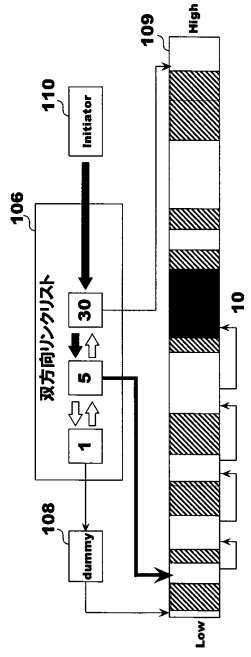
【図７】

図７



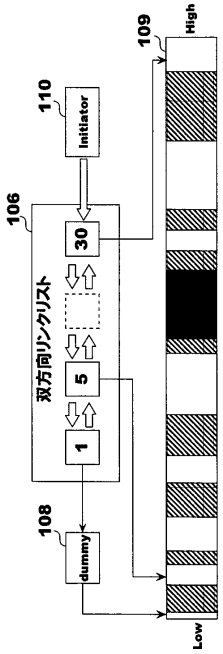
【図８】

図８



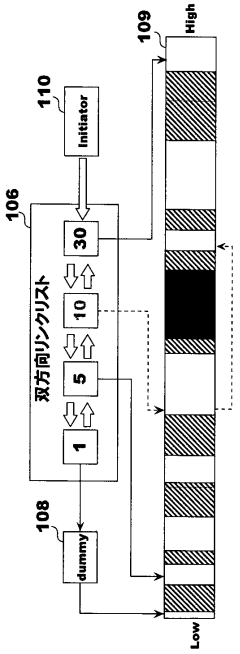
【図９】

図9



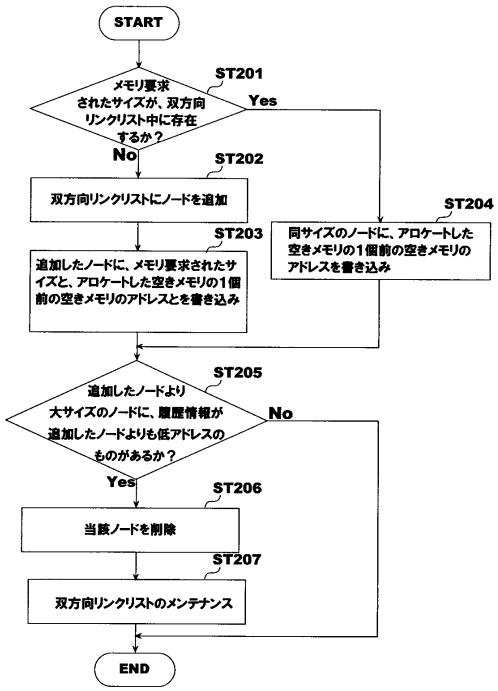
【図１０】

図10



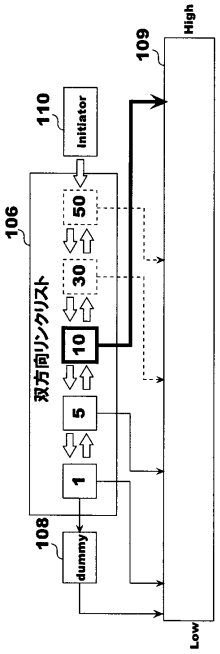
【図１１】

図11



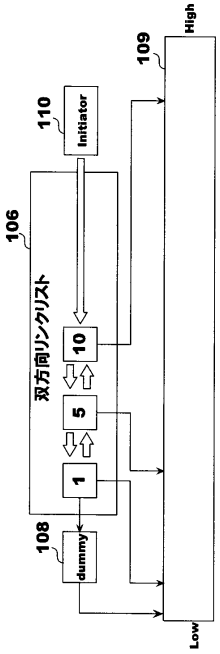
【図１２】

図12



【図１３】

図13



フロントページの続き

(72)発明者　岡　智幸
東京都新宿区西早稲田２－１８－１８　株式会社アプリックス内

審査官　桜井　茂行

(56)参考文献　特開平０９－１６０８２３（ＪＰ，Ａ）
特開２００６－０４８２３７（ＪＰ，Ａ）
特開２０００－１１２８１４（ＪＰ，Ａ）
特開平８－２２１３１８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００－１２／０６