(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2015-127869
(P2015-127869A)
(43)公開日　平成27年7月9日(2015.7.9)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 12/00	(2006.01)	G06F	12/00	550E	5B060

審査請求 未請求　請求項の数 10　ＯＬ　（全 15 頁）

(21)出願番号　特願2013-272936 (P2013-272936)
(22)出願日　平成25年12月27日 (2013.12.27)

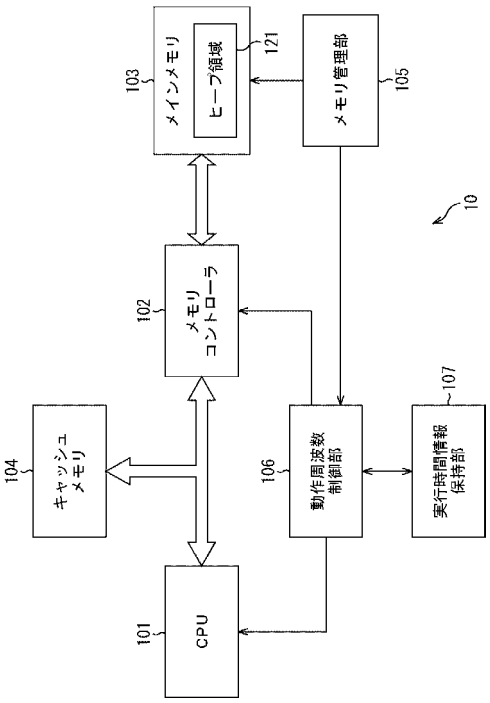
(71)出願人　504133110
国立大学法人電気通信大学
東京都調布市調布ケ丘一丁目５番地１
(74)代理人　100082131
弁理士　稲本　義雄
(74)代理人　100121131
弁理士　西川　孝
(72)発明者　鵜川　始陽
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内
(72)発明者　橘田　頼之
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内

最終頁に続く

(54)【発明の名称】電子機器、制御方法、及び、プログラム

(57)【要約】
【課題】ＧＣの処理の実行時におけるCPUの動作周波数を適切に制御して、消費電力を抑えることができるようにする。
【解決手段】動作周波数制御部は、動作周波数を制御可能なCPUにより実行されるプログラムで使用するデータを記憶するメインメモリにおける当該プログラムが動的にメモリを確保する領域であるヒープに対する、ガベージコレクションの実行が開始される前あるいは途中に、CPUの動作周波数を下げる制御を行い、ガベージコレクションをCPUの動作周波数が下げられた状態で実行させ、ガベージコレクションの実行の途中あるいは終了したとき、CPUの動作周波数を元の周波数に戻す制御を行う。本発明は、例えば、スマートフォン等の携帯端末に適用することができる。
【選択図】図６

図6



【特許請求の範囲】
【請求項１】
動作周波数を制御可能なCPUと、
前記CPUにより実行されるプログラムで使用するデータを記憶するメインメモリと、
前記メインメモリにおける前記プログラムが動的にメモリを確保する領域であるヒープに対するガベージコレクションの実行が開始される前あるいは途中に、前記CPUの動作周波数を下げる制御を行い、前記ガベージコレクションを前記CPUの動作周波数が下げられた状態で実行させ、前記ガベージコレクションの実行の途中あるいは終了したとき、前記CPUの動作周波数を元の周波数に戻す制御を行う動作周波数制御部と
を備える電子機器。
【請求項２】
前記動作周波数制御部は、前記ガベージコレクションの実行時間を予測可能な予測情報に基づいて、前記ガベージコレクションの実行時間を予測し、当該実行時間に応じて、前記CPUの動作周波数を制御する
請求項１に記載の電子機器。
【請求項３】
前記予測情報は、実行済みのガベージコレクションの実行時間に関する実行時間情報、及び、次に実行されるガベージコレクションの種類に関する種類情報の少なくとも１つを含む
請求項２に記載の電子機器。
【請求項４】
前記動作周波数制御部は、前記実行時間情報に基づいて、前記ガベージコレクションの実行時に、当該ガベージコレクションにおける前記CPUの動作周波数の制御方法を決定する
請求項３に記載の電子機器。
【請求項５】
前記動作周波数制御部は、前記実行時間情報に基づいて、前記プログラムの実行時に、前記ヒープに対するガベージコレクションにおける前記CPUの動作周波数の制御方法を決定する
請求項３に記載の電子機器。
【請求項６】
前記種類情報は、前記ガベージコレクションのアルゴリズムに関する情報、及び、前記ガベージコレクションの対象となる前記ヒープに関する情報の少なくとも１つを含む
請求項３に記載の電子機器。
【請求項７】
前記動作周波数制御部は、前記CPUの動作周波数を下げる制御が行われている間に、前記メインメモリを制御するメモリコントローラの動作周波数を上げる制御を行う
請求項１に記載の電子機器。
【請求項８】
前記動作周波数制御部は、前記CPUの動作電圧を制御可能である場合、前記CPUの動作周波数を下げる制御が行われている間に、前記CPUの動作電圧を下げる制御を行う
請求項１に記載の電子機器。
【請求項９】
動作周波数を制御可能なCPUと、前記CPUにより実行されるプログラムで使用するデータを記憶するメインメモリと、前記CPUの動作周波数を制御する動作周波数制御部とを有する電子機器の制御方法において、
前記動作周波数制御部が、
前記メインメモリにおける前記プログラムが動的にメモリを確保する領域であるヒープに対するガベージコレクションの実行が開始される前あるいは途中に、前記CPUの動作周波数を下げる制御を行い、前記ガベージコレクションを前記CPUの動作周波数が下げられ

た状態で実行させ、前記ガベージコレクションの実行の途中あるいは終了したとき、前記CPUの動作周波数を元の周波数に戻す制御を行うステップを含む
制御方法。
【請求項１０】
コンピュータを、
動作周波数を制御可能なCPUにより実行されるプログラムで使用するデータを記憶するメインメモリにおける前記プログラムが動的にメモリを確保する領域であるヒープに対する、ガベージコレクションの実行が開始される前あるいは途中に、前記CPUの動作周波数を下げる制御を行い、前記ガベージコレクションを前記CPUの動作周波数が下げられた状態で実行させ、前記ガベージコレクションの実行の途中あるいは終了したとき、前記CPUの動作周波数を元の周波数に戻す制御を行う動作周波数制御部
として機能させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、電子機器、制御方法、及び、プログラムに関し、特に、消費電力を抑えることができるようにした電子機器、制御方法、及び、プログラムに関する。
【背景技術】
【０００２】
近年、スマートフォンに代表される携帯端末が普及している。この種の携帯端末は、バッテリ駆動となるため、プログラムの実行速度の向上とともに、消費電力の削減も重要となる。携帯端末が消費する電力のうち、CPU(Central Processing Unit)が消費する電力には、CPUが動作していなくても消費されるスタティック電力と、CPUが動作することにより消費されるダイナミック電力があり、これらの電力の合計が、CPUで消費される電力となる。
【０００３】
これらの電力のうち、ダイナミック電力は、おおよそCPUの動作周波数に比例することが知られている。したがって、CPUの動作周波数を下げることによって、消費電力を抑えることができる。しかしながら、その一方で、動作周波数を下げてしまうと、プログラムの実行速度は遅くなってしまう。そのため、最適なCPUの動作周波数は、プログラムが実行中であるかどうかや、どのようなプログラムを実行しているかによって異なることになる。
【０００４】
例えばインテル社のIntel x86やARM社のARMプロセッサなど、多くのCPUは、DVFS(Dynamic Voltage and Frequency Scaling)と呼ばれる、実行中にアプリケーション(プログラム)からCPUの動作周波数と動作電圧を変更できる機能を備えている。このDVFSを用いれば、プログラムの実行状態などの状況に応じて、CPUの動作周波数を変更することができる。例えば、プログラムを実行していないときは動作周波数を下げる一方、プログラムの実行中は動作周波数を高くするという制御が一般的に行われている。
【０００５】
また、このようなプログラムが実行中かどうかによって動作周波数を制御するよりも、さらに積極的な周波数制御の方法として、CPUのキャッシュミスが多いときに、動作周波数を下げるという制御が提案されている。具体的には、CPUがメモリアクセスを行うと、最初はメインメモリに直接アクセスする。そして、一度アクセスしたメモリアドレス周辺は、キャッシュメモリという高速な記憶装置に保存され、しばらくの間は、高速にアクセスできるようになる。キャッシュメモリにないデータにアクセスしようとすると、メインメモリへのアクセスが発生し、これがキャッシュミスと呼ばれるものである。
【０００６】
メインメモリへのアクセスには、CPUの命令実行に比べて長い時間がかかるため、キャッシュミスが発生すると、CPUは待たされることになる。そのため、キャッシュミスが頻

繁に起きる状況では、プログラムの実行中であっても、CPUが待っている状態が多くなり、動作周波数を下げるほうがよいことがある。そして、キャッシュミスが多いときに動作周波数を下げるには、キャッシュミスが多い状態が長時間継続する時間を見つける必要がある。キャッシュミスが多い状態が長時間継続する必要があるのは、仮にキャッシュミスが多い状態が短時間しか持続しなければ、動作周波数を変更しても効果が少ないからである。

【０００７】

このようなキャッシュミスに応じてCPUの動作周波数を制御する方法としては、例えば、特許文献１や特許文献２が知られている。特許文献１には、キャッシュミスを検出するハードウェアを用い、キャッシュミスが頻繁に起きているかどうかを監視して、CPUの動作周波数を制御する方法が開示されている。特許文献２には、オペレーティングシステム(以下、「ＯＳ」という。ＯＳは、Operating Systemの略)のカーネルがプログラムやハードウェアのイベントを検知し、そのイベントから今後キャッシュミスが頻繁に起きるかどうかを予測する方法が開示されている。このイベントとしては、プログラムがＯＳの機能を利用するときに行うシステムコールや、ハードウェアの割り込みなどがあり、当該イベントが起きるとCPUの動作周波数を下げることになる。

【０００８】

また、ガベージコレクション(以下、「ＧＣ」ということもある。ＧＣは、Garbage Collectionの略)と呼ばれる、CPUにより実行されるプログラムが動的にメモリを確保する領域(以下、「ヒープ」という)のうち、不要になった領域を解放する機能が知られている。

【０００９】

非特許文献１には、仮想機械(以下、「ＶＭ」という。ＶＭは、Virtual Machineの略)が行う処理の一つであるＧＣは、CPUの動作周波数が高く、大きなキャッシュメモリを搭載した高性能のCPUを用いて実行しても、消費電力が増える割には、あまり高速にならないことが示されている。ここでは、プログラムを実行するCPUとは別に、ＧＣを専用に行うCPUを追加することが提案されている。

【先行技術文献】

【特許文献】

【００１０】

【特許文献１】特開４８６０１０４号

【特許文献２】特開４８３７４５６号

【非特許文献】

【００１１】

【非特許文献１】Ting Cao, Stephan M.Blackburn, Kathryn S.McKinley:'The Yin and Yang of Power and Performance for Asymmetric Hardware and Managed Software',In proceedings of the 39th Annual International Symposium on Computer Architecture 40 (3),pp.225-236,2012

【発明の概要】

【発明が解決しようとする課題】

【００１２】

ところで、特許文献１に開示されている方法であると、キャッシュミスを検出するハードウェアが必要となる。また、キャッシュミスが頻繁に起きていることを精度よく判定するためには、キャッシュミスが頻繁に起きる状況がしばらく続いていることを確認する必要があり、実際にキャッシュミスが頻繁に起きる状況になってから、CPUの動作周波数を制御するまでには時間がかかるため、その間は、動作周波数が高い状態で実行が続くことになる。

【００１３】

また、特許文献２に開示されている方法では、ＯＳのカーネルにより検知されるイベントとして、キャッシュミスが頻繁に発生するまえぶれとなるイベントをプログラムしておいて、そのようなイベントが起きたときに、CPUの動作周波数を下げることになる。この

方法では、ＯＳがイベントを検出して、CPUの動作周波数を制御するため、ＯＳ上で実行されているＶＭやプログラム自身では、その中で閉じている処理の動作周波数を変更することはできない。例えば、携帯端末用のＯＳであるAndroid(登録商標)では、アプリケーション(プログラム)は、全てＶＭの上で実行される仕組みになっているため、ＯＳのカーネルによるイベントの検知だけでは十分ではない。

【００１４】

さらに、非特許文献１に開示されている方法では、通常のアプリケーション(プログラム)を高性能なCPU、ＧＣを低性能のCPUで処理することで、消費電力の向上を図っているが、ＧＣを専用に行うCPUを追加する必要がでてくる。

【００１５】

このように、特許文献１、２に開示されている方法では、適切にCPUの動作周波数の制御が行われているとは言い難く、また、非特許文献１に開示されている方法では、ＧＣの処理の実行時に、CPUの動作周波数の制御は行われていない。

【００１６】

本発明はこのような状況に鑑みてなされたものであり、ＧＣの処理の実行時におけるCPUの動作周波数を適切に制御して、消費電力を抑えることができるようにするものである。

【課題を解決するための手段】

【００１７】

本発明の一側面の電子機器は、動作周波数を制御可能なCPUと、前記CPUにより実行されるプログラムで使用するデータを記憶するメインメモリと、前記メインメモリにおける前記プログラムが動的にメモリを確保する領域であるヒープに対するガベージコレクションの実行が開始される前あるいは途中に、前記CPUの動作周波数を下げる制御を行い、前記ガベージコレクションを前記CPUの動作周波数が下げられた状態で実行させ、前記ガベージコレクションの実行の途中あるいは終了したとき、前記CPUの動作周波数を元の周波数に戻す制御を行う動作周波数制御部とを備える。

【００１８】

前記動作周波数制御部は、前記ガベージコレクションの実行時間を予測可能な予測情報に基づいて、前記ガベージコレクションの実行時間を予測し、当該実行時間に応じて、前記CPUの動作周波数を制御することができる。

【００１９】

前記予測情報は、実行済みのガベージコレクションの実行時間に関する実行時間情報、及び、次に実行されるガベージコレクションの種類に関する種類情報の少なくとも１つを含むようにすることができる。

【００２０】

前記動作周波数制御部は、前記実行時間情報に基づいて、前記ガベージコレクションの実行時に、当該ガベージコレクションにおける前記CPUの動作周波数の制御方法を決定することができる。

【００２１】

前記動作周波数制御部は、前記実行時間情報に基づいて、前記プログラムの実行時に、前記ヒープに対するガベージコレクションにおける前記CPUの動作周波数の制御方法を決定することができる。

【００２２】

前記種類情報は、前記ガベージコレクションのアルゴリズムに関する情報、及び、前記ガベージコレクションの対象となる前記ヒープに関する情報の少なくとも１つを含むようにすることができる。

【００２３】

前記動作周波数制御部は、前記CPUの動作周波数を下げる制御が行われている間に、前記メインメモリを制御するメモリコントローラの動作周波数を上げる制御を行うことができる。

【００２４】
前記動作周波数制御部は、前記CPUの動作電圧を制御可能である場合、前記CPUの動作周波数を下げる制御が行われている間に、前記CPUの動作電圧を下げる制御を行うことができる。
【００２５】
本発明の一側面の制御方法及びプログラムは、上述した本発明の一側面の電子機器に対応する制御方法及びプログラムである。
【００２６】
本発明の一側面の電子機器、制御方法、及び、プログラムにおいては、動作周波数を制御可能なCPUにより実行されるプログラムで使用するデータを記憶するメインメモリにおける前記プログラムが動的にメモリを確保する領域であるヒープに対する、ガベージコレクションの実行が開始される前あるいは途中に、前記CPUの動作周波数を下げる制御が行われて、前記ガベージコレクションが、前記CPUの動作周波数が下げられた状態で実行される。そして、前記ガベージコレクションの実行の途中あるいは終了したとき、前記CPUの動作周波数を元の周波数に戻す制御が行われる。
【発明の効果】
【００２７】
本発明の一側面によれば、少ない実行速度の低下で効果的に消費電力を抑えることができる。
【００２８】
なお、ここに記載された効果は必ずしも限定されるものではなく、本開示中に記載されたいずれかの効果であってもよい。
【図面の簡単な説明】
【００２９】
【図１】仮想機械によりプログラムを実行するシステムのアーキテクチャを示す図である。
【図２】CPUの動作周波数の制御を行わない場合のＧＣの処理の実行を示すタイミングチャートである。
【図３】CPUの動作周波数の制御を行った場合のＧＣの処理の実行を示すタイミングチャートである。
【図４】ＧＣの処理の実行時に、ＧＣの処理を考慮してCPUの動作周波数の制御を行った場合における各ベンチマークプログラムの消費電力の削減状況を示す図である。
【図５】各ベンチマークプログラムにおける平均ＧＣ時間を示す図である。
【図６】本発明を適用した電子機器の一実施の形態の構成を示すブロック図である。
【発明を実施するための形態】
【００３０】
以下、図面を参照しながら本発明の実施の形態について説明する。
【００３１】
図１は、仮想機械によりプログラムを実行するシステムのアーキテクチャを示す図である。
【００３２】
図１に示すように、ハードウェアとして、動作周波数を制御可能なCPUを搭載しているスマートフォン等の電子機器では、そのCPU上でオペレーティングシステム(ＯＳ)が動作し、さらにその上で各アプリケーション(プログラム)を動作させるための仮想機械(ＶＭ)が動作する。これにより、各プログラムは、ＶＭ上で実行されることになる。また、ＶＭは、ガベージコレクション(ＧＣ)の機能を備えており、プログラムが利用可能なメモリが少なくなってくると、ＧＣの処理が実行される。
【００３３】
ＯＳとしては、例えば、Linux(登録商標)が用いられ、Java(登録商標)のプログラムを実行する場合には、ＶＭとして、Java仮想機械(Java VM)が動作することになる。また、

近年、携帯端末用のＯＳとして、Android(登録商標)が普及しているが、Androidでは、全てのプログラムがDalvik仮想機械(Dalvik VM)と呼ばれるＶＭ上で動作し、ＧＣの機能も、このDalvik VMの中で動作することになる。

【００３４】

ここで、上述したように、ＧＣは、CPUにより実行されるプログラムが使用しているメモリの中から不要になったデータを自動的に探し出し、そのデータが占める領域を、再利用できるようにする機能である。

【００３５】

典型的には、プログラムが実行中に継続的にメモリを消費し、空きメモリがなくなったときに、ＧＣが呼び出される。ＧＣが呼び出されると、プログラム本体の処理が停止して、ＧＣの処理が実行される。そして、ＧＣの処理が終了すると、再度、プログラム本体に制御が戻される。ただし、これらの処理は全て、ＶＭ上で閉じた処理となっているため、ＯＳのカーネルからは検出することはできない。

【００３６】

ＧＣでは、不要になったデータを探すため、プログラムが使用しているメモリ全体を不規則な順序でアクセスする。そのため、ＧＣは、まとまった量の処理となりで、かつ、キャッシュミスが発生しやすいという特徴を有している。また、広範囲なメモリへのランダムアクセスは、キャッシュヒットしやすい狭い範囲へのランダムアクセスと比べて、CPUの動作周波数を下げて実行しても実行速度に与える影響が少ないということが知られている。

【００３７】

すなわち、メモリに対してアクセスする範囲が広いときに、CPUが、より低い動作周波数で実行されるようにすることで、アクセスする範囲が狭いときに比べて、より少ない実行速度の低下で、より大きい消費電力の削減ができることが期待される。そこで、本発明では、ＧＣの処理の実行が開始される前に、CPUの動作周波数を下げ、ＧＣの処理の実行が終了したとき、CPUの動作周波数を元の周波数に戻す制御を行うことで、キャッシュミスが頻発すると予想されるＧＣの処理を、より低い動作周波数で実行するようにして、消費電力を抑えることができるようにする。

【００３８】

図２は、CPUの動作周波数の制御を行わない場合のＧＣの実行を示すタイミングチャートである。図２において、上側の図２Ａは、時間ごとのCPUの動作周波数を表しており、その時間軸の方向は、図中の左から右に向かう方向とされる。また、下側の図２Ｂは、図２Ａと同一の時間軸において、CPUにより実行される処理の内容を表している。この処理の内容としては、プログラム本体の処理（図中の「通常の処理」）と、ＧＣの処理（図中の「ＧＣの処理」）があり、図２には、これらの処理を実行する際のCPUの動作周波数が時系列で示されていることになる。

【００３９】

すなわち、CPUによって、時刻t0から時刻t1までの間は通常の処理、時刻t1から時刻t2までの間はＧＣの処理がそれぞれ実行され、その後も、通常の処理とＧＣの処理が繰り返し実行されるが、その間における動作周波数は一定となる。このように、図２では、CPUにより実行される処理の内容が通常の処理であろうが、ＧＣの処理であろうが、CPUの動作周波数は一定の周波数となっている。

【００４０】

一方、図３には、CPUの動作周波数の制御を行った場合のＧＣの実行を示すタイミングチャートを図示している。図３Ａと図３Ｂの関係は、上述した図２Ａと図２Ｂとの関係と同様であるが、図３では、CPUの動作周波数の制御を行っているため、CPUにより実行される処理の内容によって、CPUの動作周波数が異なっている。

【００４１】

すなわち、時刻t0から時刻t1までの間は、CPUにより通常の処理が実行され、時刻t1から時刻t2までの間は、CPUによりＧＣの処理が実行されるが、その動作周波数は、通常の

処理よりもＧＣの処理のほうが低くなるように制御されている。同様に、時刻t2～t3、時刻t4～t5、時刻t6～t7、時刻t8～t9、時刻t10～t11、時刻t12～t13に実行される通常の処理と、時刻t3～t4、時刻t5～t6、時刻t7～t8、時刻t9～t10、時刻t11～t12、時刻t13～t14に実行されるＧＣの処理と比べると、通常の処理の実行時よりもＧＣの処理の実行時のほうがCPUの動作周波数が低くなっている。

【００４２】

このように、本発明では、図２に示したように、通常の処理とＧＣの処理を実行するCPUの動作周波数を同一の周波数とするのではなく、図３に示したように、ＧＣの処理の実行が開始される前に、CPUの動作周波数を下げ、当該ＧＣの処理が終了したとき、CPUの動作周波数を元の周波数に戻して、通常の処理が再開されるようにすることで、キャッシュミスが頻発すると予想されるＧＣの処理を、より低い動作周波数で実行するようにしている。その結果、ＧＣの処理の実行時における消費電力を抑えることができる。

【００４３】

なお、図２及び図３においては、ＧＣの処理の実行時には、通常の処理の実行を停止するものとして説明しているが、ＧＣの処理は、通常の処理と並行して実行されるようにしてもよい。

【００４４】

ところで、ＧＣの処理の実行時に、このようなCPUの動作周波数の制御を行うことで、消費電力が抑えられることは、本発明の発明者により行われた詳細なるシミュレーションの結果、見出されたものである。そこで、次に、図４及び図５を参照して、当該シミュレーションの詳細について説明する。

【００４５】

なお、当該シミュレーションは、次の環境で行われている。すなわち、当該シミュレーションでは、ワンボードコンピュータのパンダボード(Pandaboard)を用い、このパンダボード上にAndroid(登録商標)を動作させている。したがって、全てのプログラムは、Dalvik VMと呼ばれる仮想機械上で動作し、ＧＣの機能も、このDalvik VMの中で動作することになる。

【００４６】

また、パンダボードは、以下のような特徴を有している。

【００４７】

・CPU：OMAP 4430(ARMアーキテクチャ)
・CPU周波数：300MHz～1008MHz
・キャッシュメモリ：１次 32KB(データ・命令別)/２次 1MB(データ・命令共通)
・メインメモリ：1GB

【００４８】

このような環境で、６つのベンチマークプログラムを実行して、さらに、CPUの動作周波数の制御を行った。ここでは、ＧＣの処理の実行中にだけCPUの動作周波数を下げた場合の実行（以下、「実行Ａ」という）の消費電力と、ＧＣの処理を考慮せずに、実行Ａと同じ時間で処理が終了するように、CPUの動作周波数を下げた場合の実行（以下、「実行Ｂ」という）の消費電力との比を比較している。つまり、実行Ａと実行Ｂのうち、実行Ａが、本発明を適用したCPUの動作周波数の制御方法に相当するものである。

【００４９】

図４には、ＧＣの処理の実行時に、ＧＣの処理を考慮してCPUの動作周波数の制御を行った場合における各ベンチマークプログラムの消費電力の削減状況を示している。図４においては、縦軸は、実行Ｂを基準としたときの実行Ａの消費電力の比を示しており、この比が、１より小さい場合には、実行Ｂよりも実行Ａを用いた方が、消費電力が少ないことを意味している。また、横軸の「antlr」、「hsqldb」、「luindex」、「lusearch」、「pmd」、「xalan」は、各ベンチマークプログラムを示している。

【００５０】

すなわち、図４には、６つのベンチマークプログラムごとに、本発明を適用したCPUの

動作周波数の制御方法を用いた場合における消費電力の削減状況を示しているが、特に、hsqldb、pmdは、消費電力の比が、0.8前後となっており、本発明を適用したCPUの動作周波数の制御方法による消費電力の削減の効果が大きいことが確認できる。

【００５１】

図５は、各ベンチマークプログラムにおける平均ＧＣ時間を示している。図５においては、６つのベンチマークプログラムごとの、総ＧＣ時間(s)、ＧＣ回数、及び、平均ＧＣ時間(s)が示されている。なお、総ＧＣ時間は、ＧＣの処理にかかった総時間を示し、ＧＣ回数は、ＧＣの処理を行った回数を示している。また、平均ＧＣ時間は、１回のＧＣの処理にかかる時間の平均であって、総ＧＣ時間をＧＣ回数で割ることで得られる値である。

【００５２】

図５に示すように、antlrは、総ＧＣ時間が84.61(s)で、ＧＣ回数が719(回)であるから、平均ＧＣ時間は、0.118(s)となる。hsqldbは、総ＧＣ時間が113.79(s)で、ＧＣ回数が19(回)であるから、平均ＧＣ時間は、5.99(s)となる。luindexは、総ＧＣ時間が39.39(s)で、ＧＣ回数が204(回)であるから、平均ＧＣ時間は、0.193(s)となる。

【００５３】

また、lusearchは、総ＧＣ時間が272.32(s)で、ＧＣ回数が914(回)であるから、平均ＧＣ時間は、0.298(s)となる。pmdは、総ＧＣ時間が279.98(s)で、ＧＣ回数が336(回)であるから、平均ＧＣ時間は、0.833(s)となる。xalanは、総ＧＣ時間が68.82(s)で、ＧＣ回数が312(回)であるから、平均ＧＣ時間は、0.221(s)となる。

【００５４】

ここで、図４において、本発明を適用したCPUの動作周波数の制御方法による消費電力の削減の効果が大きいことが確認されたhsqldb、pmdに注目すれば、平均ＧＣ時間がそれぞれ、5.99(s)、0.833(s)となり、他のベンチマークプログラムに比べて長いことが分かる。したがって、ＧＣの処理の時間が長い場合に、そのＧＣの処理の実行中にだけ、選択的にCPUの動作周波数を下げることは、消費電力を抑える上でその効果が大きいと考えられる。そして、プログラムが使用するデータの量が急激に変化することは少ないと考えられることから、過去のＧＣの処理にかかった時間に基づいて、次のＧＣの処理でCPUの動作周波数の制御を行うかどうかを決定することが有効であると考えられる。

【００５５】

すなわち、実行済みのＧＣの処理について、その処理の実行にかかった時間に関する情報（以下、「実行時間情報」という）を、プログラムごとに保持することで、次のＧＣの処理を実行する際に、その実行時間情報を参照して、CPUの動作周波数の制御を行うかどうかを決定することができる。例えば、図５の平均ＧＣ時間を、実行時間情報として保持しておくことで、hsqldbやpmdにおけるＧＣでは、平均ＧＣ時間が長く、消費電力を抑えることが期待されるので、ＧＣの処理に応じて、CPUの動作周波数の制御を行うことが可能となる。

【００５６】

ここで、CPUの動作周波数の制御方法を決定するタイミングであるが、任意のタイミングで行うことができる。例えば、ＧＣの処理の実行時に、実行時間情報を参照して、当該ＧＣの処理におけるCPUの動作周波数の制御を行うかどうかを決定することができる。また、hsqldbやpmd等のプログラムの実行時に、実行時間情報を参照して、ヒープに対するＧＣの処理におけるCPUの動作周波数の制御を行うかどうかを決定するようにしてもよい。

【００５７】

このような、次のＧＣの処理の実行時間を予測可能な予測情報としては、上述した実行時間情報のほか、例えば、次のＧＣの処理の種類に関する情報（以下、「種類情報」という）を用いるようにしてもよい。この種類情報としては、例えば、ＧＣの処理のアルゴリズムに関する情報や、ＧＣの処理が実行されるヒープに関する情報を用いることができる。

【００５８】
具体的には、世代別ＧＣ(Generational Garbage Collection)では、ヒープの不要なデータが多くなる傾向にある領域のみを対象としたマイナーＧＣの処理を頻繁に行い、ヒープの全領域を対象とするメジャーＧＣをたまに行うことで、効率よくＧＣが行われるようにしている。この世代別ＧＣにおいては、マイナーＧＣの処理にはあまり時間がかからないが、メジャーＧＣの処理では、長い時間がかかることが知られている。したがって、本発明を適用したCPUの動作周波数の制御方法では、このようなＧＣの対象となるヒープによるＧＣの処理の実行時間の違いから、メジャーＧＣの処理でのみ、CPUの動作周波数を下げる制御が行われるようにする。
【００５９】
また、ヒープの使用状況によって、不要なデータのメモリ領域を再利用するだけでなく、使用中のデータを再配置するようなＧＣも存在し、このようなＧＣにおいて、データの再配置が発生した場合には、ＧＣの処理に長い時間がかかることになるので、この場合には、CPUの動作周波数を下げる制御が行われるようにする。
【００６０】
以上のように、ＧＣの処理の実行時間が短い場合には、ＧＣの処理を、CPUの動作周波数を下げて実行しても効果が少ないので、多少の消費電力削減より実行速度を優先させたい場合は、本発明では、次のＧＣの処理の実行時間を予測可能な予測情報（例えば、実行時間情報や種類情報など）を用い、次のＧＣの処理の実行時間を予測して、当該ＧＣの処理におけるCPUの動作周波数の制御を行うかどうかを決定することができるようにしている。次に、このようなCPUの動作周波数の制御方法を実現可能な電子機器について説明する。
【００６１】
図６は、本発明を適用した電子機器の一実施の形態の構成を示すブロック図である。
【００６２】
図６の電子機器１０は、例えば、スマートフォン、タブレット端末、携帯電話機、眼鏡型や腕時計型のウェアラブルコンピュータなどの携帯端末である。なお、電子機器１０は、携帯端末に限らず、ノート型パーソナルコンピュータやパーソナルコンピュータ、データセンタ等に配置されるサーバ装置など、メモリ管理の機能として、ＧＣの機能を備えた機器であればよい。
【００６３】
図６に示すように、電子機器１０は、CPU１０１、メモリコントローラ１０２、メインメモリ１０３、キャッシュメモリ１０４、メモリ管理部１０５、動作周波数制御部１０６、及び、実行時間情報保持部１０７を含むようにして構成される。なお、実際には、電子機器１０には、ディスプレイやスピーカ、操作部、通信部、記録部などの機能も設けられるが、ここでは、説明を簡略化するため、それらの機能については省略している。
【００６４】
CPU１０１は、電子機器１０の各部の動作を制御する。また、CPU１０１は、各種のプログラムを実行する。なお、CPU１０１は、動作周波数と動作電圧を制御することが可能である。
【００６５】
メモリコントローラ１０２は、CPU１０１とメインメモリ１０３との間に設けられ、CPU１０１からの制御に従い、メインメモリ１０３におけるデータの読み込みや書き出しを行う。
【００６６】
メインメモリ１０３は、メモリコントローラ１０２からの制御に従い、CPU１０１により実行されるプログラムで使用されるデータを記憶するRAM(Random Access Memory)である。メインメモリ１０３は物理メモリであるが、CPU１０１によるプログラムの実行時には、ＶＭなどによって、ヒープ領域１２１が割り当てられる。
【００６７】

キャッシュメモリ１０４は、CPU１０１とメインメモリ１０３との間に設けられ、使用頻度の高いデータを蓄積して、CPU１０１による低速なメインメモリ１０３へのアクセスを減らすことで、CPU１０１により実行される処理の速度を高速化する。

【００６８】

メモリ管理部１０５は、メインメモリ１０３の管理を行う。具体的には、例えば、メモリ管理部１０５は、メインメモリ１０３におけるヒープ領域１２１に対するＧＣの処理を行う。メモリ管理部１０５は、ＧＣの処理の実行を開始するとき、その開始を動作周波数制御部１０６に通知する。また、メモリ管理部１０５は、ＧＣの処理の実行を終了するとき、その終了を動作周波数制御部１０６に通知する。

【００６９】

なお、メモリ管理部１０５によるＧＣの処理は、ＶＭの一機能として提供されるほか、例えば、ＧＣの処理をライブラリとして提供して、プログラムにリンクさせるようにしてもよい。

【００７０】

動作周波数制御部１０６は、メモリ管理部１０５からの通知に応じて、CPU１０１の動作周波数を制御する。具体的には、動作周波数制御部１０６は、メモリ管理部１０５からのＧＣの処理の開始の通知に応じて、CPU１０１の動作周波数を下げる制御を行う。また、動作周波数制御部１０６は、メモリ管理部１０５からのＧＣの処理の終了の通知に応じて、CPU１０１の動作周波数を元の周波数に戻す制御を行う。

【００７１】

すなわち、動作周波数制御部１０６が、メモリ管理部１０５からのＧＣの処理の開始の通知と終了の通知に応じて、CPU１０１の動作周波数の制御を行うことで、例えば、図３に示したように、時刻t0～t1や時刻t2～t3などの通常の処理と、時刻t1～t2や時刻t3～t4などのＧＣの処理を比べると、通常の処理の実行時よりもＧＣの処理の実行時のほうが、CPU１０１の動作周波数が低くなることになる。なお、CPU１０１の動作周波数が下げられた状態で、所与の時間、ＧＣの処理が実行されるように制御されるのであれば、動作周波数制御部１０６がCPU１０１の動作周波数を下げるタイミングは、ＧＣの処理の途中でもよい。また、下げられていたCPU１０１の動作周波数を上げるタイミングも、ＧＣの処理の途中でもよい。

【００７２】

また、動作周波数制御部１０６は、メモリ管理部１０５からのＧＣの処理の開始の通知と終了の通知に応じて、ＧＣの処理にかかった時間を計時し、実行時間情報として、実行時間情報保持部１０７に保持する。これにより、実行時間情報保持部１０７には、CPU１０１により実行されたプログラムごとに、実行済みのＧＣの処理にかかった時間が保持されることになる。

【００７３】

動作周波数制御部１０６は、メモリ管理部１０５によるＧＣの処理の実行時や、CPU１０１によるプログラムの実行時などに、実行時間情報保持部１０７に保持された実行時間情報に基づいて、ＧＣの処理の実行時に、CPU１０１の動作周波数の制御を行うかどうかを決定する。そして、動作周波数制御部１０６は、CPU１０１の動作周波数の制御を行うと決定したときには、ＧＣの処理の実行時に、CPU１０１の動作周波数を下げる制御を行う。なお、ここでは、例えば、対象のプログラムについて、直近のＧＣの処理（例えば、対象のプログラムにおける１回前のＧＣの処理）の実行時間が所定の閾値を超えるかどうかを判定することなどにより、CPU１０１の動作周波数の制御を行うかどうかが判定される。

【００７４】

また、メモリ管理部１０５は、自身が実行するＧＣの処理のアルゴリズムや、ＧＣの処理が実行されるヒープ領域１２１に関する種類情報を、適宜、動作周波数制御部１０６に供給する。動作周波数制御部１０６は、メモリ管理部１０５からの種類情報に基づいて、ＧＣの処理の実行時に、CPU１０１の動作周波数の制御を行うかどうかを決定する。そし

て、動作周波数制御部１０６は、CPU１０１の動作周波数の制御を行うと決定したときには、ＧＣの処理の実行時に、CPU１０１の動作周波数を下げる制御を行う。

【００７５】

このように、動作周波数制御部１０６は、次のＧＣの処理の実行時間を予測可能な予測情報（例えば、実行時間情報及び種類情報の少なくとも１つ）に基づいて、ＧＣの実行時間を予測し、当該実行時間に応じて、CPU１０１の動作周波数を制御することができる。

【００７６】

なお、動作周波数制御部１０６は、メモリ管理部１０５からの通知に応じて、メモリコントローラ１０２の動作周波数を制御するようにしてもよい。具体的には、動作周波数制御部１０６は、CPU１０１の動作周波数を下げる制御が行われている間に、メモリコントローラ１０２の動作周波数を上げる制御を行うことになる。これにより、ＧＣの処理を高速に行うことができる。

【００７７】

また、動作周波数制御部１０６は、メモリ管理部１０５からの通知に応じて、CPU１０１の動作電圧を制御するようにしてもよい。具体的には、動作周波数制御部１０６は、CPU１０１の動作周波数を下げる制御が行われている間に、CPU１０１の動作電圧を下げる制御を行うことになる。これにより、消費電力をさらに抑えることができる。

【００７８】

以上のように構成される電子機器１０においては、動作周波数制御部１０６によって、メインメモリ１０３から割り当てられたヒープ領域１２１に対するＧＣの処理の実行が開始される前に、CPU１０１の動作周波数を下げる制御が行われ、当該ＧＣの処理の実行が終了したとき、CPU１０１の動作周波数を元の周波数に戻す制御が行われる。

【００７９】

このようなCPU１０１の動作周波数の制御を行うことで、キャッシュミスが頻発すると予想されるＧＣの処理を、より低い動作周波数で実行することになるので、消費電力を抑えることができる。

【００８０】

また、このようなCPU１０１の動作周波数の制御を行うことで、CPU１０１が消費するダイナミック電力（CPU１０１の動作に伴って消費される電力）と、プログラムの実行速度のトレードオフをよくすることができる。例えば、同一のプログラムを、同一の時間で実行する場合に、より少ない消費電力で実行することが可能となる。また、キャッシュヒット率の変化から遅延なく、かつ、ハードウェアを追加することなく、CPUの動作周波数を変更することができる。

【００８１】

上述した一連の処理は、ハードウェアにより実行することもできるし、ソフトウェアにより実行することもできる。一連の処理をソフトウェアにより実行する場合には、そのソフトウェアを構成するプログラムが、コンピュータにインストールされる。ここで、コンピュータには、専用のハードウェアに組み込まれているコンピュータや、各種のプログラムをインストールすることで、各種の機能を実行することが可能な、例えば汎用のパーソナルコンピュータなどが含まれる。

【００８２】

このコンピュータでは、例えば、ROM(Read Only Memory)や記録部にインストールされたプログラムが、RAMにロードされて実行されることにより、上述した一連の処理が行われる。なお、コンピュータが実行するプログラムは、例えば、パッケージメディア等としてのリムーバブルメディアに記録して提供することができる。また、プログラムは、ローカルエリアネットワーク、インターネット、デジタル衛星放送といった、有線又は無線の伝送媒体を介して提供することができる。その他、プログラムは、ROMや記録部に、あらかじめインストールしておくことができる。

【００８３】

なお、本発明の実施の形態は、上述した実施の形態に限定されるものではなく、本発明

の要旨を逸脱しない範囲において種々の変更が可能である。

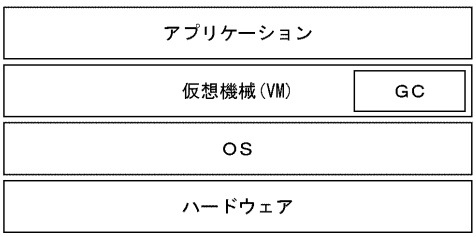
【符号の説明】

【００８４】

１０　電子機器，　１０１　CPU，　１０２　メモリコントローラ，　１０３　メインメモリ，　１０４　キャッシュメモリ，　１０５　メモリ管理部，　１０６　動作周波数制御部，　１０７　実行時間情報保持部，　１２１　ヒープ領域

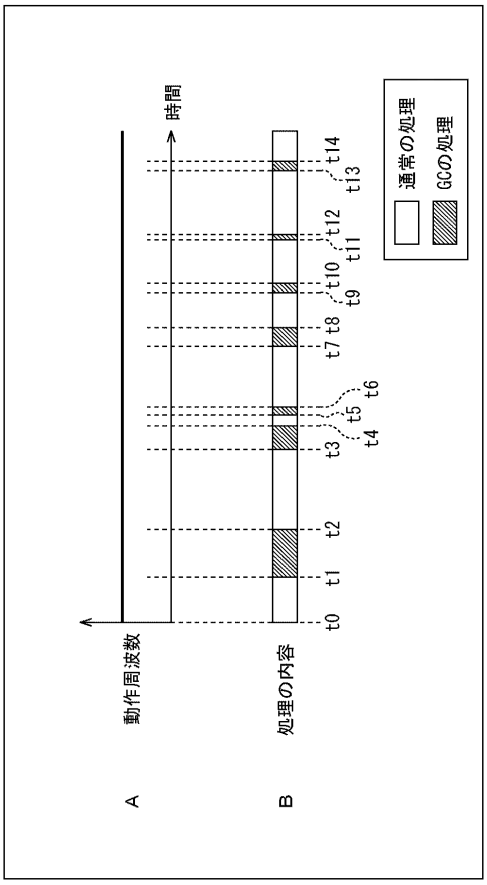
【図１】

図1



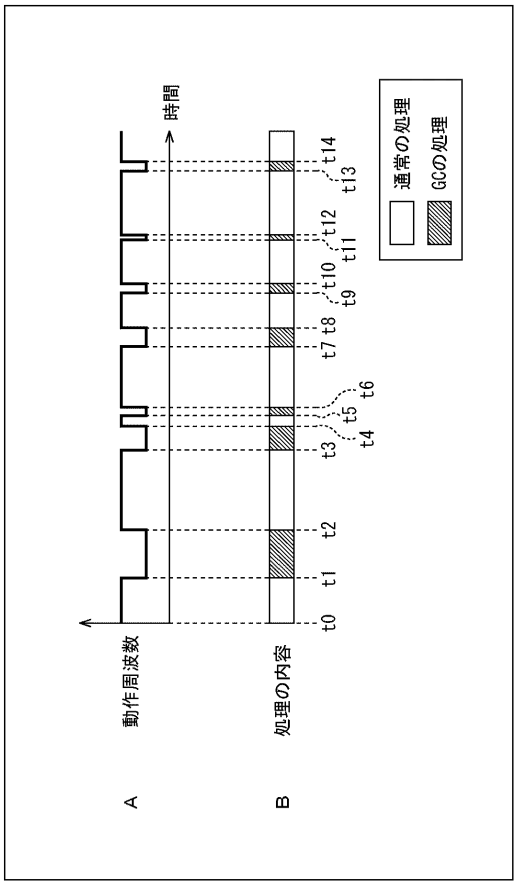
【図２】

図2



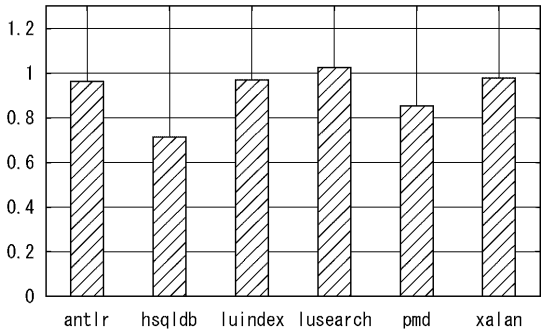
【図３】

図3



【図４】

図4



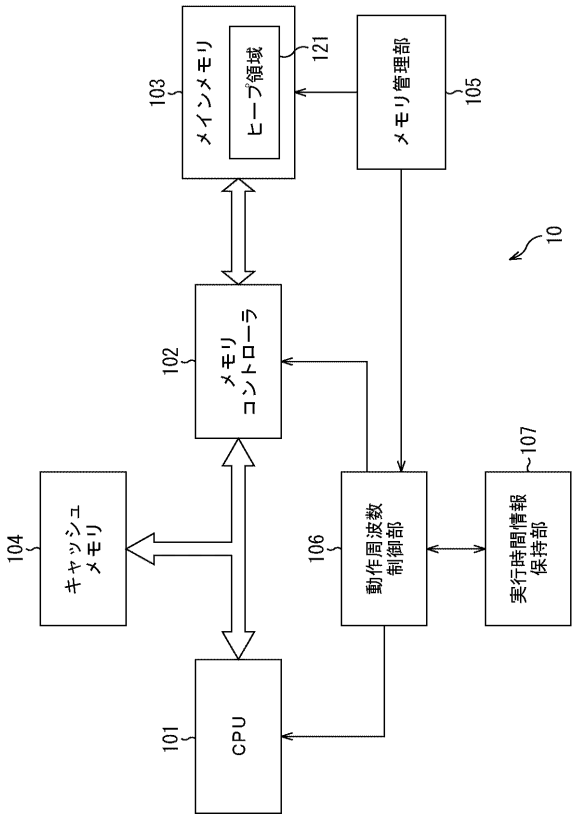
【図５】

図5

ベンチマーク	antlr	hsqldb	luindex	lusearch	pmd	xalan
総GC時間[s]	84.61	113.79	39.39	272.32	279.98	68.82
GC回数	719	19	204	914	336	312
平均GC時間[s]	0.118	5.99	0.193	0.298	0.833	0.221

【図６】

図6



フロントページの続き

(72)発明者　岩崎　英哉
東京都調布市調布ケ丘一丁目５番地１　国立大学法人電気通信大学内

Ｆターム(参考) 5B060 AA10　CC03