(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2004-38715
(P2004-38715A)
(43)公開日　平成16年2月5日(2004.2.5)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 15/177	G06F	15/177	674B	5B045
G06F 9/46	G06F	9/46	360B	5B098
	G06F	9/46	360C	

審査請求 未請求　請求項の数 1　OL　（全 7 頁）

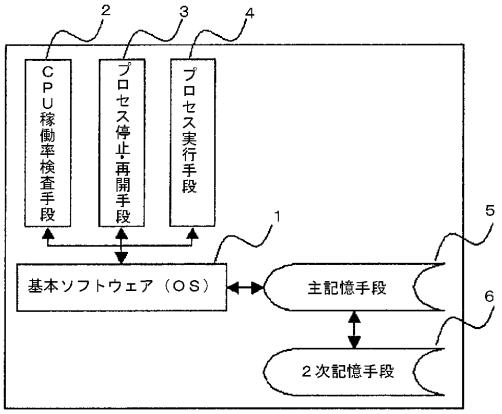
(21)出願番号　特願2002-196882(P2002-196882)
(22)出願日　平成14年7月5日(2002.7.5)

(71)出願人　390023928
日立エンジニアリング株式会社
茨城県日立市幸町３丁目２番１号
(74)代理人　100093872
弁理士　高崎　芳紘
(72)発明者　吉川　暁
茨城県日立市幸町３丁目２番１号　日立エンジニアリング株式会社内

Fターム(参考)　5B045 BB28　BB42　GG04
5B098 AA10　GA04　GC01　GC08

(54)【発明の名称】並列計算処理方法

(57)【要約】
【課題】並列計算機の処理能力を有効に活用してＣＰＵ稼働率を高めた並列計算処理方法を提供する。
【解決手段】各々の中央演算装置のプロセスに対する稼働率を監視し、全中央演算装置のプロセスに対する稼働率の合計値が所定値より小さい場合に実行状態のプロセス数を減らすか、または新たに実行状態のプロセスを発生させるかの実行状態のプロセス数の増減を行ない、改めて各々の中央演算装置のプロセスに対する稼働率を評価して実行状態のプロセス数の増減以前との全中央演算装置のプロセスに対する稼働率の合計値を比較し、全中央演算装置のプロセスに対する稼働率の合計値が小さくなった場合には実行状態のプロセスの増減以前と同じ状態に戻すプロセス停止再開手段３を設け、このプロセス停止再開手段３により中央演算装置のプロセスに対する稼働率の合計値を最大にする。
【選択図】　図１



【特許請求の範囲】
【請求項１】
複数の中央演算装置を有する並列計算機で複数のプロセスを同時に処理する並列計算処理方法において、各々の上記中央演算装置のプロセスに対する稼働率を監視し、上記全中央演算装置のプロセスに対する稼働率の合計値から実行状態のプロセス数の増減を行ない、改めて各々の上記中央演算装置のプロセスに対する稼働率を評価して実行状態のプロセス数の増減以前との上記全中央演算装置のプロセスに対する稼働率の合計値を比較し、上記全中央演算装置のプロセスに対する稼働率の合計値が小さくなった場合に実行状態のプロセスの増減以前と同じ状態に戻すことを特徴とする並列計算処理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、複数の中央演算装置（以後、ＣＰＵと称する）を有する並列計算機による並列計算処理方法に関する。
【０００２】
【従来の技術】
並列計算機は、ＳＭＰタイプ、ＭＰＰタイプ、およびクラスタタイプの３種類に大別される。ＳＭＰタイプは、１つのＯＳを利用して１つの共有メモリーに複数のＣＰＵを接続する構成の並列計算機であり、ＭＰＰタイプは、１つのＯＳを利用して複数のメモリーに１つないしは複数のＣＰＵを接続する構成の並列計算機である。またクラスタタイプは、単一または複数のＣＰＵを持つ計算機をネットワークにより複数台接続する構成の並列計算機である。このような並列計算機は、従来から科学技術、産業、流通、金融などの多くの分野において利用されている。
【０００３】
しかし、一般的に並列計算機の並列計算処理時には、各々のＣＰＵ稼働率は１００％とはならず、諸所の原因にて稼働率は１００％より小さい。これらの原因には、負荷バランスの不均等、通信などの同期待ち、Ｉ／Ｏ待ちなどがあげられる。この負荷バランスの不均等に対して、特にクラスタタイプの並列計算機においては、例えば、ＥｎＦｕｚｉｏｎ（ＴｕｒｂｏＬｉｎｕｘ社）やＬＳＦ（プラットフォーム社）といった負荷分散および負荷平準化ユーティリティを用いることにより、ＣＰＵを効率良く利用する工夫が行われている。また通信などの同期待ちに対しては、例えば、特開平７－９３２６５号公報に記載のようにＣＰＵの無駄時間を無くすという工夫が行われている。
【０００４】
【発明が解決しようとする課題】
しかしながら、従来の負荷分散および負荷平準化ユーティリティを用いた並列計算処理方法は、空きＣＰＵや負荷の低いＣＰＵに対してプロセスをディスパッチするだけであり、ディスパッチ後の制御を行なわない。そのため、あるＣＰＵにプロセスをディスパッチした後に、そのＣＰＵが過負荷状態になり稼働率が低下したとしても、何等解決策を持たない。さらに、多くの並列処理方法の考え方は、単一のジョブに対して、そのジョブを如何にして高速に行なうかという観点で考えられたものであるため、全体時間最適化という観点が無い。ここで全体時間最適化とは、複数のジョブを含めた場合に、ＣＰＵ稼働率を高めることによりトータルとして計算時間を最短にすることである。
【０００５】
本発明の目的は、並列計算機の処理能力を有効に活用してＣＰＵ稼働率を高めた並列計算処理方法を提供することにある。
【０００６】
【課題を解決するための手段】
本発明は上記目的を達成するために、複数の中央演算装置を有する並列計算機で複数のプロセスを同時に処理する並列計算処理方法において、各々の上記中央演算装置のプロセスに対する稼働率を監視し、上記全中央演算装置のプロセスに対する稼働率の合計値から実

行状態のプロセス数の増減を行ない、改めて各々の上記中央演算装置のプロセスに対する稼働率を評価して実行状態のプロセス数の増減以前との上記全中央演算装置のプロセスに対する稼働率の合計値を比較し、全中央演算装置のプロセスに対する稼働率の合計値が小さくなった場合に実行状態のプロセスの増減以前と同じ状態に戻すことを特徴とする。

【０００７】

本発明による並列計算処理方法は、各々の中央演算装置のプロセスに対する稼働率を監視し、全中央演算装置のプロセスに対する稼働率の合計値から実行状態のプロセス数の増減を行ない、改めて各々の中央演算装置のプロセスに対する稼働率を評価して実行状態のプロセス数の増減以前との全中央演算装置のプロセスに対する稼働率の合計値を比較し、全中央演算装置のプロセスに対する稼働率の合計値が小さくなった場合に実行状態のプロセスの増減以前と同じ状態に戻すようにしたため、並列処理度数が大きくなることにより発生するメモリなどのリソース不足やプロセス間の同期待ちなどより、並列処理度数を増加させてもＣＰＵ稼働率が大きくならない場合、適切な並列処理度数を自動的に選択して処理することができ、並列計算機の処理能力を有効に活用してＣＰＵ稼働率を高めることができ、並列計算機の保有コストの低減や並列計算機の有効利用をはかることができる。

【０００８】

【発明の実施の形態】

以下、本発明の実施の形態を図面に基づいて説明する。

図１は、本発明の一実施の形態による並列計算処理方法を採用した並列計算機の要部を示すブロック構成図である。

並列計算機は、基本ソフトウェア（ＯＳ）１と、ＣＰＵ稼働率検査手段２と、プロセス停止再開手段３と、プロセス実行手段４と、主記憶手段５と、二次記憶手段６とを有して構成されている。

【０００９】

図２は、上述したプロセス停止再開手段３の処理動作を示すフローチャートである。

ここでＣＰＵ稼働率は、測定全体時間Δｔ＝１０秒で、プロセス実行にＣＰＵが利用された合計時間ｔｓ＝２＋２＋１＝５秒であったとすると、ＣＰＵ稼働率＝合計時間ｔｓ／測定全体時間Δｔ×１００（％）＝５／１０×１００（％）＝５０％となる。このＣＰＵ稼働率に注目すると、あるＣＰＵで実行されているプロセスがメモリバウンドやＩ／Ｏバウンドの場合、システムによるメモリスワップ処理やＩ／Ｏ待ち処理の時間が相対的に大きくなり、ＣＰＵ稼働率は小さくなり、逆に、プロセスがＣＰＵバウンドの場合、システムによるメモリスワップ処理やＩ／Ｏ待ち処理の時間が相対的に小さくなり、ＣＰＵ稼働率は大きくなる傾向を持っている。

【００１０】

まず、図３に示したタイミングチャートを用いてプロセス停止再開手段３の処理動作について説明する。同図は、ＣＰＵ数が４の並列計算機上で複数のプロセスを動作させたときの各ＣＰＵと合計のＣＰＵ稼働率を示したタイミングチャートである。

外部のオペレータ等からあるジョブの実行が時刻ｔ０に指示されると、基本ソフトウェア１により、プロセス実行手段４が各ＣＰＵ上でプロセスの実行を開始する。ここでは時刻ｔ０においてプロセス１～プロセス３がＣＰＵ１～ＣＰＵ３上で実行され、この時刻ｔ０以降ではＣＰＵ１～ＣＰＵ３の稼働率が各々１００％となり、合計で１００％×３＝３００％とになっている。

【００１１】

また時刻ｔ０より定期的に予め決められた測定全体時間Δｔの間隔でＣＰＵ稼働率検査手段２はＣＰＵ稼働率を監視し、ＣＰＵ稼働率検査手段２で得られたＣＰＵ稼働率により、プロセス停止再開手段４は図２に示したフローチャートに従って処理を実行する。ステップＳ１において合計ＣＰＵ稼働率＜所定値、かつ、１＜実効状態プロセス数か否か判定を行ない、ここで所定値が２００％に設定されていると仮定すると、合計ＣＰＵ稼働率（３００％）＞所定値（２００％）となり、ステップＳ８の終了となる。

【００１２】

次に、このジョブは時刻ｔ１以降において主記憶手段５の利用量が大きくなり、基本ソフトウェア１により主記憶手段５と二次記憶手段６との間でメモリースワッピングが発生し、時刻ｔ１以降では、ＣＰＵ１～ＣＰＵ３の稼働率が各々５０％となり、合計で５０％×３＝１５０％に小さくなったとする。

【００１３】

このときプロセス停止再開手段３は、ステップＳ１において、合計ＣＰＵ稼働率（１５０％）＜所定値（２００％）、かつ、１＜実行プロセス数（３）であるため、ステップＳ２へ進み時刻ｔ２でＣＰＵ３の実行プロセスを停止する。このｔ２以降では、プロセスが１つ減ったために主記憶手段５に余裕ができ、ＣＰＵ１～ＣＰＵ２の稼働率が９０％に大きくなったとする。従って、ステップＳ３で合計ＣＰＵ稼働率が大きくなったか否かが判断させると、合計ＣＰＵ稼働率は９０％×２＝１８０％となりステップＳ２以前より大きくなっているためステップＳ８で終了となる。

【００１４】

一方、時刻ｔ２においてプロセスが１つ減ったにも拘わらず主記憶手段５に十分な余裕ができない場合について、図４に示したタイミングチャートを用いて説明する。

時刻ｔ３～時刻ｔ５までは、図３の時刻ｔ０～時刻ｔ２と同様に推移する。しかし時刻ｔ５において主記憶手段５に十分な余裕ができないために、ＣＰＵ１～ＣＰＵ２の稼働率が７０％までしか大きくならなかったとする。この場合、ステップＳ３では合計ＣＰＵ稼働率が７０％×２＝１４０％と、ステップＳ２以前より小さくなっているためステップＳ４へ進み、先のステップＳ２で停止したＣＰＵ３のプロセスを時刻ｔ６で元に戻して再開する。さらにステップＳ５では時刻ｔ７で実行プロセス数を増やしてＣＰＵ４でプロセスを実行する。ここで、時刻ｔ７以降ＣＰＵ１～ＣＰＵ４の稼働率が５０％になったとすると、ステップＳ６で合計ＣＰＵ稼働率が大きくなったか否かが判定され、合計ＣＰＵ稼働率が４０％×４＝１６０％となりステップＳ５以前より大きくなっているため、ステップＳ８で終了となる。

【００１５】

また、時刻ｔ７においてプロセスが１つ増えたために主記憶手段５に十分な余裕が無くなった場合について、図５に示したタイミングチャートを用いて説明する。

時刻ｔ８～時刻ｔ１２までは、図４の時刻ｔ３～時刻ｔ７と同様に推移し、時刻ｔ１２において主記憶手段５に十分な余裕ができないために、ＣＰＵ１～ＣＰＵ４の稼働率が３０％までしか大きくならなかったとする。この場合、ステップＳ６では、合計ＣＰＵ稼働率が３０％×４＝１２０％となり、ステップＳ５以前より小さくなっているためステップＳ７へ進む。このステップＳ７では、ステップＳ５の処理を元に戻してＣＰＵ４のプロセスを停止し、その後、ステップＳ８で終了となる。

【００１６】

上述の説明から分かるように、並列処理度数が大きくなることにより発生するメモリなどのリソース不足やプロセス間の同期待ちなどより、並列処理度数を増加させてもＣＰＵ稼働率が大きくならない場合、適切な並列処理度数を自動的に選択して処理することができる。

【００１７】

また、Ｉ／Ｏバウンドの場合は使用するＣＰＵ数より大きな値を並列処理度数とすることで、ＣＰＵ稼働率を大きくすることも可能である。これは、ラウンドロビンＣＰＵ割り当て方式において、タイムクォンタムの設定値を減少させたときに、Ｉ／Ｏバウンドのプロセスの処理時間が相対的に短くなることと同じ効果を得られる。つまり、例えば１つのＣＰＵに対して２つのＩ／Ｏバウンドのプロセスを割り当てると、１つのプロセスがＩ／Ｏ処理を行なっている間に、もう一方のプロセスがＣＰＵを利用できるため、全体としてＣＰＵ稼働率が大きくなることを示している。さらに、単一のジョブの並列処理に対してだけでなく、複数の異なるジョブの同時並行処理に対しても有効である。

【００１８】

尚、上述した実施の形態では、合計ＣＰＵ稼働率と比較する所定値を設定し、合計ＣＰＵ

稼働率が所定値より小さい場合、実効状態のプロセス数を減らすか、または新たに実効状態のプロセスを発生させるかの実行状態のプロセス数の増減を行なうようにしたため、合計ＣＰＵ稼働率が所定値より大きい状態では、プロセス停止再開手段３による調整をあえて行なわないので、合計ＣＰＵ稼働率の望ましい状態での不安定な動作を繰り返すこともない。しかし、このような動作が許容されるなら、所定値を必ずしも設定する必要はない。

【００１９】

【発明の効果】

以上説明したように本発明の並列計算処理方法によれば、並列計算機の処理能力を有効に活用してＣＰＵ稼働率を高めることができ、並列計算機の保有コストの低減や並列計算機の有効利用をはかることができる。

【図面の簡単な説明】

【図１】本発明の一実施の形態による並列計算処理方法を採用した並列計算機の要部を示すブロック構成図である。

【図２】図１に示した並列計算機のプロセス停止再開手段の処理動作を示すフローチャートである。

【図３】図１に示した並列計算機による並列処理を示すタイミングチャートである。

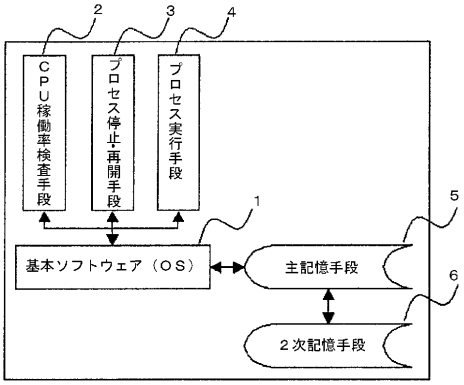
【図４】図１に示した並列計算機による他の並列処理を示すタイミングチャートである。

【図５】図１に示した並列計算機によるさらに他の並列処理を示すタイミングチャートである。

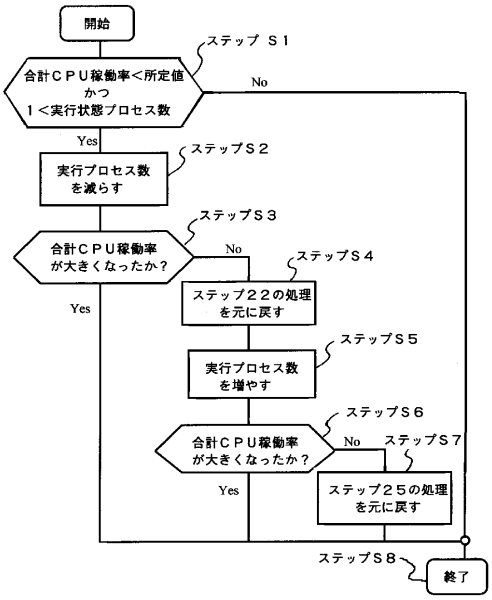
【符号の説明】

１　基本ソフトウェア（ＯＳ）
２　ＣＰＵ稼働率検査手段
３　プロセス停止再開手段
４　プロセス実行手段
５　主記憶手段
６　二次記憶手段

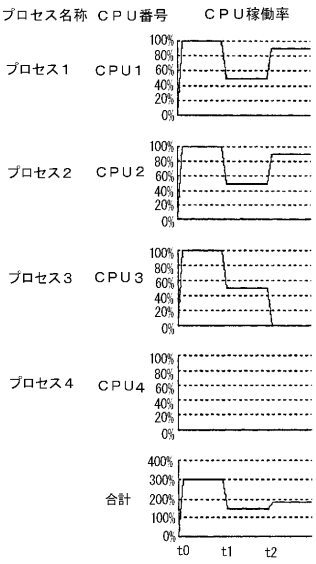
【図１】


2
3
4
ＣＰＵ稼働率検査手段
プロセス停止・再開手段
プロセス実行手段
1
5
6
基本ソフトウェア（ＯＳ）
主記憶手段
２次記憶手段

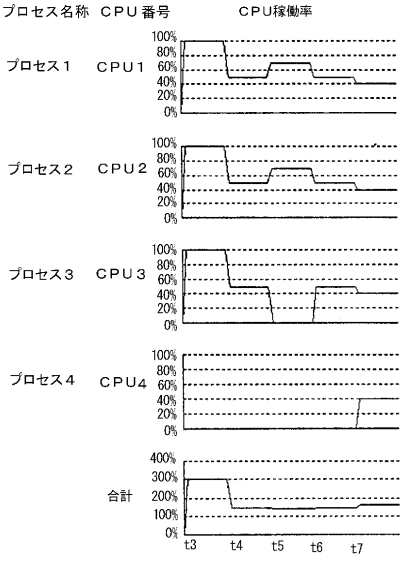
【図２】


開始
ステップ Ｓ１
合計ＣＰＵ稼働率＜所定値
かつ
１＜実行状態プロセス数
No
Yes
実行プロセス数
を減らす
ステップＳ２
ステップＳ３
合計ＣＰＵ稼働率
が大きくなったか？
No
ステップＳ４
ステップ２２の処理
を元に戻す
Yes
実行プロセス数
を増やす
ステップＳ５
ステップＳ６
合計ＣＰＵ稼働率
が大きくなったか？
No
ステップＳ７
Yes
ステップ２５の処理
を元に戻す
ステップＳ８
終了

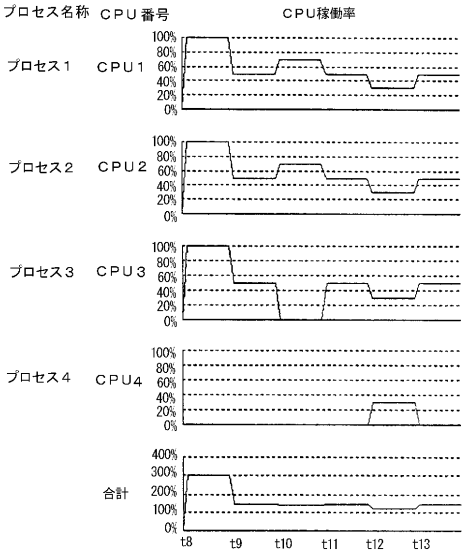
【図３】


プロセス名称
ＣＰＵ番号
ＣＰＵ稼働率
プロセス１
ＣＰＵ１
プロセス２
ＣＰＵ２
プロセス３
ＣＰＵ３
プロセス４
ＣＰＵ４
合計
100%
80%
60%
40%
20%
0%
400%
300%
200%
100%
t0
t1
t2

【図４】


プロセス名称
ＣＰＵ番号
ＣＰＵ稼働率
プロセス１
ＣＰＵ１
プロセス２
ＣＰＵ２
プロセス３
ＣＰＵ３
プロセス４
ＣＰＵ４
合計
100%
80%
60%
40%
20%
0%
400%
300%
200%
100%
t3
t4
t5
t6
t7

【図５】


プロセス名称
CPU番号
CPU稼働率
プロセス１
CPU１
プロセス２
CPU２
プロセス３
CPU３
プロセス４
CPU４
合計
100%
80%
60%
40%
20%
0%
400%
300%
200%
100%
t8
t9
t10
t11
t12
t13