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

特許第6468053号
(P6468053)

(45)発行日　平成31年2月13日(2019.2.13)　(24)登録日　平成31年1月25日(2019.1.25)

(51)Int.Cl.		F I		
G06F 9/52	(2006.01)	G06F	9/52	12OB
G06F 12/00	(2006.01)	G06F	12/00	572A

請求項の数 5　(全 26 頁)

(21)出願番号　特願2015-91361(P2015-91361)
(22)出願日　平成27年4月28日(2015.4.28)
(65)公開番号　特開2016-207130(P2016-207130A)
(43)公開日　平成28年12月8日(2016.12.8)
審査請求日　平成30年2月6日(2018.2.6)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100094525
弁理士　土井　健二
(74)代理人　100094514
弁理士　林　恒徳
(72)発明者　田▲邨▼　優人
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　中島　耕太
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

最終頁に続く

(54)【発明の名称】情報処理装置、並列処理プログラム、及び、共有メモリアクセス方法

(57)【特許請求の範囲】
【請求項１】
共有メモリ領域を有する記憶部と、
１つまたは複数のスレッドを実行する処理部と、を有し、
前記処理部は、
あるスレッドが前記共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行し、
前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行する、
情報処理装置。
【請求項２】
請求項１において、
前記処理部は、前記複数のスレッドを実行中ではない場合、新たなスレッドの実行を開始して前記複数のスレッドを実行中の場合に遷移したとき、前記第１の制御に基づく前記

アクセス処理中は、前記新たなスレッドによる前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の開始を待機する、
情報処理装置。
【請求項３】
請求項１または２において、
前記第２の制御は、前記一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する前記他のスレッドによる書き込みが発生しない場合に、前記アクセス処理を完了させる、
情報処理装置。
【請求項４】
あるスレッドが共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行し、
前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行する、
処理をコンピュータに実行させる並列処理プログラム。
【請求項５】
処理部が、あるスレッドが共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
処理部が、前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行し、
処理部が、前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記あるスレッドによる当該共有メモリ領域へのアクセス処理を実行する、
共有メモリアクセス方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、情報処理装置、並列処理プログラム、及び、共有メモリアクセス方法に関する。
【背景技術】
【０００２】
並列処理を行う情報処理装置は、複数のスレッドがアクセスする共有メモリ領域のデータの整合性を保つために、排他制御の機能を備える。
【０００３】
排他制御の方式として、１つのスレッドが共有メモリへのアクセス処理中は、他のプロセッサが、共有メモリへのアクセス処理の開始を待機する方式（以下、ロック方式と称する）がある。各スレッドは、例えば、共有メモリ領域の排他状態を示す変数を参照して、共有メモリ領域にアクセス可能であるか否かを判定する。
【０００４】
一方、情報処理装置のプロセッサが備える、ハードウェア・トランザクション・メモリ（Hardware Transactional Memory：ＨＴＭ）を使用する排他制御の方式（ＨＴＭ方式と

称する）がある。ＨＴＭの機構は、ユーザが指定した命令列（以下、対象ルーチン）が、他のスレッドが実行する処理に対して、アトミックなトランザクションとして実行されることを保証する。ＨＴＭは、対象ルーチンの実行中に、他のスレッドとのメモリアクセスの競合が発生した場合に、対象ルーチンの実行をロールバックする。ＨＴＭに関する技術は、例えば、特許文献１～３に記載される。

【０００５】

ユーザは、プログラムの生成時に、ロック方式とＨＴＭ方式とから、プログラムに採用する排他制御の方式を選択する。

【先行技術文献】

【特許文献】

【０００６】

【特許文献１】特表２０１３－５１３８８８号公報

【特許文献２】特表２０１３－５２０７５３号公報

【特許文献３】特開２０１２－１２８６２８号公報

【発明の概要】

【発明が解決しようとする課題】

【０００７】

しかしながら、ＨＴＭ方式の排他制御に基づくプログラムの処理時間は、共有メモリにアクセスするスレッドの数が１つの場合、ロック方式の排他制御に基づくプログラムに対して遅くなる場合がある。実行中のスレッドの数は、プログラムの処理に応じて、変化する。したがって、プログラムの生成時に、プログラムに採用する排他制御の方式を、適切に選択することは容易ではない。

【０００８】

１つの側面は、本発明は、共有メモリの排他制御の性能を向上する情報処理装置、並列処理プログラム、及び、共有メモリアクセス方法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】

【０００９】

第１の側面によれば、共有メモリ領域を有する記憶部と、１つまたは複数のスレッドを実行する処理部と、を有し、前記処理部は、前記スレッドが前記共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、

前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行し、前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行する。

【発明の効果】

【００１０】

第１の側面によれば、共有メモリの排他制御が高速化し、性能が向上する。

【図面の簡単な説明】

【００１１】

【図１】ロック方式の排他制御を説明する図である。

【図２】コンフリクトが発生しない場合の、ＨＴＭ方式の排他制御を説明する図である。

【図３】コンフリクトが発生する場合における、ＨＴＭ方式に基づく排他制御を説明する図である。

【図４】同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッド数が２つである場合のメモリアクセス処理の性能を示す図である。

【図５】同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッド数が１つである場

合のメモリアクセス処理の性能を示す図である。
【図６】プログラムの実行時の、スレッド数の変化を模式的に説明する図である。
【図７】本実施の形態における情報処理装置の処理の概要を説明する図である。
【図８】本実施の形態における情報処理装置１００のハードウェア構成図である。
【図９】図８に示した情報処理装置１００のソフトウェアブロック図である。
【図１０】図９に示した同時走行スレッド数記憶領域１７０に記憶する、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数の取得処理を説明する図である。
【図１１】本実施の形態における情報処理装置１００の排他制御プログラム１３３の処理の流れを説明するフローチャート図である。
【図１２】排他制御方式の切り替わりを模式的に説明する図である。
【図１３】同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッド数が２つである場合の、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能を示す図である。
【図１４】同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッド数が１つである場合の、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能を示す図である。
【図１５】図８に示したアプリケーションプログラム１３２の一部のプログラムｐｒ１の一例を示す図である。
【図１６】図９、図１１に示した排他取得モジュール１４１のプログラムｐｒ２の一例を示す図である。
【図１７】図９、図１１に示した排他解除モジュール１５１のプログラムｐｒ３の一例を示す図である。
【図１８】ＨＴＭ方式の排他取得モジュール１４２、及び、ＨＴＭ方式の排他解除モジュール１５２の処理の流れを説明するフローチャート図である。
【図１９】ロック方式の排他取得モジュール１４３、及び、ロック方式の排他解除モジュール１５３の処理の流れを説明するフローチャート図である。
【発明を実施するための形態】
【００１２】
以下、図面にしたがって本発明の実施の形態について説明する。ただし、本発明の技術的範囲はこれらの実施の形態に限定されず、特許請求の範囲に記載された事項とその均等物まで及ぶものである。
【００１３】
並列処理を行う情報処理装置において、複数のスレッドが、共有資源に対して同時にアクセスした場合、共有資源の不整合が発生する場合がある。排他制御は、複数のスレッドが、同時に共有資源にアクセスすることを抑制する制御を示す。排他制御を行うことにより、共有資源の不整合が発生することを回避可能になる。
【００１４】
スレッドは、オペレーションシステム上で動作するプログラムの最小の実行単位を示す。本実施の形態における情報処理装置は、複数のスレッドを同時に実行するマルチスレッド処理を実現する情報処理装置である。本実施の形態における共有資源は、複数のスレッドがアクセス可能な共有メモリの領域であって、共有メモリが有する一部または全部の領域である。
【００１５】
初めに、図１～図３にしたがって、排他制御を実現する複数の方式を説明する。図１は、ロック方式の排他制御を、図２、図３は、ハードウェア・トランザクション・メモリ（Hardware Transactional Memory：ＨＴＭ）方式の排他制御を説明する。
【００１６】
［ロック方式］
図１は、ロック方式の排他制御を説明する図である。図１は、２つのスレッド（スレッドｔｈＡ、スレッドｔｈＢ）を例示する。また、図１に示す矢印は時間の遷移を示す。ス

レッドｔｈＡ及びスレッドｔｈＢ（以下、スレッドｔｈともいう）は、共有メモリの同一の領域（共有メモリ領域）にアクセスする。

【００１７】

また、図１に示す、クリティカルセクション（Critical section）は、同一の共有メモリ領域に対するアクセス命令を含む、一連の命令列の処理（以下、アクセス処理ともいう）を実行するセクションを示す。アクセス処理は、同一の共有メモリ領域に対するデータの書き込み処理、または、同一の共有メモリ領域からのデータの読み出し処理のいずれかまたは両方を含む。

【００１８】

ロック方式は、一のスレッドによる共有メモリ領域へのアクセス処理中に、他のスレッドによる共有メモリ領域へのアクセス処理の開始を待機することによって排他制御を実現する方式である。ロック方式は、例えば、スピンロック方式、ミューテックス（Mutex）、及び、セマフォ（Semaphore）等に基づくロック方式である。本実施の形態は、メモリ上のロック変数に基づくスピンロック方式を使用する場合を例示する。

【００１９】

ロック方式によると、各スレッドｔｈは、同一の共有メモリ領域に対するアクセス処理、即ち、クリティカルセクションの開始時に、ロックを取得する。メモリ上の変数を示すロック変数が非ロック状態を示す場合、ロックを取得可能である。したがって、各スレッドｔｈは、ロック変数の値を、非ロック状態からロック状態に変更してロックを取得する。

【００２０】

一方、各スレッドｔｈは、ロック変数がロック状態を示す場合には、ロックを取得できない。ロック変数がロック状態を示す場合、他のスレッドによって、ロック変数がロック状態に更新されている状態を示し、他のスレッドによってロックが取得中であることを意味する。したがって、各スレッドｔｈは、他のスレッドによってロック変数が非ロック状態に更新され、ロックが解除されるまでロックの取得を待機する。

【００２１】

各スレッドｔｈは、ロックを取得すると、クリティカルセクションを開始する。そして、各スレッドｔｈは、クリティカルセクションを終了すると、ロック変数をロック状態から非ロック状態に更新し、ロックを解除する。

【００２２】

図１によると、スレッドｔｈＡは、タイミングｔ１にロックを取得後、クリティカルセクションを開始する。そして、スレッドｔｈＡは、クリティカルセクションを終了すると、タイミングｔ２に、ロックを開放する。

【００２３】

一方、スレッドｔｈＢは、スレッドｔｈＡによるクリティカルセクション開始後のタイミングｔ３に、ロックを取得しようとする。ただし、既に、スレッドｔｈＡがロックを取得中であるため、スレッドｔｈＢは、スレッドｔｈＡによるロックの解除を待機する。そして、タイミングｔ２に、スレッドｔｈＡがロックを解除すると、スレッドｔｈＢはロックを取得し、クリティカルセクションを開始する。スレッドｔｈＢは、クリティカルセクションを終了すると、ロックを解除する。

【００２４】

図１に示すように、ロック方式によると、スレッドｔｈＡがロックを取得している間、スレッドｔｈＢは、ロックの取得を待機する。即ち、スレッドｔｈＡが、クリティカルセクションを終了するまで、スレッドｔｈＢはクリティカルセクションを開始できない。これにより、情報処理装置は、複数のスレッドが同時に共有メモリ領域にアクセスすることを回避でき、共有メモリ領域のデータの不整合が発生することを回避できる。

【００２５】

なお、スレッドｔｈＡ、及び、スレッドｔｈＢは、同一のプログラムの実行に基づいて生成されるスレッドであってもよいし、異なるプログラムの実行に基づいてそれぞれ生成

されるスレッドであってもよい。また、スレッドｔｈＡのクリティカルセクションの処理と、スレッドｔｈＢのクリティカルセクションの処理は、同一の処理であってもよいし、異なる処理であってもよい。

【００２６】

次に、図２、図３にしたがって、ＨＴＭ方式の排他制御を説明する。

【００２７】

［ＨＴＭ方式］

ＨＴＭ方式は、情報処理装置のＣＰＵ（Central Processing Unit：ＣＰＵ）が搭載する、ハードウェアのＨＴＭの機構を使用する方式である。ＨＴＭ方式は、一のスレッドによる共有メモリ領域へのアクセス処理中に、共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、アクセス処理を取り消すことによって排他制御を実現する方式である。

【００２８】

ＨＴＭは、並列プログラミングをサポートするための機構である。ＨＴＭは、並列プログラミングの実行時の排他による衝突を低減し、性能を向上する。例えば、Sun Microsystems（登録商標）のRock、IBM（登録商標）の、Blue Gene/Q Compute chip、Intel（登録商標）のHaswellマイクロアーキテクチャのCore i7等のＣＰＵは、ＨＴＭの機構を搭載する。

【００２９】

ＨＴＭは、ユーザが指定した命令列を、単一のアトミックなトランザクション（atomic and isolated transaction）として実行する。ＨＴＭは、アトミックなトランザクションとして指定された命令列（以下、対象ルーチン）が実行する処理が、他のスレッドが並列して実行する他の処理に対して、単一のトランザクションとして実行されることを保証する。ユーザは、プログラムの生成時に、アトミックなトランザクションとして処理させる対象ルーチンの前後に、ＨＴＭの開始命令、及び、終了命令を付加する。

【００３０】

ＨＴＭは、開始命令から終了命令までの間に、対象ルーチンがアクセス処理の対象とするメモリのアドレスに、他のスレッドが書き込み処理を行った場合に、コンフリクト（メモリアクセスの競合）を検出する。コンフリクトを検出すると、ＨＴＭは、対象ルーチンをアボート（中断、abort）し、対象ルーチンをロールバック（rollback）する。一方、ＨＴＭは、コンフリクトを検出しない場合は、対象ルーチンを続行し、完了させる。このように、ＨＴＭ方式によると、各スレッドｔｈは、投機的に、対象ルーチンを実行する。

【００３１】

具体的に、ＨＴＭは、開始命令の実行に応答して、前処理を行う。前処理は、プロセッサコアの内部状態（レジスタ情報）の記憶（退避）処理や、対象ルーチンがアクセス処理（読み出し、書き込み）の対象とするメモリ領域のデータの読み出し、及び、読み出したデータの一時領域への記憶処理等を示す。

【００３２】

そして、ＨＴＭ方式によると、スレッドｔｈは、対象ルーチンによる書き込み処理を、前処理で記憶した一時領域（例えば、Ｌ１（level１）キャッシュ）に対して行う。つまり、スレッドｔｈは、ＨＴＭの終了命令の実行時まで、対象ルーチンの処理結果の、メモリへの反映を保留する。また、ＨＴＭは、開始命令から終了命令までの間に、一時領域に記憶した、対象ルーチンがアクセス処理の対象とするメモリのアドレスに、他のスレッドが書き込みした場合に、コンフリクトを検出する。

【００３３】

コンフリクトを検出すると、ＨＴＭは、トランザクションをアボート（中断）する。具体的に、ＨＴＭは、対象ルーチンの処理を中断し、ＥＡＸレジスタを除く、ＣＰＵの内部状態（レジスタ情報）を開始命令の実行時の状態に戻す（ロールバック）。また、ＨＴＭは、一時領域に記憶した、書き込み処理の結果データを破棄する。ＥＡＸレジスタは、アボートの理由を示す情報を保持する。そして、ＨＴＭは、プログラムの実行を、開始命令

に指定されたアボートルーチンに遷移させる。アボートルーチンは、例えば、ＥＡＸレジスタの値に基づいて、対象ルーチンの再実行の指示等を行う。

【００３４】

一方、開始命令から終了命令までの間にコンフリクトを検出しなかった場合、ＨＴＭは、対象ルーチンの終了命令の実行時に、後処理を行う。後処理は、一時領域に保持した書き込み処理の結果データをメモリに書き込む処理等を示す。

【００３５】

図２、図３は、ＨＴＭ方式に基づく排他制御を説明する図である。本実施の形態において、ＨＴＭの対象ルーチンは、共有メモリ領域にアクセスする処理（クリティカルセクション）を示す。ユーザは、プログラムの生成時に、クリティカルセクションの前後に、ＨＴＭの開始命令及び終了命令を付加する。

【００３６】

図２は、コンフリクトが発生しない場合の、ＨＴＭ方式の排他制御を説明する図である。図２に示す矢印は時間の遷移を示す。コンフリクトが発生しない場合、即ち、一のスレッドｔｈによる共有メモリ領域へのアクセス処理中に、共有メモリ領域に対する他のスレッドｔｈによる書き込みが発生しない場合、ＨＴＭは、一のスレッドｔｈのアクセス処理を完了させる。

【００３７】

スレッドｔｈＡは、タイミングｔ１に、ＨＴＭの開始命令を実行し、クリティカルセクションを開始する。前述したとおり、クリティカルセクションの実行時、スレッドｔｈＡは、開始命令の実行時に共有メモリ領域から読み出し、一時領域（ローカルエリア）に記憶した、アクセス対象のデータに対して、クリティカルセクションの処理を実行する。したがって、スレッドｔｈＡは、クリティカルセクションの実行中に、共有メモリ領域を直接、更新しない。

【００３８】

一方、スレッドｔｈＢは、スレッドｔｈＡによる開始命令の実行後のタイミングｔ３に、開始命令を実行する。スレッドｔｈＢも、スレッドｔｈＡと同様にして、開始命令の実行時に共有メモリ領域から読み出し一時領域に記憶したデータに対して、クリティカルセクションの処理を実行する。

【００３９】

図２の例では、スレッドｔｈＢのクリティカルセクションがアクセス処理の対象とする共有メモリ領域は、スレッドｔｈＡのクリティカルセクションがアクセス処理の対象とする共有メモリ領域と異なる。即ち、スレッドｔｈＢによるクリティカルセクション中に、スレッドｔｈＢがアクセス処理の対象とする共有メモリ領域に、スレッドｔｈＡによる書き込みが発生しない場合を示す。

【００４０】

したがって、ＨＴＭは、タイミングｔ２に示す、スレッドｔｈＡの終了命令の実行時に（スレッドｔｈＡによる結果データの共有メモリ領域への書き込み時に）、コンフリクトを検出しない。したがって、ＨＴＭは、スレッドｔｈＢのクリティカルセクションの処理をアボートしない。また、ＨＴＭは、スレッドｔｈＡのクリティカルセクションの処理を確定（完了）させる。

【００４１】

そして、スレッドｔｈＢがクリティカルセクションを終了すると、スレッドｔｈＢは、タイミングｔ４に、ＨＴＭの終了命令を実行する。ＨＴＭは、スレッドｔｈＢのクリティカルセクションの処理を更新した結果データを、共有メモリ領域に書き込む。

【００４２】

図２に示すように、各スレッドｔｈによる共有メモリ領域へのアクセス処理中に、共有メモリ領域に対する他のスレッドｔｈによる書き込みが発生しない場合、複数のスレッドｔｈＡ、ｔｈＢのクリティカルセクションが並列に実行可能になる。即ち、ＨＴＭ方式によると、コンフリクトが発生しない場合、スレッドｔｈＡ、ｔｈＢが並列に実行可能にな

る。
【００４３】
図３は、コンフリクトが発生する場合における、ＨＴＭ方式に基づく排他制御を説明する図である。図３において、図２で示したものと同一のものは、同一の記号で示す。コンフリクトが発生する場合、即ち、一のスレッドｔｈによる共有メモリ領域へのアクセス処理中に、共有メモリ領域に対する他のスレッドｔｈによる書き込みが発生した場合、ＨＴＭは、アクセス処理を取り消す。
【００４４】
図３の例によると、スレッドｔｈＢのクリティカルセクションがアクセス処理の対象とする共有メモリ領域は、スレッドｔｈＡのクリティカルセクションがアクセス処理の対象とする共有メモリ領域と重複する。即ち、スレッドｔｈＢによるクリティカルセクション中に、スレッドｔｈＢがアクセス処理対象とする共有メモリ領域に、スレッドｔｈＡによる書き込みが発生する場合を示す。
【００４５】
したがって、ＨＴＭは、タイミングｔ２に示す、スレッドｔｈＡの終了命令の実行時に（スレッドｔｈＡによる結果データのメモリへの書き込み時に）、コンフリクトを検出し、スレッドｔｈＢのクリティカルセクションをアボートする。そして、ＨＴＭは、スレッドｔｈＢのクリティカルセクションの処理をロールバックする。つまり、ＨＴＭは、スレッドｔｈＢのクリティカルセクションの処理を取り消す。
【００４６】
また、スレッドｔｈＢは、コンフリクトが発生した場合、例えば、クリティカルセクションの処理を再実行する。スレッドｔｈＢは、同様にして、ＨＴＭの開始命令を実行し、クリティカルセクションを開始する。そして、コンフリクトが発生しない場合、スレッドｔｈＢは、クリティカルセクションを終了し、終了時にＨＴＭの終了命令を実行する。
【００４７】
このように、スレッドｔｈＢによる共有メモリ領域へのアクセス処理中に、共有メモリ領域に対するスレッドｔｈＡによる書き込みが発生した場合、ＨＴＭは、スレッドｔｈＢによる共有メモリ領域へのアクセス処理を取り消す。したがって、同一の共有メモリ領域に対して同時にメモリアクセス処理が発生することを回避可能になり、共有メモリ領域が記憶するデータの不整合を回避可能になる。
【００４８】
図２、図３に示すように、ＨＴＭは、メモリアクセスの競合（コンフリクト）を検出した場合にのみ、クリティカルセクションの処理をロールバックする。したがって、ＨＴＭ方式によると、メモリアクセスの競合が発生しない場合には、複数のスレッドｔｈによるクリティカルセクションを、並列に実行可能になる。これにより、共有メモリ領域へのアクセス処理を、効率的に実行可能になる。
【００４９】
［排他制御の方式による性能］
次に、図４、図５にしたがって、図１～図３で説明した、ロック方式とＨＴＭ方式の排他制御方式に基づくメモリアクセス処理の性能の相違を説明する。図４、図５は、同一の共有メモリ領域にアクセスする、実行中のスレッドｔｈの数に応じた性能を示す。図４、図５の例に示す性能は、共有メモリ領域へのアクセス処理を有するプログラムの処理時間に基づいて算出した性能を示す。
【００５０】
図４は、同一の共有メモリ領域にアクセスする、実行中のスレッド数が２つである場合のメモリアクセス処理の性能を示す図である。図４に示すグラフの横軸は、一度の排他制御に基づいて読み書きする対象データのサイズ（Byte）を示し、縦軸は、性能を正規化した値を示す。縦軸の値は、値「１」に近づくほど、プログラムの処理時間が短く抑えられ、性能が高いことを示す。
【００５１】

図４は、ロック方式、及び、ＨＴＭ方式の排他制御方式に基づくメモリアクセス処理の性能を示す。グラフに示す図形（丸、四角、三角、ひし形）のそれぞれは、テストパターンに対応する。また、白色で示す各図形はロック方式の排他制御に基づくメモリアクセス処理の性能を示し、黒色で示す図形はＨＴＭ方式の排他制御に基づくメモリアクセス処理の性能を示す。

【００５２】

図４のグラフによると、読み書き対象のデータのサイズが、64Byteから4096Byteまで間、ＨＴＭ方式の排他制御に基づくプログラムは、ロック方式の排他制御に基づくプログラムに対して、性能が高い。

【００５３】

図２、図３で説明したとおり、ＨＴＭは、対象ルーチン（クリティカルセクション）を投機的に実行する。したがって、ＨＴＭ方式によると、情報処理装置は、メモリアクセスの競合が発生しない場合、複数のスレッドｔｈによる共有メモリ領域へのメモリアクセス処理を並列に実行できる。これに対し、ロック方式によると、情報処理装置は、メモリアクセス処理を並列に実行できない。したがって、実行中のスレッド数が２つの場合、ＨＴＭ方式の排他制御に基づくプログラムは、ロック方式の排他制御に基づくプログラムに対して、性能が高い。

【００５４】

なお、読み書き対象のデータのサイズが、4096Byteを超える場合、各方式の排他制御に基づくプログラムの性能はほぼ同じである。図２、図３で前述したとおり、ＨＴＭは開始命令の実行時に、前処理を行う。前処理は、アクセス対象のデータを共有メモリ領域から読み出して一時領域に記憶する処理を含む。したがって、図４の例のテストパターンによると、読み書き対象のデータサイズが所定の値を超える場合、前処理の負荷が高くなり、ＨＴＭ方式の排他制御に基づくプログラムの性能が、ロック方式の排他制御に基づくプログラムの性能と同等になる。

【００５５】

図５は、同一の共有メモリ領域にアクセスする、実行中のスレッド数が１つである場合のメモリアクセス処理の性能を示す図である。図５に示すグラフの横軸及び縦軸、及び、図形は、図４と同様である。図４で説明したとおり、白色で示す各図形はロック方式の排他制御に基づくメモリアクセス処理の性能を示し、黒色で示す図形はＨＴＭ方式の排他制御に基づくメモリアクセス処理の性能を示す。

【００５６】

図５のグラフによると、読み書き対象のデータのサイズが、ＨＴＭ方式の排他制御に基づくプログラムは、ロック方式の排他制御に基づくプログラムに対して、性能が低い。したがって、図４の同一の共有メモリ領域にアクセスする実行中のスレッド数が２つの場合と異なり、スレッド数が１つの場合は、ＨＴＭ方式の排他制御に基づくプログラムより、ロック方式の排他制御に基づくプログラムの方が、性能が高い。

【００５７】

図２、図３で前述したとおり、ＨＴＭ方式によると、ＨＴＭは、前処理及び後処理を行う。これに対し、ロック方式は、前処理及び後処理を行わないため、オーバヘッドが小さい。したがって、同一の共有メモリ領域にアクセスする実行中のスレッドｔｈの数が１つのみである場合、オーバヘッドが小さいロック方式の排他制御方式に基づくプログラムは、ＨＴＭ方式の排他制御に基づくプログラムより性能が高い。

【００５８】

図４、図５に示すように、ＨＴＭ方式とロック方式の間で、同一の共有メモリ領域にアクセスする実行中のスレッドｔｈの数に応じて、より性能が高い排他制御の方式が異なる。つまり、同一の共有メモリ領域にアクセスする実行中のスレッド数が複数である場合はＨＴＭ方式の性能がより高いのに対し、単数である場合はロック方式の性能がより高い。

【００５９】

図６は、プログラムの実行時の、スレッド数の変化を模式的に説明する図である。プロ

グラム実行時の、実行（走行）中のスレッドｔｈの数は、一定ではない。実行中のスレッドｔｈの数は、プログラムが実行する処理の変化に応じて、時々刻々と変化する。したがって、プログラムが実行する処理の変化に応じて、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数も変化する。

【００６０】

図６に示すように、ある時間帯は、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッドｔｈ（ｔｈ１～ｔｈｎ）の数が２つ以上であるのに対し、別の時間帯は、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッドｔｈ１の数が１つに遷移する。このように、プログラムの処理に応じて、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数は変化する。したがって、予め、プログラムの生成時に、ロック方式とＨＴＭ方式とから、適切な排他制御の方式を選択することは容易ではない。

【００６１】

［本実施の形態の概要］

したがって、本実施の形態における情報処理装置は、スレッドｔｈが共有メモリ領域Ｓｍにアクセスする際に、当該共有メモリ領域Ｓｍにアクセスする、複数のスレッドｔｈを実行中か否かを判定する。そして、情報処理装置は、複数のスレッドｔｈを実行中ではない場合は、第１の方式（ロック方式）に基づいて共有メモリ領域Ｓｍへのアクセス処理を実行する。また、情報処理装置は、複数のスレッドｔｈが実行中の場合は、第２の制御（ＨＴＭ方式）に基づいて、共有メモリ領域Ｓｍへのアクセス処理を実行する。

【００６２】

図１で前述したとおり、ロック方式によると、情報処理装置は、一のスレッドｔｈによる共有メモリ領域Ｓｍへのアクセス処理中に、他のスレッドｔｈによる共有メモリ領域Ｓｍへのアクセス処理の開始を待機する。また、図２、図３で前述したとおり、ＨＴＭ方式によると、情報処理装置は、一のスレッドｔｈによる共有メモリ領域Ｓｍへのアクセス処理中に、共有メモリ領域Ｓｍに対する他のスレッドｔｈによる書き込みが発生した場合に、アクセス処理を取り消す。

【００６３】

図７は、本実施の形態における情報処理装置の処理の概要を説明する図である。図７において、図６で示したものと同一のものは、同一の記号で示す。

【００６４】

図７に示すように、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッドｔｈの数が複数ではない場合、即ち、１つの場合、情報処理装置はロック方式を選択し、複数の場合はＨＴＭ方式を選択する。つまり、情報処理装置は、プログラムの実行中に、同一の共有メモリ領域Ｓｍにアクセスする実行中（走行中）のスレッドｔｈの数の変化に応じて、排他制御の方式を切り替える。

【００６５】

したがって、情報処理装置は、プログラムの実行中に、同一の共有メモリ領域Ｓｍにアクセスするスレッドｔｈの走行状態に基づいて、より高い性能の排他制御の方式を選択し、切り替えることができる。したがって、情報処理装置は、共有メモリ領域Ｓｍの整合性を維持しながら、各スレッドｔｈによる共有メモリ領域Ｓｍへのアクセス処理を効率的に実行することができる。つまり、情報処理装置は、共有メモリ領域Ｓｍのアクセス処理の排他制御の性能を向上できる。

【００６６】

［情報処理装置のハードウェア構成］

図８は、本実施の形態における情報処理装置１００のハードウェア構成図である。図８に示す情報処理装置１００は、例えば、ＣＰＵ１０１、メモリ１０２、通信インタフェース部１０３を有する。各部は、バス１０６を介して相互に接続する。メモリ１０２は、ＲＡＭ（Random Access Memory：ＲＡＭ）１２０や不揮発性メモリ１２１等を備える。

【００６７】

ＣＰＵ１０１は、バス１０６を介してメモリ１０２等と接続するとともに、情報処理装

置１００の全体の制御を行う。また、図８に示すＣＰＵ１０１は、図示していないが、複数のプロセッサコアを有し、マルチスレッド処理を実現する。また、図８に示すＣＰＵ１０１は、図２、図３で説明したＨＴＭ２００の機構を備える。また、通信インタフェース部１０３は、他の装置（図示せず）と通信して、データの送受信等を行う。

【００６８】

メモリ１０２のＲＡＭ１２０は、ＣＰＵ１０１が処理を行うデータ等を記憶する。また、例えば、ＲＡＭ１２０は、共有メモリ領域Ｓｍを有する。ただし、この例に限定されるものではなく、不揮発性メモリ１２１が、共有メモリ領域Ｓｍを有していてもよい。

【００６９】

メモリ１０２の不揮発性メモリ１２１は、オペレーションシステム格納領域１３１、アプリケーションプログラム格納領域１３２を備える。不揮発性メモリ１２１は、例えば、不揮発性半導体メモリ等を示す。

【００７０】

オペレーションシステム格納領域１３１のオペレーションシステム（以下、オペレーションシステム１３１）は、ＣＰＵ１０１の実行によって、情報処理装置１００で動作するオペレーションシステムの処理を実現する。また、オペレーションシステム格納領域１３１は、排他制御プログラム格納領域１３３を有する。排他制御プログラム格納領域１３３の排他制御プログラム（以下、排他制御プログラム１３３）は、共有メモリ領域Ｓｍの排他制御処理を実現する。排他制御プログラム１３３の処理は、図９にしたがって後述する。

【００７１】

アプリケーションプログラム格納領域１３２のアプリケーションプログラム（以下、アプリケーションプログラム１３２）は、ＣＰＵ１０１の実行によって、オペレーションシステム１３１上で動作し、所定の処理を実現する。また、アプリケーションプログラム１３２は、共有メモリ領域Ｓｍにアクセスする際に、排他制御プログラム１３３を呼び出す。

【００７２】

［情報処理装置１００のソフトウェアブロック］

図９は、図８に示した情報処理装置１００のソフトウェアブロック図である。図８に示した排他制御プログラム１３３は、排他取得モジュール１４１、排他解除モジュール１５１を有する。各モジュールの処理の詳細は、図１１のフローチャート図にしたがって後述する。

【００７３】

排他取得モジュール１４１は、ＨＴＭ方式の排他取得モジュール１４２と、ロック方式の排他取得モジュール１４３とを有する。また、排他解除モジュール１５１は、ＨＴＭ方式の排他解除モジュール１５２と、ロック方式の排他解除モジュール１５３とを有する。

【００７４】

排他取得モジュール１４１は、ＲＡＭ１２０等のメモリが有する同時走行スレッド数記憶領域１７０を参照し、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数を取得する。そして、排他取得モジュール１４１は、取得したスレッド数に基づいて、ＨＴＭ方式の排他取得モジュール１４２、または、ロック方式の排他取得モジュール１４３のいずれかを呼び出す。

【００７５】

ＨＴＭ方式の排他取得モジュール１４２は、ＨＴＭ方式に基づく排他制御の開始処理を行う。具体的に、ＨＴＭ方式の排他取得モジュール１４２は、ＨＴＭ２００（図８）が処理対象とするトランザクション（対象ルーチン）の開始を、ＨＴＭ２００に通知する、開始命令を呼び出す。

【００７６】

ロック方式の排他取得モジュール１４３は、ＲＡＭ１２０等のメモリ上のロック変数１６０に基づいて、ロック方式に基づく排他制御の開始（取得）処理を行う。具体的に、ロ

ック方式の排他取得モジュール１４３は、ロック変数１６０が非ロック状態に遷移するまで、クリティカルセクションの開始を待機する。また、ロック方式の排他取得モジュール１４３は、ロック変数１６０が非ロック状態に遷移すると、ロック変数１６０をロック状態に更新する。

【００７７】

排他解除モジュール１５１は、排他取得モジュール１４１と同様に、同時走行スレッド数記憶領域１７０を参照し、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数を取得する。そして、排他解除モジュール１５１は、取得したスレッド数に基づいて、ＨＴＭ方式の排他解除モジュール１５２、または、ロック方式の排他解除モジュール１５３のいずれかを呼び出す。

【００７８】

ＨＴＭ方式の排他解除モジュール１５２は、ＨＴＭ方式に基づく排他制御の終了処理を行う。具体的に、ＨＴＭ方式の排他解除モジュール１５２は、ＨＴＭ２００が処理対象とするトランザクションの終了をＨＴＭ２００に通知する、終了命令を呼び出す。また、ロック方式の排他解除モジュール１５３は、ロック方式に基づく排他制御の終了（解除）処理を行う。具体的に、ロック方式の排他解除モジュール１５３は、ロック変数１６０を非ロック状態に更新する。

【００７９】

［スレッド数］

図１０は、図９に示した同時走行スレッド数記憶領域１７０に記憶する、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数の取得処理を説明する図である。

【００８０】

並列処理を行う情報処理装置１００は、例えば、スレッドスケジューラ１８０を実行する。スレッドスケジューラ１８０は、例えば、スレッドｔｈのスケジューリングを行うオペレーションシステム１３１のプロセスである。スレッドスケジューラ１８０は、実行開始すべきスレッドを選択し、ＣＰＵ１０１（図８）のプロセッサコア（図示せず）に割り当てる。また、スレッドスケジューラ１８０は、同一の共有メモリ領域にアクセスする実行中のスレッド数（同時走行スレッド数ともいう、numThreads）を取得し、同時走行スレッド数記憶領域１７０に記憶する。

【００８１】

各スレッドｔｈは、例えば、同時走行スレッド数記憶領域１７０を参照し、同一の共有メモリ領域Ｓｍにアクセスする実行中スレッド数を取得する（図１０のｐ１）。そして、スレッドｔｈは、取得したスレッド数に基づいて選択した排他制御の方式に基づいて、共有メモリ領域Ｓｍにアクセスする（ｐ２）。

【００８２】

なお、スレッドｔｈが、同一の共有メモリ領域Ｓｍにアクセスする実行中スレッド数を取得する方法は、図１０の例に限定されるものではない。例えば、情報処理装置１００のオペレーションシステム１３１が、同一の共有メモリ領域Ｓｍにアクセスする実行中スレッド数を管理してもよい。この場合、スレッドｔｈは、オペレーションシステム１３１のシステムコールを実行することによって、同一の共有メモリ領域Ｓｍにアクセスする実行中スレッド数を取得する。

【００８３】

次に、図１１にしたがって、図８、図９で説明した排他制御プログラム１３３の処理の流れを説明する。

【００８４】

［排他制御プログラム１３３の処理］

図１１は、本実施の形態における情報処理装置１００の排他制御プログラム１３３の処理の流れを説明するフローチャート図である。

【００８５】

Ｓ１１：アプリケーションプログラム１３２は、クリティカルセクションの実行開始前

に、排他制御プログラム１３３の排他取得モジュール１４１を呼び出す。

【００８６】

Ｓ１２：排他取得モジュール１４１は、図１０で説明した同時走行スレッド数記憶領域１７０を参照し、同一の共有メモリ領域Ｓｍにアクセスする、同時走行スレッド数が２個以上であるか否かを判定する。

【００８７】

Ｓ１３：同時走行スレッド数が２個以上である場合（Ｓ１２のＹｅｓ）、排他取得モジュール１４１は、ＨＴＭ方式の排他取得モジュール１４２を読み出す。ＨＴＭ方式の排他取得モジュール１４２は、ＨＴＭ方式の実行開始命令を実行して、ＨＴＭ方式の前処理を行う。工程Ｓ１３の処理の詳細は、図１８のフローチャート図で後述する。

【００８８】

Ｓ１４：一方、同時走行スレッド数が１個の場合（Ｓ１２のＮｏ）、排他取得モジュール１４１は、ロック方式の排他取得モジュール１４３を読み出す。ロック方式の排他取得モジュール１４３は、ロック変数１６０に基づいてロックを取得する。工程Ｓ１４の処理の詳細は、図１９のフローチャート図で後述する。

【００８９】

Ｓ１５：排他取得処理（工程Ｓ１３、または、工程Ｓ１４）が終了すると、排他取得モジュール１４１は、アプリケーションプログラム１３２に制御を戻す。そして、スレッドは、アプリケーションプログラム１３２の処理である、共有メモリ領域Ｓｍへのアクセス処理（クリティカルセクション）を実行する。

【００９０】

なお、ＨＴＭ方式の排他制御を選択した場合、クリティカルセクションの実行中に、ＨＴＭ２００がコンフリクト（メモリアクセスの競合）を検出すると、ＨＴＭ２００は、クリティカルセクションをアボートし、ロールバックする。例えば、スレッドｔｈがクリティカルセクションの処理を再実行する場合、スレッドｔｈは、再度、ＨＴＭ方式の実行開始命令を実行する（Ｓ１３）。

【００９１】

Ｓ１６：クリティカルセクションが終了すると、アプリケーションプログラム１３２は、排他制御プログラム１３３の排他解除モジュール１５１を呼び出す。

【００９２】

Ｓ１７：排他解除モジュール１５１は、排他取得処理（Ｓ１３、Ｓ１４）が、ＨＴＭ方式またはロック方式のいずれの方式に基づくかを判定する。

【００９３】

Ｓ１８：排他取得処理がＨＴＭ方式に基づく場合（ＨＴＭ方式）、排他解除モジュール１５１は、ＨＴＭ方式の排他解除モジュール１５２を読み出す。ＨＴＭ方式の排他解除モジュール１５２は、ＨＴＭ方式の実行終了命令を実行し、ＨＴＭ方式の後処理を行う。工程Ｓ１８の処理の詳細は、図１８のフローチャート図で後述する。

【００９４】

Ｓ１９：排他取得処理がロック方式に基づく場合（ロック方式）、排他解除モジュール１５１は、ロック方式の排他解除モジュール１５３を読み出す。ロック方式の排他解除モジュール１５３は、ロック変数１６０に基づいてロックを解除する。工程Ｓ１９の処理の詳細は、図１９のフローチャート図で後述する。

【００９５】

図１１に示すように、排他制御プログラム１３３は、排他取得モジュール１４１の方式と同様の方式にしたがって、排他解除モジュール１５１の処理を行う。したがって、排他制御プログラム１３３は、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数の数が遷移した場合であっても、排他取得時の排他制御方式に基づいて、適切に排他解除処理を行うことができる。

【００９６】

次に、図１１に示したフローチャート図にしたがって排他制御の方式を選択した場合の

、排他制御方式の切り替わりを説明する。

【００９７】

［排他制御の切り替わり］

図１２は、排他制御方式の切り替わりを模式的に説明する図である。図１２に示す矢印ｔｔは時間の遷移を示す。また、図１２に示す、点線の横線で示す矩形はロック方式の排他制御に基づくクリティカルセクションを、縦線で示す矩形はＨＴＭ方式の排他制御に基づくクリティカルセクションを示す。また、右上がりの斜線で示す矩形は、同時走行スレッド数記憶領域１７０（図１０）の値（同一の共有メモリ領域にアクセスするスレッドの同時走行数）の取得処理を示す。

【００９８】

図１２は、アプリケーションプログラム１３２（図８）が、スレッドｔｈＡ、ｔｈＢを実行する場合を例示する。また、図１２は、スレッドｔｈＡが走行を開始後に、スレッドｔｈＢが走行を開始する場合を例示する。スレッドｔｈＡ、ｔｈＢは、同一の共有メモリ領域Ｓｍにアクセスする。

【００９９】

アプリケーションプログラム１３２は、タイミングｔ１１に、スレッドｔｈＡの走行を開始する。スレッドｔｈＡの走行開始により、スレッドスケジューラ１８０は、同時走行スレッド数記憶領域１７０の値を「０」から「１」に更新する。

【０１００】

スレッドｔｈＡは、スレッドｔｈＢが走行を開始する前に、クリティカルセクションを開始する。スレッドｔｈＡは、排他取得モジュール１４１を呼び出し（図１１のＳ１１）、スレッドスケジューラ１８０が更新した同時走行スレッド数記憶領域１７０の値「１」に基づいてロック方式を選択する（Ｓ１２）。そして、スレッドｔｈＡは、ロック方式に基づいて排他を取得し（Ｓ１４）、クリティカルセクションを実行する（Ｓ１５）。

【０１０１】

一方、アプリケーションプログラム１３２は、スレッドｔｈＡの走行中に（タイミングｔ１２）、スレッドｔｈＢの走行を開始する。スレッドｔｈＢの走行開始により、スレッドスケジューラ１８０は、同時走行スレッド数記憶領域１７０の値を「１」から「２」に更新する。そして、スレッドｔｈＢは、クリティカルセクションの開始前（タイミングｔ１３）に、同時走行スレッド数記憶領域１７０の値「２」の情報に基づいて、ＨＴＭ方式を選択する（Ｓ１２）。

【０１０２】

ただし、タイミングｔ１３の時点では、既に、スレッドｔｈＡが、ロック方式に基づいて排他を取得中である。同一の共有メモリ領域Ｓｍに対して、異なる排他制御方式に基づいて排他制御を行っても、排他制御の機能は成立しない。即ち、同一の共有メモリ領域Ｓｍに対する排他制御方式は、同一の排他制御方式である必要性がある。したがって、スレッドｔｈＢは、スレッドｔｈＡが、ロック方式に基づいて排他を解除するまで（図１１のＳ１９）の間、ＨＴＭ方式に基づく排他取得処理を待機する。

【０１０３】

そして、タイミングｔ１４に、スレッドｔｈＡが、排他取得時に選択した方式（即ち、ロック方式）に基づいて排他の解除を行うと（図１１のＳ１９）、スレッドｔｈＢは、ＨＴＭ方式の排他取得処理を行う（Ｓ１２、Ｓ１３）。そして、スレッドｔｈＢは、クリティカルセクションを開始する（Ｓ１５）。クリティカルセクションの終了後、スレッドｔｈＢは、排他取得時に選択したＨＴＭ方式に基づいて、排他の解除処理を行う（Ｓ１８）。

【０１０４】

このように、複数のスレッドｔｈを実行中ではない場合、スレッドｔｈＡはロック方式を選択するが、ロック方式の排他取得中に、新たなスレッドｔｈＢが走行を開始し、同時走行スレッド数記憶領域１７０の値が「１」から「２」に遷移する場合がある。この場合、スレッドｔｈＢは、ロック方式の排他制御に基づく共有メモリ領域Ｓｍへのアクセス処

理中は、ＨＴＭ方式の排他制御に基づく共有メモリ領域Ｓｍへのアクセス処理（クリティカルセクション）の開始を待機する。

【０１０５】

即ち、情報処理装置１００は、複数のスレッドを実行中ではない場合に、新たなスレッドの実行を開始し複数のスレッドの実行中に遷移したとき、ロック方式に基づくアクセス処理中は、新たなスレッドによるＨＴＭ方式に基づくアクセス処理の開始を待機する。これにより、情報処理装置１００は、アクセス処理中に、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数が１個から複数個に増加した場合であっても、複数個のスレッドｔｈに共通の排他制御方式にしたがって、適切に排他制御を実現できる。

【０１０６】

タイミングｔ１４から、スレッドｔｈＡが走行を終了するタイミングｔ１５までの間、同時走行スレッド数記憶領域１７０は値「２」である。したがって、スレッドｔｈＡ、ｔｈＢは、ＨＴＭ方式の排他制御に基づいて、共有メモリ領域Ｓｍのアクセス処理（クリティカルセクション）を行う。

【０１０７】

また、スレッドｔｈＡのクリティカルセクションの終了命令の実行時に、スレッドｔｈＢのクリティカルセクションとの間でメモリアクセスの競合が発生した場合、ＨＴＭ２００は、スレッドｔｈＢのクリティカルセクションをアボートし、ロールバックする（ｘ１）。クリティカルセクションを再実行する場合、スレッドｔｈＢは、同時走行スレッド数記憶領域１７０の値に基づいて、ＨＴＭ方式に基づいて排他を取得し（Ｓ１３）、クリティカルセクションを実行する（Ｓ１５）。

【０１０８】

そして、タイミングｔ１５にスレッドｔｈＡが走行を停止（終了）すると、スレッドスケジューラ１８０は、同時走行スレッド数記憶領域１７０を値「２」から値「１」に更新する。なお、スレッドｔｈＢは、同時走行スレッド数記憶領域１７０が値「１」に更新された後であっても、クリティカルセクションの終了時（タイミングｔ１６）に、排他取得時に選択した方式（即ち、ＨＴＭ方式）に基づいて、排他解除処理を行う（Ｓ１８）。

【０１０９】

即ち、情報処理装置１００は、複数のスレッドを実行中の場合に、いずれかのスレッドの実行が終了して複数のスレッドを実行中ではない場合に遷移したとき、ＨＴＭ方式に基づくアクセス処理の終了時に、ＨＴＭ方式に基づく終了（排他解除）処理を行う。これにより、情報処理装置１００は、アクセス処理中に、同一の共有メモリ領域Ｓｍにアクセスする実行中のスレッド数が複数個から１個に減少した場合であっても、排他取得時の排他制御方式に基づいて、適切に、排他解除処理を行うことができる。

【０１１０】

そして、スレッドｔｈＢは、スレッドｔｈＡの停止後であるタイミングｔ１７に、クリティカルセクションを開始する。このとき、スレッドｔｈＢは、同時走行スレッド数記憶領域１７０の値「１」に基づいてロック方式を選択する（図１１のＳ１２）。したがって、スレッドｔｈＢは、ロック方式の排他制御に基づいて、共有メモリ領域Ｓｍのアクセス処理（クリティカルセクション）を行う。

【０１１１】

次に、図１３、図１４にしたがって、本実施の形態におけるメモリアクセス処理の性能を説明する。図１３、図１４は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数のパターンに応じた、本実施の形態における排他制御方式の性能を示す。

【０１１２】

［本実施の形態における排他制御方式の性能］

図１３は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数が２つである場合の、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能を示す図である。図１３は、図４に示した、ロック方式、及び、ＨＴＭ方式の排他制御方式に基づくメモリアクセス処理の性能に加えて、本実施の形態における排他制御方式に基づくメ

モリアクセス処理の性能を示す。

【０１１３】

図１３に示すグラフの横軸及び縦軸、及び、図形は、図４、図５と同様である。図１３の、右上がりの斜線で示す各図形は、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能を示す。

【０１１４】

本実施の形態における排他制御方式は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数が２つ以上の場合に、ＨＴＭ方式の排他制御方式を採用する。したがって、図１３に示すグラフによると、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能は、黒色の図形に示すＨＴＭ方式に基づくメモリアクセス処理の性能と同様である。

【０１１５】

図１４は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数が１つである場合の、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能を示す図である。図１４に示すグラフの横軸及び縦軸、及び、図形は、図１３と同様である。

【０１１６】

本実施の形態における排他制御方式は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッドｔｈの数が１つの場合は、ロック方式の排他制御方式を採用する。したがって、図１４に示すグラフによると、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能は、白色の図形に示すロック方式に基づくメモリアクセス処理の性能と同様である。

【０１１７】

図１３、図１４に示すように、本実施の形態の排他制御方式に基づくメモリアクセス処理の性能は、ロック方式とＨＴＭ方式のうち、実行中のスレッドｔｈの数に応じた、より性能が高い方式に基づくメモリアクセス処理と同様の性能となる。このように、情報処理装置１００は、プログラムの実行中に、スレッドｔｈの走行状態に基づいて性能をより向上する排他制御方式に切り替えることによって、メモリアクセス処理を効率的に実行し、排他制御の性能を向上できる。

【０１１８】

次に、図１５～図１７にしたがって、図８に示したアプリケーションプログラム１３２の一例と、図９に示した排他取得モジュール１４１、及び、排他解除モジュール１５１のプログラム例を説明する。

【０１１９】

［プログラムの例］

図１５は、図８に示したアプリケーションプログラム１３２の一部のプログラムｐｒ１の一例を示す図である。図１５に示す、記述ｃ１は排他取得モジュール１４１（図９）の呼出し命令を示し、記述ｃ２は、排他解除モジュール１５１（図９）の呼出し命令を示す。また、命令群ｃ３は、共有メモリ領域Ｓｍにアクセスする処理（クリティカルセクション）を実行する命令である。

【０１２０】

プログラムｐｒ１は、クリティカルセクション（ｃ３、図１１のＳ１５）の実行開始前に、記述ｃ１を実行する。これにより、プログラムｐｒ１は、本実施の形態の排他取得モジュール１４１を呼出し、排他を取得する（図１１のＳ１１）。また、プログラムｐｒ１は、クリティカルセクション（ｃ３、Ｓ１５）の終了後、記述ｃ２を実行する。これにより、プログラムｐｒ１は、本実施の形態の排他解除モジュール１５１を呼出し、排他を解除する（Ｓ１６）。

【０１２１】

図１６は、図９、図１１に示した排他取得モジュール１４１のプログラムｐｒ２の一例を示す図である。図１６に示す排他取得モジュール１４１は、図１５に示した記述ｃ１によって呼び出されるモジュールである。

【０１２２】
図１６に示す記述ｃ１１は、ロック変数「spinlock」１６０の宣言文を示す。また、記述ｃ１２は、同一の共有メモリ領域Ｓｍにアクセスする、実行中のスレッド数「numThreads」（図１０の同時走行スレッド数記憶領域１７０）の値が１より大きいか否かを判定する記述である（図１１のＳ１２）。
【０１２３】
記述ｃ１３は、実行中のスレッド数「numThreads」の値が１より大きい場合（図１１のＳ１２のＹｅｓ）の処理を示す。具体的に、記述ｃ１３は、排他制御の方式「access_form」をＨＴＭ方式に設定し、ＨＴＭ方式の排他取得モジュール１４２（rtm_wrapped_lock()）を呼び出す命令（Ｓ１３）を示す。
【０１２４】
記述ｃ１４は、実行中のスレッド数「numThreads」の値が１以下である場合の処理を示す（図１１のＳ１２のＮｏ）。具体的に、記述ｃ１４は、排他制御の方式「access_form」をロック方式に設定し、ロック方式の排他取得モジュール１４３（spin_lock()）を呼び出す命令（Ｓ１４）を示す。なお、図１６に図示していないが、ロック方式の排他取得モジュール１４３（spin_lock()）は、ロック変数「spinlock」１６０を参照する。
【０１２５】
図１７は、図９、図１１に示した排他解除モジュール１５１のプログラムｐｒ３の一例を示す図である。図１７の排他解除モジュール１５１は、図１５に示した記述ｃ２によって呼び出されるモジュールである。
【０１２６】
図１７に示す記述ｃ２１は、ロック変数「spinlock」１６０の宣言文を示す。また、記述ｃ２２は、排他取得モジュール１４１が設定した排他制御の方式「access_form」がＨＴＭ方式であるか否かを判定する（図１１のＳ１７）記述である。
【０１２７】
記述ｃ２３は、排他取得モジュール１４１が設定した排他制御の方式「access_form」がＨＴＭ方式である場合に（図１１のＳ１７のＨＴＭ方式）、ＨＴＭ方式の排他解除モジュール１５２（rtm_wrapped_unlock()）を呼び出す命令（Ｓ１８）を示す。また、記述ｃ２４は、排他取得モジュール１４１が設定した排他制御の方式「access_form」がロック方式である場合に（Ｓ１７のロック方式）、ロック方式の排他解除モジュール１５３（spin_unlock()）を呼び出す命令（Ｓ１９）を示す。なお、図１７に図示していないが、ロック方式の排他解除モジュール１５３（spin_unlock()）は、ロック変数「spinlock」１６０を参照する。
【０１２８】
次に、図１８にしたがってＨＴＭ方式の排他取得モジュール１４２、及び、ＨＴＭ方式の排他解除モジュール１５２の処理の流れを説明する。また、図１９にしたがってロック方式の排他取得モジュール１４３、及び、ロック方式の排他解除モジュール１５３の処理の流れを説明する。
【０１２９】
［ＨＴＭ方式の処理］
図１８は、ＨＴＭ方式の排他取得モジュール１４２、及び、ＨＴＭ方式の排他解除モジュール１５２の処理の流れを説明するフローチャート図である。
【０１３０】
図１８の（Ａ）は、ＨＴＭ方式の排他取得モジュール１４２の処理（図１１のＳ１３）の流れを示すフローチャート図である。
【０１３１】
Ｓ２１：ＨＴＭ方式の排他取得モジュール１４２は、ロック方式に基づくロックが解放されているか否かを判定する。図１２で説明したとおり、同一の共有メモリ領域Ｓｍに対する、異なる排他制御方式に基づく排他制御は有効ではない。したがって、排他を取得しようとするスレッドｔｈの、ＨＴＭ方式の排他取得モジュール１４２は、排他取得中のス

レッドｔｈがロック方式に基づいて排他を解除するまでの間、ＨＴＭ方式に基づく排他取得処理の実行を待機する。

【０１３２】

Ｓ２２：ロック方式に基づくロックが解放済みの場合、または、ロック方式に基づいて排他が解除された場合（Ｓ２１のＹｅｓ）、排他取得モジュール１４１は、ＨＴＭ２００の開始命令を実行し、ＨＴＭ方式の前処理を行う。ＨＴＭ方式の前処理は、図２、図３で前述したとおりである。

【０１３３】

図１８の（Ｂ）は、ＨＴＭ方式の排他解除モジュール１５２の処理の流れを示すフローチャート図である。

【０１３４】

Ｓ３１：ＨＴＭ方式の排他解除モジュール１５２は、ＨＴＭ２００の終了命令を実行し、ＨＴＭ方式の後処理を行う。ＨＴＭ方式の後処理は、図２、図３で前述したとおりである。これにより、共有メモリ領域Ｓｍへのアクセス処理（クリティカルセクションの処理）が確定（完了）する。

【０１３５】

［ロック方式の処理］

図１９は、ロック方式の排他取得モジュール１４３、及び、ロック方式の排他解除モジュール１５３の処理の流れを説明するフローチャート図である。

【０１３６】

図１９の（Ａ）は、ロック方式の排他取得モジュール１４３の処理（図１１のＳ１４）の流れを示すフローチャート図である。

【０１３７】

Ｓ４１：ロック方式の排他取得モジュール１４３は、ロック方式に基づくロックが解放されているか否かを判定する。ロック方式の排他取得モジュール１４３は、ロック変数「spinlock」１６０（図１６、図１７）の値がロック状態を示すか否かに基づいて、ロックが解放されているか否かを判定する。

【０１３８】

Ｓ４２：ロック方式に基づくロックが解放済みの場合、または、ロック方式に基づいて排他が解除された場合（Ｓ４１のＹｅｓ）、排他取得モジュール１４１は、ロックを取得する。即ち、排他取得モジュール１４１は、ロック変数１６０の値を、非ロック状態を示す値から、ロック状態を示す値に更新する。

【０１３９】

図１９の（Ｂ）は、ロック方式の排他解除モジュール１５３の処理（図１１のＳ１９）の流れを示すフローチャート図である。

【０１４０】

Ｓ５１：ロック方式の排他解除モジュール１５３は、ロックを開放する。即ち、ロック方式の排他解除モジュール１５３は、ロック変数１６０の値を、ロック状態を示す値から、非ロック状態を示す値に更新する。

【０１４１】

［他の実施の形態］

上記の実施の形態は、オペレーションシステム１３１が、本実施の形態の排他制御プログラム１３３を有する場合を例示した。ただし、この例に限定されるものではない。アプリケーションプログラム１３２が、本実施の形態の排他制御プログラム１３３を含んでいてもよい。

【０１４２】

以上の実施の形態をまとめると、次の付記のとおりである。

【０１４３】

（付記１）

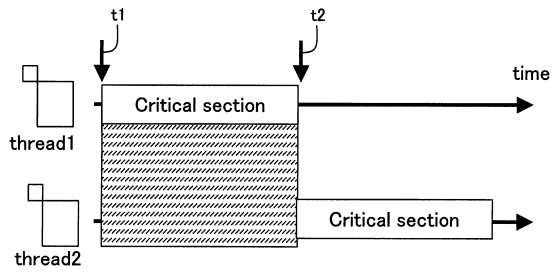
共有メモリ領域を有する記憶部と、

１つまたは複数のスレッドを実行する処理部と、を有し、
前記処理部は、
前記スレッドが前記共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行し、
前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行する、
情報処理装置。
【０１４４】
（付記２）
付記１において、
前記処理部は、前記複数のスレッドを実行中ではない場合、新たなスレッドの実行を開始して前記複数のスレッドを実行中の場合に遷移したとき、前記第１の制御に基づく前記アクセス処理中は、前記新たなスレッドによる前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の開始を待機する、
情報処理装置。
【０１４５】
（付記３）
付記１または２において、
前記第２の制御は、前記一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する前記他のスレッドによる書き込みが発生しない場合に、前記アクセス処理を完了させる、
情報処理装置。
【０１４６】
（付記４）
付記１乃至３のいずれかにおいて、
前記処理部は、前記複数のスレッドを実行中の場合、いずれかの前記スレッドの実行が終了して前記複数のスレッドを実行中ではない場合に遷移したとき、前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の終了時に、前記第２の制御に基づく終了処理を行う、
情報処理装置。
【０１４７】
（付記５）
スレッドが共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行し、
前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行する、
処理をコンピュータに実行させる並列処理プログラム。
【０１４８】
（付記６）
付記５において、

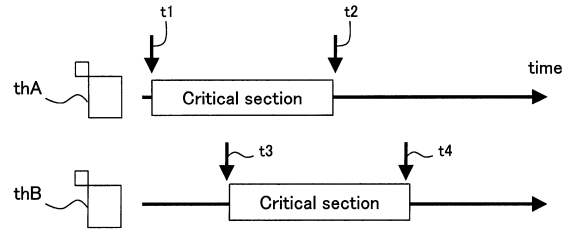
前記複数のスレッドを実行中ではない場合の前記アクセス処理の実行は、新たなスレッドの実行を開始して前記複数のスレッドを実行中の場合に遷移したとき、前記第１の制御に基づく前記アクセス処理中は、前記新たなスレッドによる前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の開始を待機する、
並列処理プログラム。
【０１４９】
（付記７）
付記５または６において、
前記複数のスレッドを実行中の場合の前記アクセス処理の実行は、前記一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する前記他のスレッドによる書き込みが発生しない場合に、前記アクセス処理を完了させる、
並列処理プログラム。
【０１５０】
（付記８）
付記５乃至７のいずれかにおいて、
前記複数のスレッドを実行中の場合の前記アクセス処理の実行は、いずれかの前記スレッドの実行が終了して前記複数のスレッドを実行中ではない場合に遷移したとき、前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の終了時に、前記第２の制御に基づく終了処理を行う、
並列処理プログラム。
【０１５１】
（付記９）
処理部が、スレッドが共有メモリ領域のアクセス処理を実行する際に、当該共有メモリ領域にアクセスする複数のスレッドを実行中か否かを判定し、
処理部が、前記複数のスレッドを実行中ではない場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、他のスレッドによる前記共有メモリ領域へのアクセス処理の開始を待機する第１の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行し、
処理部が、前記複数のスレッドを実行中の場合、一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する他のスレッドによる書き込みが発生した場合に、前記アクセス処理を取り消す第２の制御に基づいて、前記共有メモリ領域へのアクセス処理を実行する、
共有メモリアクセス方法。
【０１５２】
（付記１０）
付記９において、
前記複数のスレッドを実行中ではない場合の前記アクセス処理の実行は、新たなスレッドの実行を開始して前記複数のスレッドを実行中の場合に遷移したとき、前記第１の制御に基づく前記アクセス処理中は、前記新たなスレッドによる前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の開始を待機する、
共有メモリアクセス方法。
【０１５３】
（付記１１）
付記９または１０において、
前記複数のスレッドを実行中の場合の前記アクセス処理の実行は、前記一のスレッドによる前記共有メモリ領域へのアクセス処理中に、前記共有メモリ領域に対する前記他のスレッドによる書き込みが発生しない場合に、前記アクセス処理を完了させる、
共有メモリアクセス方法。
【０１５４】
（付記１２）

付記９乃至１１のいずれかにおいて、
前記複数のスレッドを実行中の場合の前記アクセス処理の実行は、いずれかの前記スレッドの実行が終了して前記複数のスレッドを実行中ではない場合に遷移したとき、前記第２の制御に基づく前記共有メモリ領域へのアクセス処理の終了時に、前記第２の制御に基づく終了処理を行う、
共有メモリアクセス方法。
【符号の説明】
【０１５５】
１００：情報処理装置、１０１：ＣＰＵ、１０２：メモリ、１０３：通信インタフェース部、１０６：バス、Ｓｍ：共有メモリ領域、ｔｈ：スレッド、１３１：オペレーションシステム、１３３：排他制御プログラム、１３２：アプリケーションプログラム

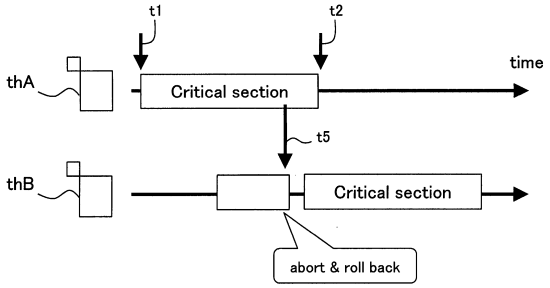
【図１】



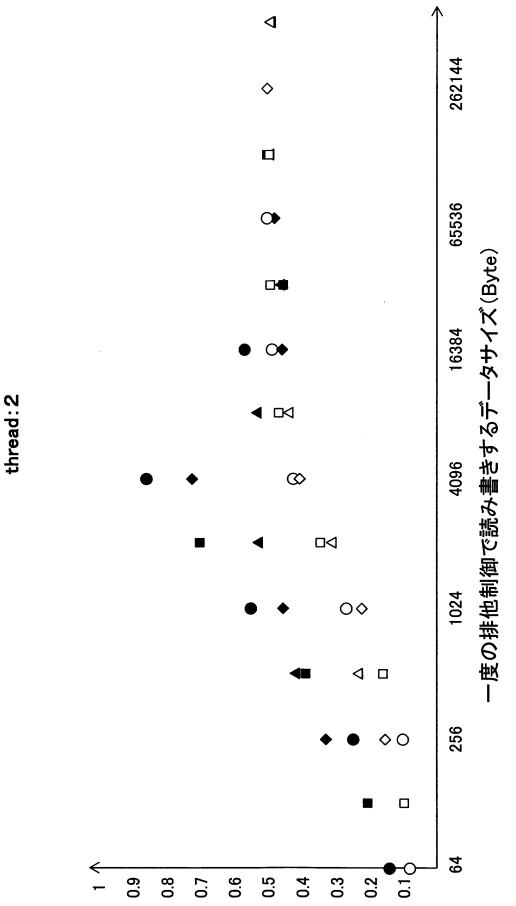
【図２】



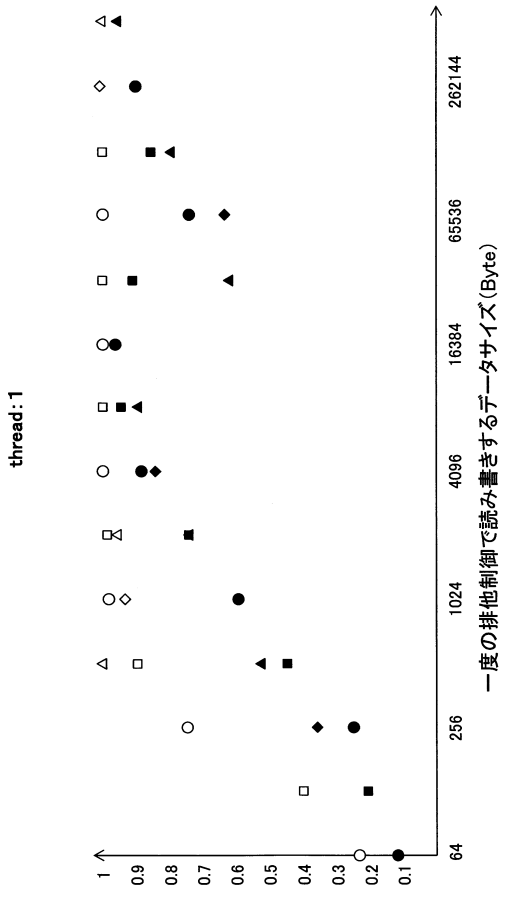
【図３】



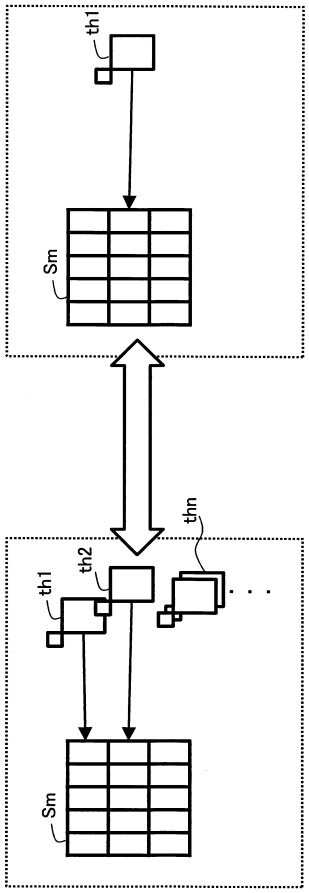
【図４】


thread:2
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
64
256
1024
4096
16384
65536
262144
一度の排他制御で読み書きするデータサイズ(Byte)

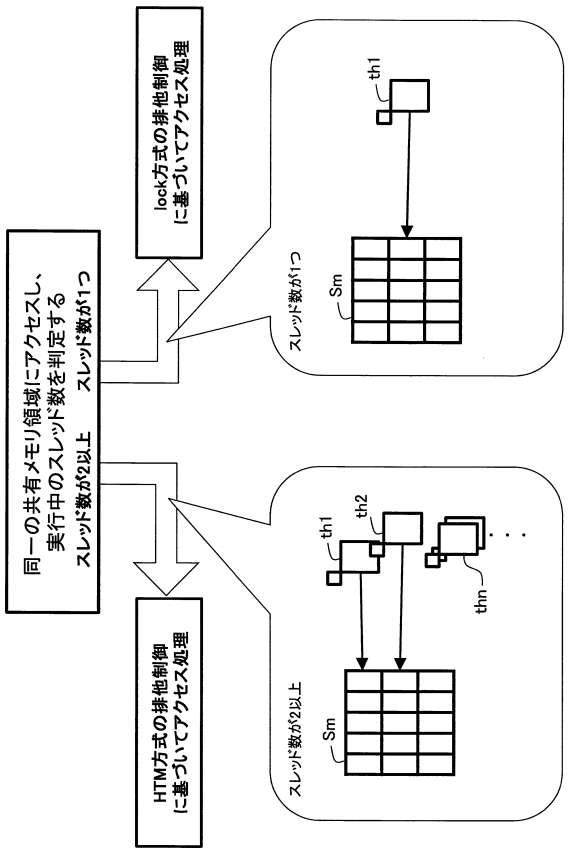
【図５】


thread:1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
64
256
1024
4096
16384
65536
262144
一度の排他制御で読み書きするデータサイズ(Byte)

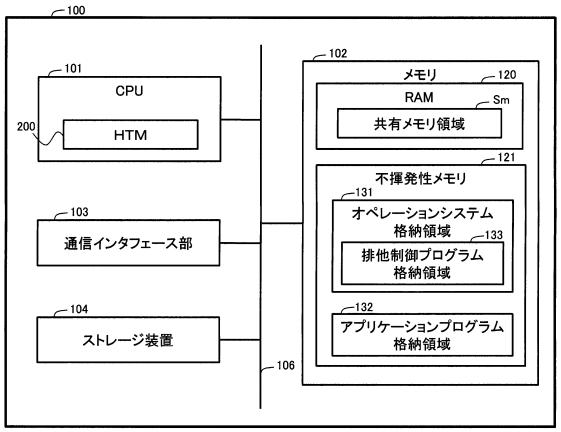
【図６】


th1
Sm
th1
th2
thn
Sm

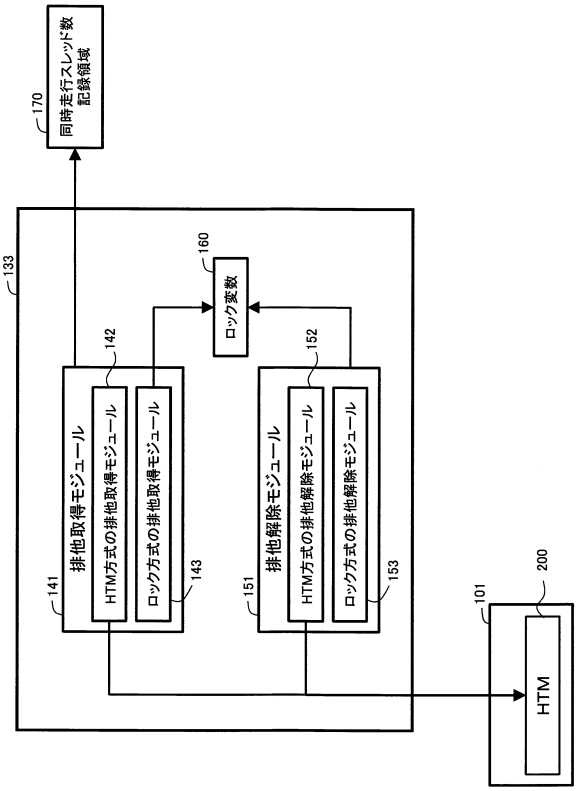
【図７】


同一の共有メモリ領域にアクセスし、
実行中のスレッド数を判定する
スレッド数が2以上　スレッド数が1つ
lock方式の排他制御
に基づいてアクセス処理
HTM方式の排他制御
に基づいてアクセス処理
スレッド数が1つ
th1
Sm
スレッド数が2以上
th1
th2
thn
Sm

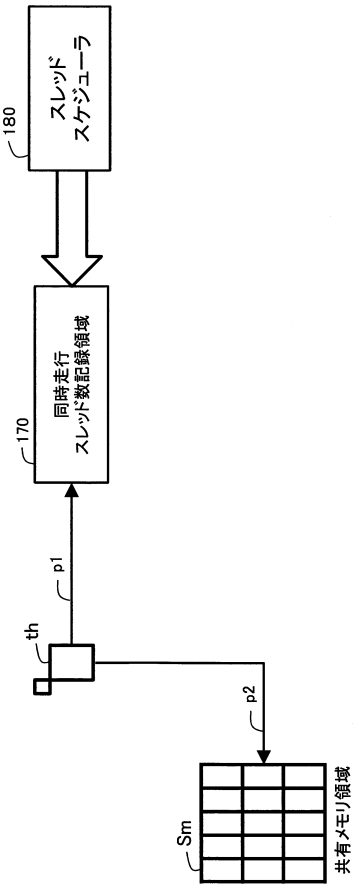
【図８】


100
101
CPU
200
HTM
102
メモリ
120
RAM
Sm
共有メモリ領域
121
不揮発性メモリ
131
オペレーションシステム
格納領域
133
排他制御プログラム
格納領域
132
アプリケーションプログラム
格納領域
103
通信インタフェース部
104
ストレージ装置
106

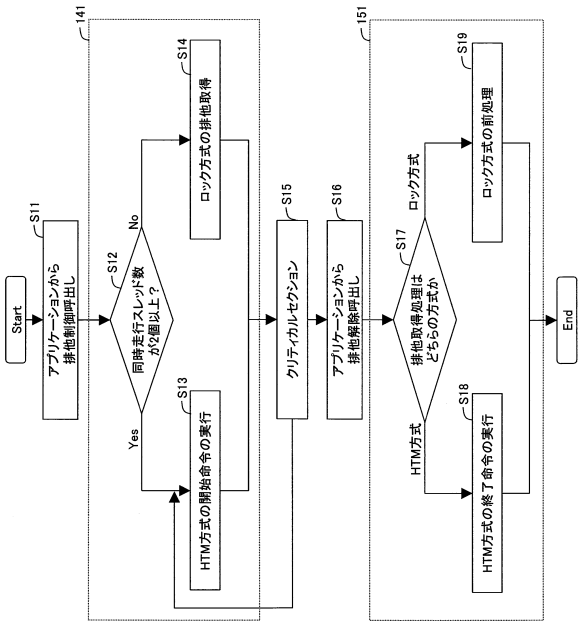
【図９】


170
同時走行スレッド数
記録領域
133
160
ロック変数
142
排他取得モジュール
HTM方式の排他取得モジュール
ロック方式の排他取得モジュール
143
141
152
排他解除モジュール
HTM方式の排他解除モジュール
ロック方式の排他解除モジュール
153
151
101
200
HTM

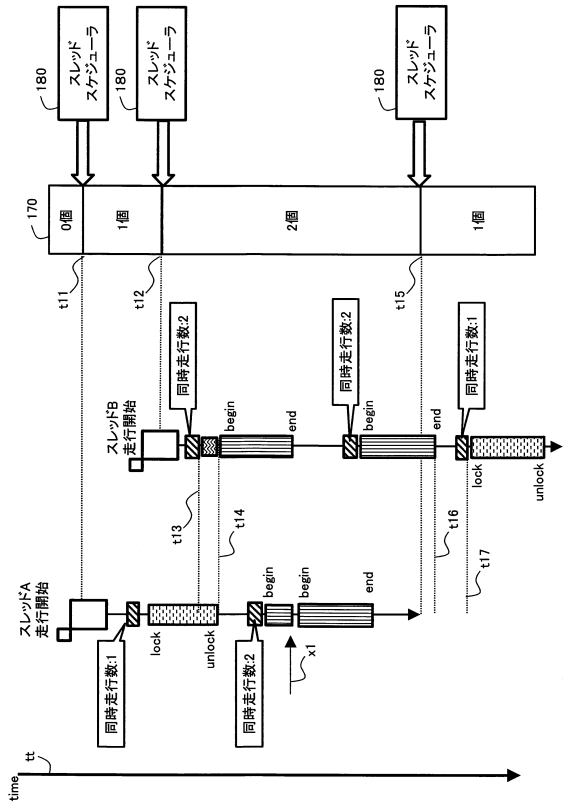
【図１０】


180
スレッド
スケジューラ
170
同時走行
スレッド数記録領域
p1
th
p2
Sm
共有メモリ領域

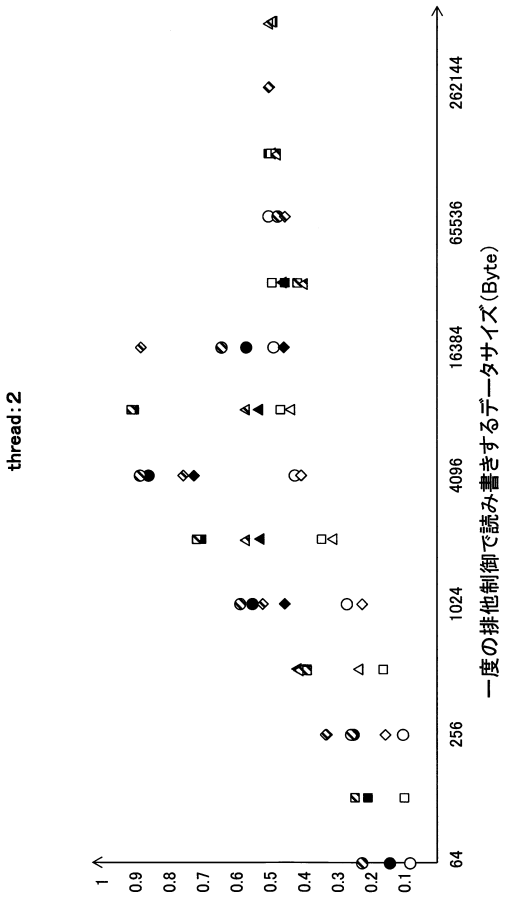
【図１１】


141
151
Start
S11
アプリケーションから
排他制御呼出し
S12
同時走行スレッド数
が2個以上？
No
Yes
S13
HTM方式の開始命令の実行
S14
ロック方式の排他取得
S15
クリティカルセクション
S16
アプリケーションから
排他解除呼出し
S17
排他取得処理は
どちらの方式か
HTM方式
ロック方式
S18
HTM方式の終了命令の実行
S19
ロック方式の前処理
End

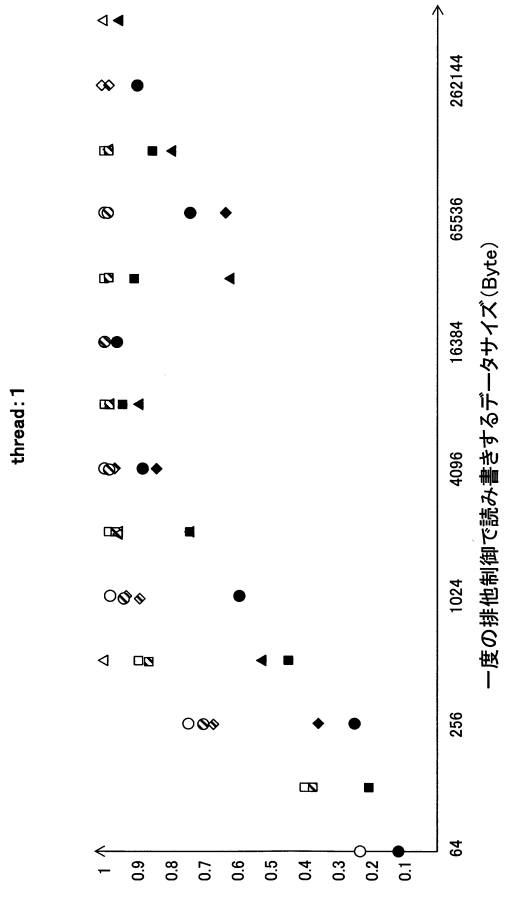
【図１２】



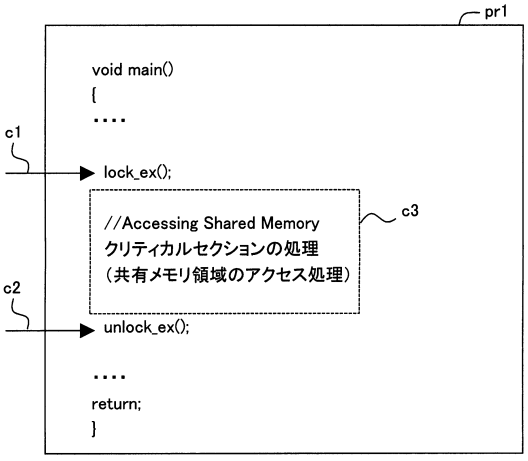
【図１３】



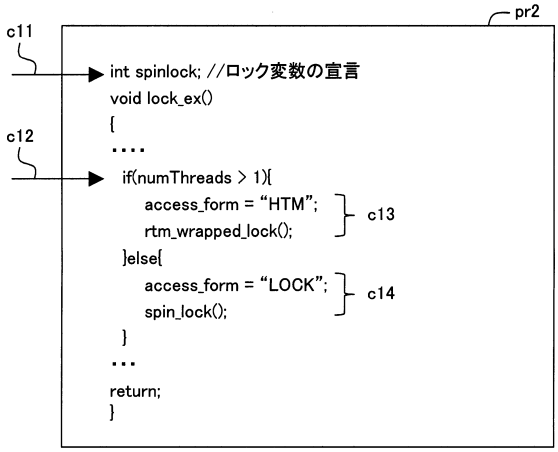
【図１４】



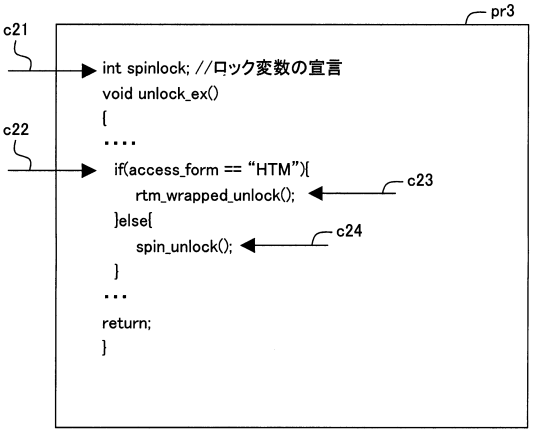
【図１５】



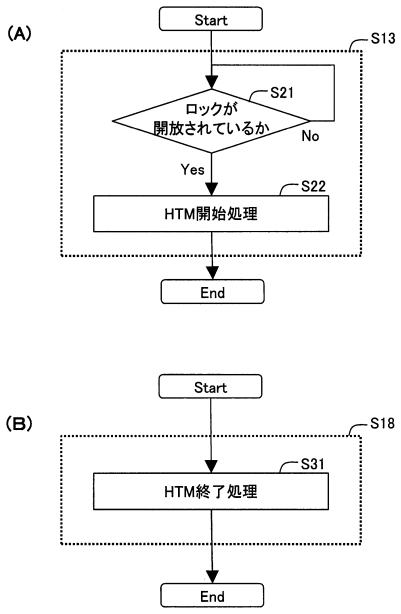
【図１６】



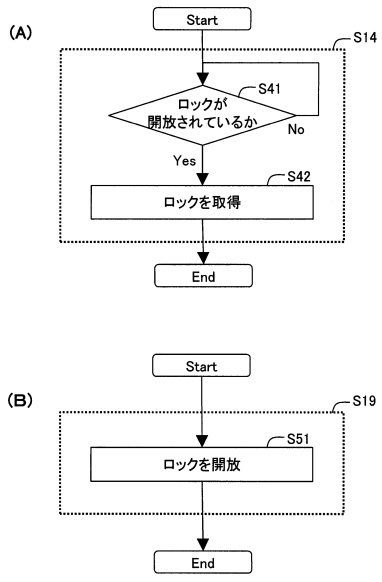
【図１７】



【図１８】



【図１９】



フロントページの続き

審査官　井上　宏一

(56)参考文献　国際公開第２０１５／０５５０８３（WO，Ａ１）
特開２０１０－　６１５２２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４５５－９／５４
Ｇ０６Ｆ　１２／００