

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号

特許第7152474号

(P7152474)

(45)発行日 令和4年10月12日(2022.10.12)

(24)登録日 令和4年10月3日(2022.10.3)

(51)国際特許分類

F I

G 0 6 F 11/34 (2006.01)

G 0 6 F 11/34 1 7 6

G 0 6 F 12/0815(2016.01)

G 0 6 F 12/0815

請求項の数 20 (全44頁)

(21)出願番号 特願2020-509444(P2020-509444)
 (86)(22)出願日 平成30年6月22日(2018.6.22)
 (65)公表番号 特表2020-534589(P2020-534589 A)
 (43)公表日 令和2年11月26日(2020.11.26)
 (86)国際出願番号 PCT/US2018/038875
 (87)国際公開番号 WO2019/055094
 (87)国際公開日 平成31年3月21日(2019.3.21)
 審査請求日 令和3年5月19日(2021.5.19)
 (31)優先権主張番号 62/559,780
 (32)優先日 平成29年9月18日(2017.9.18)
 (33)優先権主張国・地域又は機関 米国(US)
 (31)優先権主張番号 15/915,930
 (32)優先日 平成30年3月8日(2018.3.8)

最終頁に続く

(73)特許権者 314015767
 マイクロソフト テクノロジー ライセン
 シング, エルエルシー
 アメリカ合衆国 ワシントン州 9 8 0 5
 2 レッドモンド ワン マイクロソフト
 ウェイ
 (74)代理人 100079108
 弁理士 稲葉 良幸
 (74)代理人 100109346
 弁理士 大貫 敏史
 (74)代理人 100117189
 弁理士 江口 昭彦
 (74)代理人 100134120
 弁理士 内藤 和彦
 (74)代理人 100108213

最終頁に続く

(54)【発明の名称】 キャッシュコヒーレンスプロトコルデータを用いたキャッシュベーストレース記録

(57)【特許請求の範囲】

【請求項1】

複数の処理ユニットと、

1つ又は複数の補助記憶からデータをキャッシュするために使用され、前記複数の処理ユニットによって共有される複数のキャッシュラインを含むキャッシュメモリであって、前記複数のキャッシュライン内のデータと前記1つ又は複数の補助記憶との間の整合性はキャッシュコヒーレンスプロトコル(CCP)に従って管理される、キャッシュメモリと、少なくとも以下の条件、つまり

(i) 或る操作が前記複数のキャッシュラインの特定のキャッシュラインと前記1つ又は複数の補助記憶との間の相互作用を引き起こしたこと、

(i i) 前記操作を引き起こした前記複数の処理ユニットの特定の処理ユニットについてロギングが有効にされていること、

(i i i) 前記特定のキャッシュラインがロギングのパーティシパントであること、及び

(i v) 前記操作に基づいてトレースにログすべきデータがあることを前記CCPが示していること

が満たされていると判定すること、並びに

前記条件が満たされていると判定することに少なくとも基づいて前記データを前記トレースにログさせることであって、前記データは前記操作を再実行するために使用可能である、ログさせること

を少なくとも行うように計算装置を構成する記憶済みの制御ロジックと

を含む、計算装置。

【請求項 2】

前記操作後に前記特定のキャッシュラインがロギングのパーティシパントのままかどうかを示すために、前記記憶済みの制御ロジックが前記特定のキャッシュラインに関連する1つ又は複数のアカウンティングビットを更新するように前記計算装置を更に構成する、請求項 1 に記載の計算装置。

【請求項 3】

前記特定のキャッシュラインに関連する前記 1 つ又は複数のアカウンティングビットが (i) 単一ビット、 (i i) 前記複数の処理ユニットの 1 つにそれぞれ対応する複数のビット、又は (i i i) プロセッサインデックス値を記憶する複数のビットのうちの 1 つを含む、請求項 2 に記載の計算装置。

10

【請求項 4】

前記特定のキャッシュラインに関連する前記 1 つ又は複数のアカウンティングビットが、1 つ又は複数の補助記憶からのデータをキャッシュするために使用されるキャッシュラインとは別の 1 つ又は複数の予約済みキャッシュライン内に記憶される、請求項 2 に記載の計算装置。

【請求項 5】

前記データを前記トレースにログさせることが、前記データをバッファに書き込むことを含み、前記バッファからトレースファイルにデータをフラッシュすることがメモリバスのアクティビティに基づいて先延ばしされる、請求項 1 に記載の計算装置。

20

【請求項 6】

前記記憶済みの制御ロジックが、アソシアティブキャッシュ内のグループ及びウェイを参照することによって少なくとも 1 つのキャッシュ追い出しをログするように前記計算装置を更に構成する、請求項 1 に記載の計算装置。

【請求項 7】

前記ログされるデータが、異なる C C P 状態間の遷移を含む、請求項 1 に記載の計算装置。

【請求項 8】

前記ログされるデータが、書き込み状態から読み出し状態への遷移、書き込み状態から書き込み状態への遷移、又は読み出し状態から書き込み状態への遷移のうちの少なくとも 1 つを含む、請求項 1 に記載の計算装置。

30

【請求項 9】

前記 C C P を使用してトレースにログすべきデータがあることを識別することが、読み出し状態から読み出し状態への遷移は前記トレースにログする必要がないことを識別することを含む、請求項 1 に記載の計算装置。

【請求項 10】

処理ユニットごとのデータが少なくとも 1 つの別個のデータストリームにログされる、請求項 1 に記載の計算装置。

【請求項 11】

2 つ以上の処理ユニットのデータが同じデータストリームにログされるが、処理ユニット識別子でタグ付けされる、請求項 1 に記載の計算装置。

40

【請求項 12】

前記トレースにログすべき前記データが順序付け情報を含む、請求項 1 に記載の計算装置。

【請求項 13】

前記ログすべきデータがエンクレーブによって前記特定のキャッシュラインに書き込まれるデータを含み、前記データを前記トレースにログさせることが

前記特定のキャッシュラインと前記 1 つ又は複数の補助記憶との間の前記相互作用を引き起こした前記操作が前記エンクレーブと相互作用するスレッドに対応する場合、前記スレッドに対応するトレースデータストリーム内に前記データをログさせること、又は

50

前記特定のキャッシュラインと前記1つ又は複数の補助記憶との間の前記相互作用を引き起こした前記操作が前記エンクレーブに対応する場合、前記スレッドに対応する前記トレースデータストリームから分離されるように前記データをログさせることを含む、請求項1に記載の計算装置。

【請求項14】

複数の処理ユニット、及び1つ又は複数の補助記憶からデータをキャッシュするために使用され、前記複数の処理ユニットによって共有される複数のキャッシュラインを含むキャッシュメモリを含む計算環境内で実装される方法であって、前記複数のキャッシュライン内のデータと前記1つ又は複数の補助記憶との間の整合性はキャッシュコヒーレンスプロトコルに従って管理され、キャッシュコヒーレンスプロトコル（CCP）データを用いたキャッシュベーストレース記録を実行するための方法であり、

10

少なくとも以下の条件、つまり

（i）或る操作が前記複数のキャッシュラインの特定のキャッシュラインと前記1つ又は複数の補助記憶との間の相互作用を引き起こしたこと、

（ii）前記操作を引き起こした前記複数の処理ユニットの特定の処理ユニットについてロギングが有効にされていること、

（iii）前記特定のキャッシュラインがロギングのパーティシパントであること、及び

（iv）前記操作に基づいてトレースにログすべきデータがあることを前記CCPが示していること

が満たされていると判定すること、並びに

20

前記条件が満たされていると判定することに少なくとも基づいて前記データを前記トレースにログさせることであって、前記データは前記操作を再実行するために使用可能である、ログさせること

を含む、方法。

【請求項15】

前記操作後に前記特定のキャッシュラインがロギングのパーティシパントのままかどうかを示すために、前記特定のキャッシュラインに関連する1つ又は複数のアカウントिंगビットを更新することを更に含む、請求項14に記載の方法。

【請求項16】

前記データを前記トレースにログさせることが、前記データをバッファに書き込むことを含み、前記バッファからトレースファイルにデータをフラッシュすることがメモリバスのアクティビティに基づいて先延ばしされる、請求項14に記載の方法。

30

【請求項17】

前記ログされるデータは異なるCCP状態間の遷移を含む、請求項14に記載の方法。

【請求項18】

前記ログされるデータは、書き込み状態から読み出し状態への遷移、書き込み状態から書き込み状態への遷移、又は読み出し状態から書き込み状態への遷移のうちの少なくとも1つを含む、請求項14に記載の方法。

【請求項19】

前記CCPを使用してトレースにログすべきデータがあることを識別することが、読み出し状態から読み出し状態への遷移は前記トレースにログする必要がないことを識別することを含む、請求項14に記載の方法。

40

【請求項20】

複数の処理ユニット、及び1つ又は複数の補助記憶からデータをキャッシュするために使用され、前記複数の処理ユニットによって共有される複数のキャッシュラインを含むキャッシュメモリを含む計算装置において使用するためのコンピュータプログラムであって、前記複数のキャッシュライン内のデータと前記1つ又は複数の補助記憶との間の整合性はキャッシュコヒーレンスプロトコル（CCP）に従って管理され、前記コンピュータプログラムは、

少なくとも以下の条件、つまり

50

(i) 或る操作が前記複数のキャッシュラインの特定のキャッシュラインと前記1つ又は複数の補助記憶との間の相互作用を引き起こしたこと、

(i i) 前記操作を引き起こした前記複数の処理ユニットの特定の処理ユニットについてロギングが有効にされていること、

(i i i) 前記特定のキャッシュラインがロギングのパーティシパントであること、及び

(i v) 前記操作に基づいてトレースにログすべきデータがあることを前記CCPが示していること

が満たされていると判定すること、並びに

前記条件が満たされていると判定することに少なくとも基づいて前記データを前記トレースにログさせることであって、前記データは前記操作を再実行するために使用可能である、ログさせること

を少なくとも前記計算装置に行わせるために1つ又は複数の処理ユニットによって実行可能なコンピュータ実行可能命令を記憶しているコンピュータ可読媒体を含む、コンピュータプログラム。

【発明の詳細な説明】

【背景技術】

【0001】

背景

【0001】 ソフトウェアアプリケーションの開発中にコードを書くとき、一般に開発者はランタイムエラー及び他のソースコードエラーを見つけるためにコードを「デバッグ」するのにかなりの時間を費やす。その際、異なる入力に基づくプログラムの挙動を観察すること、(例えば変数値をプリントするための、実行の分岐を追跡するための等)デバッグコードを挿入すること、コードの一部を一時的に除去すること等、開発者はソースコードのバグを再現しローカライズするために幾つかの手法を取ることができる。ランタイムエラーを極めて正確なコードバグまで突き止めることは、アプリケーション開発時間のかなりの部分を占める場合がある。

【0002】

【0002】 コードのデバッグプロセスで開発者を支援するために、多くの種類のデバッグアプリケーション(「デバッガ」)が開発されている。これらのツールは、コンピュータコードの実行を追跡し、視覚化し、変更する能力を開発者に提供する。例えばデバッガは、とりわけコード命令の実行を視覚化することができ、コードの実行中の様々な時点におけるコード変数値を提示することができ、開発者がコードの実行パスを変更できるようにすることができ、及び／又は開発者が関心のあるコード要素上で(実行中に到達したときコードの実行を中断させる)「ブレークポイント」及び／又は「ウォッチポイント」を設定できるようにし得る。

【0003】

【0003】 デバッグアプリケーションの新興形態は「タイムトラベル」、「リバース」、又は「履歴」デバッグを使用可能にする。「タイムトラベル」デバッグでは、プログラム(例えばスレッド等の実行可能エンティティ)の実行が、トレースアプリケーションによって1つ又は複数のトレースファイル内に記録／トレースされる。これらのトレースファイルは、順方向分析及び逆方向分析の両方のために後でプログラムの実行を再実行するために使用され得る。例えば「タイムトラベル」デバッガは、開発者が(従来のデバッガのように)順方向ブレークポイント／ウォッチポイントを設定すること、並びにリバースブレークポイント／ウォッチポイントを設定することを可能にし得る。

【発明の概要】

【課題を解決するための手段】

【0004】

概要

【0004】 本明細書の実施形態は、トレースファイルにどのデータをログすべきかを決定するために、プロセッサの共有キャッシュをそのキャッシュコヒーレンスプロトコル(C

10

20

30

40

50

ＣＰ）と共に利用することによって「タイムトラベル」デバッグ記録を向上させる。これにより従来の手法と比較してトレースファイルのサイズを桁違いに低減し、それによりトレース記録のオーバーヘッドを大幅に減らすことができる。

【０００５】

【０００５】 一部の実施形態では、（ｉ）複数の処理ユニットと、（ｉｉ）１つ又は複数の補助記憶からデータをキャッシュするために使用され、複数の処理ユニットによって共有される複数のキャッシュラインを含むキャッシュメモリとを含む計算環境内で実装される。複数のキャッシュライン内のデータと１つ又は複数の補助記憶との間の整合性はキャッシュコヒーレンスプロトコルに従って管理される。

【０００６】

【０００６】 これらの実施形態は、ＣＣＰデータを用いたキャッシュベーストレース記録を実行することを含む。これらの実施形態は、或る操作が複数のキャッシュラインの特定のキャッシュラインと１つ又は複数の補助記憶との間の相互作用を引き起こしたと判定すること、その操作を引き起こした複数の処理ユニットの特定の処理ユニットについてロギングが有効にされていると判定すること、特定のキャッシュラインがロギングのパーティシパントだと判定すること、及びトレースにログすべきデータがあることをＣＣＰが示していると判定することを含む。少なくともこれらの判定に基づき、実施形態はこのデータをトレースにログさせる。データは操作を再実行するために使用可能である。

【０００７】

【０００７】 この概要は、以下の詳細な説明の中で更に説明する一連の概念を単純化した形式で紹介するために提供する。この概要は特許請求の範囲に記載する内容の重要な特徴又は必須の特徴を識別することは意図せず、特許請求の範囲に記載する内容の範囲を決定する際の助けとして使用されることも意図しない。

【０００８】

図面の簡単な説明

【０００８】 本発明の上記で挙げた及び他の利点及び特徴を得ることができるやり方を説明するために、上記で簡潔に説明した本発明のより詳細な説明を添付図面に示すその具体的実施形態に関して行う。これらの図面は本発明の典型的な実施形態を示すに過ぎず、従って本発明の範囲の限定だと解釈すべきではないことを理解し、添付図面を使用することによって本発明を更なる特定性及び詳細と共に記載し説明する。

【図面の簡単な説明】

【０００９】

【図１】【０００９】キャッシュコヒーレンスプロトコル（ＣＣＰ）データを用いた共有キャッシュによるコード実行の「ビット精度」トレースの記録を助ける計算環境の一例を示す。

【図２】【００１０】共有キャッシュの一例を示す。

【図３】【００１１】ＣＣＰデータを用いたキャッシュベーストレース記録を実行するための方法の一例の流れ図を示す。

【図４Ａ】【００１２】１つ又は複数の追加のアカウントティングビットを用いてそのキャッシュラインのそれぞれを拡張する共有キャッシュの一例を示す。

【図４Ｂ】【００１３】従来のキャッシュラインに適用されるアカウントティングビットを記憶するための１つ又は複数の予約済みキャッシュラインを含む共有キャッシュの一例を示す。

【図５】【００１４】アソシアティブキャッシュマッピングの一例を示す。

【図６Ａ】【００１５】共有キャッシュ内の単一ライン上の４つの処理ユニットによる読み出し及び書き込みアクティビティの一例を示す表を示す。

【図６Ｂ】【００１６】図６Ａに示す読み出し及び書き込みアクティビティに基づく、追跡されたキャッシュコヒーレンス状態の一例を示す表を示す。

【図６Ｃ】【００１７】図６Ａに示す読み出し及び書き込みアクティビティに基づく、共有キャッシュのアカウントティングビット（即ちユニットビット、インデックスビット、及び／又はフラグビット）内に記憶されたデータの一例を示す表を示す。

【図６Ｄ】【００１８】図６Ａに示す読み出し及び書き込みアクティビティに関連してトレース

10

20

30

40

50

ファイルに書き込まれ得るログデータの一例を示す表を示す。

【図 7 A】[0019]プロセッサがどのように追跡されるのかにもよるが、一部の読み出し→読み出し遷移をトレースから省略することができる一例を示す。

【図 7 B】[0020]図 7 A で強調した読み出し→読み出し遷移を省略したロギングデータの一例を示す。

【図 7 C】[0021]「インデックスビット」が使用され、インデックスが読み出し時に更新される場合に記録され得るロギングデータの一例を示す表を示す。

【図 8 A】[0022]4つの処理ユニット及び L1～L3 キャッシュをそれぞれ含む2つのプロセッサを含む計算環境の一例を示す。

【図 8 B】[0023]図 8 A の処理ユニットの一部によって実行される読み出し及び書き込み操作の一例を示す表を示す。

10

【図 9 A】[0024]2つの処理ユニットによる読み出し及び書き込みの一例を示す表を示す。

【図 9 B】[0025]ログエントリが作成され得る場合を比較する表を示すの一例を示す、CCP インデックス情報とキャッシュラインフラグビットとを提供する環境に対して、CCP ユニット情報とキャッシュラインフラグビットとを提供する環境。

【図 10 A】[0026]メモリアドレスの異なる部分、及びアソシアティブキャッシュに対するそれらの部分の関係の一例を示す。

【図 10 B】[0027]アソシアティブキャッシュ内のキャッシュミス及びキャッシュ追い出しをログする一例を示す。

20

【発明を実施するための形態】

【0010】

詳細な説明

【0028】 本明細書の実施形態は、トレースファイルにどのデータをログすべきかを決定するために、プロセッサの共有キャッシュをそのキャッシュコヒーレンスプロトコル（CCP）と共に利用することによって「タイムトラベル」デバッグ記録を向上させる。これにより従来の手法と比較してトレースファイルのサイズを桁違いに低減し、それによりトレース記録のオーバーヘッドを大幅に減らすことができる。

【0011】

【0029】 図 1 は、キャッシュコヒーレンスプロトコルデータを用いた共有キャッシュによるコード実行の「ビット精度」トレースの記録を助ける計算環境 100 の一例を示す。図示のように、実施形態は例えば1つ又は複数のプロセッサ 102、システムメモリ 103、1つ又は複数のデータストア 104、及び／又は入出力ハードウェア 105 等のコンピュータハードウェアを含む専用又は汎用コンピュータシステム 101 を含み又は利用し得る。

30

【0012】

【0030】 本発明の範囲に含まれる実施形態は、コンピュータ実行可能命令及び／又はデータ構造を搬送し又は記憶するための物理媒体及び他のコンピュータ可読媒体を含む。かかるコンピュータ可読媒体は、コンピュータシステム 101 によってアクセス可能な任意の利用可能な媒体とすることができる。コンピュータ実行可能命令及び／又はデータ構造を記憶するコンピュータ可読媒体はコンピュータ記憶装置である。コンピュータ実行可能命令及び／又はデータ構造を搬送するコンピュータ可読媒体は伝送媒体である。従って、限定ではなく例として、本発明の実施形態は少なくとも2つの明確に異なる種類のコンピュータ可読媒体、つまりコンピュータ記憶装置及び伝送媒体を含むことができる。

40

【0013】

【0031】 コンピュータ記憶装置は、コンピュータ実行可能命令及び／又はデータ構造を記憶する物理的なハードウェア装置である。コンピュータ記憶装置は、RAM、ROM、EEPROM、ソリッドステートドライブ（「SSD」）、フラッシュメモリ、相変化メモリ（「PCM」）、光ディスクストレージ、磁気ディスクストレージ若しくは他の磁気記憶装置、又はコンピュータ実行可能命令若しくはデータ構造形式でプログラムコードを

50

記憶するために使用でき、開示する本発明の機能を実装するためにコンピュータシステム 101 によってアクセスされ実行され得る他の任意のハードウェア装置等の様々なコンピュータハードウェアを含む。従って、例えばコンピュータ記憶装置は図示のシステムメモリ 103、コンピュータ実行可能命令及び／又はデータ構造を記憶することができる図示のデータストア 104、又は後述のオンプロセッサストレージ等の他のストレージを含み得る。

【0014】

【0032】 伝送媒体は、コンピュータ実行可能命令又はデータ構造形式でプログラムコードを搬送するために使用することができ、コンピュータシステム 101 によってアクセス可能なネットワーク及び／又はデータリンクを含み得る。「ネットワーク」は、コンピュータシステム及び／又はモジュール及び／又は他の電子装置間の電子データの搬送を可能にする 1 つ又は複数のデータリンクとして定義する。情報がネットワーク又は別の通信接続（ハードワイヤード、無線、又はハードワイヤード若しくは無線の組み合わせ）上でコンピュータシステムに転送され又は提供される場合、コンピュータシステムはその接続を伝送媒体として見ることができる。上記の組み合わせもコンピュータ可読媒体の範囲に含まれるべきである。例えば入出力ハードウェア 105 は、コンピュータ実行可能命令又はデータ構造形式でプログラムコードを搬送するために使用可能なネットワーク及び／又はデータリンクを接続するハードウェア（例えばネットワークインタフェースモジュール（例えば「NIC」））を含み得る。

【0015】

【0033】 更に、様々なコンピュータシステムコンポーネントに到達すると、コンピュータ実行可能命令又はデータ構造形式のプログラムコードを伝送媒体からコンピュータ記憶装置に（又はその逆に）自動的に転送することができる。例えばネットワーク又はデータリンク上で受信されるコンピュータ実行可能命令又はデータ構造は NIC（例えば入出力ハードウェア 105）内の RAM の中にバッファし、コンピュータシステム 101 におけるシステムメモリ 103 及び／又は揮発性の低いコンピュータ記憶装置（例えばデータストア 104）に最終的に転送することができる。従ってコンピュータ記憶装置は、伝送媒体も（又は伝送媒体を主に）利用するコンピュータシステムコンポーネントに含まれ得ることを理解すべきである。

【0016】

【0034】 コンピュータ実行可能命令は、例えばプロセッサ 102 において実行されるとき、コンピュータシステム 101 に特定の機能又は機能群を実行させる命令及びデータを含む。コンピュータ実行可能命令は、例えば 2 進数、アセンブリ言語等の中間形式命令、更にはソースコードとすることができる。

【0017】

【0035】 本発明は、パーソナルコンピュータ、デスクトップコンピュータ、ラップトップコンピュータ、メッセージプロセッサ、携帯端末、マルチプロセッサシステム、マイクロプロセッサベース又はプログラム可能な家庭用電化製品、ネットワーク PC、ミニコンピュータ、メインフレームコンピュータ、携帯電話、PDA、タブレット、ページャ、ルータ、スイッチ等を含む、多くの種類のコンピュータシステム構成を有するネットワーク計算環境内で実施できることを当業者なら理解されよう。本発明は、ネットワークを介して（ハードワイヤードデータリンク、無線データリンク、又はハードワイヤードデータリンクと無線データリンクとの組み合わせによって）リンクされるローカル及びリモートコンピュータシステムがどちらもタスクを実行する分散システム環境内でも実施することができる。そのため分散システム環境では、コンピュータシステムが複数の構成コンピュータシステムを含み得る。分散システム環境では、プログラムモジュールがローカルメモリ記憶装置及びリモートメモリ記憶装置の両方の中に位置し得る。

【0018】

【0036】 本発明はクラウドコンピューティング環境内で実施できることも当業者なら理解されよう。必須ではないがクラウドコンピューティング環境は分散されてもよい。分散

される場合、クラウドコンピューティング環境は組織内で国際的に分散させることができ、及び／又は複数の組織にわたって所有されるコンポーネントを有し得る。この説明及び添付の特許請求の範囲では、「クラウドコンピューティング」は、構成可能な計算資源（例えばネットワーク、サーバ、ストレージ、アプリケーション、及びサービス）の共有プールへのオンデマンドネットワークアクセスを可能にするためのモデルとして定義する。「クラウドコンピューティング」の定義は、適切に展開されたときにかかるモデルから得ることができる他の多くの利点の何れにも限定されない。

【0019】

【0037】 クラウドコンピューティングモデルは、オンデマンドセルフサービス、ブロードネットワークアクセス、資源プーリング、急速な弾性、度数制等の様々な特性で構成され得る。クラウドコンピューティングモデルは、例えばSoftware as a Service（「SaaS」）、Platform as a Service（「PaaS」）、及びInfrastructure as a Service（「IaaS」）等の様々なサービスモデルの形でももたらされ得る。クラウドコンピューティングモデルは、プライベートクラウド、コミュニティクラウド、パブリッククラウド、ハイブリッドクラウド等の様々な展開モデルを使用して展開することもできる。

【0020】

【0038】 クラウドコンピューティング環境等の一部の実施形態は、それぞれ1つ又は複数の仮想マシンを実行することができる1つ又は複数のホストを含むシステムを含み得る。動作中、仮想マシンは動作中の計算システムをエミュレートし、オペレーティングシステム及びことによると1つ又は複数の他のアプリケーションもサポートする。一部の実施形態では、各ホストは、仮想マシンの点から抽象化される物理資源を使用して仮想マシンのための仮想資源をエミュレートするハイパーバイザを含む。ハイパーバイザは、仮想マシン間の適切な分離も提供する。従って、任意の所与の仮想マシンの観点から、ハイパーバイザは仮想マシンが物理資源の外観（例えば仮想資源）としかインタフェースしていなくても物理資源とインタフェースしている錯覚をもたらす。物理資源の例は、処理容量、メモリ、ディスク空き容量、ネットワーク帯域幅、メディアドライブ等を含む。

【0021】

【0039】 図示のように、データストア104は、例えばトレーサ104a、オペレーティングシステムカーネル104b、及びアプリケーション104c（例えばトレーサ104a及び1つ又は複数のトレースファイル104dによるトレースの対象であるアプリケーション）等のアプリケーションプログラムを表すコンピュータ実行可能命令及び／又はデータ構造を記憶することができる。これらのプログラムが（例えばプロセッサ102を使用して）実行されている間、システムメモリ103はランタイムデータ構造、コンピュータ実行可能命令等の対応するランタイムデータを記憶することができる。従って図1は、（例えばアプリケーション104cにそれぞれ対応する）ランタイムアプリケーションコード103a及びアプリケーションランタイムデータ103bを含むものとしてシステムメモリ103を示す。

【0022】

【0040】 トレーサ104aは、アプリケーション104c等のアプリケーションの実行のビット精度トレースを記録し、トレースファイル104d内にトレースデータを記憶するために使用可能である。一部の実施形態では、トレーサ104aが独立型アプリケーションであるのに対し、他の実施形態ではトレーサ104aが、オペレーティングシステムカーネル104b、ハイパーバイザ、クラウドファブリック等の別のソフトウェアコンポーネントに統合される。トレースファイル104dはデータストア104内に記憶されているものとして図示するが、トレースファイル104dはシステムメモリ103内に又は他の何らかの記憶装置において排他的に又は一時的に記録されてもよい。

【0023】

【0041】 図1は、プロセッサ102の内部ハードウェアコンポーネントの簡略化した表現を含む。図示のように、各プロセッサ102は複数の処理ユニット102aを含む。各処理ユニットは、物理的（即ち物理的なプロセッサコア）及び／又は論理的（即ち物理コ

10

20

30

40

50

アにおいて複数のアプリケーションスレッドが実行されるハイパースレッディングをサポートする物理コアによって示される論理コア）であり得る。従って、例えばプロセッサ102が一部の実施形態で単一の物理処理ユニット（コア）のみを含み得る場合でも、その単一の物理処理ユニットによって示される2つ以上の論理処理ユニット102aを含むことができる。

【0024】

【0042】 各処理ユニット102aは、アプリケーション（例えばトレサ104a、オペレーティングカーネル104b、アプリケーション104c等）によって定められるプロセッサ命令を実行し、それらの命令は既定のプロセッサ命令セットアーキテクチャ（ISA）の中から選択される。各プロセッサ102の特定のISAは、プロセッサの製造業者及びモデルに基づいて異なる。一般的なISAは、INTEL, INC.のIA-64アーキテクチャ及びIA-32アーキテクチャ、ADVANCED MICRO DEVICES, INC.のAMD64アーキテクチャ、及びARM HOLDINGS, PLCの様々なAdvanced RISC Machine（「ARM」）アーキテクチャを含むが、他の多数のISAが存在し本発明によって使用され得る。概して「命令」は、プロセッサによって実行可能なコードの最小の外部可視（即ちプロセッサにとって外部の）ユニットである。

【0025】

【0043】 各処理ユニット102aは、共有キャッシュ102bからプロセッサ命令を取得し、共有キャッシュ102b内のデータに基づいて、レジスタ102d内のデータに基づいて、及び／又は入力データなしにプロセッサ命令を実行する。概して、共有キャッシュ102bは、システムメモリ103及び／又は別のキャッシュ等の補助記憶の一部のオンプロセッサコピーを記憶する少量の（即ちシステムメモリ103の典型的な量に対して少ない）ランダムアクセスメモリである。例えばアプリケーションコード103aを実行するとき、共有キャッシュ102bはアプリケーションランタイムデータ103bの一部を含む。処理ユニット102aが共有キャッシュ102b内にまだ記憶されていないデータを必要とする場合は「キャッシュミス」が発生し、そのデータがシステムメモリ103からフェッチされる（ことによると共有キャッシュ102bから他の一部のデータを「追い出す」）。

【0026】

【0044】 典型的には、共有キャッシュ102bは複数の「キャッシュライン」を含み、「キャッシュライン」のそれぞれは補助記憶からのメモリのチャンクを記憶する。例えば図2は、複数のキャッシュライン203を含む共有キャッシュ200の少なくとも一部の一例を示し、複数のキャッシュライン203のそれぞれはアドレス部分201及び値部分202を含む。各キャッシュライン203のアドレス部分201は、ラインが対応する補助記憶（例えばシステムメモリ103）内のアドレスを記憶することができ、値部分202は補助記憶から受信される値を最初に記憶することができる。値部分202は処理ユニット102aによって修正され、最終的に補助記憶に追い出され得る。楕円によって示すように、共有キャッシュ200は多数のキャッシュラインを含むことができる。例えば現代のINTELプロセッサは、512以上のキャッシュラインを含むレイヤ1キャッシュを含み得る。このキャッシュでは、各キャッシュラインが典型的には8バイト（64ビット）のメモリアドレスを参照して64バイト（512ビット）の値を記憶するために使用可能である。

【0027】

【0045】 各キャッシュライン203のアドレス部分201内に記憶されるアドレスは、システムメモリ103内の実際のメモリアドレス等の物理アドレスであり得る。或いは各キャッシュライン203のアドレス部分201内に記憶されるアドレスは、抽象化を行うために物理アドレスに割り当てられるアドレスである仮想アドレスであり得る。かかる抽象化は、例えばプロセッサ102において実行される異なるプロセス間のメモリ分離を助けるために使用され得る。仮想アドレスが使用される場合、プロセッサ102は、物理メモリアドレスと仮想メモリアドレスとの間のマッピングを保つ変換索引バッファ（TLB

10

20

30

40

50

) 102f (通常はメモリ管理ユニット (MMU) の一部である) を含むことができる。

【0028】

[0046] 共有キャッシュ102bは、コードキャッシュ部分及びデータキャッシュ部分を含むことができる。例えばアプリケーションコード103aを実行するとき、共有キャッシュ102bのコード部分はアプリケーションコード103a内に記憶されるプロセッサ命令の少なくとも一部を記憶し、共有キャッシュ102bのデータ部分はアプリケーションランタイムデータ103bのデータ構造の少なくとも一部を記憶する。多くの場合、プロセッサキャッシュは別個のティア／層 (例えば層1 (L1)、層2 (L2)、及び層3 (L3)) に分割され、一部のティア (例えばL3) はことによるとプロセッサ102から離れて存在する。従って、共有キャッシュ102bはこれらの層の1つ (L1) を含むことができ、又はこれらの層の複数を含むことができる。

10

【0029】

[0047] 複数のキャッシュ層が使用される場合、処理ユニット102aは最下位層 (L1) と直接相互作用する。大抵の場合、データは層間を流れる (例えば読み出し時にL3キャッシュがシステムメモリ103と相互作用しL2キャッシュにデータを供給し、それを受けてL2キャッシュがL1キャッシュにデータを供給する)。処理ユニット102aが書き込みを行う必要がある場合、処理ユニット102a間で共有されていたデータに影響を与えたキャッシュがそのデータをもう有さないことを確実にするためにキャッシュが連携する。この連携は (後で論じる) キャッシュコヒーレンスプロトコルを使用して行われる。

20

【0030】

[0048] キャッシュは、包含的、排他的とすることができ、又は包含的挙動及び排他的挙動の両方を含むことができる。例えば包含的キャッシュでは、L3層はその下のL2層内のデータの超集合を記憶し、L2層はその下のL1層の超集合を記憶する。排他的キャッシュでは層をばらばらにすることができ、例えばL1キャッシュが必要とするデータがL3キャッシュ内にある場合、それらのキャッシュはデータ、アドレス等の情報をスワップすることができる。

【0031】

[0049] 各処理ユニット102はマイクロコード102cも含み、マイクロコード102cは、プロセッサ102の動作を制御する制御ロジック (即ち実行可能命令) を含み、プロセッサのハードウェアと実行中のアプリケーションにプロセッサ102によって公開されるプロセッサISAとの間のインタプリタとして概して機能する。マイクロコード102は、ROM、EEPROM等のオンプロセッサストレージ上に実装され得る。

30

【0032】

[0050] レジスタ102dは、プロセッサ102のISAに基づいて定められ、プロセッサ命令によって読み出され及び／又は書き込まれるハードウェアベースの記憶場所である。例えばレジスタ102dは、命令による使用のために共有キャッシュ102bからフェッチされる値を記憶するために、命令の実行結果を記憶するために、及び／又は命令を実行した副次的作用の一部 (例えば値が変化するサイン、値がゼロに到達するサイン、繰り上げが発生するサイン等)、プロセッサのサイクルカウント等のステータス又は状態を記憶するためによく使用される。従って一部のレジスタ102dは、プロセッサ命令を実行することによって引き起こされる一部の状態変化を知らせるために使用される「フラグ」を含むことができる。一部の実施形態では、プロセッサ102は、プロセッサ動作の様々な側面を制御するために使用される制御レジスタも含み得る。

40

【0033】

[0051] 一部の実施形態では、プロセッサ102が1つ又は複数のバッファ102eを含むことができる。本明細書で後述するように、バッファ102eはトレースデータのための一時的な記憶場所として使用され得る。従って、例えばプロセッサ102はトレースデータの一部を記憶することができる、バッファ102e、使用可能なメモリバス帯域幅があるとき等の適切な時点においてトレースファイル104dにそのデータをフラッシュ

50

することができる。一部の実装形態では、バッファ 102 e が共有キャッシュ 102 b の一部であり得る。

【0034】

【0052】 上記のように、共有キャッシュ 102 b を有するプロセッサはキャッシュコヒーレンスプロトコル（「CCP」）に従ってキャッシュを操作する。具体的には CCP は、様々な処理ユニット 102 a が共有キャッシュ 102 b 内のデータから読み出しし、かかるデータに書き込むとき、共有キャッシュ 102 b 内のデータと補助データストア（例えばシステムメモリ 103 又は別のキャッシュ）との間で整合性をどのように保つのか、及び様々な処理ユニット 102 a が共有キャッシュ 102 b 内の所与の位置から有効なデータを常に読み出すことをどのように確実にするのかを定める。CCP は、典型的にはプロセッサ 102 の ISA によって定められるメモリモデルに関係し、かかるメモリモデルを使用可能にする。

【0035】

【0053】 よくある CCP の例は、MSI プロトコル（即ち修正、共有、及び無効）、MESI プロトコル（即ち修正、排他的、共有、及び無効）、及び MOESI プロトコル（即ち修正、所有、排他的、共有、及び無効）を含む。これらのプロトコルのそれぞれは、共有キャッシュ 102 b 内の個々の位置（例えばライン）の状態を定める。「修正」キャッシュ位置は共有キャッシュ 102 b 内で修正されているデータを含み、従って補助記憶（例えばシステムメモリ 103 又は別のキャッシュ）内の対応するデータと潜在的に矛盾する。「修正」状態を有する位置が共有キャッシュ 102 b から追い出されると、よくある CCP はそのデータが補助記憶に書き戻されること、又は別のキャッシュがこの責任を引き継ぐことをキャッシュが保証することを要求する。「共有」キャッシュ位置は、補助記憶内のデータから修正されておらず、読み出し専用状態にあり、処理ユニット 102 a によって共有されるデータを含む。共有キャッシュ 102 b は、補助記憶に書き込むことなくこのデータを追い出すことができる。「無効」キャッシュ位置は有効なデータを含まず、空だと思ふことができ、キャッシュミスに由来するデータを記憶するために使用可能であり得る。「排他的」キャッシュ位置は補助記憶と一致するデータを含み、単一の処理ユニット 102 a のみによって使用される。「排他的」キャッシュ位置は何時でも（即ち読み出し要求に応答して）「共有」状態に変更することができ、又はそこに書き込むときに「修正」状態に変更することができる。「所有」キャッシュ位置は 2 つ以上の処理ユニット 102 a によって共有されるが、処理ユニットの 1 つは「所有」キャッシュ位置に対する変更を行うための排他的権利を有する。その処理が変更を加える場合、その処理が他の処理ユニットに直接又は間接的に通知し、それは通知される処理ユニットが CCP 実装に基づいて無効化又は更新する必要があるからである。

【0036】

【0054】 異なる CCP がキャッシュコヒーレンスを追跡し、そのキャッシュコヒーレンスデータをトレーサ 104 a に提供する粒度は異なり得る。例えばスペクトルの一端において、一部の CCP はキャッシュラインごとに並びに処理ユニットごとにキャッシュコヒーレンスを追跡する。従って、各キャッシュラインの状態は各処理ユニットに関係するので、これらの CCP は各キャッシュラインの状態を追跡することができる。図 6 A ～ 図 6 D に関連して以下の例で示すように、これは単一のキャッシュラインが各処理ユニット 102 a に関係するのでその状態に関する情報を有し得ることを意味する。他の CCP の粒度は更に低く、キャッシュコヒーレンス、キャッシュラインのレベルだけを追跡する（処理ユニットごとの情報を欠く）。スペクトルのもう一方の端において、プロセッサの製造業者は効率を求めてキャッシュコヒーレンス、キャッシュラインのレベルだけを追跡することに決めることができ、それは一度に 1 つのプロセッサしかラインを排他的に所有できないからである（排他的、修正等）。中間粒度の一例として、CCP はキャッシュラインごとのキャッシュコヒーレンス並びに現在のキャッシュライン状態を有する処理ユニットへのインデックス（例えば 4 つの処理ユニットプロセッサでは 0、1、2、3）を追跡することができる。

【0037】

【0055】 実施形態は、プロセッサの共有キャッシュ102bを利用して、アプリケーション104c及び／又はオペレーティングシステムカーネル104bの実行のビット精度トレースを効率的に記録する。これらの実施形態は、プロセッサ102（共有キャッシュ102bを含む）が半閉鎖系又は準閉鎖系を形成するという所見に基づいて成り立つ。例えばプロセスのためのデータの一部（即ちコードデータ及びランタイムアプリケーションデータ）が共有キャッシュ102b内にロードされると、プロセッサ102は連続した時間にわたって半閉鎖系又は準閉鎖系として如何なる入力もなしにそれ自体で実行し得る。具体的には、処理ユニット102aの1つ又は複数が、共有キャッシュ102bのデータ部分内に記憶されるランタイムデータを使用し、及びレジスタ102dを使用し、共有キャッシュ102bのコード部分からの命令を実行する。

10

【0038】

【0056】 処理ユニット102aが（例えば実行中の、実行する、又は実行する可能性がある命令が共有キャッシュ102b内にまだないコード又はランタイムデータにアクセスすることを理由に）情報の幾らかの流入を必要とする場合、「キャッシュミス」が発生し、その情報がシステムメモリ103から共有キャッシュ102b内に入れられる。例えば、実行された命令がアプリケーションランタイムデータ103b内のメモリアドレスにおいてメモリ操作を実行するときにデータキャッシュミスが発生する場合、そのメモリアドレスのデータが共有キャッシュ102bのデータ部分のキャッシュラインの1つに入れられる。同様に、命令がシステムメモリ103内に記憶されるアプリケーションコード103a内のメモリアドレスにおいてメモリ操作を実行するときにコードキャッシュミスが発生する場合、そのメモリアドレスのコードが共有キャッシュ102bのコード部分のキャッシュラインの1つに入れられる。その後、処理ユニット102aは（例えば別のキャッシュミス又は非キャッシュ読み出しによって）新たな情報が共有キャッシュ102b内に再び入れられるまで共有キャッシュ102b内の新たな情報を使用して実行を継続する。

20

【0039】

【0057】 本発明者は、アプリケーションの実行のビット精度表現を記録するために、トレース104aが、そのアプリケーションのスレッドの実行中の共有キャッシュ102b内への情報の流入を再現できるのに十分なデータを記録できることを確認した。これを行う第1の手法は、全てのキャッシュミス及び非キャッシュ読み出し（即ちハードウェアコンポーネント及びキャッシュ不能メモリからの読み出し）を各データ片が共有キャッシュ102b内に入れられた実行中の時間と共に（例えば実行される命令のカウント又は他の何らかのカウンタを使用する）ログすることにより、共有キャッシュ102b内に入れられるデータの全てを記録することである。

30

【0040】

【0058】 第1の手法よりもはるかに小さいトレースファイルをもたらす第2の手法は、各処理ユニット102aによって「消費」されたキャッシュラインを追跡し記録することである。本明細書で使用する時、処理ユニットがキャッシュラインの現在値を知っている場合、処理ユニットはキャッシュラインを「消費」している。これは、処理ユニットがキャッシュラインの現在値を書いたものだからであり、又は処理ユニットがキャッシュライン上で読み出しを行ったからである。この第2の手法は、キャッシュラインを消費した1つ又は複数の処理ユニット102aをキャッシュラインごとにプロセッサ102が識別することを可能にする共有キャッシュ102bへの拡張を含む。

40

【0041】

【0059】 本明細書の実施形態によれば、第3の手法は、プロセッサのCCPを利用してファイル104d内に記録するための、及び共有キャッシュ102bのアクティビティを依然として再現できるようにする「消費」されたキャッシュラインのサブセットを決定することである。この第3の手法は、第1の手法及び第2の手法の両方よりも著しく小さいトレースファイル、従って著しく低いトレースオーバーヘッドをもたらす。

【0042】

50

【0060】 本明細書の一部の実施形態は、処理ユニット／スレッドに対応するトレースデータストリームを記録する。例えばトレースファイル104は、処理ユニットごとに1つ又は複数の別個のトレースデータストリームを含むことができる。これらの実施形態では、各トレースデータストリーム内のデータパケットは、データパケットが適用される処理ユニットの識別子を欠いてもよく、なぜならその情報はトレースデータストリーム自体に基づいて固有だからである。これらの実施形態では、コンピュータシステム101が複数の（即ち異なるプロセッサソケット内の）プロセッサ102を含む場合、トレースファイルは、異なるプロセッサ102内の処理ユニット102aごとに1つ又は複数の異なるトレースデータストリームを有することができる。単一のスレッドに対して複数のデータストリームを使用することさえできる。例えば一部の実施形態は、スレッドによって使用される処理ユニットに1つのデータストリームを関連付けることができ、スレッドによって使用される各共有キャッシュに1つ又は複数の追加のデータストリームを関連付けることができる。

10

【0043】

【0061】 他の実施形態では、トレースファイル104がプロセッサ102のための単一のトレースデータストリームを含むことができ、データパケットがどの処理ユニットに適用されるのかを各データパケット内で識別することができる。これらの実施形態では、コンピュータシステム101が複数のプロセッサ102を含む場合、トレースファイル104は複数のプロセッサ102のそれぞれについて別個のトレースデータストリームを含むことができる。トレースファイルのレイアウトにかかわらず、処理ユニット102aごとのデータパケットは概して他の処理ユニットと独立に記録され、異なる処理ユニット102aにおいて実行した異なるスレッドが独立に再実行されることを可能にする。しかしトレースファイルは、異なるスレッド間の部分的な順序付けを提供する（明確な又は固有の）何らかの情報を含むことができる。

20

【0044】

【0062】 図3は、CCPデータを用いたキャッシュベーストレース記録を実行するための方法300の流れ図を示す。方法300は、トレーサ104aがアプリケーション104c及び／又はオペレーティングシステムカーネル104bをトレースするときプロセッサ102によって実行される動作を含み得る。プロセッサ102によって行われる動作は、プロセッサ102内のハードコードされたロジック、ソフトコードされたロジック（即ちマイクロコード102c）、及び／又はトレーサ104a、オペレーティングシステムカーネル104b、又はハイパーバイザ等の別のソフトウェアアプリケーションに基づき得る。図3は一連の動作を示すが、実施形態はこれらの動作の多くを任意の順序で、一部を並列に実行することさえ伴って実行できることが理解されよう。そのため方法300内で示す動作の順序は非限定的である。

30

【0045】

【0063】 図示のように、方法300はキャッシュと補助記憶との間の相互作用を検出する動作301を含む。一部の実施形態では、動作301は、複数のキャッシュラインの特定のキャッシュラインと1つ又は複数の補助記憶との間の相互作用を引き起こす操作を検出することを含む。例えば処理ユニット102aの1つにおいてアプリケーション104c又はオペレーティングシステムカーネル104bのスレッドを実行する間、処理ユニットは共有キャッシュ102b内のラインと補助記憶（例えばシステムメモリ103又は別のキャッシュ）との間の相互作用を引き起こすことができる。検出は、例えば自らのマイクロコード102cを実行することに基づいてプロセッサ102によって実行され得る。

40

【0046】

【0064】 方法300は、相互作用を引き起こした処理ユニットを識別する動作302も含む。一部の実施形態では、動作302は、操作を引き起こした複数の処理ユニットの特定の処理ユニットを識別することを含む。例えばマイクロコード102cを実行することに基づき、プロセッサ102は処理ユニット102aのどれが動作301内で検出された操作を引き起こしたのかを識別することができる。

50

【0047】

【0065】 方法300は、処理ユニットについてロギングが有効にされているかどうかを判定する動作303も含む。一部の実施形態では、動作303は、1つ又は複数のロギング制御ビットを使用して特定の処理ユニットについてロギングが有効にされていると判定することを含む。例えばプロセッサ102は、1つ又は複数のロギング制御ビットに基づき、動作302内で識別された処理ユニットのロギングが有効にされているかどうかを判定することができる。ロギング制御ビットのロギングを使用することは、様々な処理ユニットのロギングが動的に有効に及び無効にされ得ることを可能にする。従ってロギング制御ビットを使用することにより、トレーサ104aはどのスレッドがトレースされているのか及び／又は異なるスレッドの実行のどの部分がトレースされているのかを動的に制御することができる。

10

【0048】

【0066】 ロギング制御ビットの特定の形式及び機能は異なり得る。一部の実施形態では、例えばロギング制御ビットが制御レジスタ等のレジスタ102dのうちの1つの一部である。これらの実施形態では、単一のロギング制御ビットが1つの処理ユニット102a又は複数の処理ユニット102aに対応することができる。従ってレジスタ102dは、（例えば全ての処理ユニット、又は特定の処理ユニット、又は処理ユニットのサブセットに対応する）単一のロギング制御ビットを含み、又はことによると（例えば1つ又は複数の処理ユニットにそれぞれ対応する）複数のロギング制御ビットを含むことができる。他の実施形態では、ロギング制御ビットが、キャッシュと補助記憶との間の相互作用を引き起こした命令に対応するアドレス空間識別子（ASID）及び／又はプロセスコンテキスト識別子（PCID）を含み、又はそれらに関連する。従って、例えば方法300は、1つ又は複数の特定のASID／PCIDに関連する命令を実行している場合にのみ処理ユニットをトレースすることができる。このようにして、方法300は指定のアドレス空間及び／又は特定のプロセスコンテキストだけを記録することができる。組み合わせも可能である。例えばロギング制御ビットはレジスタ102dの1つ又は複数の中に記憶できるが、現在のASID／PCID値に基づいて設定／クリアされ得る。ロギング制御ビットの形式にかかわらず、一部の実施形態はコンテキストスイッチにおいてロギング制御ビットを設定／クリアできる場合があり、方法300が特定のスレッドのみをトレースすることを可能にする。

20

【0049】

【0067】 方法300は、キャッシュラインがロギングに関与するかどうかを判定する動作304も含む。一部の実施形態では、動作304は、特定の処理ユニットについてロギングが有効にされていることに少なくとも基づき、特定のキャッシュラインがロギングのパーティシパントかどうかを判定することを含む。例えばプロセッサ102は、動作301内で検出された操作に関与するキャッシュラインがロギングに関与するかどうかを判定することができる。後でより詳細に論じるように、共有キャッシュ102b内のビットを使用すること又はキャッシュウェイロックを使用すること等、検出に使用可能な幾つかのメカニズムがある。

30

【0050】

【0068】 方法300は、CCPを使用してトレースにログすべきデータがあることを識別する動作305も含む。例えばプロセッサ102は自らのCCPを調べて、操作の結果としてキャッシュ状態のどの遷移が生じたのか、及びそれらの遷移がロギングデータを保証するかどうかを明らかにすることができる。CCPを使用してトレースデータを識別する詳細例を図6A～図9Bに関連して後述する。

40

【0051】

【0069】 方法300は、CCPを用いて適切なデータをトレースにログする動作306も含む。一部の実施形態では、動作306は、操作を再実行するために使用可能なデータをトレースにログさせることを含む。トレースファイルにデータをログすべき場合、特定の処理ユニットに対応するトレースデータストリーム又はプロセッサ102に概して対応

50

するトレースデータストリーム等の適切なトレースデータストリームに1つ又は複数のデータパケットを追加することができる。適切なトレースデータストリームがプロセッサ102に概して対応する場合、1つ又は複数のデータパケットが特定の処理ユニットを識別し得る。トレースデータストリームがプロセッサ102に概して対応する場合、データストリーム自体の中のデータパケットの固有の順序が、複数のデータストリームが使用される場合は入手不能であり得る幾つかの追加の順序付け情報を提供することに留意されたい。
【0052】

【0070】 共有キャッシュ102bが複数のキャッシュレベルを含む場合、一部の実施形態では方法300がシステムメモリ103と相互作用するキャッシュレベルにおいて動作し、それはキャッシュミス処理するのはそのキャッシュレベルだからであることを指摘しておく。このレベルで操作することは、各処理ユニット102aのキャッシュアクティビティが冗長である（即ちユニットのアクティビティを複数回表す）ことなしに表されることを可能にする。従って例えばコンピュータシステム101が2つのプロセッサ102（即ち2つのプロセッサソケット）を含み、ソケットごとに1つの「包含的」L3キャッシュ、並びにL3キャッシュの下に「包含的」L2キャッシュを含む場合、一部の実施形態では方法300がL3キャッシュ上で動作する。方法300は複数のキャッシュレベルで操作することもできる。例えばコンピュータシステム101が1つのプロセッサ102（即ち1つのプロセッサソケット）を含み、そのソケットのための1つの「排他的」L3キャッシュ、並びにL3キャッシュの下に「包含的」L2キャッシュを含む場合、方法300が操作し得るのはL3キャッシュ及びL2キャッシュの両方である。混合された包含的／排他的挙動を示すキャッシュ内のロギングの更なる例を以下で論じる。

【0053】

【0071】 動作304に関連して上記で述べたように、キャッシュラインが「ロギングのパーティシパント」かどうかを判定するためにプロセッサ102によって使用され得る幾つかのメカニズムがある。1つは、フラグとして、処理ユニット識別子として、又はプロセッサインデックスとして使用可能な1つ又は複数の追加の「アカウンティングビット」を用いて共有キャッシュ102bの各ラインを拡張することである。それらの「アカウンティングビット」を制御するためのロジックはプロセッサのマイクロコード102cの一部であり得る。

【0054】

【0072】 この実施形態を説明するために、図4Aは、1つ又は複数の追加のアカウンティングビット401を用いてそのキャッシュライン404のそれぞれを拡張する、図2の共有キャッシュ200と同様の共有キャッシュ400aの一例を示す。従って各キャッシュライン404は、アカウンティングビット401、従来のアドレスビット402、及び値ビット403を含む。

【0055】

【0073】 一部の実装形態では、各キャッシュラインのアカウンティングビット401は、キャッシュラインがトレースロギングに関与しているかどうかを示すためにプロセッサ102によって使用されるフラグ（即ちオン又はオフ）として機能する単一ビットを含む。プロセッサのCCPが十分な粒度を有する場合（例えばCCPがキャッシュラインごとのコヒーレンス状態を各処理ユニットに関係するものとして、又はキャッシュラインのコヒーレンス状態を所有する処理ユニットへのインデックスに関連して追跡する場合）、この単一ビットはロバストで完全に決定論的なトレース（即ちトレースされる実行の完全な再構築性を保証するもの）の記録を助けるのに十分であり得る。

【0056】

【0074】 他の実装形態では、各ラインのアカウンティングビット401が複数のビットを含む。複数のビットは幾つかのやり方で使用することができる。本明細書では「ユニットビット」と呼ぶ1つの手法を使用し、各キャッシュラインのアカウンティングビット401は、プロセッサ102の処理ユニット102aの数（例えばプロセッサ102がハイパースレッディングをサポートする場合は論理処理ユニットの数、又はハイパースレデ

10

20

30

40

50

イングがサポートされない場合は物理処理ユニットの数)に等しい幾つかのユニットビットを含むことができる。これらのユニットビットは、どの1つ又は複数の特定の処理ユニットがキャッシュラインを消費したのかを追跡する(又はキャッシュラインが消費されていない場合、処理ユニットのどれもキャッシュラインを消費していないことを表す)ためにプロセッサ102によって使用され得る。従って、例えば2つの処理ユニット102aによって共有される共有キャッシュ102bはキャッシュラインごとに2つのユニットビットを含むことができる。各キャッシュラインに追加されるこれらのユニットビットに関連し、実施形態はプロセッサのマイクロコード102cを拡張してそれらのユニットビットを利用し、各処理ユニットの代わりにキャッシュライン内の現在値が(即ちトレースファイル104d内に)ログされているかどうか、或いは処理ユニットに知られているかどうかを追跡する。プロセッサのCCPがより粗い粒度を有する場合(例えばCCPがキャッシュラインのレベルでのみコヒーレンス状態を追跡する場合)、これらのユニットビットはロバストなトレースを助けるための追加情報を提供し得る。例えばキャッシュラインがCCPによって共有又は排他的として印付けされる場合、ユニットビットを使用してどの処理ユニットがキャッシュラインを共有するのか、又はどの処理ユニットが排他性を有するのかを識別することができる。

【0057】

【0075】 本明細書では「インデックスビット」と呼ぶ別の手法を使用し、各キャッシュラインのアカウンティングビット401は、「予約済みの」値(例えば-1)と共に、ロギングに参与するコンピュータシステム101のプロセッサ102の処理ユニット102aのそれぞれへのインデックスを表すのに十分な幾つかのインデックスビットを含むことができる。例えばコンピュータシステム101のプロセッサ102が128個の処理ユニット102aを含む場合、これらの処理ユニットはキャッシュラインごとに7つのインデックスビットだけを使用するインデックス値(例えば0~127)によって識別することができる。一部の実施形態では、どのプロセッサもキャッシュラインをログしていないことを示すために、1つのインデックス値が予約される(例えば「無効」である)。従ってこれは、7つのインデックスビットが実際に127個の処理ユニット102aと予約済みの値とを表すことができることを意味する。例えば、2進値0000000~11111110はインデックス位置0~126(10進)に対応することができ、2進値11111111(例えば解釈にもよるが-1又は127、10進)は対応するキャッシュラインをどのプロセッサもログしていないことを示すための「無効」に対応し得るが、この表記は実装形態に応じて異なり得る。従ってユニットビットは、キャッシュラインがトレースロギングに参与しているかどうかを追跡するために(例えば-1以外の値)、及びキャッシュラインを消費した特定の処理ユニット(例えばキャッシュラインを直近に消費した処理ユニット)へのインデックスとしてプロセッサ102によって使用され得る。この第2の手法は、共有キャッシュ102b内のオーバーヘッドを殆ど伴わずに多数の処理ユニットをサポートする利点を有し、第1の手法よりも粒度が低い欠点を有する(即ち一度に1つの処理ユニットだけ識別される)。この場合も、プロセッサのCCPがより粗い粒度を有する場合(例えばCCPがキャッシュラインのレベルでのみコヒーレンス状態を追跡する場合)、これらのインデックスビットはロバストなトレースを助けるための追加情報を提供し得る。例えばキャッシュラインがCCPによって共有又は排他的として印付けされる場合、インデックスビットを使用してキャッシュラインを共有する少なくとも1つの処理ユニットを識別し、又はどの処理ユニットが排他性を有するのかを識別することができる。

【0058】

【0076】 キャッシュラインがロギングのパーティシパントかどうかを判定するためにプロセッサ102によって使用され得る別のメカニズムは、追加のアカウンティングビット401を用いて各キャッシュラインを拡張することなしに、図4Aに関連して論じた概念を使用することができる。代わりにこのメカニズムは、アカウンティングビットを記憶するためにキャッシュライン404の1つ又は複数を予約する。図4Bは、メモリアドレス402及び値403を記憶する従来のキャッシュライン405、並びに従来のキャッシュ

10

20

30

40

50

ライン405に適用されるアカウンティングビットを記憶するための1つ又は複数の予約済みキャッシュライン406を含む共有キャッシュ400bの一例を示す。予約済みキャッシュライン406のビットは、従来のキャッシュライン405の異なるものにそれぞれ対応するアカウンティングビットの様々なグループ内に割り当てられる。アカウンティングビットのこれらのグループは、実装形態に応じてフラグビット、ユニットビット、又はインデックスビットとして機能することができる。

【0059】

【0077】 キャッシュラインがロギングのパーティシパントかどうかを判定するためにプロセッサ102によって使用され得る別のメカニズムは、アソシアティブキャッシュ及びウェイロックを利用することである。プロセッサの共有キャッシュ102bは概してシステムメモリ103よりもはるかに（しばしば桁違いに）小さく、そのため共有キャッシュ102b内にラインがあるよりもはるかに多くの記憶域がシステムメモリ103内には通常ある。そのため各プロセッサは、システムメモリの複数の記憶域をキャッシュ内のラインにマップするためのメカニズムを定める。プロセッサは、直接マッピング及びアソシアティブマッピングという2つの一般的な技法のうちの1つを概して使用する。直接マッピングを使用し、システムメモリ103内の異なる記憶域がキャッシュ内の1つのラインだけにマップされ、そのため各記憶域はキャッシュ内の特定のラインにのみキャッシュされ得る。

【0060】

【0078】 他方でアソシアティブマッピングを使用し、システムメモリ103内の異なる位置を共有キャッシュ102b内の複数のラインの1つにキャッシュすることができる。図5は、アソシアティブキャッシュマッピングの例500を示す。ここでは、キャッシュ502のキャッシュライン504が、2つのキャッシュライン504a及び504bの第1のグループ（インデックス0として識別する）、並びに2つのキャッシュライン504c及び504dの第2のアドレスグループ（インデックス1として識別する）を含む、2つのキャッシュラインごとの異なるアドレスグループに論理的に分割される。アドレスグループ内の各キャッシュラインは異なる「ウェイ」に関連し、そのためキャッシュライン504aはインデックス0、ウェイ0によって識別され、キャッシュライン504bはインデックス0、ウェイ1等によって識別される。更に図示のように、記憶域503a、503c、503e、及び503g（メモリインデックス0、2、4、及び6）がインデックス0にマップされる。そのため、システムメモリ内のこれらの位置のそれぞれは、インデックス0のグループ内の任意のキャッシュライン（即ちキャッシュライン504a及び504b）にキャッシュすることができる。図示のマッピングの特定のパターンは例示及び概念上の目的に過ぎず、メモリインデックスをキャッシュラインにマップすることができる唯一のやり方として解釈すべきではない。

【0061】

【0079】 アソシアティブキャッシュは、概してNウェイアソシアティブキャッシュであると言われ、Nは各アドレスグループ内の「ウェイ」の数である。従って、図5のキャッシュ500は2ウェイアソシアティブキャッシュと呼ぶことができる。プロセッサはNウェイキャッシュを一般に実装し、Nは2の累乗（例えば2、4、8等）であり、4及び8のN値が一般に選択される（但し本明細書の実施形態は或る特定のN値又はN値のサブセットに限定されない）。とりわけ各アドレスグループが1つのキャッシュラインしか含まないので、1ウェイアソシアティブキャッシュは概して直接対応キャッシュと等価である。加えてNがキャッシュ内のライン数に等しい場合、キャッシュ内の全てのラインを含む単一のアドレスグループを含むので、それはフルアソシアティブキャッシュと呼ばれる。フルアソシアティブキャッシュでは、任意の記憶域をキャッシュ内の任意のラインにキャッシュすることができる。

【0062】

【0080】 図5は、全般的な原理を示すためにシステムメモリ及びキャッシュの簡略化した図を表すことを指摘しておく。例えば、図5は個々の記憶域をキャッシュラインにマッ

10

20

30

40

50

プするが、キャッシュ内の各ラインはシステムメモリ内のアドレス指定可能な複数の位置に関係するデータを概して記憶することが理解されよう。従って図5では、システムメモリ(501)内の各位置(503a~503h)はアドレス指定可能な複数の記憶域を実際に表すことができる。加えてマッピングは、システムメモリ501内の実際の物理アドレスとキャッシュ502内のラインとの間とすることができ、又は仮想アドレスの中間層を使用することができる。

【0063】

[0081] ウェイロックを使用することにより、キャッシュラインがロギングのパーティシパントかどうかを判定するためにアソシアティブキャッシュを使用することができる。ウェイロックは、何らかの目的のためにキャッシュ内の特定のウェイをロックし又は予約する。具体的には、本明細書の実施形態はトレースされているスレッドのための1つ又は複数のウェイを予約するためのウェイロックを利用し、そのためそのスレッドの実行に関係するキャッシュミス記憶のためにロック／予約されたウェイが排他的に使用される。従って再び図5を参照し、トレースされたスレッドに関して「ウェイ0」がロックされた場合、キャッシュライン504a及び504c(即ちインデックス0、ウェイ0、及びインデックス1、ウェイ0)はそのスレッドの実行に関係するキャッシュミスのために排他的に使用され、残りのキャッシュラインは他の全てのキャッシュミスのために使用される。従って、特定のキャッシュラインがロギングのパーティシパントかどうかを判定するために、プロセッサ102はキャッシュラインがトレースされているスレッドのために予約されているウェイの一部かどうかを判定するだけでよい。

【0064】

[0082] 図6A~図6Dは、図1、図2、図4A、図4B、及び図5の脈絡で、図3の方法300の適用の具体例600を示す。図6Aは、共有キャッシュ102b内の単一ライン上の4つの処理ユニット102a(即ちP0~P3)による読み出し及び書き込みアクティビティを示す第1の表600aを示す。図6Bは、それらの読み出し及び書き込みに基づく、追跡されたキャッシュコヒーレンス状態(例えばプロセッサのCCPを使用して追跡される)の一実施形態を示す第2の表600bを示す。図6Cは、仮にアカウントティングビットが使用される場合、(図4A及び図4Bに関連して説明した)共有キャッシュ102bのアカウントティングビット内に記憶され得る内容を示す第3の表600cを示す。典型的には1種類のアカウンティングビット(即ちラインごとのユニットビット、ラインごとのインデックスビット、又はラインごとのフラグビット)のみが使用されるが、説明の完全性のために、表600cはユニットビット603、インデックスビット604、及びフラグビット605のそれぞれを示す。最後に図6Dは、各操作に関連してトレースファイル104dに潜在的に書き込まれ得るログデータ606の種類の例を示す第4の表600dを示す。

【0065】

[0083] 説明を簡単にするために、表600aは一度に単一の処理ユニット102aだけによる操作を示すが、本明細書に記載の原理は同時アクティビティ(例えば同じキャッシュラインの2つ以上の処理ユニットによる同時読み出し)がある状況にも適用されることが理解されよう。加えて図6A~図6Dに関連して説明する例は、追跡が処理ユニットP0~P2について有効であり、処理ユニットP3については無効だと仮定する。例えば上記で論じたように、これは各処理ユニットに対応する制御されたaビットとすることができ、制御レジスタのかかる複数のビット。

【0066】

[0084] 最初に説明を容易にするために、この例は、上記で論じたCCP(即ちMSI、MESI、及びMOESI)で使用されるキャッシュライン状態(即ち修正、所有、排他的、共有、及び無効)から導出される単純化したキャッシュライン状態を使用する。この単純化において、これらの状態は「読み出し」状態(即ちキャッシュラインが読み出されている)又は「書き込み」状態(即ちキャッシュラインが書き込まれている)の何れかにマップする。以下の表1はこれらのマッピングの一例を示す。これらのマッピングは一

例として使用するに過ぎず、非限定的であることに留意されたい。例えば本明細書で論じる以外のCCP及び状態が存在することができ、当業者は本明細書の開示に鑑みて多くの異なるCCPを用いて同様のマッピングを行えることを理解されよう。

【0067】

【表1】

プロトコル状態	マップされた状態
修正	書き込み
所有	読み出し
排他的	書き込み
共有	読み出し
無効	マッピングなし-キャッシュラインは空だと見なされる

表1

【0068】

【0085】 とりわけ実施形態は、どのデータがプロセッサ102から入手できるのかに応じて及び／又は実装の選択に基づいて、様々なレベルでCCPデータをログすることができる。例えばCCPデータは、(表1に示すような)「マップされた」CCP状態に基づいて、プロセッサ102によって可視化された実際のCCP状態(例えば修正、所有、排他的、共有、及び／又は無効)に基づいて、及び／又は典型的にはプロセッサ102によって可視化できない低レベル「生」CCPデータに基づいてさえログすることができる。

【0069】

【0086】 図6A～図6Dに移り、表600aは、操作間のグローバル順序を指定するために使用される識別子(ID)を示す第1の列601を含む。表600aは、処理ユニットの1つにそれぞれ対応する4つの追加の列602a～602dも含む。簡潔にするためにこの例はグローバルIDを使用するが、実際には各処理ユニットは通常自らの独立した識別子の組を使用して操作を順序付けることが理解されよう。それらのIDは命令カウンタ(IC)、又は「ジャンプカウンタ」+プログラムカウンタ等、操作間の順序付けを指定するための他の任意の適切なメカニズムを含み得る。この例はMSI、MESI、及びMOESI CCPと整合するやり方でメモリを使用するが、簡潔にするために「修正」状態、「共有」状態、及び「無効」状態だけを使用することに留意されたい。但し一部のCCPは、トレースエントリを強く順序付けるためにトレース内に(例えば全てのパケット内に又は不定期のパケット内に)記録することもできる独自の固有の及び／又は単調増分するIDを提供できることを指摘しておく。CCPがかかるIDを提供しなくても、ソケットタイマの値(例えばTSC)又は順序付け可能な別のIDをことによると使用することができる。

【0070】

【0087】 表600aに示すように、識別子ID[0]において処理ユニットP0が読み出しを行い、その読み出しはデータDATA[1]をキャッシュライン内に運ぶキャッシュミスを引き起こす。これに対応し、表600bはキャッシュラインがP0によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表600cは、ユニットビット603が使用される場合、処理ユニットP0がキャッシュラインを消費した(即ち読み出した)こと(及び処理ユニットP1～P3が消費していないこと)をユニットビット603が示すこと、インデックスビット604が使用される場合、P0がキャッシュラインを消費したことをインデックスビット604が示すこと、及びフラグビット605が使用される場合、何らかの処理ユニットがキャッシュラインを消費したことをフ

ラグビット 6 0 5 が示すことを示す。この状態を所与とし、動作 3 0 3 で、プロセッサ 1 0 2 が P 0 についてロギングが有効にされていると判定し、動作 3 0 4 で、（即ちユニットビット 6 0 3、インデックスビット 6 0 4、フラグビット 6 0 5、又はウェイロックを使用し）キャッシュラインがロギングに関与すると判定する。従って動作 3 0 6 で、プロセッサ 1 0 2 は C C P を利用して必要に応じてトレースファイルに適切なデータをログする。ここでは、キャッシュラインが無効（空）状態から読み出し（表 6 0 0 a）／共有（表 6 0 0 b）状態に移行しているのでデータをログすべきである。表 6 0 0 d のログデータ 6 0 6 に示すように、プロセッサ 1 0 2 は、必要に応じて（即ちデータパケットが処理ユニットごとに別個のデータストリームにログされているか、又は単一のデータストリームにログされているかに応じて）処理ユニット（P 0）、キャッシュラインアドレス（@）、命令カウント又は他の何らかのカウント、及びキャッシュラインに入れられたデータ（DATA [1]）を示すことができる。上記で論じたように、命令カウントは典型的には処理ユニットに固有の値だが、簡単にするために表 6 0 0 d は対応するグローバル ID（即ちこの例では IC [0]）に関して命令カウントを参照する。

【0071】

【0088】 一部の実施形態では、キャッシュラインアドレス（@）及びデータ（例えば DATA [1]）は、トレースファイル 1 0 4 d 内で圧縮され得ることを指摘しておく。例えばメモリアドレスは、前に記録されたメモリアドレス内の「高」ビットを（明示的に又は暗黙的に）参照することにより、メモリアドレスの「高」ビットを記録するのを控えることによって圧縮することができる。データは、複数のビットをそれぞれ含む複数のグループにデータ値のビットをグループ化し、各グループを対応する「フラグ」ビットに関連付けることによって圧縮することができる。グループが特定のパターン（例えば全て 0、全て 1 等）に等しい場合、フラグビットを設定することができ、そのビットのグループはトレース内に記憶する必要がない。

【0072】

【0089】 次に表 6 0 0 a は、ID [1] において処理ユニット P 1 がキャッシュライン上の読み出しを行い、データ DATA [1] を読み出すことを示している。表 6 0 0 b はキャッシュラインが P 0 及び P 1 によって現在「共有」されていることをプロセッサの C C P が示していることを示す。表 6 0 0 c は、処理ユニット P 0 及び P 1 がキャッシュラインを消費したこと（ユニットビット 6 0 3）、P 1 がキャッシュラインを消費したこと（インデックスビット 6 0 4）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット 6 0 5）を示す。インデックスビット 6 0 4 が P 1 の代わりに P 0 を引き続き参照することも正しいことに留意されたい。表 6 0 0 d は、C C P を使用し、操作の記録をログすべきだとプロセッサ 1 0 2 が判定することを示す。図示のようにプロセッサ 1 0 2 は、処理ユニット（P 1）、キャッシュラインアドレス（@）、命令カウント（IC [1]）、キャッシュラインが読み出し（共有）状態から読み出し（共有）状態に移行したこと、及び P 0 が前のキャッシュラインへの前のアクセスを有していたが、現在は P 0 及び P 1 がアクセスを有することを示すことができる。

【0073】

【0090】 次に表 6 0 0 a は、ID [2] において処理ユニット P 0 がキャッシュラインへの書き込みを行い、データ DATA [2] を書き込むことを示している。表 6 0 0 b はキャッシュラインが P 0 によって現在「修正」され、P 1 について「無効」であることをプロセッサの C C P が示していることを示す。表 6 0 0 c は、処理ユニット P 0 だけがキャッシュラインを消費した（即ちその値を更新した）こと（ユニットビット 6 0 3）、P 0 がキャッシュラインを消費したこと（インデックスビット 6 0 4）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット 6 0 5）を示す。キャッシュラインが書き込まれている／修正されているので、表 6 0 0 d は、C C P を使用し、操作の記録をログする必要があるとプロセッサ 1 0 2 が判定することを示す。図示のようにプロセッサ 1 0 2 は、処理ユニット（P 0）、キャッシュラインアドレス（@）、命令カウント（IC [2]）、キャッシュラインが読み出し（共有）状態から書き込み（修正）状

態に移行したこと、並びに P 0 及び P 1 が前のキャッシュラインへの前のアクセスを有していたが、現在は P 0 だけがアクセスを有することを示すことができる。

【0074】

[0091] 表 6 0 0 b に示す C C P 状態を使用し、どの処理ユニットがキャッシュラインへの前のアクセスを有していたかに関する情報を知ることができることに留意されたい。しかし、一部の C C P はそれを行うのに十分な情報を保たない場合があることを指摘しておく（例えばキャッシュラインレベルでのみコヒーレンス状態を追跡する C C P）。或いはユニットビット 6 0 3 が使用される場合、この情報はユニットビットから導出することができる。従って図 6 D に示すログデータ 6 0 6 は、この情報を保つロバスタな C C P 又はユニットビット 6 0 3 の使用を仮定する。

【0075】

[0092] これらの何れかが使用されない場合（例えば C C P がさほどロバスタではない場合、及びユニットビット 6 0 3 の代わりにインデックスビット 6 0 4、フラグビット 6 0 5、又はウェイロックが使用される場合）、ログデータ 6 0 6 はより徹底的でなく又はより大きい可能性がある。第 1 の例として、C C P がキャッシュラインレベルでのみコヒーレンス状態を追跡し、インデックスビット 6 0 4 が使用される場合、その 2 つを使用して、キャッシュライン状態が（全ての処理ユニットについて）無効であること、（修正した処理ユニットのインデックスと共に）修正されていること、（排他的にした処理ユニットのインデックスと共に）排他的であること、又は共有されている（及び全ての処理ユニットがアクセスを有することができる）ことを識別することができる。このことはより単純なハードウェア実装をもたらすことができ、キャッシュラインを共有から修正又は排他的に変更するとき、キャッシュラインを共有するためにより粒度の細かい C C P によって知られるものだけでなく全ての処理ユニットに知らせる必要があるという不利点が伴う。第 2 の例として、インデックスビット 6 0 4 を使用してキャッシュラインにアクセスした最後の処理ユニットを識別することができる。キャッシュが包含的である場合（即ちそのため L 2 又は L 1 キャッシュレベルにおけるアクセスの背後に多くの読み出しが隠されている場合）、処理ユニットが同じキャッシュラインを読み出し続けていても、L 3 キャッシュは同じ処理ユニットから相対的に少ない繰り返し要求を認める可能性がある。読み出し→読み出しに関する全てのインデックス変化をログし、次いで読み出し→書き込み、書き込み→書き込み、及び書き込み→読み出しのログを有し、ことによると僅かに大きいトレースの代償を払ってインデックスもユニットビット 6 0 3 の使用と同じデータを与える。第 3 の例として、各キャッシュラインは単一のフラグビットを含み得るが、C C P はキャッシュラインのコヒーレンス状態を所有する処理ユニットへのインデックスに関連してキャッシュラインごとのコヒーレンス状態を追跡することができる。ここでトレースは、ユニットビットが使用される場合又は C C P が個々の処理ユニットを追跡する場合よりも多くのキャッシュラインの動きを記録し得るが、トレースは依然として完全に決定論的であり得る。各処理ユニットに関する情報を有する場合と、プロセッサインデックスに関する情報だけを有する場合とのトレースファイルサイズの大まかな比較を図 9 A 及び図 9 B に関して以下に示す。

【0076】

[0093] 図 6 A に戻り、表 6 0 0 a は I D [3] において処理ユニット P 1 がキャッシュラインからの読み出しを行い、データ D A T A [2] を読み出すことを示している。表 6 0 0 b はキャッシュラインが P 0 及び P 1 によって現在「共有」されていることをプロセッサの C C P が示していることを示す。表 6 0 0 c は、処理ユニット P 0 及び P 1 がキャッシュラインを消費したこと（ユニットビット 6 0 3）、P 1 がキャッシュラインを消費したこと（インデックスビット 6 0 4）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット 6 0 5）を示す。インデックスビット 6 0 4 が P 1 の代わりに P 0 を引き続き参照することも正しいことに留意されたい。キャッシュラインが書き込み（修正）状態から読み出し（共有）状態に移行しているので、表 6 0 0 d は、C C P を使用し、操作の記録をログする必要があるとプロセッサ 1 0 2 が判定することを示す

。図示のようにプロセッサ102は、処理ユニット(P1)、キャッシュラインアドレス(@)、命令カウント(IC[3])、キャッシュラインが書き込み(修正)状態から読み出し(共有)状態に移行したこと、及びP0が前のキャッシュラインへの前のアクセスを有していたが、現在はP0及びP1がアクセスを有することを示すことができる。

【0077】

【0094】次に表600aはID[4]において処理ユニットP0が再びキャッシュラインへの書き込みを行い、今回はデータDATA[3]を書き込むことを示している。表600bはキャッシュラインが再びP0によって「修正」され、P1について「無効」であることをプロセッサのCCPが示していることを示す。表600cは、処理ユニットP0だけがキャッシュラインを消費したこと(ユニットビット603)、P0がキャッシュラインを消費したこと(インデックスビット604)、又は何らかの処理ユニットがキャッシュラインを消費したこと(フラグビット605)を示す。キャッシュラインが書き込まれている／修正されているので、表600dはCCPを使用し、操作の記録をログする必要があるとプロセッサ102が判定することを示す。図示のようにプロセッサ102は、処理ユニット(P0)、キャッシュラインアドレス(@)、命令カウント(IC[4])、キャッシュラインが読み出し(共有)状態から書き込み(修正)状態に移行したこと、並びにP0及びP1が前のキャッシュラインへの前のアクセスを有していたが、現在はP0だけがアクセスを有することを示すことができる。

【0078】

【0095】次に表600aは、ID[5]において処理ユニットP2がキャッシュラインからの読み出しを行い、データDATA[3]を読み出すことを示している。表600bはキャッシュラインがP0及びP2によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表600cは、処理ユニットP0及びP2がキャッシュラインを消費したこと(ユニットビット603)、P2がキャッシュラインを消費したこと(インデックスビット604)、又は何らかの処理ユニットがキャッシュラインを消費したこと(フラグビット605)を示す。インデックスビット604がP2の代わりにP0を引き続き参照することも正しいことに留意されたい。キャッシュラインが書き込み(修正)状態から読み出し(共有)状態に移行しているため、表600dはCCPを使用し、操作の記録をログする必要があるとプロセッサ102が判定することを示す。図示のようにプロセッサ102は、処理ユニット(P2)、キャッシュラインアドレス(@)、命令カウント(IC[5])、キャッシュラインが書き込み(修正)状態から読み出し(共有)状態に移行したこと、及びP0が前のキャッシュラインへの前のアクセスを有していたが、現在はP0及びP2がアクセスを有することを示すことができる。

【0079】

【0096】次に表600aはID[6]において処理ユニットP1がキャッシュラインからの読み出しを行い、同じくデータDATA[3]を読み出すことを示している。表600bはキャッシュラインがP0、P1、及びP2によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表600cは、処理ユニットP0、P1、及びP2がキャッシュラインを消費したこと(ユニットビット603)、P1がキャッシュラインを消費したこと(インデックスビット604)、又は何らかの処理ユニットがキャッシュラインを消費したこと(フラグビット605)を示す。インデックスビット604がP1の代わりにP0又はP2を引き続き参照することも正しいことに留意されたい。表600dは、CCPを使用し、操作の記録をログすべきだとプロセッサ102が判定することを示す。図示のようにプロセッサ102は、処理ユニット(P1)、キャッシュラインアドレス(@)、命令カウント(IC[6])、キャッシュラインが読み出し(共有)状態から読み出し(共有)状態に移行したこと、並びにP0及びP2が前のキャッシュラインへの前のアクセスを有していたが、現在はP0、P1、及びP2がアクセスを有することを示すことができる。

【0080】

【0097】次に表600aはID[7]において処理ユニットP3がキャッシュラインが

らの読み出しを行い、同じくデータDATA [3]を読み出すことを示している。表600bはキャッシュラインがP0、P1、P2、及びP3によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表600cは、ユニットビット603、インデックスビット604、又はフラグビット605のどれも更新されていないことを示す。これはロギングがP3について無効にされており、トレーシングのために、P3が読み出しを行うことによってキャッシュラインを「消費」していないからである。表600dは、データがログされていないことを示す。これは動作303において、P3についてロギングを有効にしないとプロセッサ102が判定するからである。

【0081】

[0098] 次に表600aはID [8]において処理ユニットP3が書き込みを行う、キャッシュライン、データDATA [4]を書き込むことを示している。表600bはキャッシュラインがP0、P1、及びP2について現在「無効」であり、P3によって「修正」されることをプロセッサのCCPが示していることを示す。表600cは、任意の処理ユニットによって消費されないものとして、ユニットビット603、インデックスビット604、及びフラグビット605が全てキャッシュラインを反映することを示す。これはロギングがP3について無効にされており、そのためトレーシングのために、P3が書き込みを行ったときP3はキャッシュラインを「消費」せず、更に書き込みが他の処理ユニットのキャッシュライン内の値を無効にしたからである。表600dは、データがログされていないことを示す。この場合も、これは動作303においてP3についてロギングが有効にされないとプロセッサ102が判定するからである。

【0082】

[0099] 次に表600aはID [9]において処理ユニットP0がキャッシュラインへの書き込みを行い、データDATA [5]を書き込むことを示している。表600bはキャッシュラインがP0によって現在「修正」され、P3について「無効」であることをプロセッサのCCPが示していることを示す。表600cは、処理ユニットのどれもキャッシュラインを消費していないことを示す。それは表600d内に反映されているようにこの操作に関してログエントリが作成されなかったからである。書き込まれたデータは、P0のスレッドの命令の通常実行によって再現されるので、ログエントリを作成する必要がない。しかしこの状況では、トレースにエントリを任意選択的に書き込んで（即ちロギングが有効にされた処理ユニットによってログされていないキャッシュラインへの書き込み）トレースの消費側に追加のデータを提供することができる。この状況ではログエントリは、キャッシュライン値の読み出し+DATA [5]の書き込みとして扱うことができる。

【0083】

[00100] 次に表600aはID [10]において処理ユニットP2がキャッシュラインからの読み出しを行い、データDATA [5]を読み出すことを示している。表600bはキャッシュラインがP0及びP2によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表600cは、処理ユニットP2がキャッシュラインを消費したこと（ユニットビット603）、P2がキャッシュラインを消費したこと（インデックスビット604）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット605）を示す。値キャッシュラインが前にログされていない（即ちID [9]においてログされなかった）ので、表600dはCCPを使用し、操作の記録をログする必要があるとプロセッサ102が判定することを示す。図示のようにプロセッサ102は、処理ユニット（P2）、キャッシュラインアドレス（@）、命令カウント（IC [10]）、キャッシュラインに入れられたデータ（DATA [5]）、及びP2がキャッシュラインへのアクセスを有することを示すことができる。特定のCCP及びアカウンティングビットがどの情報を提供するのにもよるが、P0もキャッシュラインへのアクセスを有することをログすることも可能であり得る。

【0084】

[00101] 次に表600aはID [11]において処理ユニットP1がキャッシュラインからの読み出しを行い、同じくデータDATA [5]を読み出すことを示している。表

6 0 0 bはキャッシュラインがP 0、P 1、及びP 2によって現在「共有」されていることをプロセッサのCCPが示していることを示す。表6 0 0 cは、処理ユニットP 1及びP 2がキャッシュラインを消費したこと（ユニットビット6 0 3）、P 1がキャッシュラインを消費したこと（インデックスビット6 0 4）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット6 0 5）を示す。インデックスビット6 0 4がP 1の代わりにP 2を引き続き参照することも正しいことに留意されたい。表6 0 0 dは、CCPを使用し、操作の記録をログすべきだとプロセッサ1 0 2が判定することを示す。図示のようにプロセッサ1 0 2は、処理ユニット（P 1）、キャッシュラインアドレス（@）、命令カウント（IC [1 1]）、キャッシュラインが読み出し（共有）状態から読み出し（共有）状態に移行したこと、及びP 2が前のキャッシュラインへの前のアクセスを有していたが、現在はP 1及びP 2がアクセスを有することを示すことができる。ID [1 0]においてP 2によってログされたので、値（DATA [5]）はログする必要がないことに留意されたい。

【0 0 8 5】

[00102] 次に表6 0 0 aはID [1 2]において処理ユニットP 0がキャッシュラインからの読み出しを行い、同じくデータDATA [5]を読み出すことを示している。表6 0 0 bはキャッシュラインがP 0、P 1、及びP 2によって現在「共有」されていることをプロセッサのCCPが引き続き示していることを示す。表6 0 0 cは、処理ユニットP 0、P 1、及びP 2がキャッシュラインを消費したこと（ユニットビット6 0 3）、P 0がキャッシュラインを消費したこと（インデックスビット6 0 4）、又は何らかの処理ユニットがキャッシュラインを消費したこと（フラグビット6 0 5）を示す。インデックスビット6 0 4がP 0の代わりにP 1又はP 2を引き続き参照することも正しいことに留意されたい。表6 0 0 dは、CCPを使用し、操作の記録をログすべきだとプロセッサ1 0 2が判定し得ることを示す。この場合プロセッサ1 0 2は、処理ユニット（P 0）、キャッシュラインアドレス（@）、命令カウント（IC [1 2]）、キャッシュラインが読み出し（共有）状態から読み出し（共有）状態に移行したこと、並びにP 1及びP 2が前のキャッシュラインへの前のアクセスを有していたが、現在はP 0、P 1、及びP 2がアクセスを有することを示すことができる。P 2から入手できるので、値（DATA [5]）はログされない。

【0 0 8 6】

[00103] 或いは、P 0はキャッシュラインの値を既に有するので（即ちID [9]においてその値を書き込んだので）、プロセッサ1 0 2がID [1 2]でのみP 0を参照することが可能であり得る。ID [1 2]での如何なるロギングも控えることさえ可能な場合があり、なぜならば再実行時にヒューリスティックを使用して、P 0を参照する情報がトレース内になくても値（即ちDATA [5]）を回復できるからである。しかし、それらの技法は計算的に高価であり、再実行が「脱線」したときを検出するシステムの能力を低減し得る。ヒューリスティックの一例は、処理ユニット間のメモリアクセスが概して（CCPデータに基づいて）強く順序付けられていると認識することであり、そのため再実行は所与の記憶域についてそれらのユニット間の最後の値を使用することができる。

【0 0 8 7】

[00104] 次に表6 0 0 aは、ID [1 3]においてキャッシュラインが追い出されることを示している。その結果、表6 0 0 bはCCPエントリが空であることを示し、表6 0 0 cはキャッシュラインを消費したもののとしてアカウンティングビットがどの処理ユニットも反映していないことを示し、表6 0 0 dはデータがログされていないことを示す。

【0 0 8 8】

[00105] 完全性のために、ログデータ6 0 6は全ての終了アクセス状態（即ちどの処理ユニットがキャッシュラインへのアクセスを現在有するのか）を列挙するが、この情報は潜在的に暗黙的であり、これを省略することによってトレースファイルのサイズを低減できることに留意されたい。例えば書き込み→読み出しの遷移において、読み出し後にアクセスを有する処理ユニットのリストは常に、前にアクセスを有した処理ユニットと読

10

20

30

40

50

み出しを行った処理ユニットとである。読み出し→書き込みの遷移又は書き込み→書き込みの遷移では、書き込み後に書き込みアクセスを有する処理ユニットのリストは常に書き込みを行った処理ユニットである。読み出し→読み出しの遷移では、読み出し後にアクセスを有する処理ユニットのリストは常に、遷移前にアクセスを有した処理ユニットと読み出しを行った処理ユニットとである。

【0089】

【00106】 概して、完全に決定論的なトレースファイルを生成するために、CCPは処理ユニット間の（例えばP0からP1への）全ての遷移（即ち書き込み→読み出し、書き込み→書き込み、読み出し→書き込み、及び読み出し→読み出し）がログされることを指示する。但し、同じ処理ユニット内の（例えばP0からP0への）遷移はログする必要がない。かかる遷移をログする必要がないのは、その処理ユニットにおいて実行されたスレッドの通常実行によってかかる遷移が再現されるからである。

【0090】

【00107】 上記の例でログされるようなデータを使用し、及び記録が行われたプロセッサ102によって使用されるCCPについての更なる知識を用いて、各スレッドにおいて発生した操作の完全な順序付けを再構築することができ、異なる処理ユニット間の操作の少なくとも部分的な順序付けを再構築できることが理解されよう。従って、インデクシングプロセス及び／又はトレースファイルの再実行により、トレースファイル104d内に全て明示的に記録されていなくても、上記の操作のそれぞれを再構築することができる。

【0091】

【00108】 一部の実施形態では、処理ユニット間の操作の順序付けのロギングを向上させるために、トレーサ104aがトレースファイル104d内に追加のデータパケットを記録することができる。例えばトレーサ104aは、スレッドにわたる単調増分数(MIN)(又は他のカウンタ／タイマ)を有するイベントの完全な順序付けを提供するために、MIN(又は他の何らかのカウンタ／タイマ)等の何らかのイベント順序付け情報と共に記録することができる。これらのMINは、スレッド間で「順序付け可能」だと定められるイベントに対応するデータパケットを識別するために使用され得る。これらのイベントは、スレッドが共有メモリによってどのように相互作用し得るのか、及びメモリ内のデータの共有使用を定める「トレースメモリモデル」に基づいて定められ得る。別の例としてトレーサ104aは、定義済みの決定論的アルゴリズム及び定義済みの1組のレジスタ(例えばプログラムカウンタ、スタック、汎用レジスタ等)に基づき、プロセッサ状態のハッシュを(周期的に又は無作為に)記録することができる。更に別の例として、トレーサ104aはキャッシュラインデータを(周期的に又は無作為に)強制的にログすることができる。更に別の例として、トレーサ104aは、ハッシュをログする「遷移」パケットをトレース内に含めることができる、それらが暗黙的に運ぶデータの全て又は一部(例えば数ビット)。従ってこの暗黙データが再実行時に再構築されると、暗黙データの適切な部分をハッシュし、これらの遷移パケットにマッチしてその順序付けを識別するのを助けることができる。これは例えばキャッシュラインが共有状態にある場合、CCPがキャッシュラインに関連するプロセッサインデックスを追跡できない場合に有用であり得る。

【0092】

【00109】 順序付けを向上させるためにトレーサ104aがトレースファイル104d内に追加のデータパケットを記録する場合、処理ユニット間の遷移の一部の記録を省略することが可能であり得る。例えば、スレッド間の一部の読み出し→読み出し遷移の記録を省略することが可能であり得る。一部の読み出しの順序付けをトレース及びCCPに基づいて完全には再構築できない場合があるので、これは一部の状況では「脆弱化された」非決定論的トレースをもたらす可能性があるが、トレースの再実行を「脱線」させない読み出しの有効な順序付けを見つけるために、追加の順序付け情報(例えばMIN、プロセッサ状態のハッシュ、追加のキャッシュラインデータ)が再実行中の探索空間を減らすのを助けることができる。スレッド間の読み出し→読み出し遷移の一部を省略することの利点は、トレーシングを容易にするためのトレースサイズ及びプロセッサ101に対する

10

20

30

40

50

潜在的に単純化された修正を含む。

【0093】

[00110] 図7Aは、プロセッサがどのように追跡されるのかにもよるが、トレースから一部の読み出し→読み出し遷移を省略することができる一例を示す。図6Aと同様に、図7Aは、グローバルID701及び3つの処理ユニット(P0~P2)に対応する3つの列(702a~702c)を有する表700aを含む。一部の読み出し→読み出し遷移を省略することは2つの考えの上に成り立つ。第1に、書き込みは順序付ける必要があるが、連続する2つの書き込み間の読み出しの全て(例えばID[3]~ID[7]の読み出し)は同じ値を読み出し、そのためそれらの読み出し間の順序は問題にならない(従ってそれらの読み出し→読み出し遷移を省略するトレースは決定論的であり得る)。第2に、再実行時に読み出しを書き込みに「クロス」させること(即ち同じキャッシュラインに対する読み出し及び書き込みが間違った順序で再実行されること)は再実行に正しいデータが使用されていないことを意味するが、この間違いを犯すのを回避するためのデータ(例えばMIN等)を有することは有効な順序付けを識別するのに役立つ。

【0094】

[00111] 表700aに示す例では、処理ユニットP2は共有データへの読み出しのみを行い、(例えばID[9]がキャッシュラインを共有されたままにしておくと仮定し)それらの共有された読み出しは他の読み出しから「スチール」するだけである。読み出し→読み出し遷移(即ちID[4]~ID[7]及びID[10])の何れについてもログエントリが作成されない場合、P2の読み出しを適切に配置するための情報がトレース内にはない。書き込みに基づいて、(即ちID[2]における書き込みがP2からスチールしなかったので)P2は値DATA[1]を決して読み出さなかったと結論付けることができ、P2の読み出し→読み出し遷移(即ちID[4]、ID[7]、及びID[10])についてのログエントリがなく、P2について結論付けることができる全てはID[2]とID[8]との間でP2による少なくとも1回の読み出しがあったことである。しかしID[4]及びID[10]のログエントリがあった場合、ログする必要がない可能性がある残りの読み出し(即ち図7Bに示すID[5]~ID[7])を突き止めることができる。これらの読み出しのそれぞれは、最後に(即ちID[4]において)ログされた読み出しと同じ書込間セクション(inter-write section)に属する。従って、これらの読み出しは、書き込みが何処からスチールするのかに基づいて突き止めることができる(どの操作も読み出しからスチールしない場合、次にログされるパケットまでその後の書き込みはない)。

【0095】

[00112] 表700aに照らして、図7Bは「ユニットビット」を使用する場合に記録され得る、図7Aで強調した読み出し→読み出し遷移を省略したロギングデータを示す表700bを示す。図7Cは、「インデックスビット」が使用され、インデックスが読み出し時に更新される場合に記録され得るロギングデータを示す表700cを示す。

【0096】

[00113] 上記で簡潔に述べたように、一部のキャッシュは包含的な層及び排他的な層の両方(即ち不完全に包含的なキャッシュ)を含む。本明細書に記載のロギング技法は、これらのキャッシュ並びに純粹に包含的な又は排他的なキャッシュに適用可能である。一例として図8Aは、2つのプロセッサ801a/801b(例えば対応するソケット内の2つのプロセッサ)を含む計算環境800aを示す。各プロセッサ801は、4つの処理ユニット802a/802b(例えば物理的な又は論理的な処理ユニット)を含む。各プロセッサ801は、L1層803a/803b、L2層804a/804b、及びL3層805a/805bを含む3層キャッシュも含む。図示のように、各キャッシュは処理ユニット802の1つにそれぞれ対応する4つのL1キャッシュ803を含む。加えて各キャッシュは、処理ユニット802の2つにそれぞれ対応する2つのL2キャッシュ804を含む。加えて各キャッシュは、プロセッサ801内の処理ユニット802の全てのための1つのL3キャッシュ805を含む。処理ユニット及びキャッシュの一部は、個々に識

別され、例えば処理ユニット802aのプロセッサ801aは、A0～A3として識別され、L2キャッシュは、A4及びA5として識別され、L3キャッシュは、A6として識別される。プロセッサ801b内の対応するコンポーネントに同様の識別子を使用する。処理ユニットA0、A1、A2、B0、及びB1に関連付けられたアスタリスク(*)は、これらの処理ユニットについてロギングが有効にされていることを示す。

【0097】

【00114】 計算環境800aでは、キャッシュが包含的挙動及び排他的挙動の混合を示すことができる。例えば処理ユニットA0だけが或るキャッシュラインを使用している場合、プロセッサ801aのA6 L3キャッシュがそのキャッシュラインを記憶するのは非効率であり得る。代わりにこの場合、キャッシュラインは、A0のL1キャッシュ及びA4 L2キャッシュ内に記憶され得るが、A1のL1キャッシュ又はA5 L2キャッシュ若しくはそれよりも下位のキャッシュ上には記憶されない。空間を解放するために、この状況では一部のキャッシュはA6 L3キャッシュがそのキャッシュラインを追いつくことを可能にし得る。この追いつきが行われると、A1は包含的キャッシュでは正常であるようにA4 L2キャッシュからキャッシュラインを取得することができる。しかし、キャッシュラインがA6 L3キャッシュ又はA5 L2キャッシュ内に存在しないので、一部のキャッシュ実装はA0のL1キャッシュからA2又はA3のL1キャッシュ等へのキャッシュラインの横方向の移動も可能にし得る。この横方向の移動はCCPを用いたトレーシングに幾らかの課題を提示し得る。以下の例は、かかる状況においてCCPを用いたトレーシングをどのように実現できるのかを示す。

【0098】

【00115】 図8Bは、処理ユニット802の一部によって実行される読み出し及び書き込み操作の一例を示す表800bを示す含む。表800bの形式は表600aの形式と同様である。計算環境800a及び表800bに照らして、それぞれ異なるキャッシュ挙動を使用する3つの異なるロギングの例を示す。これらの例は、CCPを用いたロギングのための以下の原理に関連して説明する：

(1) 概して、アドレス(キャッシュライン)が「未ログ」から「ログ済み」になる(即ち動作304においてキャッシュラインがロギングに関与していると判定することに基づいて)場合はデータをログする。

(2) 概して、キャッシュラインが「ログ済み」から「未ログ」又は「追出し済み」になる場合はロギングを控える(このデータがログされる場合はログが依然として有効である)。しかし、追いつきをログすることは有効である。そうすることによりトレースサイズが大きくなるが、トレースデータストリーム間の順序付けを識別するのを助けことができ、トレースの再実行が何時「脱線」したのかを識別するのを助けことができ、更なるトレース分析を提供することができる追加情報を与える。トレース分析に関して、追いつきをログすることはキャッシュがどのように使用されたのかに関するより多くの情報を提供することができ、実行されるコードの性能特性を識別するために使用することができ、所与のキャッシュラインが特定の値を記憶した時間窓を突き止めるのを助けすることができる。追いつきをロギングするための実施形態は、図10A及び図10Bに関連して後述する。

(3) 新たな情報を提供するやり方でキャッシュラインがコア又はキャッシュコヒーレンシ状態を横断する場合は動きをログする。

(4) 処理ユニットが書き込みを行う場合、他の全ての処理ユニットについてキャッシュラインを無効化する。キャッシュラインが処理ユニットについてまだログされていない場合、システムはそのキャッシュラインをログしないことができ、又はキャッシュライン内のメモリが処理ユニットにとって読み出し可能だと仮定し、書き込みを(i)読み出し(即ちキャッシュラインをログし、ロギング追跡をオンにする)として(ii)書き込みと共に扱うことができる。プロセッサがロギングをオフにして書き込みをログしないことは正当だが効率性で劣り得る。しかし、そのようにすることは再実行時に再構築される必要がある情報を失い、データのフルキャッシュラインを後でログするよりも参照をログし

10

20

30

40

50

た方が安価なので平均ではトレースサイズが非効率であり得る。

(5) トレースを後で再構築するのを助けるために、(例えば上記の原理2にあるように) オーバログすることは有効である。このことはトレースサイズを増大させるが、正確さには影響を与えない。例えば(図7A~図7Cに関連して上記で説明したように) 一部の読み出し→読み出し遷移を省略することができるが、書き込みで始まる又は終わる任意のコア間遷移は明示的に又は暗黙的にログすべきである。別の例では、実施形態は(例えば追加の順序付け情報及び/又はハッシュを提供する) 追加のデータパケットをトレースに何時でも追加することができる。更に別の例では、実施形態は、書き込み状態へのそのCCP遷移後にキャッシュラインに書き込みが最初に完遂されたときにログすることができる(即ち投機的実行はキャッシュラインを書き込み状態に遷移させ得るが、キャッシュラインへの如何なる書き込みも実際に完遂しない場合があるからである)。これらの書き込みをログすることは、コアごとのトレースストリームの分離を後で容易にすることができる。更に別の例では、実施形態は間接ジャンプ又はトレースデータストリームを分離する際に探索空間を素早く減らすのを助ける他の情報をログし得る。

10

(6) 非フルログ(即ち遷移の全てはログしないもの)を引き続き使用してトレースを再実行することができる。このことは、欠落部分を計算するための余分な計算コストを再実行時に引き起こし得る。

【0099】

[00116] 図8Cに示す第1の例では、CCPが処理ユニットごとにキャッシュライン状態を追跡する(即ち各コアは独自の読み出し状態及び書き込み状態を有する)。この例では、ロギング時に利用できないキャッシュ間又はソケット間を移動するデータがあり得ることを除き、キャッシュが包含的キャッシュのように挙動する。簡潔にするために、これらの例では処理ユニット802を「コア」と呼び、プロセッサ801a及び801bをプロセッサA及びプロセッサB又はソケットA及びソケットBと呼ぶ。加えて、「ID: Core: From: Transition (即ちfrom→to)」の簡略化したロギング表記を使用してロギング可能なデータの種類を表す。この表記についてはインラインでより詳細に説明する。第1の例ではロギングが以下を含み得る:

20

【0100】

[00117] ID[0]において、「0:A0:R[DATA]→[1]」、即ちID[0]においてコアA0がDATA[1]を読み出すことを上記の原理1に従ってログする。

30

【0101】

[00118] ID[1]において、「1:B0:R[DATA]→[1]」、即ちID[1]においてコアB0がDATA[1]を読み出すことを同じく上記の原理1に従ってログする。A0がデータを既にログしていることをプロセッサB内のキャッシュが認識していない場合、プロセッサBがそのデータを自らログする。或いは、A0がDATA[1]をログしたことをプロセッサB内のキャッシュが認識している場合、ログエントリは「1:B0:R[A0]→R」を含むことができる。

【0102】

[00119] ID[2]において、「2:A1:R[A0]→R」、即ちID[2]においてコアA1が読み出し→読み出し遷移を行ったこと、及びA0がアクセスを有していたことをログする。キャッシュライン状態はプロセッサBと共有されるので、エントリは「2:A1:R:[A0,B0]→R」とすることができ、即ちID[2]においてA1が読み出し→読み出し遷移を行ったこと、並びにA0及びB0がアクセスを有していたことをログする。ソケットを横断することは典型的にはソケット内のロギングよりも高価なので、読み出し→読み出し遷移について第1のログエントリが好ましい場合がある。しかし、ソケットを横断する書き込みとの間でロギングする場合、ロギングもソケットを横断する。

40

【0103】

[00120] ID[3]において、一部の実施形態は何もログしない。或いはA2コアは

50

まだ何もログしておらず、最初に行うことが書き込みなので、これは読み出し→書き込みとしてログすることができる。何れにしても、書き込みが発生したので他のコアは自らのキャッシュライン状態が無効化される。ID [3] において読み出し→書き込みをログする（例えばトレースデータに関する）コストは、典型的にはID [4] において実際のデータをログするよりも少なく、そのためここでログするのが有益であり得る。この場合、ログエントリは「3 : A2 : R [A0, B1, B0] → W」を含むことができ、即ちコアA2は読み出し→書き込み遷移を行い、コアA0、B1、及びB0がアクセスを有していた。

【0104】

[00121] ID [4] において何が発生するのかは、ID [3] で何がログされたのかによって決まる。ID [3] において何もログされなかった場合はデータをログする（即ち「4 : A2 : R [DATA] → [2]」）。他方でID [3] においてパケットがログされている場合はログするものは何もない。

10

【0105】

[00122] ID [5] において、コアを横断する読み出しがある。しかし、A2コアが依然として修正された（又は同等の）キャッシュラインを有する場合、キャッシュラインが要求を供給する（メモリから供給することはできない）。その場合、ソケットBはそれがソケットAから来たことを知り、データを再ロギングするのを回避することができ、「5 : B0 : W [A2] → R」としてログすることができる。キャッシュがメインメモリからデータを得た場合（これはソケットAがメインメモリを更新し、ラインに関する自らのキャッシュコヒーレンス状態を共有できる場合に該当し得る）、エントリは「5 : B0 : R [DATA] → 2」とすることができる。

20

【0106】

[00123] ID [6] において、操作は通常の読み出しである。ID [2] における読み出しと同様に、ソケットBはソケットAのデータを知っている又は知らない可能性がある。知っている場合、ログエントリは「6 : B1 : R [B0, A2] → R」を含むことができ、さもなければ「6 : B1 : R [B0] → R」を含むことができる。

【0107】

[00124] ID [7] において、B0のキャッシュラインが追い出されていない場合、ログするものは何もない。B0のキャッシュラインが追い出されている場合、プロセッサBは別のコアから来るものとしてデータをログし、又はキャッシュラインデータをログする。ソケット内の他のコアではなくこの1つのコアの追い出しは、概して完全に包含的なキャッシュでは発生しない。完全に包含的なキャッシュでは、ソケット内の任意のコアがそのL1キャッシュ内にキャッシュラインを有する場合、L3がキャッシュラインを有し、そのためキャッシュラインは別のコアではなく1つのコアについて追い出すことはできない。

30

【0108】

[00125] ID [8] において、A0コアは何もログしておらず、ログする最初の操作は書き込みなので、ここでの操作はID [3] における操作と同様である。プロセッサAはこれを読み出し→書き込みとしてログすることができるが、代わりに、但しことによると好ましくないが、プロセッサAは何もログしなくてもよい。パケットがログされる場合、その内容はソケットAがソケットBを認めることができるのかどうかによって異なる。認めることができない場合、パケットは「8 : A0 : R [A2] → W」を含むことができるが、認めることができる場合、パケットは「8 : A0 : R [B0, B1, A2] → W」を含むことができる。

40

【0109】

[00126] ID [9] において、ID [8] においてパケットがログされている場合は（既にログされているキャッシュに対する書き込みなので）ログするものは何もないが、まだ無効化されていない場合は他のコアのキャッシュライン状態が典型的には無効化される。

50

【0110】

[00127] ID [10] において、ロギングはID [8] において何がログされたのかによって決まる。ID [8] においてデータがログされなかった場合はここで行う必要がある、そのためパケットは「10: A1: R [DATA] → [4]」を含むことができる。ID [8] においてパケットがログされた場合、これは通常書き込み→読み出しパケット（例えば「10: A1: W [A0] → R」）である。

【0111】

[00128] ID [11] において、読み出し→読み出し遷移がログされる。ID [8] においてパケットがログされた場合、A0はコアのソースリスト上にあり（例えば「11: A2: R [A0, A1] → R」）、さもないとリスト内にはない（例えば「11: A2: R [A1] → R」）。

10

【0112】

[00129] ID [12] において、ソケットBがソケットAを認めることができる場合、これは読み出し→読み出しパケット（例えば「12: B0: R [A0, A1, A2] → R」）である。認めることができない場合、それは完全なデータログ（例えば「12: B0: R [DATA] → [4]」）である。

【0113】

[00130] ID [13] において、データはB0から、加えてソケットAが可視であればソケットAから来る（例えば「13: B1: R [A0, A1, A2, B0] → R」）。ID [8] において書き込みがログされなかった場合、リストはコアA0を省略することができる。

20

【0114】

[00131] ID [14] において、パケットがID [8] においてログされている、既にログしている場合、何もログする必要はない。さもないと、A0はA1及びA1から、加えてソケットBが可視であればことによるとソケットBからデータを得る。従って、パケットは「14: A0: R [A1, A2, B0, B1] → R」を含むことができる。

【0115】

[00132] この例はソケットを一緒にログしたが、スレッドを分離してログできるやり方と同様に、各ソケットを分離してログすることは正しいことに留意されたい。分離してログすることはより大きなトレースの原因となる可能性があるが、プロセッサ内のどのソケット間通信メカニズムも変更する必要がないという利点を有する。

30

【0116】

[00133] 更に、任意の瞬間においてキャッシュラインを追い出すことができ、これはデータを別のコアから集める必要があること又は再びログする必要があることを意味する。例えばID [11] の前にA0のキャッシュラインが追い出された場合、A2はA1から値を得る。A1及びA0の両方が追い出された場合、プロセッサAはキャッシュライン値をA2のトレース内にログする必要があると得る。

【0117】

[00134] 最後に、一部のプロセッサはデータが別のソケットから来ることを知っているが、そのソケット内のどのコアかは知らない可能性がある。これらの場合、プロセッサはソケットIDとして先行（ソース）をログし、データ自体をログし、又はソケットID及びデータのハッシュをログする（即ちトレースに全データをログする必要なしにソケット間アクセスの順序付けを助けるために）ことができる。

40

【0118】

[00135] 図8Dに示す第2の例では、CCPは、各コアのキャッシュコヒーレンスを別々に追跡する代わりにインデックスを使用する。この環境では、インデックスはソケット間で又はソケット内で追跡することができる。ソケット間通信対ソケット内通信の性能のために、後者の事例（ソケット内）の方が実用的であり得る。インデックスがソケット内で追跡される場合、トレースはデータがソケット間を移動するとき何かをログする必要があると得る。これは、他のソケットからのインデックスをログすること（しかしこれは必

50

ずしも決定論的トレースにとって十分に一意でない可能性がある）、キャッシュライン値の1つ又は複数の部分のハッシュをログすること、又はデータが送信されたことを示すために送信側のソケットのトレース上のパケットをログすることを含み得る。

【0119】

【00136】 不完全に包含的なキャッシュを使用する場合にコアインデックスを追跡するとき、L1キャッシュがL3キャッシュ内にないデータを有し得る場合に困難な状況が生じる。そのため例えば以下の一連のイベントを想定されたい：(i) A0が自らのL1キャッシュ内にラインを取得し（従ってインデックスビットはA0を参照する）、(ii) A1が自らのL1キャッシュ内にラインを取得し（従ってインデックスビットはA1を参照する）、(iii) L3キャッシュがラインを追いつき出し、(iv) A1がL1キャッシュであるからラインを追いつき出し、(v) A2が自らのL1キャッシュ内にA0からキャッシュラインを取得する。ここでA2はA0からキャッシュラインを取得するが、インデックスはA0を参照しない。このことはトレース内へのマッピングのログを複雑にする。一部の解決策は（上記で説明したように）キャッシュラインデータの1つ又は複数の部分のハッシュ等の追加情報を追加すること、汎用レジスタのハッシュ等の冗長情報を周期的に追加すること等を含み得る。追いつきをログすることも役立ち得るが、かかるログはトレースファイルのサイズを大幅に増大させ、ロギングを複雑にし得る（例えばL2キャッシュ又はL3キャッシュ内にないL1キャッシュの追いつきはログするが、L2キャッシュ又はL3キャッシュ内にあるL1キャッシュの追いつきはログしない）。

【0120】

【00137】 一部の実施形態では、データがL3キャッシュから子のL2キャッシュ又はL1キャッシュに移動するとき、インデックスが変わる場合にのみログエントリが作成される。例えばA0が自らのL1キャッシュ内にラインを有する（従ってインデックスビットはA0を参照する）と仮定すると、A1は自らのL1キャッシュ内のラインを取得し（A1のインデックス）、その後どちらもキャッシュラインを追いつき出すが、共通のL2（又はL3）はそのキャッシュラインを依然として有する。L2キャッシュがA1に供給する場合、ログするものは何もない。L2キャッシュがA0に供給する場合、A0が既にそのデータを有していることが分かっている場合はログエントリを作成する必要はないが、A0が既にそのデータを有しているかどうか分からない（又は判定できない）場合、プロセッサは読み出し→読み出しをログする必要があると得る。

【0121】

【00138】 表800dは表800bの操作のログを示し、ソケットが独立にログし、追跡がインデックスによって行われ、追加の隠れた追いつきがなく、CCPに影響を与え且つロギングがオンにされるとき発生する全ての書き込みがログされる（例えば同じコアによる連続した書き込みがあり、別のコア又は他の外部エンティティによるアクセスがそれらの書き込みの間でない場合1つの書き込みをログする必要がある）と仮定する。第2の例ではロギングが以下を含み得る：

【0122】

【00139】 ID [0] では、「0 : A0 : R [DATA] → [1]」。

【0123】

【00140】 ID [1] では、「1 : B0 : R [DATA] → [1]」、即ち各ソケットが別々にログされることを思い出されたい。

【0124】

【00141】 ID [2] では、「2 : A1 : R [A0] → R」。

【0125】

【00142】 ID [3] では、「3 : A2 : R [A1] → W」。

【0126】

【00143】 ID [4] では、何ものなし。

【0127】

【00144】 ID [5] では、「5 : B0 : R [DATA] → [2]」。これはID [

3]における書き込みが全てのソケットにわたりラインを無効にし、ソケットが（上記のように）独立にトレースされているからである。

【0128】

[00145] ID [6] では、「6 : B1 : R [B0] → R」。

【0129】

[00146] ID [7] では、B0のキャッシュラインが追い出されていない場合ログするものは何もない。

【0130】

[00147] ID [8] では：「8 : A0 : R [A2] → W」であり、それはロギングビットがオンだから（且つそれにもかかわらずこのコアが前にデータをログしていないから）である。このエントリは、どのようにソケット内の最後のオーナの知識しかいないことをインデックスで示す。

【0131】

[00148] ID [9] では、ログするものは何もない。

【0132】

[00149] ID [10] では、「10 : A1 : W [A0] → R」。

【0133】

[00150] ID [11] では、「11 : A2 : R [A1] → R」。

【0134】

[00151] ID [12] では、「12 : B0 : R [data] → [4]」。これはID [8] において全てのソケットにわたりキャッシュラインが無効化されたからである。

【0135】

[00152] ID [13] では、「13 : B1 : R [B0] → R」。

【0136】

[00153] ID [14] では、「14 : A0 : R [A2] → R」。ID [11] において、インデックスがA2であるように更新されたことに留意されたい。このコアが既にデータを有していたかは分からず（即ちID [9]）、それはインデックスがその情報を運ばないからだが、以前はプロセッサごとの状態（ユニットビット）がその情報を運ぶことができたことにも留意されたい。

【0137】

[00154] 第3の例では、環境800a内のキャッシュは、どのコアがキャッシュラインへの最後の共有（読み出し）アクセスを有するのかを追跡することができない。従ってこの例では、最後のリーダのインデックスを追跡することができず、それはそのように追跡するためのビットがないからである。ここでCCPは、（例えばMSIプロトコルを使用し）共有ラインを知らせるために（どのコアにもマップしない）或るインデックス値を使用し、無効なラインを知らせるために別のインデックス値を使用し、「修正」状態のためにプロセッサインデックスを使用することができる。この第3の例では、ロギングはコアのインデックスの代わりにパケット内のキャッシュのインデックスをログすることを含み得る。親から子への動きはログする必要がないが、追加データとしてログしてもよい。親から子への動きをログしない場合、ログを解釈するために親から子へのキャッシュ階層を提供する必要があるが得る。

【0138】

[00155] 上記で述べたように、一部の環境ではキャッシュの各キャッシュラインが単一のフラグビットを含み得るが、プロセッサのCCPは、キャッシュラインのコヒーレンス状態を所有する処理ユニットへのインデックスを参照して各キャッシュラインのコヒーレンス状態を追跡することができる。上記のように、これは完全に決定論的なトレースをもたらすが、処理ユニットごとの情報を有する事例よりも大きいトレースの原因になり得る（例えばキャッシュラインごとのフラグビットと組み合わせて処理ユニットごとに追跡するCCP）。図9A及び図9Bは、これらの2つの状況（即ちCCPユニット情報+キャッシュラインフラグビット対CCPインデックス+キャッシュラインフラグビット）に

10

20

30

40

50

においてロギングがどのように異なり得るのかを示す。図 9 A は 2 つの処理ユニット（P 0 及び P 1）による読み出し及び書き込みを示す表 9 0 0 a を示し、図 9 B はこれらの 2 つの環境内でログエントリが作成され得るときを比較する表 9 0 0 b を示す。これらの例では、フラグビットがクリアされて始まり、キャッシュラインへのアクセスをどの処理ユニットも有さないことをユニット／インデックスビットが示すと仮定されたい。

【0139】

【00156】最初に、CCP がユニット情報を追跡し、キャッシュラインがフラグビットを使用する場合、ロギングは以下のように進めることができる。表 9 0 0 b に示すように、ログされていないキャッシュラインに対する書き込みなので、ID [0] において何もログする必要はない（或いは書き込み前の値をログすることができ、フラグビットをオンにすることができる）。この時点で、CCP は P 0 も P 1 もキャッシュラインにアクセスしないことを示し得る。ID [1] において、P 1 についてキャッシュラインデータをログすることができる。フラグビットをオンにすることができ、CCP は P 1 がキャッシュラインへのアクセスを有することを示し得る。ID [2] において、読み出し→読み出しパケットをログすることができ、P 0 は P 1 からキャッシュラインを取る（フラグビットがオンだったのでこれはログされ、P 0 がアクセスを有さなかったと判定するために CCP が使用される）。フラグビットは既にオンであり、CCP は今では P 0 もキャッシュラインの状態にアクセスできることを示す。ID [3] において、何もログする必要はない（キャッシュラインはこのコアについて既にログ内にある）。フラグビットがオンであり、P 1 がキャッシュラインへのアクセスを既に有していたことを CCP が示すのでこれ判定される。ID [4] において、P 0 について読み出し→書き込みパケットをログすることができる。これは、フラグビットがオンであり、P 0 がキャッシュラインへのアクセスを既に有していたからである。これは書き込みだったので、CCP は他の全てのプロセッサについてキャッシュラインを無効化することができる（即ち P 0 はアクセスを有し、P 1 はアクセスを有さない）。ID [5] において、P 1 について書き込み→読み出しパケットをログすることができる。これは、フラグビットはオンだが P 1 が（CCP によって示されるように）トレース内のデータを有さないからである。ID [4] 及び ID [5] における 2 つの参照パケットは、ID [4] において何もログせず ID [5] においてデータをログしなければならないよりも小さいことに留意されたい。CCP は、P 0 に加えて P 1 がキャッシュラインへのアクセスを現在有することを示す。

【0140】

【00157】今度は CCP がインデックス情報だけを追跡し、キャッシュラインがフラグビットを使用する場合、ロギングは以下のように進めることができる。表 9 0 0 b に示すように、ID [0] においてフラグビットがオフであり、これが書き込みなので何もログする必要はない。前述のように、メモリが P 0 によって読み出し可能である場合、これは代わりに読み出し+書き込みとしてログすることができる。ID [1] において、P 1 についてキャッシュラインデータをログすることができる。フラグビットをオンにすることができ、CCP 及び P 1 を指すようにインデックスを更新する。ID [2] において、P 0 について読み出し→読み出しパケットをログすることができる。これはフラグビットが既にオンであり、インデックスが P 1 上にあるからである。CCP はインデックスを P 0 に更新することができる。それ ID [3]、P 1 について読み出し→読み出しパケットをログすることができる。この事例は今のところ ID [2] と区別できず、それはどちらの場合も他のプロセッサ上のインデックス、フラグビットがオンであり、キャッシュラインが共有状態にあるからである。CCP はインデックスを P 1 に更新することができる。ID [4] において、P 0 について読み出し→書き込みパケットをログすることができる。フラグビットはオンであり、そのためパケットは参照によってログ可能である。これにより CCP のインデックスが P 0 に更新される。ID [5]、P 1 について書き込み→読み出しパケットをログすることができる。フラグビットはオンであり、パケットは参照によってログする。キャッシュラインは共有状態に移行し、そのため CCP はインデックスを P 1 に更新する。表 9 0 0 b に示すように、インデックスの事例はユニットの事

例よりも大きなトレースファイルの原因となるが、依然として完全に決定論的なトレースをもたらす。

【0141】

【00158】 本明細書の実施形態の一部は、キャッシュラインデータを後で記録するよりも、（可能な場合は）別の処理ユニットが保持するデータを参照するデータパケットを記録する方がトレースファイルのサイズに関して有益であり得ることを示した（例えば前の例のそれぞれにおけるID[4]）。参照による記録の他の利点も生じ得る。例えば再実行時に参照による一連のログエントリがある場合、キャッシュラインデータ内で外部介入が発生していないと推論することができる。なぜならフルキャッシュラインデータが再びログされる場合、それはキャッシュラインが追い出された又は無効化されたことを意味するからである。従って、ログエントリが厳密に必要な可能性のある状況でさえ参照によるログエントリを含めることは、再実行時に又はデバッグに有用な情報であり得る、外部介入がないことに関する暗黙的な情報を提供することができる。

10

【0142】

【00159】 一部の実装形態では、トレースエントリ内に記録されるアドレス（例えば上記の「@」エントリ）が物理メモリアドレスを含む。これらの実装形態では、プロセッサ102がTLB102fの1つ又は複数のエントリをトレースファイル104d内に記録することができる。かかる記録は異なる処理ユニットのためのトレースデータストリームの一部として、又はもう1つの追加のトレースデータストリームの一部としてであり得る。かかる記録は、再実行ソフトウェアがこれらの物理アドレスを後で仮想アドレスにマップできるようにする。

20

【0143】

【00160】 加えて、物理アドレスは時として（例えばユーザモードのレベルで記録するとき）「秘密」情報と見なされ得るので、一部の実施形態は物理アドレス自体ではなく実際の物理アドレスの何らかの表現を記録する。この表現は、物理アドレスを明かすことなしにその識別子を物理アドレスに一意にマップする任意の表現とすることができる。一例は各物理アドレスのハッシュであり得る。これらの表現が使用され、TLB102fのエントリがトレースファイル104d内に記録される場合、プロセッサ102は仮想アドレスに対する物理アドレスではなく、これらの表現と仮想アドレスとの間のマッピングを記録する。

30

【0144】

【00161】 先に述べたように、プロセッサ102は1つ又は複数のバッファ102eを含むことができる。これらのバッファは、トレースファイル102fにトレースファイルエントリを実際に書き込む前の、かかるエントリ用の一時的な記憶場所として使用することができる。従って、動作305がデータをトレースにログさせる場合、動作305はバッファ102eにデータを書き込むことを含み得る。一部の実施形態では、プロセッサ102及びメモリバスに対する、トレースデータを書き込む影響を減らすために、プロセッサ102がデファードロギング技法を使用する。これらの実施形態では、プロセッサ102はトレースデータをバッファ102e内に記憶し、メモリバス上に利用可能な帯域幅があるまで又はバッファ102eが満杯であるまでトレースファイル102fへの書き込みを先延ばしすることができる。

40

【0145】

【00162】 同じく先に述べたように、一部の実施形態はキャッシュの追い出しをログすることができる。図10A及び図10Bは、アソシアティブキャッシュの特性を活用する効率的な方法（即ちトレースファイルのサイズに関して）でキャッシュの追い出しをログできるやり方の一部の実施形態を示す。最初に図10Aは、メモリアドレスの異なる部分、及びアソシアティブキャッシュに対するそれらの部分の関係の一例1000を示す。図示のようにメモリアドレスは、アドレスの下位ビットであり、典型的にはゼロである第1の複数のビット1001を含む。メモリアドレスは典型的にはメモリアドレスのサイズ（例えば32ビット、64ビット等）に調整されるので、第1の複数のビット1001はゼ

50

口である。従って、第1の複数のビット1001の数はメモリアドレスのサイズに依存する。例えばメモリアドレスが32ビット（即ち 2^5 ビット）である場合、（メモリアドレスが32の倍数であるように）第1の複数のビット1001が5ビット含み、メモリアドレスが64ビット（即ち 2^6 ）である場合、（メモリアドレスが64の倍数であるように）第1の複数のビット1001が6ビット含む等である。メモリアドレスは、メモリアドレスのデータを記憶すべきアソシアティブキャッシュ内の特定のアドレスグループを決定するためにプロセッサ102によって使用され得る第2の複数のビット1002も含む。図10Aの例1000では、例えば第2の複数のビット1002は、8つのアドレスグループを有するアソシアティブキャッシュに対応する3ビットを含む。従って、第2の複数のビット1002の数はアソシアティブキャッシュの特定のジオメトリに依存する。メモリアドレスは、メモリアドレスの残りの上位ビットを含む第3の複数のビット1003も含む。

【0146】

【00163】 図10Aの脈絡において、図10Bはアソシアティブキャッシュにおけるキャッシュミス及びキャッシュ追い出しをログする一例1004を示す。最初に、例1004は3つのメモリアドレス1005（即ちアドレス1024）、1006（即ちアドレス@2112）、及び1007（即ちアドレス@2048）を示す。図10Bは、4つのウェイをそれぞれ含む8つのグループを有するアソシアティブキャッシュ1010も示す。これらのグループ及びウェイの2進数識別は、括弧内の対応する10進表現と共に列1008（グループ）及び1009（ウェイ）内に示す。従って、例えばキャッシュ1010内のキャッシュライン（0, 0）、即ちグループ0、ウェイ0は、グループ「000」（列1008）及びウェイ「00」（列1009）として2進数で示され、キャッシュ1010内のキャッシュライン（0, 1）（グループ0、ウェイ1）は、グループ「000」（列1008）及びウェイ「01」（列1009）として2進数で示され、キャッシュ1010内のキャッシュライン（8, 3）、即ちグループ8、ウェイ3がグループ「111」（列1008）及びウェイ「11」（列1009）として2進数で示されるまで同様に続く。

【0147】

【00164】 ここで、アドレス1005（即ち@1024）上で第1のキャッシュミスがあると仮定する。ここでは、その第2の複数のビット1002が「000」なので、プロセッサ102はアドレス1005に対応するデータをキャッシュ1010のグループ0内に記憶すべきだと判定することができる。グループ0内の特定のウェイは、典型的にはプロセッサ固有のロジックによって選択される。しかし例1004では、（矢印によって示すように）1011aデータがウェイ0内に記憶されると仮定する。このキャッシュミスに関連して、トレーサ104aによって記録されるログデータは、メモリアドレス（即ち@1024）及びデータが記憶されたウェイ（即ちウェイ0）を含むことができる。トレーサ内にメモリアドレスを記憶するのに必要なビット数を減らすために、任意の数の圧縮技法を使用できることに留意されたい。グループ（即ちグループ0）は、メモリアドレスの第2の複数のビット1002から得ることができるのでログする必要はない。

【0148】

【00165】 次に、アドレス1006（即ち@2112）上で第2のキャッシュミスがあると仮定する。今度は第2の複数のビット1002が「010」なので、プロセッサ102は、アドレス1006に対応するデータをキャッシュ1010のグループ2内に記憶すべきだと判定することができる。ここでもグループ2内の特定のウェイは、典型的にはプロセッサ固有のロジックによって選択される。しかし例1004では、（矢印によって示すように）1011bデータがウェイ0内に記憶されると仮定する。このキャッシュミスに関連して、トレーサ104aによって記録されるログデータは、メモリアドレス（即ち@2112）及びデータが記憶されたウェイ（即ちウェイ0）を含むことができる。この場合もやはり、グループ（即ちグループ2）は、メモリアドレスの第2の複数のビット1002から得ることができるのでログする必要はない。

【0149】

【00166】 今度はアドレス1007（即ち@2048）上で第3のキャッシュミスがあると仮定する。第2の複数のビット1002はここでも「000」なので、プロセッサ102は、アドレス1007に対応するデータをキャッシュ1010のグループ0内に記憶すべきだと判定することができる。ここでも特定のウェイはプロセッサ固有のロジックによって選択されるが、（矢印1011cによって示すように）プロセッサがウェイ0を選択すると仮定する。このキャッシュミスに関連して、トレーサ104aによって記録されるログデータは、メモリアドレス（即ち@2048）及びデータが記憶されたウェイ（即ちウェイ0）を含むことができる。この場合もやはり、グループ（即ちグループ0）は、メモリアドレスの第2の複数のビット1002から得ることができるのでログする必要はない。

10

【0150】

【00167】 このキャッシュライン（0，0）はアドレス1005に現在対応しているので、アドレス1007上のこの第3のキャッシュミスはアドレス1005をキャッシュ1010から追い出させる。しかし実施形態は、この追い出しを記録する如何なるトレースデータの記録も控えることができる。これは追い出しがトレース内に既にあるデータ、即ちウェイ0内へのアドレス1007上の第2のキャッシュミスと共にウェイ0内へのアドレス1005上の第1のキャッシュミスから推論できるからである。グループ（即ちグループ0）はトレース内で明示的にログされない場合があっても、これらのアドレスから推論することができる。そのため、このトレースデータの再実行は追い出しを再現することができる。

20

【0151】

【00168】 一部の追い出しはキャッシュミス以外のイベントから生じる。例えばCCPは、異なるキャッシュ間の整合性を保つために追い出しを発生させる場合がある。例えばCCPイベントによってアドレス1006がキャッシュ1010のキャッシュライン（2，0）から追い出されると仮定する。ここでは、追い出しのグループ（即ち「010」）及びウェイ（即ち「00」）を記録することにより、追い出しを明示的にログすることができる。とりわけ追い出されたアドレスは、キャッシュライン（2，0）内にアドレス1006を運んだ第2のキャッシュミスをログする際に既に捕捉されているのでログする必要はない。従ってこの例では、追い出しは単なる5ビットのログデータ（任意の形式の圧縮前）でトレースファイル104d内に完全に捕捉することができる。

30

【0152】

【00169】 一部の実施形態は、処理ユニットにおいて実行されているスレッドが安全なエンクレープと相互作用する場合でも、その処理ユニットのアクティビティを安全にトレースすることもできる。当業者によって理解されるように、エンクレープは、ことによるとプロセッサ102において実行される最低レベルのソフトウェアからさえも機密情報（例えば暗号鍵、資格情報、バイオメトリックデータ等）を保護することができるハードウェアベースのセキュリティ機能である。従ってユーザモードプロセスから機密情報を保護することに加え、エンクレープはカーネル及び／又はハイパーバイザから機密情報を保護することさえできる。多くの実装形態では、エンクレープは、実行プロセスにはプロセスのアドレス空間にマップされたメモリの暗号化部分として映る。これは例えば、実行プロセス及びエンクレープのための異なるメモリページテーブルを使用することによって実装され得る。プロセスがエンクレープと相互作用するとき、プロセスは自らのマップされたメモリとの間で読み書きすることができ、エンクレープは自らのマップされたメモリ及び／又はプロセスのマップされたメモリとの間で読み書きすることができる。

40

【0153】

【00170】 第1のエンクレープアウェアなトレーシングの実施形態は、プロセスが相互作用するエンクレープのトレーシングを控えながら実行プロセスをトレースすると同時に、トレース対象プロセスを完全に再実行することを依然として可能にする。これらの実施形態では、そのアドレス空間への実行プロセスによるメモリ読み出しは、本明細書に既に

50

記載した1つ又はメカニズムを使用してトレース／ログされる。しかし、エンクレーブへのコンテキストスイッチがある場合、実施形態はトレース対象プロセスによって前に読み出され、その実行中にエンクレーブによって書き込まれる任意の記憶域を追跡することができる。トレース対象プロセスがエンクレーブへの切り替え後に再び実行される場合、これらの記憶域はトレース対象プロセスによってログされていないものとして扱われる。このようにして、トレース対象プロセスがこれらの記憶域から再び読み出す（ことによるとエンクレーブによってそれらの位置に配置されたデータを読み出す）場合、それらの読み出しはトレースにログされる。事実上、これはトレース対象プロセスにとって可視のエンクレーブの実行の任意の副次的作用が、エンクレーブの実行をトレースする必要なしにトレース内に捕捉されることを意味する。このようにして、エンクレーブの実行を実際に再実行する（或いはできる）必要なしに、トレース対象プロセスはこれらの副次的作用を利用して後で再実行することができる。アカウンティングビット（例えばフラグビット、ユニットビット、インデックスビット）、ウェイロック、CCPデータの使用等、トレース対象プロセスによって前に読み出され、その実行中にエンクレーブによって書き込まれる記憶域を追跡するために使用可能な（前述した）幾つかのメカニズムがある。

【0154】

【00171】 第2のエンクレーブアウェアなトレーシングの実施形態は、（例えば自らのアドレス空間へのアクセス及び／又はトレース対象プロセスのアドレス空間へのアクセスに基づいて）エンクレーブもトレースしながら（例えば自らのアドレス空間への読み出し等のアクセスに基づいて）実行プロセスをトレースする。これらの実施形態は、カーネル／ハイパーバイザとエンクレーブとの間に必要な信頼レベルがある場合に実装することができる。これらの実施形態では、エンクレーブの実行に関係するトレースデータを別個のトレースデータストリーム内にログする及び／又は暗号化することができ、そのため再実行を行う任意のエンティティは、エンクレーブの別個のトレースデータストリーム及び／又はエンクレーブの実行に関係するトレースデータを復号するために使用可能な暗号鍵へのアクセスなしにエンクレーブを再実行することができない。

【0155】

【00172】 第3のエンクレーブアウェアなトレーシングの実施形態は、第1の実施形態と第2の実施形態とを組み合わせる。従ってこれらの第3の実施形態は、エンクレーブ自体のトレース（即ち第2の実施形態）と共に、そのプロセスのエンクレーブ使用の副次的作用を含む実行プロセスのトレース（即ち第1の実施形態）を記録することができる。このことは、必要な特権レベル及び／又は暗号鍵を有するユーザもエンクレーブ自体の実行を再実行できるようにしながら、必要な特権レベル及び／又は暗号鍵を欠くユーザがトレース対象プロセスの実行を再実行することを可能にする。

【0156】

【00173】 これらのエンクレーブトレーシングの実施形態のそれぞれはエンクレーブを超えて、及びトレース中にその実行を保護する必要がある別のエンティティ（ここでは保護エンティティと呼ぶ）とトレース対象エンティティが相互作用する任意の状況に適用することができる。例えばカーネルモードプロセスと相互作用するユーザモードプロセスをトレースするとき、これらの実施形態の何れも使用することができる（ここではカーネルモードプロセスはエンクレーブと大差なく扱うことができる）。別の例では、ハイパーバイザと相互作用するカーネルモードプロセスをトレースするとき、これらの実施形態の何れも使用することができる（ここではハイパーバイザはエンクレーブと大差なく扱うことができる）。

【0157】

【00174】 トレース対象プロセスによって前に読み出されたどの記憶域が、その実行中に保護エンティティによって書き込まれるのかを追跡することが（例えば性能又はセキュリティの考察により）实际的でない、（例えばハードウェアサポートがないことにより）不可能である、又は望ましくない環境があり得る。かかる環境は上記のエンクレーブトレーシングの実施形態の使用を妨げ得る。しかし、これらの状況におけるトレーシングのた

めの技法もある。

【0158】

[00175] 第1の技法は、保護エンティティからのコンテキストスイッチ後に無効化されているようにプロセッサキャッシュを扱うことである。無効化されているものとしてプロセッサキャッシュを扱うことは、保護エンティティからのリターン後のトレース対象エンティティによる読み出しに（ログされ得る）キャッシュミスを生じさせる。これらのキャッシュミスは、保護エンティティによってトレース対象エンティティのアドレス空間内で修正され、トレース対象エンティティによって後で読み出された任意の値を含む。この技法は上記で説明した3つの実施形態よりも多くのトレースデータを生成し得るが、トレース対象エンティティによって依拠された保護エンティティの実行の効果を捕捉する。一部の実施形態では、この第1の技法は、保護エンティティからトレース対象エンティティへのリターン時に、（例えばプロセッサレジスタのスナップショットを含む）1つ又は複数のキーフレームを記録することもできる。キーフレームは、トレースデータの連続性が（即ち保護エンティティの実行中に）欠如していても、保護エンティティからのリターン後にトレース対象エンティティの再実行を開始できるようにする。

10

【0159】

[00176] 第2の技法は、トレース対象エンティティのアドレス空間からの保護エンティティによる読み出し、並びにトレース対象エンティティのアドレス空間内に保護エンティティによって行われる書き込みに関係するキャッシュミスをログすることである。このことは、保護エンティティの書き込みを生成した保護エンティティの命令にアクセスする必要なしに、トレースの再実行が保護エンティティの書き込みを再現することを可能にする。このことは、保護エンティティが読み出し、トレース対象エンティティが後でアクセスした（トレース対象エンティティのアドレス空間内の）データへの再実行アクセスも与える。保護エンティティの書き込みを（トレース対象エンティティのアドレス空間内に）ログし得るが、その読み出しは（無効化されたものとしてキャッシュを扱うことによりそれらの読み出しが後でログされる場合）ログしないハイブリッド手法が（CCPデータ等の十分なブックキーピング情報が利用可能な場合は）可能である。

20

【0160】

[00177] 本発明は、その趣旨又は本質的特性から逸脱することなく他の特定の形式で実施することができる。記載した実施形態はあらゆる点で例示に過ぎず、限定的ではないと解釈すべきである。従って、本発明の範囲は上記の説明ではなく添付の特許請求の範囲によって示す。特許請求の範囲の等価の意味及び範囲に含まれる全ての変更は、特許請求の範囲に包含されるべきである。

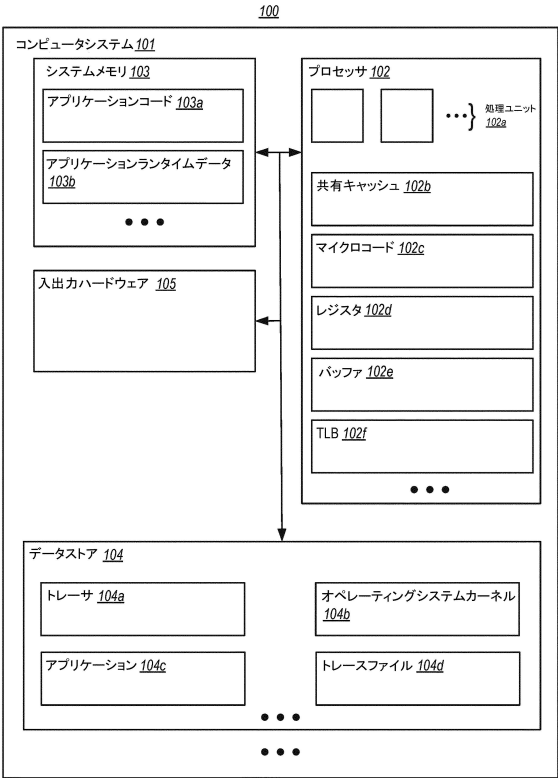
30

40

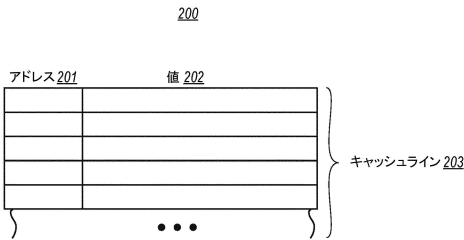
50

【図面】

【図 1】



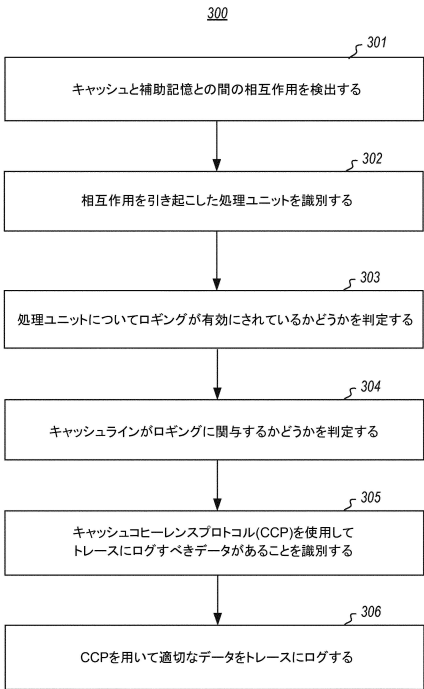
【図 2】



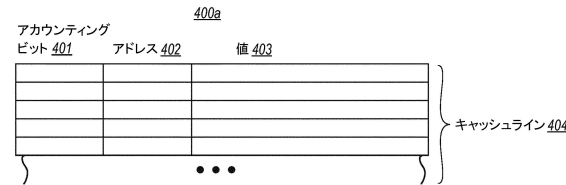
10

20

【図 3】



【図 4 A】

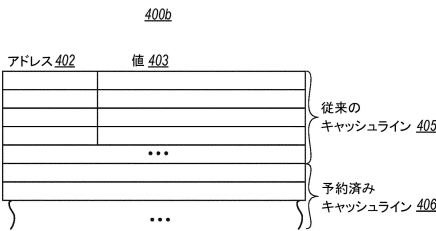


30

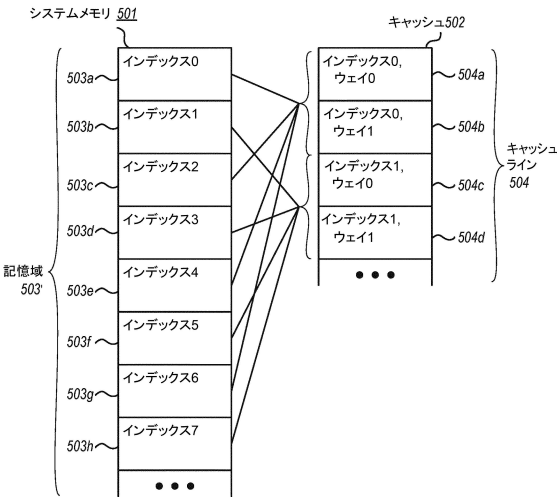
40

50

【図 4 B】



【図 5】



10

20

【図 6 A】

600a

601	602a	602b	602c	602d
ID	P0	P1	P2	P3
0	R[1]			
1		R[1]		
2	W[2]			
3		R[2]		
4	W[3]			
5			R[3]	
6		R[3]		
7				R[3]
8				W[4]
9	W[5]			
10			R[5]	
11		R[5]		
12	R[5]			
13	ラインの追い出し			

【図 6 B】

600b

601	602a	602b	602c	602d
ID	P0	P1	P2	P3
0	S			
1	S	S		
2	M	I		
3	S	S		
4	M	I		
5	S		S	
6	S	S	S	
7	S	S	S	S
8	I	I	I	M
9	M			I
10	S		S	
11	S	S	S	
12	S	S	S	
13	-	-	-	-

30

40

FIG. 6B

50

【図 6 C】

600c						
ID	603 ユニットビット				604 インデックス	605 フラグ
0	1	0	0	0	0	1
1	1	1	0	0	1	1
2	1	0	0	0	0	1
3	1	1	0	0	1	1
4	1	0	0	0	0	1
5	1	0	1	0	2	1
6	1	1	1	0	1	1
7	1	1	1	0	1	1
8	0	0	0	0	-1	0
9	0	0	0	0	-1	0
10	0	0	1	0	2	1
11	0	1	1	0	1	1
12	1	1	1	0	0	1
13	0	0	0	0	-1	0

【図 6 D】

600d	
ID	ログデータ
0	P0; @; IC[0]; DATA [1]
1	P1; @; IC[1]; R->R; P0->P0, P1
2	P0; @; IC[2]; R->W; P0, P1->P0
3	P1; @; IC[3]; W->R; P0->P0, P1
4	P0; @; IC[4]; R->W; P0, P1->P0
5	P2; @; IC[5]; W->R; P0->P0, P2
6	P1; @; IC[6]; R->R; P0, P2->P0, P1, P2
7	
8	
9	
10	P2; @; IC[10]; DATA[S], P2
11	P1; @; IC[11]; R->R; P2->P1, P2
12	P0; @; IC[12]; R->R; P1, P2->P0, P1, P2
13	

【図 7 A】

700a				
ID	701 P0	702a P1	702b P2	702c
0	R[1]			
1		R[1]		
2	W[2]			
3		R[2]		
4			R[2]	
5	R[2]			
6		R[2]		
7			R[2]	
8	W[3]			
9	R[3]			
10			R[3]	

【図 7 B】

700b	
0	P0; @; IC[0]; DATA [1]
1	P1; @; IC[1]; R->R; P0->P0, P1
2	P0; @; IC[2]; R->W; P0, P1->P0
3	P1; @; IC[3]; W->R; P0->P0, P1
4	P2; @; IC[4]; R->R; P0, P1->P0, P1, P2
5	
6	
7	
8	P0; @; IC[8]; R->W; P0, P1, P2->P0
9	
10	P2; @; IC[10]; R->R; P0->P0, P2

FIG. 7A

FIG. 7B

10

20

30

40

50

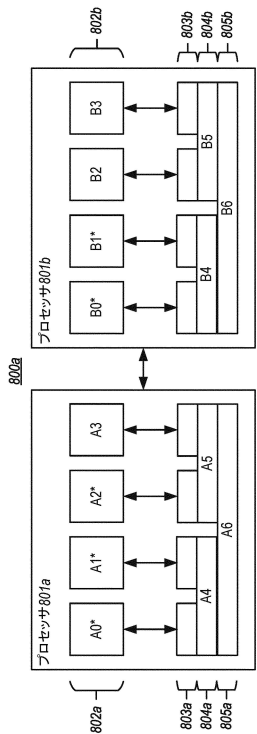
【図 7 C】

700c

0	P0; @; IC[0]; DATA [1]
1	P1; @; IC[1]; R->R; P0->P1
2	P0; @; IC[2]; R->W; P1->P0
3	P1; @; IC[3]; W->R; P0->P1
4	P2; @; IC[4]; R->R; P1->P2
5	P0; @; IC[5]; R->R; P2->P0
6	P1; @; IC[6]; R->R; P0->P1
7	P2; @; IC[7]; R->R; P1->P2
8	P0; @; IC[8]; R->W; P2->P0
9	
10	P2; @; IC[10]; R->R; P0->P2

FIG. 7C

【図 8 A】



【図 8 B】

800b

ID	A0	A1	A2	A3	B0	B1	B2	B3
0	R[1]							
1					R[1]			
2		R[1]						
3			W[2]					
4			R[2]					
5					R[2]			
6						R[2]		
7					R[2]			
8	W[3]							
9	W[4]							
10		R[4]						
11			R[4]					
12					R[4]			
13						R[4]		
14	R[4]							

FIG. 8B

【図 9 A】

900a

ID	P0	P1
0	W[1]	
1		R[1]
2	R[1]	
3		R[1]
4	W[2]	
5		R[2]

FIG. 9A

10

20

30

40

50

【図 9 B】

900b

ID	CCPユニット+フラグ	CCPインデックス+フラグ
0		
1	ログデータ	ログデータ
2	ログ R->R	ログ R->R
3		ログ R->R
4	ログ R->W	ログ R->W
5	ログ W->R	ログ W->R

【図 1 0 A】

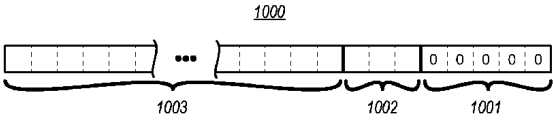


FIG. 10A

10

【図 1 0 B】

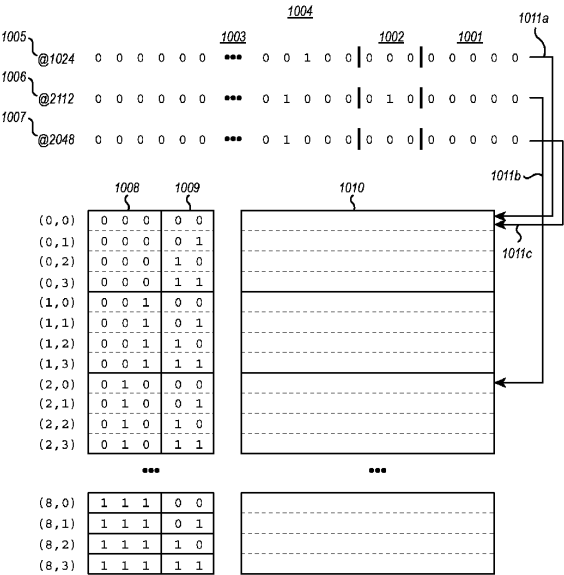


FIG. 10B

30

40

50

フロントページの続き

(33)優先権主張国・地域又は機関

米国(US)

弁理士 阿部 豊隆

(74)代理人 100134027

弁理士 松田 達也

(72)発明者 モラ, ジョルディ

アメリカ合衆国, ワシントン州 98052-6399, レッドモンド, ワン マイクロソフト ウ
エイ, マイクロソフト テクノロジー ライセンシング, エルエルシー

審査官 山本 俊介

(56)参考文献 特開平11-161545 (JP, A)

特開2000-148533 (JP, A)

特開2002-304328 (JP, A)

特開2007-207223 (JP, A)

(58)調査した分野 (Int.Cl., DB名)

G06F 11/34

G06F 12/0815