(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7666632号
(P7666632)

(45)発行日　令和7年4月22日(2025.4.22)

(24)登録日　令和7年4月14日(2025.4.14)

(51)国際特許分類

Ｇ０６Ｆ　11/34　(2006.01)	Ｇ０６Ｆ	11/34	１８０
Ｇ０６Ｆ　15/82　(2006.01)	Ｇ０６Ｆ	11/34	１７６
	Ｇ０６Ｆ	15/82	６１０Ｎ

FI

請求項の数　9　（全25頁）

(21)出願番号　特願2023-559356(P2023-559356)
(86)(22)出願日　令和3年11月12日(2021.11.12)
(86)国際出願番号　PCT/JP2021/041778
(87)国際公開番号　WO2023/084749
(87)国際公開日　令和5年5月19日(2023.5.19)
審査請求日　令和6年5月7日(2024.5.7)

(73)特許権者　000004226
日本電信電話株式会社
東京都千代田区大手町一丁目５番１号
(74)代理人　100098394
弁理士　山川 茂樹
(74)代理人　100153006
弁理士　小池 勇三
(74)代理人　100064621
弁理士　山川 政樹
(74)代理人　100121669
弁理士　本山 泰
(72)発明者　有川 勇輝
東京都千代田区大手町一丁目５番１号
日本電信電話株式会社内
(72)発明者　三浦 直樹

最終頁に続く

(54)【発明の名称】　コンピュータシステムおよびその制御方法

(57)【特許請求の範囲】

【請求項１】

複数の演算部とホスト部とを備え、前記複数の演算部がそれぞれトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記入力データの種別ごとに、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、

前記ホスト部と前記演算部とのいずれかが、前記トレースバッファ内に記録された、前記イベントの検出時刻が属する前記入力データの種別を判定するステップと、

前記ホスト部と前記演算部とのいずれかが、判定の結果に応じて、同一の前記入力データの種別に属する前記イベントの検出時刻のうち、最新の前記イベントの検出時刻を記録し、前記最新の前記イベントの検出時刻以外の前記イベントの検出時刻を消去するステップと

を備えるコンピュータシステムの制御方法。

【請求項２】

前記ホスト部が、前記トレースバッファを定期的に監視するステップとを備え、

前記ホスト部が、複数の前記イベントの検出時刻が同一の前記入力データの種別に属する場合に、前記イベントの検出時刻に基づき、前記トレースバッファ内に記録された前記イベントの検出時刻を消去する

ことを特徴とする請求項１に記載のコンピュータシステムの制御方法。

【請求項３】

前記演算部が、前記記録の前に、前記判定を実行し、前記イベントの検出時刻が同一の前記入力データの種別に属する場合に、前記イベントの検出時刻に基づき、前記トレースバッファ内に記録された前記イベントの検出時刻を上書きする
ことを特徴とする請求項１に記載のコンピュータシステムの制御方法。
【請求項４】
複数の演算部がそれぞれ演算器とトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、
前記複数の演算部のうち、一の演算部の演算器が、前記入力データを処理し、他の演算部の演算器に転送するステップと、
前記他の演算部が、受信完了通知を発信するステップと、
前記一の演算部が、前記受信完了通知を受信すると、前記一の演算部の前記トレースバッファに記録される前記イベントの検出時刻を消去するステップと
を備えるコンピュータシステムの制御方法。
【請求項５】
複数の演算部がそれぞれ演算器とトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、
所定の時間の経過を契機として、前記トレースバッファに記録された前記イベントの検出時刻を消去することを特徴とするコンピュータシステムの制御方法。
【請求項６】
入力データを処理するコンピュータシステムであって、
複数の演算部と、
前記複数の演算部と接続し、前記複数の演算部を制御するホスト部と
を備え、
前記複数の演算部の間で前記処理されたデータが転送され、
前記演算部が、前記入力データからの所定のイベントの検出を契機として、トレースデータを記録するトレースバッファ
を備え、
前記トレースデータが、前記演算部の動作周波数を基にする前記イベントの検出時刻であるタイムスタンプ値を有し、
前記ホスト部と前記演算部とのいずれかが、所定の条件で、前記トレースバッファに記録される前記トレースデータを消去する
ことを特徴とするコンピュータシステム。
【請求項７】
前記トレースデータが、さらに、前記入力データの種別と、前記イベントを検出する箇所を示す情報と、前記イベントの内容を区別する情報と、任意のデータとの少なくともいずれかを有する
ことを特徴とする請求項６に記載のコンピュータシステム。
【請求項８】
前記ホスト部と前記演算部とのいずれかが、前記トレースバッファ内に記録された、複数の前記トレースデータが同一の前記入力データの種別に属する場合には、前記複数の前記トレースデータのうち、最新の前記トレースデータを保持し、前記最新の前記トレースデータ以外の前記トレースデータを消去する
ことを特徴とする請求項６又は請求項７に記載のコンピュータシステム。
【請求項９】
前記複数の演算部のうち、一の演算部の演算器が、前記入力データを処理し、他の演算部の演算器に転送し、
前記他の演算部が、前記転送されたデータを受信し、受信完了通知を発信し、
前記一の演算部が、前記受信完了通知を受信し、前記一の演算部の前記トレースバッフ

ァに記録される前記イベントの検出時刻を消去する
ことを特徴とする請求項６又は請求項７に記載のコンピュータシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、複数の演算部を有するコンピュータシステムおよびその制御方法に関する。
【背景技術】
【０００２】
機械学習や人工知能（ＡＩ）やＩｏＴ（Ｉｎｔｅｒｎｅｔ ｏｆ Ｔｈｉｎｇｓ）など多くの分野で技術革新が進み、様々な情報やデータを活用することで、サービスの高度化・付加価値の提供が盛んに行われている。このような処理では、大量の計算をする必要があり、そのための情報処理基盤が必須である。
【０００３】
例えば、非特許文献１では、既存の情報処理基盤をアップデートする試みが展開されているが、急増するデータに対して現状のコンピュータが対応しきれていない。今後進展するためには、ムーアの法則を越える「ポストムーア技術」が確立されなければならないと指摘している。
【０００４】
ポストムーア技術として、例えば、非特許文献２では、フローセントリックコンピューティングという技術が開示されている。フローセントリックコンピューティングでは、データのある場所で処理を行うという従来のコンピューティングの考えではなく、計算機能が存在する場所にデータを移動して処理を行うという新たな概念が導入された。
【０００５】
上記のフローセントリックコンピューティングを実現するためには、データ移動に必要な広帯域な通信ネットワークが必要になるだけでなく、所望の演算性能が得るためには計算リソースを効率よく制御する必要がある。
【０００６】
フローセントリックコンピューティング（例えば、非特許文献２）では、複数の演算機能を連動させる手法が開示されている。
【先行技術文献】
【非特許文献】
【０００７】
【文献】“NTT Technology Report for Smart World 2020,”日本電信電話株式会社, 2020年. https://www.rd.ntt/_assets/pdf/techreport/NTT_TRFSW_2020_EN_W.pdf.
【文献】R. Takano and T. Kudoh, “Flow-centric computing leveraged by photonic circuit switching for the post-moore era,” Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Nara, 2016, pp. 1-3. https://ieeexplore.ieee.org/abstract/document/7579339.
【発明の概要】
【発明が解決しようとする課題】
【０００８】
しかしながら、複数の演算部が連動するコンピュータシステムにおいて、ホスト部を経由せずに演算部同士が主体的にデータを移動させるため、演算部内で生じた障害を特定することが困難であった。
【０００９】
また、入力データがある時刻に通過した演算部を特定するなどコンピュータシステムの内部状態を把握することが困難であった。
【課題を解決するための手段】
【００１０】

上述したような課題を解決するために、本発明に係るコンピュータシステムの制御方法は、複数の演算部とホスト部とを備え、前記複数の演算部がそれぞれトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記入力データの種別ごとに、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、前記ホスト部と前記演算部とのいずれかが、前記トレースバッファ内に記録された、前記イベントの検出時刻が属する前記入力データの種別を判定するステップと、前記ホスト部と前記演算部とのいずれかが、判定の結果に応じて、同一の前記入力データの種別に属する前記イベントの検出時刻のうち、最新の前記イベントの検出時刻を記録し、前記最新の前記イベントの検出時刻以外の前記イベントの検出時刻を消去するステップとを備える。

【００１１】

また、本発明に係るコンピュータシステムの制御方法は、複数の演算部がそれぞれ演算器とトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、前記複数の演算部のうち、一の演算部の演算器が、前記入力データを処理し、他の演算部の演算器に転送するステップと、前記他の演算部が、受信完了通知を発信するステップと、前記一の演算部が、前記受信完了通知を受信すると、前記一の演算部の前記トレースバッファに記録される前記イベントの検出時刻を消去するステップとを備える。

【００１２】

また、本発明に係るコンピュータシステムの制御方法は、複数の演算部とホスト部とを備え、複数の演算部がそれぞれトレースバッファを備え、入力データからの所定のイベントの検出を契機として、前記演算部の動作周波数を基に取得される前記イベントの検出時刻を前記トレースバッファに記録するコンピュータシステムの制御方法であって、前記ホスト部と前記演算部とのいずれかが、所定の時間の経過を契機として、前記トレースバッファに記録された前記イベントの検出時刻を消去することを特徴とする。

【００１３】

また、本発明に係るコンピュータシステムは、入力データを処理するコンピュータシステムであって、複数の演算部と、前記複数の演算部と接続し、前記複数の演算部を制御するホスト部を備え、前記複数の演算部の間で前記処理されたデータが転送され、前記演算部が、前記入力データからの所定のイベントの検出を契機として、トレースデータを記録するトレースバッファを備え、前記トレースデータが、前記演算部の動作周波数を基にする前記イベントの検出時刻であるタイムスタンプ値を有し、前記ホスト部と前記演算部とのいずれかが、所定の条件で、前記トレースバッファに記録される前記トレースデータを消去することを特徴とする。

【発明の効果】

【００１４】

本発明によれば、障害が生じた場合に、容易に、障害が生じた箇所を特定でき、障害発生時の内部でのデータ状況を把握できるコンピュータシステムおよびその制御方法を提供できる。

【図面の簡単な説明】

【００１５】

【図１】図１は、本発明の第１の実施の形態に係るコンピュータシステムの構成を示すブロック図である。

【図２】図２は、本発明の第１の実施の形態に係るコンピュータシステムにおける演算部の構成を示すブロック図である。

【図３Ａ】図３Ａは、本発明の第１の実施の形態に係るコンピュータシステムにおける演算部の拡張および縮小を説明するための図である。

【図３Ｂ】図３Ｂは、本発明の第１の実施の形態に係るコンピュータシステムにおける演算部の拡張および縮小を説明するための図である。

【図４Ａ】図４Ａは、本発明の第１の実施例に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【図４Ｂ】図４Ｂは、本発明の第１の実施例に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【図４Ｃ】図４Ｃは、本発明の第１の実施例に係るコンピュータシステムの制御方法を説明するための図である。
【図５】図５は、本発明の第２の実施例に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【図６】図６は、本発明の第３の実施例に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【図７Ａ】図７Ａは、本発明の第１の実施の形態に係るコンピュータシステムの制御方法の一例を説明するための図である。
【図７Ｂ】図７Ｂは、本発明の第１の実施の形態に係るコンピュータシステムの制御方法の一例を説明するための図である。
【図８】図８は、本発明の第２の実施の形態に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【図９】図９は、本発明の第２の実施の形態に係るコンピュータシステムの制御方法を説明するためのフローチャート図である。
【発明を実施するための形態】
【００１６】
＜第１の実施の形態＞
本発明の第１の実施の形態に係るコンピュータシステムとその制御方法について、図１～図３を参照して説明する。
【００１７】
＜コンピュータシステムの構成＞
本実施の形態に係るコンピュータシステム１０は、図１に示すように、Ｎ個の演算部１１＿１～１１＿Ｎ（Ｎは１以上の整数）と、演算部１１＿１～１１＿Ｎを接続する内部通信部１３と、演算部１１＿１～１１＿Ｎに対して動作パラメータを設定・管理するホスト部１２とを備える。
【００１８】
演算部１１＿１～１１＿Ｎは、プロセッサやアクセラレータなどにより構成され、トレース部１４＿１～１４＿Ｎを備える。
【００１９】
トレース部１４＿１～１４＿Ｎは、演算部１１＿１～１１＿Ｎが連動する際に、各演算部１１＿１～１１＿Ｎの任意の観測ポイントにおいて、所定のイベントの検出を契機として、各演算部１１＿１～１１＿Ｎの動作周波数を基にしたイベント検出時刻を記録する。
【００２０】
ここで、トレース部１４＿１～１４＿Ｎは、データ種別またはイベント種別ごとに、イベント検出時刻を記録できる。さらに、任意のデータを記録してもよい。
【００２１】
演算部１１＿１～１１＿Ｎの連動において、演算部１１＿１で処理されたデータが、内部通信部１３を介して、演算部１１＿２に転送される。引き続き、データの転送が繰り返され、データが演算部１１＿Ｎに転送される。
【００２２】
なお、演算部１１＿１～１１＿Ｎの連動の方法として、複数の演算部を直列に接続する処理方法や、複数の演算部を並列に接続する処理方法や、両者を組み合せた処理方法などが挙げられる。複数の演算部が連動することで、所望のサービスを提供し、アプリケーションを処理する。
【００２３】
演算部１１＿１～１１＿Ｎ（Ｎは１以上の整数）は、コンピュータシステム１０の外部

から入力される入力データに対して所定の演算処理を実行する機能を有する。演算処理とは、例えば、画像データが入力された際に画像サイズを縮小・拡大する処理や、画像データから特定の物体を検出する処理、画像データを復号・暗号化する処理など、入力データに対する加工、集計、結合などの一般的な演算処理である。

【００２４】

また、演算部１１＿１～１１＿Ｎは、当該システムの停止中・稼働中を問わず、追加・削除してもよい。例えば、演算器の一部分のみを動的再構成が可能なデバイスであるＦＰＧＡを用いることで実現することができる。また、演算部１１＿１～１１＿Ｎの実装方法として、特定の演算に特化した専用回路を具備するアクセラレータカードを追加しても良い。また、演算部の中に演算機能を提供する演算器を複数具備することもできる。

【００２５】

ホスト部１２は、演算部１１＿１～１１＿Ｎに対して動作パラメータを設定・管理する機能を有し、詳細には演算部１１＿１～１１＿Ｎを制御する機能や、データを記憶する機能を有する。動作パラメータは、例えば、画像処理において複数のアルゴリズムを切り替えて使う場合、アルゴリズムを特定するための情報であり、演算処理における係数や閾値などである。

【００２６】

また、当該システムの稼働開始後であっても演算部の追加・削除ができる場合、ホスト部１２は、その演算部に対して、所望の処理内容を実行するための回路情報を演算部に対して設定するなど、コンピュータシステム１０全体の管理を行う。

【００２７】

内部通信部１３は、演算部１１＿１～１１＿Ｎを接続するとともに、演算部１１＿１～１１＿Ｎ間でデータの授受を行うための通信機能を有する。具体的には、ＰＣＩｅやイーサネットなど市中の通信規格と前記通信規格を満足する物理構成、すなわちＰＣＩｅスイッチやイーサネットスイッチが挙げられる。

【００２８】

また、演算部１１＿１～１１＿Ｎの中に演算機能を提供する演算器を複数具備する場合、前記観測ポイントを演算部１１＿１～１１＿Ｎの中に複数設けてもよい。

【００２９】

＜演算部の構成＞

コンピュータシステム１０における演算部１１＿１は、図２に示すように、複数（Ｎ台）の演算器１５＿１（１）～１５＿Ｎ（１）と、トレース部１４＿１とを備える。ここで、演算器は１台であってもよい。

【００３０】

トレース部１４＿１は、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎと、タイムスタンプ部１７と、トレースバッファ１８とを備える。

【００３１】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎは、演算器１５＿１（１）～１５＿Ｎ（１）の入力側と出力側それぞれに接続される。タイムスタンプ部１７の出力が、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎに接続される。イベントジェネレータ１６＿１＿１～１６＿２＿Ｎの出力は、トレースバッファ１８に接続される。

【００３２】

ここで、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎは、演算器１５＿１（１）～１５＿Ｎ（１）の入力側と出力側いずれかに配置されればよく、任意の箇所に配置されればよく、少なくとも１台配置されればよい。また、トレースバッファ１８は複数台配置されてもよく、少なくとも１台配置されればよい。

【００３３】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎは、演算部１１＿１～１１＿Ｎの任意の箇所に挿入されて、データの種別（ユーザＩＤ、セッションＩＤ、ストリームＩＤ、サービスＩＤ）ごとにイベント（ストリームの先頭、末尾）を検出し、検出時刻（以下、

「タイムスタンプ値」という。）を含むトレースデータを後述するトレースバッファ１８に記録する契機を発生させる。

【００３４】

また、データの種別は上記に限らず、データを整理するために用いるパケットのヘッダ情報や、データと並走する信号が有する情報など、データの整理に利用できる情報であれば適用することができる。

【００３５】

タイムスタンプ部１７は、少なくとも１台のクロックカウンタを備え、複数のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎ（観測ポイント）間を同期させるとともに、演算部１１＿１～１１＿Ｎの動作周波数の精度で時刻を取得する。ここで、演算部１１＿１～１１＿Ｎの動作周波数（クロック周波数）は、ＦＰＧＡ（ｆｉｅｌｄ－ｐｒｏｇｒａｍｍａｂｌｅ　ｇａｔｅ　ａｒｒａｙ）を用いて当該機能を実現する場合、通常、数ナノ秒程度である。

【００３６】

トレースバッファ１８は、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎによるイベント検出を契機として、トレースデータを記録する。ここで、トレースデータは、タイムスタンプ部１７から取得した各検出時刻（タイムスタンプ値）と、インスタンスＩＤと、イベント種別（イベントＩＤ）と、データ種別（ＴＩＤ）と、任意のデータとを有する。ここで、トレースデータは、少なくともタイムスタンプ値を有すればよい。

【００３７】

また、トレースバッファ１８は、イベントジェネレータの数に依存することなく、一定のバッファ量を提供する。

【００３８】

ここで、タイムスタンプ値は、演算部（ＦＰＧＡ）内で統一された値である。

【００３９】

また、インスタンスＩＤは、イベントジェネレータ・インスタンスを区別し、前記イベントを検出する箇所（観測ポイント）を示すＩＤである。

【００４０】

また、イベント種別（イベントＩＤ）は、イベント内容を区別するＩＤである。例えば、ストリームの先頭の通過またはストリームの終端の通過により区別する。また、データの任意の箇所にイベント検出用のフラグ等を用意して、当該フラグが通過したことを検出する。

【００４１】

また、任意のデータは、画像データ、数値データ、文章データなど通常コンピュータシステムで処理されるデータである。

【００４２】

また、データ種別は、例えば、入力データの属性などを識別・分類したりするために用いられ、ユーザＩＤ、セッションＩＤ、ストリームＩＤ、サービスＩＤなど、データ本体に付属する情報である。また、データ種別を識別するための情報は必ずしもパケットのヘッダに付与されなくともよく、例えば、パケットのペイロードに独自に定義してもよい。また、演算部の内部において、データと並走する信号を用いる場合、データ種別を取得するのに並走信号を用いてもよい。

【００４３】

＜コンピュータシステムの動作＞

本実施の形態に係るコンピュータシステム１０の動作を、以下に説明する。

【００４４】

コンピュータシステム１０の演算部１１＿１において、演算器１５＿１（１）～１５＿Ｎ（１）にデータが入力される。入力されるデータは、様々な要素から構成され、イベント種別（イベントＩＤ）と、データ種別（ＴＩＤ）と、任意のデータを含む。

【００４５】

ここで、演算部１１＿１の演算器１５＿１（１）で処理されたデータは送信され、演算部１１＿２の演算器１５＿１（２）に入力する。

【００４６】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎにおいて、まず、演算部１１＿１～１１＿Ｎに入力するデータの制御信号（動作）を観測する。

【００４７】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎがイベントを検出すると、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎが、入力データから、イベント種別と、データ種別と、任意のデータを取得する。ここで、イベント発生は、例えば、ストリームの先頭が通過した時、またはストリームの先頭が通過した時である。

【００４８】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎで、取得されたイベント種別とデータ種別と任意のデータに、インスタンスＩＤと、タイムスタンプ部１７から送信されたタイムスタンプ値とが加えられる。その結果、トレースデータは、タイムスタンプ値と、インスタンスＩＤと、イベント種別と、データ種別と、任意のデータとで構成される。

【００４９】

ここで、トレースデータは、少なくともタイムスタンプ値を含めばよく、タイムスタンプ値を基に処理時間、データ流量などのコンピュータシステム内の情報を把握できる。また、不具合が生じた時刻なども把握できる。

【００５０】

さらに、トレースデータは、インスタンスＩＤを有することにより、不具合が生じた箇所を把握できる。

【００５１】

また、トレースデータは、イベント種別を有することにより、イベント発生時を把握できる。

【００５２】

また、トレースデータは、データ種別を有することにより、データ種別ごとの稼働状況を把握でき、データ消去時の判定（後述）に用いることができる。

【００５３】

また、トレースデータは、任意のデータを有することにより、処理の再開（後述）時に用いることができる。

【００５４】

また、トレースデータは、サービスの優先度情報を有することにより、優先度に基づくトレースデータの管理に用いることができる。

【００５５】

最後に、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎが、トレースデータをトレースバッファ１８に送信する。

【００５６】

また、タイムスタンプ部１７において、ホスト部１２から送信されたカウントの開始又は停止設定が受信（書き込み）される。

【００５７】

これを契機に、クロックカウンタにカウントの開始又は停止が設定される。

【００５８】

カウントの開始設定によりカウントは開始され、カウントは各演算部１１＿１～１１＿Ｎの動作周波数を基にして実行される。一方、カウントの停止設定によりカウントは停止される。

【００５９】

イベントジェネレータ１６＿１＿１～１６＿２＿Ｎはイベントを検出すると、クロックカウンタがカウントする時刻を、タイムスタンプ値として読み出し、タイムスタンプ値がイベントジェネレータ１６＿１＿１～１６＿２＿Ｎに送信される。

【００６０】
　また、イベントの検出は例えば、データと並走する信号のうち、有効なデータであるか否かを示す信号のON/OFF用いて判定したり、データの特定の領域にイベント検出用のフィールドを用意して、当該フィールドのビット列を用いたりすることで、イベントを検出する。
【００６１】
　ここで、必要に応じて、演算部（ＦＰＧＡ）間で同期させる。なお、演算部間で同期させる方法として、ホスト部から同期をする演算部に対して同期させるための信号や、リセット信号、を入力することで、同期を図る。
【００６２】
　また、ホスト部１２は、トレースバッファ１８からトレースデータを読み出す時に、タイムスタンプ部１７にリセット信号を送信し、クロックカウンタの値をリセットする。
【００６３】
　トレースバッファ１８において、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎより受信されたトレースデータが、トレースバッファ１８に記録され蓄積される。
【００６４】
　ここで、複数のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎから送信されたトレースデータが記録される。また、トレースデータの書き込み、読み出しは全ＴＩＤで共通のＦＩＦＯ（Ｆｉｒｓｔ－Ｉｎ　Ｆｉｒｓｔ－Ｏｕｔ）で実行される。
【００６５】
　次に、ホスト部１２が、トレースデータを、トレースバッファ１８から読み出す。
【００６６】
　最後に、ホスト部１２で、読み出した（回収）データの後処理を実行する。詳細には、ＴＩＤごとに検索（ＧＲＥＰ）する。引き続き、タイムスタンプ部１７でソートした後に可視化する。
【００６７】
　ここでは、イベントジェネレータが、イベント検出時に入力データから、イベント種別と、データ種別と、任意のデータを取得する例を示したが、イベント種別と、データ種別と、任意のデータを取得しなくとも、上述の通り、少なくともタイムスタンプ値を取得すればコンピュータシステム１０を動作できる。
【００６８】
＜演算部および演算器の追加・削除＞
　コンピュータシステム１０において、演算部１１＿１～１１＿Ｎを、コンピュータシステム１０の停止中・稼働中を問わず、追加または削除できる。
【００６９】
　また、例えば、図３Ａに示す演算部１１＿１において、図３Ｂに示すように、イベントジェネレータ１６＿１＿２～１６＿２＿Ｎを新たに配置（追加）してタイムスタンプ部１７とトレースバッファ１８と接続することにより、演算器１５＿１（１）～１５＿Ｎ（１）を追加できる。
【００７０】
　また、演算部１１＿１において、イベントジェネレータ１６＿１＿２～１６＿２＿Ｎを削除することにより、演算器１５＿１（１）～１５＿Ｎ（１）を削除できる。
【００７１】
　演算器１５＿１（１）～１５＿Ｎ（１）を追加するとき、トレース部１４＿１～１４＿Ｎにおいて、複数のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎを任意の箇所に配置することができる。
【００７２】
　ここで、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎを演算器１５＿１（１）～１５＿Ｎ（１）の入力の前段と出力の後段との両方に配置してもよく、入力の前段と出力の後段とのいずれか一方に配置してもよい。

【００７３】
演算器１５＿１（１）～１５＿Ｎ（１）を削除するとき、トレース部１４＿１～１４＿Ｎにおいて、演算器１５＿１（１）～１５＿Ｎ（１）の前後のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎを削除すればよい。
【００７４】
ここで、削除する演算器１５＿１（１）～１５＿Ｎ（１）の入力の前段と出力の後段との両方のイベントジェネレータを削除してもよく、入力の前段と出力の後段とのいずれか一方のイベントジェネレータを削除してもよい。
【００７５】
本実施の形態によれば、複数のイベントジェネレータを任意の箇所に追加できるため、イベントジェネレータのみを新たに追加することにより、容易に演算部の内部に新たに演算器を追加でき、演算器における処理時間の計測、トレースデータの収集が可能となる。
【００７６】
また、複数のイベントジェネレータを任意の箇所から削除できるため、イベントジェネレータのみを削除すれば、容易に演算部の内部で演算器を削除できる。
【００７７】
また、演算器を追加する場合にイベントジェネレータとタイムスタンプ部とトレースバッファとを全て追加する場合に比べて、回路規模を縮小でき、消費電力を抑制できる。
【００７８】
また、演算部から演算器を削除する場合、演算器の前後に配置されているイベントジェネレータを削除してもよい。このとき、イベントジェネレータに関するトレースデータを削除してもよい。また、削除されるイベントジェネレータで検出したイベントに付随するトレースデータは、必ずしも削除しなくともよい。
【００７９】
＜第１の実施例＞
本発明の第１の実施例に係るコンピュータシステム１０の制御方法について、図４Ａ、Ｂを参照して説明する。
【００８０】
本実施例では、コンピュータシステム１０において、演算部１１＿１～１１＿Ｎのトレース部１４＿１～１４＿Ｎが記録するトレースデータを用いて、不具合時に効率的に処理を再開する。ここで、不具合とは、通常発生し得るパケットロスや、演算部１１＿１～１１＿Ｎの内部の機能ブロックにおける処理のスタックなどである。
【００８１】
コンピュータシステム１０において、例えば、演算部１１＿１のトレース部１４＿１は、トレースバッファ１８で任意のトレースデータとして、タイムスタンプ値と、インスタンスＩＤ（イベントを検出する箇所を示す情報）と、任意のデータとを記録する。
【００８２】
本実施例に係るコンピュータシステム１０の制御方法の一例として、ホスト部１２が制御（管理）する場合を、図４Ａを参照して説明する。
【００８３】
初めに、ホスト部１２が所定の周期でコンピュータシステムのトレースバッファ１８をモニタして、演算部内の処理時間を計測する（ステップＳ１１Ａ）。
【００８４】
ここで、処理時間の計測において、まず、ホスト部１２が、演算器１５＿１（１）の入力側のイベントジェネレータ１６＿１＿１（第１のイベントジェネレータ）と、演算器１５＿１（１）の出力側のイベントジェネレータ１６＿２＿１（第２のイベントジェネレータ）それぞれの所定のイベント（第１のイベント、第２のイベント）の検出を契機とする第１のタイムスタンプ値と第２のスタンプ値を、トレースバッファ１８から読み出す（取得する）。
【００８５】

引き続き、ホスト部１２が、第１のタイムスタンプ値と第２のスタンプ値との差分より、処理時間を算出する。

【００８６】

このように、任意の箇所（区間、例えば、演算器１５＿１（１））の処理時間は、任意の箇所（区間、例えば、演算器１５＿１（１）の前後）に配置されたイベントジェネレータそれぞれを入力データが通過する時刻（第１のタイムスタンプ値と第２のスタンプ値）の差分により取得される。

【００８７】

次に、計測された処理時間を、予め設定された所定の閾値と比較する（ステップＳ１２Ａ）。その結果、処理時間が所定の閾値より長い場合に、不具合が発生したと判定する。

【００８８】

最後に、不具合が発生した場合には、インスタンスＩＤより処理検出箇所を把握して、この処理検出箇所よりも前段のいずれかのトレースバッファ１８に記録される任意のデータを用いて、処理を再開する（ステップＳ１３Ａ）。

【００８９】

また、本実施例に係るコンピュータシステム１０の制御方法の一例として、演算部１１＿１～１１＿Ｎが制御（管理）する場合を、図４Ｂ、Ｃを参照して説明する。

【００９０】

初めに、演算部１１＿１～１１＿Ｎは、トレースデータを記録するとともに、所定の周期でトレースバッファ１８を監視し、任意の演算部（例えば演算部１１＿１）のトレースバッファ１８にタイムスタンプ値が記録されてから、次段の演算部（例えば演算部１１＿２）の演算器１５＿１（２）にデータが入力されるまでの時間（以下、「演算部間の処理時間」という。）を計測する（ステップＳ１１Ｂ）。

【００９１】

この演算部間の処理時間の計測では、例えば、図４Ｃに示すように、まず、演算部１１＿１のトレースバッファ１８の前段のイベントジェネレータ１６＿２＿１がタイムスタンプ値（第１のスタンプ値）を取得し、演算部１１＿１のトレースバッファ１８で記録される。

【００９２】

引き続き、演算部１１＿１のトレースバッファ１８から演算部１１＿２の演算器１５＿１（２）に通知信号が送信され（図中、点線矢印）、この信号を契機に、演算部１１＿１の演算器１５＿１（１）から演算部１１＿２の演算器１５＿１（２）にデータが転送される。

【００９３】

引き続き、演算部１１＿２の演算器１５＿１（２）に入力されるデータのイベントの検出を契機に、演算部１１＿２の演算器１５＿１（２）の前段のイベントジェネレータ１６＿１＿１が、タイムスタンプ値（第２のスタンプ値）を取得し、演算部１１＿２のトレースバッファ１８で記録される。

【００９４】

これらの第１のスタンプ値と第２のスタンプ値の差分より、演算部間の処理時間が計測される。

【００９５】

次に、計測された処理時間を、予め設定された所定の閾値と比較する（ステップＳ１２Ｂ）。その結果、計測された時間が所定の閾値より長い場合に、不具合が発生したと判定する。

【００９６】

最後に、不具合が発生した場合には、トレースバッファ１８に記録されるインスタンスＩＤにより把握される処理検出箇所よりも前段のいずれかのトレースバッファ１８に記録される任意のデータを用いて、処理を再開する（ステップＳ１３Ｂ）。

【００９７】

ここで、トレースバッファに記録したタイムスタンプ値を用いて処理時間を計測する例を示したが、イベントジェネレータが取得したタイムスタンプ値を直接用いて処理時間を計測してもよい。

【００９８】

このように、本実施例に係るコンピュータシステムの制御方法では、任意の位置に配置されるイベントジェネレータが、所定のイベントを検出し、これを契機として、一のタイムスタンプ値を取得し、他の位置に配置されるイベントジェネレータが、同様に他のタイムスタンプ値を取得し、一のタイムスタンプ値と、他のタイムスタンプ値との差分を算出することにより、不具合の発生を判定し、処理を再開する。

【００９９】

本実施例に係るコンピュータシステムの制御方法によれば、トレース部１４＿１～１４＿Ｎが記録するトレースデータを用いることにより、不具合が発生した場合に、処理を最初から再開せずに、正常に動作していた箇所まで遡り処理を再開できる。

【０１００】

また、トレースデータの処理を再開させる契機は、ホスト部１２が管理する。または、演算部１１＿１～１１＿Ｎが管理してもよい。

【０１０１】

また、処理の再開は、必ずしも特定の機能ブロックに限定しなくともよく、例えば、演算部の入力まで遡って処理を再開してもよい。

【０１０２】

本実施例では、インスタンスＩＤにより処理検出箇所を把握する例を示したが、これに限らず、予め設定される演算器の処理速度と、計測されるタイムスタンプ値とを用いて処理検出箇所を導出してもよい。

【０１０３】

本実施例では、不具合の発生を把握するために、処理時間を用いる例を示したが、これに限らず、データ流量（後述）などを用いてもよい。

【０１０４】

本実施例では、トレースデータは、データ種別、イベント種別を有してもよい。また、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。

【０１０５】

＜第２の実施例＞

本発明の第２の実施例に係るコンピュータシステムの制御方法について、図５を参照して説明する。本実施例では、コンピュータシステムにおいて、演算部１１＿１～１１＿Ｎのトレース部１４＿１～１４＿Ｎを用いて、システムの品質管理（ステート管理／ヘルスチェック）を実行する。

【０１０６】

コンピュータシステム１０において、例えば、演算部１１＿１のトレース部１４＿１は、複数のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎを備える。

【０１０７】

図５に、本実施例に係るコンピュータシステム１０の制御方法のフローチャート図を示す。

【０１０８】

初めに、例えば、コンピュータシステム１０の演算部１１＿１は、複数のイベントジェネレータ１６＿１＿１～１６＿２＿Ｎを用いて、データが各イベントジェネレータ１６＿１＿１～１６＿２＿Ｎを通過する時刻を収集する（ステップＳ２１）。

【０１０９】

詳細には、演算部１１＿１が、異なる箇所に配置されるイベントジェネレータ（例えば、イベントジェネレータ１６＿１＿１とイベントジェネレータ１６＿２＿１）それぞれでの所定のイベントの検出に基づき、タイムスタンプ値（例えば、第１のタイムスタンプ値と第２のタイムスタンプ値）を収集する。

【０１１０】
次に、収集された第１のタイムスタンプ値と第２のタイムスタンプ値との差分を求めることにより、演算部１１＿１内の特定区間の通過に要する時間すなわち処理時間を計算する（ステップＳ２２）。
【０１１１】
次に、処理時間を、予め設定された所定の閾値と比較する（ステップＳ２３）。
【０１１２】
比較の結果、処理時間が閾値より大きい場合に、コンピュータシステム１０の異常検知を通知する（ステップＳ２４）。
【０１１３】
このように、本実施例に係るコンピュータシステムの管理方法によれば、コンピュータシステムが正常に稼働しているか否かをモニタできる。
【０１１４】
また、必ずしもすべてのデータに対して時間の解析処理を実行しなくともよい。例えば、予め指定した計測間隔で演算部１１＿１～１１＿Ｎ内の特定区間の通過に要した時間（処理時間）を観測し、所定の範囲内に収まるか、もしくは所定の閾値よりも処理時間が伸びているか否かを判定することにより、コンピュータシステムが正常に稼働しているか否かをモニタできる。
【０１１５】
また、テスト用のデータを入力して、このデータが演算部１１＿１～１１＿Ｎの内部を通過するのに要する時間を解析してもよい。
【０１１６】
本実施例では、イベントジェネレータが取得したタイムスタンプ値を直接用いて処理時間を計測する例を示したが、トレースバッファにタイムスタンプ値を記録した後に、トレースバッファに記録したタイムスタンプ値を用いて処理時間を計測してもよい。
【０１１７】
本実施例では、処理時間を用いてコンピュータシステムの状況を把握する例を示したが、これに限らず、データ流量などを用いてもよい。
【０１１８】
本実施例では、演算部がコンピュータシステムを制御する例を示したが、ホスト部がコンピュータシステムを制御してもよい。
【０１１９】
本実施例では、トレースデータは、タイムスタンプ値とともに、データ種別を有してもよい。さらに、インスタンスＩＤ、イベント種別、任意のデータを有してもよい。また、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。
【０１２０】
＜第３の実施例＞
本発明の第３の実施例に係るコンピュータシステム１０の制御方法について、図６を参照して説明する。本実施例では、コンピュータシステム１０において、演算部１１＿１～１１＿Ｎのトレース部１４＿１～１４＿Ｎを用いて、コンピュータシステム１０のフロー管理を実行する。
【０１２１】
コンピュータシステム１０において、演算部１１＿１～１１＿Ｎのトレース部１４＿１～１４＿Ｎがトレースデータとして、タイムスタンプ値とインスタンスＩＤ（イベントを検出する箇所を示す情報）をトレースバッファ１８に記録し、ホスト部１２がトレースデータを読み出す。
【０１２２】
図６に、本実施例に係るコンピュータシステム１０の制御方法のフローチャート図を示す。
【０１２３】

初めに、ホスト部１２が、トレースバッファ１８よりトレースデータとして、例えば、任意の箇所でイベントジェネレータ１６＿１が異なるイベントで取得したタイムスタンプ値とインスタンスＩＤ（イベントを検出する箇所を示す情報）とを収集する（ステップＳ３１）。

【０１２４】

詳細には、イベントジェネレータ１６＿１において、通過するデータのイベント、例えば先頭および終端それぞれの検出を契機として取得され、トレースバッファ１８に記憶された先頭および終端のタイムスタンプ値を収集する。

【０１２５】

次に、先頭のタイムスタンプ値と終端のタイムスタンプ値との差分をデータの通過時間として算出する。

【０１２６】

次に、予め設定されている入力データ（または出力データ）のデータ量を、データの通過時間で除することにより、所定の箇所における単位時間当たりのデータ量（データ流量）を計算する（ステップＳ３２）。

【０１２７】

次に、データ流量を、予め設定された所定の閾値と比較する（ステップＳ３３）。

【０１２８】

比較の結果、データ流量が閾値より大きい場合に、データの集中を回避するようにデータフロー（経路）を設定する。例えば、経路を割り当てる際に、インスタンスＩＤ（イベントを検出する箇所を示す情報）により把握される所定の閾値を超えた経路を回避して、経路を設定する（ステップＳ３４）。

【０１２９】

このように、本実施例に係るコンピュータシステムの管理方法によれば、データの集中を回避するようにデータフロー（経路）を設定できる。

【０１３０】

本実施例で、トレースバッファに記録したタイムスタンプ値を用いて処理時間を計測する例を示したが、イベントジェネレータが取得したタイムスタンプ値を直接用いて処理時間を計測してもよい。

【０１３１】

また、ホスト部がコンピュータシステムを制御する例を示したが、演算部がコンピュータシステムを制御してもよい。

【０１３２】

また、トレースデータが、データ種別に関する情報を有すれば、ホスト部１２は、データ種別ごとの稼働状況を把握できる。

【０１３３】

これにより、コンピュータシステム１０では、特定のフローにのみデータが集中している場合に、当該フローから他の負荷の低いフローにデータを移行させることにより、データの集中を回避でき、フロー管理を実行できる。

【０１３４】

また、トレースデータは、インスタンスＩＤ、イベント種別、任意のデータが記録されてもよい。また、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。

【０１３５】

また、特定のフローや特定の箇所に発生した障害を検出すれば、障害が発生した際に、当該フローを迂回する経路を設定するようにフローを管理できる。

【０１３６】

また、データの経路が異なる複数のフローを有する演算部において不具合が発生した場合に、当該フロー以外のフローを他の演算部に回避させた後に、当該演算部の交換、リセット、解析などを実施できる。

【０１３７】
＜演算部の計測例＞
本実施例に係るコンピュータシステム１０の制御方法において、演算部における処理時間とデータ流量との計測の一例を、図７Ａ、Ｂを参照して説明する。
【０１３８】
本計測例では、例えば、演算器１５＿１（１）の前段のイベントジェネレータ１６＿１＿１で入力データが観測され、後段のイベントジェネレータ１６＿２＿１で入力データが観測される。
【０１３９】
入力データの先頭がイベントジェネレータ１６＿１＿１を通過する時をイベントの契機として、イベントジェネレータ１６＿１＿１に入力データの先頭のタイムスタンプ値が取得される。
【０１４０】
同様に、入力データの終端がイベントジェネレータ１６＿１＿１を通過する時をイベントの契機として、イベントジェネレータ１６＿１＿１に入力データの終端のタイムスタンプ値が取得される。
【０１４１】
一方、出力データの先頭がイベントジェネレータ１６＿２＿１を通過する時をイベントの契機として、イベントジェネレータ１６＿２＿１に出力データの先頭のタイムスタンプ値が取得される。
【０１４２】
同様に、出力データの終端がイベントジェネレータ１６＿２＿１を通過する時をイベントの契機として、イベントジェネレータ１６＿２＿１に出力データの終端のタイムスタンプ値が取得される。
【０１４３】
図７Ａに、トレースデータの一例を示す。図７Ａにおいて、タイムスタンプ値（Ｔｉｍｅｓｔａｍｐ：Ｄｅｃ、Ｔｉｍｅｓｔａｍｐ：０ｘ）、インスタンスＩＤ（Ｉｎｓ）、イベントＩＤ（Ｅｖｔ）、デコード化されたイベントＩＤ（Ｄｅｃ）、ＴＩＤ、イベントデータ（ＥｖｅｎｔＤａｔａ）を示す。
【０１４４】
デコード化されたイベントＩＤ（Ｄｅｃ）において、Ｈはデータの先頭、Ｌはデータの終端を示す。
【０１４５】
入力データと出力データのデータ量は１ＭＢである。また、演算部１１＿１～１１＿Ｎの動作周波数は２５０ＭＨｚ、４ｎｓ／サイクル）である。
【０１４６】
入力データの先頭のタイムスタンプ値（Ｔｉｍｅｓｔａｍｐ：Ｄｅｃ）は、「４０６５１４」であり、インスタンスＩＤ（Ｉｎｓ）が「１０」である（点線四角４１内上段）。また、イベントＩＤ（Ｄｅｃ）の「Ｈ－Ｒ－」がイベント発生としてデータ先頭の通過を示す。
【０１４７】
同様に、出力データの先頭のタイムスタンプ値（Ｔｉｍｅｓｔａｍｐ：Ｄｅｃ）は、「４０１６５６」であり、インスタンスＩＤ（Ｉｎｓ）が「１１」である（点線四角４１内下段）。また、イベントＩＤ（Ｄｅｃ）の「Ｈ－Ｒ－」がイベント発生としてデータ先頭の通過を示す。
【０１４８】
また、入力データの終端のタイムスタンプ値（Ｔｉｍｅｓｔａｍｐ：Ｄｅｃ）は、「５４７７９１」であり、インスタンスＩＤ（Ｉｎｓ）が「１０」である（点線四角４２内上段）。また、イベントＩＤ（Ｄｅｃ）の「－ＬＲ－」がイベント発生としてデータ終端の通過を示す。

【０１４９】
同様に、出力データの先頭のタイムスタンプ値（Ｔｉｍｅｓｔａｍｐ：Ｄｅｃ）は、「５４７７９４」であり、インスタンスＩＤ（Ｉｎｓ）が「１１」である（点線四角４２内下段）。また、イベントＩＤ（Ｄｅｃ）の「－ＬＲ－」がイベント発生としてデータ終端の通過を示す。
【０１５０】
図７Ｂに、入力データ４３と出力データ４４との関係を模式的に示す。入力データ４３がイベントジェネレータ１６＿１＿１を通過する時間は、終端のタイムスタンプ値（５４７７９１サイクル）と先頭のタイムスタンプ値（４０１５６４サイクル）との差分（矢印４５）より、１４６２２７サイクル＝５８４．９μｓｅｃと算出される。したがって、入力スループットすなわちデータ流量として、１ＭＢ／５８４．９μｓｅｃ＝約１．８ＧＢ／ｓｅｃが得られる。
【０１５１】
同様に、出力データ４４の終端のタイムスタンプ値と先頭のタイムスタンプ値との差分（矢印４６）より、入力スループットすなわちデータ流量を算出できる。
【０１５２】
このように、演算部において、データ流量は、データ量を、データのタイムスタンプ値の先頭と終端との差分で除することにより得られる。
【０１５３】
また、処理時間（レイテンシ）として、出力開始と入力開始のタイムスタンプ値の差分すなわち出力データ４４の先頭のタイムスタンプ値（４０１６５６サイクル）と入力データ４３の先頭のタイムスタンプ値（４０１５６４サイクル）との差分より、９２サイクルが得られる。
【０１５４】
このように、演算部において、入力データの処理時間は、出力開始と入力開始のタイムスタンプ値の差分より得られる。
【０１５５】
以上のように、本発明に係るコンピュータシステムの制御方法において、入力データと出力データのタイムスタンプ値を用いて、データの処理時間とデータ流量とを取得できる。
【０１５６】
＜効果＞
本発明の実施の形態および実施例に係るコンピュータシステムおよびその管理方法によれば、コンピュータシステムにおける演算処理が、途中で停止する場合または正常に処理が完了しない場合に、複数の演算部のうち、処理が停止した演算部を容易に検出、特定できる。
【０１５７】
また、トレースバッファ１８がデータを記録しているため、処理の途中から再開することができる。その結果、すでに実行した処理を再度、最初から繰り返す必要がなくなる。また、正常に処理が完了しない場合に、処理時間を短縮できる。
【０１５８】
また、ホスト部１２で各演算部の状態を一元管理できるため、例えば、処理が停止した演算部を経由しないデータのフローを設定でき、正常に処理が完了しないデータの数を削減できる。
【０１５９】
また、データ種別（ユーザＩＤ、セッションＩＤ、ストリームＩＤ、サービスＩＤ）ごとにイベントを検出できトレースデータを蓄積できるため、容易に、特定のデータ種別に着目して品質管理や不具合解析等を実行できる。
【０１６０】
また、演算部と独立してトレース部１４＿１～１４＿Ｎを備えるため、演算部の異常状態を保持できる。

【０１６１】
　また、ユーザ毎（セッション毎）の粒度でフローを管理できる。
【０１６２】
　また、イベント検出するイベントジェネレータを任意の部分に挿入できるので、特定のフローのみに発生する不具合を検出できる。
【０１６３】
＜第２の実施の形態＞
　本発明の第２の実施の形態に係るコンピュータシステムおよびその制御方法を、図８を参照して説明する。本実施の形態に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０１６４】
　第１の実施の形態に係るコンピュータシステム１０では、トレースバッファ１８がオーバーフローして、トレースデータを記録することが困難となる。そこで、トレースデータを消去する必要がある。
【０１６５】
＜コンピュータシステムの制御方法＞
　図８に、本実施の形態に係るコンピュータシステムの制御方法のフローチャート図を示す。
【０１６６】
　本実施の形態に係るコンピュータシステムでは、トレースバッファ１８に、トレースデータとして、タイムスタンプ値とともに、少なくともデータ種別が記録される。ここで、トレースデータとして、インスタンスＩＤ、イベント種別、任意のデータが記録されてもよい。または、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。
【０１６７】
　初めに、ホスト部１２が、演算部１１＿１～１１＿Ｎのトレースバッファ１８を所定の周期で監視する（ステップＳ５１）。
【０１６８】
　次に、トレースバッファ１８内に同一のデータ種別に対するトレースデータが複数記録されているか否かを判定する（ステップＳ５２）。
【０１６９】
　判定の結果、トレースバッファ１８内に同一のデータ種別に対するトレースデータが複数記録されている場合には、この複数のトレースデータのうち、タイムスタンプ値が最新のトレースデータを保持（記録）するとともに、過去に記録したトレースデータ（最新のトレースデータ以外のトレースデータ）を消去する（ステップＳ５３）。このように、前記イベントの検出時刻に基づき、前記トレースバッファ内に記録された前記イベントの検出時刻を消去する。
【０１７０】
　本実施の形態に係るコンピュータシステムおよびその制御方法によれば、トレースバッファ１８の蓄積容量が有限であっても、バッファのオーバーフローを抑制でき、物理的なバッファ量を削減できる。
【０１７１】
　また、バッファ量を過剰に搭載する必要がないので、演算部の消費電力削減でき、電力効率を向上できる。
【０１７２】
　また、データ種別（ユーザＩＤ、セッションＩＤ　、ストリームＩＤ　、サービスＩＤ）ごとにトレースデータを記録することにより、特定のデータ種別（例えば最高優先のサービスなど）については、比較的長時間トレースデータを保持することで信頼性を高めるなど、柔軟性の高いコンピュータシステムを提供できる。
【０１７３】

また、本実施の形態は、当然、第１の実施の形態と同様の効果を奏する。
【０１７４】
＜第２の実施の形態の変形例１＞
本発明の第２の実施の形態の変形例１に係るコンピュータシステムおよびその制御方法を説明する。本変形例に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０１７５】
本変形例に係るコンピュータシステムでは、トレースバッファ１８に、トレースデータとして、タイムスタンプ値とともに、少なくともデータ種別が記録される。ここで、トレースデータとして、インスタンスＩＤ、イベント種別、任意のデータが記録されてもよい。または、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。
【０１７６】
本変形例では、コンピュータシステム１０の演算部１１＿１～１１＿Ｎが、イベントジェネレータ１６＿１＿１～１６＿２＿Ｎより受信されたトレースデータ（最新のトレースデータ）を、トレースバッファ１８に記録する（書き込む）前に、トレースバッファ１８が既に保持しているトレースデータの中に、最新のトレースデータと同一のデータ種別のトレースデータが記録されているか否かを判定する。
【０１７７】
判定の結果、同一のデータ種別のトレースデータが記録されている場合には、既に保持しているデータ（最新のデータ以外のデータ）を消去し、引き続き、最新のトレースデータを記録する、すなわち最新のトレースデータを上書きする。このように、前記イベントの検出時刻に基づき、前記トレースバッファ内に記録された前記イベントの検出時刻を上書きする。
【０１７８】
ここで、上書きを防ぐためにフラグ等を付与し、フラグの有無で上書きの可否を判定してもよい。
【０１７９】
これにより、本実施の形態と同様の効果を奏する。
【０１８０】
＜第２の実施の形態の変形例２＞
本発明の第２の実施の形態の変形例２に係るコンピュータシステムおよびその制御方法を説明する。本変形例に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０１８１】
本変形例に係るコンピュータシステムでは、トレースバッファ１８に、トレースデータとして、少なくともタイムスタンプ値が記録される。ここで、トレースデータとして、データ種別、インスタンスＩＤ、イベント種別、任意のデータが記録されてもよい。または、トレースデータが、データ種別またはイベント種別ごとに記録されてもよい。
【０１８２】
本変形例では、初めに、前段の演算部１１＿１の演算器１５＿１（１）から後段の演算部１１＿２の演算器１５＿１（２）に、演算器１５＿１（１）で処理されたデータが送信される。
【０１８３】
次に、後段の演算部１１＿２が、前段の演算部１１＿１に受信完了通知を送付する。
【０１８４】
最後に、前段の演算部１１＿１が後段の演算部１１＿２から受信完了通知を受信すると、前段の演算部１１＿１のトレースバッファ１８のデータを消去する。
【０１８５】
または、後段の演算部１１＿２が、ホスト部１２に受信完了通知を送付して、ホスト部１２の指示により前段の演算部１１＿１のトレースバッファ１８のデータを消去してもよ

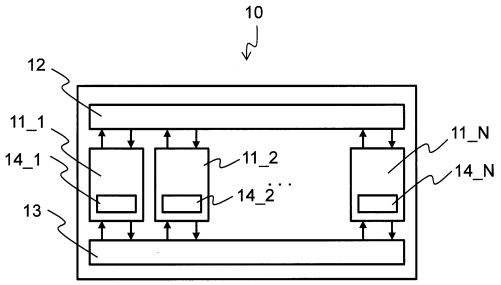
い。
【０１８６】
これにより、本実施の形態と同様の効果を奏する。
【０１８７】
また、本実施の形態において、トレースバッファ１８より大きい記憶容量を有するホスト部１２が、トレースバッファ１８のデータを回収し記録する場合には、ホスト部１２によるデータの回収または回収の完了を契機として、トレースバッファ１８のデータを消去してもよい。
【０１８８】
また、本実施の形態において、予め設定した時間の経過を契機として、トレースバッファ１８のデータを消去してもよい。
【０１８９】
＜第３の実施の形態＞
本発明の第３の実施の形態に係るコンピュータシステムおよびその制御方法を説明する。本実施の形態に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０１９０】
第１の実施の形態に係るコンピュータシステム１０では、複数の演算部が連動して動作するので、複数の演算部が各々計測するタイムスタンプの時刻を同期させることが困難である。
【０１９１】
とくに、各演算部の動作周波数が異なる場合、動作周波数をベースとしたクロックカウンタの値が異なるため、タイムスタンプの時刻を同期させることが困難である。
【０１９２】
＜コンピュータシステムの制御方法＞
本実施の形態に係るコンピュータシステムおよびその制御方法では、ホスト部１２が、タイムスタンプ値を所定の値、例えば初期値に一元的に設定する。図９に、本実施の形態に係るコンピュータシステムの制御方法のフローチャート図を示す。
【０１９３】
初めに、ホスト部１２がタイムスタンプ値の初期値として所定の値を設定する（ステップＳ６１）。
【０１９４】
ここで、演算部１１＿１～１１＿Ｎが入力データの計測を開始する時のタイムスタンプ値を初期値としてもよい。
【０１９５】
または、ホスト部１２が、タイムスタンプ部１７にカウントの開始を書き込み、カウントが開始される時のタイムスタンプ値を初期値としてもよい。
【０１９６】
次に、各演算部１１＿１～１１＿Ｎが、タイムスタンプ値を取得する（ステップＳ６２）。
【０１９７】
次に、ホスト部１２が、取得されたタイムスタンプ値と、タイムスタンプ値に対する所定の基準値とを比較する（ステップＳ６３）。
【０１９８】
ここで、タイムスタンプ値に対する所定の基準値は、予め設定される。所定の基準値はタイムスタンプ値の所定の値、例えば初期値に対するずれの許容範囲を示すものである。
【０１９９】
最後に、各演算部１１＿１～１１＿Ｎのタイムスタンプ値が所定の基準値の許容範囲を超えた場合に、ホスト部１２が、各演算部１１＿１～１１＿Ｎのタイムスタンプ値を所定の値、例えば初期値に再設定する（ステップＳ６４）。

【０２００】
本実施の形態に係るコンピュータシステムおよびその制御方法によれば、動作周波数の異なる演算部が共存する場合（例えば、演算部の演算内容によって動作周波数は異なる場合）に、タイムスタンプ値を統一的に扱えるので、トレースデータ回収後に容易に解析できる。
【０２０１】
また、複数の演算部を容易に同期させることが可能となるため、スケールアウト性を向上できる。
【０２０２】
また、ホスト部１２が、計測開始などを契機として一元的にタイムスタンプ値を所定の値、例えば初期値に設定するため、演算部間のクロックカウンタのずれを容易に補正できる。
【０２０３】
また、本実施の形態は、当然、第１の実施の形態と同様の効果を奏する。
【０２０４】
＜第３の実施の形態の変形例１＞
本発明の第３の実施の形態の変形例１に係るコンピュータシステムおよびその制御方法を説明する。本変形例に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０２０５】
本変形例では、各演算部１１＿１～１１＿Ｎで周波数が異なる場合に、ホスト部１２が各演算部１１＿１～１１＿Ｎの動作周波数の差を調整する。
【０２０６】
例えば、演算部１１＿１の周波数が１００ＭＨＺ、１クロックサイクルが１０ナノ秒であり、演算部１１＿２の周波数が２００ＭＨＺ、１クロックサイクルが５ナノ秒である場合に、ホスト部１２が演算部１１＿１のタイムスタンプ値を２倍にすることで、演算部１１＿１と演算部１１＿２のタイムスタンプ値を同期させる。
【０２０７】
また、ホスト部１２が演算部１１＿２のタイムスタンプ値を１／２倍することで、演算部１１＿１と演算部１１＿２のタイムスタンプ値を同期させてもよい。
【０２０８】
また、ホスト部１２がトレースデータを読み出した後に、換算基準値（例：１００ＭＨＺ）を設け、全演算部のカウンタ値を基準値１００ＭＨＺに合わせて換算してもよい。
【０２０９】
このように、本変形例に係るコンピュータシステムおよびその制御方法では、ホスト部が、一の演算部（例えば、演算部１１＿１）の動作周波数に対して、他の演算部（例えば、演算部１１＿２）の動作周波数が同一になるように他の演算部（例えば、演算部１１＿２）に設定される係数を、他の演算部（例えば、演算部１１＿２）のタイムスタンプ値に乗ずる。ここで、他の演算部は複数であってもよい。このように、演算部ごとに異なるカウンタ値の差を調整する。
【０２１０】
これにより、各演算部のカウンタ値が同一になるので、本実施の形態と同様の効果を奏する。
【０２１１】
＜第３の実施の形態の変形例２＞
本発明の第３の実施の形態の変形例２に係るコンピュータシステムおよびその制御方法を説明する。本変形例に係るコンピュータシステム１０は、第１の実施の形態と同様の構成を有する。
【０２１２】
本変形例では、各演算部１１＿１～１１＿Ｎで周波数が異なる場合は、各演算部１１＿

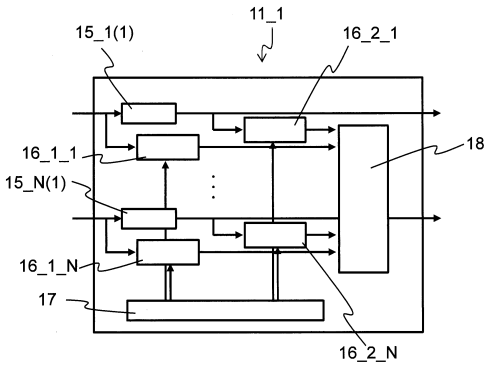
１～１１＿Ｎで動作周波数の差を調整する。
【０２１３】
まず、予め、演算器ごとに周波数に応じて異なる換算値（係数）を設定する。例えば、演算部１１＿１の周波数が１００ＭＨＺ、演算部１１＿２の周波数が２００ＭＨＺである場合に、基準値を１００ＭＨＺとすれば、演算部１１＿１の換算値（係数）は「１」、演算部１１＿２の換算値（係数）は「１／２」となる。
【０２１４】
次に、トレースバッファ１８にタイムスタンプ値を記録する前段で、タイムスタンプ値に換算値（係数）を乗じた値を記録する。
【０２１５】
また、トレースバッファ１８にタイムスタンプ値を記録した後に、クロックカウンタ値を読み出す際に同様に換算をしてもよい。
【０２１６】
このように、本変形例に係るコンピュータシステムおよびその制御方法では、演算部のトレースバッファの前段で、一の演算部（例えば、演算部１１＿１）の動作周波数に対して、他の演算部（例えば、演算部１１＿２）の動作周波数が同一になるように他の演算部（例えば、演算部１１＿２）に設定される係数を、他の演算部（例えば、演算部１１＿２）のタイムスタンプ値に乗ずる。ここで、他の演算部は複数であってもよい。このように、演算部ごとに異なるカウンタ値の差を調整する。
【０２１７】
これにより、各演算部のカウンタ値が同一になるので、本実施の形態と同様の効果を奏する。
【０２１８】
本発明の実施の形態で、演算部が処理時間、データ流量等の計測、不具合などの判定などを実行する場合、演算部における演算器が実行してもよいし、演算部内に別途計測、判定などの処理機能を設けてもよい。
【０２１９】
本発明の実施の形態では、第１の実施の形態と、第２の実施の形態と、第３の実施の形態とを組み合わせることにより、さらなる効果を奏することができる。
【０２２０】
本発明の実施の形態では、コンピュータシステムの構成、管理方法などにおいて、各構成部の構造、寸法、材料等の一例を示したが、これに限らない。コンピュータシステムの機能を発揮し効果を奏するものであればよい。
【産業上の利用可能性】
【０２２１】
本発明は、情報処理分野におけるコンピュータシステムに適用することができる。
【符号の説明】
【０２２２】
１０　コンピュータシステム
１１＿１～１１＿Ｎ　演算部
１２　ホスト部
１３　内部通信部
１４＿１～１４＿Ｎ　トレース部

【図面】

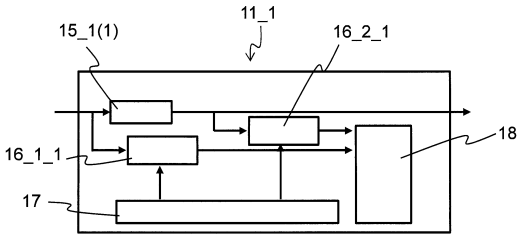
【図１】



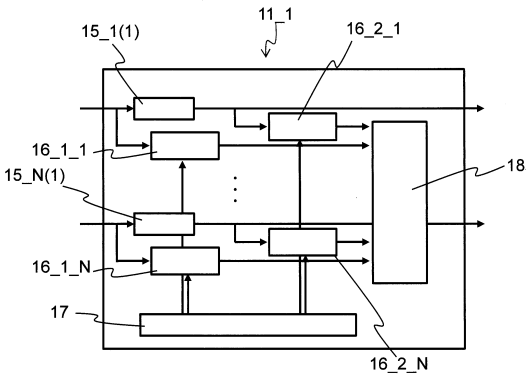
【図２】



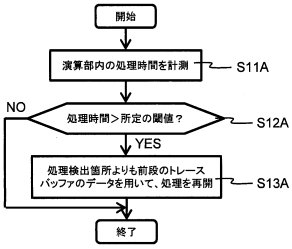
【図３Ａ】



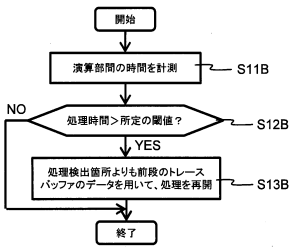
【図３Ｂ】



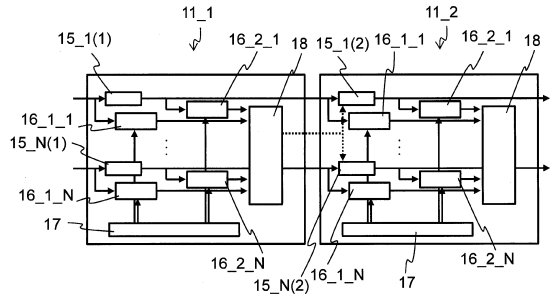
【図４Ａ】



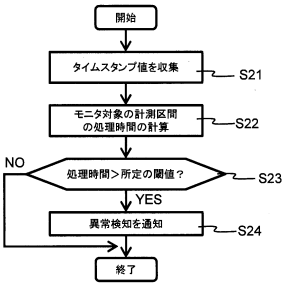
【図４Ｂ】



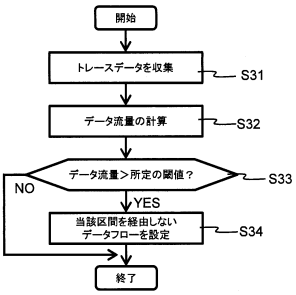
【図４Ｃ】



【図５】



【図６】



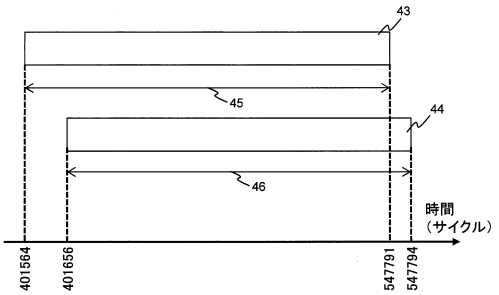
【図７Ａ】

Timestamp:Dec,	Timestamp:0x,	Ins,	Evt,	Dec,	TID ,	EventData
6983,	1b47,	f0,	07,	HLR-,	00000000,	00000000
6996,	1b54,	f1,	07,	HLR-,	00000000,	00000000
401563,	6209b,	f0,	05,	H-R-,	00000001,	00000000
401564,	6209c,	10,	05,	H-R-,	00000001,	00000000
401656,	62058,	11,	05,	H-R-,	00000001,	00000000
402995,	62633,	f1,	05,	H-R-,	00000000,	00000000
404019,	62a33,	f1,	06,	-LR-,	00000000,	00000000
418315,	6620b,	f1,	05,	H-R-,	00000000,	00000000
419339,	6660b,	f1,	06,	-LR-,	00000000,	00000000
428625,	68a51,	f1,	05,	H-R-,	00000000,	00000000
429649,	68e51,	f1,	06,	-LR-,	00000000,	00000000
438909,	6b27d,	f1,	05,	H-R-,	00000000,	00000000
439933,	6b67d,	f1,	06,	-LR-,	00000000,	00000000
448499,	6d7f3,	f1,	05,	H-R-,	00000000,	00000000
449523,	6dbf3,	f1,	06,	-LR-,	00000000,	00000000
457367,	6fa97,	f1,	05,	H-R-,	00000000,	00000000
458391,	6fe97,	f1,	06,	-LR-,	00000000,	00000000
466369,	71dc1,	f1,	05,	H-R-,	00000000,	00000000
467393,	721c1,	f1,	06,	-LR-,	00000000,	00000000
475235,	74063,	f1,	05,	H-R-,	00000000,	00000000
476259,	74463,	f1,	06,	-LR-,	00000000,	00000000
484240,	76390,	f1,	05,	H-R-,	00000000,	00000000
485264,	76790,	f1,	06,	-LR-,	00000000,	00000000
493267,	786d3,	f1,	05,	H-R-,	00000000,	00000000
494291,	78ad3,	f1,	06,	-LR-,	00000000,	00000000
502432,	7aaa0,	f1,	05,	H-R-,	00000000,	00000000
503456,	7aea0,	f1,	06,	-LR-,	00000000,	00000000
511523,	7ce23,	f1,	05,	H-R-,	00000000,	00000000
512547,	7d223,	f1,	06,	-LR-,	00000000,	00000000
520549,	7f165,	f1,	05,	H-R-,	00000000,	00000000
521573,	7f565,	f1,	06,	-LR-,	00000000,	00000000
529688,	81518,	f1,	05,	H-R-,	00000000,	00000000
530712,	81918,	f1,	06,	-LR-,	00000000,	00000000
538726,	83866,	f1,	05,	H-R-,	00000000,	00000000
539750,	83c66,	f1,	06,	-LR-,	00000000,	00000000
547790,	85bce,	f0,	06,	-LR-,	00000001,	00000000
547791,	85bcf,	10,	06,	-LR-,	00000001,	00000000
547794,	85bd2,	11,	06,	-LR-,	00000001,	00000000
547796,	85bd4,	f1,	05,	H-R-,	00000000,	00000000
548820,	85fd4,	f1,	06,	-LR-,	00000000,	00000000

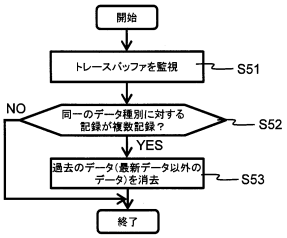
41

42

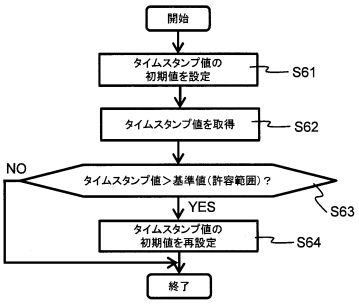
【図７Ｂ】


43
45
44
46
時間
（サイクル）
401564
401656
547791
547794

【図８】


開始
トレースバッファを監視
S51
NO
同一のデータ種別に対する
記録が複数記録？
S52
YES
過去のデータ（最新データ以外の
データ）を消去
S53
終了

【図９】


開始
タイムスタンプ値の
初期値を設定
S61
タイムスタンプ値を取得
S62
NO
タイムスタンプ値＞基準値（許容範囲）？
S63
YES
タイムスタンプ値の
初期値を再設定
S64
終了

フロントページの続き

東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内
(72)発明者 田仲 顕至
東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内
(72)発明者 伊藤 猛
東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内
(72)発明者 坂本 健
東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内
(72)発明者 村中 勇介
東京都千代田区大手町一丁目５番１号 日本電信電話株式会社内

審査官 真木 健彦

(56)参考文献
国際公開第２０１８／１８２７８２（WO，Ａ１）
米国特許出願公開第２０１２／０３３１３５４（ＵＳ，Ａ１）
特開２０１２－２５２６３６（ＪＰ，Ａ）
田仲 顕至 ほか，分散深層学習を高速化させるＦＰＧＡ Ｒｉｎｇ－Ａｌｌｒｅｄｕｃｅの検討，第８２回（２０２０年）全国大会講演論文集（１） コンピュータシステム ソフトウェア科学・工学 データとウェブ，日本，一般社団法人情報処理学会，2020年02月20日，7A-01，P. 1-31 ～ 1-32

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １１／２８ － １１／３６
Ｇ０６Ｆ １１／０７
Ｇ０６Ｆ １５／１６ － １５／１７７
Ｇ０６Ｆ １５／８２