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

特許第6446531号
(P6446531)

(45)発行日　平成30年12月26日(2018.12.26)　(24)登録日　平成30年12月7日(2018.12.7)

(51)Int.Cl.		F I		
G06F 17/30	***(2006.01)***	G06F	17/30	415
G06F 12/00	***(2006.01)***	G06F	17/30	220Z
		G06F	17/30	170Z
		G06F	12/00	518A

請求項の数 20　(全 28 頁)

(21)出願番号	特願2017-504017 (P2017-504017)
(86)(22)出願日	平成27年7月24日 (2015.7.24)
(65)公表番号	特表2017-529588 (P2017-529588A)
(43)公表日	平成29年10月5日 (2017.10.5)
(86)国際出願番号	PCT/US2015/041951
(87)国際公開番号	WO2016/014925
(87)国際公開日	平成28年1月28日 (2016.1.28)
審査請求日	平成29年12月19日 (2017.12.19)
(31)優先権主張番号	62/028,999
(32)優先日	平成26年7月25日 (2014.7.25)
(33)優先権主張国	米国(US)

(73)特許権者　509123208
アビニシオ　テクノロジー　エルエルシー
アメリカ合衆国　０２４２１　マサチューセッツ州　レキシントン　スプリング　ストリート　２０１
(74)代理人　100107984
弁理士　廣田　雅紀
(74)代理人　100102255
弁理士　小澤　誠次
(74)代理人　100096482
弁理士　東海　裕作
(74)代理人　100188352
弁理士　松田　一弘
(74)代理人　100131093
弁理士　堀内　真

最終頁に続く

(54)【発明の名称】ランダムに発生するイベントの遅延の適応のための変更可能な時系列

(57)【特許請求の範囲】

【請求項１】

イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義する前記イベントのシーケンスからのイベントをコンピューティングシステムに処理させるための方法であって、

第１の変数を定義するステップと、

前記第１の変数に関して、受け取られたイベントに関連する前記第１の変数に対する操作の第１の時系列を定義するステップと、

前記第１の変数に関する第１のイベントを受け取るステップと、

前記第１の変数に対して第１の操作を実行するステップであって、前記第１の操作が、前記第１の時系列の第１の更新をもたらす、ステップと、

前記第１のイベントを受け取った後に、前記第１の変数に関する遅延されたイベントを受け取るステップと、

前記第１の変数に対して第２の操作を実行するステップであって、前記第２の操作が、前記第１の時系列の第２の更新をもたらす、ステップと、

前記第１の更新が有効であるか又は無効であるかを決定するステップとを含み、

前記遅延されたイベントが、前記シーケンス内で前記第１のイベントに先立ち、

前記第１の更新が、前記第１のイベントに基づき、

前記第２の更新が、前記遅延されたイベントに基づく、前記方法。

【請求項２】

第１の更新が有効であるか又は無効であるかを決定するステップが、前記第１の更新が有効であると決定するステップを含む、請求項１に記載の方法。
【請求項３】
第１の更新が有効であると決定するステップが、前記第１の更新の前に、前記第１の更新が書き込み操作の挿入であると決定するステップを含む、請求項２に記載の方法。
【請求項４】
第１の更新が有効であると決定するステップが、前記第１の更新の前に、遅延されたイベントが読み取り操作の挿入をもたらしたと決定するステップを含む、請求項２に記載の方法。
【請求項５】
第１の更新が有効であると決定するステップが、前記第１の更新の前に、遅延されたイベントが更新操作の挿入を必要としたと決定するステップを含む、請求項２に記載の方法。
【請求項６】
第１の更新が有効であると決定するステップが、前記第１の更新が第２の更新によって影響を受けないと決定するステップを含む、請求項２に記載の方法。
【請求項７】
第１の更新が有効であるか又は無効であるかを決定するステップが、前記第１の更新が第２の更新によって影響を受けると決定するステップと、前記第１の更新が無効であると決定するステップに応じて前記第１の更新を撤回するステップとを含む、請求項１に記載の方法。
【請求項８】
第１の更新が有効であるか又は無効であるかを決定するステップが、前記第１の更新が第２の更新によって影響を受けると決定し、前記第１の更新を無効にさせるステップを含む、請求項１に記載の方法。
【請求項９】
第１の更新が第２の更新によって影響を受けると決定するステップが、前記第１の更新に対応する時系列のエントリの前の、第１の変数を書き込むことを含む操作を示す時系列のエントリの挿入を遅延されたイベントが必要としたと決定するステップを含む、請求項８に記載の方法。
【請求項１０】
第１の更新が第２の更新によって影響を受けると決定するステップが、前記第１の更新に対応する時系列のエントリの前の、第１の変数に対する操作を示す既存の時系列のエントリを更新することを遅延されたイベントが必要としたと決定するステップを含む、請求項８に記載の方法。
【請求項１１】
イベントのシーケンスの正しい順序が、不完全に順序付けられ、前記方法が、イベントの前記シーケンスを完全に順序付けさせるステップをさらに含む、請求項１に記載の方法。
【請求項１２】
第１のイベントを受け取るステップが、完全に順序付けられたイベントの組から前記イベントを受け取るステップを含む、請求項１に記載の方法。
【請求項１３】
第１の時系列内のデータの喪失を検出するステップと、
前記第１の時系列内の第１の変数の有効な値を示す第１のチェックポイント値を取り出すステップと、
前記チェックポイント値に基づいて、イベントのリストからイベントのサブセットを取り出すステップであって、イベントの前記リストが第１のイベントを含む、ステップと、
イベントの前記サブセットを再処理して、それによって、データの前記喪失の際に失われた前記データを再構築するステップと、

データの前記喪失の際に失われた前記データを含むように前記第１の時系列を改訂するステップとをさらに含む、請求項１に記載の方法。

【請求項１４】

第１の時系列が、遅延されたイベントに対応する遅延されたイベントの操作に対応するエントリの後にある複数のイベントの結果として生じる操作に対応する複数のエントリを含み、

第２の操作を実行するステップが、前記複数のエントリ内の無効性の伝播を遮断する防壁操作を特定するステップと、

エントリが前記遅延されたイベントの操作に対応するエントリと前記防壁操作に対応するエントリとの間にある複数の操作からの操作のみを改訂するステップとを含む、請求項１に記載の方法。

【請求項１５】

防壁操作を特定するステップが、書き込み操作を防壁操作として特定するステップを含む、請求項１４に記載の方法。

【請求項１６】

第１の更新が有効であるかどうかを決定するステップが、

前記第１の更新を正しく実行するために行われたに違いない第１の時系列へのアクセスを特定するステップと、

前記第１の時系列を調べるステップと、

前記調べるステップに基づいて、前記アクセスが行われたかどうかを決定するステップとを含む、請求項１に記載の方法。

【請求項１７】

第１の情報及び第２の情報を保有するステップであって、

前記第１の情報が、第１の時系列へのアクセスを特定し、

前記第２の情報が、第１のイベントを処理するために必要とされるアクセスの範囲を特定する、ステップと、

前記第１の情報がアクセスの前記範囲内にあると決定するステップとをさらに含む、請求項１６に記載の方法。

【請求項１８】

コンピューティングシステムが、ネットワークを介して通信する複数のノードを含み、前記方法が、前記コンピューティングシステムに、

第１の変数及び第１の時系列を第１のノードに記憶するステップと、

第２の変数、及び前記第２の変数に対する操作の第２の時系列を第２のノードに記憶するステップと、

前記第１の変数と前記第２の変数との両方に対して操作を実行することによってイベントのシーケンスの少なくとも一部のイベントを処理し、前記イベントを処理することに応じて前記第１の時系列及び前記第２の時系列の両方を更新するステップとを実行させるステップをさらに含む、請求項１に記載の方法。

【請求項１９】

イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義する前記イベントのシーケンスからのイベントをコンピューティングシステムに処理させるための、コンピュータ可読媒体上に非一時的形態で記憶されたソフトウェアであって、コンピューティングシステムに、

第１の変数を定義することと、

前記第１の変数に関して、受け取られたイベントに関連する前記第１の変数に対する操作の第１の時系列を定義することと、

前記第１の変数に関する第１のイベントを受け取ることと、

前記第１の変数に対して第１の操作を実行することであって、前記第１の操作が、前記第１の時系列の第１の更新をもたらす、実行することと、

前記第１のイベントを受け取った後に、前記第１の変数に関する遅延されたイベントを

受け取ることと、
前記第１の変数に対して第２の操作を実行することであって、前記第２の操作が、前記第１の時系列の第２の更新をもたらす、実行することと、
前記第１の更新が有効であるか又は無効であるかを決定することとを行わせるための命令を含み、
前記遅延されたイベントが、前記シーケンス内で前記第１のイベントに先立ち、
前記第１の更新が、前記第１のイベントに基づき、
前記第２の更新が、前記遅延されたイベントに基づく、前記ソフトウェア。
【請求項２０】
イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義する前記イベントのシーケンスからのイベントを処理するためのコンピューティングシステムであって、
前記イベントを受け取るように構成された入力デバイス又はポート、及び
イベントの前記シーケンスの処理に参加するように構成された少なくとも１つのプロセッサを含み、前記処理が、
第１の変数を定義することと、
前記第１の変数に関して、受け取られたイベントに関連する前記第１の変数に対する操作の第１の時系列を定義することと、
前記第１の変数に関する第１のイベントを受け取ることと、
前記第１の変数に対して第１の操作を実行することであって、前記第１の操作が、前記第１の時系列の第１の更新をもたらす、実行することと、
前記第１のイベントを受け取った後に、前記第１の変数に関する遅延されたイベントを受け取ることと、
前記第１の変数に対して第２の操作を実行することであって、前記第２の操作が、前記第１の時系列の第２の更新をもたらす、実行することと、
前記第１の更新が有効であるか又は無効であるかを決定することとを含み、
前記遅延されたイベントが、前記シーケンス内で前記第１のイベントに先立ち、
前記第１の更新が、前記第１のイベントに基づき、
前記第２の更新が、前記遅延されたイベントに基づく、前記コンピューティングシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願の相互参照
本出願は、米国特許出願第６２／０２８，９９９号明細書の２０１４年７月２５日の優先日の利益を主張するものである。
【０００２】
本開示は、イベント処理に関し、特に、イベントに応答して変数を更新することに関する。
【背景技術】
【０００３】
典型的なコンピュータシステムは、イベントを受け取り、それらのイベントに応答して状態を更新する。
【０００４】
状態は、さまざまな主体に関する口座残高を表す状態変数の組からなる可能性がある。その場合、イベントは、１つの主体によって行われた引き出し又は送金である可能性がある。イベントに応答して、状態変数のうちの１又は２以上を更新することによって状態を変更することが必要である可能性がある。
【０００５】
多くの場合、状態は、決定性の（deterministic）ものである。これは、特定の時間の

前のすべてのイベントが知られており、すべてのそれらのイベントが正しい順序で処理される場合に状態が知られることを意味する。

【０００６】

残念なことに、イベントが正しい順序で到着する保証はない。結果として、コンピュータがそうだと信じる状態である記録された状態（recorded state）は、コンピュータが関連するイベントを正しい順序で受け取っていたならばコンピュータがそう記録したであろう状態である実際の状態（actual state）と常に一致するとは限らない。

【０００７】

例えば、複合イベント処理（ＣＥＰ，complex event processing）システムにおいては、イベントのストリームが処理される一方、同時に、結果に基づいてアクションが行われる。ＣＥＰシステムのための作業データの組は、受け取られたイベントデータ（例えば、データベースの要求、気象データ、又は交通データなどのイベントデータ）の異なるそれぞれのストリームに基づいて操作されている異なる状態変数を含み得る。個々のイベントの元のシーケンスの順序は、実行される処理に対して影響がある可能性がある。場合によっては、イベントの処理がバラバラの順序で行われる場合、状態変数によって反映される状態が正しくない可能性がある。

【０００８】

状態を訂正するプロセスは、特にマルチコア又はマルチノード環境においてはそれほど簡単ではない。多くの場合、状態を更新することは、作業ストレージ内の複数の状態変数を更新することを伴う。これは、前に進むために互いに依存する２つの異なる更新の計算の間のデッドロックの危険を生じる。このデッドロックの危険を減らすために、一部のシステムは、並行性制御（concurrency control）の形態（例えば、ロック及び２相コミットを使用する悲観的並行性制御、又はロックを用いず、非ローカル検証手順を用いる楽観的並行性制御）を使用する。加えて、システム全体は、少なくともノード又はコアの故障をくぐり抜け、そのような故障の後に状態を復元するのに十分なだけの耐障害性があるべきである。

【０００９】

イベント処理のこれまでの手法は、複雑で時間のかかる分散型アルゴリズムを組み込む耐障害分散型データベースを含む。概して、そのようなシステムを構築し、サービスの高いピークレベルにおいてそのシステムが高い性能で（例えば、低いレイテンシーで）働くようにすることは難しい。例えば、そのシステムが数百万イベント毎秒のレベルにおいてミリ秒未満の応答性で動作するようにすることは、控え目に言っても、見通しが非常に厳しい。

【発明の概要】

【課題を解決するための手段】

【００１０】

一態様においては、概して、イベントが入力デバイス又はポートを介して受け取られる順序とは無関係であるそれらのイベントの正しい順序を定義するそれらのイベントのシーケンスからのイベントをコンピューティングシステムに処理させるための方法が、第１の変数を定義するステップと、第１の変数に関して、受け取られたイベントに関連する第１の変数に対する操作の第１の時系列を定義するステップと、第１の変数に関する第１のイベントを受け取るステップと、第１の時系列の第１の更新をもたらす、前記第１の変数に対して第１の操作を実行するステップと、第１のイベントを受け取った後に、第１の変数に関する遅延されたイベントを受け取るステップと、第１の時系列の第２の更新をもたらす、前記第１の変数に対して第２の操作を実行するステップと、第１の更新が有効であるか又は無効であるかを決定するステップとを含み、遅延されたイベントは、シーケンス内で第１のイベントに先立ち、第１の更新は、第１のイベントに基づき、第２の更新は、遅延されたイベントに基づく。

【００１１】

一部の実践において、第１の更新が有効であるか又は無効であるかを決定するステップ

は、第１の更新が有効であると決定するステップを含む。これらの実践の中には、第１の更新が有効であると決定するステップが、第１の更新の前に、第１の更新が書き込み操作の挿入であると決定するステップを含む実践と、第１の更新が有効であると決定するステップが、第１の更新の前に、遅延されたイベントが読み取り操作の挿入をもたらしたと決定するステップを含む実践と、第１の更新が有効であると決定するステップが、第１の更新の前に、遅延されたイベントが更新操作の挿入を必要としたと決定するステップを含む実践とがある。

【００１２】

方法の一部の実践において、第１の更新が有効であると決定するステップは、第１の更新が第２の更新によって影響を受けないと決定するステップを含む。

【００１３】

一部の実践において、第１の更新が有効であるか又は無効であるかを決定するステップは、第１の更新が第２の更新によって影響を受けると決定するステップと、第１の更新が無効であると決定するステップに応じて第１の更新を撤回するステップとを含む。

【００１４】

その他の実践において、第１の更新が有効であるか又は無効であるかを決定するステップは、第１の更新が第２の更新によって影響を受けると決定し、前記第１の更新を無効にさせるステップを含む。これらの実践の中には、第１の更新が第２の更新によって影響を受けると決定するステップが、第１の更新に対応する時系列のエントリの前の、前記第１の変数を書き込むことを含む操作を示す時系列のエントリの挿入を遅延されたイベントが必要としたと決定するステップを含む実践と、第１の更新が第２の更新によって影響を受けると決定するステップが、第１の更新に対応する時系列のエントリの前の、前記第１の変数に対する操作を示す既存の時系列のエントリを更新することを遅延されたイベントが必要としたと決定するステップを含む実践とがある。

【００１５】

イベントのシーケンスの正しい順序が不完全に順序付けられるいくつかの場合、方法の実践は、イベントを完全に順序付けさせるステップを含む。

【００１６】

代替的な実践において、第１のイベントを受け取るステップは、完全に順序付けられたイベントの組からイベントを受け取るステップを含む。

【００１７】

方法の実践の中には、第１の時系列内のデータ（例えば、故障に遭ったデバイス又はノードに記憶された第１の時系列の一部）の喪失を検出するステップと、第１の時系列内の第１の変数の有効な値を示す第１のチェックポイント値を取り出すステップと、チェックポイント値に基づいて、イベントのリストからイベントのサブセットを取り出すステップであって、イベントのリストが第１のイベントを含む、ステップと、イベントのサブセットを再処理して、それによって、データの喪失の際に失われたデータを再構築するステップと、データの喪失の際に失われたデータを含むように第１の時系列を改訂するステップとをさらに含む実践がある。

【００１８】

さらにその他の実践において、第１の時系列は、遅延されたイベントに対応する遅延されたイベントの操作に対応するエントリの後にある複数のイベントの結果として生じる操作に対応する複数のエントリを含む。これらの実践の一部は、第１の時系列の第２の更新を引き起こす第２の操作を実行するステップが、複数のエントリ内の無効性（invalidity）の伝播を遮断する防壁操作を特定するステップと、エントリが遅延されたイベントの操作に対応するエントリと防壁操作に対応するエントリとの間にある複数の操作からの操作のみを改訂するステップとを含む実践を含む。これらの実践の中には、防壁操作を特定するステップが、書き込み操作を防壁操作として特定するステップを含む実践がある。

【００１９】

代替的な実践において、第１の更新が有効であるかどうかを決定するステップは、第１

の更新を正しく実行するために行われたに違いない第１の時系列へのアクセスを特定するステップと、第１の時系列を調べるステップと、調べるステップに基づいて、アクセスが行われたかどうかを決定するステップとを含む。これらの実践の中には、第１の時系列へのアクセスを特定する第１の情報、及び第１のイベントを処理するために必要とされるアクセスの範囲を特定する第２の情報を保有するステップと、第１の情報がアクセスの範囲内にあると決定するステップとをさらに含む実践がある。

【００２０】

その他の実践において、コンピューティングシステムは、ネットワークを介して通信する複数のノードを含み、前記方法は、コンピューティングシステムに、第１の変数及び第１の時系列を第１のノードに記憶するステップと、第２の変数、及び第２の変数に対する操作の第２の時系列を第２のノードに記憶するステップと、第１の変数と第２の変数との両方に対して操作を実行することによってイベントのシーケンスの少なくとも一部のイベントを処理し、イベントを処理することに応じて第１の時系列及び第２の時系列の両方を更新するステップとを実行させるステップをさらに含む。

【００２１】

その他の実践は、上述の実践のいずれかの矛盾のない組合せを含む。

【００２２】

別の態様において、本発明は、イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義するそれらのイベントのシーケンスからのイベントをコンピューティングシステムに処理させるための、コンピュータ可読媒体上に非一時的形態で記憶されたソフトウェアであって、コンピューティングシステムに上述の方法のいずれかを遂行させるための命令を含むソフトウェアを特色とする。

【００２３】

さらに別の態様において、本発明は、イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義するそれらのイベントのシーケンスからのイベントを処理するためのコンピューティングシステムであって、前記イベントを受け取るように構成された入力デバイス又はポート、及びイベントのシーケンスの処理に参加するように構成された少なくとも１つのプロセッサを含み、処理が、上述の方法のいずれかを含む、コンピューティングシステムを特色とする。

【００２４】

さらに別の態様において、本発明は、イベントが入力デバイス又はポートを介して受け取られる順序とは無関係である前記イベントの正しい順序を定義するそれらのイベントのシーケンスからのイベントを処理するためのコンピューティングシステムであって、前記イベントを受け取るための手段と、上述のステップのいずれかを遂行するための手段とを含む、コンピューティングシステムを特色とする。

【００２５】

態様は、以下の利点のうちの１又は２以上を有する可能性がある。

【００２６】

本明細書において説明される技術は、すべての計算に明示的な時間的な次元（dimension）を追加するが、それにもかかわらず、この時間的な次元を計算が実行される順序から切り離すことに基づく。結果は、イベント及びその結果として起こる状態変化に応じて実行される操作を示すエントリ（「フレーム」と呼ばれる）の時系列を漸進的に構築するシステムである。受け取られたイベントに応じて時系列が更新されるとき、時系列は、結局のところ、決定性の最終状態に収束する。ついに、すべてのイベントが受け取られ、それらのイベントの結果として生じるすべての計算が計算されると、時系列は、一意の時系列に収束しているであろう。例えば、システムは、イベントが遅延される可能性がある遅延の期間に対する知られている制限に基づいて、イベントのシーケンスが終了したと決定する可能性がある。さらなるイベントが到着しない時点が存在する場合、システムは、決定性の最終状態に収束する。しかし、システムは、必ずしも決定性でない経路をたどることによってそのようにし得る。

【００２７】
一部の実施形態においては、（計算時間及び／又はリソースの観点で）高価な並行性制御手順の必要性が、避けられ得る。例えば、２相コミットのような手順は、必要とされない。加えて、リモートノード間の通信を必要とする非ローカル検証手順は必要ない。その代わりに、検証は、ローカルの処理のみを使用して実現され得る。さらに、すべての記憶されたイベントを再処理することを必ずしも必要とせずに、ノードの故障後に、正しいローカルの状態が復元され得る復元手順が、説明される。遅延されたイベントの後のローカルの状態の更新も、順序が遅延されたイベントの後であるすべての記憶されたイベントを再処理することを必ずしも必要とせずに実行され得る。
【図面の簡単な説明】
【００２８】
【図１】イベント処理装置を示す図である。
【図２】イベントを構成する追加的なフィールドを示す図である。
【図３】時系列の縮小された表現及び拡大された表現を示す図である。
【図４Ａ－４Ｂ】時系列に対する重大である及び重大でない操作の例を示す図である。
【図５】イベントが正しい順序で到着するときの方法の実行のステップを示す図である。
【図６】イベントがバラバラの順序で到着するときの方法の実行のステップを示す図である。
【図７】図５の例と同様のより複雑な例を示す図である。
【図８】図６の例と同様のより複雑な例を示す図である。
【図９】実装形態がなぜラッチを必要としないのかを示す図である。
【図１０】時系列の再構築を示す図である。
【図１１】生成カウンタ（generation counter）の使用を示す図である。
【図１２】図１１に示された活動に関する時系列のあり得る状態を示す図である。
【図１３】図１１に示された活動に関するイベントフレームのあり得る状態を示す図である。
【図１４】図１２及び１３の状態に基づく後戻りの例を示す図である。
【発明を実施するための形態】
【００２９】
図１を参照すると、イベントインジェスタ（event ingestor）１２が、ネットワーク１４への接続を有する。さまざまなイベントソース１６、１８、２０が、ネットワーク１４にやはり接続される。ただ１つのイベントインジェスタ１２が示されているが、２以上のそのようなマシンが設けられる可能性がある。概して、それぞれのイベントインジェスタ１２は、それ自体のローカルに記憶されたデータだけでなく、必要に応じてネットワーク１４を介してメッセージをやりとりすることによって別のイベントインジェスタによって記憶されたデータにもアクセスし得る。一部の実施形態においては、マルチノードコンピューティングシステムが、１若しくは２以上のイベントインジェスタ及び／又は１若しくは２以上のイベントソースを含むようにそれぞれが構成されるさまざまなノードを含む。一部の実施形態において、それぞれのイベントインジェスタは、互いのノードから物理的に又は地理的に分けられているが、ネットワーク１４を通じた何らかの経路を介して通信する、ネットワーク１４内のそのイベントインジェスタ自体のノード上に実装される。
【００３０】
サロゲートタイムスタンプ（surrogate timestamp）
それぞれのイベントソース１６、１８、２０は、イベントを生じる。ネットワーク１４は、例えば、ネットワーク１４を介してイベントデータを送信すること（又はブロードキャストすること）によってイベントソース１６、１８、２０からイベントインジェスタ１２にこれらのイベントを転送する。イベントデータがイベントインジェスタ１２において受け取られるとき、そのイベントデータは、イベントハンドラによって扱われるべきイベントを表すものとして認識され得る。例えば、イベントハンドラは、イベントに応答して計算を実行する可能性があり、この計算は、下でより詳細に説明されるように、作業状態

（working state）を表す１又は２以上の変数に対する操作を実行することを伴う可能性がある。

【００３１】

イベントは、順序付けられる。一部の例において、順序付けは時間的なものであり、それぞれのイベントは特定の時点で起こる。その発生時間は、さまざまな方法で表され得る。例えば、イベントの発生時間は、イベントデータとともに含まれるタイムスタンプの値に対応する可能性がある。いかなる２つのイベントもまったく同じ時間に起こることがない場合、結果として得られるイベントのシーケンスは完全に順序付けられる。

【００３２】

実際問題として、たとえ２つのイベントの実際の発生時間にいくらかの小さな差があったとしても、それらの２つのイベントが同時に発生したものとして印を付けられることがあり得る。これのよくあり得る起こり方は、時間が測定される粒度（granularity）よりも互いに接近して２つのイベントが起こる場合である。この場合、シーケンスは、不完全に順序付けられる。

【００３３】

イベントインジェスタ１２が本明細書において説明される方法を実装するためには、イベントのシーケンスが、単に不完全に順序付けられるのではなく、完全に順序付けられなければならない。不完全に順序付けられたシーケンスを完全に順序付けられたシーケンスに変換する１つの方法は、タイブレーカー（tiebreaker）として機能するためのサロゲートタイムスタンプを提供することである。

【００３４】

サロゲートタイムスタンプを割り当てる１つの方法は、１つ１つがイベントソース１６、１８、２０に対応するカウンタの組を保有することである。そのとき、それぞれのカウンタは、一意のシード（seed）に初期化される。一部の例において、シードは、イベントソースの数未満である正の整数である。例えば、３つのイベントソースに関して、シードは「１」又は「２」である可能性がある。

【００３５】

イベントソース１６は、新しいイベントを生じさせるとき、イベントソースの数だけそのイベントソース１６のカウンタを増加させる。それから、イベントソース１６は、カウンタの増加させられた値がイベントの元のタイムスタンプに付加されるようにして元のタイムスタンプを含むサロゲートタイムスタンプによってイベントをタグ付けする。イベントソース１６が新しいイベントをイベントインジェスタ１２に送るのはそのときのみである。このプロセスは、イベントインジェスタ１２が完全に順序付けられたイベントの組のみを受け取ることを保証する。解説を簡単にするために、本開示の残り全体を通じて、元のタイムスタンプの値を明示的に示すことなしに、サロゲートタイムスタンプは、単純に、そのようなカウンタの整数値であると仮定される（これは、イベントの完全に順序付けられたシーケンスを引き続き保証する）。

【００３６】

その他の例においては、時間的な順序付けの代わりに、イベントは、イベントの完全に順序付けられたシーケンス、又はタイブレーカーを使用して完全に順序付けられたシーケンスに変換され得るイベントの不完全に順序付けられたシーケンスを決定する何らかのその他の種類の計量基準に関連付けられる可能性がある。

【００３７】

イベントの保存

イベントは、イベントインジェスタ１２において受け取られ、サロゲートタイムスタンプを与えられた後、維持され、持続可能にされる。これは、さまざまなやり方で遂行され得る。例えば、イベントが、ハードディスクドライブ（又はその他の不揮発性ストレージ媒体）などの持続性のあるストレージ媒体に書き込まれる可能性がある。又は、イベントは、別の完全に独立したイベント処理装置に送られる可能性がある。

【００３８】

イベントの持続性は、イベント処理に関連して、かつ、データが失われた場合の復元を容易にするためにイベントを再処理することを認める。イベントインジェスタ１２は、イベントが持続可能にされるまでイベント処理を始めない。

【００３９】

遅延されたイベント及び処理順序

イベントソース１６、１８、２０からイベントインジェスタ１２までの移動の行程で、イベントは、遅延されることになる可能性がある。これは、ネットワークの混雑、伝播遅延、又は破損若しくはパケットの喪失が原因であるイベントデータの再送信などのさまざまな理由による可能性がある。イベントの遅延の結果として、イベントは、（例えば、それらのイベントのサロゲートタイムスタンプの順序に従って）イベントが実際に発生した順序と一致している順序でイベントインジェスタ１２に到着するとは限らない。

【００４０】

データの構造

イベントインジェスタ１２は、イベントリスト２４、第１の及び第２の変数２６Ａ、２６Ｂ（全体として２６）の組、並びに対応する第１の及び第２の時系列２８Ａ、２８Ｂ（全体として２８）の組が記憶されるストレージ２２を特色とする。２つの変数及び２つの時系列が示されるが、これは例示のためであるに過ぎない。本明細書において説明される方法は、１又は２以上の変数及び１又は２以上の対応する時系列に適用可能である。

【００４１】

変数２６の値は、イベントインジェスタ１２の作業状態を定義する。イベントハンドラ３０は、イベントが到着するときにイベントインジェスタ１２が作業状態を変更することを可能にする。例えば、イベントハンドラ３０は、規則処理エンジンを使用して、到着するイベントデータに基づいて規則の組を評価し、特定の規則の条件が満たされることに応じて特定の規則をトリガし得る。そして、トリガされた規則は、対応するアクションを実行し得る。

【００４２】

したがって、それぞれの到着するイベントに関して、イベントハンドラ３０は、変数２６に対する操作を伴う可能性があるアクションを実行する。これらのアクションは、（イベントのサロゲートタイムスタンプに従って）イベントに対応するどんな時点とも結びつけられる。しかし、アクションがイベントの到着に応答して遂行され、イベントが正しい順序で到着するとは限らないので、アクションも正しい順序で実行されない。実際、アクションが関連付けられる基礎を成すイベントの時間的な順序と一致しない順序でそれらのアクションが実行されることは極めてよくあることである。結果として、それぞれの変数の状態がイベントが順序通りに到着していたならばそうであったであろう状態と一致しない時間が存在する。

【００４３】

それぞれの時系列２８は、対応する変数２６に関連付けられる。時系列２８は、その変数に影響を与えるイベントの組の記録と、そのイベントの発生の結果として行われるアクションとを含む。これらのアクションは、規則の呼び出しと、時系列２８に対応する変数２６に対して実行された特定の操作の形態の状態変化とを含む。

【００４４】

イベントが到着するとき、イベントハンドラ３０は、下でより詳細に説明されるように、例えば、新しいフレームを追加することによって受け取られたそれぞれの新しいイベントに関して時系列２８を更新することによって時系列２８を徐々に構築する。しかし、時系列２８は、変更不可能ではない。遅延されたイベントが到着し、バラバラの順序で処理されるとき、時系列２８は改訂される可能性がある。

【００４５】

イベントの構造

図２に示されるように、それぞれのイベントは、サロゲートタイムスタンプ６４及び本体６６を有する。サロゲートタイムスタンプ６４は、イベントのシーケンスが完全に順序

付けられることを保証する。本体６６は、イベントに関連してどのタスクが遂行されるべきかを規定する。加えて、それぞれのイベントは、状態フィールド６８及びアクセスされた変数フィールド７０をさらに有する。

【００４６】

イベントの状態フィールド６８は、イベントが保留中であるのか、有効であるのか、又は無効であるのかを示す。あらゆるイベントは、保留中として始まる。そのイベントは、イベントに関連するすべてのタスクが完了されるまで保留中のまま留まる。それから、イベントは有効になる。有効なイベントは、時間的にそのイベントに先立つ遅延されたイベントが到着する場合、無効になる可能性がある。イベントが無効になるとき、そのイベントを再処理するためにイベントハンドラ３０が生成されなければならない。

【００４７】

イベントのアクセスされた変数フィールド７０は、そのイベントが処理されたときにアクセスされたすべての変数を列挙する。イベントのアクセスされた変数フィールド７０は、初めは空白であり、イベントが処理又は再処理された後、更新される。

【００４８】

イベントを受け取ると、イベントインジェスタ１２は、そのイベントのためのイベントハンドラ３０を生成する。イベントハンドラは、イベントリスト２４のエントリを生成し、そのエントリに「保留中」と印を付ける。そして、イベントハンドラ３０は、イベントハンドラ３０によって決定された操作（例えば、変数２６の値を読み取るか、書き込むか、又は読み取りと書き込みとの両方を行う操作）を使用して１又は２以上の変数にアクセスする。イベントハンドラ３０は、そのようにするとき、アクセスされた変数のリストを累積し、そのイベントハンドラ３０が終了するとき、アクセスされた変数フィールド７０にこのリストを記憶する。

【００４９】

完了すると、イベントハンドラ３０は、イベントに関する状態フィールド６８を調べる。状態フィールド６８がまだ「保留中」に設定されている場合、イベントハンドラ３０は、その状態フィールド６８を「有効」に設定する。しかし、イベントハンドラ３０がそのイベントハンドラ３０のイベント処理でふさがっている間に、遅延されたイベントが到着した可能性がある。これは、イベントインジェスタ１２がその遅延されたイベントのためのイベントハンドラを生成し、そして、そのイベントハンドラが状態フィールドを「保留中」から「無効」に設定した可能性があることを意味する。これが起こる場合、イベントハンドラ３０は、状態フィールドを「保留中」に設定し直し、イベントを再処理する。

【００５０】

イベントハンドラ３０は、イベントを再処理するとき、まず、すべてのアクセスされた変数のリストを受け取る。イベントハンドラ３０は、再処理中に変数にアクセスするとき、アクセスされた変数のリストを使用して、再び取りあげられている変数を特定する。イベントハンドラ３０は、アクセスされた変数を再び取りあげるとき、有効であるものとしてその変数に印を付ける。イベントハンドラは、すべての変数を知っており、アクセスされたすべての変数を知っているので、どの変数がアクセスされなかったかを容易に決定することができる。そして、イベントハンドラは、下でより詳細に説明されるように、それらの変数に関連するフレームを削除することに取りかかる。

【００５１】

時系列の構造

時系列は、時間をキーとするフレーム５４（図３）のシーケンスとして表され、フレーム５４のそれぞれが、イベントに対応する。図３は、典型的な時系列を縮小されたフォーマット５０と拡大されたフォーマット５２との両方で示す。拡大されたフォーマット５２は、フレーム５４のそれぞれのフィールドを明示的に示す。縮小されたフォーマット５０は、特定のフィールドを省略するが、取り消し線及びアステリスクの賢明な使用によってそれらのフィールドに情報を盛り込む。表現を簡潔にするために、本開示は、時系列が変数に関する操作及び結果として得られた値のシーケンスを列挙する変数の名前を値フィー

ルドのヘッダにおいてやはり特定する縮小されたフォーマット５０を利用することが多い。

【００５２】

示された時系列５２は、５つのフレームを示し、それらのフレームのそれぞれは、４つのフィールドを有する。イベントを受け取ることに応答して、イベントハンドラ３０は、フレーム５４を追加する。この新しいフレーム５４は、時間フィールド５６、種類フィールド５８、値フィールド６０、及び有効性フィールド６２を有する。

【００５３】

時間フィールド５６は、サロゲートタイムスタンプを記憶する。縮小されたフォーマットにおいて、空白は、時間の始まりを示す。時系列の「下流」方向は、タイムスタンプの値が増える方向である。「上流」方向は、下流方向の反対である。

【００５４】

種類フィールド５８は、そのフレームに関連するイベントに応答して変数に対して実行される操作の種類に対応するフレームの種類を示す。基本的に、２種類のフレーム、すなわち、読み取りフレーム及び書き込みフレームが存在する。書き込みフレームは、さらに２つの種類、すなわち、ブラインド書き込み（blind-write）フレーム及び読み取－書き込み（read-write）フレームに細分化される。

【００５５】

「読み取り」フレームは、受け取られたイベントの結果として、時系列に対応する変数の値が読み取られるが、書き込みが行われないときに追加される。示された時系列は、t = 20の１つ及びt = 40のもう１つの２つの読み取りフレームを示す。縮小されたフォーマットにおいて、アステリスクは、読み取りフレームを示す。

【００５６】

「読み取－書き込み」フレームは、受け取られたイベントの結果として、時系列に対応する変数の値が読み取られ、しかも、その変数の新しい値が時系列に書き込まれたときに追加される。概して、書き込まれた値は、読み取られた値に依存する。示された時系列は、t = 30の１つ及びt = 50のもう１つの２つの読み取り－書き込みフレームを有する。これらは、縮小されたフォーマットにおいては、値の上に横棒のない書き込まれた値によって示される。表現を簡潔にするために、拡大されたフォーマットは、読み取り－書き込みフレームを「更新」と呼ぶ。

【００５７】

「ブラインド書き込み」フレームは、受け取られたイベントの結果として、時系列内の値が上書きされたが、読み取られなかったときに追加される。示された時系列は、２つのブラインド書き込みのみを有する。１つは、t = 0において初期化が済んだときのものであり、もう１つは、t = 10にある。縮小されたフォーマットにおいて、ブラインド書き込みは、ブラインド書き込みされた値の上の実線によって示される。表現を簡潔にするために、拡大されたフォーマットにおいて、ブラインド書き込みは、「上書き」と呼ばれる。

【００５８】

読み取りと、読み取り－書き込みと、ブラインド書き込みとの間の区別は、フレームを生成したイベントが遅延されたイベントを受け取ることに応答して再処理される必要があるかどうかを決定するためにそれらの区別がやがて使用されるので重要である。

【００５９】

値フィールド６０は、フレーム５４が追加された時点での変数の最も新しい値を含む。この値フィールド６０は、フレーム５４がブラインド書き込みフレーム又は読み取り－書き込みフレームであることを種類フィールド５８が示すときにのみ使用される。

【００６０】

最後に、有効性フィールド６２は、フレーム５４が有効であるか否かを示す。フレームは、有効であるものとして始まる。しかし、新しいイベントが到着するとき、有効性フィールド６２は無効にされる可能性がある。縮小されたフォーマットにおいて、有効及び無効の値は、取り消し線がないこと及びあることによって示される。

【００６１】
イベントに応答して、イベントハンドラ３０は、（変数に対して実行された「操作」とは異なる）時系列のフレームを含む時系列に対する操作を実行することによって時系列を更新する（これは変数に対して実行された「更新」とは異なる）。概して、時系列に対する４種類の操作、すなわち、「付加」、「挿入」、「修正」、及び「削除」が存在する。
【００６２】
「挿入」操作は、時系列の最後のフレームの前にフレームを追加する。「付加」操作は挿入と似ているが、時系列の最後のフレームの後にフレームを挿入する点が異なる。「修正」は、フレーム中の何かを変更する。「削除」は、時系列からフレームを完全に削除する。
【００６３】
重大である操作及び重大でない操作
時系列２８を更新するための特定の操作は、「重大である」又は「重大でない」のどちらかである。「重大である」操作は、時系列の後の時点で目に見える値に対する改訂又は変更をもたらす。より詳細には、「重大である」操作は、遅延されたイベントよりも時間的に後にあったアクションの結果に影響を与え、その結果として、それらのアクションの結果として生じる必然的な状態の変化は、撤回される必要がある可能性がある。
【００６４】
図４Ａ－４Ｂは、時系列におけるフレームの挿入が重大である挿入であるのか又は重大でない挿入であるのかをどのようにして決定するかのいくつかの例を示す。概して、挿入操作は、その挿入操作がその挿入操作から下流にある値に疑問を呈する場合、重大である。
【００６５】
図４Ａに示される操作１１０において、イベントは、t = 35の値＄３００を有する付加フレームをもたらす付加操作を引き起こした。この操作は、影響を与えられる可能性がある、t = 35の後の下流の値が存在しないので重大ではない。概して、付加操作は、定義により最後のフレームの後であるので重大ではあり得ない（つまり、常に重大でない）。
【００６６】
操作１１２において、遅延されたイベントが、t = 8の読み取りフレームの挿入を引き起こした。この操作は、読み取りが時系列内の値に影響を与え得ないので重大ではない。概して、読み取り操作は、読み取られている値を変更しないので常に重大でない。
【００６７】
ブラインド書き込み又は読み取り－書き込みは、後続のフレームを無効にすることがあり得るので重大であることがあり得る。
【００６８】
動作１１４において、t = 17の遅延されたイベントが、読み取り－書き込みフレームの挿入をもたらす読み取り－書き込み操作を引き起こした。t = 20の下流の読み取りフレームが生じさせられた時点で、遅延されたイベントはまだ到着していなかったであろう。したがって、イベントプロセッサは、値＄２．２５を読み取ったであろう。しかし、t = 17の遅延されたイベントの結果として、これは今や正しくない。結果として、t = 20の下流の読み取りフレームは、無効化されなければならない。この無効性は、後で、t = 20のイベントを再処理することによって訂正される。
【００６９】
同様に、t = 30の下流の読み取り－書き込みフレームは、読み取り操作を伴っていたので無効化される。この読み取り－書き込みフレームが追加されたときには遅延されたイベントはまだ到着していなかったであろうから、読み取り－書き込みフレームは、現在正しくないことが知られている値を読み取ったことに基づいている。結果として、この読み取り－書き込みフレームも、無効であるものとして印を付けられなければならない。
【００７０】
概して、読み取り－書き込みフレームは、読み取り操作の結果としてであるのか又は読

み取り－書き込み操作の結果としてであるのかにかかわらず、読み取りを伴うすべての下流のフレームを無効化する。しかし、読み取りを必要としないブラインド書き込み操作は、無効化されない。

【００７１】

すべての読み取り－書き込み操作が無効性の伝播をトリガするわけではない。操作１１６において、遅延されたイベントが、ｔ＝10のブラインド書き込みフレームの直前のｔ＝9の読み取り－書き込みフレームの挿入を引き起こした。しかし、これは、ブラインド書き込みフレームを無効化しない。これは、ブラインド書き込みフレームが何かを読み取ることを含まないからである。したがって、ブラインド書き込みにおいて書き込まれた値は、時系列の上流の値によって影響を受けることがあり得ない。読み取りを伴う、ブラインド書き込みから下流のすべてのフレームは、そのような読み取りがブラインド書き込みフレームの値を読み取るのでやはり無効でない。したがって、ブラインド書き込みフレームは、下流への無効性の伝播を遮断するある種の防壁フレームとして働く。

【００７２】

読み取り－書き込みオペレータの挿入は、したがって、その読み取り－書き込みオペレータの挿入の地点から下流に無効性を伝播させる。この無効性は、時系列の終わりか、又はブラインド書き込みフレームなどの防壁フレームかのどちらかが到達されるまで伝播する。したがって、防壁フレームは、無効性がそこを超えて伝播するのを止める防壁として働く。これは、遅延されたイベントに応答して必要とされる再処理の範囲を制限する。読み取り－書き込みフレームの挿入後に来るすべてのイベントを再処理しなければならない代わりに、ブラインド書き込みの前に来るイベントのみが再処理される必要がある。

【００７３】

読み取り－書き込みフレームは、読み取りを含む下流のフレームを常に無効化するとは限らない。例えば、構成１１８においては、遅延されたイベントが、ｔ＝10のブラインド書き込みの直後にｔ＝13の読み取り－書き込みフレームの挿入を引き起こした。しかし、この読み取り－書き込みフレームは、ｔ＝13の読み取り－書き込みフレームの値がｔ＝10のブラインド書き込みフレームの値と同じであるので下流への無効性の伝播をトリガしない。したがって、ｔ＝13の読み取り－書き込みフレームは、実のところ、まったく違いをもたらさない。いかなる下流の読み取り操作又は読み取り－書き込み操作によって読み取られた値も、ｔ＝13の読み取り－書き込みフレームの挿入の結果として変わらなかったであろう。

【００７４】

図４Ａ－４Ｂは、時系列におけるフレームの修正が重大であるのか又は重大でないのかをどのようにして決定するかのいくつかの例も示す。挿入操作と同様に、修正操作は、いつもではなく、ときに、下流のフレームを無効にすることがある。

【００７５】

操作１２０において、遅延されたイベントが、ｔ＝15のフレームを、新しい値＄２．３０を持つように修正させた。これは、下流への無効性の伝播を開始する。無効性の伝播は、防壁フレーム又は時系列の終わりのどちらかに到達するまで続く。無効性の伝播がそのようになる理由は、読み取り－書き込みフレームに関連して検討された理由と同じである。

【００７６】

一方で、操作１２２においては、遅延されたイベントが、ｔ＝15のフレームを修正させた。しかし、ｔ＝15のフレームの修正された値は、そのフレームの元の値と同じである。これは、明らかに、下流のいかなるものにも影響を与え得ない。したがって、修正は重大でない。

【００７７】

図４Ａ－４Ｂは、時系列からのフレームの削除が重大であるのか又は重大でないのかをどのようにして決定するかのいくつかの例も示す。挿入操作及び修正操作と同様に、削除操作は、いつもではなく、ときに、下流のフレームを無効にすることがある。

【００７８】
まず第一に、操作１２４に示されるように読み取りフレームを削除することは、けっして重大ではない。詰まるところ、読み取りフレームを挿入することが違いをもたらさない場合、同じ読み取りフレームを削除することが何らかの違いをもたらすことは奇妙に見える。
【００７９】
一方で、ブラインド書き込みフレーム又は読み取り－書き込みフレームを削除することは、操作１２６に示されるように値を変更した場合、重大であるが、操作１２８に示されるように値に対する変更をしなかった場合、重大ではない。
【００８０】
操作１３０は、無効性が下流に伝播することを防ぐ防壁フレームとして働くブラインド書き込みフレームを示す。示された例においては、遅延されたイベントが、t = 10のフレームに対する修正を引き起こした。通常、この修正は、操作１２０に関連して上で検討された理由により重大である。しかし、すぐ後に続くt = 15のフレームは、ブラインド書き込みフレームである。これは、下流への無効性の伝播を直ちに遮断するように働く。したがって、いずれのフレームも無効にされない。
【００８１】
一部の実施形態においては、すべての潜在的に無効なフレームが印を付けられるわけではない。これは、場合によっては、フレームが無効であるか否かが１又は２以上の無効なフレームの正しい値の関数であるためである。
【００８２】
例えば、操作１３２において、読み取り－書き込みフレームがt = 15に挿入される。予測されるように、t = 20の読み取りは無効であるものとして印を付けられる。t = 30の読み取り－書き込みフレームはt = 20において読み取られるものに依存するので、その読み取り－書き込みフレームも無効であるものとして印を付けられる。t = 35の読み取りフレームもt = 15の重大な挿入から下流にあるので、ある者は、その読み取りフレームも無効であるものとして印を付けられると予測する可能性がある。実際、一実施形態において、これは当てはまる。
【００８３】
しかし、t = 30のイベントが最終的に再処理されるときに、t = 30の値は結局のところ＄２．７５によって置き換えられることもあり得る。これは、その値が変化していないであろうことを意味する。その場合、t = 35の読み取りフレームは無効ではない。
【００８４】
t = 35の読み取りフレームが最終的に有効になる可能性があるので、その読み取りフレームは、有効であるものとして印を付けられたままである。その読み取りフレームが本当に有効であるのか否かは、t = 30のイベントが再処理されるまで知られない。t = 30のイベントを再処理すると、t = 35の読み取りフレームが無効にされたことが分かる場合、その場合に限って、t = 35の読み取りフレームは無効であるものとして印を付けられる。
【００８５】
有効なフレームの有効性がそのフレームから上流の重大な操作にさらされて未解決のままである上述の方法は、不必要なイベントの再処理を避けることによって計算負荷を削減する。したがって、そのようなフレームも、ブラインドフレームが防壁フレームであるのと同じようにして防壁フレームである。２つの間の違いは、有効性が未解決のフレームが条件付きの防壁を形成する一方、ブラインド書き込みフレームによって形成される防壁が無条件の防壁を形成することである。
【００８６】
操作１３４は、t = 30のフレームが読み取り－書き込みフレームではなくブラインド書き込みフレームであることを除いて操作１３２と同様である。この場合、t = 30のフレームは、操作１３２の場合のように条件付きの防壁フレームなのではなく無条件の防壁フレームである。

【００８７】
本明細書において説明される方法においては、計算が進むにつれて、時系列２８内の状態変化を撤回する可能性がゼロに近づく。結果として、時系列２８は、その時系列２８の決定性の最終状態に収束する。
【００８８】
例１：イベントの遅延のない漸増的最大
図５は、イベントリスト３２に関してイベントの遅延がない場合の方法の実行のステップを示す。示されるプロセスは、２つの変数「ＡＡＡ」及び「ＢＢＢ」のそれぞれによって達成される最大値を追跡することに関係する。これらの変数は、例えば、株価を表す可能性がある。
【００８９】
イベントリスト３２Ｆは、６つのイベントを示し、それらのイベントのそれぞれは、２つの変数のうちの１つの値に対応する。
【００９０】
第１のイベントを受け取ると、イベントハンドラ３０は、第１の変数の値を示すようにイベントリスト３２Ａを更新する。また、イベントハンドラ３０は、第１の変数に関する第１の時系列３４Ａを初期化し、適切な値を入力する。
【００９１】
第２のイベントを受け取ると、イベントハンドラ３０は、第２の変数の値を示すようにイベントリスト３２Ｂを更新する。また、イベントハンドラ３０は、第２の変数に関する第２の時系列３６Ｂを初期化し、適切な値を入力する。
【００９２】
第３のイベントを受け取ると、イベントハンドラ３０は、イベントリスト３２Ｃを更新する。イベントは第２の変数に関係するので、第１の時系列３４Ｃは変更されない。イベントハンドラ３０は、第２の時系列３６Ｃをのぞき見る。のぞき見るこの行為は、第２の時系列３６Ｃに記録される。第２の時系列３６Ｃの図示の中のアステリスクは、のぞき見るこの行為の記録を表す。しかし、第３のイベントにおいて受け取られた値は第２の時系列３６Ｃ内に既にある値よりも小さいので、いかなる変更も第２の時系列３６Ｃになされなくてよい。
【００９３】
第４のイベントを受け取ると、イベントハンドラ３０は、イベントリスト３２Ｄを更新する。イベントは第２の変数に関係するので、第１の時系列３４Ｄは変更されない。イベントハンドラ３０は、第２の時系列３６Ｄをのぞき見て、今回は第２の変数の最大値が変わったことを発見する。結果として、イベントハンドラ３０は、第２の時系列３６Ｄを更新する。
【００９４】
第５のイベントを受け取ると、イベントハンドラ３０は、イベントリスト３２Ｅを更新する。イベントは第１の変数に関係するので、第２の時系列３６Ｅは変更されない。イベントハンドラ３０は、第１の時系列３４Ｅをのぞき見て、第１の変数の最大値が変わったことを発見する。したがって、イベントハンドラは、第１の時系列３４Ｅを更新する。
【００９５】
第６のイベントを受け取ると、イベントハンドラ３０は、イベントリスト３２Ｆを更新する。イベントは第１の変数に関係するので、第２の時系列３６Ｆは変更されない。イベントハンドラ３０は、第１の時系列３６Ｆをのぞき見る。のぞき見るこの行為は、第１の時系列３４Ｆに記録される。第１の時系列３４Ｆの図示の中のアステリスクは、のぞき見るこの行為の記録を表す。しかし、第６のイベントにおいて受け取られた値は第１の時系列３４Ｆ内に既にある値よりも小さいので、いかなる変更も第１の時系列３４Ｆになされなくてよい。
【００９６】
明らかなように、第１の及び第２の時系列３４Ｆ、３６Ｆは、イベントの完全なリスト

３６Ｆを前提として人が期待する状態に収束した。

【００９７】

例２：イベントの遅延のある漸増的最大

図６は、イベントリスト３６Ｆに示されるイベントが誤った順序で到着する場合を考慮する。

【００９８】

第１のイベントを受け取ると、イベントハンドラ３０は、第１の変数の値を示すようにイベントリスト３８Ａを更新する。また、イベントハンドラ３０は、第１の変数に関して第１の時系列４０Ａを初期化し、適切な値を入力する。

【００９９】

第２のイベントを受け取ると、イベントハンドラ３０は、第２の変数の値を示すようにイベントリスト３８Ｂを更新する。また、イベントハンドラ３０は、第２の変数に関して第２の時系列４２Ｂを初期化し、適切な値を入力する。

【０１００】

しかし、イベントの遅延の結果として、到着する次のイベントは、しかし、第３のイベントではなく、第４のイベントである。第４のイベントを受け取ると、イベントハンドラ３０は、第２の変数の値を示すようにイベントリスト３８Ｃを更新する。イベントハンドラ３０は、第２の時系列４２Ｃも更新する。イベントは第１の変数を含まないので、イベントハンドラ３０は、第１の時系列４０Ｃはそのままにする。

【０１０１】

それから、第３のイベントがついに到着する。第３のイベントを受け取ると、イベントハンドラ３０は、第３のイベントを正しい位置に挿入することによってイベントリスト３８Ｄを更新する。第３のイベントは第２の変数にのみ関係するので、第１の時系列４０Ｄはそのままにされ得る。

【０１０２】

イベントハンドラ３０は、第２の時系列４２Ｄをのぞき見る。それから、イベントハンドラ３０は、アステリスクによって示されるように、こののぞき見ることを記録することによって第２の時系列４２Ｄを更新する。操作によって引き起こされた更新の性質に基づいて、イベントハンドラ３０は、ここで、この操作を重大であるものとして又は重大でないものとして分類しなければならない。

【０１０３】

そのようにするために、イベントハンドラは、第２の時系列４２Ｄをのぞき見たことから、第３のイベントの第２の変数の値が何かを変更するほど大きくないことを観測する。結果として、操作は、重大でないと考えられ、イベントハンドラ３０は、第２の時系列４２Ｄ内の第２の変数の値をやはりそのままにする。

【０１０４】

再び、イベントの遅延の結果として、到着する次のイベントは、第５のイベントではなく、第６のイベントである。イベントハンドラは、それに応じてイベントリスト３８Ｅを更新し、第１の時系列４０Ｅを更新し、第２の時系列４２Ｅをそのままにする。しかし、実際は、第１の時系列は、遅延された第５のイベントにおいてより大きな値が反映されるので今や正しくない。

【０１０５】

最終的に、第５のイベントが到着する。イベントハンドラ３０は、第５のイベントを正しい位置に挿入することによってイベントリスト３８Ｆを更新する。第５のイベントは第２の時系列４２Ｆに影響を与えないので、イベントハンドラ３０は、第１の時系列４０Ｆにのみ関心を持つ。

【０１０６】

特に、イベントハンドラ３０は、第５のイベントを第１の時系列の正しい位置に挿入することによって第１の時系列を更新する。そして、イベントハンドラ３０は、この第５のイベントに関連する操作によって引き起こされた更新が操作を重大であるものとして分類

するのに十分であるかどうかを確かめる。
【０１０７】
そのようにする際に、イベントハンドラ３０は、第１の時系列４０Ｆ内の第５のイベントの後のイベントの値をのぞき見る。イベントハンドラ３０は、第６のイベントを扱ったイベントハンドラが更新によって正しくない状態にされた値に依拠していたことを発見する。したがって、この操作が引き起こした更新に基づいて、第１の時系列４０Ｆに対するこの操作は、重大であると考えられる。
【０１０８】
操作が重大であると決定することに応じて、第５のイベントのためのイベントハンドラ３０は、第６のイベントのためのイベントハンドラ３０によって遂行されたアクションに無効であるものとして印を付ける。これは、第１の時系列４０Ｆ及びイベントリスト３８Ｆ内の取り消し線を付けられたエントリによって示される。
【０１０９】
重大である操作が行われたので、イベントハンドラは、すべての影響を受けるイベントのためのイベントハンドラを再実行する。この場合、唯一影響を受けるイベントは第６のイベントである。
【０１１０】
そして、第６のイベントのためのイベントハンドラ３０は、第６のイベントをもう一度処理する。そのようにする際、第６のイベントのためのイベントハンドラ３０は、イベントリスト３８Ｇ内の第６のイベントのエントリを上書きし、第１の時系列４０Ｇをのぞき見る。第１の時系列４０Ｇ内のアステリスクは、のぞき見たことが記録されたことを示す。そのとき、イベントハンドラ３０は、第６のイベントの値が現在の最大値未満であるので、第１の時系列４０Ｇに対するいかなる更新も必要ないことを認識する。
【０１１１】
図５及び図６の時系列がどうにか同じ最終値に収束したことは、明らかである。しかし、図５においては、時系列が常に正しい状態であるのに対して、図６の時系列の中間状態は少なくとも１つの正しくない状態を含んでいた。したがって、図５及び６の初期状態から最終状態までたどられる経路は、異なっていた。
【０１１２】
イベントが到着すると、イベント処理システムは、イベントを処理するためのイベントハンドラ３０を生成する。イベントハンドラは、変数と、それらの変数の時系列とにアクセスする。これらの変数は、共有され、その他のイベントインジェスタ１２によってアクセスされ得る。イベントハンドラは、そのようにするとき、どのイベントが再処理される必要があるかを決定することに関連して使用されるフレーム及びその他のデータ構造を構築する。それから、イベントハンドラ３０は終了する。
【０１１３】
例３：支払いの清算：イベントの遅延なし
図７は、銀行の顧客にそれぞれが関連付けられる３つの口座Ｘ、Ｙ、及びＺ、並びに当座借越（overdraft）のための料金を集金する第４の口座ＯＤが存在する別の例を示す。この例において、顧客の残高が定義された上限未満の額だけ借り越される場合、＄１０の当座借越料金が、顧客の口座から引き落とされ、銀行の当座借越口座に振り込まれる。
【０１１４】
この筋書きは、遅延されたイベントによって引き起こされる可能性がある困難の種類に対する準備が特に整った筋書きである。例えば、高額の小切手を預けることによって使い果たされた口座にお金を補充し、それから直ちに代金の支払いを開始することは非常によくある。これが、競争を開始した。代金が競争に勝ったならば、つまり、小切手が清算される前に代金が処理される場合は、当座借越の費用が突発するであろう。
【０１１５】
以前は、これは、代金が郵便によって支払われることが多く、競争者の一方に有効な不利な条件を課していたので滅多に起こらなかった。しかし、近頃の取引の処理においては

、多くの代金が電子的に支払われ、効果的に、この組み込まれた不利な条件をなくす。
【０１１６】
図７においては、遅延されたイベントはない。プロセスは、口座Ｚから口座Ｘにお金を送金することから始まり（ステップ４２Ａ）、それによって、Ｘの残高を＄１０から＄２０に増やす。結果として、ＸがＸの＄１５の負債をＹに返す（ステップ４２Ｂ）とき、Ｘの＄２０の残高は十分で有り、当座借越は発生しない。
【０１１７】
例４：支払いの清算：イベントの遅延
図７は、同じ初期状態４４Ａから始まる。しかし、口座Ｚから口座Ｘへの送金を伝達するイベントが、遅延される（ステップ４４Ｂ）。結果として、口座Ｘから口座Ｙへの＄１５の送金が処理されるとき、口座Ｘは、＄１０しかない。このため、当座借越料金が、口座ＯＤに振り込まれる（ステップ４４Ｃ）。
【０１１８】
最終的に、遅延されたイベントが到着する（ステップ４４Ｄ）。第１のステップは、口座Ｚ及びＸを、前者から＄１０を引き落とし（ステップ４４Ｅ）、後者に＄１０を振り込む（ステップ４４Ｆ）ことによって更新することである。口座Ｘに対する更新は、以前のイベントを無効にする。Ｙの口座の値は無効化されないことに留意されたい。
【０１１９】
無効にされた結果として、以前のイベントは、再処理されなければならない。これは、口座Ｘの値を修正し（ステップ４４Ｇ）、口座ＯＤの当座借越を削除する（ステップ４４Ｈ）結果となる。
【０１２０】
今度も、４つの時系列は、正しい状態に収束した。
【０１２１】
例５：非同期処理
イベント処理する過程で、いくつかの変数を更新する必要がある可能性がある。少なくとも通常の方法においては、イベント処理操作が原子性を持っている(to be atomic)ことが有利である。概して、これは、更新されるすべての変数に対するラッチ（又は「ロック」）を取得することを必要とする。これは、時間的に高価な操作である。加えて、これは、デッドロックの可能性を生じる。
【０１２２】
本明細書において説明される方法は、イベント処理操作の原子性(atomicity)の必要性をなくす。複数のラッチを取得する代わりに、変数は、大域的に非同期に更新される。時折、変数値の結果として得られる構成は、正しくない可能性がある。しかし、方法の性質は、長期的に見て、変数値の構成が正しい状態に収束するようなものである。時系列に対する操作の原子性のある組（例えば、フレームを挿入すること及び削除すること）を使用可能にするために、時系列に対するラッチが取得される可能性があり、時系列に対するそのような短い局所的なラッチは、複数のノードの間で共有される可能性がある変数自体に対するラッチよりも安価である。
【０１２３】
図８に示されるように、第１のイベント７２Ａ及び第２のイベント７２Ｂが、ほとんど同時に到着し、その結果として、第１の及び第２のイベントハンドラが、同時に実行されている。第１のイベント７２Ａは、口座Ｚから口座Ｘに＄１０が送金されることを必要とする。これは、２つの操作、口座Ｚに対する引き落としと、口座Ｘに対する振り込みとを必要とする。
【０１２４】
第１のイベントハンドラは、口座Ｚに関する時系列８８を更新することによって口座Ｚからの引き落としをどうにか実行する（ステップ７４）。しかし、第１のイベントハンドラが口座Ｘに対する対応する振り込みを遂行する機会を持つ前に、第２のイベントハンドラが第２のイベント７２Ｂを処理し始める。これは、口座Ｘからの＄１５の引き落としを

示す口座Ｘに関する時系列８４に対する更新をもたらす。これは、口座Ｘの当座借越をトリガする（ステップ７６）。また、これは、口座Ｘに移されたはずであった＄１０が見かけ上消える結果をもたらす。

【０１２５】

第１のイベントハンドラの邪魔をしたという事実にまったく気づいていない第２のイベントハンドラは、口座ＯＤに関する時系列９０を更新し、それによって、銀行の当座借越口座に振り込む（ステップ７８）ことによって第２のイベント７２Ｂの処理を完了する。

【０１２６】

最終的に、第１のイベントハンドラは、第１のイベント７２Ａの処理を終了する。これは、行方不明の＄１０を口座Ｘに復元し、さらに、第２のイベントハンドラによって遂行された処理を無効化する（ステップ８０）。

【０１２７】

第２のイベント７２Ｂの無効化は、第２のイベント７２Ｂを再処理するためのイベントハンドラの生成をトリガする。これは、残高の訂正と、当座借越の時系列からのフレームの削除とをもたらす（ステップ８２）。結果として、時系列８４、８６、８８、９０は、ラッチを使用することなく正しい状態に収束した。

【０１２８】

例６：時系列の再構築

潜在的に頻発するイベント及びそれらのイベントの関連する操作が原因で、典型的には、時系列は、比較的高速なストレージ（例えば、メモリ又はその他の揮発性ストレージ）に記憶される。しかし、これは、時系列を、ハードウェアが故障した場合のデータの喪失に対して脆弱なままにする。本明細書において説明される方法は、ハードウェアの故障の結果として失われた可能性がある時系列のすべて又は一部を再構築する簡単な方法を提供する。この特徴を与えるために、イベントインジェスタ１２は、持続性のあるストレージ（例えば、ディスク又はその他の不揮発性ストレージ）にチェックポイントを書き込む。チェックポイントは、時系列に対応する変数の値と、その変数の値が有効であった最も新しいイベントとを特定する。このチェックポイント及びイベントリストを備えているので、完全な時系列を再構築することが可能である。イベントリスト自体は、イベントが改訂されず、したがって、それぞれのイベントが一回記憶される必要があるだけであるので、持続性のあるストレージに保有するのにずっと安価であると考えられる。

【０１２９】

図１０は、復元されるべき時系列９２から始まる（ステップ９４）。何らかの故障の結果として、時系列９２の一部が失われた（ステップ９６）。残されているのはチェックポイント値９８だけである。

【０１３０】

復元の第１のステップは、処理される必要があるイベントを特定することである。これは、この場合、第１の及び第２のイベント１０２、１０４の無効化という結果を生じる（ステップ１００）。

【０１３１】

無効化の結果として、第１の及び第２のイベントハンドラが、対応する第１の及び第２のイベント１０２、１０４を再処理するために生成される。これは、第１のイベント１０２の再処理（ステップ１０６）及び第２のイベント１０４の再処理（ステップ１０８）という結果を生じる。

【０１３２】

再構築方法は、すべてのイベントの再処理を必要としないが、失われた時系列９４を再構築するために必要とされるイベントのみの再処理を必要とする。

【０１３３】

例７：部分的な時系列の再構築

場合によっては、時系列は、イベントが適切に処理されたかどうかを決定することを難しくするような形で損傷を受けているであろう。その場合、イベントが再処理されるべき

か否か分からない。保守的な選択肢は、前述の例において説明されたように、すべてのイベントを再処理することである。しかし、これは、計算の無駄が多い。

【０１３４】

代替的な実施形態において、それぞれのフレームは、そのフレームを含む時系列へのそれぞれのデータアクセスに関連する生成カウント（generation count）をさらに含む。この実施形態においては、あらゆる時系列アクセスに、アクセスされたフレームの生成カウントを変更することが付随する。特定のフレームの生成カウントとそのフレームにアクセスすることにつながったイベントのサロゲートタイムスタンプとの組合せが、一意のアクセスタプルを定義する。イベントを処理するとき、イベントプロセッサが、まず初めに、開始生成カウント及び終了生成カウントを記憶する。これらは、イベントが再処理されなければならないかどうかを決定するために後で使用され得る。

【０１３５】

生成カウントを使用するプロセスが、図１１に示される。

【０１３６】

図１１に示されるステップ１３６において、処理されるイベントは、４つの時系列の初期化である。このイベントを処理することは、４つの時系列の各々にアクセスすることを必要とした。それぞれのアクセスは、生成カウントに関連付けられる。図に示されるように、それぞれの時系列は、そのイベントに関連する一意の生成者カウント（generator count）を有する。初期化ステップは、生成者カウントの値１から始まり、生成カウントの値４で終わった。

【０１３７】

ステップ１３８において、処理されるイベントは、Ｘの時系列からＹの時系列への＄１５の送金である。処理は、開始生成カウントs = 1を有するイベントフレームから始まる。これは、このイベントを処理する間に割り振られる生成カウントの第１の値である。終了生成カウントｅは、イベントが完全に処理されるまで空白のままにされる。

【０１３８】

イベントを処理する次のステップは、＄１５の差し引き及び＄１０の当座借越料金を反映するためにＸに読み取り－挿入フレームを付加することである（ステップ１３９）。これは、Ｘの時系列へのデータアクセスを必要とする。このデータアクセスを示すため、イベントプロセッサは、Ｘの時系列に生成カウントg = 1によって印を付ける。

【０１３９】

イベント処理は、読み取り－書き込みフレームがＹの時系列に付加されるステップ１４０に続く。これは、Ｙの時系列へのアクセスを必要とする。このアクセスは、イベントの処理中になされる第２のアクセスである。結果として、その読み取り－書き込みフレームは、生成カウントg = 2を用いてタグ付けされる。

【０１４０】

イベント処理は、もう１つの時系列、ＯＤの時系列へのアクセスを必要とする。ステップ１４４に示される次のステップは、ＯＤの時系列に読み取り－書き込みフレームを付加することである。これは特定のイベントを処理する過程でなされる第３の時系列アクセスであるので、この付加される読み取り－書き込みフレームは、生成カウントg = 3によってタグ付けされる。

【０１４１】

この時点で、イベントの処理は完了しており、残っているのはこのイベントの終了生成カウントｅを「３」に更新することだけである。

【０１４２】

しかし、この完了の状態は、長くは続かない。ステップ１４６において、t = 1の遅延されたイベントが到着する。このイベントは、Ｚの口座からＸの口座に＄１０を送金する命令を含む。このイベントがもっと早く、つまり、t = 2のイベントの前に到着していたならば、当座借越は起こらなかった。今やこのイベントが到着したので、当座借越が元に戻されなければならず、他のすべてが適切な状態にされなければならない。

【０１４３】
ステップ１４６に示されるように、ｔ＝1のイベントは、イベント処理の終わりに示される。ｔ＝1のイベントを処理することは、２つのアクセスを必要とした。これは、ｔ＝1のイベントフレームにおいて開始生成カウントｓ=1及び終了生成カウントｅ＝2をもたらす。ｔ＝1のイベントの遅い到着の結果として、ｔ＝2のイベントは今や無効である。ここで、ｔ＝2のイベントの状態は、ｔ＝2のイベントの再処理が始まったことを示すために「保留中」であるものとして印を付けられる。
【０１４４】
別のアクセスを見越して、ｔ＝2のイベントのイベントフレームの開始生成者カウントは、今や、ｓ＝4に増加させられる。そのイベントフレームの終了生成カウントｅは、イベントの再処理が完了するまで分からないので削除される。
【０１４５】
ステップ１４８において、ｔ＝2のイベントの再処理が完了する。イベントの終了生成カウンタが、書き込まれる。時系列は、すべて、すべての知られているイベントに一致する状態に収束した。
【０１４６】
図１２は、図１１の時系列が通ったすべての時系列の状態を一覧化する。図１２のそれぞれの列は、１つのノードを表す。図１３は、すべてのイベントフレームに関して同じことをする。図１３のそれぞれの列は、１つのイベントに対応する。
【０１４７】
イベントが有効であるために存在しなければならないいくつかの条件が存在することは、図１１及び１２の例から明らかである。第１に、イベントフレームが、有効であるものとして印を付けられるべきである。第２に、イベントフレームのアクセスフィールドが、時系列のアクセスと一致するべきである。第３に、値フレームが、すべて有効であるものとして印を付けられる。最後に、時系列の生成カウントが、イベントフレームにおいて規定された生成カウントの範囲と一致するべきである。これらの条件のいずれか１つでも満たされない場合、イベントは有効ではあり得ない。
【０１４８】
故障の後に、時系列及びイベントリストを何らかの以前の状態に復元することが可能であるように、非集中型のログ記録システムが存在し、時系列に対する又はイベントリストに対するすべての変更がログに保存されることが仮定される。ログ記録が非集中的であり、すべての時系列が同じノードに記録されるわけではないので、ノードの故障後に、図１１の状態の任意の組合せを持つことがあり得る。着想は、もしあるとすればどのイベントが再処理される必要があるかを決定することである。下で説明されるように生成カウンタを使用して、イベントの組合せが有効であるかどうかを決定することが可能である。これは、イベントを不必要に再処理する必要性をなくす。
【０１４９】
図１４は、時系列が正しいかどうかを決定するためにイベントフレーム及び上の４つの条件をどのようにして使用すべきかの例を示す。
【０１５０】
例１５０においては、すべてのイベントフレームが、有効であるものとして印を付けられる。しかし、実際は、さらに詳しく、特に生成カウンタを調べてみると、ｔ＝2のイベントが無効であり、再処理を必要とすることは明らかである。特に、ｔ＝2のイベントの開始生成カウンタ及び終了生成カウンタは、閉区間［４，５］を定義する。したがって、時系列は、区間［４，５］内の生成カウンタを示すべきである。しかし、時系列は、それを示さない。Ｘの時系列は、区間の外にある生成者カウンタ（generator counter）１を有する。
【０１５１】
例１５２においては、すべてのイベントが有効であるものとして印を付けられるが、ｔ＝1のイベントは、無効でなければならない。これは、イベントに関する時系列が区間［

１，２］内の生成カウントを持つべきであることをイベントフレームが規定するためである。実際には、唯一の生成カウントは、ｇ＝1である。Ｘの時系列は、見たところ、まだアクセスされていない。ｔ＝2のイベントは、対応する時系列のすべての生成カウントが区間［４，５］内にあることをイベントリストが必要とするのでやはり無効である。しかし、生成カウントは、実際は、区間［１，５］内にある。したがって、両方のイベントが、再処理を必要とする。

【０１５２】

イベントフレームのアクセスフィールドを調べることによって同じ結果に到達することが可能である。ｔ＝1のイベントフレームのアクセスフィールドによれば、Ｚの時系列とＸの時系列との両方へのアクセスがあったはずである。時系列自体によれば、Ｚの時系列のみがアクセスされた。

【０１５３】

例１５４においては、ｔ＝1のイベントとｔ＝2のイベントとの両方が無効である。これは、ｔ＝1のイベントに関するイベントフレームのアクセスフィールドがＸの時系列とＺの時系列との両方へのアクセスがあったはずであることを示すからである。明らかなように、何もＸの時系列にアクセスしていない。同じ推論が、ｔ＝2のイベントに当てはまる。

【０１５４】

例１５６においては、ｔ＝1のイベントフレームによれば、生成カウントがすべて区間［１，２］内にあるべきであるので、ｔ＝1のイベントは無効である。しかし、対応する時系列において、生成カウントは、縮退範囲（degenerate range）［１，１］内にある。ｔ＝2のイベントは、イベントフレームの予測される生成カウントの範囲［４，５］が時系列内にあるものと一致しないのでやはり無効である。

【０１５５】

上述のイベント処理手法は、例えば、好適なソフトウェア命令を実行するプログラミング可能なコンピューティングシステムを用いて実装される可能性があり、又はフィールドプログラマブルゲートアレイ（ＦＰＧＡ，field-programmable gate array）などの好適なハードウェアで、若しくは何らかの混成の形態で実装される可能性がある。例えば、プログラミングされる手法において、ソフトウェアは、それぞれが少なくとも１つのプロセッサ、（揮発性及び／又は不揮発性メモリ及び／又はストレージ要素を含む）少なくとも１つのデータストレージシステム、（少なくとも１つの入力デバイス又はポートを用いて入力を受け取るため、及び少なくとも１つの出力デバイス又はポートを用いて出力を与えるための）少なくとも１つのユーザインターフェースを含む（分散、クライアント／サーバ、又はグリッドなどのさまざまなアーキテクチャである可能性がある）１又は２以上のプログラミングされた又はプログラミング可能なコンピューティングシステム上で実行される１又は２以上のコンピュータプログラムの手順を含み得る。ソフトウェアは、例えば、データフローグラフの設計、構成、及び実行に関連するサービスを提供するより大きなプログラムの１又は２以上のモジュールを含む可能性がある。プログラムのモジュール（例えば、データフローグラフの要素）は、データリポジトリに記憶されたデータモデルに準拠するデータ構造又はその他の編成されたデータとして実装され得る。

【０１５６】

ソフトウェアは、ある期間（例えば、ダイナミックＲＡＭなどのダイナミックメモリデバイスのリフレッシュ周期の間の時間）媒体の物理特性（例えば、表面ピット及びランド、磁区、又は電荷）を使用して、揮発性若しくは不揮発性ストレージ媒体又は任意のその他の非一時的媒体に具現化されるなど、非一時的形態で記憶され得る。命令をロードするのに備えて、ソフトウェアは、ＣＤ－ＲＯＭ又は（例えば、多目的若しくは専用のコンピューティングシステム若しくはデバイスによって読み取り可能な）その他のコンピュータ可読媒体などの有形の非一時的媒体上に提供されるか、或いはそのソフトウェアが実行されるコンピューティングシステムの有形の非一時的媒体にネットワークの通信媒体を介して配信される（例えば、伝搬信号に符号化される）可能性がある。処理の一部又はすべて

は、専用のコンピュータで、又はコプロセッサ若しくはフィールドプログラマブルゲートアレイ（ＦＰＧＡ）若しくは専用の特定用途向け集積回路（ＡＳＩＣ，application-specific integrated circuit）などの専用のハードウェアを用いて実行される可能性がある。処理は、ソフトウェアによって規定された計算の異なる部分が異なるコンピューティング要素によって実行される分散された方法で実装される可能性がある。それぞれのそのようなコンピュータプログラムは、本明細書において説明された処理を実行するためにストレージデバイスの媒体がコンピュータによって読み取られるときにコンピュータを構成し、動作させるために、多目的又は専用のプログラミング可能なコンピュータによってアクセスされ得るストレージデバイスのコンピュータ可読ストレージ媒体（例えば、ソリッドステートメモリ若しくは媒体、又は磁気式若しくは光学式媒体）に記憶されるか又はダウンロードされることが好ましい。本発明のシステムは、コンピュータプログラムで構成された有形の非一時的媒体として実装されると考えられる可能性もあり、そのように構成された媒体は、本明細書において説明された処理ステップのうちの１又は２以上を実行するために特定の予め定義された方法でコンピュータを動作させる。

【０１５７】

本発明のいくつかの実施形態が、説明された。しかしながら、上述の説明は、添付の請求項の範囲によって定義される本発明の範囲を例示するように意図されており、限定するように意図されていないことを理解されたい。したがって、その他の実施形態も、添付の請求項の範囲内にある。例えば、本発明の範囲を逸脱することなくさまざまな修正がなされ得る。さらに、上述のステップの一部は、順序に依存しない可能性があり、したがって、説明された順序とは異なる順序で実行される可能性がある。

【０１５８】

本発明及びその好ましい実施形態を説明したが、新規であると主張され、特許証（Letters Patent）によって保証されるのは、以下のものである。

【図１】

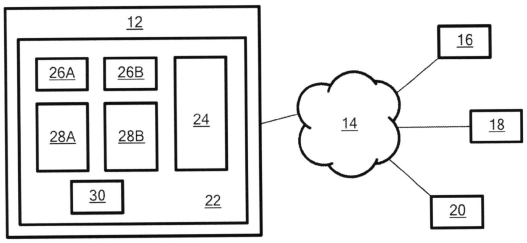

FIG. 1

【図２】

t	イベント	状態	アクセス
	初期化する	有効	
1	z,x,$10.00	有効	z,x
2	x,z,$1.00	保留中	
3	x,y,$15.00	無効	x,y,OD

64 66 68 70

FIG. 2

【図３】

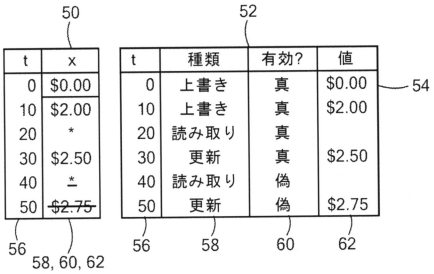

FIG. 3

【図４Ａ】

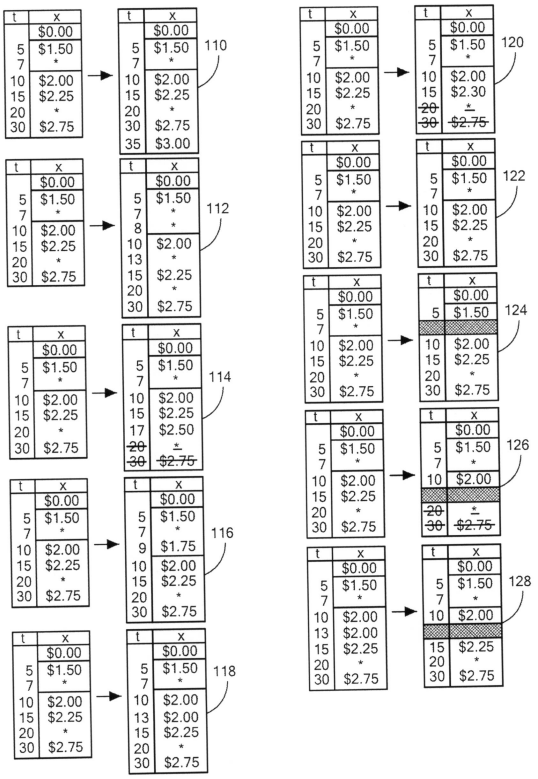

FIG. 4A

【図４Ｂ】

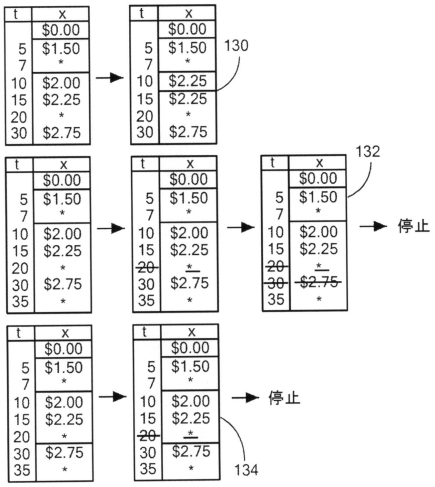

FIG. 4B

【図５】

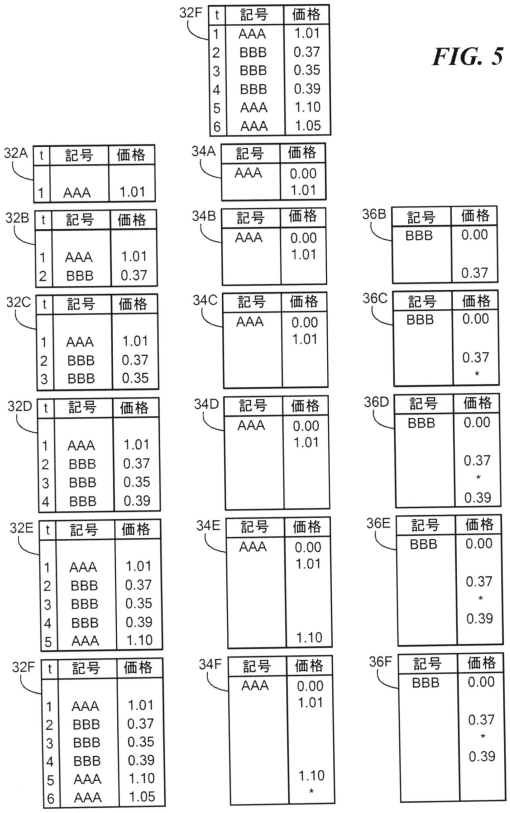

FIG. 5

【図６】

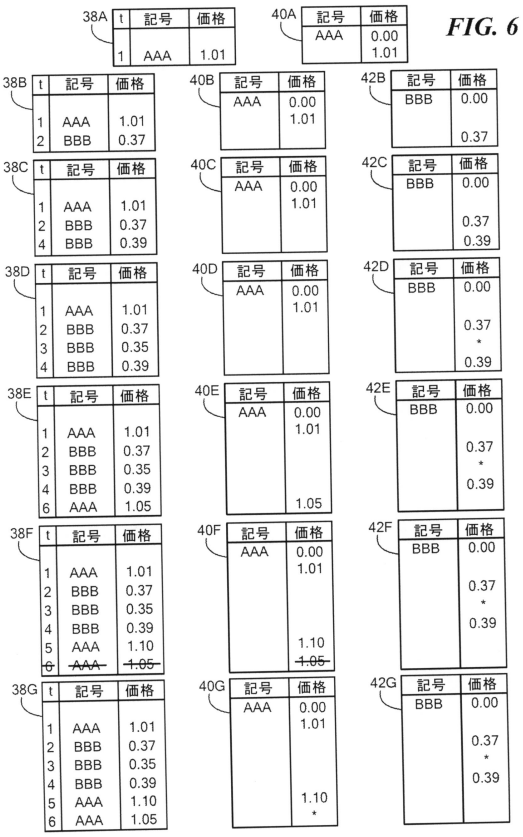

FIG. 6

【図７】

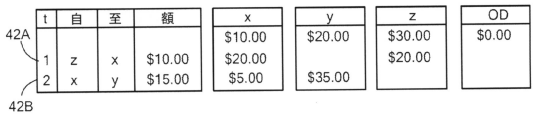

FIG. 7

【図８】

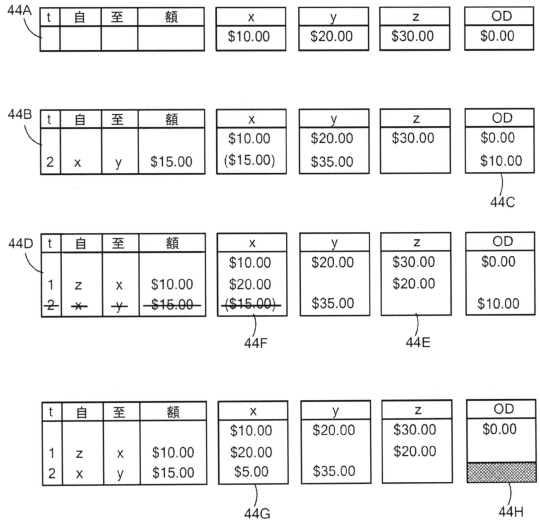

FIG. 8

【図９】

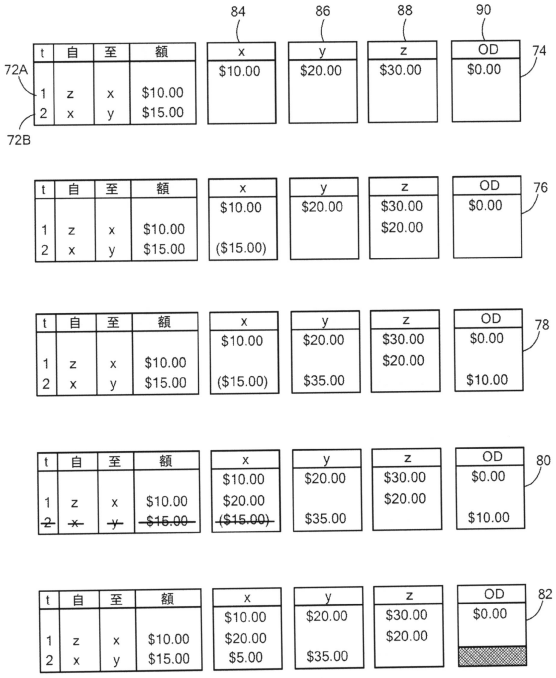

FIG. 9

【図１０】

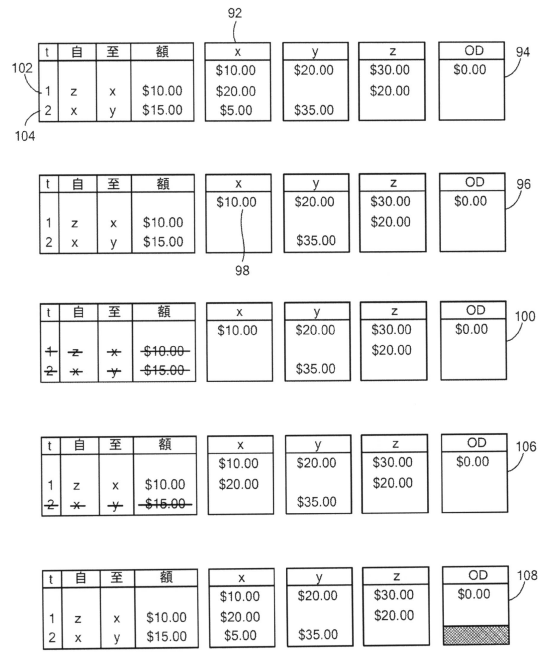

FIG. 10

【図１１】

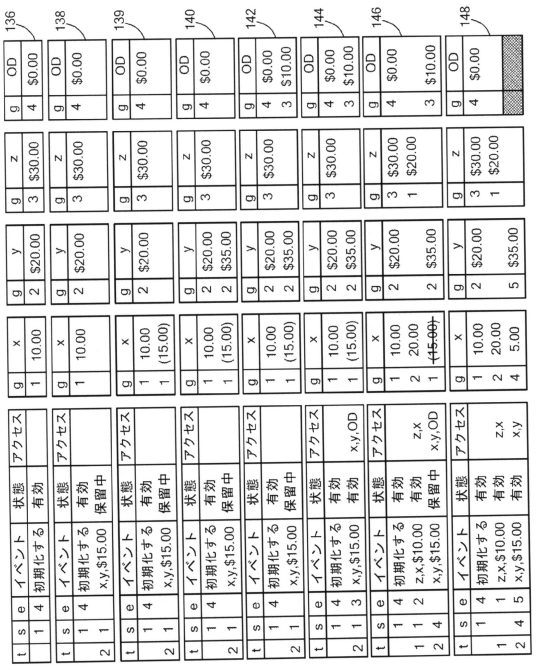

FIG. 11

【図１２】

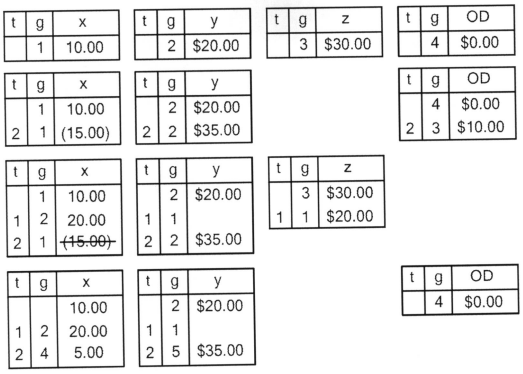

FIG. 12

【図１３】

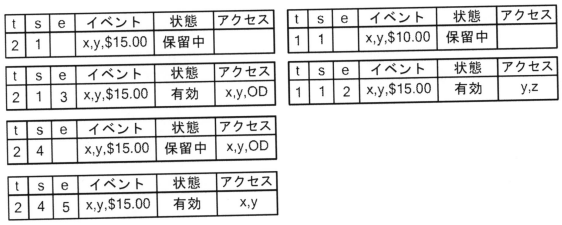

FIG. 13

【図１４】

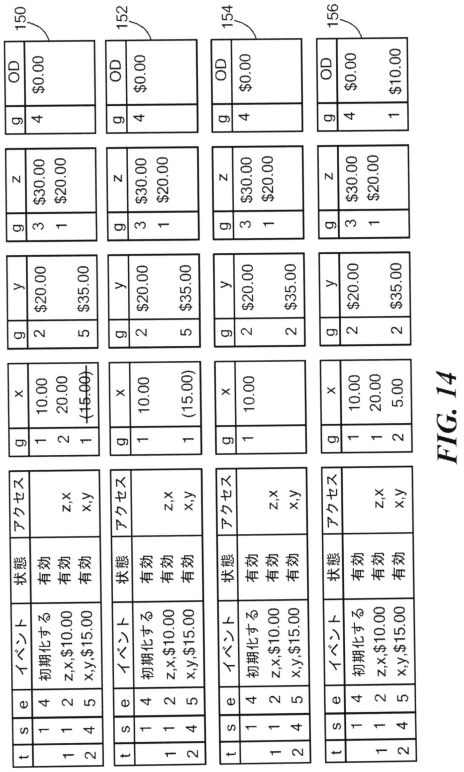

FIG. 14

フロントページの続き

(74)代理人 100150902
弁理士 山内 正子
(74)代理人 100141391
弁理士 園元 修一
(74)代理人 100198074
弁理士 山村 昭裕
(74)代理人 100145920
弁理士 森川 聡
(74)代理人 100096013
弁理士 富田 博行
(72)発明者 スタンフィル クレイグ ダブリュー.
アメリカ国 ０１７７３ マサチューセッツ リンカーン ハックルベリーヒルロード４３

審査官 山本 俊介

(56)参考文献 特開２０１３－０１２０２６（ＪＰ，Ａ）
特開２００３－２７１２０８（ＪＰ，Ａ）
特表２０１０－５２５４８４（ＪＰ，Ａ）
米国特許出願公開第２０１３／０１７９７２９（ＵＳ，Ａ１）
特開２０１１－１９８１５６（ＪＰ，Ａ）
特開２００５－３４６５６４（ＪＰ，Ａ）
特開２００９－２６６００７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／３０
Ｇ０６Ｆ １２／００