

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2014年3月20日(20.03.2014)



(10) 国際公開番号
WO 2014/041673 A1

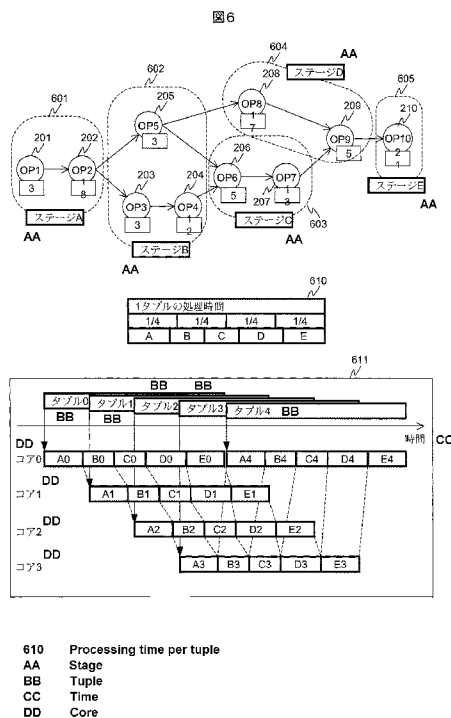
- (51) 国際特許分類:
G06F 12/00 (2006.01) G06F 17/30 (2006.01)
- (21) 国際出願番号: PCT/JP2012/073568
- (22) 国際出願日: 2012年9月14日(14.09.2012)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (71) 出願人(米国を除く全ての指定国について): 株式会社日立製作所(HITACHI, LTD.) [JP/JP]; 〒1008280 東京都千代田区丸の内一丁目6番6号 Tokyo (JP).
- (72) 発明者; および
- (75) 発明者/出願人(米国についてのみ): 今木 常之(IMAKI Tsuneyuki) [JP/JP]; 〒1858601 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内 Tokyo (JP). 西澤 格(NISHIZAWA Itaru) [JP/JP]; 〒1858601 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内 Tokyo (JP).
- (74) 代理人: 井上 学, 外(INOUE Manabu et al.); 〒1008220 東京都千代田区丸の内一丁目6番1号 株式会社日立製作所内 Tokyo (JP).
- (81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.
- (84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[続葉有]

(54) Title: STREAM DATA MULTIPROCESSING METHOD

(54) 発明の名称: ストリームデータ多重処理方法

[図6]



(57) Abstract: This invention simultaneously provides enhanced throughput in proportion to the number of computation cores, and low latency performance achieved by processing a query graph using the shortest path. A query parser for converting a query definition into a query graph and determining an inter-operator execution sequence is provided, a set of consecutive operators in the same execution sequence is called a stage, the total computation cost of the operators comprising each stage is called a stage computation cost, and the query graph is divided into a plurality of stages such that each stage computation cost constitutes a value that is smaller than the value obtained by dividing the total cost of all the operators by the number of computation cores. Each of the computation cores fetches the tuples one by one from an input stream and, when the task of processing the tuples from the entry to the exit of the query graph is executed, confirms whether or not the stage processing has been completed for the tuple immediately preceding the assigned tuple before each of the stages is executed.

(57) 要約: クエリグラフを最短パスで処理することで得られる低レイテンシ性能と、計算コア数分のスループット向上を両立する。クエリ定義をクエリグラフに変換してオペレータ間の実行順序を決定するクエリパーサを備え、同実行順序において連続するオペレータの集合をステージと呼び、各ステージを構成するオペレータの計算コストの合計を、当該ステージの計算コストと呼び、各ステージの計算コストが、全オペレータの合計コストを計算コア数で割った値よりも小さな値となるように、クエリグラフを複数のステージに分割し、各計算コアは、入力ストリームから一つずつタプルを取り出して、クエリグラフの入口から出口まで当該タプルの処理を担当して実行する際に、各ステージの実行に先立って、担当タプルの一つ前のタプルについて当該ステージの処理が

完了しているか否かを確認する。



添付公開書類:

— 国際調査報告 (条約第 21 条(3))

明 細 書

発明の名称：ストリームデータ多重処理方法

技術分野

[0001] 本発明は、ストリームデータ処理のスケールアップ技術に関する。

背景技術

[0002] 株取引の自動化、交通情報処理の高度化、クリックストリームの解析といった、高いレートで継続的に発生する情報をリアルタイムに解析し、重要なイベントの発生に対して瞬時にアクションを起こす要求の高まりを背景に、高レートデータのリアルタイム処理を実現する、ストリームデータ処理が注目されている。ストリームデータ処理は、様々なデータ処理に適用可能な汎用ミドルウェア技術であるため、個別案件ごとにシステムを構築するのでは間に合わないようなビジネス環境の急激な変化にも応えつつ、実世界のデータをリアルタイムにビジネスに反映することを可能とする。

[0003] ストリームデータ処理が対象とするストリームとは、タイムスタンプ付きのデータであるタプルが、連続して到来する時系列データである。ストリームデータ処理のユーザが、このストリームに対する監視ルールをクエリとして定義すると、クエリ定義をクエリグラフに変換する。クエリグラフは、オペレータと呼ばれる処理単位をノードとし、同オペレータ間のタプルキューをエッジとする、有向グラフである。入力ストリームを構成する個々のタプルについて、クエリグラフを通過させることで、データフロー的に処理を進める。データフロー型の処理であるため、クエリグラフを多段分割し、複数の計算資源によってパイプライン的に並列処理することで、スループット向上が可能である。

[0004] 一方で、イベントの発生からアクションを起こすまでの時間、即ちレイテンシについても、ミリ秒～マイクロ秒オーダーの、非常に厳しい要求が課せられる。従って、ストリームデータ処理においては、レイテンシ性能とスループット性能の両立が、重要な技術課題である。

- [0005] 本技術分野の背景技術として、特開２０１０－２０４８８０号公報（特許文献１）がある。この公報には、「ストリームデータ処理システムのストリームデータのクエリ処理に関するスループットを向上させる。サーバ装置のスケジューラが、クエリをグループ化するクエリグループへのデータ停滞が発生すると、クエリグループを構成するクエリごとに、そのクエリの入力流量情報およびレイテンシ情報の内の少なくとも１つの情報を基に、そのクエリの負荷評価値を計算し、クエリグループを構成するクエリを、互いにクエリの負荷評価値の和が略均等になるように、複数のクエリグループへと分割し、分割後の複数のクエリグループを、それぞれ異なるプロセッサに再割り当てする。」と記載されている（特許文献１の要約参照）。
- [0006] また、特開２００８－１４６５０３号公報（特許文献２）がある。この公報には、「制御用のプロセッサユニット（ＰＵ）と、それぞれがローカルメモリをもつ複数の演算用のサブプロセッサユニット（ＳＰＵ）と、メインメモリとを含むマルチプロセッサシステムを提供する。複数のＳＰＵ上で動作するオペレーティングシステムは、各ＳＰＵの計算資源を時分割して複数のタスクに割り当てることにより、複数のタスクが並列に実行されるマルチタスク環境において、タスクの実行結果を別のタスクに与えることにより、負荷の異なる複数のタスクからなる特定処理を実行するためのパイプライン処理系を構築し、当該パイプライン処理系を複数動作させる機能と、メインメモリにコンテキストが退避されて実行可能状態にあるタスクをいずれのタスクも実行していないＳＰＵのローカルメモリにロードして実行させる機能とを含む。」と記載されている（特許文献２の要約参照）。
- [0007] また、特開２０１０－１０８１５２号公報（特許文献３）がある。この公報には、「再帰的処理を含む一般のデータ処理を低レイテンシで実現できるストリームデータ処理方法及びシステムを提供する。ストリームデータ処理システムは、複数のクエリの実行木から単一のオペレータグラフを構築し、ストリーム化演算の実行が、入力から出力に向かって一方通行で進むように、オペレータの実行順序を決定し、システム外部のデータを入力する外部発

火オペレータと時限的にデータを生成する内部発火オペレータの発火時刻を監視し、オペレータ実行制御部は、最も早い発火時刻のオペレータを起点として、決定したオペレータ実行順序に従って、当該時刻のオペレータグラフにおける処理を完結させる処理を繰り返す。」と記載されている（特許文献3の要約参照）。

先行技術文献

特許文献

[0008] 特許文献1：特開2010-204880号公報

特許文献2：特開2008-146503号公報

特許文献3：特開2010-108152号公報

発明の概要

発明が解決しようとする課題

[0009] 前項で述べた文献の個々の技術では、レイテンシとスループットの両立の点で課題が残る。特許文献3の方法は、各タプルをクエリグラフの入口から出口まで、一つの計算スレッドで集中処理するため、個々のタプルについて最短パスで処理することになり、レイテンシを抑えることが可能となっている。しかしながら、シングルスレッド実行が前提となるため、マルチCPU、マルチコアといった複数の計算リソースを、スループットの向上に資することが出来ない。

[0010] 特許文献1の方法は、複数の計算リソースによるパイプライン処理において、各ステージの計算負荷を監視し、負荷が増大したステージは、さらに複数のステージに再分割することで、処理のボトルネックを回避する。但し、パイプライン処理では、完全にボトルネックを回避することが難しい。

[0011] 図2、図3に示す例を用いて、ここで注目している課題を説明する。図2に示したクエリグラフは、OP1～OP10の10個のオペレータ201～210によって構成されている。ここで、各オペレータの計算コストは、オペレータに付与された長方形内の値をとることとする。計算コストの合計、

即ち 1 タプル当りの処理時間は 100 である。また、本例では 4 つの計算コアでのクエリ実行を想定する。ここで、このクエリグラフを、図 3 のように A ～ D の 4 つのステージ 301 ～ 304 に分割する。各ステージの計算コストは、それぞれステージ A (301) が 21、ステージ B (302) が 23、ステージ C (303) が 30、ステージ D (304) が 26 となる。この分割は、各ステージの計算コストが、合計コスト合計の 4 (コア数) 分の 1 の 25 に近くなるように、定めたものである。1 タプルの処理時間に対する、各ステージの処理時間の割合は 310 に示す通りである。

[0012] ここで、タプル処理時間の $1/4$ の間隔でタプルが到来する場合、各タプルは、各計算コアにおいて、タイムチャート 311 に示すようなスケジュールで処理されることになる。なお、ステージ X におけるタプル n の処理を、 X_n と表記する。また、ステージ A、B、C、および D の処理は、それぞれ計算コア 0、1、2、および 3 上で実行されることとする。タイムチャート 311 は、計算コア 2 で実行されるステージ C の処理において、タプル n の処理 C_n が、その一つ前のタプル $n-1$ のタプルの処理 $C(n-1)$ の完了まで待たされる様子を表している。このように、パイプライン処理においては、最も処理時間の長いステージに律速されて、スループット性能、レイテンシ性能ともに向上しない、という課題が存在する。4 つのステージの処理時間を、全て 1 タプルの処理時間の丁度 $1/4$ となるように、均等にクエリグラフを分割できれば、この課題は回避されるが、ほとんどのクエリにおいて、そのような条件は成立しない。

[0013] 特許文献 2 は、一つの特定処理を、それぞれ処理時間の異なる複数のタスクに分割した場合の、タスクスケジューリング方法を示している。特定処理を、一つのタプルについてのクエリ処理、タスクをステージと読み替えることで、ストリームデータのパイプライン処理にも適用可能となる。本方法に従えば、各計算コアは、一つのステージの処理が終わる度に、一旦その処理コンテキスト (タプルの中間処理状態) をキューイングし、処理が空いた他の計算コアが、キューの先頭から取り出したコンテキストの処理を再開する

、といった実行様式になる。この方法では、一つのタプルについての処理が、ステージの終了の都度一旦途切れるため、レイテンシ性能に対するオーバーヘッドとなる。

[0014] パイプライン型の並列処理に対し、データ分割による並列処理も方法も存在する。株価の処理を例に取れば、同一のクエリを実行するクエリグラフを銘柄別で用意し、入力タプルの銘柄属性の値に応じて、クエリグラフを切り替えるような方法である。ここで、クエリグラフを構成する各オペレータは、実行状態を持っている（例えば、価格の移動平均を計算する場合は、スライディングウィンドウ上の価格の和、および価格データ数の値が、実行状態として保持されており、新しいタプルを入力した際には、これら実行状態を更新する）ため、同じ銘柄属性を持つタプルは、常に同じクエリグラフで処理する必要がある。そのため、仮に特定の銘柄のデータが集中すると、並列化の効果が得られなくなってしまう。また、データ分割処理は、属性毎に独立に計算可能なクエリにしか適用できない。例えば、全銘柄の株価を対象とした株式指数の算出には適用不可である。

課題を解決するための手段

[0015] 上記課題を解決するための本発明の代表的処理方法の特徴を挙げるならば、

クエリ定義をクエリグラフに変換してオペレータ間の実行順序を決定するクエリパーサを備え、かつ複数のクエリ実行スレッドを備えストリーム処理システムにより実行するストリームデータ多重処理方法であって、前記実行順序において連続するオペレータの集合をステージと呼び、各ステージを構成するオペレータの計算コストの合計を、当該ステージの計算コストと呼び、各ステージの計算コストが、全オペレータの合計コストを計算コア数で割った値よりも小さな値となるように、クエリグラフを複数のステージに分割し、各計算コアは、入力ストリームから一つずつタプルを取り出して、クエリグラフの入口から出口まで当該タプルの処理を担当して実行する際に、各ステージの実行に先立って、担当タプルの一つ前のタプルについて当該ステ

ージの処理が完了しているか否かを確認することに特徴がある。

発明の効果

[0016] 上記本発明の特徴に従うストリームデータ多重処理方法によれば、クエリグラフを最短パスで処理することで得られる低レイテンシ性能と、計算コア数分の複数スレッドを並列動作させることで得られるスループット向上を両立する。したがって、低レイテンシ化、高スループット化の効果が得られる。

図面の簡単な説明

[0017] [図1]本発明の実施例のストリームデータ処理サーバが動作する計算機環境を示す図である。

[図2]クエリグラフの例である。

[図3]従来のストリームデータ処理の多重化方法における動作を示す図である。

[図4]上記実施例におけるストリームデータ処理サーバの構成を示す図である。

[図5]上記実施例におけるストリームデータ処理サーバのクエリ実行スレッドの動作を説明するフローチャートである。

[図6]上記実施例のストリームデータ処理の多重化方法における動作を示す図である。

[図7]上記実施例のステージ分割決定部で用いるコスト閾値を設定するユーザインタフェース画面を示す図である。

[図8]本発明の実施態様の変形例3における複数タプルー括処理に関する設定をするユーザインタフェース画面を示す図である。

発明を実施するための形態

[0018] 以下、実施例を、図面を用いて説明する。

[0019] 図1を用いて、本発明の実施例のストリームデータ処理サーバが動作する、計算機環境の一例を説明する。ストリームデータ処理サーバ100は、CPU90、91、92、93、メモリ103、ネットワークインタフェース

105、ストレージ106、およびそれらを結合するバス104によって構成される計算機である。メモリ103上に、ストリームデータ処理の論理動作を定義するストリームデータ処理部110を配置する。ストリームデータ処理部110は、CPU90～93によって解釈実行可能な実行イメージである。

[0020] ストリームデータ処理サーバ100は、ネットワークインタフェース105を介して外部ネットワーク150に接続される。外部ネットワーク150に接続されたホスト計算機130上で動作するクエリ登録コマンド実行インタフェース131を介して、ユーザによって定義されたストリームクエリ132を、ストリームデータ処理サーバ100が受取る。すると、ストリームデータ処理部110は、そのクエリ定義に従ってストリームデータ処理を実行可能なクエリグラフを構築する。この後、外部ネットワーク150に接続されたデータ発生器120によって送信されるタプル121を、ストリームデータ処理サーバ100が受取ると、ストリームデータ処理サーバ100は該クエリグラフに従ってこれを処理し、結果タプル141を生成する。この結果タプル141は、ネットワーク150に接続されたデータ受信機140に送信される。ストレージ106は、ストリームデータ処理部110の実行イメージの他、一度受取った前記クエリ132のテキストファイルを保存する。ストリームデータ処理部110は、起動時にストレージ106からこのクエリのファイルをロードし、クエリグラフを構築することも可能である。

[0021] 図4を用いて、ストリームデータ処理部110の論理構成を説明する。ユーザがストリームクエリ132の登録操作を実行すると、クエリパーサ402が、同クエリ132をクエリグラフ構成情報403に変換する。オペレータ実行順序決定部404は、同構成情報403を処理して、クエリグラフを構成するオペレータ間の実行順序を決定し、オペレータ実行順情報405として出力する。このオペレータ間の実行順序を決定する処理の一実現形態として、特許文献3に記載のアルゴリズムを利用可能である。オペレータ処理コスト算出部406は、該クエリグラフにおける各オペレータの処理時間見

積りを計算コストとして算出し、ステージ分割決定部407に入力する。同決定部407は、前記クエリグラフを複数のステージに分割する。

[0022] ここでのステージとは、前記のクエリグラフを構成するオペレータ間の実行順序において、連続する一つ以上のオペレータの集合を指す。該クエリグラフにステージ分割結果を付与し、最終的にクエリ実行制御情報408を生成する。図の例では、ステージA、B、C、D、およびEのそれぞれが、{オペレータ1}、{オペレータ2}、{オペレータ3}、{オペレータ4、オペレータ5} および {オペレータ6} の、それぞれのオペレータで構成されている。ステージ分割方法の動作は後述する。以上、クエリ実行制御情報408の生成までが、クエリ登録時の処理となる。

[0023] 次に、クエリ実行時の動作を説明する。クエリ実行時には、クエリ実行スレッド410～413が、当該計算機環境における計算リソース数に併せて動作を開始する。本例では、4つのCPUコアが利用可能である環境で、クエリ実行スレッドが4つ起動するケースを想定する。

[0024] ここで、本実施例でのストリームデータ多重処理の特徴を述べると、順次到来するタプルの一つ一つに対してデータ処理を実行するスレッドを一つずつバインドして並列に処理を進めるのが特徴である。つまり、複数の計算コアが各々分割されたステージを担当し、順次到来するタプルをその複数コア間で渡していくパイプライン処理とは異なり、一つのタプルについての分割された複数のステージの処理を一つの計算コアが完遂する。このため、タプル入力部450は、入力タプル121を受取ると、タプル通番管理部409に渡す。タプル通番管理部409は、個々のタプルに、1ずつインクリメントする整数の通番を付与して、クエリ実行スレッド410～413に渡す。同スレッドのうち、実行休止中であるスレッドの一つが、前記タプル121を処理する。

[0025] ここでは、タプル121に通番88が付き、スレッド410が同タプルの処理を担当すると仮定する。スレッド410は、この通番88を、ローカルな担当タプル実行状態保持領域420内に、担当タプル通番（図4の符号4

21)として保持する。また、同タプルの一つ前のタプル、即ち通番87のタプルの処理をスレッド413が担当すると仮定する。

[0026] スレッド410は、通番88のタプルについて、ステージAの処理から実行する。その実行に先立って、担当の通番88のタプルの一つ前のタプル、即ちスレッド413が担当する通番87のタプルについて、ステージAの処理が完了しているか否かを確認する。この確認は、ステージA実行済タプル通番フラグ437に格納されている値を、担当通番88と比較することで実現する。本例では、クエリ実行スレッド410～413の各々は、自らが担当するタプルの各ステージの実行が終了するごとに、クエリ実行状態保持領域430に作られた対応するステージの実行済みタプル通番フラグの値を、担当するタプル（その当該ステージの実行が終了したタプル）の通番から、次の通番に書き換える。スレッド410は、ステージA実行済タプル通番フラグ437の値を確認して、自らが担当するタプルの通番88と一致しない場合、即ち一つ前の通番87のタプルのステージAの処理が完了していない場合には待機する。スレッド413は通番87のタプルについてステージAの処理（即ち、オペレータ1の処理）を完了した段階で、実行済タプル通番フラグ437の値を87から88に変更する。スレッド410によるステージAの実行に先立つ実行済タプル通番フラグ437の確認は、例えば一定周期で繰り返し行い、このフラグ値の変更を認識した段階で、通番88についてのステージAの処理を開始する。

[0027] このように、各クエリ実行スレッドが、各ステージの処理を開始する前に、直前のタプルについて、当該ステージの処理が完了しているか否かを、タプル通番フラグ437～441を介して、クエリ実行スレッド間で確認するプロトコルに従って、多重処理を推進する。ここで、各オペレータ1～6の実行状態は、クエリ実行状態保持領域430上の、オペレータ実行状態431～436で管理し、全てのスレッド410～413で共有するため、このようなプロトコルによる一貫性管理が必要になる。

[0028] クエリグラフ中に、時間ウィンドウ演算、遅延演算などの時限発火型のオ

オペレータが存在する場合は、これらオペレータが出力するタプルの処理順序も、併せて考慮する必要がある。例えば、遅延時間を4分とする遅延演算は、タイムスタンプ10:00丁度のタプルを入力したら、その4分後に、そのタプルにタイムスタンプ10:04を付けて出力するオペレータである。即ち、10:00のタプルを入力した時点で、10:04のタプルの出力を予定することになる。その後、10:04まではクエリ実行スレッド410～413の全てが実行中であり、10:04を若干経過した時刻に、クエリ実行スレッド410が実行休止状態に移行したとする。このとき、クエリ実行スレッド410では、仮に10:04の数マイクロ秒後のタイムスタンプが付いたタプルが入力ストリームに存在していたとしても、遅延演算が予定していたタイムスタンプ10:04のタプルの出力を、先に実行する必要がある。そのため、時限発火オペレータにおいて、次に出力する予定のタプルは、タプル通番管理部409に登録する。タプル通番管理部409は、登録されたタプルと、入力ストリームの先頭のタプルのタイムスタンプを比較して、より過去の方から順に通番を付与して、クエリ実行スレッドに渡す。

[0029] 次に、図5のフローチャートを用いて、各クエリ実行スレッドの動作を説明する。処理501で、タプル通番管理部409より入力タプルを取得し、処理502で、入力タプルのデータ、および入力タプルに付与されたタプル通番を、担当タプル実行状態保持領域420に格納する。その後、クエリグラフを構成する全ステージについて、処理504～507を繰り返す。図5のボックス503、508はループ繰り返しを示す。ループの最初の処理504では、実行済タプル通番フラグ437～441を確認し、当該スレッドが担当するタプルの通番と、値が一致するか否かによって条件分岐する。一致しない場合は、処理504をスピンで繰り返し、一致する場合は処理505に進む。

[0030] 処理505では、実行状態保持領域420に保持しているタプル（群）について、当該ステージに属するオペレータの処理を実行する。処理506では、その結果のタプル（群）を実行状態保持領域420に格納する。ここで

、オペレータによっては、一つのタプルを入力して複数のタプルを出力する場合もあるため、実行状態保持領域420で管理されるデータは複数存在し得る。処理507では、担当タプルについて、当該ステージの処理が完了したため、当該ステージのタプル通番フラグを1インクリメントし、次のステージの処理に移る。全てのステージの処理が完了すると、処理509にて、結果タプルをタプル出力部451から出力し、当該担当タプルについての、全クエリグラフの処理を完了する。当該スレッドは再び休止状態に入る。全てのクエリ実行スレッドが、同一のフローに従って動作する。

[0031] 次に、図6を用いて、ステージ分割決定部407における、ステージの分割方法を説明する。本説明では、図2に示したクエリグラフの例をサンプルとしている。同クエリグラフについて、オペレータ実行順序決定部404が決定するオペレータの実行順序は、OP1～10の順番になるとする。また、本例では4つの計算コアでのクエリ実行を想定する。まず、クエリグラフの全計算コストを計算コア数で割った値である25に対して、さらにマージンをとった22を、計算コストの閾値とする。前記オペレータ実行順に従って、オペレータの計算コストを加算して行き、前記閾値を超えないようにステージを分割する。

[0032] 本例では、OP1～2の計算コストの和が21、OP1～3の和が24となるため、閾値以下に収まるOP2までを、最初のステージAとする。以降、同様の分割ポリシーを適用することで、OP3～5、OP6～7、OP8～9、およびOP10に分けられ、最終的にステージA～Eの計5ステージ（601～605）となる。各ステージの計算コストは、それぞれステージA（601）が21、ステージB（602）が18、ステージC（603）が18、ステージD（604）が22、ステージE（605）が21となる。1タプルの処理時間に対する、各ステージの処理時間の割合は610に示す通りである。

[0033] ここで、タプル処理時間の1／4の間隔でタプルが到来するケースにおいて、図4に示した構成と、図5に示したフローに従って処理した場合、各タ

プルは、各計算コアにおいて、タイムチャート611に示すようなスケジュールで処理されることになる。図3のタイムチャート311と異なり、処理の空白期間や待ちは発生しない。詳しく述べると、従来のパイプライン処理によるクエリ処理では、タプルが等間隔で順次到来するという、いわば理想的な入力状況であっても、図3のタイムチャート311で示したように最も処理時間が長いステージに処理速度が律速され、他のステージをそれぞれ担当する計算コアに処理の空白、つまり無駄時間が生じてしまう。これに対し、本実施例のプロトコルによる処理では、タプルが等間隔で順次到来する入力状態では、図6のタイムチャート611に示すとおり全く空白時間が発生しない。

[0034] タプルの到来が不等間隔になった場合に処理待ちが発生するのは、従来のパイプライン処理と本実施例の両者に共通である。但し、図6の実施例のプロトコルにおいて、タプルの到来が不当間隔になった場合に、それにより処理待ちが発生するのは、図6の例では、分割した各ステージの処理コストが図3の例に比べてさほど小さくないからである。先に述べた説明では、全計算コストを計算機コア数4で割った値25に対して計算コストの閾値を22とし、この閾値を超過しない範囲で順次オペレータを統合してステージを決定していた。タプルの到来が不等間隔になったときの空白時間の発生は、この計算コスト閾値の大きさに依存する。全計算コストを計算機コア数で割った値に対するマージンをより大きくし、つまり計算コスト閾値をより小さな値として各計算コアに分担させるステージの分割を細かくし、一つのステージの計算時間を十分小さくとれば、タプルの到来時間がばらつくことによる空白時間の発生は回避可能である。タプルの到来時間のばらつきによる最短到来間隔が分かっているならば、その最短到来間隔よりも各計算コアの分担ステージの処理時間を短くすることで空白時間の発生を完全に回避できる。したがって、本実施例では、タプルが等間隔で順次到来するという限られた条件下のみでなく、到来時間がばらついても、低レイテンシ化、高スループット化の効果が得られる。

[0035] 本実施例のシステムは、ステージ分割決定部407での計算コア間のステージ分担の決定に用いる計算コスト閾値の設定に関わる係数（コスト閾値決定係数）をユーザに決定させるユーザインタフェースを有する。図7はそのユーザインタフェース画面を示す。ユーザインタフェース画面中の設定領域701には、0より大かつ1.0以下の任意の小数をコスト閾値決定係数として設定可能であり、かつ設定された値が表示される。ステージ分割決定部407は、クエリグラフの全計算コストを計算コア数で割った値に、設定されたコスト閾値決定係数を掛けて計算コスト閾値とする。この計算コスト閾値を超過しない範囲で順次オペレータを統合して、各計算機コアに分担させるステージを決定する。ユーザはストリームデータのタプル到来の状況に応じてコスト閾値決定係数を設定し、これによって各計算コアに分担させるステージ分割の細かさを設定できる。したがって、タプル到来間隔のばらつきの大小に応じたステージ分割が可能であり、実質的に待ちを回避して、各計算コアの計算時間をフルに利用することが可能となる。上記のコスト閾値決定係数を決定させるユーザインタフェースは、ストリーム処理サーバ100のコンソールユニットに実現される。もしくはホスト計算機130に上記ユーザインタフェースを設けても良い。

[0036] <変形例1>

上記した実施例では、クエリ実行状態保持領域430に格納されている、各オペレータの実行状態431～436を、クエリ実行スレッド410～413間で引き継ぐ際に、CPUコア間のキャッシュコヒーレンス制御のオーバーヘッドが、性能劣化要因となる可能性がある。一方、611のタイムチャートでは、連続するタプルの各ステージでの処理の間に、若干の猶予期間がある。そこで変形例1では、それぞれのタプルを担当する計算コアでは、実行中のステージが完了する少し前に、次のステージに属するオペレータの実行状態をその計算コア内のキャッシュにプリフェッチする。例えば、ステージBの処理に関して、計算コア0におけるタプル0についての処理B0の終了後、後続の、計算コア1におけるタプル1についての処理B1が開始する

までには、若干の時間がある。この期間を利用して、実行中のステージの処理を完了する少し前に、次のステージに属するオペレータの実行状態を、計算コアのキャッシュにプリフェッチする。例えば、計算コア 1 におけるタプル 1 についてのステージ A の処理 A 1 が終了する（即ち、処理 B 1 が開始する）直前に、ステージ B に属するオペレータ（OP 3～5）の実行状態をプリフェッチする。

[0037] この変形例 1 の構成をとることにより、上記の例で計算コア 1 による処理 B 1 を開始しようとした時点で計算コア 1 内のキャッシュ上でステージ B に属するオペレータの実行状態がダーティのまま残ることが回避される。すなわち、クエリ実行がスレッド間で引き継がれる際の CPU 待ちの時間内に行うプリフェッチにより、キャッシュコヒーレンス制御を改めて実施することが回避される。これを有効に行うためには、上記の例でステージ B に属するオペレータの実行状態のプリフェッチは、一つ前のタプル 0 を担当する計算コア 0 によるステージ B の処理 B 0 が完了している可能性が高いタイミングで行うのが良い。とくに、タプルの到来の周期が安定している場合に、次のタプルの到来の予測時刻の直前で先頭のステージ A に属するオペレータの実行状態のプリフェッチを行うことが可能であり、キャッシュコヒーレンス制御のオーバーヘッド削減の効果を確実に得ることが出来る。

[0038] <変形例 2>

ストリームデータ処理では、同一タイムスタンプで複数のタプルが到来するようなストリームを処理する場合もある。先の実施例でも問題にしていたタプル到来周期のばらつきが極端に著しい場合であるとも言える。変形例 2 では、入力ストリームの先頭に、同一タイムスタンプのタプルが連続している場合に、各クエリ実行スレッド 410～413 が、当該連続タプルを一括して処理するようにした。

[0039] この変形例 2 では、同一のタイムスタンプの複数タプルについて、同じ計算コア内で順次各ステージの処理が行われる。したがって、その計算コア内でキャッシュのヒット率が向上するので、その分だけ 1 タプル当たりの処理

時間の短縮が期待される。ただし、同一タイムスタンプのタプル数が大きく変動するストリームの場合には各スレッドに分担するタプル数が不揃いになる、つまり処理時間が不揃いになるため、処理の空白や待ちが生じてしまう。同一タイムスタンプのタプル数が比較的安定して推移する場合には、処理の空白は軽減され、上記した１タプル当たりの処理時間の短縮によりストリーム処理のスループット向上の効果が得られる。

[0040] <変形例３>

上記変形例２で生じる各クエリ実行スレッドの処理時間の不揃いの発生を軽減するため、変形例３ではより詳細な設定を可能する。図８は実施例３の同一計算機コア内の複数タプルー括処理の設定に関するユーザインタフェース画面を示す。ユーザインタフェース画面８０１の中の設定領域８０３には一括処理の候補とするタプルと判断するタイムスリット幅の設定値が設定される。領域８０４には設定領域８０３への設定のメニューが示されており、チェックマークを入力すると対応する値（図では２．０マイクロセカンド）がタイムスリット幅として設定される。入力ストリームの先頭のタプルに続き、設定されたタイムスリット幅の期間内に後続のタプルが到来したなら、これらタプルは、クエリ実行スレッド４１０～４１３のうち実行停止状態であるいずれかが一つが一括して処理するタプルの候補となる。ただし、設定領域８０５には一括処理をする最大タプル数が設定される。ここでも領域８０６に示されたメニューのなかから選択された値（図では４）が設定される。この場合には上記期間内に多数のタプルが到来しても一つのスレッドで処理されるタプルの数は４である。なおメニューのなかから「無制限」が選択されて設定された場合には、設定されたタイムスリットの期間内に到来した全てのタプルが一つのスレッドで一括処理される。またメニューのなかから１が選択されて設定された場合には、一括処理のタプル数は１であり、つまり一つのスレッドで連続する複数タプルの一括処理は行わない。

[0041] このように変形例３では連続タプルの一括処理について多様な設定が可能でありユーザが入力ストリームの状況に合わせた柔軟な指定が可能である。

このようなユーザインタフェース画面は、ストリームデータ処理サーバ 100 のコンソールか、ホスト計算機 130 のいずれかに設けられる。

[0042] 以上、本発明の実施形態について説明したが、これらは本発明を説明するための例示であり、本発明の適応範囲を例示した形態のみに限定するものではない。また、前述した実施形態のいかなる組み合わせも本発明の実施形態となり得る。

符号の説明

[0043] 100 ストリームデータ処理サーバ
90～93 CPU
103 メモリ
104 バス
105 ネットワークインタフェース
106 ストレージ
120 データ発生器
130 ホスト計算機
131 クエリ登録インタフェース
132 ストリームクエリ
140 データ受信器
121、141 タプル
150 ネットワーク
201～210 オペレータ
301～304 パイプライン処理におけるステージ
310 パイプライン処理における各ステージの処理時間の比率
601～605 本発明におけるステージ
610 本発明における各ステージの処理時間の比率
311、611 各計算コアにおける各タプルについての各ステージの処理時間
402 クエリパーサ

- 4 0 3 クエリグラフ構成情報
- 4 0 4 オペレータ実行順序決定部
- 4 0 5 オペレータ実行順情報
- 4 0 6 オペレータ処理コスト算出部
- 4 0 7 ステージ分割決定部
- 4 0 8 クエリ実行制御情報
- 4 0 9 タプル通番管理部
- 4 1 0 ~ 4 1 3 クエリ実行スレッド
- 4 2 0 担当タプル実行状態保持領域
- 4 3 0 クエリ実行状態保持領域
- 4 3 1 ~ 4 3 6 オペレータ別実行状態
- 4 3 7 ~ 4 4 1 ステージ別実行済タプル通番フラグ
- 4 5 0 タプル入力部
- 4 5 1 タプル出力部
- 5 0 1 ~ 5 0 9 クエリ実行スレッドのフローチャート

請求の範囲

[請求項1]

タイムスタンプが付与されたタプルの系列であるストリームを対象として、ユーザが定義したクエリ定義を、クエリグラフに変換し、前記クエリグラフを構成するオペレータ間の実行順序を決定するクエリパーサを備え、かつ前記クエリグラフを構成するオペレータの集合である複数のステージのいずれをも実行可能にされた複数のクエリ実行スレッドを備えたストリームデータ処理システムによるストリームデータ多重処理方法であって、

前記クエリグラフを、各々が連続する複数のオペレータの集合である前記複数のステージに分割し、

前記クエリ実行スレッドのうち、実行停止状態にある第1のスレッドが、その時点において前記ストリームの先頭に位置する、第1のタプルを抽出し、

前記ストリームにおいて、前記第1のタプルの一つ前に位置していた、第2のタプルについて、前記複数のステージのうちの、最初に実行すべき第1のステージにおける処理が完了しているか否かを確認し、前記処理の完了が確認され次第、前記第1のタプルについて、前記第1のステージの処理を実行し、前記第1のステージの処理が完了次第、前記第2のタプルについて、次に実行すべき第2のステージにおける処理が、完了しているか否かを確認し、処理の完了が確認され次第、前記第1のタプルについて前記第2のステージを実行するという順次処理を、前記第一のクエリ実行スレッドが、前記第1のタプルについて、前記複数ステージの全ての処理が完了するまで継続し、

もって前記複数のクエリ実行スレッドが各々抽出したクエリの処理を同時並行で実行することを特徴とするストリームデータ多重処理方法。

[請求項2]

請求項1に記載のストリームデータ多重処理方法であって、
前記クエリグラフを複数のステージに分割する処理は、

前記全オペレータの処理コストの合計をクエリ実行スレッド数で割った値以下の値をコスト閾値として設定し、

前記オペレータ間の実行順序に従って並べた先頭から、各ステージを構成するオペレータの処理コスト合計が、設定された前記コスト閾値を超過しないように前記クエリグラフを各ステージに分割する手順を含むことを特徴とする、ストリームデータ多重処理方法。

[請求項3] 請求項2に記載のストリームデータ多重処理方法であって、

前記コスト閾値として、前記全オペレータの処理コストの合計をクエリ実行スレッド数で割った値と、0より上かつ1、0以下であってユーザにより選択された係数との積を設定することを特徴とする、ストリームデータ多重処理方法。

[請求項4] 請求項1に記載のストリームデータ多重処理方法であって、

前記クエリグラフが、時間ウィンドウ、遅延オペレータといった、時限発火型のオペレータを含む場合に、前記時限発火型オペレータが生成する、タプルのタイムスタンプと、前記入力ストリームの先頭のタプルのタイムスタンプを比較し、より過去のタイムスタンプを持つタプルから順に処理することを特徴とする、ストリームデータ多重処理方法。

[請求項5] 請求項1に記載のストリームデータ多重処理方法であって、

前記第1のクエリ実行スレッドは、前記第1のステージの処理完了に若干先立って、

前記第2のタプルについて、前記第2のステージにおける処理が、完了しているか否かを確認し、前記第2のステージの処理の実行状態等を、キャッシュメモリにプリフェッチすることを特徴とする、ストリームデータ多重処理方法。

[請求項6] 請求項1に記載のストリームデータ多重処理方法であって、

実行停止状態にある前記第一のクエリ実行スレッドは、その時点において前記ストリームの先頭に連続して位置する、同一のタイムスタ

ンプを持つ複数のタプルを抽出し、

前記順次処理を、前記第一のクエリ実行スレッドが、前記複数のタプルについて、全てのステージの処理が完了するまで継続する、

ことを特徴とする、ストリームデータ多重処理方法。

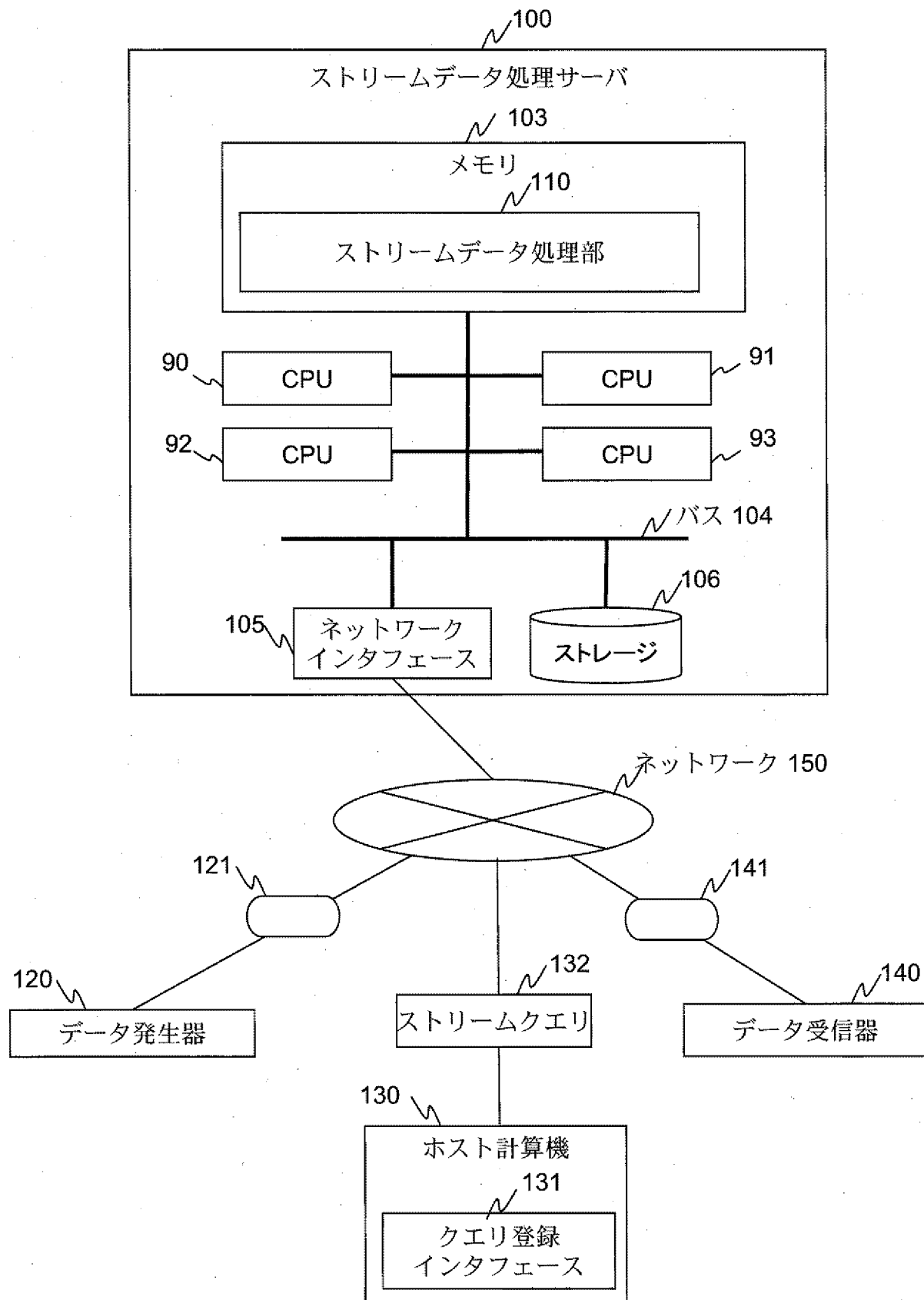
[請求項7]

請求項1に記載のストリームデータ多重処理方法であって、

実行停止状態にある前記第1のクエリ実行スレッドは、その時点において、前記ストリームの先頭に、ユーザが指定したタイムスタンプ範囲内で連続して位置する、複数のタプルを抽出し、該第1のクエリ実行スレッドが、前記複数のタプルについて、全てのステージの処理が完了するまで前記順次処理を継続することを特徴とする、ストリームデータ多重処理方法。

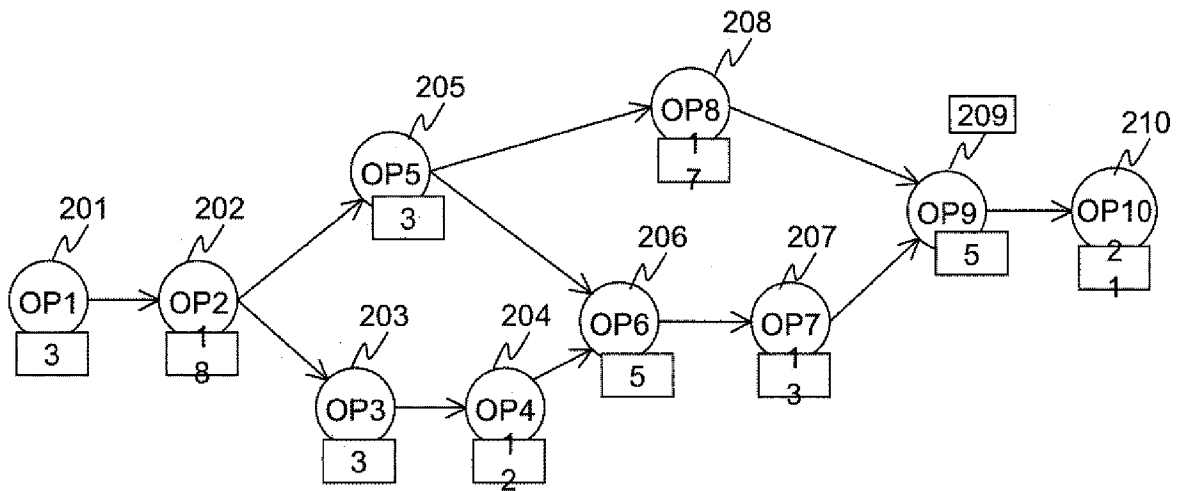
[図1]

図1

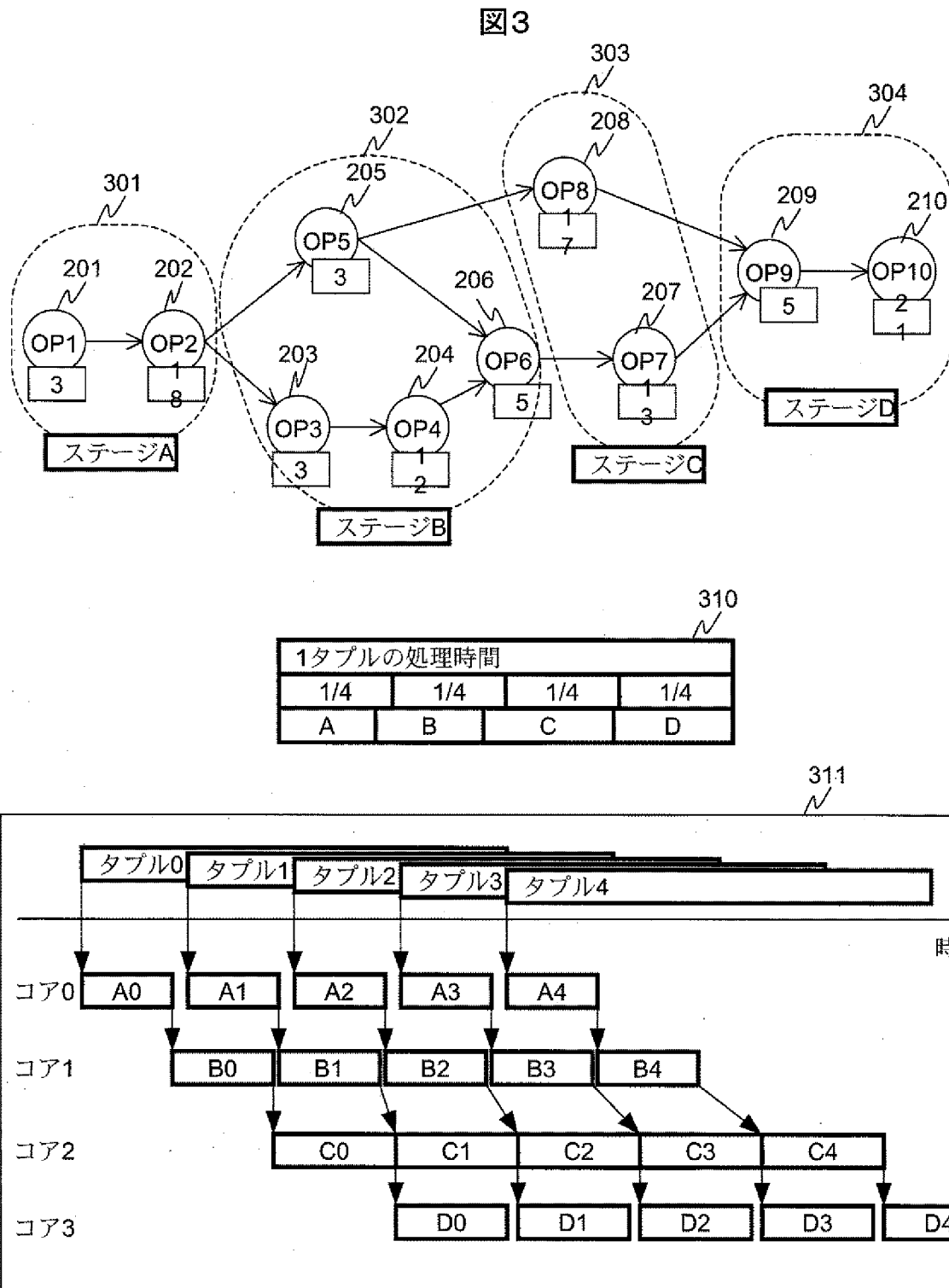


[図2]

図2

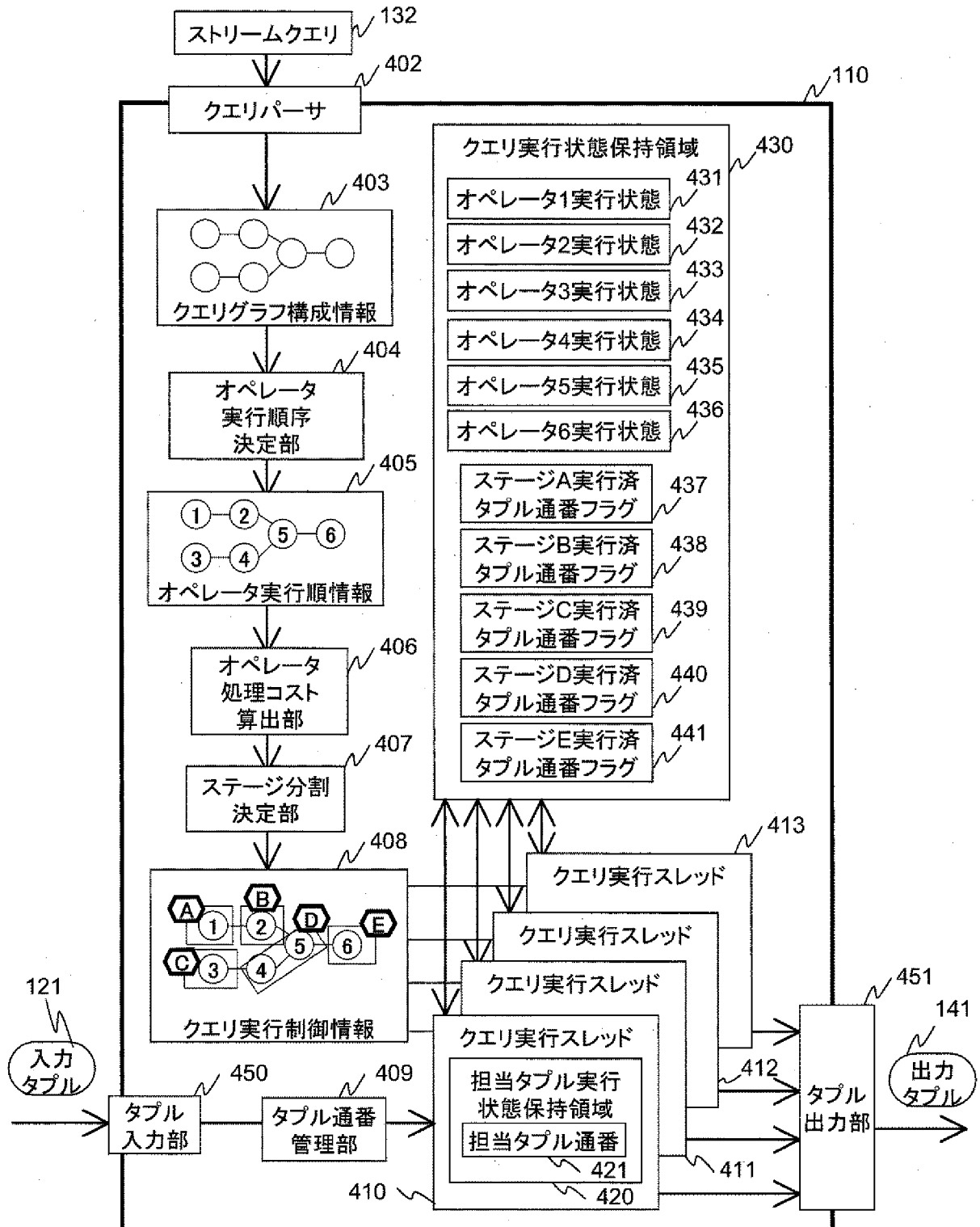


[図3]



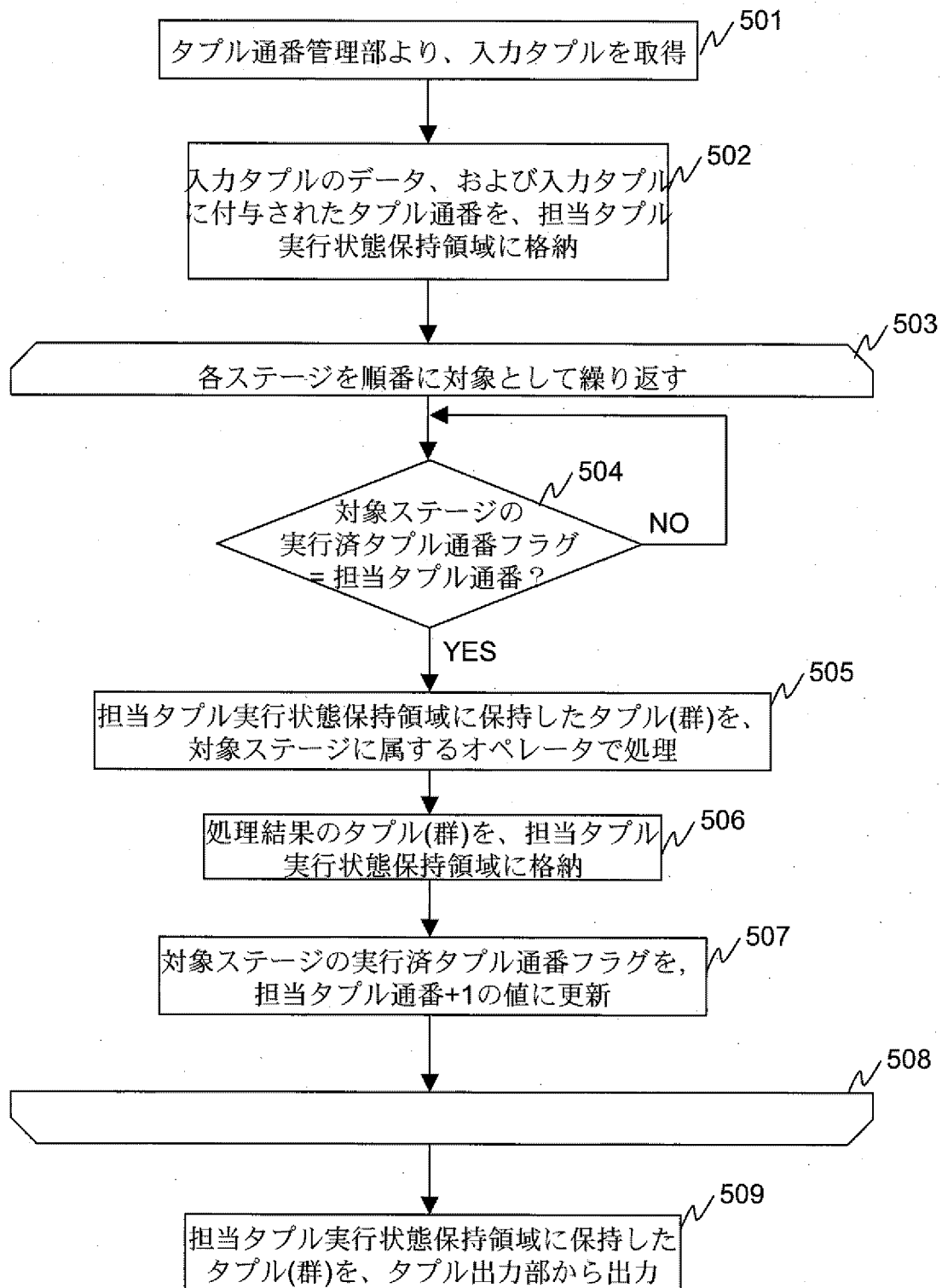
[図4]

図4



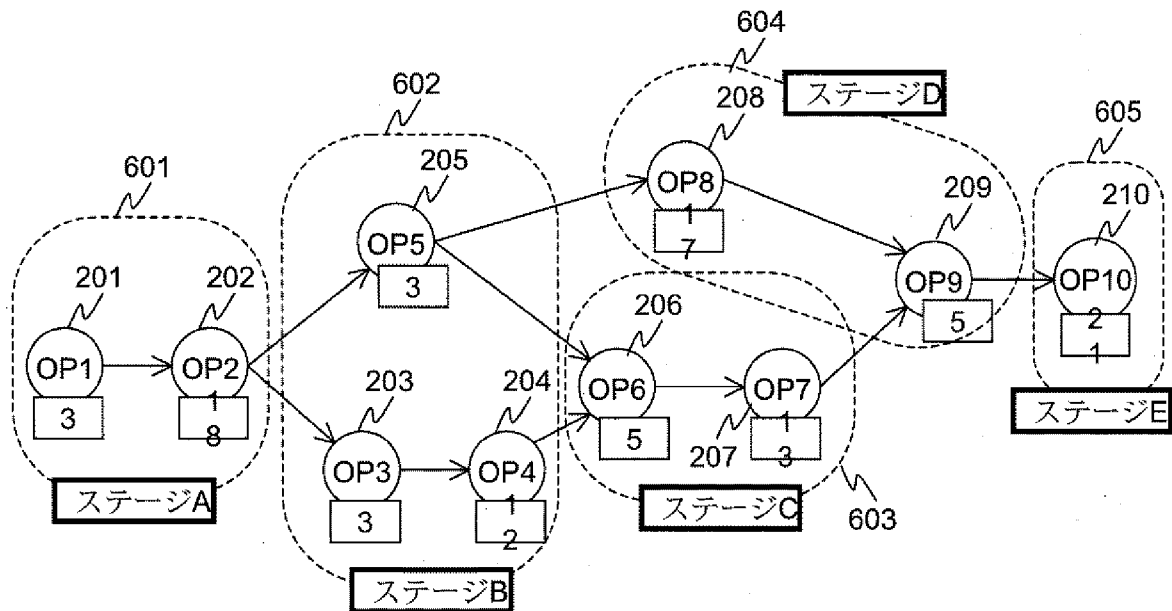
[図5]

図5



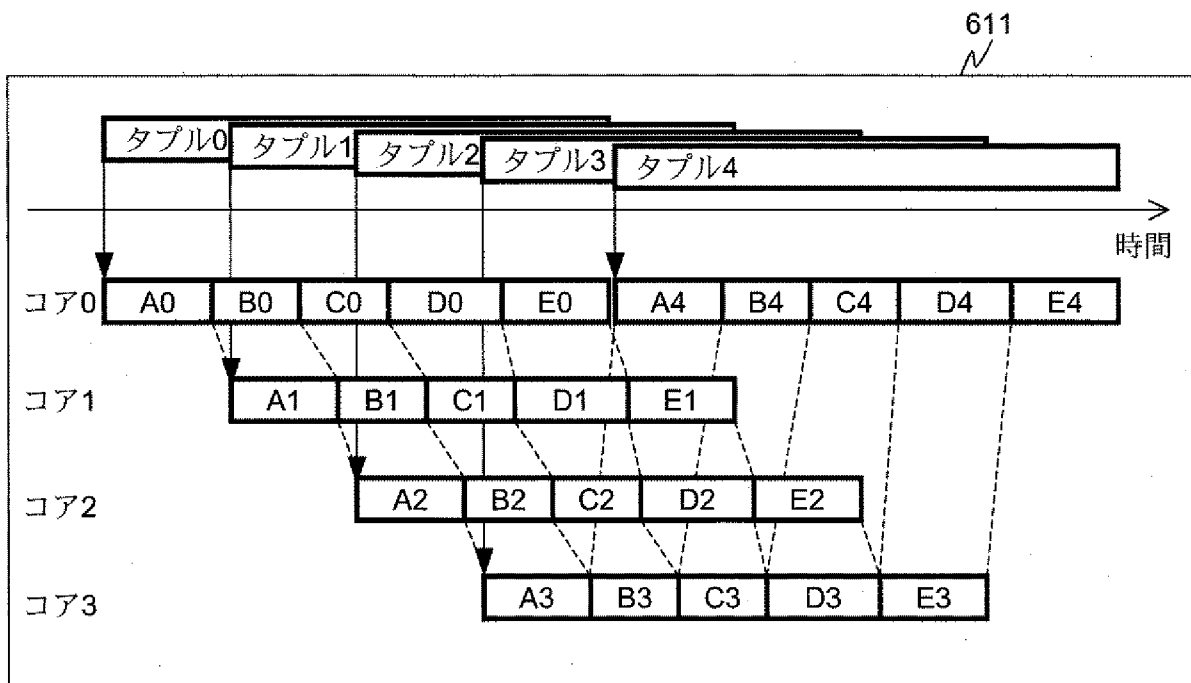
[図6]

図6



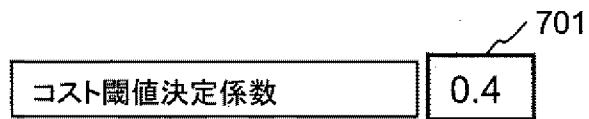
1タプルの処理時間

1/4	1/4	1/4	1/4
A	B	C	D



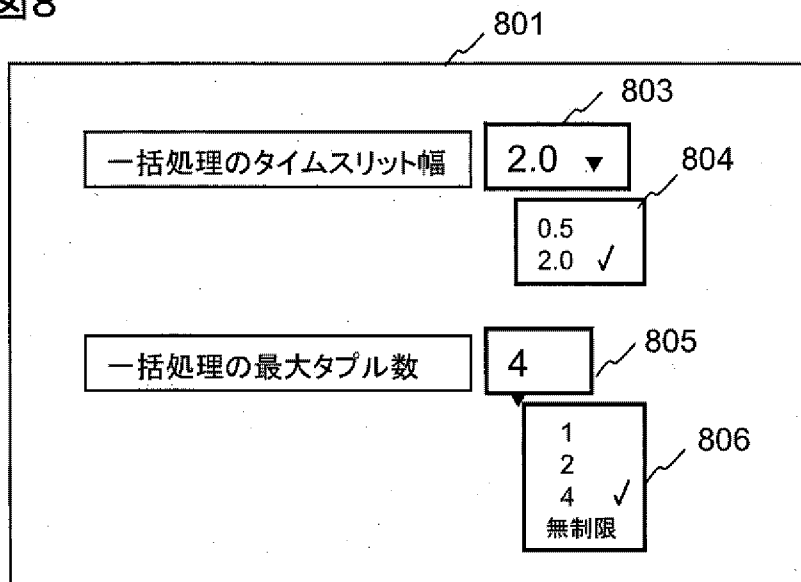
[図7]

図7



[図8]

図8



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2012/073568

A. CLASSIFICATION OF SUBJECT MATTER

G06F12/00 (2006.01) i, G06F17/30 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F12/00, G06F17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2012
Kokai Jitsuyo Shinan Koho	1971-2012	Toroku Jitsuyo Shinan Koho	1994-2012

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2012/046316 A1 (Hitachi, Ltd.), 12 April 2012 (12.04.2012), claim 1 (Family: none)	1-7
A	JP 2011-034255 A (Hitachi, Ltd.), 17 February 2011 (17.02.2011), claim 1 & US 2011/0029554 A1	1-7
A	JP 2010-204880 A (Hitachi, Ltd.), 16 September 2010 (16.09.2010), claim 1 & US 2010/0229178 A1	1-7

☒ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search
24 October, 2012 (24.10.12)

Date of mailing of the international search report
06 November, 2012 (06.11.12)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2012/073568

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2010-108152 A (Hitachi, Ltd.), 13 May 2010 (13.05.2010), claim 1 & US 2010/0106946 A1	1-7
A	JP 2008-146503 A (Sony Computer Entertainment Inc.), 26 June 2008 (26.06.2008), claim 1 (Family: none)	1-7

A. 発明の属する分野の分類（国際特許分類（I P C））
Int.Cl. G06F12/00(2006.01)i, G06F17/30(2006.01)i

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（I P C））

Int.Cl. G06F12/00, G06F17/30

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1922-1996年
日本国公開実用新案公報	1971-2012年
日本国実用新案登録公報	1996-2012年
日本国登録実用新案公報	1994-2012年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	WO 2012/046316 A1（株式会社日立製作所）2012.04.12, 請求項 1 （ファミリーなし）	1-7
A	JP 2011-034255 A（株式会社日立製作所）2011.02.17, 請求項 1 & US 2011/0029554 A1	1-7
A	JP 2010-204880 A（株式会社日立製作所）2010.09.16, 請求項 1 & US 2010/0229178 A1	1-7

☒ C 欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー	の日の後に公表された文献
「A」特に関連のある文献ではなく、一般的技術水準を示すもの	「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの	「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）	「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「O」口頭による開示、使用、展示等に言及する文献	「&」同一パテントファミリー文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願	

国際調査を完了した日

24.10.2012

国際調査報告の発送日

06.11.2012

国際調査機関の名称及びあて先
日本国特許庁（I S A/J P）
郵便番号100-8915
東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）

田川 泰宏

電話番号 03-3581-1101 内線 3565

5 U

4 2 3 6

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	JP 2010-108152 A (株式会社日立製作所) 2010.05.13, 請求項 1 & US 2010/0106946 A1	1-7
A	JP 2008-146503 A (株式会社ソニー・コンピュータエンタテインメント) 2008.06.26, 請求項 1 (ファミリーなし)	1-7